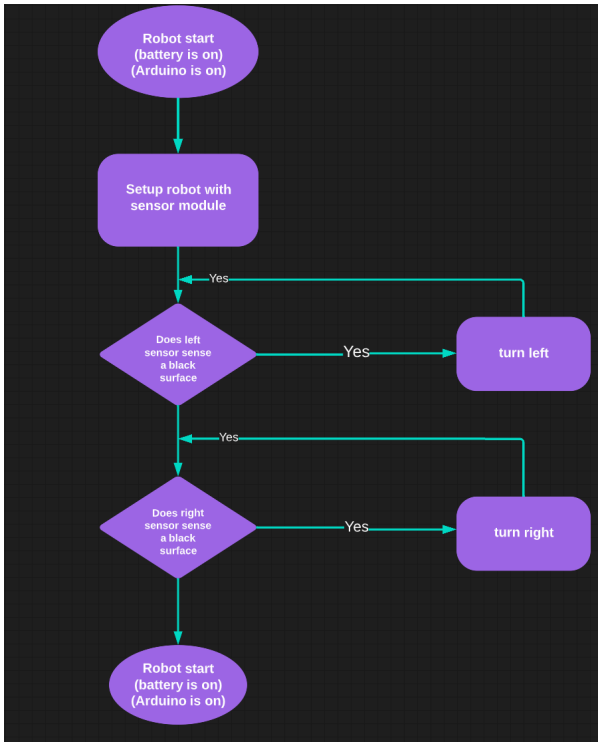
Pre-Lab 2 Report

Shashwat Choudhry
ECE 100-L04
Teammate(s):Ryan Butnariu
Prof. Zhou
Lab Date: 09/08/2022
TA:
Due Date: 09/08/2022

Problem Statement

As a team how do we build upon our fundamentals from our last lab, to complete a line follower path created by our class(ECE - L04) TA. Once we complete the build the problem is how do we build the most optimal line follower robot ?

Investigation/Research



In our previous lab we completed the build of the OSOYOO V.2.1 which is an Arduino based robot car. It includes the Arduino IDE which we received some sample code for; then using said sample code we were able to manipulate the starter code we were able to draw shapes and designs. Now for our second lab we are given the task of tracking a white line on a black. The most important parts of the car in this lab experiment are the tracking sensor modules, they are referred to as the line tracking sensors as well. The sensors we received track the reflective lines on the floor using a infrared(IR) emitting LED light and a infrared(IR) receiving pair. [1]

The IR emitting LED will sense if the surface is black or white(it then sends a signal to the Arduino as either 0 or 1 [2]).

If the sensor on the left senses the surface to be black, the LED turns on and the car is signaled to turn left. If the sensor on the right notices the surface to be black, the LED turns on again and the car turns right. In this fashion, the car keeps moving in a zig-zag pattern on the edges of the black tape and avoids the white surface. So if we are able to design an effective piece of code the car would move through the lines very quickly through the tape. The started code provided is based upon a black line on white surface, so we must edit the code so we can work within our competition environment

Alternative Solutions

Now if the robot had only 2 light sensors installed on the front wing area the sensor would provide an inaccuracy by only sensing the middle of the white line, which the edges of the lines wouldn't be tracked. So in this case we have an error with the sensors not detecting the edges of the line or any simple irregularities(only 2 sensors checking versus 5).
In this scenario, the sensors wouldn't notice if the wheels of the car are slightly off the black tape, as long as the sensors in the middle are on the black tape. Now, this can be remedied by adding sensors: 3 sensors, 4 sensors, and so on. Now how can we make this the most effective robot given any sensor setup? Now if one of our sensors is faulty or is damaged during a test what are our fail cases or anything we can do to make sure even with a sensor failure our robot still can continue on the line following the path? Ideally, we can have sensors on the from and back that relay data from two points to create more unison within the front and the back of the car. So with extra sensors, we can create a more optimal solution.

Since these solutions rely on both hardware and software optimization; there always is one other choice, brute force. It's a form of pure hardware optimization and a non-effective solution, where we code the program to just follow the line instead of detecting it

Optimum Solution

The robot car is fitted with five line tracking sensors – one near the front-left wheel, one in the front-middle part, one in the front-right wheel, one in the rear-left wheel, and one in the rear-right wheel. The sensor in the middle tracks if the car is on the black line. If the sensor near the front-left wheel notices the surface to be black, it means that the car is deviating from the black line towards the right. Then the LED indicator then turns on and the car is signaled to turn left. If the sensor on the front-right notices the surface to be black, the car is deviating from the black line towards the left. The LED indicator turns on again and the car turns right.

However, the sensors near the rear wheels work in the opposite direction from the sensors on their sides near the front wheels. If the sensor near the rear-left wheel notices the surface to be black, it means that the car is deviating from the black line towards the left. The sensor then signals the car to turn right. If the sensor near the rear-right wheel notices the surface to be black, then the car is deviating from the black line towards the right. Thus, the car turns left.

In this way, the front-right wheel works in tandem with the rear-left wheel, while the front-left wheel works in tandem with the rear-right wheel. The car tracks the line more accurately and adjusts quicker, since two sensors at a time signal a left or right turn.

Once this is calibrated, the code is edited to work on white tape on a black surface. To do this, the '0' and '1' value in the code is interchanged. So with the starer code below:

```
 }
 if (    sensorval=="00001"  ){ //The black line is  on the right of the car, need  right turn
   go_Right();  //Turn right
   set_Motorspeed(FAST_SPEED,FAST_SPEED);
 }
```

The code is then altered to the following to make it work on white line instead:

```
 }
 if (    sensorval=="11110"  ){ //The black line is  on the right of the car, need  right turn
   go_Right();  //Turn right
   set_Motorspeed(FAST_SPEED,FAST_SPEED);
 }
```

References

[1] https://protosupplies.com/product/optical-tracking-sensor-module/

[2] -


```
/*read sensor value string, 1 stands for black, 0 starnds for white, i.e 10000 means the first
sensor(from left) detect black line, other 4 sensors detected white ground */
string read_sensor_values(){
   int sensorvalue=32;
  sensor[0]= !digitalRead(LFSensor_0);

  sensor[1]=!digitalRead(LFSensor_1);

  sensor[2]=!digitalRead(LFSensor_2);

  sensor[3]=!digitalRead(LFSensor_3);

  sensor[4]=!digitalRead(LFSensor_4);
  sensorvalue +=sensor[0]*16+sensor[1]*8+sensor[2]*4+sensor[3]*2+sensor[4];
```

```
  String senstr= String(sensorvalue,BIN);
  senstr=senstr.substring(1,6);


  return senstr;
}
```

Appendix

- Lab-02 Default Code
- Lab-01 Default Code