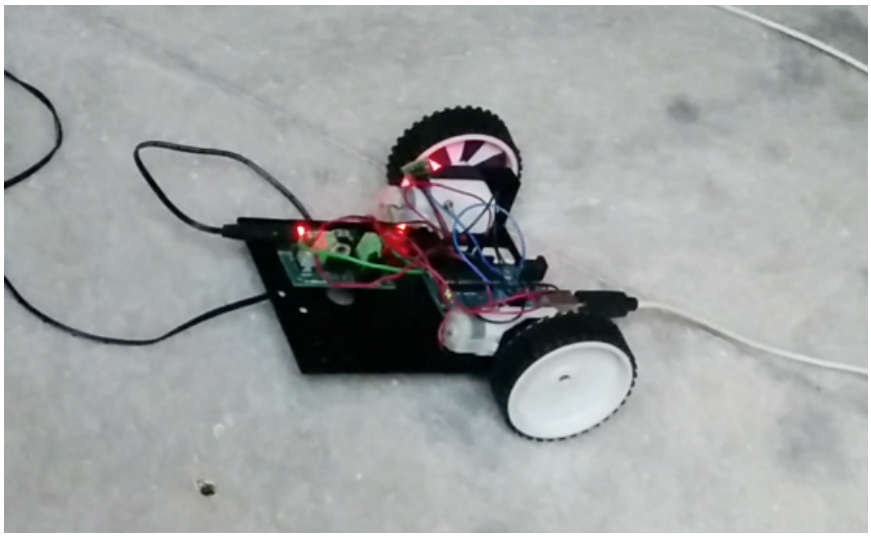
GO-TO-GOAL-BOT

(AUTONOMOUS DEFFIRENTIAL DRIVE WITH WHEEL ENCODER)



Skanda S Bharadwaj
Sumukha B.N
Chandan R Kumar
Sughosh HK

PES Institute of Technology

Abstract:

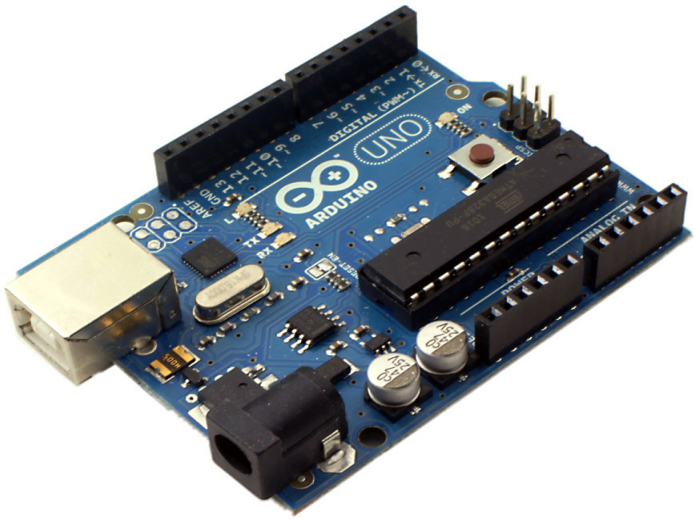
Gotogoal bot is an autonomous differential drive bot, which is designed to reach a specified particular destination. The bot is made intelligent enough to understand its environment when the initial conditions are specified. When the initial and the final positions are specified the bot follows the Pythagoras algorithm to reach its destination. The plant uses PD controller to minimize the error and to govern the differential drive.

Introduction:

Gotogoal autonomous differential drive bot uses wheel encoder to measure the amount of distance travelled. Based on this, the bot travels to the specified point or the destination. The wheel is encoded and an IR sensor is used to read the encoded value and this is fed to the controller, which will control the actuator movement, in this case, it's the motors. Considering the bot as a plant a controller is installed in order to process the feedback obtained. PD controller is used in this to process the feedback. Encoder is used to fetch the feedback. At every instant the distance travelled is calculated and compared with the original value. Based on this error, the controller governs the plant. The bot uses Pythagoras algorithm to serve the purpose. This algorithm involves two major calculations. First, the angle through which it has to rotate and second, the linear distance it has to travel in order to reach the final position. Based on these calculations the bot reaches its destination.

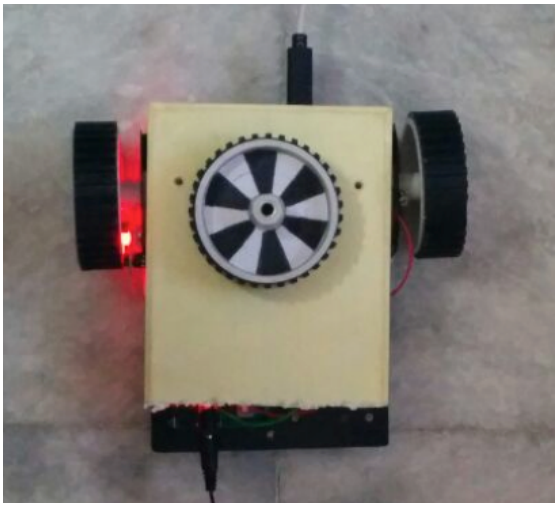
Bot anatomy:

1. Arduino-UNO: A single board micro-controller based on ATMEGA328

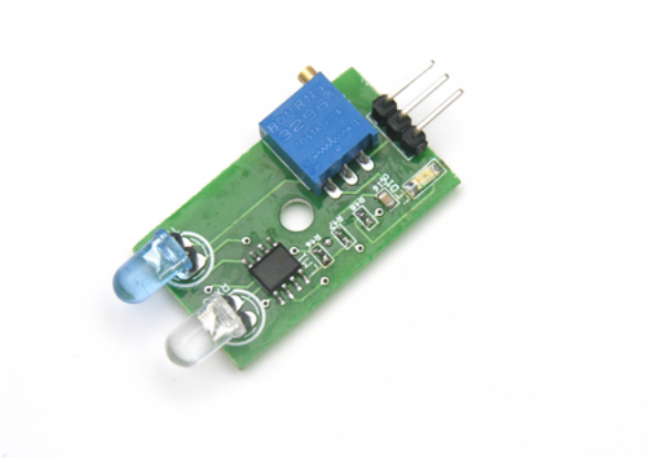


2. H-Bridge: motor driving circuit with L293D with input ranging form 5-36v.

3. Encoded wheels- Regular wheel but encoded using strips of black and white



4. IR-Sensor



Resolution of the bot:

- Linear movement resolution: 1.8cm
- Angular resolution: 15'

Working Procedure:

The robot should be informed about its initial position so that it will understand it's surrounding. The liberty to place the bot anywhere in the co-ordinate system is given to the user but the constraint is that; it has be placed parallel to the co-ordinate axes. Also, the flexibility of placing the bot has been extended in code using a variable for position (pos).
Position 1 - Parallel to x-axis along positive direction.
Position 2 - Parallel to y-axis along positive direction.
Position 3 - Parallel to x-axis along negative direction.
Position 3 - Parallel to y-axis along negative direction.

After specifying these initial conditions the bot has to be given with its destination. The initial points are (x1, y1) and the final points are (x2, y2). The bot first calculates the angle of rotation based on the initial condition and the corresponding distance to be traversed to complete that particular angle. The wheels are encoded in order to calculate these measurements. The Ir sensor that is present will send a notice to the controller as and when the focus changes from black to white or vice - versa. Based on these counts and the dimensions of the wheel the resolution of the bot is calculated. Depending on the distance to be travelled the resolution is multiplied with a corresponding factor. The bot turns around through the calculated angle. It then, calculates the amount of linear distance to be travelled. This will be shortest distance between the two given points. Similar to the above explanation the bot will move through the calculated distance to reach its destination. The angle and the distance is calculated using the following formulae

$$\theta = tan^{-1}\left(\frac{y2-y1}{x2-x1}\right)$$

$$\text{Distance} = \sqrt{(x2-x1)^2 + (y2-y2)^2}$$

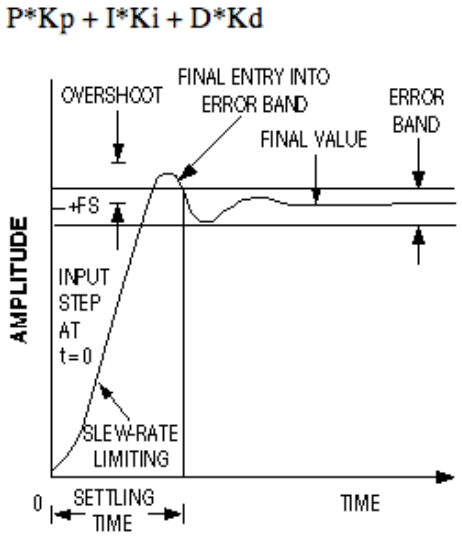
PD Controller:

The PD controller governs the bot. This controller is nothing but a few lines of code that minimizes the error in reaching the destination. PD controller was used here. p stands for proportional controller and D for derivative controller. Proportional controller is the continuous error-checking variable at every instant. a particular set point is fixed. Based on that point the positional error of the bot is calculated. The set point in this case is the calculated amount of distance to be travelled. The difference between this set point and the current amount of distance travelled is the error. For linear movement only proportional controller is sufficient. But for the rotation both proportional and derivative were introduced. Since the rotation will cause oscillations about the line of linear motion derivative controller was introduced. Derivative controller can be explained as the difference of rate of error at the current position and the previous position. This will reduce the oscillation or in other words the settling time is reduced and the bot will be stabilized soon enough. Basically, PD controller will process the feedback from the output and governs the speed of the actuators accordingly. More the error higher is the speed to reach the goal. Lesser the error lower is the speed in order to reduce the effect of the inertia of the body due to acceleration. Controller algorithm is as follows

```
error = traversal_distance-current_distance;
error2 = error1;
errort1 = error;

pd =kp*error +kd*(error2-error1)*(16/pow(10,6));
pwm = pd;
```

The pwm is controlled by the variable pd. Kp and kd are the proportional and derivative constants. Pwm is fed to the actutaors. The figure below gives a clear picture on PID controller.



Conclusion:

The bot is autonomous with differential drive and also intelligent enough to understand the environment. The specified goal is reached with minimum error.

Code:

```
//variables
int x;
int temp =0;
int count=-1;
double errort1;
double errort2;
double traversal;
double traversal_turn;
int pwmt=40;
//without turning/
int mu=60;// previous value= 180,160,170,160,60,PROPER=90/80 even 0 worked,:
for turn 80/75/70
//with turn//
//int mu=5;//prev val=10;
int pwm_v=0;// previous value =0,-5,-10;
//without turn//
double kp=20;// previous value = 10,20,21,22,23,24 PROPER=22/23/(21,20): for
turn 23/24/20
//with turn//
//double kp=21;//previous value=14,12;
double kd=0;

int mut=25;// previous value=20,10
int pwm_vt=0;// previous value=0
int tkp=21;// previous value=22,10 mostly proper

double cnt;
double theta;
int pwm;

double distance1=0;
double distance2=0;

int  x2=0;
int  y2=-65;
int  x1=0;
int  y1=0;
int pos=1;
```

```
//variables end

void setup()
{
 pinMode(7, INPUT);
 pinMode(8,OUTPUT);
// pinMode(9, OUTPUT);
 //pinMode(10,OUTPUT);
 pinMode(11, OUTPUT);
 Serial.begin(9600);

 angleofrotation(pos);
}


//main

void loop()
{
 x= digitalRead(7);

 if(theta!=0)
 {

  if(distance2<=traversal_turn)
  {
 //   pid_turn();
    turn(theta);
     Serial.print(" tr: ");
 Serial.print(traversal_turn);
 Serial.print(" d2: ");
 Serial.println(distance2);
  }
  else
  {
     pid_fwd();
       go();
  }
 }
 else
 {
    pid_fwd();
    go();
 }
```

```
 //printt();
  //delay(10);
}

//main end


//calculation of go distance
int calculate_go()
{

  if (temp != x)
  {
    count++;
  }

  if (count==-1) count=0;
  temp = x;
  distance1=0;
  distance1 = ((double)count/12)*22.7;

  return(distance1);
}
//calculation end

//calculation of turn distance
int calculate_turn(int initdist)
{

  distance2=(double)initdist;
  if (temp != x)
  {
    count ++;
  }

  if (count==-1) count=0;
  temp = x;

  distance2 = ((double)count/12)*22.7;

  return(distance2);
}
//calculation end
```

```
//go
int go()
{

 traversal=sqrt((pow((x2-x1), 2))+(pow((y2-y1), 2)));

 //distance1=0;
 calculate_go();

 if((distance1+distance2)<traversal)
  forward();
 else
 {
 // backward();
// delay(1);
  stopp();
 }
}




//turn bot
double turn(double degree)
{
 cnt=degree/15;

 traversal_turn=abs(((double)cnt/12)*22.7);

 calculate_turn(0);

 if(distance2<abs(traversal_turn))
 {
  if(cnt>0)
    left();
  else if(cnt<0)
    right();
  else stopp();
 }
 else stopp();
}
//end turn
```

```
//angle calculation
double angleofrotation(int p)
{
 double angle=abs((atan2((y2-y1),(x2-x1)))*(180/3.14159));

 if    (x2>x1 && y2>y1)  theta=angle;
 else if(x2>x1 && y2<y1)  theta=-angle;
 else if(x2<x1 && y2>y1)  theta=angle;
 else if(x2<x1 && y2<y1)  theta=angle+180;
 else if(y1==y2 && x2>x1) theta=0;
 else if(y1==y2 && x2<x1) theta=180;
 else if(x1==x2 && y2>y1) theta=90;
 else if(x1==x2 && y2<y1) theta=-90;

 if    (p==1) theta=theta;
 else if(p==2) theta=theta-90;
 else if(p==3) theta=theta-180;
 else if(p==4) theta=theta+90;

 return (theta);

}


//print statements
int printt()
{
 Serial.print("angle: ");
 Serial.print(theta);
 Serial.print(" tt: ");
 Serial.print(traversal_turn);
 Serial.print(" d2: ");
 Serial.print(distance2);
 Serial.print(" tr: ");
 Serial.print(traversal);
 Serial.print(" d1: ");
 Serial.println(distance1+distance2);
}
//prtint end


//diections
int forward()
{
 digitalWrite(8,HIGH);
 analogWrite(9,255-pwm);
```

```
 analogWrite(10,pwm);
 digitalWrite(11,LOW);
}

int stopp()
{
 digitalWrite(8,LOW);
 analogWrite(9,0);
 analogWrite(10,0);
 digitalWrite(11,LOW);
}

int backward()
{
 digitalWrite(8,LOW);
 analogWrite(9,30);
 analogWrite(10,225);
 digitalWrite(11,HIGH);
}

int right()
{
 digitalWrite(8,HIGH);
 analogWrite(9,255-pwmt);
 analogWrite(10,255-pwmt);
 digitalWrite(11,HIGH);
}

int left()
{
 digitalWrite(8,LOW);
 analogWrite(9,pwmt);
 analogWrite(10,pwmt);
 digitalWrite(11,LOW);
}
//directions end

void pid_fwd()
{
 double error=0;
 double v;

 error = traversal-distance1;
 //Serial.print(error);
 double p =kp*error;
// Serial.print(p);
```

```
  //optimal lower =25
  v= map(p,0,kp*traversal,45,70);// previous range : 10,150 / (45,100//not so
perfect),

  if (p<mu && v>= 45)
  {
     pwm=0;
   //Serial.println(v);
  }
  else if(p>=mu)
  {
  pwm=v-pwm_v;

 // Serial.println(v);
  }
}
void pid_turn()
{
  double error=0;
  double v;

  error = traversal_turn-distance2;
  errort2=errort1;
  errort1=error;

  double p =tkp*error;//+kd*(errort2-errort1)*(16/pow(10,6));

  v= map(p,0,tkp*error,40,70);//optimal lower=25

  if (p<=mut && v>= 40)
  {
    pwmt=0;
   //Serial.println(v);
  }
  else if(p>mut)
  {
  pwmt=v-pwm_vt;

  //Serial.println(v);
  }
}
```