Robotics Project

Functional Requirements

Stage 1: The first aim of this project was to program the robot to drive over and read four different barcodes (a series of four thick or thin lines), this was done by reading light values from three LDRs.

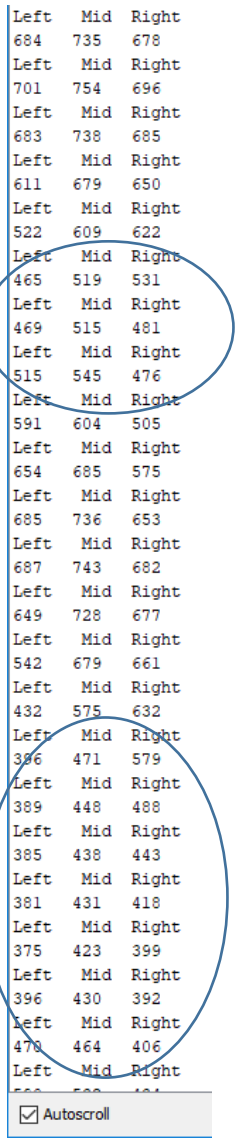
Stage 2: The second aim was to program the robot to avoid obstacles using an infrared transmitter LED and receiver. When an obstacle is detected the robot should perform a series of actions to avoid the obstacle. The robot should look for obstacles while also looking to read barcodes.

Stage 3: The third aim was to recognise a fifth barcode and if it is read, the robot should dance and flash its LEDs for 20 seconds.

How I met these aims

Stage 1: Reading Barcodes (COMPLETED – works well overall**)**

My robot reads barcodes by sensing changes in the light levels – the readings from the LDRs for a typical run are displayed on the right. The circled data shows where the black stripes are; causing the light levels to decrease below the average dark reading and boundary (for example: the middle LDR dark boundary is 580 on average) set by the LDR calibration routine. The first set circled is a thin stripe and therefore takes less time to go over than the second set which corresponds to a thick stripe.



When the robot detects the first stripe, the program enters the barcodeDetection() function which starts a timer and calculates the time width of each bar by stopping the timer each time a white stripe is reached.

```
TIMEOUT
Bar 1: 425 s
Bar 2: 965 b
Bar 3: 435 s
Bar 4: 985 b
s b s b
r
```

If all four bars are sensed, the function will return 1, but if after seven seconds the barcode is still incomplete, it will timeout and return 0 from the function. This is because the robot would have gone pass the bar by then and missed one of the bars. The timeout prevents it from reading a future bar incorrectly. The data to the left is an example of when the robot has timed-out and gone on to sense another barcode. It also shows how the program sequences the barcode.

If the barcode is completed, the program will continue to work out the sequence of thick and thin stripes from the widths using the function barcodeRead(). The program will then call the getType() function to check the sequence against pre-programed arrays to get the type of barcode that it just went over. It will then call a different function depending on what the barcode was asking it to do (LEFT, RIGHT, UTURN, STOP or BOOGIE). The program then resets all the values and starts moving again to find the next barcode.

Issues: At the beginning the robot did not read the bars successfully, this was because I had programmed it to drive over them too fast and it wasn't able to determine the transitions quick enough. Conversely, when I set the speed too low, the robot started veering off even though its should've been going straight.

The robot has some problems reading the second barcode once it has turned. I think this is because the light levels change due to its own shadow - this affects the LDR readings processed by the robot. I tried to minimise this by increasing the bound for the LDR readings so that the robot can sense 'darker white'.

Stage 2: Detecting Obstacles (COMPLETED – works well**)**

Detecting objects was done using the function to the right. It uses the inbuilt tone() function to send a specific frequency of infrared from the transmitter. The program then checks if the receiver has picked up an incoming signal. It is implemented in the main loop after the robot looks for the first bar.

```
/* IR FUNCTIONS */

int detect() {
  tone(IRLED, IRfreq);
  delay(5);
  if (digitalRead(IRREC) == LOW) {
    return 1; // Obstacle detected
  } else {
    return 0;
  }
  noTone(IRLED);
}
```

Issues: When I first created this function it didn't sense any obstacles – I realised this was because I had set the frequency to 380000 rather than 38000. Even when I corrected this, it still didn't work properly as it was constantly sensing objects. I fixed this by rearranging the function so it looked like this.

The angles turned and distances moved to avoid the obstacle may not be totally accurate due to the battery problem mention later in the next section.

Stage 3: Dancing (COMPLETED – works fairly well**)**

My robot successfully reads the new barcode, dances and flashes its LEDs for twenty seconds. I have used a while loop that compares the current time to a timer to make sure that the dance doesn't take longer than twenty seconds.

This stage was fairly easy to implement as it only required adding one more type of barcode to my array of types and sequences and creating a function to make the robot do a few moves.

Issues: The robot doesn't always end up facing the right direction as the robot has been calibrated to turn for a certain number of milliseconds for a full turn with fully charged batteries. However, when the batteries are slightly drain, it takes longer to turn as the servos receive slightly less power. This effect gets larger the more discharged the batteries are. This can also affect the length of the dance, so it does not always dance for twenty seconds.

Calibration

The robot is calibrated using the two push buttons on the top of the shield. The calibration routine is as follows:

* Right Servo, then Left Servo – using the buttons to find the value when the servos are stopped, then press both buttons simultaneously to store the stop value.

* LDR Light and Dark values – place the LDRs over a thin white stripe and press the left button, then place them over a thin dark stripe and press the right button.

```
90 r
91 b
91 r
92 l
91 l
90 l
89 l
88 l
87 b
LDR DARK VALUES: 456 532 489
LDR LIGHT VALUES: 702 762 702
```

Example calibration values

The routine is built into the program, such that it can be accessed by pressing the right button (on pin 2) when the robot is turned on (if the left one is pressed, pre-set values are used).

Conclusion

In conclusion, I think that I have fulfilled all the functional requirements to a level that demonstrates a good understanding of the problem and the technologies used. I am therefore grading the work in the 60%-69% range.