AIN SHAMS UNIVERSITY
FACULTY OF ENGINEERING
MECHATRONICS ENGINEERING DEPARTMENT

Spring 2022

MCT 333: Mechatronic System Design

PID Control on Motor using Microcontroller (Discrete Time Control System)

- ## Lab Objectives
  - Discrete Approximation of PID Controller Algorithm.
  - Understanding How to handle an Encoder using a microcontroller.
  - Measuring Speed (RPM) of DC Motor.
  - Applying PID Controller using a microcontroller.

- ## Lab Requirements
  - Arduino Uno.
  - DC Motor with integrated Magnetic Encoder.
  - DC Motor Driver.
  - 6-12V Power Supply
  - 10k Pot

- ## Discrete Approximation of PID controller Algorithm

  A PID (Proportional Integral Derivative) is a linear controller that calculates an 'error' value as the difference between a measured Input and a desired setpoint. The controller attempts to minimize the error by adjusting an Output. It is used in industrial control applications to regulate temperature, flow, pressure, speed and other process variables. And is considered one of the most stable controllers.

  - PID Mathematical Formula

$$U(t) = K_p\, e(t) + K_i \int_0^{t} e(t) \quad + K_d \frac{dd}{ddtt} e(t)$$

Where:

e(t): Error between Measured action and Desired setpoint.

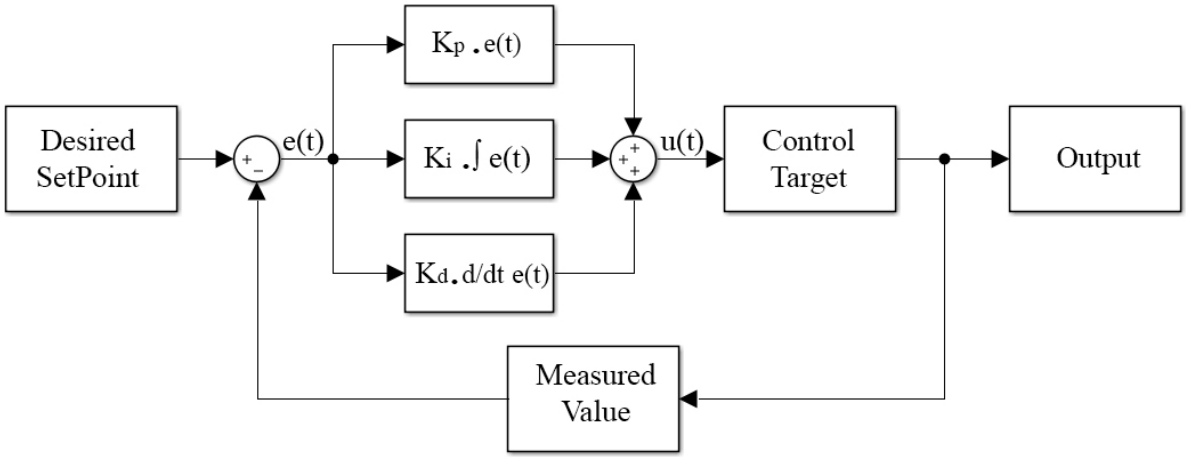
K_p: Proportional Constant Value.

K_i: Integral Constant Value.

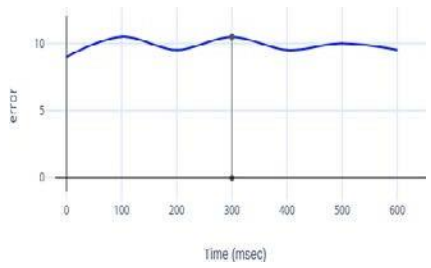
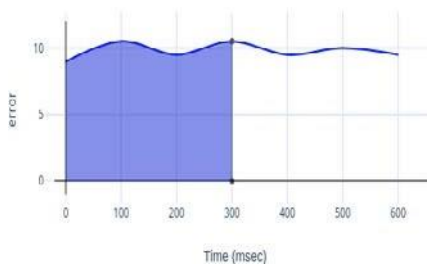
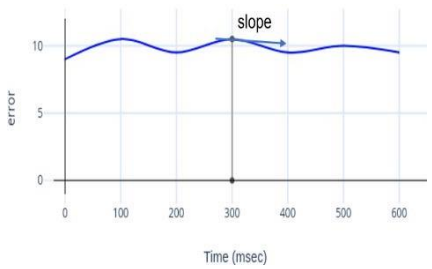
K_d: Derivative Constant Value.

U(t): Output Control Signal.

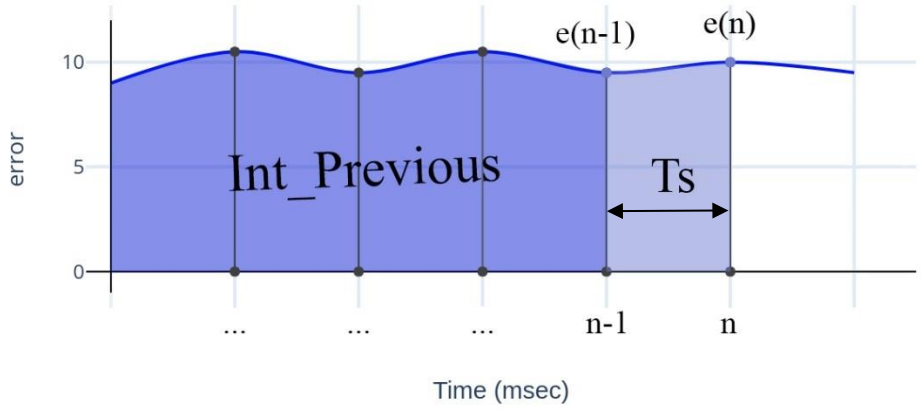
- PID Block Diagram



- PID is a Linear Controller

		
Present	**History**	**Future**
K_p: Accounts for present values of the error (e.g. if the error is large and positive, the control variable will be large and negative)	K_i: Accounts for past values of the error. (e.g. if the output is not sufficient to reduce the size of the error, the control variable will accumulate over time, causing the controller to apply a stronger action.)	K_d: Accounts for possible future values of the error, based on its current rate of change.
K_p: The BIGGER the number the harder the controller pushes.	K_i: the LARGER the number, the larger the settling time.	K_d: the BIGGER the number the more the controller dampens oscillations (to the point where performance can be hindered).
$X_p = K_p\, e(t)$	$X_i = K_i \int_0^{tt} e(t)$	$X_d = K_d \frac{dd}{ddtt} e(t)$

- Implementing PID Controller (PID Discretization)



- $X_p(n) = K_p . e(n)$
- $X_i(n) = K_i . (Area) = K_i . \left(\frac{TTTT * [ee(nn) + ee(nn-1)]}{2} + Int\_previous\right)$
- $X_d(n) = K_d . (Slope) = K_d . \left(\frac{ee(nn) - ee(nn-1)}{TTTT}\right)$
- Therefore:

$$X(n) = K_p . e(n) + K_i . \left(\frac{TTTT * [ee(nn) + ee(nn-1)]}{2} + Int\_previous\right) + K_d . \left(\frac{ee(nn) - ee(nn-1)}{TTTT}\right)$$

$$= \left[K_p + \left(\frac{KK_d}{TTTT}\right) + \left(\frac{KK_i . Ts}{2}\right)\right] . e(n) + \left[\left(\frac{-KK_d}{TTTT}\right) + \left(\frac{KK_i . Ts}{2}\right)\right] . e(n-1) + K_i . Int\_previous$$

$$= A . e(n) + B . e(n-1) + C . Int\_previous$$

  - $A = \left[K_p + \left(\frac{K_d}{TTTT}\right) + \left(\frac{K_i . Ts}{2}\right)\right]$
  - $B = \left[\left(\frac{-K_d}{TTTT}\right) + \left(\frac{K_i . Ts}{2}\right)\right]$
  - $C = K_i$

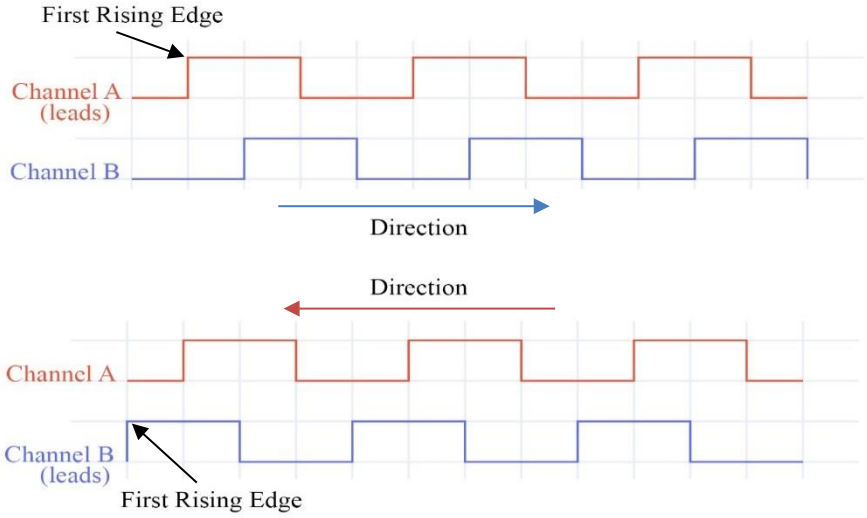
Where:

- e(n): Error between Measured action and Desired setpoint at a certain time.
- X(n): Output Control Signal.
- Ts: Sampling Time
- Int_previous: Summation of Area from previous errors.

- Pseudo Code for PID Discretization

```
/*
 * Applying PID Controller on the velocity of a motor
 * Assuming there is a function "readVelocity()" which
 * Which handles the input from the encoder and return
 * the measured Velocity.
 */
//Parameters Initialization
    Kp = …;
    Ki = …;
    Kd = …;
    Ts = …;
    A = [Kp + (Kd/TTTT) + (Ki.Ts/2)];
    B = [(−Kd/TTTT) + (Ki.Ts/2)];
    C = Ki;
    e(n) = 0;
    e(n-1) = 0;
    int_Previous = 0;
    ωω_desired = 50; //RPM
    ωω_actual = 0;

// Continuous loop
    void loop () {
        ωω_actual = readVelocity();
        e(n) = ωω_desired –
        ωω_actual;
        X(n) = A. e(n) + B. e(n-1) + C. int_Previous;
        analogWrite(…, X(n));
        int_Previous = int_Previous + ((TTTT*[ee(nn)+ee(nn−1)])/2);
        e(n-1) = e(n);
    } // end void loop
```

- **Using Two Channel Encoders**
  - **To determine Motor Rotation Direction:**



  - **To determine Motor Rotation Speed:**

$$\text{Speed in RPM} = \left(\frac{\text{EEnnEEEEddeeEE aaEEttaaaaaa CCEEaannteeddPPaaaaTTeeT T}}{\text{EEnnEEEEddeeEE PPaaaTTeeTT ppeeEE MMEEttEEEE SShaaaatt RREEttaattttEEnn}}\right) * \left(\frac{1000.0}{\text{CCEEnnTTttaannttt TTttTTee llnnteeEEllaaaa}}\right) * 60.0$$

Ex 1: Reading Direction and Velocity using an Arduino

- download “SimpleTimer.h” library, https://github.com/jfturcot/SimpleTimer
- Using an 10k potentiometer as analog input to change the speed of the motor manually.
- Attach the two channels of the encoder to pin 2 and 3 in your Arduino using them as interrupts.

```
/*
 *  To Read Velocity or To Use PID Controller Correctly, a constant Time
 *  Interval is required, using SimpleTimer Library for arduino allows
 *  to creat a timer interrupt, the library download link >>
 *  https://github.com/jfturcot/SimpleTimer
 */
#include <SimpleTimer.h>

//Defining the pins used on the Arduino
#define in1 6
#define in2 9
#define EncoderPinA 2 //Encoder Channel A
#define EncoderPinB 3 //Encoder Channel B
#define PotValue A0

//Analog Pot Value
int PotVal = 0;
//PWM control signal for motor
unsigned char pwmOutput = 0;

//Encoder Counter
volatile signed long encoderCount = 0;
//Encoder Pin B pervious Value
bool perviousPinB = LOW;
//Encoder Pulse Per Gearbox output Revolution
const float encoderPPR = 2150.0;
```

```
//Speed in Revolution per Minutes
float SpeedRPM = 0.0:

//Timer Objecr
SimpleTimer timer;
//Timer interval length in 100 ms
coast int timerInterval = 100L:

void setup() {
  pinMbde(inl, OUTPUT);
  pinMbde(in2, OUTPUT):

  pinMbde(AnooderPinA, I¥PUT);
  pinMbde(KnooderPinB, I¥PUT);

  // attalinQ diQital pin S >> whil is referenoed by digit (0)
  //as am znterrupt for the First Anooder pin A
  attashInterrupt(0, AnooderEvent, CHANGA);

  /'FDr arduino uno there are two external interrupt pins,
   * which allows us to use upto two encDders
   * attaching digital pin 3 » which is referenced by digit (1)
   * as an interrupt lor the Second 2ncoder pin A
   '/
  attachInterrupt(1, AnooderEvent, CHANGR):

  Z/Setting Timer to read botor Speed every 100 m5ec
  cimer.setInterval(cimerIxterval, timerRoutine):

  Ssrlal.begin(9600):
}

flint fimxderEvent({
  /*
    In This Lab, using an integrated 2-channel Magnetic BncDder with 49 PPR, which operates as follows:
     ______   ______   ______   ______

   *      _______   _______   _______   _______      Encode r Pin A
       ______   ______   ______   ______

   '
   * ___|     ______   ______   ______   _______     Encoder Pin B
   *
   * tD determine bDth speed and directiDn oI the motDr, it is possible to use one channel as an interrupt,
   * but it is more accurate to use both channels as interrupts,
   * iI decide to use the two channels as interrupts use the following code:
   /
  if(digitalRead(AnooderPinA) HIGH){
    if(digitalRead(AnooderPlñB) LOH cc pervlousPlnB LCTY){
      // clockwise direction
      emxAerCount++;
      //mororDirecriou = 'CY':
      pervlousPlnB = LDH;}
    eLse if(digitalRead(AnooderPinB) HIGH z* pervlousPlnB LOH){
      // clockwise direction
      emxderCoust++:
      //motorDirection = '€Y';
      pervlousPirJB = HIGH:}
```

```
 9SEJ    else if(digitalRead(AnooderPinB)     LOH ** perviousPinB     HIGH){
 9f        // oounter-cloc)cwise direction
 97        enooder€ount--;
           //notorDirecriox ='CCH':
 9?        perviousPinB = LOH;}
 90EJ    else if(digitalRead(AnooderPinB)     HIGH ** perviousPinB     HIGH)(
 ?1        // oounter-clockwise direction
 ?2        encodexCounc——T
 ?3        //notorDirectiox ='CCH':
 ?4        perviousPinB = HIGH;}
 95    }//end if(digitalRead(AnooderPinA)     HIGH)
 ?flE  else{ /Zif(digitalRead(AnooderPinA)     LON)
 9753    if(digitalRead(AnoOderPinB)     LOH ** perviousPinB     POH){
 ?9        // oounter-clockwise direction

100        //eororDirecrion ='CCH':
101            iou      = LOH:}
10253    else if(digitalRead(AnooderPinB)     HIGH ** perviousPinB     LOH){
103        // oounter-clockwise direction
104        enooderCount--:
105        //eororDirecrion ='CCH':
10f            iou      = HIGH:}
107a     else if(digitalRead(AnooderPinB)     LOH *t perviousPinB     HIGH){
109        // clockwise direction
10?        enxMerCoust++:
110        //motorDirection ='CH';
111        perviousPinB = LOH:}
112-     else if(digitalRead(AnooderPinB)     HIGH ** perviousPinB     HIGH){
113        // cLockwise direction
114        enooderCount++:
115        ZZmotorDirection ='CH';
11f        perviousPinB = HIGH;}
117        }ZZend else (digitalRead(&nooderPinA)     LON)
119  }//&nooderEvent ISR
11?
120lEvoid timerRoutine(){
121    PotVaL = analogRead(PotVaLue);
lZZ    pwm0utput = map(PotVal, 0, 1023, 0, 2fifi);
123    analogVrite(inl, pnaoutput):
124    digitalVrite(in2,LOH);
lZ5    TmedRPE= (euooderGount/exooderPRW*(1000.0/timerIxterval*60.0;
12f    enooderCount = 0:
127    Serial.print('Speed RPM=');
lZ@    Serial.println(TmedRRfl;
128 }
130
13lEJvoid loop() (
132    ZZSpeed Pulling teclmique
133    timer.rnn(;
134 }
135
```

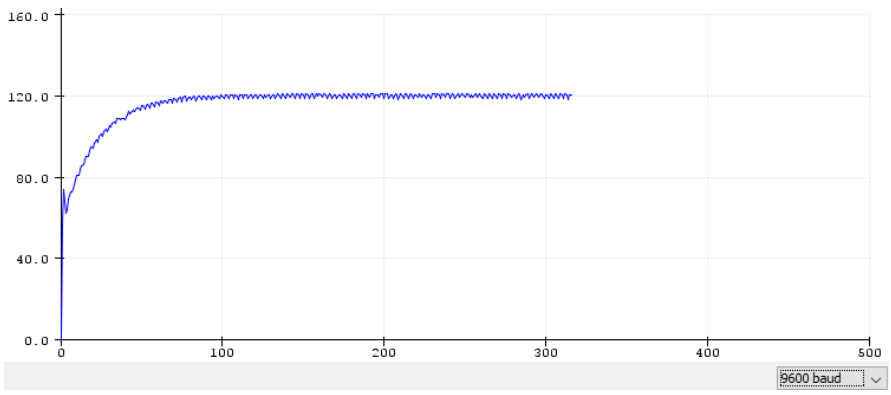
Ex 2: PID Library for Arduino

- download “PID_v1.h” library, https://github.com/br3ttb/Arduino-PID-Library
- Attach the two channels of the encoder to pin 2 and 3 in your Arduino using them as interrupts.
- Using the PID library with Constants Kp = 1, Ki = 1, Kd = 0.01 (without Tuning).
- Using the same EncoderEvent ISR and the same Velocity function.

```
/*
 *  To Read Velocity or To Use PID Controller Correctly, a constant Time
 *  Interval is required, using SimpleTimer Library for arduino allows
 *  to creat a timer interrupt, the library download link >>
 *  https://github.com/jfturcot/SimpleTimer
 */
#include <SimpleTimer.h>
/*
 *  Using PID Library for Arduino, the library download link >>
 *  https://github.com/br3ttb/Arduino-PID-Library
 */
#include <PID_v1.h>

//Defining the pins used on the Arduino
#define in1 6
#define in2 9
#define EncoderPinA 2 //Encoder Channel A
#define EncoderPinB 3 //Encoder Channel B

//Output PWM control signal for motor
double pwmOutput = 0;
//Input Speed in RPM
double SpeedRPM = 0.0;
//Required Speed in RPM
double Setpoint = 120;
//PID tuning parameters
float Kp = 1, Ki = 1, Kd = 0.01;
//Specify the initial parameters for PID
PID myPID(&SpeedRPM, &pwmOutput, &Setpoint, Kp, Ki,Kd, DIRECT);

//Encoder Counter
volatile signed long encoderCount = 0;
//Encoder Pin B pervious Value
bool perviousPinB = LOW;
//Encoder Pulse Per Gearbox output Revolution
const float encoderPPR = 2150.0;

//Timer Object
SimpleTimer timer;
//Timer interval length in 100 ms
const int timerInterval = 100L;

```

```
43  void setup() {
44    pinMode(in1, OUTPUT);
45    pinMode(in2, OUTPUT);
46    pinMode(EncoderPinA, INPUT);
47    pinMode(EncoderPinB, INPUT);
48
49    // attaching digital pin 2 >> which is referenced by digit (0)
50    //as an interrupt for the First Encoder pin A
51    attachInterrupt(0, EncoderEvent, CHANGE);
52    /*For arduino uno there are two external interrupt pins,
53     * which allows us to use upto two encoders
54     * attaching digital pin 3 >> which is referenced by digit (1)
55     * as an interrupt for the Second Encoder pin A
56     */
57    attachInterrupt(1, EncoderEvent, CHANGE);
58
59    //turn the PID on
60    myPID.SetMode(AUTOMATIC);
61    //Setting Timer to read Motor Speed every 100 mSec
62    timer.setInterval(timerInterval, timerRoutine);
63
64    Serial.begin(9600);
65  }
66
67  int EncoderEvent(){
126
127 void timerRoutine(){
128
129   SpeedRPM = (encoderCount/encoderPPR)*(1000.0/timerInterval)*60.0;
130
131   myPID.Compute();
132
133   analogWrite(in1, pwmOutput);
134   digitalWrite(in2,LOW);
135
136   encoderCount = 0;
137   Serial.print("Speed RPM = ");
138   Serial.println(SpeedRPM);
139 }
140
141 void loop() {
142   //Speed Pulling technique
143   timer.run();
144 }
```

- Motor Speed Graph on serial plotter from Arduino IDE. (Time axis scale: 100 ≈ 10 secs or 100* timerInterval)



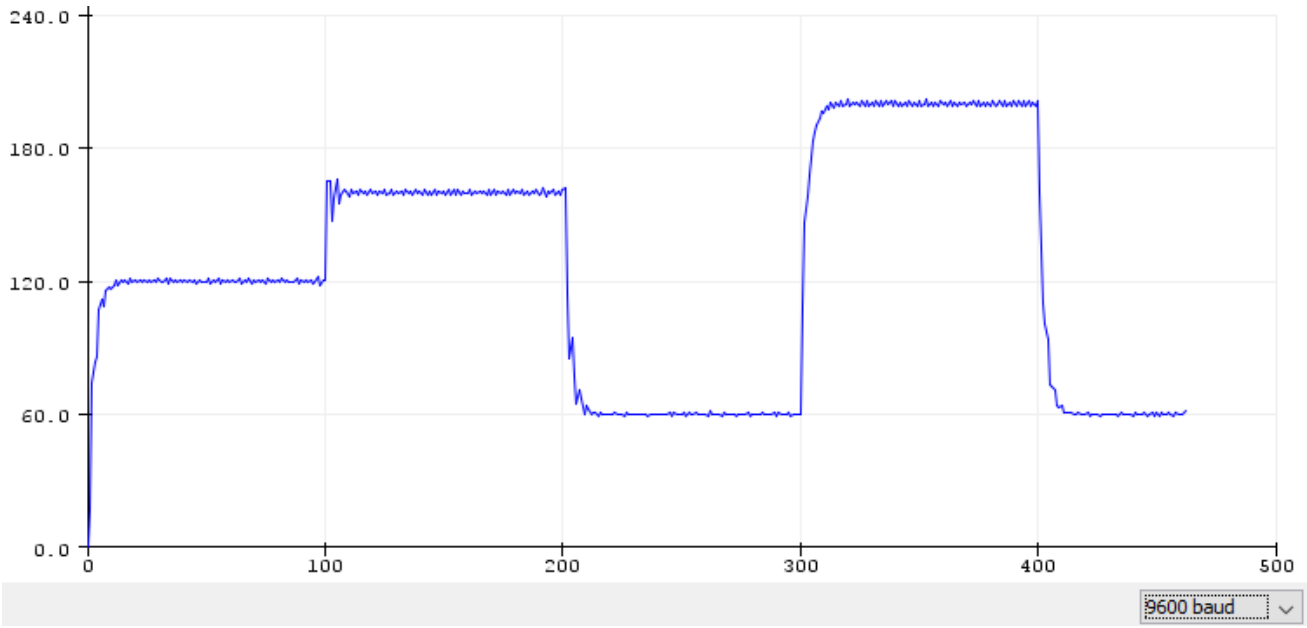
Ex 3: PID Library for any Microcontroller

- Attach the two channels of the encoder to pin 2 and 3 in your Arduino using them as interrupts.
- Writing PID controller Equation with Constants Kp = 2, Ki = 1, Kd = 0.01 (without Tuning).
- Using the same EncoderEvent ISR and the same Velocity function.
- Changing the Setpoint, to see the difference in output.

```
/*
 *  To Read Velocity or To Use PID Controller Correctly, a constant Time
 *  Interval is required, using SimpleTimer Library for arduino allows
 *  to creat a timer interrupt, the library download link >>
 *  https://github.com/jfturcot/SimpleTimer
 */
#include <SimpleTimer.h>

//Defining the pins used on the Arduino
#define in1 6
#define in2 9
#define EncoderPinA 2 //Encoder Channel A
#define EncoderPinB 3 //Encoder Channel B

//Timer Object
SimpleTimer timer;
//Timer interval length in 100 ms
const int timerInterval = 150L;

//Input Speed RPM, PWM Output, Setpoint RPM
float SpeedRPM = 0.0, Setpoint = 120.0;
unsigned char pwmOutput = 0;
unsigned char outMax = 255, outMin = 30;

int Setpointcount = 0;

//PID tuning parameters
float kp = 2.0, ki = 1.0, kd =0.01;
//Specify the initial parameters for PID
float iTerm = 0.0, dTerm = 0.0, prevIntegration = 0.0;
float currentError = 0.0, prevError = 0.0;

//Encoder Counter
volatile signed long encoderCount = 0;
//Encoder Pin B pervious Value
bool perviousPinB = LOW;
//Encoder Pulse Per Gearbox output Revolution
const float encoderPPR = 2150.0;

void setup() {
  pinMode(in1, OUTPUT);
  pinMode(in2, OUTPUT);
  pinMode(EncoderPinA, INPUT);
  pinMode(EncoderPinB, INPUT);
```

```
45
461a  /* attaching digital pin 2 >> which is re2erenced by digit (0}
47     * as an interrupt 2Dr rhe First 6CcDder pin A
48     * For arduino uno rhere are two external interrupt pins,
4S     * which allows us tD use upto two encDders
50     * attaching digital pin 3 >> which is re2erenced by digit (l}
51     * as an interrupt 2Dr rhe Second £CcDder pin A
52     */
53    atrachInrezrupt(0, &nooderRveut, CHAHG&);
54    atrachInrezrupt(1, &nooderRveut, CHAHG&);

56    //Setting Tiner ro read Moror Speed every 100 m5ec
57    timer.setInrerval(timerInrervaL, timexRoutine:
58
5S    SerLal.begin(9600):
60  }

62⊞int EncoderEvent(){
121
122
123f3void cimerRoucine()(
124
125     SpeedRPM = (enooderCount/enooderP2R)*(1000.0/ciserInterval)*60.0;
126     currentError = Setpoint - SpeedRPM;
127
129     17ezat = ki*  ( (c  zzxezztAxxox+pcevBxxox) /2  +  prevIztLegzac?on)
129     dTezm = kd“ (cuxxencExxoz—pxevZzxoz) /tIza=zIncexvaLT
130
131     pwm0utput = Ep*currenrArror+ iTem + dTerDr
132
133@-   if(paMutput  >  ourMax)(
134       m•amtPue = o°rMa•:I
135@-   else if(pwm0utput < ourHin){
136       p••d)utpuc = ourHin'}

139     analogWrlEe(inl, paMutput):
I35     digital@zite(in@,LON);
140
141     enoodezCoun€ = 0;
142     prevInreg*atlonw (kl * nurrenrKrroz);
143
144     Setpointcount++;
145     if(Setpointcount>= 100)
146       Secpoin€ = 180;
147
148     if(5etpoinroount>= 200)
149       SetpDin€ = 60:
150
151     if(Setpointoountw 300)
152       SeEpo zt£ = 200

154     if(Setpoinroount>= 400)
155       SetpDin€  =  60:
156
157     Serial.print("Speed RPM = ");
159     Serial.println(SpeedRPM);

160
161£3void Loop()  (
162     // Speed PuLL Mtg  Wcttn1gue

164   )
```

- Motor Speed Output Graph on serial plotter from Arduino IDE.



- **Integral/Integrator Windup**

  When using PID controller, the integrator term continues accumulating and increasing the controller output which results in:

  - The controller reaches the actuator physical limits, then the actuator becomes saturated and the system operates in open loop.
  - The controller output may become really large which results in large output overshoot.
  - The controller signal remains saturated even if the error begins to decrease, and windup induced lag is introduced to the system.

  As a solution, an anti-windup compensator that prevents the error from building up excessively in the integral term of the controller.