DATE: 11/09/2013
DAY : wednesday

ASSIGNMENT #2

LINE FOLLOWER WITH PID CONTROLLERS

(TEAM 1**- SUMUKHA BN; CHANDAN KUMAR R; SKANDA S BHARADWAJ; SUGHOSH HK **)

❖ *CONTROL SYSTEMS AND ITS COMPONENTS THAT GOVERNS THE SYSTEM:*

- Target – It is the position of the robot that we are concerned about, with respect to the line.
- Current Position – It is the current position of the robot with respect to the line.
- Error - It is the difference between the current position and the target. It can be negative, positive or zero.
- Proportional – It tells us how far the robot is from the line like – to the right , to the extreme right, to the left or a little to the left. Proportional is the fundamental term used to calculate the other two.
- Integral – It gives the accumulated error over time. It tells us if the robot has been on the line in the last few moments or not.
- Derivative – It is the rate at which the robot oscillates to the left and right about the line.

Kp, Ki and Kd are the constants used to vary the effect of Proportional, Integral and Derivative terms respectively.What the controller does is first calculate the current position. Then calculate the error based on the current position. It will then command the motors to take a hard turn, if the error is high or a small turn if the error is low. Basically, the magnitude of the turn taken will be proportional to the error. This is a result of the Proportional control. Even after this, if the error does not decrease slowly, the controller will then increase the magnitude of the turn further and further over time till the robot centers over the line. This is a result of the Integral control. In the process of centering over the line the robot may overshoot the target position and move to the other side of the line where the above process is followed again. Thus the robot may keep oscillating about the line in order to center over the line. To reduce the oscillating effect over time the Derivative control is used.

❖ *Sensors*

Sensors are the first requirement in the process of line following. There are many sensors one can use for line following – IR LED and a Photodiode, LED and LDR (photoresistor), etc. When selecting the type of sensor there are three things you need to keep in mind – response time, sensitivity and ambient light protection.

LDRs (photoresistors) are excellent in terms of sensitivity, they can sense not only between two contrasting colours(one dark and one light) but also between two primary colours. Some competitions make the line following tougher by having a white line on a coloured surface (blue, red, grey). In such cases LDRs are the sensors to be used. But there is one big disadvantage in using LDRs - their slow responce time. A LDRs reaction time is around 0.1s. This may not seem much, but if you have a robot using LDRs following a line with a speed of 1 m/s then it will be getting data from the sensor every (0.1 * 1 = 0.1m) 10 cm !

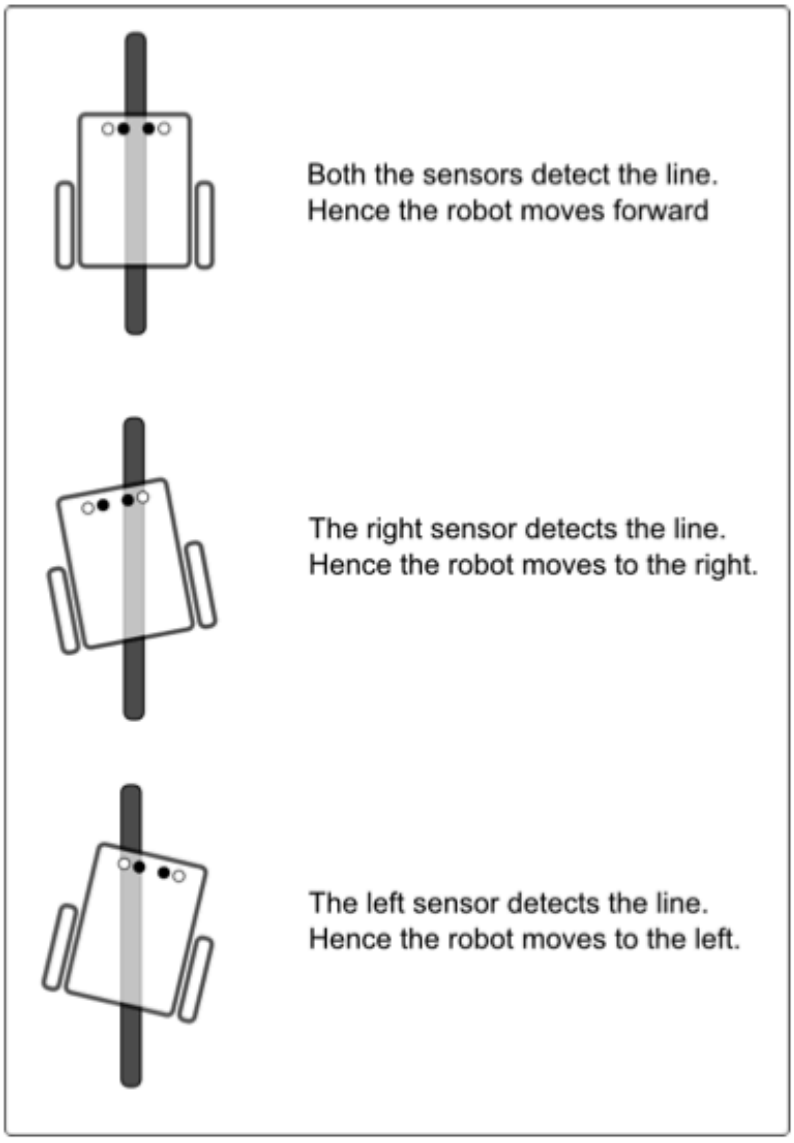
IR LED and photodiodes give good response time. But they have two drawbacks, both of which can be dealt with – they can sense only between two contrasting colours and they are easily affected by ambient light.

Sensitivity can be improved in two ways. One by using LED which emits light of any other color other than the surface color and using a suitable photodiode to sense the light. Basically, if there is a white line on a red surface, you can use any other color LED, say green, to emit light and photodiodes which react to green colour. What happens is the surface, being red, absorbs all the green light emitted by the LED and no light reaches the photodiode. The line on the other hand, being white, reflects all the green light falling on it back to the photodiode. Using this method significantly increases the sensitivity of the sensors. Another way of improving sensitivity is by making modifications to the sensor circuit. Just by varying the resistance of some of the resistors in the sensor circuit, you can tune it to sense between two particular colors. Take a look at www.societyofrobots.com/schematics_infraredemitdet.shtm .

The second draw back, that is, interference from ambient light, is much more difficult to deal with. IR photodiodes detect light of a particular wavelength, which usually between 700 - 850 nanometers. Unluckily, the

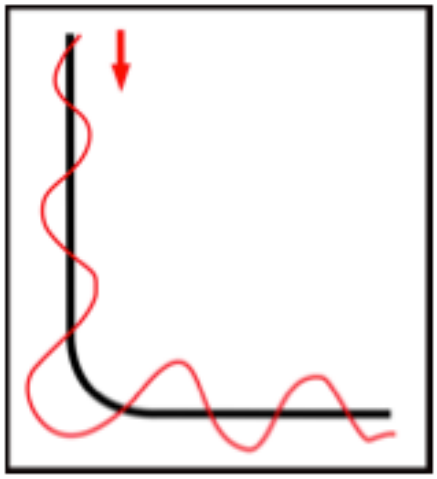
wavelength of most of the IR radiations around us fall in this range. To the photodiode these radiations are same as the IR radiations emitted by the IR LED, and hence they get falsely triggered by them. Even red light (which comes just beyond the IR radiations in the electromagnetic spectrum) can affect these sensors. There are many methods you can employ to protect IR sensors from interference from ambient light. One of the easiest method is to physically shield the sensors from ambient light. You can also use sensors that provide ambient light protection like the TSOP 17** sensors.

THE PROCESS BEING CARRIED OUT:

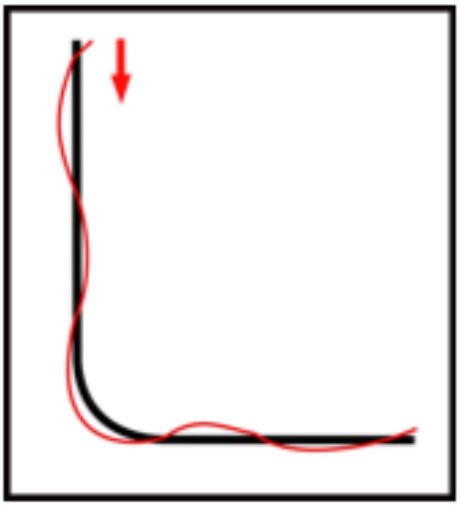


WHY PID CONTROLLER??

As we can see clearly in the picture, the robot oscillates a lot about the line, wasting valuable time and battery power. By now you must have realized that there is a maximum speed beyond which you cannot use this algorithm, other wise the robot will overshoot the line.



A robot with PID control will follow the line as shown below



And now we can see clearly the robot will follow the line smoothly keeping its centre always above the line. In straight lines, the robot will gradually stabiles go straight unlike a robot with the left-right algorithm. This enables the robot to follow the line faster and more efficiently.

❖ *Tuning the PID control:*

This is the most interesting part in building a PID control. In this step one should tune the Kp, Ki and Kd values to get the best results. I cannot give you the values because what works for me will not work for you. The optimum Kp, Ki and Kd values vary a lot from robot to robot. And the best way to determine the optimum values is by trial and error.

First, set all values to 0 and start with tuning the Kp value. First time I just gave an approximate value. Seeing the robot perform will determine what you should do next. If the robot wobbles a lot reduce the Kp value, if the doesn't follow the line (goes straight in curves) increase the Kp value. Tune the Kp value till the robot smoothly follows the line. By now you will have observed that the robot goes with no acceleration on straight lines. Now, tune the Kd term. After tuning the Kd term move to the Ki term. After which, you will see the robot first center over a straight line and then accelerate also. The optimum Kp, Ki and Kd values vary a lot even from track to track. You can only know these optimum values by testing.

❖ *ACTUATORS:*

The actuators used here are dc motors.

A DC motor is a mechanically commutated electric motor powered from direct current (DC). The stator is stationary in space by definition and therefore its current. The current in the rotor is switched bythe commutator to also be stationary in space. This is how the relative angle between the stator and rotor magnetic flux is maintained near 90 degrees, which generates the maximum torque.

The dc motor is being controlled by the microcontroller. Based on the inputs of the sensors the controller decides the amount of torque to be produced and the direction in which the plant has to rotate so as to make the robot follow the line. As an example when the bot moves a bit towards the right of the line, a corresponding input is given to the controller which will guide the motors to rotate such that the bot moves a bit left and remains along the line. The left motor is stopped temporarily and the right motor is moved so as to make the bot move left and stay along the line.

❖ *COMPUTATIONAL ELEMENTS:*

Following are computational part of the system to tune the PID controller.

Error = target_position – current_position //calculate error

P = Error * Kp //error times proportional constant gives P

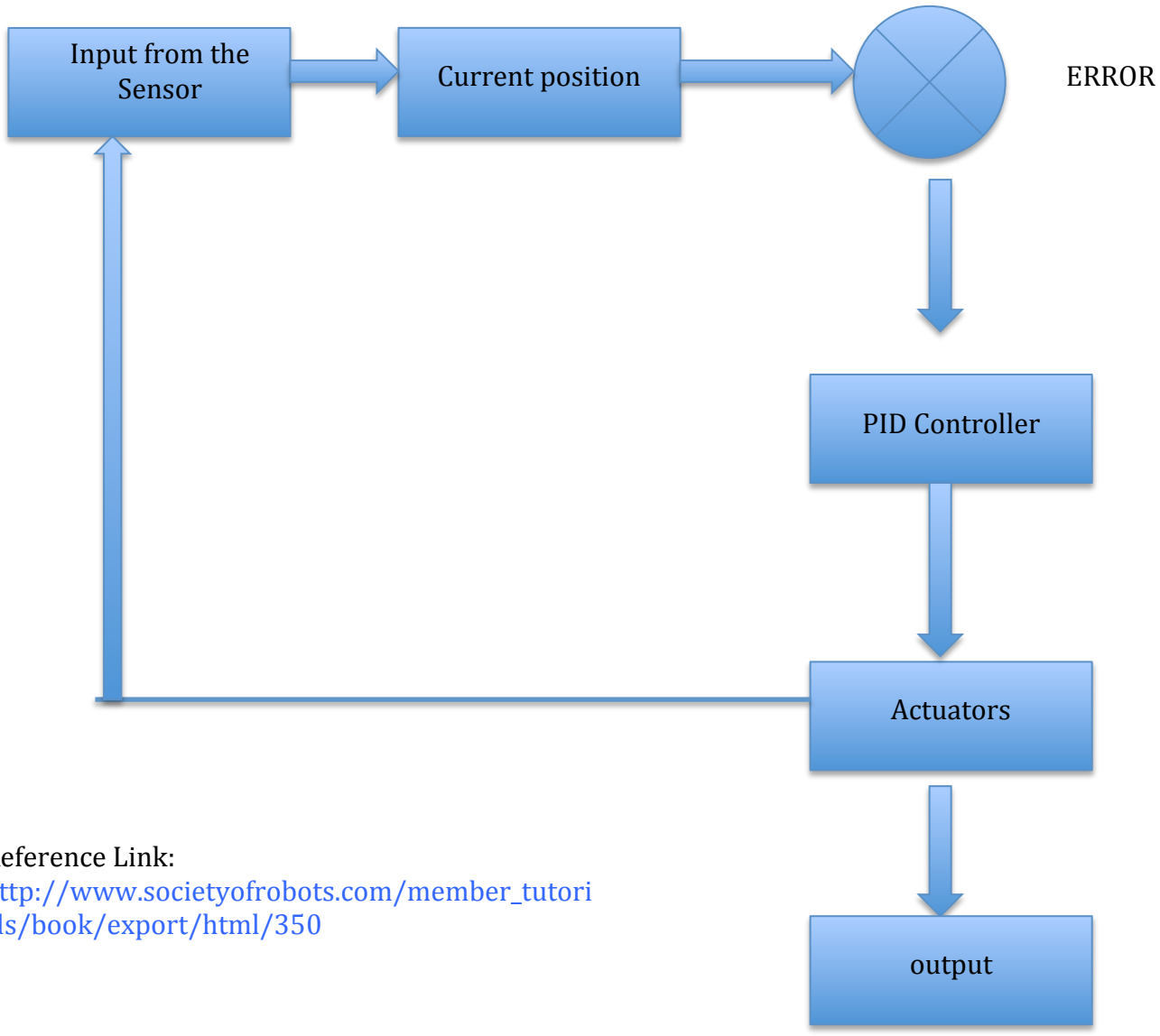
I = I + Error //integral stores the accumulated error

I = I * Ki //calculates the integral value

D = Error – Previous_error //stores change in error to derivate`

Correction = P + I + D

Signal Flow in the plant:



Reference Link:
http://www.societyofrobots.com/member_tutorials/book/export/html/350