MICROCONTROLLER IMPLEMENTATION OF DIGITAL PID CONTROLLER

THESIS

Submitted in Partial Fulfillment of the Requirements for the Degree of

BACHELOR OF TECHNOLOGY in
ELECTRICAL ENGINEERING

By

SHAHRUKH ALAM
110EE0188

Under the supervision of

Prof. Sandip Ghosh

Department of Electrical Engineering
National Institute of Technology
Rourkela -769 008, Orissa, India.
2014

CERTIFICATE

This is to certify that the thesis entitled, "**Microcontroller Implementation of Digital PID Controller**" submitted by Shri **Shahrukh Alam** in partial fulfilment of the requirements for the award of Bachelor of Technology degree in Electrical Engineering at the National Institute of Technology, Rourkela, is an authentic work carried out by him under my supervision and guidance.

To the best of my knowledge the matter embodied in the thesis has not been submitted to any other University/Institute for the award of any degree or diploma.

Date:	Prof. Sandip Ghosh
Place: Rourkela	Department of Electrical Engineering
	National Institute of Technology Rourkela

ACKNOWLEDGEMENTS

I have been very fortunate to start my thesis work under the supervision and guidance of Prof. Sandip Ghosh. He introduced me to the field of Control systems, educated me with the methods and principles of research, and guided me through the details of PID controllers. He is the whole Philosopher and Guide behind this thesis. Working with him, a person of values has been a rewarding experience.

I am highly indebted and express my deep sense of gratitude for his invaluable guidance, constant inspiration and motivation with enormous moral support during difficult phase to complete the work. I acknowledge his contributions and appreciate the efforts put by him for helping me complete the thesis.

I would like to take this opportunity to thank Prof. A.K Panda, the Head of the Department for letting me use the laboratory facilities for my project work. I am thankful to him for always extending every kind of support to me.

At this moment I would also like to express my gratitude for the technical staff of our laboratories. They have always helped me in every-way they can during my experimental phase of the work.

ABSTRACT

A **proportional-integral-derivative (PID)** controller is widely used in industrial control systems to get the desired response by feedback. In this project, we attempt to implement a digital controller in a microcontroller. The primary difference between a digital controller and an analogue controller is that with a digital controller the actual value is not measured continuously, rather it is periodically sampled at some fixed time interval.

To study the issues in implementing a digital PID controller in Arduino microcontroller is the main objective of the project. Once the implementation issues are solved then one can tune the K_p, K_d and K_i gains of the PID controller. Based upon the error occurred and by changing the values suitably, the required output from the system can be obtained.

In this project, PID controllers with input and output features are implemented in Arduino and its frequency response is studied in order to adjudge whether the implementation is correct or not. Two methods are used for generating the PID controller output. One by using a properly tuned RC filter that filters out the PWM signal generated by the Arduino and the other by using a digital to analog converter from the digital output of the Arduino.

CONTENTS

LIST OF FIGURES AND TABLES

1. INTRODUCTION TO PROJECT:

The basic ingredients of a control system can be described by:

- Objectives of control
- Control system components
- Results or output

OBJECTIVE
CONTROL SYSTEM
RESULT

Fig. 1.1: BASIC INGREDIENTS OF A CONTROL SYSTEM

2. OBJECTIVE OF PROJECT:

In this project, the objective is to explore the capability of the Arduino by developing PID for any input signal (alternating or non-alternating, sinusoidal, triangular, rectangular or constant voltage signal) and to get desired output signal for any application (DC motor speed control, Intelligent tuning and other Industrial purposes) by controlling the parameters.

The project will include:

1. Supplying any arbitrary signal (Sinusoidal signal in this project) using Signal Generator
2. Voltage inversion using OPAMP (LM 324N in this project)
3. Knowledge of Arduino Uno R3 kit and Arduino programming environment
4. Developing the PID controller using Arduino Uno R3 and interfacing the Arduino hardware by dumping the program
5. Digital to analog conversion using R-C Filter
6. Digital to analog conversion using DAC (0808 in this project)
7. Observing the output in an Oscilloscope
8. Plotting the Lissajous curve between input (x-axis) and output (y-axis)
9. Calculating the Gain and Phase Margin between input and output from the plot

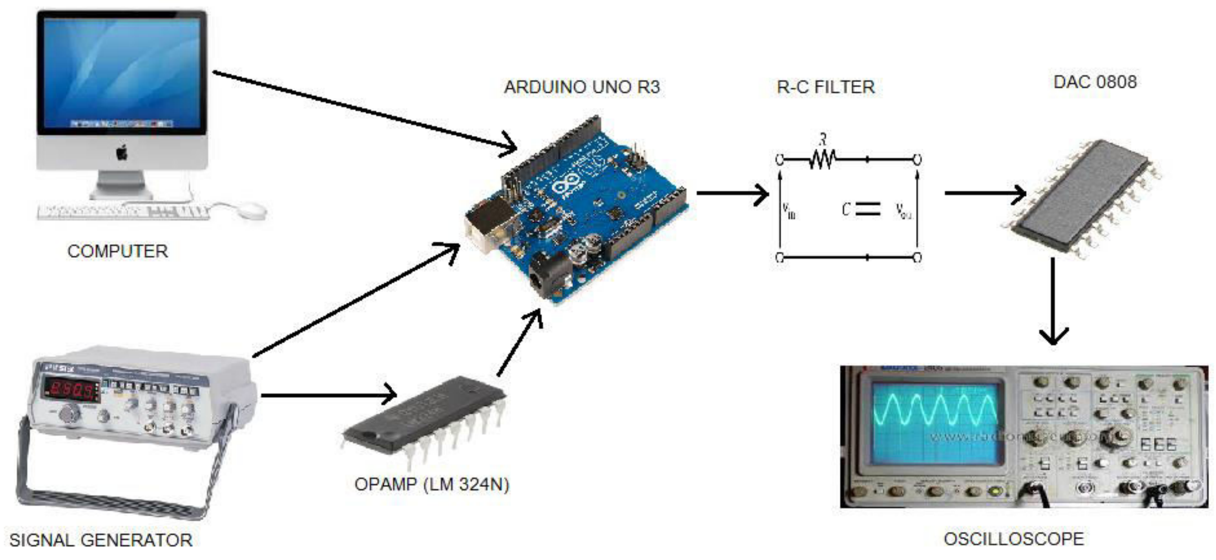


Fig 2.1: Schematic diagram of the setup

3. PROPORTIONAL-PLUS-INTEGRAL-PLUS-DERIVATIVE CONTROLLERS:-
"PID control" is the method of feedback control that uses the PID controller as the main tool. The basic structure of conventional feedback control systems is shown in Figure below, using a block diagram representation. In this figure, the process is the object to be controlled. The purpose of control is to make the process variable *y* follow the set-point value *r*. To achieve this purpose, the manipulated variable *u* is changed at the command of the controller. As an example of processes, consider a heating tank in which some liquid is heated to a desired temperature by burning fuel gas. The process variable *y* is the temperature of the liquid, and the manipulated variable *u* is the flow of the fuel gas [5].
The "disturbance" is any factor, other than the manipulated variable, that influences the process variable. Figure below assumes that only one disturbance is added to the manipulated variable. In some applications, however, a major disturbance enters the process in a different way, or plural disturbances need to be considered. The error *e* is defined by $e = r - y$. The compensator *C(s)* is the computational rule that determines the manipulated variable *u* based on its input data, which is the error *e* in the case of Figure. The last thing to notice about the Figure is that the process variable *y* is assumed to be measured by the detector, which is not shown explicitly here, with sufficient accuracy instantaneously that the input to the controller can be regarded as being exactly equal to *y* [5].

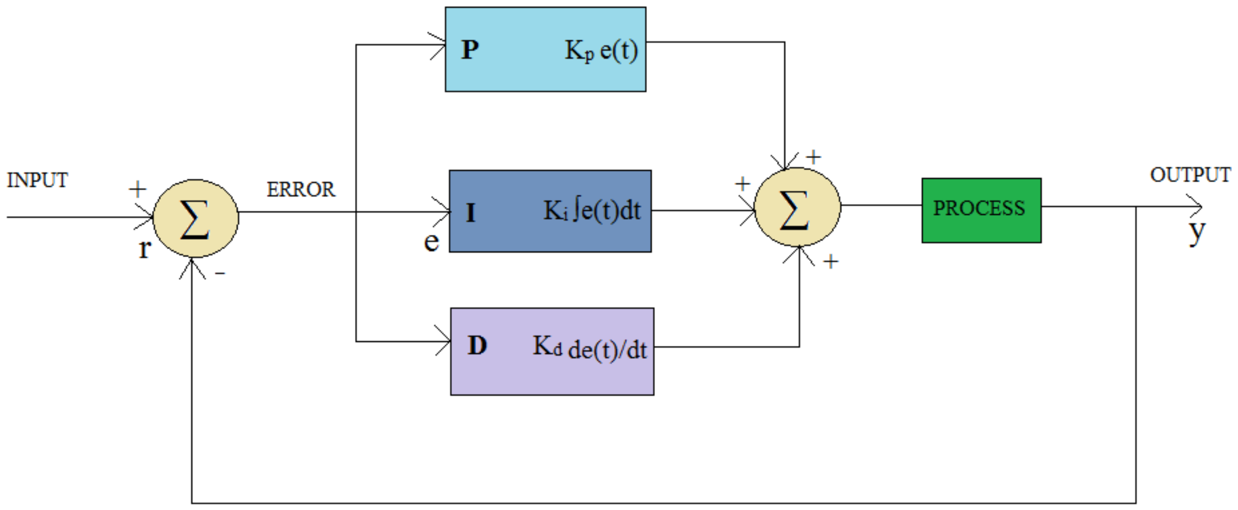


Fig. 3.1 PID CONTROLLER [5].
When used in this manner, the three element of PID produces outputs with the following nature:

- P element: proportional to the error at the instant *t*, this is the "present" error.
- I element: proportional to the integral of the error up to the instant *t*, which can be interpreted as the accumulation of the "past" error.
- D element: proportional to the derivative of the error at the instant *t*, which can be interpreted as the prediction of the "future" error [5].

Thus, the PID controller can be understood as a controller that takes the present, the past, and the future of the error into consideration. The transfer function $G_c(s)$ of the PID controller is [5]:

$$G_C(s) = K_P\left[1 + \frac{1}{sT_i} + T_d s\right] = K_P + \frac{K_i}{s} + T_d s \qquad (1)$$

4. **INTRODUCTION TO ARDUINO UNO:**
Arduino is a tool for making computers that can sense and control more of the physical world than our desktop computer. It's an open-source physical computing platform based on a simple microcontroller board, and a development environment for writing software for the board.
Arduino can be used to develop interactive objects, taking inputs from a variety of switches or sensors, and controlling a variety of lights, motors, and other physical outputs. Arduino projects can be stand-alone, or they can communicate with software running on your computer (e.g. Flash, Processing and MaxMSP.). The Arduino programming language is an implementation of Wiring, a similar physical computing platform, which is based on the Processing multimedia programming environment. [11]

5. INTERACTIVE ALGORITHM:

Controller manufacturers arrange the Proportional, Integral and Derivative modes into three different controller algorithms or controller structures. [9]
These are called Interactive, Noninteractive, and Parallel algorithms. Some controller manufacturers allow you to choose between different controller algorithms as a configuration option in the controller software [9].

$$CO = K_p \left[e + \frac{1}{T_i}\int edt\right] \times \left[1 + T_d \frac{de(t)}{dt}\right] \qquad (2)$$

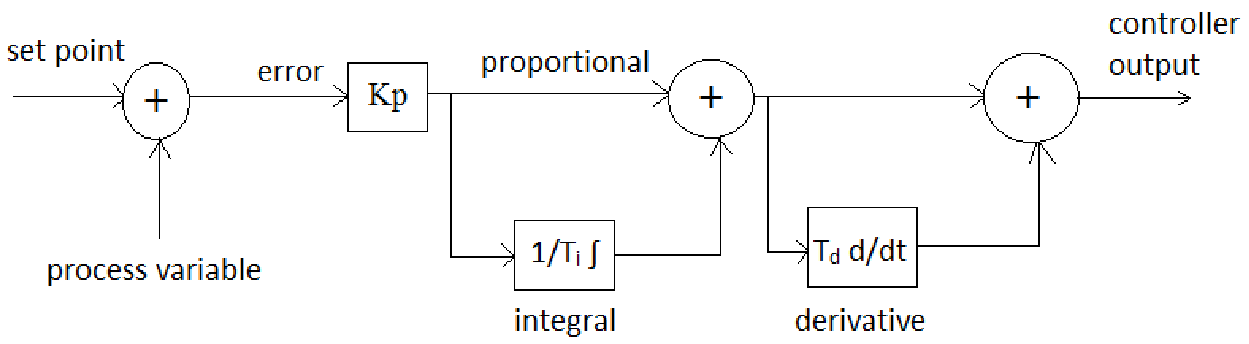


Fig 5.1: INTERACTIVE CONTROLLER ALGORITHM [8]

5.1 CONTROLLER PROGRAM IN ARDUINO UNO ENVIRONMENT:

```
int analogInpin1=A0;        //signal generator connected to analogpin A0.

int analogInpin2=A1;        //signal generator through OPAMP connected to analogpin A1.

float Input;                //variable to store the input signal.

float Output;               //variable to store the output of the controller to be observed by Oscilloscope.

int k1=100;                 //variable to store Kp value.

int k2 = 200;               //variable to store Kd value.

int k3=300;                 //variable to store Ki value.
float sum=0;                //variable to store the integration value or sum of areas up to time‘t’.
void setup ()               //setup function to run only once for initialization
{
Serial.begin (9600);        //setup serial
}
void loop (){                         //loop function to run continuously
If (analogRead (analogInpin1) >= 0)        //checks if voltage signal is in positive half
Input = analogRead (analogInpin1);       //read the analogpin1
```

```
else                                                    //if voltage signal is in negative half
Input=analogRead (analogInpin2);              //reads the analogpin2
Serial.print ("Input=");                              //prints in serial monitor
Serial.println (Input);                               //debug value
Input= (Input*5)/1023;                         //converts the value from 0-1023 to it's absolute value
Serial.print ("Input=");
Serial.println (Input);
float P;                                        //variable to store the proportional term's value
P=k1*Input;
Serial.print ("P=");
Serial.println (P);
float D;                                          //variable to store the derivative term's value
D= k2*(diff (Input, 1));                         //calls the function diff ()
Serial.print ("D=");
Serial.println (D);
int I;                                              //variable to store the integral term's value
sum= sum + integ (Input, 1);                //calls the function integ () and modifies the value of sum
I=k3*(sum);
Serial.print ("I=");
Serial.println (I);
Output=P+I+D;
Output= (Output/4);                         //converts the output from 0-1023 to 0-255
Output= (int) Output;                      //converts the float value (0-255) of output to int (0-255)
bin (Output);                                 //calls the function bin ()
delay (1);                                     //waits for 1 ms
}
//function definition of diff () to find the derivative of a function or signal at a point using differential calculus.
float diff (float x, int n)
{
int y;
delay (1);
```

```
if (analogRead (analogInpin1) >= 0)

y= analogRead (analogInpin1);

else

y= analogRead (analogInpin2);

y= (5*y)/1023;

int z;

z=(y-x)/n;

return (z);

}

//function definition of integ () to find the integration or area under the curve of a function or signal from 0 to time‘t’
using integral calculus.

float integ (float a, int m){

int b;

delay (1);

if (analogRead (analogInpin1) >= 0)

b= analogRead (analogInpin1);

else

b= analogRead (analogInpin2);

b = (5*b)/1023;

int c;

c = ((a + b)*m)/2;

return (c);}

//function definition of bin () to convert 0-255 PWM output signal to binary 0 or 1.

int bin (int p){

int i, q, r;

for (i=6; i<=13; i++){

q=p/2;

r=p%2;

digitalWrite (i, r);      //sets the digital pin ‘i’ on or off depending on the ‘1’ or ‘0’ value of ‘r’

p=q;

}}
```

6. DESIGN OF LOW PASS R-C FILTER:

The pulse width modulated (PWM) signal outputs are variable duty cycle square-waves with 5 volt amplitude. These signals can each be decomposed into a D.C. component plus a new square-wave of identical duty-cycle but with a time-average amplitude of zero. Figure 6.1 depicts this graphically. It will be shown that the amplitude of the D.C. component is directly proportional to the PWM duty cycle. [4]

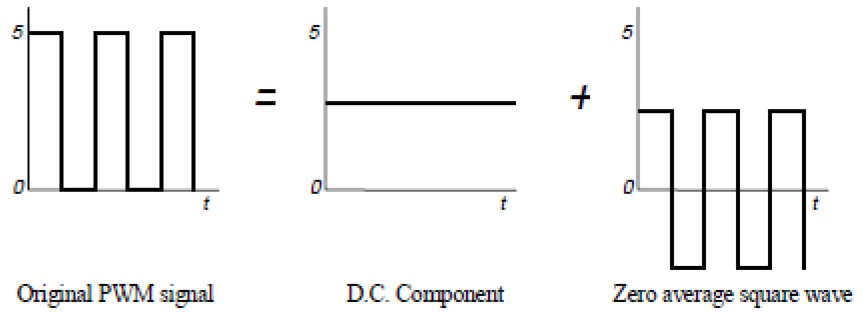


Fig 6.1 DECOMPOSITION OF PWM SIGNAL (shown for 50% duty cycle) [4]

The idea behind realizing D/A output from the PWM signal is to analog low-pass filter the PWM output to remove most of the high frequency components, thereby leaving only the low frequency (D.C.) components. This is depicted in Figure 6.2. The bandwidth of the low-pass filter will essentially determine the bandwidth of the D/A. A frequency analysis of the PWM signal is given in the next section in order to provide a theoretical basis for the filtering strategy. [4]

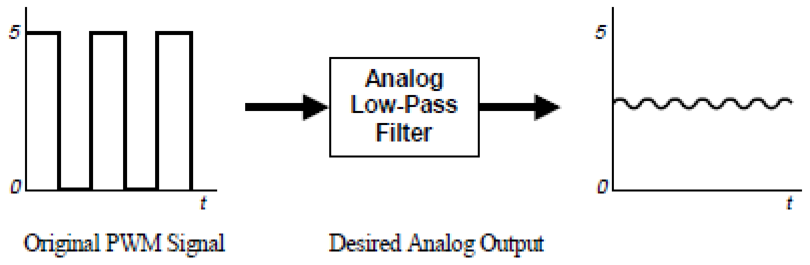


Fig 6.2 Analog Filtering of PWM Signal [4]

6.1 CALCULATING R-C VALUES BY BACK OF THE ENVELOPE FILTER DESIGN:

We have the Arduino Uno with an f_{PWM} of 8 kHz, a V_{PWM} of 5 V, and we are targeting LSB / 2 of ripple for our 10-bit A-to-D converter. For this case, LSB / 2 corresponds to about 2.4 mV of voltage ripple. So, the first thing we need to do is calculate the attenuation factor we need.

Using equation 1: $A_{dB} = 20 \times \log(\frac{0.0024\ (V)}{5\ (V)})$ (3)

Now that we know the attenuation factor, the next step is to compute the bandwidth of the filter we need to design, using equation 3.

We will do this twice, once for a first order filter:

$$f_{3\ dB} = 8000\ (HZ)\ \times\ 10^{\frac{-66.375\ (dB)}{-20}} \quad (4)$$

and again for a second order filter:

$$f_{3\,dB} = 8000\ (HZ) \times 10^{-\frac{-66.375\ (dB)}{-40}} = 175\ (HZ) \qquad (5)$$

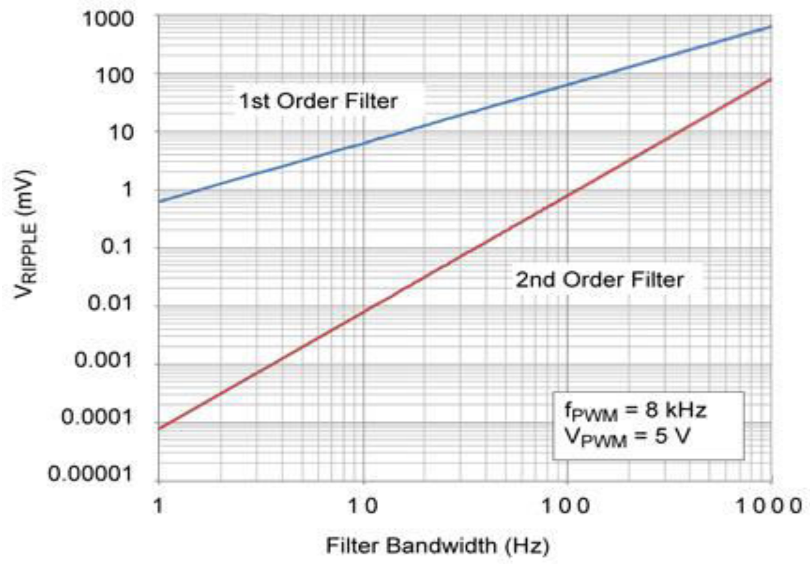


Fig 6.3 Ripple Voltage versus 3-dB frequency (used for estimating first and second order filters)

6.2 BUILDING THE FILTER:

A first order filter uses one capacitor and one resistor, and a second order filter uses two resistors and two capacitors. The extra resistor, R_L, is there to represent a typical input resistance of the measurement system. [4]

We will build the first order filter first. Building a first order filter is as simple as choosing a starting capacitor value, and then computing the resistor value.

Given: $f_{3\,dB} = \frac{1}{2\,\Pi \times R_F \times C_F}$ and $R_F = \frac{1}{2\,\Pi \times C_F \times f_{3\,dB}}$ (6)

For the first order filter (f_{3dB} = 3.84 Hz), we can chose a capacitor, C_F , value of 10 µF and use equation 6 to compute that the filter resistor, R_F , value:

$$R_F = \frac{1}{2\,\Pi \times 10\ (\mu F) \times 3.84\ (HZ)} = 4.144\ (K\Omega) \qquad (7)$$

R_F should be rounded up to the closest 1% resistor value of 4.22 kΩ, and the corresponding f_{3dB} value would be 3.77 Hz.

The implementation for a second order passive low pass filter is simply cascading two first order filters in series. In this example, we will create two first order filters, each with an f_{3dB} of 175 Hz as calculated above using equation 3. For this example, we chose 1 µF for C_F and computed R_F to be 909 Ω (standard 1% value) by substituting into equation 8: [1]

$$R_F = \frac{1}{2\,\Pi \times 1\ (\mu F) \times 175\ (HZ)} = 909\ (\Omega) \qquad (8)$$

6.3 SIMULATED RESULTS OF R-C FILTER:

The following several figures illustrate simulation results of the two filters designed in this article. The PWM output was configured such that f_{PWM} = 8 kHz, D = 50%, and V_{PWM} = 5 V, as shown in figure 6.4. This figure represents the unfiltered input signal. It should be remembered that we will be applying a much slower filter to this waveform, so many of the following graphs are at a different time scale.

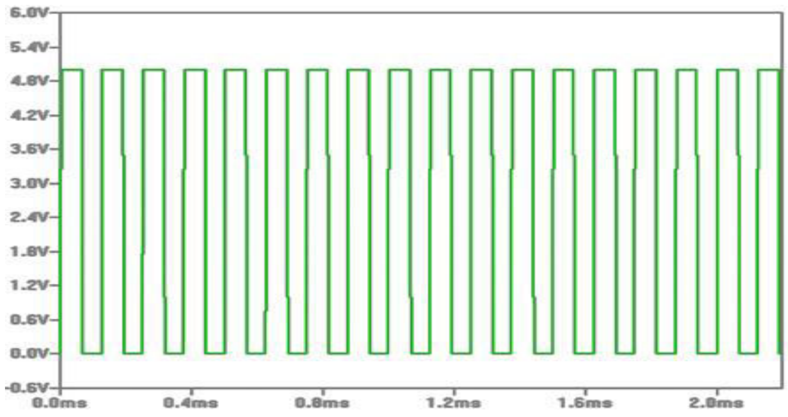


Fig 6.4 Input PWM signal (f_{PWM} = 8 kHz)

Figure 6.5 illustrates the output transfer function of the waveforms in figure 6.4 for both the first order (green trace) and the second order (blue trace) filters. It is obvious to see that the lower frequency f_{3dB} of the first order filter causes a much slower response. It is also apparent that the series resistance of the filter does impact the voltage of the output signal, as there is indeed a resistor divider that reduces the voltage at the system input (4.22 kΩ for the filter and 50 kΩ for the approximated input resistance of the system). Overall, the response of the curves is what was expected from our calculations.

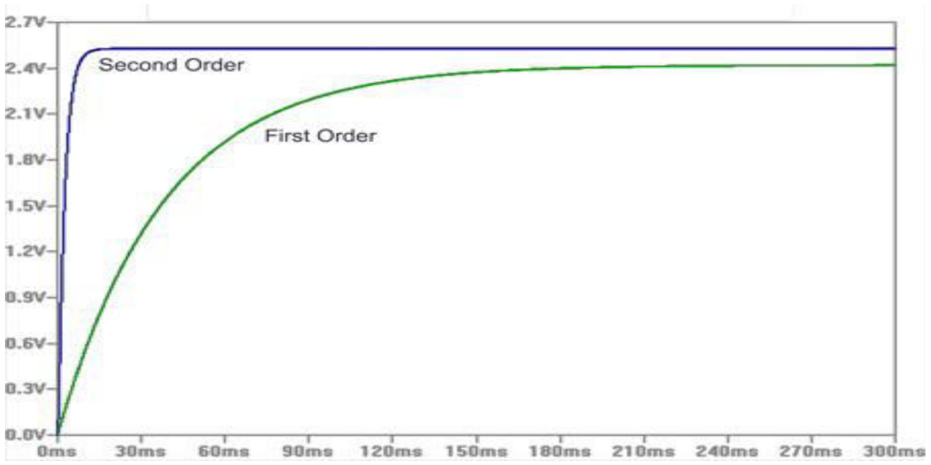


Fig 6.5 Output Response of first and second order filters

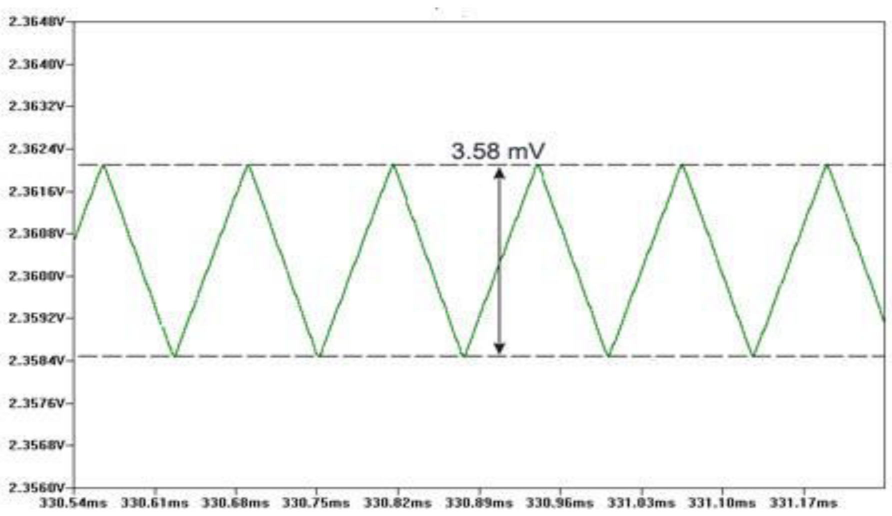


Fig 6.6 Detail of output ripple for first order filter simulation, ripple = 3.58 mV

Next we need to have a closer look at the ripple voltage of the settled output waveform. Figure 6.6 shows the detailed view of the ripple of the first order filter output. It shows that the final ripple value is 3.58 mV for our f_{3dB} of 3.77 Hz. It is a little bit higher than we were targeting, but as stated before, it was expected that we may be off by a little bit, based on our assumptions. However, our filter is definitely performing in that region and could be slightly adjusted. Increasing R_F or C_F slightly will reduce the ripple. Increasing R_F or C_F will also lower f_{3dB}. Simulation experiments will show that moving f_{3dB} down to about 2.57 Hz by changing R_F to 6.19 kΩ will bring the ripple into specification.

Figure 6.7 shows a zoomed in version of the output ripple for the second order low pass filter. Here we can see that we were even closer with our original design estimate. This filter achieved a ripple voltage of 2.86 mV. Not bad since we were shooting for 2.4 mV. Once again, this ripple value can be reduced by modifying the filter slightly and lowering f_{3dB} in a similar manner as the first order example.

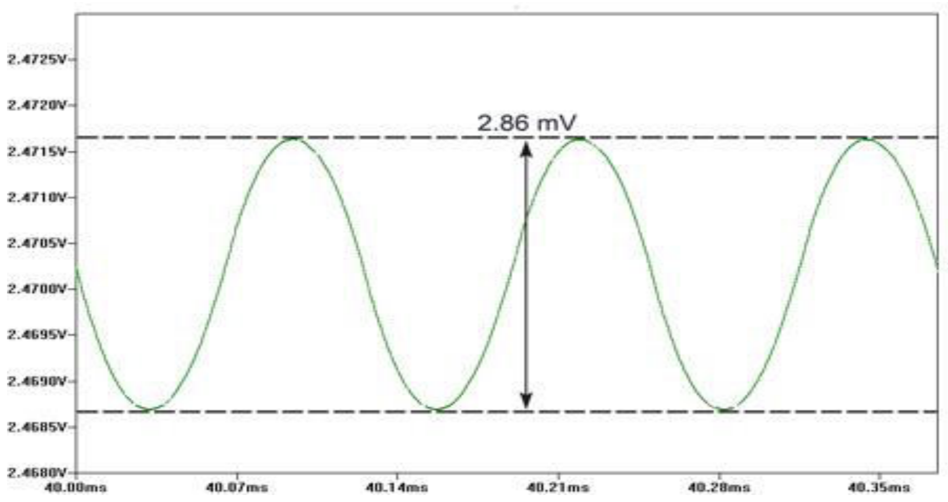


Fig 6.7 Detail of output ripple for second order filter simulation, ripple = 2.86 mV

7. R-2R LADDER DAC DESIGN:

The binary weighted ladder DAC requires exponentially increasingly large resistor values as the number of bits of resolution increases. This circuit configuration thus is poorly suited for implementation as an IC, on which the large area required for large resistors values can make resistors with large values quite expensive. In contrast, the *R-2R Ladder DAC* circuit, shown below configured for 4-bit resolution, requires only 2 values of resistors (3 values including RF) regardless of how many bits of resolution we require. [2]

For a set of input voltages [V1 V2 V3 V4] that represents the value of some quantity expressed as a 4-bit binary number, the (negative of the) output voltage, given by:

$$-V_{out} = \frac{R_F}{16\,R}\,(8\,V_1 + 4\,V_2 + 2\,V_3 + 1\,V_1) \qquad (8)$$

We should choose values for R, and R_F so that – V_{out} is the decimal equivalent of the 4-bit binary input [V1 V2 V3 V4] in which a binary 1 is represented by 5V and a binary 0 by 0V. [3]

7.1 CHOOSING THE R AND R_F VALUES:

Standard base resistor values are given by *The Electronic Industries Association (EIA)* and for the most commonly used tolerances (1%, 5%, 10%) along with typically available resistance ranges are shown in Tables 6.2, 6.3 and 6.4 respectively. To determine values other than the base, the base value is multiplied by 10; 100; 1,000; or 10,000. [6]

In this project 1% tolerance resistance values are used for better accuracy and precision in the output of the R-2R Ladder DAC.

For 8 bit R-2R Ladder DAC needed to convert digital output from Arduino to analog output, we know:

$$\mathbf{-V_{OUT} = (R_F / 1024\,R)\,[128\,V_1 + 64\,V_2 + 32\,V_3 + 16\,V_4 + 8\,V_5 + 4\,V_6 + 2\,V_7 + 1\,V_8]} \qquad (9)$$

But for the 8 bit DAC 0-5 V analog is scaled to 0-1023 digital, so, we know:

$$\mathbf{-V_{OUT} = (5 / 1023)\,[128\,V_1 + 64\,V_2 + 32\,V_3 + 16\,V_4 + 8\,V_5 + 4\,V_6 + 2\,V_7 + 1\,V_8]} \qquad (10)$$

So, we have $\mathbf{R_F / 1024\,R = 5 / 1023 \Rightarrow R_F = 5.00489\,R \Rightarrow R_F \approx 5\,R}$ (11)

7.2 RESULT OF DAC OUTPUT:

In this project the R-2R Ladder DAC method is used to convert the binary to analog output signal. Binary digits are obtained on different pins from the PWM output of Arduino analogWrite () function by the bin () function of the program burned on the Arduino platform. To design the Ladder R = 20 Ω and R_F = 100 Ω is used. The analog sinusoidal output was observed and studied in an Oscilloscope for sinusoidal input signal with different frequencies from a Signal Generator.

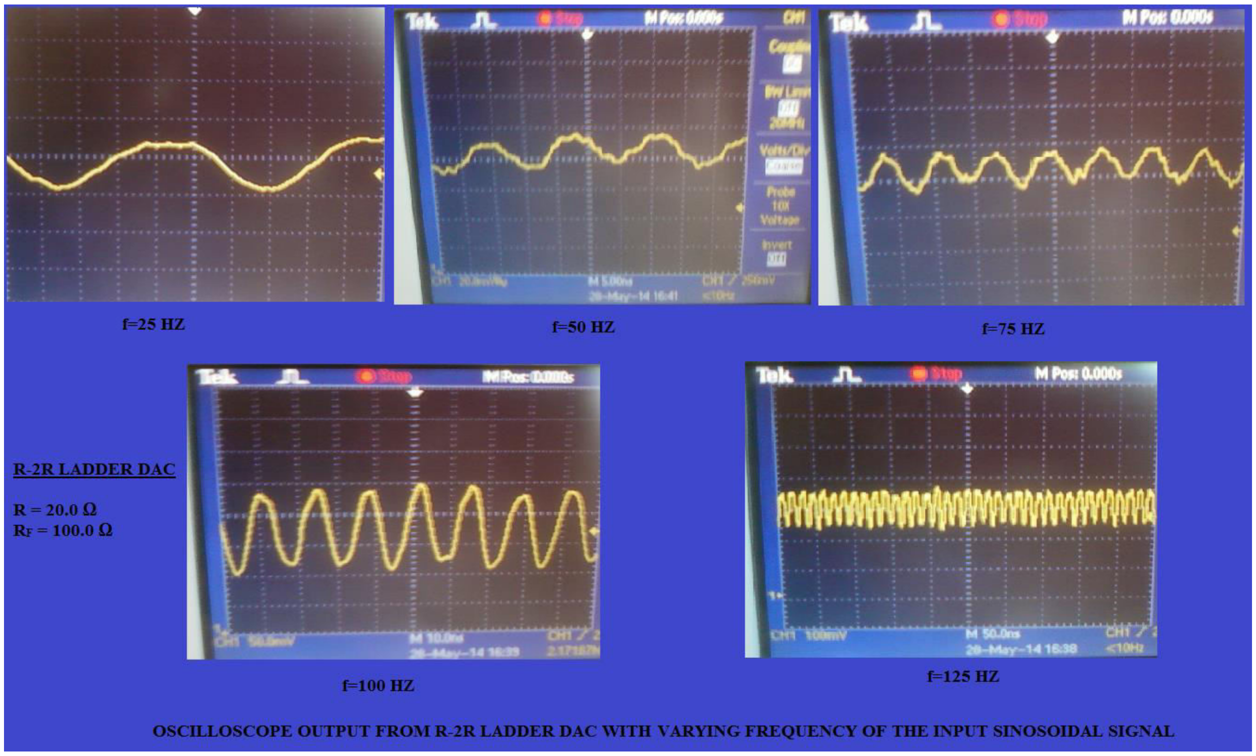
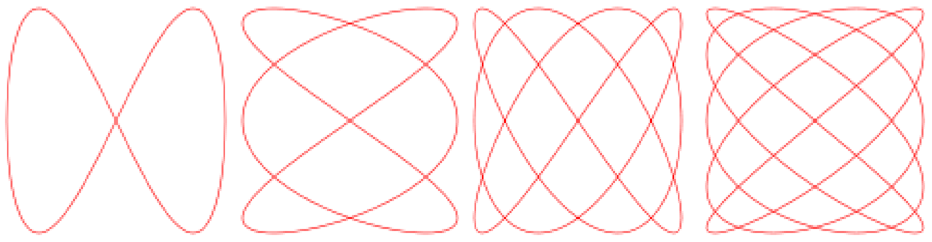

f=25 HZ
f=50 HZ
f=75 HZ
R-2R LADDER DAC
R = 20.0 Ω
RF = 100.0 Ω
f=100 HZ
f=125 HZ
OSCILLOSCOPE OUTPUT FROM R-2R LADDER DAC WITH VARYING FREQUENCY OF THE INPUT SINOSOIDAL SIGNAL

Fig 7.1 Oscilloscope output with varying frequencies of input sinusoidal signal

8. MEASURING PHASE DIFFERENCE AND GAIN BY LISSAJOUS CURVE:

Lissajous curve is the graph of a system of parametric equations:

$$x = A \sin(at + \delta),\ y = B \sin(bt)$$



a = 1, b = 2 (1:2) a = 3, b = 2 (3:2) a = 3, b = 4 (3:4) a = 5, b = 4 (5:4)

Fig 8.1 Lissajous figures with $\delta = \pi/2$, an odd natural number *a*, an even natural number *b [5]*

On an oscilloscope, we suppose *x* is CH1 and *y* is CH2, *A* is amplitude of CH1 and *B* is amplitude of CH2, *a* is frequency of CH1 and *b* is frequency of CH2, so *a*/*b* is a ratio of frequency of two channels, finally, *δ* is the phase shift of CH1.When the input to an LTI system is sinusoidal, the output is sinusoidal with the same frequency, but it may have a different amplitude and some phase shift [10].

For the system of equations:

$$x = A \sin(at + \delta),\ y = B \sin(bt) \text{ and } a = b$$

$$\text{Gain} = G = A/B \text{ and Phase Difference} = \delta$$

The scope display to XY mode is switched to observe the Lissajous pattern. We should be sure to note the sensitivity setting of each input in our measurement. The amplitude values should be recorded in volts rather than divisions. We should set the sensitivities so that the major axis of the ellipse is at an angle of about 45° and several divisions in length. The pattern should be centered on the screen so that the central chord of the ellipse, can be measured with the vertical centerline of the scope graticule. [7]

- The vertical amplifier input is grounded and the trace with horizontal center line is aligned.
- The vertical amplifier to DC is switched and the horizontal amplifier input is grounded. The trace is centered horizontally. The length of this trace, Y_2 is measured.
- The horizontal amplifier to DC is switched and Y_1 is measured.
- The process is repeated for different frequencies. [7]

$$\sin \delta = Y_1 / Y_2 = X_1 / X_2 \qquad (12)$$

$$\Rightarrow \text{Phase Difference} = \delta = \sin^{-1}(Y_1 / Y_2) = \sin^{-1}(X_1 / X_2) \qquad (13)$$

$$\text{Gain} = G = Y_2 / X_2 \qquad (14)$$

8.1 RESULT OF LISSAJOUS CURVE:

The frequency of the input sinusoidal signal is varied and the Lissajous curve between the output sinusoidal and the input sinusoidal is plotted in the X-Y mode of the oscilloscope. Then the phase and gain margin is calculated.

f (HZ)	Lissajous Curve	Measured Values				Calculated Values	
		Y_1	Y_2	δ	G	δ	G
125	ellipse	2.4	2.8	58.99	0.98	57.86	1
100	ellipse	2.8	3.8	47.46	1.3	46.7	1
75	ellipse	1.2	2.0	36.87	0.89	35.88	1
50	ellipse	2.2	4.8	27.28	1.06	27.95	1
25	ellipse	0.8	2.6	17.92	1.24	17.66	1

Table 8.1 Experimental and theoretical results of phase and gain margin from Lissajous curve

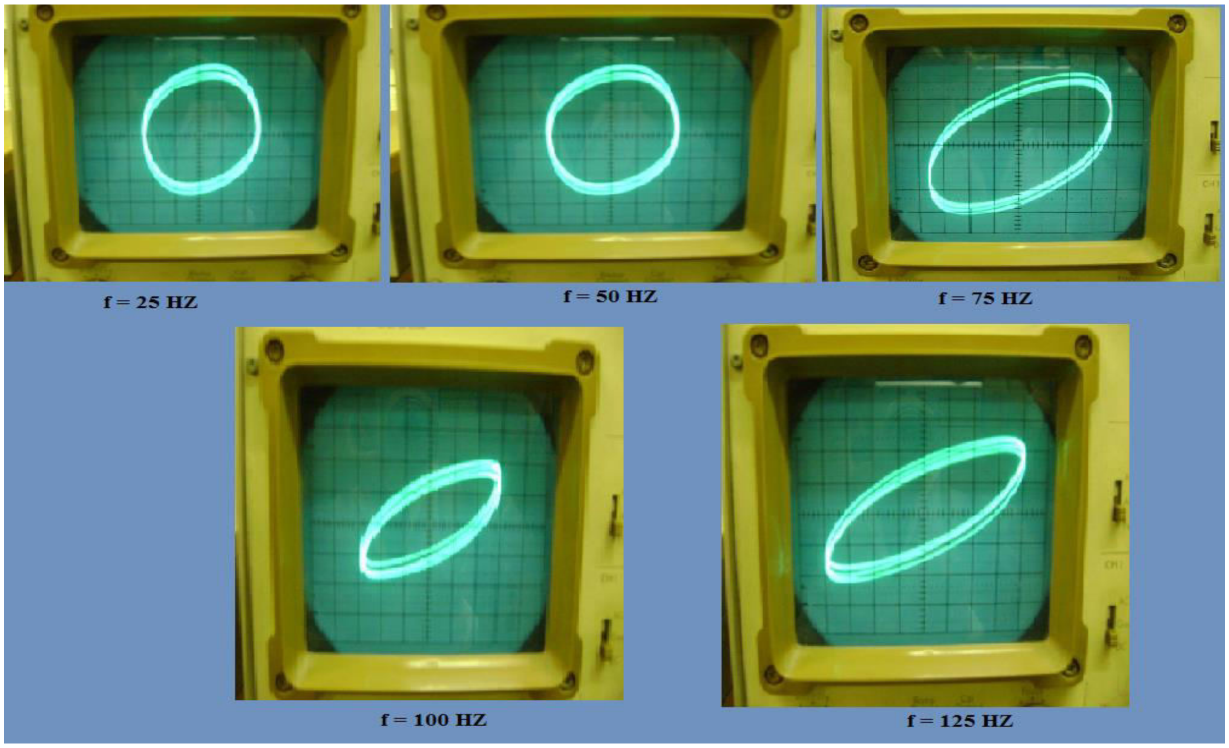


Fig 8.2 Lissajous Curves for different frequencies

9. CONCLUSION:

A detailed study of Proportional, Integral, Derivative, Proportional plus Integral, Proportional plus Derivative, Proportional plus Integral plus Derivative controllers is done. A good understanding of Arduino Uno hardware, software, Arduino programming platform with libraries and functions is acquired.

The different design algorithms of PID controller, Interactive; Non-Interactive and parallel controller algorithms are studied and Interactive controller algorithm is chosen for the design in this project.

Controller program is written in Arduino IDE and burnt in Arduino. The program is well explained by comments and separate functions are written for different PID terms and for the conversion of PWM signal to binary. Detailed description of the program is provided for each step for better understanding of the work. Inverting, Summing and Voltage follower configurations of OP-AMP are studied and used in voltage conversion in the negative half cycle, digital to analog conversion and maintaining a constant voltage between Arduino input and output respectively.

Study of PWM and low pass R-C filter for converting PWM to analog voltage is done and the design of R-C filter is achieved by Back of the envelope method. The attenuation factor, ripple voltage and 3-db frequency are calculated accurately for both 1st and 2nd order filter.

Another way to convert PWM to analog signal output is realized by DAC-0808. Detailed study of pin-configurations and DAC circuit design is done. In realizing DAC, Binary Weighted Ladder DAC and R-2R Ladder DAC configurations of OP-AMP are studied and R-2R Ladder DAC circuit is used for the purpose of the project for more accuracy, less resistor requirements and hence cost. For better accuracy and precision in the design of R-2R Ladder DAC, R and R_F values are chosen from the 1% standard tolerance values of resistors.

R-2R Ladder DAC method is used to convert the binary to analog output signal. Binary digits are obtained on different pins from the PWM output of Arduino analogWrite () function by the bin () function of the program burned on the Arduino platform. The analog sinusoidal output was observed and studied in an Oscilloscope for sinusoidal input signal with different frequencies from a Signal Generator.

Lissajous curve is studied in detail. The frequency of the input sinusoidal signal is varied and the Lissajous curve between the output sinusoidal and the input sinusoidal is plotted in the X-Y mode of the oscilloscope. Then the phase and gain margin is calculated.

As the input signal is sinusoidal, the output signal obtained on the oscilloscope by changing the parameters of PID controller appears to be sinusoidal with almost the same frequency but with some phase delay as expected. The outputs with different frequencies of the input signal are also plotted. Lissajous curve plot shows that there is a relationship between the frequency of the input and the phase margin between the two input and output signals. The Gain margin comes out to be almost 1 as the parameters are varied accordingly.

This project can be used for any input signal, not just sinusoidal and the output from the plant or any microcontroller can be measured controlling the parameters of the controller in analog even if the by default output is PWM and studied by the Lissajous curve for gain and phase margin between the two signals.

10. BIBLIOGRAPHY:

[1] Shenton, A.T., & Shafiei, Z., Relative stability for control system with adjustable parameters, J. of guidance, Control and Dynamics 17(1994) 304-310.

[2] Xue Dingyu, Chen Yang Quan and Atherton P. Derek, "Linear Feedback Control".

[3] Ogata Katsuhiko, Modern control Engineering, fourth edition, 2002.

[4] Kuo C. Benjamin, Automatic Control System, seventh edition, October 2000.

[5] Wikipedia.org

[6] Process Control Simultor PCS 327 Manual, Feedback Instruments.

[7] Ho, W. K., Hang C. C. & Cao L. S., Tuning of PID controllers based on gain and phase margin specifications. Automatica, 31(3) (1995), 497-502.

[8] Astrom, K. J., & Hagglund, T. Automatic tuning of simple regulators with specifications on phase and amplitude margins. *Automatica*, *20*(1984), 645-651

[9] Ziegler, J. G., & Nichols, N. B. Optimum settings for automatic controllers. Transactions of the ASME, 64(1942), 759-768.

[10] Ho, W. K., Hang, C. C., Zhou, J. H., Self-tuning PID control of a plant with under-damped response with specifications on gain and phase margins. IEEE Transactions on Control Systems Technology, 5 (4) (1997), 446-452.

[11] www.arduino.cc