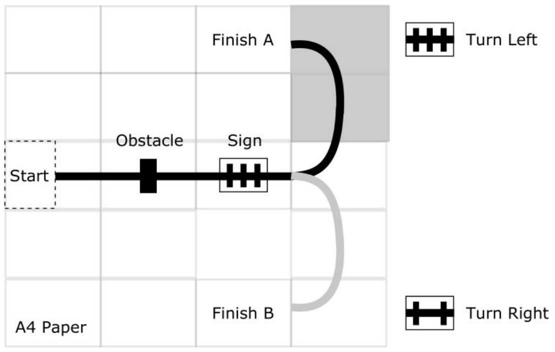
Report

Problem

Description

I was tasked with writing a program that had a robot drive through an obstacle course. I had to make sure the robot would be able to drive through a variety of different courses, meaning I had to do more than just tell it how to move and when. I also had to check the robot would work almost 100%, as if not it would cause even more problems when it came to running it through new courses.



Objectives

To achieve the goal of making the robot drive through the obstacle course, there were a few objectives I had to reach. This included making the robot follow a black line, avoid an obstacle and scan a barcode. To carry all these out I had to complete smaller tasks, test running each element to make sure they would work when I put them in the actual course. Making this required me to use all the functionality of the robot, which includes the servos, the light sensors and the front sensor.

Program

Speed

First off to make the robot move I had to find the values to have the robot in a stationary position. To find these I used trial and error, discovering that the wheels span in different directions, meaning I had to lower one and raise the other to make it go forwards.

```
int setSpeed(int a, int b) {     //Takes speed values
  int leftServoVal = leftStop + a;     //Adds speed to Servo
  int rightServoVal = rightStop - b - 1;
  leftServo.write(leftServoVal);
  rightServo.write(rightServoVal);
}
```

I then created a function called ‘setSpeed’ which I used to be able to change the speed easily. One thing I found is that when I changed the speed of the robot it would drift slightly so I decided to have the value lower by 1 on the right wheel, each time the speed was set. I found that this solved my problem.

Turn angles

Making my robot follow a black line involved me using the light sensors on the bottom of the robot. I decided to make it so the robot would use the outer sensors to test when it ran into a line. To make the robot able to turn when running into a line I would have to write an angle turning function.

```
void turnAngle(int angle) {
  if (angle < 0) {    //Checks which direction the robot will turn
    setSpeed(-10, 10);    //Turns left
    angle = abs(angle);   //makes value positive
  }
  else {
    setSpeed(10, -10);    //Turns right
  }
  delay(28.72 * angle);   //Time taken to turn 1° multiplied by requested angle
  stop();
}
```

I first found how long it took to turn 90 degrees and then divided it by 90. This gave me 1 degree which could then be multiplied by how much the robot needs to be rotated. To make the robot turn the opposite direction I made that you enter a negative number in the 'turnAngle' function instead. To make these values work with the time delay I have used the abs function to turn the angle into a positive number.

<u>Follow line</u>

```
 if ((analogRead(middleSensor) > middleWhite - 20) and (turnRight == false)) {
   turn(rightBlack, leftBlack);
 }
```

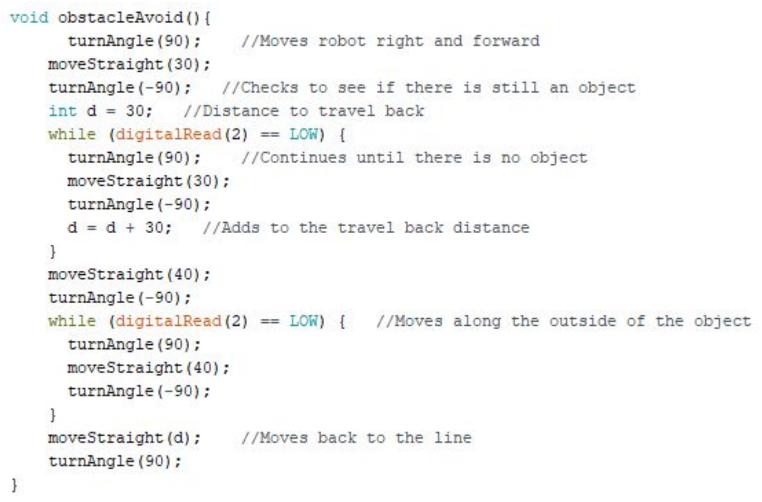
I decided that it would be better to use the middle sensor as well as it would allow me to read the barcodes further on in the program.

```
void turn(int value1, int value2) {   //Takes in which sensors to use
  if (analogRead(rightSensor) < value1 + 20 and analogRead(rightSensor) > value1 - 20) {
    turnAngle(10);
  }

  else if (analogRead(leftSensor) < value2 + 20 and analogRead(leftSensor) > value2 - 20) {
    turnAngle(-10);
  }
}
```

I then wrote a function to allow me to reuse the code, when getting the robot to follow the grey line after reading the barcode. With this code it will turn the robot 10 degrees left or right depending on which side is touching the black.

<u>Obstacle</u>
With this function now in place, it allowed me to go about creating the obstacle avoidance function.

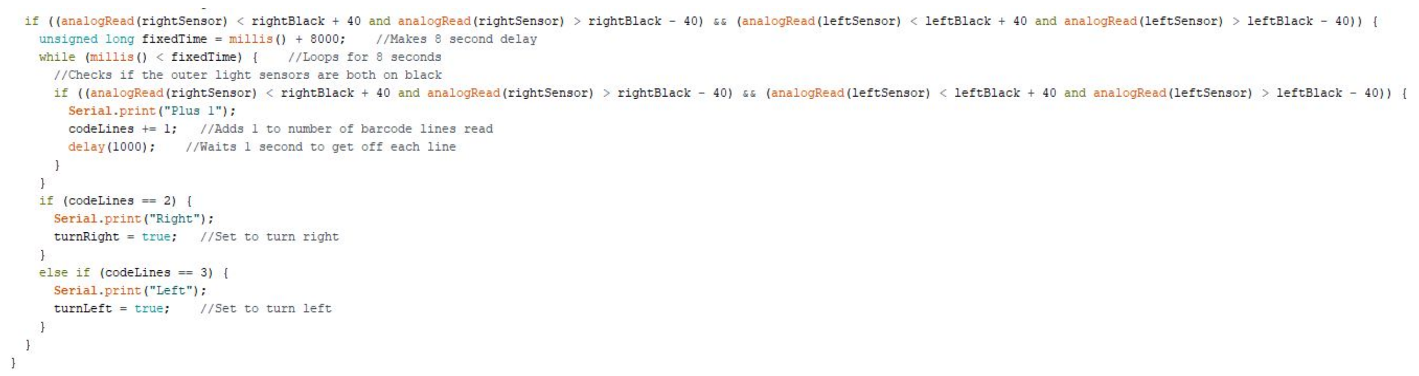


```
void obstacleAvoid(){
      turnAngle(90);    //Moves robot right and forward
    moveStraight(30);
    turnAngle(-90);   //Checks to see if there is still an object
    int d = 30;   //Distance to travel back
    while (digitalRead(2) == LOW) {
      turnAngle(90);    //Continues until there is no object
      moveStraight(30);
      turnAngle(-90);
      d = d + 30;   //Adds to the travel back distance
    }
    moveStraight(40);
    turnAngle(-90);
    while (digitalRead(2) == LOW) {   //Moves along the outside of the object
      turnAngle(90);
      moveStraight(40);
      turnAngle(-90);
    }
    moveStraight(d);    //Moves back to the line
    turnAngle(90);
}
```

To write my code I had to first find a way to detect if there was an object in front of the robot. I used the front sensor to check for an obstacle and if there is one run the ‘obstacleAvoid’ function. My function has the robot turn 90 degrees and move straight 30cm. It then turns back to check that the obstacle has gone. If not it will turn back and continue until there is no obstacle, recording the distance it is travelling using the ‘d’ variable.

<u>Barcode reading</u>

To make the robot be able to read the barcode I have made it so all light sensors have to be on the black line at once to then start to could the lines. This allows me to count the total lines on the barcode. This runs for 8 seconds and has 1 second gaps between counting to allow the robot to get off each line.



```
  if ((analogRead(rightSensor) < rightBlack + 40 and analogRead(rightSensor) > rightBlack - 40) && (analogRead(leftSensor) < leftBlack + 40 and analogRead(leftSensor) > leftBlack - 40)) {
    unsigned long fixedTime = millis() + 8000;    //Makes 8 second delay
    while (millis() < fixedTime) {    //Loops for 8 seconds
      //Checks if the outer light sensors are both on black
      if ((analogRead(rightSensor) < rightBlack + 40 and analogRead(rightSensor) > rightBlack - 40) && (analogRead(leftSensor) < leftBlack + 40 and analogRead(leftSensor) > leftBlack - 40)) {
        Serial.print("Plus 1");
        codeLines += 1;   //Adds 1 to number of barcode lines read
        delay(1000);    //Waits 1 second to get off each line
      }
    }
    if (codeLines == 2) {
      Serial.print("Right");
      turnRight = true;   //Set to turn right
    }
    else if (codeLines == 3) {
      Serial.print("Left");
      turnLeft = true;    //Set to turn left
    }
  }
}
```

Results

I found my code to work very well when going through the testing phases, having it able to follow the line with ease. Having the robot drive around the obstacle also worked fairly well, with it working almost without flaw. I did however run into a problem when testing the barcode and going round a corner.

I found that the robot would sometimes not read the barcode properly and would sometimes end drifting slightly, botching the read. I tried to fix this by adjusting the obstacle avoidance script, which increased the success rate by a lot. I then found that when the robot tried to go round the corner it wouldn’t. I soon realised that this was due to the lighting conditions getting worse as the day went on meaning I had to readjust the light values of the sensors.

Challenges

The most difficult issue I ran into when programming my robot is that it would sometimes not listen to the code at all and start spinning the wheels at random speeds. This took lots of time to realise that the batteries in the robot were dying and needed to be changed to make it more responsive.

I was also unable to figure out how to use EEPROM after trying for many hours, with my code not accepting the file that I was trying to have it read. If I had more time on the project I would go back and try again to add EEPROM to the code as it would allow me to store values easier than I currently do.