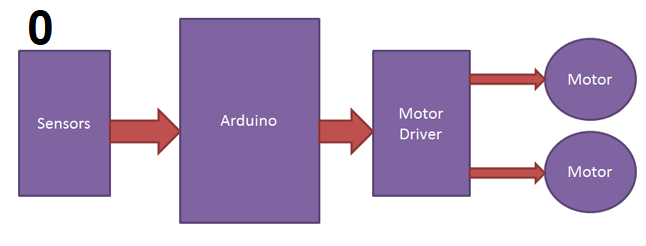
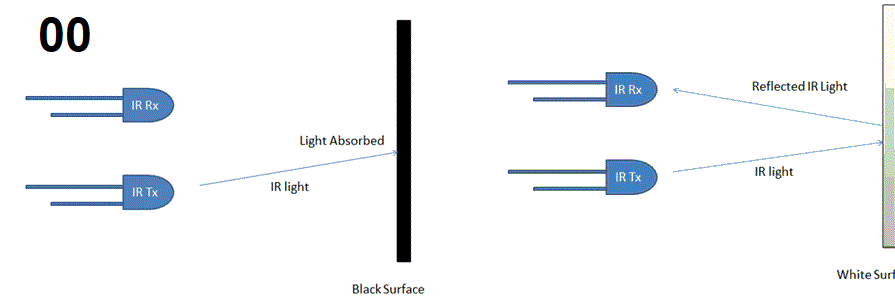
**1-What is the Line tracker?**

* It’s an infrared sensor.

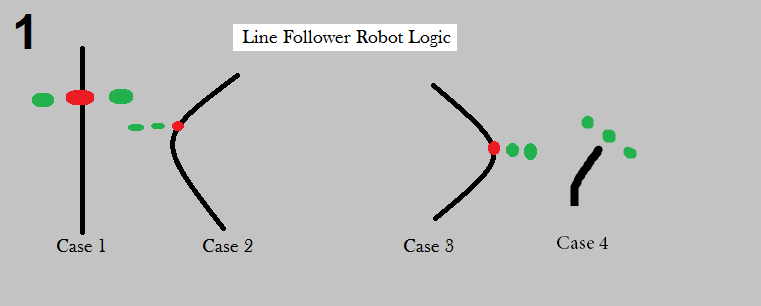
# 2-How is the line is sensed?

# The IR emitter emits a constant IR beam. The white surface reflects most of the beam while the black surface absorbs most of the beam. This reflected beam is picked up by the IR detector and its conduction increases and hence a voltage variation in the output pin (0v for absence of IR rays and 5V for maximum intensity). This is given to the comparator to compare with a reference signal generated by the potentiometer. That is how the line is sensed by the sensors.





Let’s assume these are the only four possible conditions for the sake of understanding. The red dots are the sensors that are just above the black line and the green ones are on the white background.



Case 1 is when the robot has to move forward as only the center sensor is low. Similarly Case 2 is towards right, Case 3 is towards left and case 4 is stop as all the sensors read high. This is how your code should look like (the terms, LEFT\_SENSOR, RIGHT\_SENSOR, CENTER\_SENSOR, GO\_FRONT, TURN\_RIGHT, TURN\_LEFT, STOP are all code macros for the corresponding pin in the microcontroller).

while (1)

{

if((LEFT\_SENSOR==1)&&(CENTER\_SENSOR==0)&&(RIGHT\_SENSOR==1))

LATB |= GO\_FRONT; // Case 1

if ((LEFT\_SENSOR==1)&&(CENTER\_SENSOR==1)&&(RIGHT\_SENSOR==0))

LATB |= TURN\_RIGHT // Case 2

if ((LEFT\_SENSOR==0)&&(CENTER\_SENSOR==1)&&(RIGHT\_SENSOR==1))

LATB |= TURN\_LEFT // Case 3

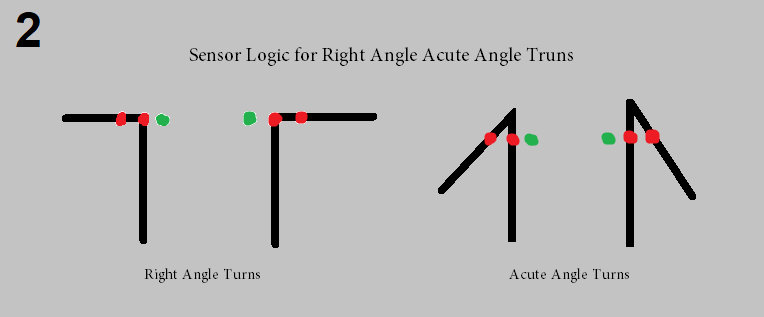
if ((LEFT\_SENSOR==1)&&(CENTER\_SENSOR==1)&&(RIGHT\_SENSOR==1))

LATB |= STOP; // case 4

}

This code should be good for any linear track with normal bends. But this is not going to work if your track has right and acute angle turns. That is when we have to add some more logic to the existing one.

Say you are facing a right angle, what would your sensor data read? The center sensor along with either of the two side sensors will read LOW. This is when you have to tweak the direction logic a little.



Now you have to define two kinds of left turns. One of it will rotate your right motor forward and the left motor backward. Call this one “Power Left” as it turns left quite fast. The other one is to rotate the right motor forward while the one on the doesn’t left rotate at all. Call this just “Left”. Do the same thing with the right turn logic.

For normal conditions use the “Left” or “Right” turn. When the bot encounters a right turn or an acute turn use the “Power Left” and “Power Right” this way your bot will respond faster to these cases and act accordingly. Now your code should have these lines too,

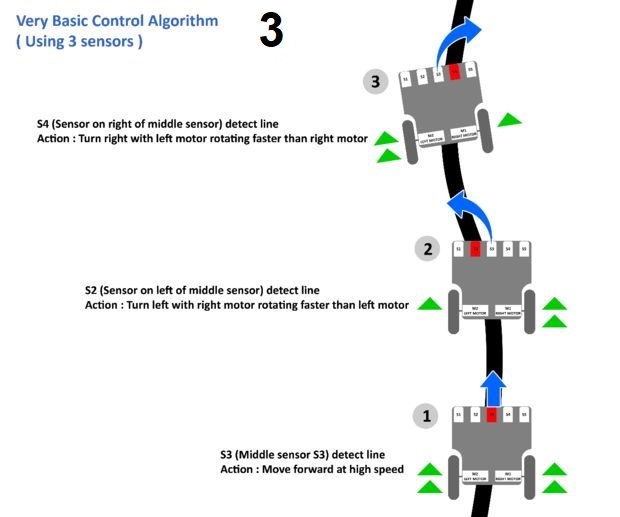
if ((LEFT\_SENSOR==1)&&(CENTER\_SENSOR==1)&&(RIGHT\_SENSOR==0))

LATB |= POWER\_RIGHT // Case 2

if ((LEFT\_SENSOR==0)&&(CENTER\_SENSOR==0)&&(RIGHT\_SENSOR==1))

LATB |= POWER\_LEFT // Case 3

By adding this logic to your controller you are able to tackle all kinds of bends and turns in a linear track, but you won’t be able to use it on tracks that have intersections and discontinuities.



Ref : <https://embedjournal.com/programming-line-follower-robot/>

https://embedjournal.com/line-follower-robot/