Ben Weatherley

Robotic Assignment Writeup

Stage 1: Reading and obeying bar-codes

I decided to time how long it took my robot to cross each bar in the barcode, by taking a millisecond reading when it hit the first black bar and then subtracting this reading from another millisecond reading at the end of the bar, repeating this for every bar in the barcode. I chose this way as when the program comes to compare the scanned barcode to the stored barcode data, it would work regardless of the light level of the room you were in, if the calibration was done correctly. I decided the best way to figure out when one bar began and ended would be to decide on a cut-off point – a value that when exceeded meant the robot was on a white surface, and when below meant the robot was on a black surface.

To calculate this value, at start up 100 values are taken from the LDRs on a white surface and an average is calculated, then the same is done for a black surface, the two values are added together and divided by 2, to give the cut-off value. Below is some real data to show how this helps when reading a bar code.

Cut-off LDR 543

430, 424, 419, 415, 412, 408, 405, 405, 403, 402, 402, 403, 403, 404, 407, 408, 410, 414, 417, 420, 424, 429, 432, 436, 441, 445, 448, 453, 458, 461, 466, 472, 476, 480, 485, 491, 495, 500, 507, 512, 516, 521, 527, 531, 536, 542, 546, Bar 1 read in 1450

572, 575, 575, 577, 579, 579, 579, 579, 579, 577, 577, 576, 574, 571, 570, 568, 564, 561, 559, 555, 551, 548, 544, 538, Bar 2 read in 826

511, 508, 506, 505, 504, 503, 503, 505, 505, 506, 509, 512, 513, 516, 521, 523, 527, 531, 535, 538, 542, 547, Bar 3 read in 772

Above is a sample of the LDR values while the robot drives across a barcode (in this case stop). Each of the values displayed here is an average of 8 readings, which I did to reduce the amount of outlying values. You can see how even when moving across a bar which is all the same colour the results fluctuate quite a lot, but do steadily increase as the robot gets closer to the end of the bar. You may notice it takes one reading that is above the average before it then outputs the bar as being read, this is because to trigger the start of the next bar, two consecutive values must be above the average reading, again to reduce the impact of outlying values.

Though I think this was a good approach to reading the barcodes, I think I came into this assignment with the idea of a perfect world, where a black surface would always return a low value, and a white surface would always return a high value. I also experience a few hardware issues, such as the LDRs on my robot were faulty and did not return the correct values, causing my Arduino to be replaced, but leaving me with little time to adjust and fix my program. The result is that my barcode reader is not quite perfected enough to read an entire barcode every time – it successfully scans a barcode about one in 15 times, which did not leave me with enough time to gather base values for each of the barcodes, and leaves me with no other option than to consider it a failure – though I do think with a little more time and error correction I could get it to work almost all the time.

Stage 2: Obstacle avoidance

I encountered a very odd problem with my IR transmitter and receiver, that being that for the first 300ms after starting the tone, the receiver would always detect an obstacle, even if I covered it with a finger, the only way I found to fix this was my function obstacleSetup, which sends a IR pulse for 300ms (even though I know it automatically turns of after 100ms) and then turns of the tone. This causes my normal obstacle function to run perfectly, and I am very pleased with how it turned out. If using a different robot to mine I would recommend attempting to comment out the obstacleSetup function and seeing if the code runs normally, as I suspect the cause of this problem was a hardware fault.

Stage 3: Let's dance

To time the infra-red pulse that is received I decided to read pin 2, and if it was low (meaning an IR pulse was detected) I proceeded to wait for the pulse to end (as I could have started reading the pulse half way through, so timing it would be pointless), then started to time how long went by with no pulse. If the result was anywhere between 75-85ms, then I began to time the length of the next IR pulse, if this was also between 75-85ms then the robot proceeds to boogie. While all of that is happening, I had a few lines of code to make sure the program was not waiting for more than 100ms, otherwise it could have become trapped waiting forever for an IR pulse that would never come.

To time the period for which my robot dances I simply set a variable endTime to equal milli() (the current ms time the program has been running for) plus 20000, which is 20 seconds in milliseconds, then had the dance routine run while the endTime was greater than the current milli reading.

To make my LEDs flash in a sequence I had to consult the blink without delay page on the Arduino website. Through this I learned that with careful use of milli I could give the illusion of doing two things at once. To do this I made a function that blinks the LED whenever it detects a set time interval has passed. To do this I created a static value called prevMillis, which keeps its value between function calls, and is only initialised once, so it doesn't keep resetting to 0 every time the function is called. I then have an if statement, which checks if the currentMilli value minus the prevMilli value is greater than or equal to the interval, and if it is the previousMilli value becomes equal to the currentMilli value, and the LED state becomes equal to the opposite state of its read value (turn it off if its on, or on if its off). I then chose to reduce the delay value by 1, making the LED blink slightly faster each time. I then called this function whenever any dance moves take place, meaning it is constantly called throughout the dance, causing the LEDs to flash in a sequence, giving the illusion that the Arduino is doing two things at once.

Grading my own work

I believe that for the most part, my code is well written, commented, and uses sensible algorithms. Though I do think I could have created a few more functions, and that maybe in my attempt to comment well, some sections have been over commented. I feel that I completed a lot of the more complicated tasks, such as blinking the LEDs, reading when its time to party. As well as this I gave my robot incredible dance moves.

However, as the read barcode ultimately does not work (even though I believe the logic and coding behind it is well thought out, and I probably would have completed it were it not for faulty hardware), and is arguably the most important part of the entire code, I must class my assignment as a failure, as it does not do what it set out to do. Had the read barcode function worked, I think my assignment would be worth somewhere between 60-70%, as it is, I think it is worth somewhere between 50 and 60%.