Robotics Assignment: Report

Stage 1: Reading and obeying bar-codes

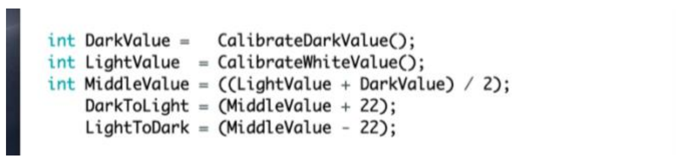
To tackle the whole problem, I decided to make a programme for each stage and then combine them together at the end. For the first stage I had to think about how the robot would record the lengths of a black bar, as well as when to start recording once a black bar had been found. The main issue I had with this stage was to change the state of the robot from looking for a barcode, to recording the lengths of black bars once the first black bar had been recognised. I first tried using different combinations of while loops and if statements to try to differentiate between the two states, however this did not prove successful. Eventually I found a method where the robot would stay in a constant loop while it was driving over lighter areas. Then when a reading was taken (The robot had found a black bar) It would switch to another loop that would record the length of the black strips. This while loop looped round four times, one for each black strip. There is only one issue with this method however and that is if the robot is driving on a dark surface leading up to the barcode, it would assume it would be driving over a black bar and would start the recording process straight away, which would obviously lead to not recording the correct values.

The way I programmed the robot to record the length of the black strips was by using the millis function. It would record the time when it first drove over a black strip and take the time when it reached the end. It would then subtract the end time from the start time to work out the length of time the robot was driving over the strip for. I then compared it to a constant integer that I had previously. This integer was worked out by me timing how long it took the robot to drive over a long strip and a short strip, and then working out the value in-between these timed values. This middle value was 1200 milliseconds, so if the recorded time was above this value, the robot would know that it had just drove over a thicker strip; and if it was lower than this value it would know it had just drove over a thinner strip.

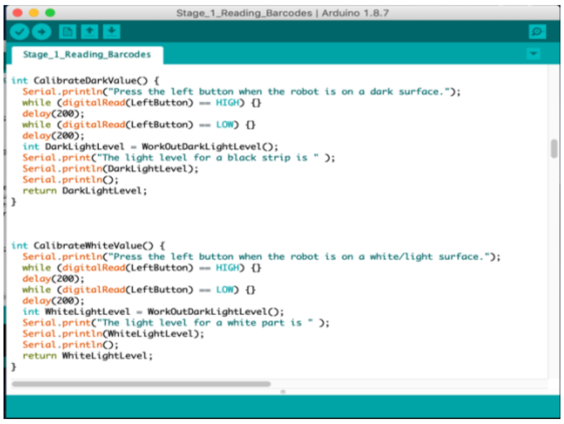
The way a stored the values to compare was by using an array of four elements. Each element either had a 1, if the recorded strip was above the threshold, or a 0, if the recorded strip was below the threshold. After all four values had been calculated and stored, I then used a series of four loops to compare the list to the lists of the known barcodes. After this, if the recorded array matched a known array, it would carry out the function corresponding to the array it matched to. For example, if the recorded array matched the array with the values for a left turn, then the left turn function would be called. If the recorded array did not match any of the known arrays, the robot would continue onwards in search of another barcode.

Before this process beings, the user must calibrate the robots dark and light values in order for it wo differentiate between the dark and white strips. The typical values that it records are around 550-600 for a dark to light transition and 350-425 for a light to dark transition. The reason for these two values is for the robot to work out the difference between light and dark areas more smoothly.

Light to dark and dark to light thresholds calculated



```
int DarkValue =   CalibrateDarkValue();
int LightValue  = CalibrateWhiteValue();
int MiddleValue = ((LightValue + DarkValue) / 2);
    DarkToLight = (MiddleValue + 22);
    LightToDark = (MiddleValue - 22);
```



```
int CalibrateDarkValue() {
  Serial.println("Press the left button when the robot is on a dark surface.");
  while (digitalRead(LeftButton) == HIGH) {}
  delay(200);
  while (digitalRead(LeftButton) == LOW) {}
  delay(200);
  int DarkLightLevel = WorkOutDarkLightLevel();
  Serial.print("The light level for a black strip is " );
  Serial.println(DarkLightLevel);
  Serial.println();
  return DarkLightLevel;
}


int CalibrateWhiteValue() {
  Serial.println("Press the left button when the robot is on a white/light surface.");
  while (digitalRead(LeftButton) == HIGH) {}
  delay(200);
  while (digitalRead(LeftButton) == LOW) {}
  delay(200);
  int WhiteLightLevel = WorkOutDarkLightLevel();
  Serial.print("The light level for a white part is " );
  Serial.println(WhiteLightLevel);
  Serial.println();
  return WhiteLightLevel;
}
```

Calibrate function which records and returns the light threshold values.

```
int WorkOutDarkLightLevel() {
  int DarkLightLevel = 0;
  for (int x=1;x<11;x++) {
    DarkLightLevel = (DarkLightLevel + LightLevel());
  }
  DarkLightLevel = (DarkLightLevel / 10);
  return DarkLightLevel;
}


int WorkOutWhiteLightLevel() {
  int WhiteLightLevel = 0;
  for (int x=1;x<11;x++) {
    WhiteLightLevel = (WhiteLightLevel + LightLevel());
  }
  WhiteLightLevel = (WhiteLightLevel / 10);
  return WhiteLightLevel;
}
```

When robot is driving over a barcode it will record the light level (AverageValue) and will record the time intervals at the start and end of each black bar.

```
long Start;
long End;
long Width;

if (AverageValue < LightToDark) {
 Start = millis();
 while (AverageValue < DarkToLight) {
   LED(1,1,1);
   AverageValue = LightLevel();
 }
}


if (AverageValue > DarkToLight) {
  End = millis();
  Width = End-Start;
  BarWidths[y] = Width;
  y++;
  if (y > 3) {
    break;
  }
  else {
    while (AverageValue > LightToDark) {
      LED(0,0,0);
      AverageValue = LightLevel();
    }
  }
}
```

Stage 2: Obstacle avoidance

When doing this stage separately I put all of my code in the void loop section. The way this code works it by switching on the motion sensor to detect if there is an object in front of the robot. If no object is detected, then the sensor will switch off for a short while and then turn on again to scan again. If the sensor does detect an object in front of the robot then it will carry out the manoeuvre specified, where it will turn right, drive forward and then turn left after driving 25 centimetres. After checking this method worked, I then turned the code into two separate functions; one for checking to see if there is an obstacle and another for carrying out the manoeuvre. The check function is called while the robot is searching for a first black strip and if it spots an obstacle, it will call the manoeuvre function. Overall this stage was fairly simple, however I did have issues implementing it with stage 1 as the delays were messing up the barcode scan.

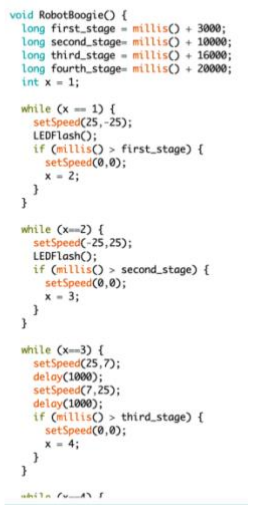
```
void ObstacleCheck() {
  tone(IROutput,38000);
  delay(200);
  if (digitalRead(2) == LOW) {
    ObstacleAvoid();
  }
  noTone(IROutput);
  delay(200);
}


void ObstacleAvoid() {
    noTone(IROutput);
    LED(1,1,1);
    TurnAngle(90);
    DriveDistance(25);
    TurnAngle(-90);
    LED(0,0,0);
    setSpeed(7,7);
}
```

ObstacleCheck function is called while searching for a barcode and if it detects an obstacle it will call ObstacleAvoid.

Stage 3: Let’s dance

For the final stage the main issue was to try to get the robot to dance for 20 seconds. To tackle this problem, I decided to use the millis function again and to split the dance into for sections. Each stage will last a certain amount of time and during each stage the robot will carry out a selection of moves. Using this method, I managed to get my robot to dance for very close to 20 seconds. The moves it carried out were spinning around and driving at curves. I also managed to get the LEDs to flash for certain parts of the dance, which was fairly tricky to implement. The way I got the LEDs to flash was to add small delays between turning them on and off and by creating a function called LEDFlash which would be called during certain parts of the dance.



```
void RobotBoogie() {
  long first_stage = millis() + 3000;
  long second_stage= millis() + 10000;
  long third_stage = millis() + 16000;
  long fourth_stage= millis() + 20000;
  int x = 1;

  while (x == 1) {
    setSpeed(25,-25);
    LEDFlash();
    if (millis() > first_stage) {
      setSpeed(0,0);
      x = 2;
    }
  }

  while (x==2) {
    setSpeed(-25,25);
    LEDFlash();
    if (millis() > second_stage) {
      setSpeed(0,0);
      x = 3;
    }
  }

  while (x==3) {
    setSpeed(25,7);
    delay(1000);
    setSpeed(7,25);
    delay(1000);
    if (millis() > third_stage) {
      setSpeed(0,0);
      x = 4;
    }
  }

  while (x==4) {
```

The boogie function where the robot performs different dance moves for each of the four stages. The LEDFlash function causes the LEDs to flash.

Conclusion

Overall, I think the hardest stage was stage 1 as there were number of issues that needed to be solved, such as working out light levels, working out the thickness of each bar, and working out what state the robot is in. However, I overcame these issues and once stage 1 was completed, I implemented stages 2 and 3 to a decent level. Having completed all 3 of these stages to a fairly solid standard, I think that I will get around 50%-60% for this assignment.