Obstacle Avoidance

# Overview

An obstacle avoidance robot will employ a pair of sensors and a pair of motors working in concert to navigate a path through an obstacle course.

# Analog Sensors

The analog sensors were the best choice for this project for the range of distance detection it could handle over the digital sensors. Distance readings range from 0 to 4316, where the max value is 3 inches away from the surface. At which point, the sensor returns quickly back to 0 as it makes contact with the surface.

# Motors

I will have a pair of motors to control. Each will responds to an analog sensor. The will be connected to ports A and B. I will have to calibrate forward and reverse for the way the motors are connected to the actual vehicle. Based on the motor labs I will assume these motors will also range from 0 to 255 each. I would like to start slow and as I am able to calibrate the distance and response time, as well as my calculations. The idea is to throttle the opposing motor in respect to the sensor. Ideally, I would incorporate at least two more sensors that would help determine the better direct to turn in the event one would be a better choice than the other. But, in the interest of time and resources. I will work with the assumption that left is better… that is unless the motors are reversed from my assumptions. Of course if designed correctly, I would be able to configure the system to conform to my assumptions once I have the vehicle.

# Design

## General

The motors on the same set of interrupts. The sensors will run off a different set of interrupts. I will set them up to run at 100 ms intervals for taking the readings and calculating the adjustments. For the naming, I will use LEFT and RIGHT. The set up will be much like they say the human brain works. The LEFT sensor will throttle the RIGHT motor. While the RIGHT sensor will throttle the LEFT motor.

The motor speed will be adjustable by the menu to facilitate the testing.

## Menu

The menu will allow adjustments to the system while it is running. So far I have identified the following commands that would come in handy:

**S <int>**: Adjust the target speed for both motors when there is no obstacle detected

**R <int>**: Adjust the top range that obstacles are detected

**r <int>:** Adjust the low range that obstacles are detected

**H <int>:** Adjust the hysteresis to minimize changes due to fluctuations

**P <float>:** Adjust the P values for each motor. Assuming the motors would behave differently and require seperate values

**I <float>:** Adjust the P values for each motor. Assuming the motors would behave differently and require seperate values

**D <float>:** Adjust the P values for each motor. Assuming the motors would behave differently and require separate values

# Considerations for Calculation and Scheduling

I will be using the Pololu libraries to help expedite the development. However, I will still need to have a PID controller to help keep the motor reactions in control. The SPEED PID controller seems like the best option at this point. The SPEED PID controller will be defined in a timer to yield the new speed for each motor. Here is the breakdown of Motor reaction to Sensor readings:

* RANGE: 700 to 900
* OD – Obstacle Detected
* N-OD – No Obstacle Detected
* MS – Maintain Target Speed
* RS – Reduce Speed
* FS – Full Stop

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Left Sensor** | **Right Sensor** | **Exit Range** | **Left Motor** | **Right Motor** |
| OD | N-OD | Low | MS | RS |
| OD | OD | Low | MS | RS |
| N-OD | OD | Low | RS | MS |
| N-OD | N-OD | Low | MS | MS |
| OD | N-OD | High | MS | RS |
| OD | OD | High | FS | FS |
| N-OD | OD | High | FS | MS |
| N-OD | N-OD | High | FS | FS |

In an ideal system I would use GPS to put the vehicle back on track. Since I do not have any GPS modules and considering the scope of this project, I will rely on the course to help ensure the path taken is appropriate. When the motors are in Full Stop state, I think it would be best to have them go to target speed once the vehicle is moved to a new location. If an obstacle is present when the vehicle is placed, it will perform the normal operations according to the table.

# Initial Settings

* + Left Sensor – Channel 0
  + Right Sensor – Channel 1
  + Motor 1 – Determine when I see the vehicle
    - Encoders will need to be set up on PD0/1 which translates to PCINT24/25
  + Motor 2 – Determine when I see the vehicle
    - Encoders will need to be set up on PD2/3 which translates to PCINT26/27
  + Target Speed – 128
  + PID Controller on Timer 3