CS12020

Robotics Assignment Report

Ahmed Zitoun
ahz1@aber.ac.uk

Robotics Assignment Report - CS12020

Ahmed Zitoun — ahz1@aber.ac.uk

Contents

1 Introduction - Stage 1

This pdf document will contain the report of the assignment (stage 4), There will be a description what I accomplished in the each stage of the assignment, what was not accomplished and what can be improved.

2 Functionality Requirements - Stage 2

2.1 Let's Dance - Stage 1

I implemented the use of the Left Button to execute the program.

- Task 1 (EEPROM Reading) - I was able to accomplish it by using the EEPROM code given in the assignment. This allowed my robot to read out the EEPROM readings at the beginning of the program execution. I had a difficult time with this part because I came back to it after completed all other tasks.
- Task 2 & 3 (Straight-line & Static 90 degree turn.) - I was able to maintain a steady movement across a 1 metre course, and at the one of the course the robot would turn 90 degrees in both orientations. I was not able to configure the movement offsets correctly because of the fluctuation of the servos.
- Task 4 (Dance) - This was the easiest part to do because it was completely up to me to choose the dance sequence, and after already completing the previous tasks I had what I needed to make it work without any further complex coding. The LEDs worked in a random sequence with varing delay times and the robot has sufficient time to complete every move of the dance.

2.2 Follow a Path - Stage 2

- Task 1 (Calibration) - The calibration of the robot was tricky at first but in the end was done efficiently in my opinion because of the use of array lists and for loops to decrease the lines of code, while also using less bytes of storage. I was able to reuse the code for the calibration to make my movement smoother.

- Task 2 (Follow Along Path) - Due to the LDR sensor algorithm written in the calibration task I was able to use it in this task to improve the accuracy of the robots movements making it travel smooth and self-align. The movement algorithm I coded with the use of if & else statements and cross function linking, helped the robot move in a steady path because of the checking of the LDR sensors with little delays to allow consistent smooth movement.

  Following the previous task, The robot was able to move along the line for 50cm with the use of the move-forward function however on it had difficulty moving backwards because of the how the sensors were placed on the robot development board, I take part of the blame for that because of movement was originally configured for forward movements, but I implemented it in the backwards, this made it lose its course at curved points on the line. When the robot had moved 50cm the green LED would flash for 2 seconds.

- Task 3 (Re-acquisition of the path) - This task was completed, as the robot was calibrated I tested this task by placing the robot off of the course, the would then find its path by moving forward for 5 seconds and if a course was detected in the time period it would align itself on the course and continue its path movement.

2.3 Detecting Obstacles - Stage 3

- Task - This task was implemented in the Stage 2 (see 2.2).

  Here I use the Infrared Sensor (IR transmitter & receiver) which are used to detect any oncoming obstacles, Using if & else statements in the function which checked if the IR receiver is detecting any objects, a text will be outputted and the robot will stop its movement for 1 seconds, turn 180 degrees clockwise. following the turn, the robot will continue its path in the opposite direction while still remaining on the path.

  The implementation of the code in the robot was the most difficult because of the sensitivity of the sensors. Also because an error which kept occurring, that kept restarting the loop, however after adjusting re-coding the entire function it worked fine except for the offsets that I spoke about previously (see 2.1)

2.4 Counting Junctions - Stage 4

- Task - This task was implemented in the Stage 2 (see 2.2).

  By tweaking the 2.2 function I was able to add the junction detection. This was done by checking if all three LDR sensor readings were within the threshold limit, then the robot will continue its forward movement along the path, However with this method the robot would jitter when reaching a junction because it would detect either one of the side (left or right) LDR before the other, making it turn in that direction.

  When it came to the junction counting, I used a separate function to allow for defensive programming. It still uses the same if statement to check for LDR readings, but if a junction is found the robot will flash the green LED, stop moving and increment a counter in a period of 0.2 seconds. If the counter reaches 3 junctions counts it will stop the robot and make it turn 180 degrees clockwise in a period of 0.8 seconds. The robot will then continue along the path in the opposite direction.

3 Conclusion

- Let's Dance - Stage 1 — I Believe I should get *(9-11)/15* marks for this stage, due to the fact I completed the whole stage and it works fine except for the offsets which were caused by the servo fluctuation.

- Follow a Path - Stage 2 — For this stage I think I should get *(16-20)/25* marks, because of my use of the efficient coding which allowed me for compress my line use (and I had to rewrite this code because the file got corrupted the first time). Furthermore my robot's movements were extremely smooth.

- Detecting Obstacles - Stage 3 — This stage should be awarded *(6-8)/15* marks because this was the stage I struggled the most at and it didn't work up to my expectations.
- Counting Junctions - Stage 4 — For this stage I believe *(14-17)/20* marks is appropriate because my code is functional with a slight error involving the robot jitter because of the LDR sensitivity.
- Report — For this report I think I should get *(18)/25* marks because I took the effort of writing it in latex and I fully described the functionality of the code and any requirements needed.

4 Proof of Readings

Figure 1: (Screenshot proof of EEPROM — LDR READINGS — ROBOT MOVEMENTS)

```
12:04:27.148 -> Storing value 85 to address 0 to save the left servo stopping point
12:04:27.195 -> Storing value 78 to address 1 to save the right servo stopping point
12:04:27.249 -> Storing value 19 to address 2 to save the left servo difference from LEFT_STOP to drive straight (when the right value is also set on the right servo
12:04:27.450 -> Storing value 20 to address 3 to save the right servo difference from RIGHT_STOP to drive straight (when the left value is also set on the left servo
12:04:27.596 -> Storing value 43 to address 4 to save the wheel diameter
12:04:27.650 -> Storing value 0 to address 10 to save the threshold to separate light and dark values for the left LDR (A2)
12:04:27.766 -> Storing value 0 to address 13 to save the threshold to separate light and dark values for the middle LDR (A1)
12:04:27.897 -> Storing value 0 to address 16 to save the threshold to separate light and dark values for the right LDR (A0)
12:04:27.997 -> Servo Attached
12:04:40.446 -> Light Sensor Calibrated
12:04:40.446 -> 181
12:04:40.446 -> These are limits
12:04:40.500 -> 144
12:04:40.500 -> These are limits
12:04:40.500 -> 238
12:04:40.500 -> These are limits
12:04:46.740 -> Dark Sensor Calibrated
12:04:46.787 -> 56
12:04:46.787 -> These are limits
12:04:46.787 -> 52
12:04:46.787 -> These are limits
12:04:46.840 -> 89
12:04:46.840 -> These are limits
12:04:46.840 -> | Threshold Limit Start |
12:04:46.887 -> 118
12:04:46.887 -> 98
12:04:46.887 -> 163
12:04:46.887 -> | Threshold Limit End |
```

```
12:09:17.537 -> Moving Forward
12:09:17.591 -> Moving Right
12:09:19.505 -> Moving Forward
12:09:19.505 -> Moving Right
12:09:19.958 -> Adding to Counter
12:09:19.958 -> 1
12:09:20.764 -> Junction Found
12:09:20.764 -> Moving Forward
12:09:20.764 -> Moving Right
12:09:20.811 -> Moving Left
12:09:20.970 -> Moving Forward
12:09:21.170 -> Moving Forward
12:09:21.372 -> Moving Forward
12:09:21.620 -> Moving Forward
12:09:21.821 -> Moving Forward
12:09:22.022 -> Moving Forward
12:09:22.259 -> Moving Forward
12:09:22.478 -> Moving Forward
12:09:22.679 -> Moving Forward
12:09:23.382 -> Moving Forward
12:09:26.147 -> Moving Forward
12:09:26.201 -> Moving Right
12:09:26.402 -> Moving Forward
12:09:26.402 -> Moving Right
12:09:26.602 -> Moving Forward
12:09:26.602 -> Moving Right
12:09:26.804 -> Moving Forward
12:09:29.671 -> Moving Forward
12:09:29.671 -> Moving Right
12:09:29.872 -> Moving Forward
12:09:30.073 -> Moving Forward
12:09:30.328 -> Moving Forward
12:09:30.529 -> Moving Forward
12:09:30.730 -> Moving Forward
12:09:30.950 -> Moving Forward
12:09:31.151 -> Moving Forward
12:09:31.198 -> Moving Left
12:09:31.400 -> Moving Forward
12:09:31.601 -> Moving Forward
12:09:31.802 -> Moving Forward
12:09:32.057 -> Moving Forward
12:09:34.573 -> Moving Forward
12:09:34.573 -> Moving Right
12:09:34.775 -> Moving Forward
12:09:34.822 -> Moving Right
12:09:35.023 -> Moving Forward
12:09:35.023 -> Moving Right
12:09:35.441 -> Adding to Counter
12:09:35.441 -> 2
12:09:40.253 -> Junction Found
12:09:40.307 -> Moving Forward
12:09:40.307 -> Moving Right
12:09:40.307 -> Moving Left
12:09:41.291 -> Moving Forward
12:09:46.021 -> Moving Forward
12:09:46.021 -> Moving Right
12:09:46.222 -> Moving Forward
```