PID Control of Line Following Robot

Ankit Singh
213070029

1 AIM

Design and implement a PID controller for the Spark V robot to make it follow a continuous track on the ground, using the IR sensors provided on the robot for this purpose.

2 OBJECTIVE

To trace the given track within 30 seconds.

3 EQUIPMENT USED

- SPARK V BOT
- A-B cable
- USB ASP AVR Programmer
- Charger
- Flat ended screwdriver

4 PROCEDURE

- The bot has to be charged (at least 2 hours) completely before commencing the experiment.
- The code for IR LED calibration is burned into the bot and it is placed in the track and a screwdriver is used to rotate the 3 potentiometers on the bot and calibrate the 3 respective IR LEDs.
- The calibration is done in such a way that the IR LED values displayed in the LCD panel of the bot show equal values for all 3 LEDs (in both white and black region).
- Once the calibration is finished, the PID logic is implemented in the bot and the PID gain values are tuned till the curves are smoothly traced.

- The PID output is used to control the velocity of each wheel of the bot using the Velocity function, where PWM signals which have duty cycle proportional to the PID output are generated. These PWM signals are given to the control pins of themotor driver IC.
- The gain values have to be adjusted to boost the speed of the bot, in-order to traverse the track within 30 seconds.
- Thus the track can be traced within the specified time.

5 CODE

```
#include<avr/io.h>
#include<avr/interrupt.h> #include<util/delay.h>
void motion_pin_config (void)
{
DDRB = DDRB | 0 x0F ;
//setdirectionofthePORTB3toPORTB0pinsasoutput PORTB
= PORTB & 0 xF0 ;
//setinitialvalueofthePORTB3toPORTB0pinstologic0 DDRD =
DDRD | 0 x30 ;
//SettingPD4andPD5pinsasoutputforPWMgeneration
PORTD = PORTD | 0 x30 ;
//PD4andPD5pinsareforvelocitycontrolusingPWM
}
void motion_ set ( unsigned char Direction)
{
unsigned char Port BRestore = 0; Direction
&= 0 x0F;
//removinguppernibbleasitisnotneeded Port
BRestore = PORTB;
//readingthePORTB'soriginalstatus Port
BRestore &= 0xF0;
//settinglowerdirectionnibbleto0 Port
BRestore |= Direction;
//addinglowernibblefordirectioncommandandrestoringthePORTB status
PORTB = Port BRestore;
//settingthecommandtotheport
}
void adc_init()
{
```

```
ADCSRA = 0 x00;
ADMUX = 0 x20;//Vref=5Vexternal---ADLAR=1---MUX4:0=0000 ACSR = 0
x80;
ADCSRA = 0 x86;//ADEN=1---ADIE=1---ADPS2:0=110
}
//ADCpinconfiguration
void adc_pin_config (void)
{
DDRA = 0 x00;//setPORTFdirectionasinput PORTA
= 0 x00;//setPORTFpinsfloating
//FunctiontoInitializePORTS void port_
init()
{
adc_ pin_ config ();
motion_pin_config();
}
//TIMER1initialize-prescale:64
//WGM:5)PWM8bitfast,TOP=0x00FF
//desiredvalue:450Hz
//actualvalue:450.000Hz(0.0 ) void
timer1_init(void)                    %
{
TCCR1B = 0x00;//stop TCNT
1H = 0xFF;//setup TCNT 1 L
= 0 x01 ; OCR 1 AH = 0 x00 ;
OCR 1 AL = 0 xFF ; OCR 1 BH
= 0 x00 ; OCR1BL = 0xFF;
ICR1H  =  0  x00  ;
ICR1L  =  0  xFF  ;
TCCR1A = 0xA1;
TCCR1B = 0x0D;//startTimer
}
//ThisFunctionacceptstheChannel Number and
returns the corresponding Analog  Value
unsigned char ADC_ Conversion( unsigned char Ch)
{
unsigned char a; Ch =
Ch & 0 x07 ; ADMUX= 0
x20| Ch;
ADCSRA = ADCSRA  | 0 x40 ;//Setstartconversionbit while((ADCSRA&0
x10)==0);//WaitforADCconversiontocomplete a=ADCH;
ADCSRA = ADCSRA|0x10;//clearADIF(ADCInterruptFlag)bywriting1toit return a;
}
```

```
void forward (void)//bothwheelsforward
{
  motion_set(0x06);
}
void back (void)//bothwheelsbackward
{
  motion_set(0x09);
}
void left (void)
//Leftwheelbackward,Rightwheelforward
{
  motion_set(0x05);
}
void right (void)
//Leftwheelforward,Rightwheelbackward
{
  motion_set(0x0A);
}

int ld,rd;
void velocity (int left_motor,int right_motor)
//velocityfunctionforcontrolling the
speed of the motors using PWM signal
{
if(left_motor>0)
//speedoftheleftmotoriscontrolled using input
received in the velocity function
{
OCR1AH = 0x00;
OCR1AL = (unsigned char)left_motor; ld=0;
}
elseif( left_motor < 0)
//aflag"ld"isusedtosavethedirection of the
motor (i. e; forward or back)
{
left_motor=left_motor*(-1); OCR1AH =
0x00;
OCR1AL = (unsigned char)left_motor; ld=1;
}
if(right_motor>0)
//speedoftherightmotoriscontrolled using
input received in the velocity function
```

```
{
OCR1BH = 0x00;
OCR1BL = (unsigned char)right_motor; rd=0;
}
elseif(right_motor<0)
//aflag"rd"isusedtosavethedirection of the
motor ( i. e; forward or back)
{
right_motor=right_motor*(-1); OCR1
BH = 0x00;
OCR1BL = (unsigned char)right_motor; rd=1;
}
if((rd==0)&&(ld==0))
{
forward();
}
elseif((rd==1)&&(ld==1))
{
back();
}
//thevaluesof"ld"and"rd"arechecked
to determine the direction to move elseif((rd==0
)&&(ld==1))
{
left();
}
elseif((rd==1)&&(ld==0))
{
right();
}
}

void init_devices (void)
{
cli();//Clearstheglobalinterrupts port_init();
adc_ init (); timer1_
init();
sei();//Enablestheglobalinterrupts
}
#definethresh40
//thethresholdissetsuchthat ifsensor
is in the black line its value is > 40 #definelowthresh25
```

```
//thelowerthresholdissetsuchthat
ifthe sensor is in the white line its value
is < 25 char  l,
c,r;
//variableswhichstoretheIRsensor value
int L,C,R;
//flagtochecckthesensorpostion
( black = 0 , white = 1 : left , centre int error ;
//storeserrorforproportionalpart int Pre
Error;
//storeserrorforderivativepart int                                    or   right)
Totalerror ;
//storeserrorforintegralpart int Lr;
//variabletocontrolthebotincaseof overshoot




float kp = 0.1;
float kd = 1; float  ki =
0.0001;float PID;
int main()
{
init_devices(); while(1)
{
l = ADC_ Conversion(3);
//GettingdataofLeftWLSensor c =
ADC_ Conversion(4);
//GettingdataofCentreWLSensor r =
ADC_ Conversion(5);
//GettingdataofRightWLSensor
//left     if(l>=
thresh)
{  L
=1;
}
elseif(l<lowthresh)
{  L
=0;
}
//centre   if(c>=
thresh)
{//thesensorvaluesarecomparedwiththe threshold values to
assign the flags L, C, R
, which is further usedformotion control
```

```
C=1;
}
elseif(c<lowthresh)
{  C
=0;
}
//right    if(r>=
thresh)
{  R
=1;
}
elseif(r<lowthresh)
{  R
=0;
}                if((R==0)&&(L==0
)&&(C==0))
{
if(Lr==0)
{
velocity(250,-60);
//Thebotismovedtotheright
because of  overshoot
}
elseif(Lr==1)
{
velocity(-60,250);
//Thebotismovedtotheleft
because of  overshoot
}
}
else
{    if(R
==1)
{  Lr
=0;
//ifbotmovedtotheleftduetoovershoot, Lr is
assigned aszero
}
elseif(L==1)
{  Lr
=1;
//ifbotmovedtotherightduetoovershoot, Lr is
assigned aszero
}
error = ( l - r); Totalerror += error ;
PID = kp*error+ kd*(error-PreError) +ki*Totalerror;
//thepidoutputiscalculated
```

```
Pre Error = error;
if(((C==1)&&(R==0)&&(L==0))||((C==1)&&(R==1)&&(L==1)))
{
velocity(255,255);
//thebotmovesforwardifthemiddle sensor
is in the black line , orif all sensors are in
black
}
else
{
velocity(150-PID,150+PID);//
forother cases it performs the PID action
}
}
}
}
```