Project Name: Line following robot using Arduino.

Introduction:

In a line follower robot, we use the behavior of light at black and white surface. When light fall on a white surface it is almost fully reflected and in case of black surface light is completely absorbed. This behavior of light is used in building a line follower robot.

In this arduino based line follower robot we have used QTR sensor for sending and receiving light. IR transmits infrared lights. When infrared rays falls on white surface, it's reflected back and caught by the sensor. When IR light falls on a black surface, light is absorb by the black surface and no rays are reflected back, thus the sensor does not receive any light or rays.

Working Principle:

We have used PID.PID means PROPRTIONAL, INTEGRATION and DERIVATIVE. We used QTR 8 array IR sensor. When IR is on black surface receiver doesn't receive any light back but when on the white surface it receives the light back. The middle two sensor must be on the black line. During turn, if three right sensors are on the black line that means robot should be turn left. Sensor reading measures error and errors are multiplied by kp and kd. And the formula "total error=kp*error +kd (error-previous error)" is used to measure total error. Then this total error measures the motor speed. This way if robot has to turn left ,right motor speed up by using above formula and turn left and vice versa. This way robot tracks the black line.

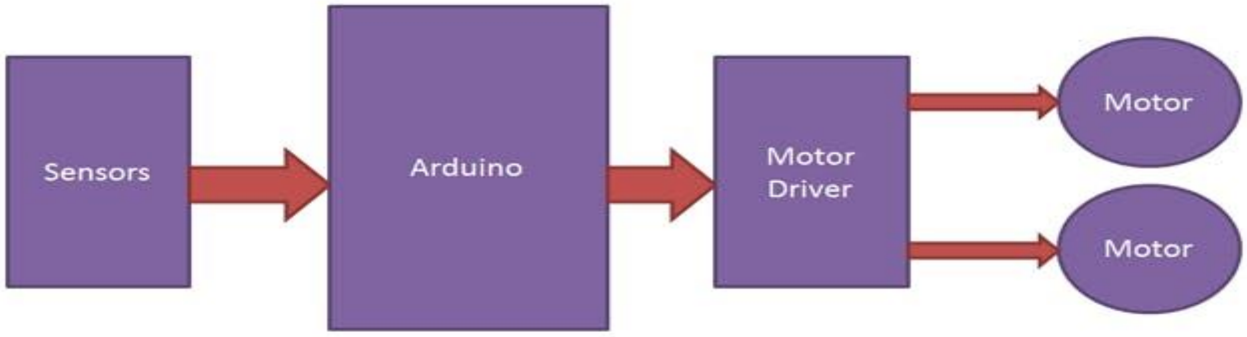

Fig- Block Diagram of Line Follower Robot

List of Components:

1. Arduino Nano V3.0(CH340)

2. Breadboard (medium size)

3. Jumper wire 40 Pcs set- jumper wire type: Male to female

4. Jumper wire 40 Pcs set- jumper wire type: Male to male

5. Line follower sensor – 8 array

6. Motor driver – Dual TB6612FNG (1A)

7. N20 12V 100 RPM Micro Metal DC Gear Motor

8. N20 DC Gear Motor Wheel 3PI miniQ Car Wheeyre

9. N20 motor running bracket

10. Plastic wood

Cost Estimation:

No of items	Product name	Total price
01	Arduino Nano v3.0(CH340)	BDT 315
02	Breadboard (medium size)	BDT 75
03	Jumper wire 40 Pcs set- jumper wire type: Male to female	BDT 70
04	Jumper wire 40 Pcs set- jumper wire type: Male to male	BDT 70
05	Line follower sensor – 8 array	BDT 499
06	Motor driver – Dual TB6612FNG (1A)	BDT 350
07	N20 12V 100 RPM Micro Metal DC Gear Motor	BDT 560
08	N20 DC Gear Motor Wheel 3PI miniQ Car Wheel Tyre	BDT 160
09	N20 motor running bracket	BDT 100
10	Plastic wood	BDT 170
11	Shipping cost (Sundarban Courier Service)	BDT 99
	Total cost	BDT 2468

Construction Procedure:

1. in this project, we have used a motor driver FNG6612, Arduino Nano and a QTR sensor (IR sensor). motor driver has 16 pins altogether. We have used two motor namely MotorA and MotorB.

Motor Driver and Motor Connection

2. From motor, two pins i.e. MA1 and MA2 goes to the AO1 and AO2 pins of the motor driver.

3. From motor, two pins i.e. MB1 and MB2 goes to the BO1 and BO2 pins of the motor driver.

Motor Driver to Arduino Connection

4. We powered up the motor driver from the arduino. For this we connected the battery with the arduino. A wire is connected from the +v of the battery to the arduino's V_{in} pin and the –v of the battery is

connected to the arduino's GND pin.

5. From there we connected the V_{in} pin to the motor driver's VM pin. Also Arduino 5V pin was connected to the V_{cc} pin of the motor driver. Arduino and motor driver was enternally grounded.

6. The rest of the connections included:

Motor Driver	Arduino
PWMA	D3
PWMB	D5
AIN1	D2
AIN2	D4
BIN1	D8
BIN2	D7
STBY	D11

QTR sensor to Arduino Connection:

7. To power up the QTR sensor ,we connected the Vcc pin of the sensor to the Arduino 5V pin and the ground of the sensor was enternally shorted with the arduino's ground pin.

8. The rest of the connections included:

QTR sensor	Arduino
1	A0
2	A1
3	A2
4	A3

5	A4
6	A5
7	A6
8	A7

Arduino Code:

```
#define left_base_speed 160
#define right_base_speed 160
#define STB 11
#include <QTRSensors.h>
QTRSensors qtr;
const uint8_t SensorCount = 6;
uint16_t sensorValues[SensorCount];
int ir[6];
int left_speed, right_speed, error, sum, previous_error, total_error;
float kp = 4.1, kd =1.5; /* kp=1.55, kd=1.2 */
int button;
int stby=0;
int sensorMax=0, sensorMin=1000;
void setup() {
  digitalWrite(2,HIGH); // Motor Driver STBY if you use TB6612FNG / FNG
  Serial.begin(9600);
  pin_setup();
  qtr.setTypeRC();
  qtr.setSensorPins((const uint8_t[]){A0,A1,A2,A3,A4,A5}, SensorCount);
  digitalWrite(STB,1);
 }
void loop() {
// motor(255,255);
  sensor_read();
  sensor_print();
  LINE_TRACK();
//  node_check();
 }
void pin_setup()
{
  pinMode(2, OUTPUT);
  pinMode(3, OUTPUT);
  pinMode(4, OUTPUT);
  pinMode(5, OUTPUT);
  pinMode(6, OUTPUT);
  pinMode(7, OUTPUT);
  pinMode(8, OUTPUT);
  pinMode(9, OUTPUT);
 }
void motor(int left_speed, int right_speed) {
  if (left_speed >= 0) {
    digitalWrite(2, HIGH);
    digitalWrite(4, LOW);
    analogWrite(3, left_speed);
  }
  else {
    digitalWrite(4, HIGH);
    digitalWrite(2, LOW);
    analogWrite(3,abs(left_speed));
  }
  if (right_speed >= 0) {
    digitalWrite(7, HIGH);
    digitalWrite(8, LOW);
    analogWrite(5, right_speed);
  }
  else {
    digitalWrite(8, HIGH);
    digitalWrite(7,LOW);
    analogWrite(5,abs(right_speed));
  }
}

void sensor_read() {
 // read raw sensor values
  qtr.read(sensorValues);
  for (uint8_t i = 0; i < SensorCount; i++)
  {
    ir[i] = sensorValues[i]>500;
  }

  // print the sensor values as numbers from 0 to 2500, where 0 means maximum
  // reflectance and 2500 means minimum reflectance
```

```
}

void sensor_print() {
  for (uint8_t i = 0; i < SensorCount; i++)
  {
//    ir[i] = sensorValues[8-i];
    Serial.print(ir[i]);
    Serial.print('\t');
  }
  Serial.println();
//  delay(200);       // delay for 200ms to observe the sensor value easily
                      // but turn off any delay while Line follower is tracking
}
void LINE_TRACK() {
  error = (ir[0]*-25+ ir[1]*-10 + ir[2]*0 + ir[3]*0 + ir[4]*10+ir[5]*25 );
  total_error = kp * error + kd * (error - previous_error);
  left_speed = left_base_speed +total_error;
  right_speed = right_base_speed - total_error;
  previous_error = error;
  if (left_speed > 255) {
    left_speed = 255;
  }
  if (right_speed > 255)
  {
    right_speed = 255;
  }
  motor(left_speed, right_speed);
  }
void node_check()
{
   /////// This function has to be edited according to the nodes of the tracks.
   // It's just a demo Node check function to easily understand how to use delay
function..
  if(ir[0]==1 && ir[1]==1&& ir[2]==1&& ir[3]==1)
  {
    motor(-50,-50); // Right_down_acute
    motor(100,100);
    delay(200);
    sensor_read();
    while(ir[3]==0&& ir[4]==0)
    {
      motor(90,-90);
      sensor_read();
    }
    motor(120,80);
    delay(400);
   }
  if(ir[7]==1 && ir[6]==1&& ir[5]==1&& ir[4]==1&& ir[3]==1)
  {
    motor(-50,-50); // Left_down_acute
    motor(100,100);
    delay(100);
    sensor_read();
    while(ir[4]==0)
    {
      motor(-90,90);
      sensor_read();
    }
     motor(80,140);
    delay(300);
   }
if(ir[2]==1&&ir[3]==1&&ir[4]==1&&ir[5]==1)
{
 while(ir[2]==1&& ir[5]==1)
{
  motor(100,100);
  sensor_read();
}
motor(0,0);
delay(50);
if(ir[0]==1&& ir[7]==1)
{
  motor(70,70);
  delay(100);
  while(ir[3]==0&& ir[4]==0)
  {
    motor(100,-100);
    sensor_read();
  }
}
```

```
}
}





void calibrate()
{
  // This function is used for sensor calibration automatically.
  // Here the sensors of the LFR reads all the value and Takes max and min
threshold of the sensors
  // The set a threshold value for sensor.. this function is not completed yet..
  // Setting threshold from the calibration function is not done..
  // You have to set threshold level on sensor_print function just after analogRead
like::
  // ir[i] = analogRead(i)< threshold
  digitalWrite(13,HIGH);

  while(millis()<500)
  {
    motor(90,-90);
  for(int i=0; i<8; i++)
  {
  ir[i]=analogRead(i);
  if(ir[i]>sensorMax)
  sensorMax=ir[i];
  }
  }
  while(millis()<1500)
  {
    motor(-90,90);
  for(int i=0; i<8; i++)
  {
    ir[i]=analogRead(i);
  if(ir[i]<sensorMin)
  sensorMin=ir[i];
  }

  }
  motor(90,-90);
  delay(500);
  motor(0,0);
  Serial.println("Sensor Max");
  Serial.println(sensorMax);
   Serial.println("Sensor Min");
  Serial.println(sensorMin);
  digitalWrite(13,LOW);
 }
void STBY() /// This function is only for FNG Motor drivers where setting STBY
pin HIGH of the driver
            // set the motor driver in operation mode.
{
 button=digitalRead(10);
 if(button)
 {
 stby=1;
 digitalWrite(6,stby);
 }
 else digitalWrite(6,0);

}
```

Application:

- **Industrial Applications**: These robots can be used as automated equipment carriers in industries replacing traditional conveyer belts.
- **Automobile applications**: These robots can also be used as automatic cars running on roads with embedded magnets.
- **Domestic applications**: These can also be used at homes for domestic purposes like house keeper etc.
- **Guidance applications**: These can be used in public places like shopping malls, museums etc. to provide path guidance.

Discussion:

We noticed that, the black track must be deep dark for the robot to follow. Also, we had to adjust our code again and again to speed up the motor.