CS12020 Robot Assignment Report

Author: Jakub Jozef Grzebinoga (jag88)

Introduction

During development of this assignment I managed to complete two out of four functional requirements. These were; following a black line on a white background and navigating around an obstacle to rejoin the path behind it.

Challenges

The main challenges that I encountered when writing the code for this assignment were; lighting changes between the practical room and home, the left button being wired to the same pin as the infra-red receiver and the batteries changing voltages. I also encountered less critical issues such as the light dependent resistors (LDRs) bending and the front skids catching on the test surface.

I tried different methods of overcoming these challenges, some of which were straightforward. The issues created by the bad batteries giving lower voltages (thus making the servos, infra-red LED and sensors work incorrectly) was overcome by buying new batteries.
The left button and infra-red receiver being wired to the same pin on the arduino board proved a more difficult challenge however as, at first, the code detected a low signal from that pin when the button was pressed down. After a lot of trial and error, I resolved the issue by adding a delay into the code before the infra-red receiver read function which seemed to fix the issue. A remaining issue stemming from this still remained after this however, meaning that - due to the buttons internal contacts being slightly loose -, the code would read a false positive when reading the output from the target pin. I tried to overcome this issue with delays, however this introduced unnecessary pauses in the main loop which were noticeable in the robots operation.

I finally resolved this issue completely by getting rid of most of the delays I introduced when trying to resolve this the first time around, instead using a method that increments a counter every time the receiver ‘sees’ an

obstacle, and resets that counter to zero if the receiver cannot ‘see’ anything in front of it. I then use the value of this variable in other functions and only take action to avoid the object if the counter is higher or equal to a value of three. This eliminates the impact of the false positive reading.
Unfortunately I could not find a solution that resolved all of the issues created by the varying light intensity I encountered whenever I changed testing environments. I ended up increasing the range from the LDR readings within which the robot moved and bending the LDRs slightly to adjust their position. I found that if they were positioned to face more towards the centre of the robot, I got a more reliable reading as the light intensity underneath the robot was more constant due to the shadow it cast. I also found that positioning the two side mounted LDRs towards the centre one, made the robot follow the path more accurately.

Testing

Based on the testing I performed during development, my robot is expected to fail when the lighting is too bright or too dim, if the obstacle is bigger than the container provided with the robot itself and when a barcode giving instructions is put on the path.
This is due to the fact that my changing testing environments caused my LDR readings to vary drastically, making marking the low and high readings (to mark the width of the barcode lines and distance between them) much more difficult than expected.
As I was unable to figure out a reliable algorithm for reading the barcodes, I decided to; remove the code I had for it from the final arduino sketch and make the robot ignore the lines if both side LDRs are reading a lower value (i.e are over a black line). This allowed the robot to continue on the rest of the path until it reached the end.

The changing test environments also meant that, depending on the environment, the robot would not reliably follow the path it was placed on. This is due to the ranges of the LDR readings being fairly narrow compared to the whole range that they are able to read. This constriction had to be put into the code as, without it, the robot would not follow any path under any conditions reliably as the likelihood of the reading being within the ‘working range’ would be much higher than it is currently.
The robot is also expected to fail, should the obstacle put in its path is larger than the container than it came with it. This is due to the fact that this is the only object that I had that was a suitable obstacle for the robot to avoid and, since I hardcoded the path it should take to go around the obstacle, if the obstacle is much bigger than the container, the robot will end up colliding with it.

The course the robot was tested on most frequently, featured a black coloured path on a white background, a gray path on a white background, a black path with horizontal black stripes on a white background and 24x17cm box. All paths were 4cm wide, placed in the middle of a 21cm wide piece of white paper. The course was factored into the design of my code in two main ways. The first being the amount of light reflected off the different colours printed on the paper the robot drove over; the code only takes action over the servo motors if the LDRs read between certain ranges (which is affected by the intensity of light in the test environment). The second being the size of the obstacle as it cannot be too low - as this will mean the infra-red receiver will not ‘see’ the object -, nor can it be too wide or too long as this will interfere with the robots pre determined avoidance path. This works off time delays which determine the length of time the servos are powered for, thus determining the angle of the robot's turn, and how far forward it drives.

Test Table and Results

Test	Test Predicted outcome	Test Actual outcome	Result	Change
1	Left button changes "servoStops[0]"	"servoStops[0]" incremented by one after each button press, value printed shows this	Success	
2	Right button changes "servoStops[1]"	"servoStops[1]" incremented by one after each button press, value printed shows this	Success	
3	Left button changes left servo speed	Left button changed right servo speed	Fail	
4	Left button changes left servo speed	Left button changed left servo speed	Success	Changed value written to servo from the value of "servoSpeed[1]" to "servoSpeed[0]"
5	Right button changes right servo speed	Right button changes right servo speed	Success	
6	Right button lights up green and red led	Right button lights up correct leds	Success	
7	Left button lights up yellow led	Left button lights up correct led	Success	
8	Left and right button quit calibration function	Left and right button increment "servoStops[0]" or "servoStops[1]" randomly	Fail	
9	Left and right button quit calibration function	Left and right button increment "servoStops[0]" or "servoStops[1]" randomly	Fail	Changed if loop syntax
10	Left and right button quit calibration function	Left and right button increment "servoStops[0]" or "servoStops[1]" randomly	Fail	Changed if loop order
11	Left and right button quit calibration function	Left and right button increment "servoStops[0]" or "servoStops[1]" randomly	Fail	Added delay into if loop
12	Left and right button quit calibration function	Left and right button increment "servoStops[0]" or "servoStops[1]" randomly	Success	Removed delay, added separate if statement with a break
13	"motorSpeedsBoth" function drives robot forward	function turns robot in a circle	Fail	
14	"motorSpeedsBoth" function drives robot forward	Function drives robot forward	Success	Took 50 away from right servo stop value, added 50 to left servo stop value
15	"motorSpeedsRight" function drives robot right	Function drives robot right	Success	
16	"motorSpeedsLeft" function drives robot left	Function drives robot left	Success	
17	"motorSpeedsStop" function stops robot	Function stops robot	Success	
18	"motorSpeedsRightBk" functions turns robot left and backwards	Function drives robot left and backwards	Success	

Test	Test Predicted outcome	Test Actual outcome	Result	Change
20	Robot detects object in front of it using ir receiver	IR blaster sends pulse, ir receiver reads ir light bouncing back	Success	
21	Enters "obstacleAvoidance()" function when object detected 3x in a row	Robot enters function as expected	Success	
22	Robot executes "irRead()" function more often when in "obstacleAvoidance()"	Robot executes function as expected	Success	
23	Robot executes obstacle avoidance sequence when object detected 3x	Robot executes sequence as expected	Success	
24	Robot turns right, goes forward, turns left, goes forward, turns left, then goes forward until it detects a line	Robot executes sequence as expected	Success	
25	Sequence has correct delays (takes robot around object)	Robot crashes into object, doesn't turn enough on first turn	Fail	
26	Sequence has correct delays (takes robot around object)	Robot crashes into object, doesn't go forward far enough after first turn	Fail	Increased time right servo spins backwards for to make robot turn 90 degrees
27	Sequence has correct delays (takes robot around object)	Robot crashes into object, doesn't turn enough on second turn	Fail	Increased time robot drives forward for after second turn
28	Sequence has correct delays (takes robot around object)	Robot crashes into object, doesn't go forward far enough after second turn	Fail	Increased time right servo spins backwards for to make robot turn 90 degrees
29	Sequence has correct delays (takes robot around object)	Goes around object	Success	
30	Robot rejoins path after avoiding object	Robot rejoins path	Success	
31	Robot carries on path after going over barcode	Robot comes off the path in a random direction everytime it encounters a barcode	Fail	
32	Robot carries on path after going over barcode	Robot carries on path over barcode	Success	Changed algorithm to not change anything if both side ldr's are below a certain value (i.e over a black line)
33	Robot carries on gray path	Robot stops when it encounters gray path	Fail	
34	Robot carries on gray path	Robot carries on gray path as expected	Success	Adjusted ldr working range