GENERATION OF SINUSOIDAL PULSE WIDTH MODULATED SIGNAL USING ARDUINO MICROCONTROLLER

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR

THE DEGREE OF

BACHELOR OF TECHNOLOGY

IN

ELECTRICAL ENGINEERING

By

Pallavi Patel (111EE0223)

NATIONAL INSTITUTE OF TECHNOLOGY, ROURKELA

CERTIFICATE

This is to certify that the thesis entitled "**Generation of Sinusoidal Pulse Width Modulated Signal using Arduino Microcontroller**", submitted by **Pallavi Patel (Roll. No. 111EE0223)** in partial fulfillment of the requirements for the award of **Bachelor of Technology** in **Electrical Engineering** during session 2014-2015 at National Institute of Technology, Rourkela. A bonafide record of research work carried out by them 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.

Dept. of Electrical Engineering
National institute of Technology
Rourkela-769008

Prof. SANDIP GHOSH
Assistant Professor

ACKNOWLEDGEMENT

On the submission of my thesis entitled "**Generation of Sinusoidal Pulse Width Modulated Signal using Arduino Microcontroller"** I would like to extend my gratitude & sincere thanks to my supervisor **Dr. Sandip Ghosh**, Asst. Professor, Department of Electrical Engineering for his constant motivation and support during the course of my work in the last one year. I truly appreciate and value his esteemed guidance and encouragement from the beginning till the end of this thesis. He guided me throughout the dissertation process, never accepting any less than my best efforts. His reliable and incredible support throughout the research work and his technical knowledge has led me through many problems easily. I thank him for his help and profiting me by his skill, support and valuable time and criticisms, without which this project would not have been possible.

I would also like to thank **Mr. Jitendra Jain**, Phd Scholar, Dept. of Electrical Engg, NIT ROURKELA for his constant support whenever needed. He has helped me in any way he can to carry out this project.

Pallavi Patel

111EE0223

ABSTRACT

Motion control is required in large number of industrial and domestic applications like transportation systems, textile mills, fans, pumps etc. Systems employed for motion control are "Drives". With the advancement of power electronics, microprocessors and digital electronics, typical electric drive systems nowadays are becoming more compact, efficient, cheaper and versatile.

Controllers for Power Modulator is built in Control Unit which usually operates at much lower voltage and power levels. In addition to operating Power Modulator as desired, it may also generate commands for the protection of Power Modulator and Motor. Input command signal, which adjusts the operating point of the drive, forms an input to the Control Unit.

In this project, "ARDUINO" is used as a control unit to generate PWM signals to control Power Modulator. A code for the SPWM signal is developed in Arduino software.

CONTENTS

Acknowledgement **1**

Abstract **2**

Contents **3**

List of Figures **5**

CHAPTER 1

INTRODUCTION

1.1 Introduction **7**

1.2 Literature Review **8**

1.3 Objective of the thesis **8**

CHAPTER 2

SINUSOIDAL PULSE WIDTH MODULATION

2.1 Sinusoidal Pulse Width Modulation **10**

2.2 Generation of SPWM in MATLAB SIMULINK **11**

CHAPTER 3

ARDUINO

3.1 Arduino **13**

3.2 Hardware **13**

3.3 Arduino UNO **14**

CHAPTER 4

ARDUINO PROGRAMMING GUIDELINES

Arduino Programming **17**

CHAPTER 5

RESULTS

5.1 Program in Matlab **22**

5.2 Program in Arduino **24**

CHAPTER 6

CONCLUSION

Conclusion **26**

REFERENCES **27**

LIST OF FIGURES

FIG NO.	TITLE	PAGE NO.
1	Electrical Drive	07
2	Simulink block for PWM generation	11
3	Arduino Microcontroller	13
4	ATmega328 Pin Diagram	14

CHAPTER 1

INTRODUCTION

Introduction

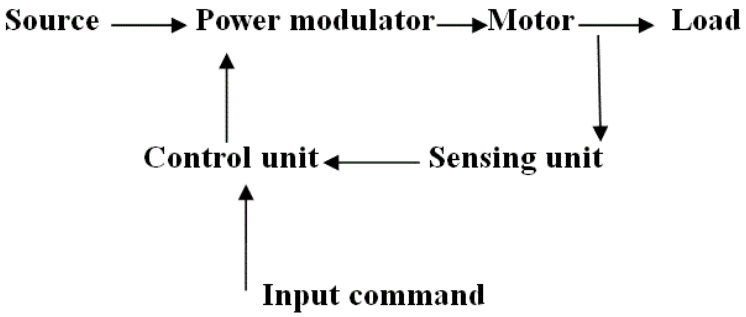
Literature Review

Objective of the Thesis

1.1 INTRODUCTION

Whatever the term electric motor or generator is utilized, we have a tendency to surmise that the rotational speed of these machines is completely controlled just by the source voltage and frequency of the source current. However, the rotational speed of an electrical machine can also be controlled decisively by actualizing the idea of drive. The principle point of interest of this idea is, the movement control is effectively streamlined with the assistance of drive. In short, the systems which control the movement of the electrical machines are known as electrical drives. A common drive system is amassed with an electric motor (may be a few) and a complex control framework that controls the pivot of the motor shaft. Presently, this control could be possible effectively with the assistance of programming. Along these lines, the controlling becomes more accurate and this idea of drive likewise gives the usability. This drive system is generally utilized as a part of substantial number of industrial and residential applications like processing plants, transportation systems, material factories, fans, pumps, engines, robots and so forth. Drives are utilized as prime movers for diesel or petrol engines, gas or steam turbines, pressure driven motors and electric motors [1].

The very basic block diagram an electric drives is shown below. The load in the figure represents various types of equipment which consists of electric motor, like fans, pumps, washing machines etc.



Block diagram of an electrical drive

Figure 1 : Electical Drive

Variable speed control of AC electrical machines makes use of forced-commutated electronic switches such as IGBTs, MOSFETs, and GTOs. Asynchronous machines fed by *pulse width modulation* (PWM) voltage sourced converters (VSC) are nowadays gradually replacing the DC motors and thyristor bridges. With PWM, combined with modern control techniques such as

field-oriented control or direct torque control, you can obtain the same flexibility in speed and torque control as with DC machines [2].

1.2 LITERATURE REVIEW

Pulse width Modulation (PWM) is the premise for control in Power Electronics. The hypothetically zero rise and fall time of a perfect PWM waveform speaks to a favoured method for driving modern semiconductor power devices. Except for some resonant converters, the greater part of power electronic circuits are controlled by PWM signals of different types. The fast rising and falling edges guarantee that the semiconductor power devices are turned on or off as quick as essentially conceivable to minimize the switching transition time and the related switching losses. Albeit different contemplations, for example, parasitic ringing and electromagnetic interference (EMI) emission, may force an upper limit on the turn-on and turn-off speed in practical circumstances, the subsequent limited rise and fall time can be overlooked in the examination of PWM signals and processes much of the time [3].

1.3 OBJECTIVE OF THE THESIS

The objective of the thesis is to generate sinusoidal pulse width modulated signal to provide gate pulse signal to the control unit of motor drive to attain variable speed operation. The speed of the motor can be varied by varying the spwm signal generated by Arduino microcontroller as output signal.

CHAPTER 2

SINUSOIDAL PULSE WIDTH MODULATION

Sinusoidal PWM

Generation of SPWM in MATLAB SIMULINK

This chapter gives the information about sinusoidal pulse width modulation technique and how it is generated.

2.1 SINUSOIDAL PWM

Generation of the desired output voltage is achieved by comparing the desired reference waveform (modulating signal) with a high-frequency triangular 'carrier' wave. In SPWM, modulating signal is taken to be sinusoidal voltage, with the following constraints-

1. The peak magnitude of the sinusoidal signal is less than or equal to the peak magnitude of the carrier signal. This ensures that the instantaneous magnitude of the modulating signal never exceeds the peak magnitude of the carrier signal.

2. The frequency of the modulating signal is several orders lower than the frequency of the carrier signal. A typical figure will be 50 Hz for the modulating signal and 20 Kilohertz for the carrier signal [4].

Depending on whether the signal voltage is larger or smaller than the carrier waveform, either the positive or negative DC voltage is applied at the output. Note that over the period of one triangular wave, the average voltage applied to the load is proportional to the amplitude of the signal during this period. The resulting chopped square waveform contains a replica of the desired waveform in its low frequency components, with the higher frequency components being at frequencies close to the carrier frequency. Notice that the root mean square value of the ac voltage waveform is still equal to the dc bus voltage, and hence the total harmonic distortion is not affected by the PWM process. The harmonic components are merely shifted into the higher frequency range and are automatically filtered due to inductances in the Ac system.

When the modulating signal is a sinusoid of amplitude A_m, and the amplitude of triangular wave is A_c, the ratio $m=A_m/A_c$ is known as modulation index. With a sufficiently high carrier frequency, the high frequency components do not propagate significantly in the ac network due to the presence of inductive elements. However, it does result in a larger number of switching per cycle and hence in an increased power loss. Typically switching frequencies in the 2-15 kHz range are considered adequate for power systems applications.

2.2 GENERATION OF PWM IN MATLAB SIMULINK:

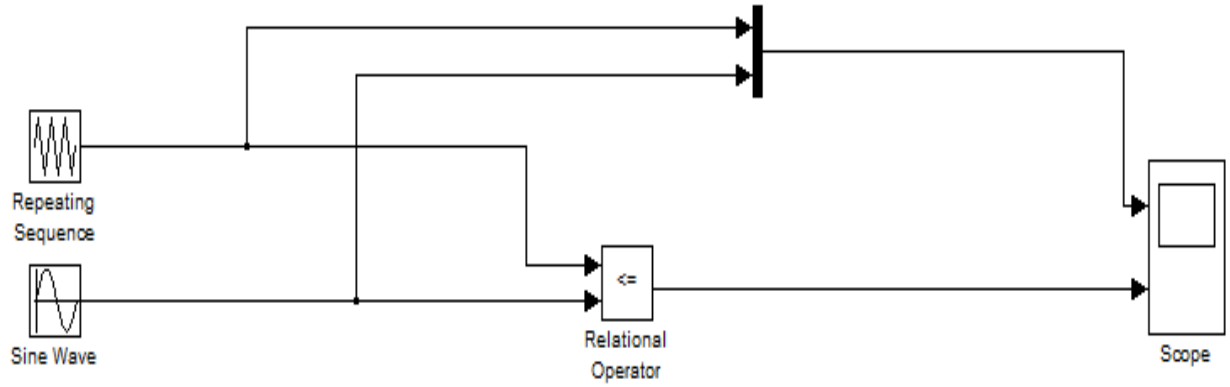
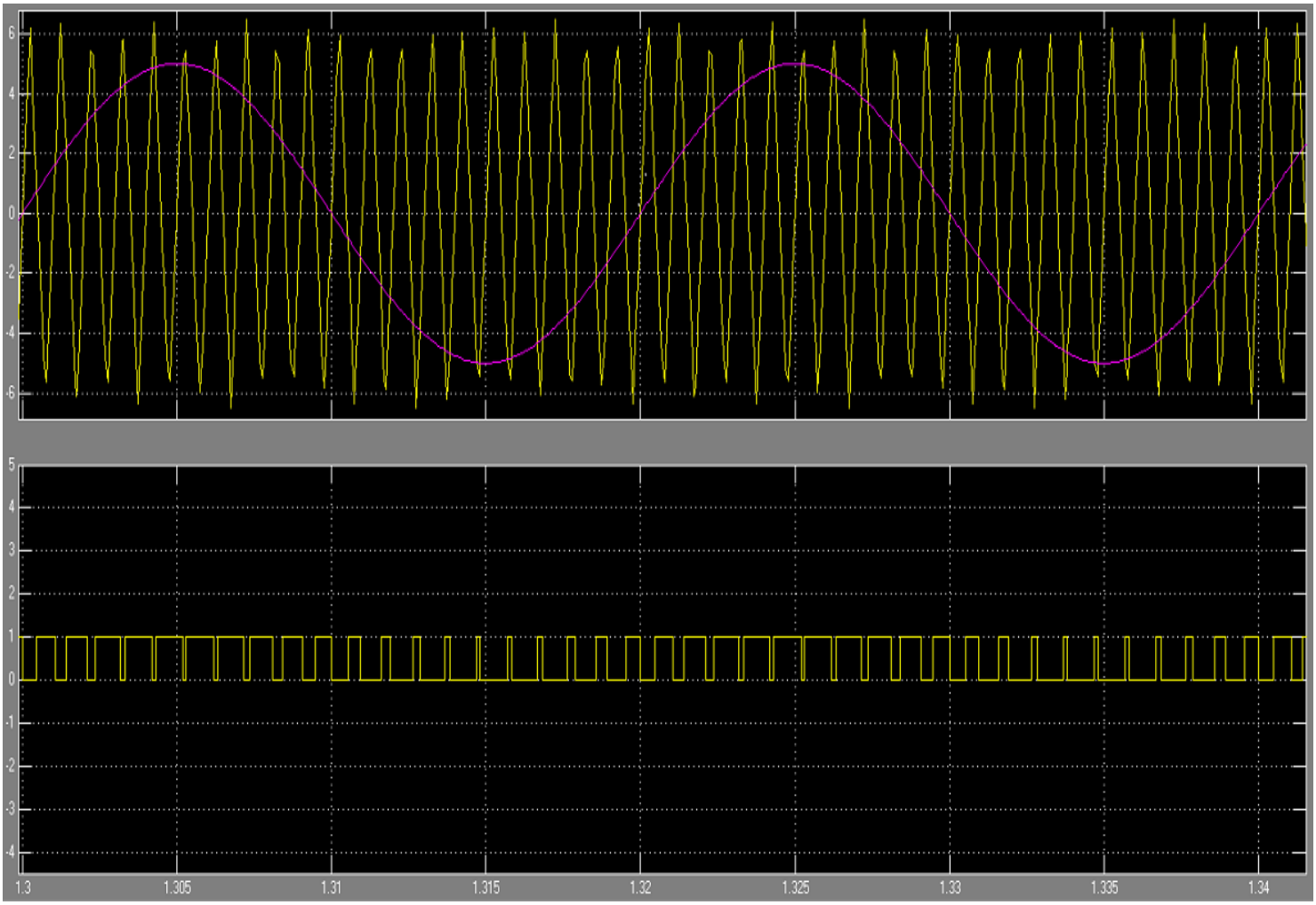


Figure 2 : Simulink block for PWM generation

Output:



CHAPTER 3

ARDUINO

Arduino

Hardware

Arduino Uno

This chapter gives an overview of Arduino and hardware.

3.1 ARDUINO

Arduino is an open-source electronics prototyping platform based on flexible, easy-to-use hardware and software [5].

The Arduino microcontroller is an easy to use yet powerful single board computer that has gained considerable traction in the hobby and professional market. The Arduino is open-source, which means hardware is reasonably priced and development software is free.

Figure 3 : Arduino Microcontroller

These systems provide sets of digital and analog I/O pins that can be interfaced to various extension boards and other circuits. The boards feature serial communications interfaces, including USB on some models, for loading programs from personal computers. For programming the microcontrollers, the Arduino platform provides an integrated development environment (IDE) based on the Processing project, which includes support for C and C++ programming languages.

Arduino can sense the environment by receiving input from a variety of sensors and can affect its surroundings by controlling lights, motors, and other actuators. The microcontroller on the board is programmed using the Arduino programming language (based on Wiring) and the Arduino development environment (based on Processing). Arduino projects can be stand-alone or they can communicate with software on running on a computer (e.g. Flash, Processing, Max/MSP).

3.2 HARDWARE

An Arduino board consists of an Atmel 8-bit, 16-bit and 32-bit AVR microcontroller with complementary components that facilitate programming and incorporation into other circuits. An important aspect of the Arduino is its standard connectors, which lets users connect the CPU board to a variety of interchangeable add-on modules known as shields. Official Arduinos have used the megaAVR series of chips, specifically the ATmega8, ATmega168, ATmega328,

ATmega1280, and ATmega2560. A handful of other processors have been used by Arduino compatibles. Most boards include a 5 volt linear regulator and a 16 MHz crystal oscillator (or ceramic resonator in some variants), although some designs such as the LilyPad run at 8 MHz and dispense with the on board voltage regulator due to specific form-factor restrictions. An Arduino's microcontroller is also pre-programmed with a boot loader that simplifies uploading of programs to the on-chip flash memory, compared with other devices that typically need an external programmer. This makes using an Arduino more straightforward by allowing the use of an ordinary computer as the programmer.

The Arduino board exposes most of the microcontroller's I/O pins for use by other circuits. The current Uno provide 14 digital I/O pins, six of which can produce pulse-width modulated signals, and six analog inputs, which can also be used as six digital I/O pins. These pins are on the top of the board, via female 0.10-inch (2.5 mm) headers. Several plug-in application shields are also commercially available.

3.3 ARDUINO UNO

The Arduino Uno is a microcontroller board based on the ATmega328.

ATmega328

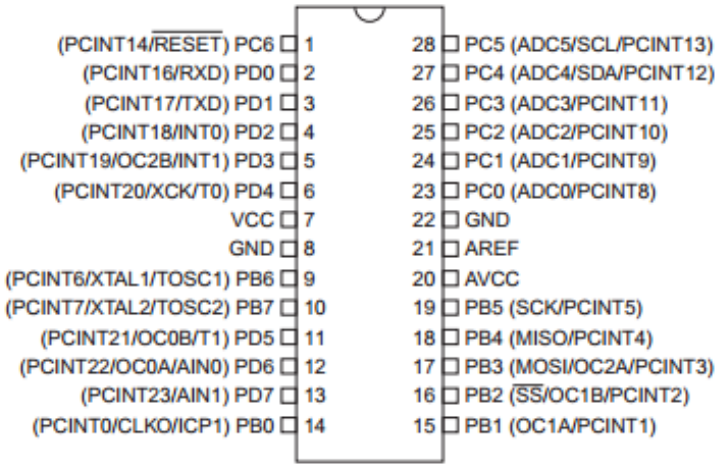


Figure 4 : ATmega328 Pin Diagram

It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz ceramic resonator, a USB connection, a power jack, an ICSP header, and a reset button.

It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started.

The Uno differs from all preceding boards in that it does not use the FTDI USB-to-serial driver chip. Instead, it features the Atmega16U2 programmed as a USB-to-serial converter.

"Uno" means one in Italian and is named to mark the upcoming release of Arduino 1.0. The Uno and version 1.0 will be the reference versions of Arduino, moving forward. The Uno is the latest in a series of USB Arduino boards, and the reference model for the Arduino platform; for a comparison with previous versions, see the index of Arduino boards..

TECHNICAL DETAILS:

Dimensions (maximum):

- 75.14 x 53.51 x 15.08mm
- 2.96 x 2.1 x 0.59in
- 27.95g/0.98oz

Microcontroller	ATmega328
Operating Voltage	5V
Digital I/O Pins	14 (of which 6 provide PWM output)
Analog Input Pins	6
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328) of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328)
EEPROM	1 KB (ATmega328)
Clock Speed	16 MHz

CHAPTER 4

Arduino Programming Guidelines

This chapter gives the guidelines for Arduino Programming.

ARDUINO PROGRAMMING

STRUCTURE:

The basic structure of the arduino programming language is fairly simple and in runs in at least two parts. These two required parts, or functions, enclose block of statements [6].

```
void setup()
{
        statements;
}
void loop()
{
        statements;
}
```

Where setup() is the preparation, loop() is the execution. Both functions are required for the program to work.

The setup function should follow the declaration of any variables at the very beginning of the program. It is the first function to run in the program, is run only once, and is used to set pinMode or initialize serial communication.

The loop function follows next and includes the code to be executed continuously-reading inputs, triggering outputs, etc. This function is the core of all Arduino programs and does the bulk of the work.

setup()

The setup function is called once when the program starts. Use it to initialize pin modes or begin serial. It must be included in the program even if there are no statements to run.

```
void setup()
{
        pinMode(pin,OUTPUT);                    //sets the pin as output pin
}
```

loop()

the loop function() does precisely what its name suggests, loops consecutively, allowing the program to change,respond, and control the Arduino board.

```
void loop()
{
        digitalWrite(pin, HIGH);            // turns 'pin' on
        delay(1000);                        // stops for 1 second
        digitalWrite(pin, LOW);             // turns 'pin' off
        delay(1000);                        // stops for 1 second

}
```

pinMode()

used in void setup() to configure a specified pin to behave either as an INPUT or an OUTPUT.

```
pinMode(pin, OUTPUT)                        // sets 'pin' as OUTPUT pin
```

Arduino digital pins default to inputs, so they do not need to be explicitly declared as inputs by using pinMode(). Pins configured ass INPUT are said to be in a high-impedence state.

There are also convinient 20KΩ pullup resistors built into the Atmega chip that can be accessed from software. These built-in pullup resistors are accessed in the following manner:

```
pinMode(pin, INPUT);                        // set 'pin' to input
digitalWrite(pin, HIGH);                    // turn on pullup resistors
```

Pullup resistors would normally be used for connecting inputs like switches. In the above example it does not convert pin to output, it is merely a method for activating the internal pull-ups.

digitalRead(pin)

Reads the value from a specified digital pin with the result either HIGH or LOW. the pin can be specified as either a variable or constant(0-13).

```
value = digitalRead(Pin);                    // sets 'value' equal to the input pin
```

digitalWrite(pin, value)

Outputs either logic level HIGH or LOW at (turns on or off) a specified digital pin. The pin can be specified as either a variable or constant(0-13).

```
digitalWrite(pin, HIGH);                     // sets 'pin' to high
```

delay(ms)

Pauses the program for the amount of time as specified in milliseconds, where 1000 equals 1 second.

```
delay(1000);                                 // waits for 1 second
```

millis()

Returns the number of milliseconds since the Arduino board began running the current program as an unsigned long value.

```
value = millis();                            // sets 'value' equal to millis()
```

Serial.begin(rate)

Opens serial port and sets the baud rate for serial data transmission. The typical baud rate for communicating with the computer is 9600 although other speeds are supported.

```
void setup()
{
  Serial.begin(9600);                        // opens serial port
}                                            // sets data rate to 9600 bps
```

Serial.println(data)

Prints data to the serial port, followed by an automatic carraige return and line feed. This command takes the same form as Serial.print(), but is easier for reading data on the Serial Monior.

Serial.println(analogValue); **// sends the value of 'analogValue'**

CHAPTER 5

RESULTS

Program in matlab

Program in arduino

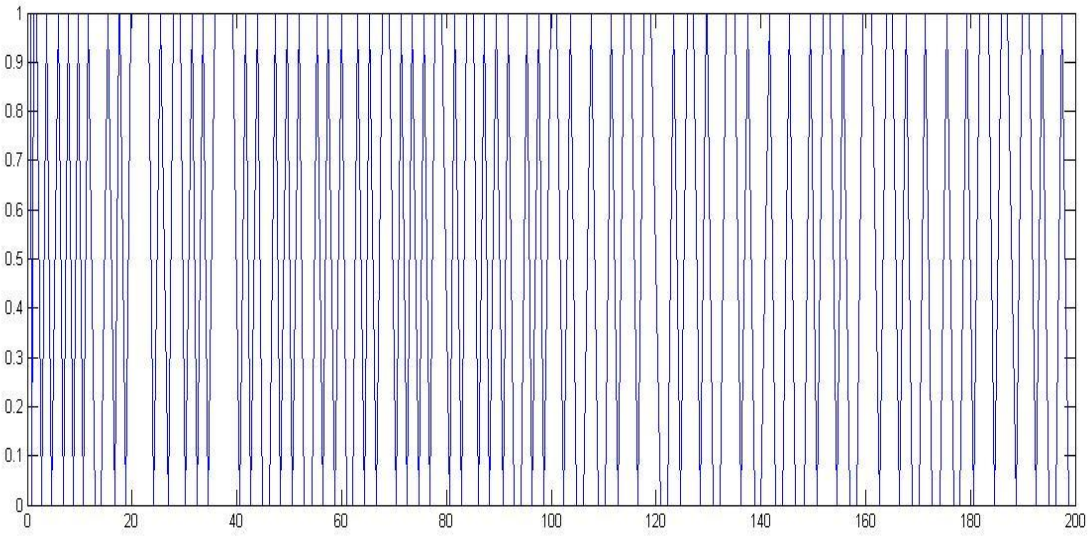
6.1 Program in matlab:

```
a= [0.00 0.00 0.31 0.31 0.31 0.63 0.63 0.94 0.94 1.26 1.26 1.26 1.88 2.83 3.77 4.71 5.97 6.91
7.85 8.79 9.73 10.68 11.62 12.87 14.13 15.39 16.64 17.58 18.84 19.78 21.04 21.98 23.24 24.18
25.43 27.00 27.95 29.20 30.14 31.40 32.34 33.60 34.54 35.80 37.05 37.99 39.25 40.51 41.76
42.70 43.96 44.90 46.16 47.41 48.36 49.61 50.55 51.81 52.75 54.32 55.26 56.52 57.46 58.72
59.97 60.92 62.17 63.11 64.37 65.31 66.57 67.82 69.08 70.34 71.28 72.53 73.48 74.73 75.67
76.93 77.87 79.13 80.70 81.64 82.90 83.84 85.09 86.04 87.29 88.23 89.49 90.75 91.69 92.94
94.20 95.46 96.40 97.65 98.60 99.85 101.11 102.36 103.62 104.88 106.13 107.70 108.96 110.21
111.47 112.73 113.98 115.24 116.49 117.75 119.01 120.89 122.15 123.40 124.66 125.91 127.17
128.43 129.68 130.94 132.19 133.45 135.02 136.28 137.53 138.79 140.04 141.61 142.87 144.13
145.38 146.64 148.21 149.46 150.72 151.98 153.23 154.49 155.74 157.00 158.26 159.51 161.08
162.65 163.91 165.16 166.42 167.68 168.93 170.19 171.44 172.70 174.27 175.53 176.78 178.04
179.29 180.55 181.81 183.38 184.63 185.89 187.14 188.71 189.97 191.23 192.48 193.74 194.99
196.25 197.51 198.76 200.02 201.59 202.84 204.41 205.67 206.93 208.18 209.44 210.69 211.95
213.21 214.78 216.03 217.29 218.54 219.80 221.06 222.31 223.57 225.14 226.39 227.96 229.22
230.48 231.73 232.99 234.24 235.50 236.76 238.01 239.27 240.52 242.09 243.35 244.61 246.18
247.43 248.69 249.94 251.20 252.46 253.71 255.28 256.54 257.79 259.05 260.31 261.56 262.82
264.07 265.33
];
pin= zeros(size(a));
l= length(a);
p=1;
s=0;
for x= 1:l
z= sin(a(x));
if (a(x)>(p+1.25664))
   p=p+1.25664;
   s=s+1.25664;
end
if (a(x)<= p)
   if (z> (4.775*(a(x)-s))-1.5)
      pin(x)=1;
   else pin(x)=0;
   end
end
if ((a(x)>p) && (a(x)<p+0.62832))
   if (z>(-4.775*(a(x)-s)+4.5))
      pin(x)=1;
   else
      pin(x)=0;
   end
end
end
plot (a,pin);
```

PLOT:

↑

P
W
M



Time →

6.2 Program in Arduino:

```
unsigned long time= millis();
unsigned long t0=time;
const float pi= 3.14;
float y,x,a,p=1,s=0,z;

void setup()
{
  Serial.begin(19200);
  pinMode(13,OUTPUT);
 }
void loop()
{
  y=millis();
  x=y-t0;
  a= 2*pi*50*x/1000;
  Serial.println(a);
  z= sin(a);

  if (a>p+1.25664)
   {
     p=p+1.25664;
     s=s+1.25664;
   }

  if (a<= p)
   {
       if (z> (4.775*(a-s))-1.5)
       {
         digitalWrite(13, HIGH);
       }
       else {
         digitalWrite(13, LOW);
    }}

  if ((a>p) && (a<p+0.62832))
    {
       if (z>(-4.775*(a-s)+4.5))
       {
         digitalWrite(13, HIGH);
       }
       else {
         digitalWrite(13, LOW);
       }}}
```

CHAPTER 6

CONCLUSION

CONCLUSION

By carrying out this project, I concluded that generating exact SPWM signals by following the basic definition, i.e, by comparing a sinusoid (modulating) signal and a triangular signal, is mind storming.

I developed a program which gives discrete values of sinusoidal signal and compare it with corresponding value of triangular waveform and give the HIGH/LOW level of PWM accordingly. The ticks of the time, I fetched by internal oscillator of the Arduino.

Interpolation can be done to reduce the time interval between consecutive discrete values and thus increasing the accuracy quotient of the output. Internal oscillator not being very accurate as it depends on the voltage supply consistency and temperature, external oscillators can be used with Arduino board to get a better result.

REFERENCES

[1] Rashid. M.H, “Power Electronics circuits devices and applications”, Prentice Hall of India, New Delhi. 2004

[2] Beechner, T., Sun, J.: Asymmetric interleaving—a new approach to operating parallel converters. In: Proc. of the IEEE Energy Conversion Congress and Exposition, San Jose, California, USA, pp. 99–105 (2009)

[3] Iyer, V. M., and Vinod J.. "Low-frequency dc bus ripple cancellation in single phase pulse-width modulation inverters." IET Power Electronics 8.4: 497-506. 2015.

[4] Bimbhra .P.S "Power Electronics", Khanna Publishers, New Delhi, 2003.

[5] Durfee W.,” Arduino Microcontroller Guide’, University of Minnesota, 2011.

[6] Evans B. W., “Arduino Programming Notebook”, http://playground.arduino.cc/uploads/Main/arduino_notebook_v1-1.pdf, August 2007.