CS-301

MICROPROCESSOR BASED SYSTEM DESIGN

COMPLEX ENGINEERING PROBLEM (CEP)

GROUP MEMBERS:

MUHAMMAD SAAD(CS-092)

MUHAMMAD ANAS(CS-094)

ZOBIA KHAN(CS-100)

H.SAMARA MOHSIN(CS-088)

BATCH: 2017

Submission date:23RD Jan,2020

NED UNIVERSITY OF ENGINEERING & TECHNOLOGY

DEPARTMENT OF COMPUTER & INFORMATION SYSTEMS

ABOUT PROJECT:

This is a Microcontroller based application in which speed of fan is controlled using PWM logic.This system would automatically take decision whether to turn fan on or off also to increase or decrease the speed of fan over the range specified in code.

MAJOR COMPONENTS:

- AVR Atmega162
- ADC 0808
- Potentiometer(variable resistor)
- ADC Fan Motor

OTHER COMPONENTS:

- LCD(16x2)
- switch

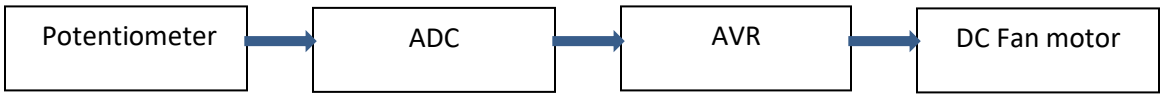
SOFTWARE REQUIRMENTS:

- Atmel Studio 7.0(source code in programming language C#)
- Proteus 8 Professional

MICROCONTROLLER SIMMULATION COMPONENTS:

- ATmega162
- ADC 0808
- 3WATT100R
- BUTTON
- CLOCK
- DIPSW-3
- FAN-DC
- L293D
- LED-BLUE
- POT
- POT-HG
- RES-AVR

BLOCK DIGRAM:



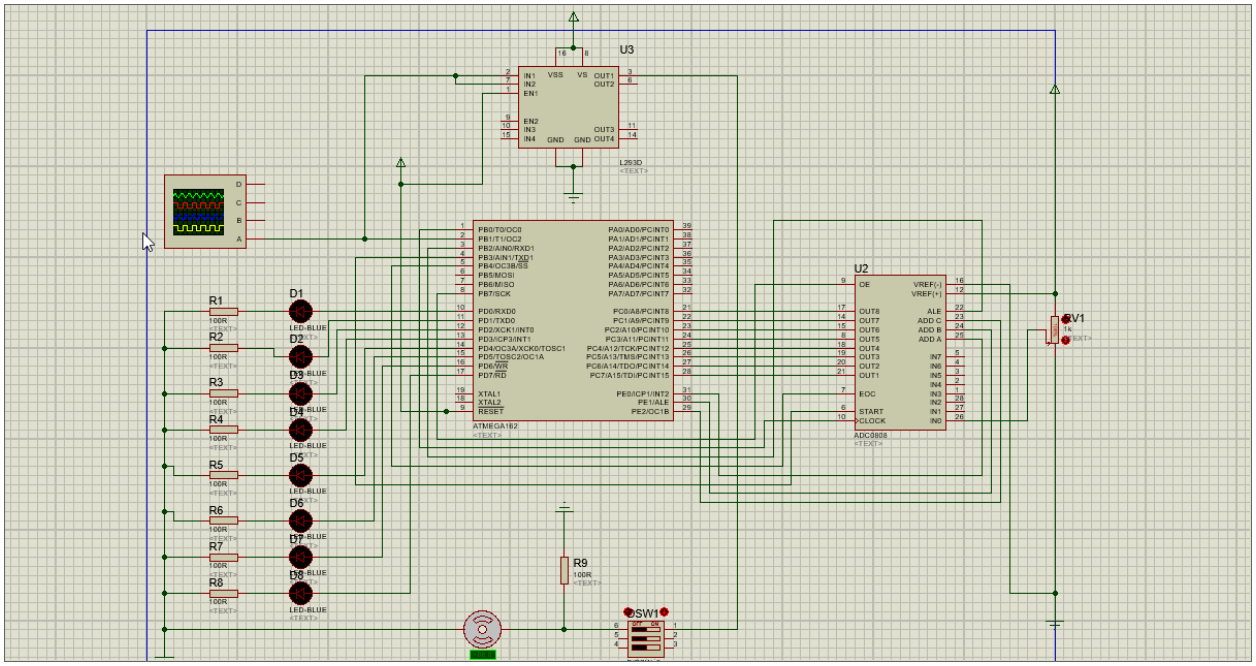
PWM Technique:

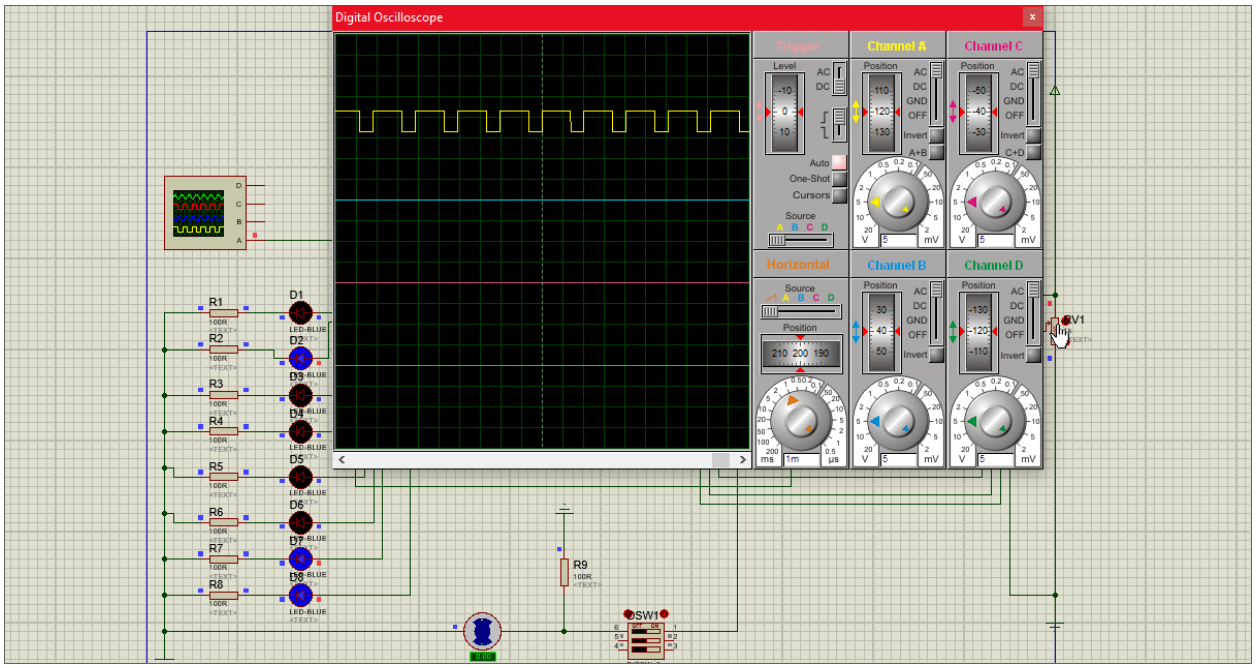
Pulse Width Modulation, or PWM, is a technique for getting analog results with digital means. Digital control is used to create a square wave, a signal switched between on and off. This on-off pattern can simulate voltages in between full on (5 Volts) and off (0 Volts) by changing the portion of the time the signal spends on versus the time that the signal spends off. The duration of "on time" is called the pulse width. To get varying analog values, we change, or modulate, that pulse width. If we repeat this on-off pattern fast enough with an LED for example, the result is as if the signal is a steady voltage between 0 and 5v controlling the brightness of the LED. In the graphic below, the green lines represent a regular time period. This duration or period is the inverse of the PWM frequency.

Working:

The project works on the principle of Analog to Digital Conversion. The Analog data from the Potentiometer is given to the analog to digital converter ADC0808. This analog signal is given to the ADC, which converts the analog values to digital values. As per the changes in the temperature, the output of the ADC is generated. The digital output of the ADC is given to Microcontroller which will control the speed of DC fan motor. A switch is there if user wants to keep fan off in all situations he or she can press it.

SIMMULATION:





SOURCE CODE:

```
#include <avr/io.h>
#define F_CPU 1000000UL
#include <util/delay.h>
void adc(void);
void read_adc(void);


int main(void)
{
    DDRA = 0b00000000;
    DDRC = 0b00000000;
    DDRE = 0xFF;
    DDRB = 0b00101111;
//  DDRB = 0x10011111;
    DDRD = 0xFF;
    TCCR2 |= (1<<WGM21)|(1<<WGM20)|(1<<COM21)|(1<<COM20)|(1<<CS21);
    TCCR0 |=(1<<WGM01)|(1<<COM00)|(1<<CS00);
    OCR0 = 2;
    PORTB = 0B00010010;
    while(1)
    {
        adc();
    }
}
void adc(void)
{
    PORTE = 0x00;
    read_adc();
}
void read_adc(void)
{
    char num = 0;
    PORTB = PORTB | 0b00001110;
    _delay_ms(100);
    PORTB = PORTB&0b11110011;
    while((PINB&0b00010010) == 0b00010010);
    while((PINB&0b00010010) == 0b00000010);
    PORTB = PORTB | (0b00100010);
    num = PINC;
    _delay_ms(200);
    PORTB = PORTB&(0b11011111);
    PORTD = num;
    if(num == 0b00000000)
    {
        OCR2=255;
    }
    else if((num > 0b00000000) & (num <= 0b00110011))
    {
        OCR2=200;
    }
    else if((num > 0b00110011) & (num <= 0b01100110))
    {
        OCR2=160;
    }
    else if((num > 0b01100110) & (num <= 0b10011001))
    {
        OCR2=127;
    }
    else if((num > 0b10011001) & (num <= 0b11001100))
    {
        OCR2=80;
    }
    else if((num > 0b11001100) & (num <= 0b11111111))
    {
        OCR2=30;
    }
        else if(num == 0b11111111)
        {
    else if((num > 0b11001100) & (num <= 0b11111111))
    {
        OCR2=30;
    }
        else if(num == 0b11111111)
        {
            OCR2=0;
        }
}
```