NAME: VISHNU KUMAR

PROJECT

Abstract: A pc-based data acquisition system that performs conversion of analog signal to digital data and the digital data to analog signal is interfaced to a pc to implement the functions of a measurement and control instrumentation applications. The data acquisition is a process where raw data from the physical world are collected, processed and stored and used. Data acquisition systems are widely used in the industries.

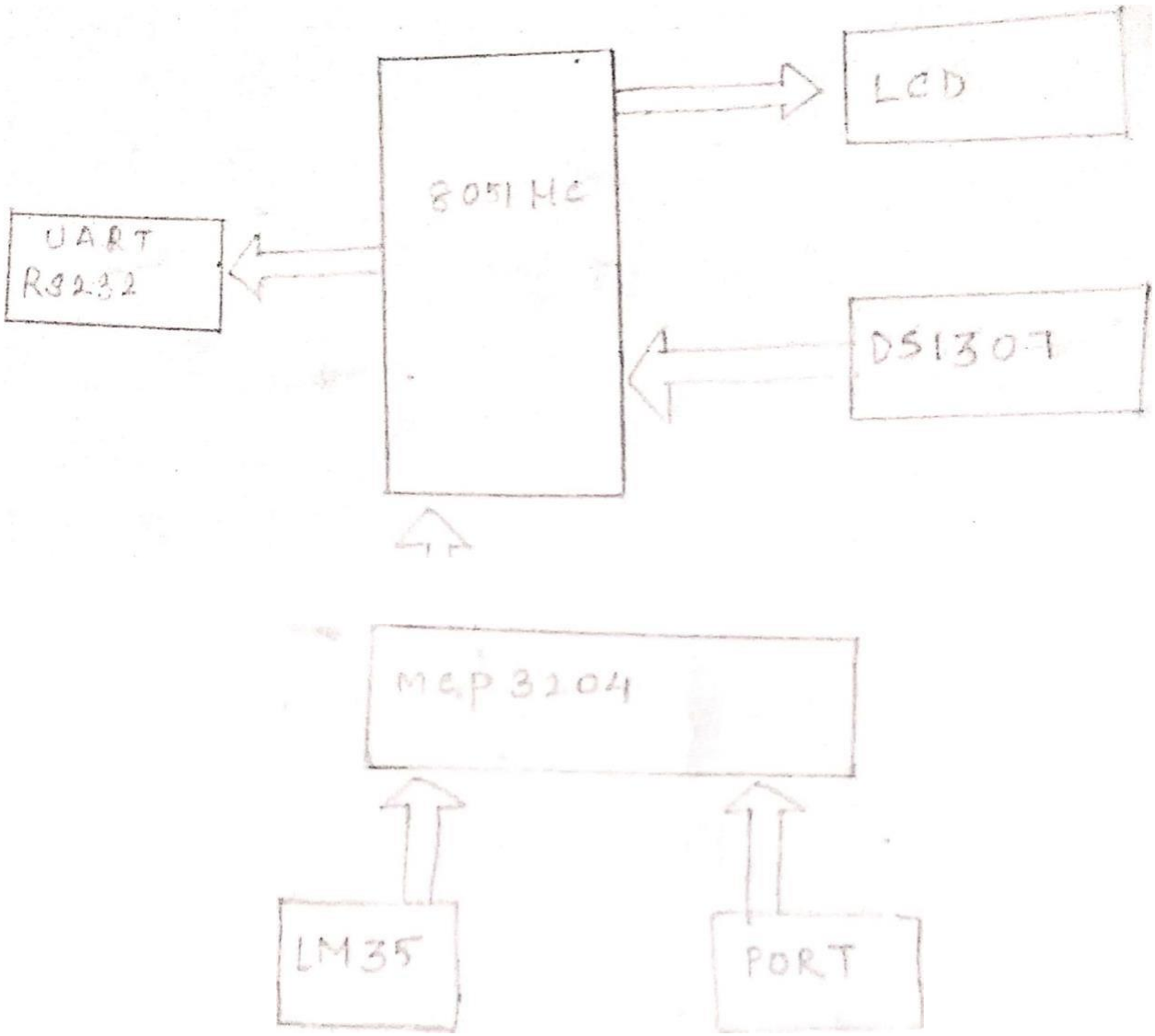
It is the process of measuring an electrical or physical phenomenon such as pressure, or sound with a computer. A data acquisition (DAQ) system consists of sensors, DAQ measurement hardware, and a computer with programmable software. It improves the efficiency and reliability of process or machinery. Steel mills, utilities, or a research lab have some data acquisition device that silently monitors some parameter.

These collects data can be used to improve efficiency, ensure reliability or ensure that machinery operates safely.

Software Required: Keil and Proteus.

Hardware Required: Microcontroller (8051), LCD (16*2), Jumper wires, DS1307(RTC), mcp3204(ADC), Temperature sensor (LM35), Port, UART-RS232, Breadboard, Power supply, Crystal, Resistances(1k)

Block Diagram:



CODE

UART:

```
#include<reg51.h>



/*void uart_init(void );
unsigned char uart_rx();
void uart_tx(unsigned char);
void uart_str(unsigned char *);
void uart_num(int);
void uart_float(float);*/




void uart_init(void )
{
        SCON=0x40; // Uart Mode 1
        TMOD = 0x20; // Timer 1 Mode 2 is Selected
        TH1=TL1=253;
        TR1=1;
        REN=1;
}
void uart_tx(unsigned char d)
{
        SBUF=d; // Tx is started
        while(TI==0);
                TI=0;
```

```
}



unsigned char uart_rx()
{
        while(RI==0);
                RI=0;
        return SBUF;
}



void uart_str(unsigned char *s)
{
        while(*s)
                uart_tx(*s++);
}



void uart_num(int n)
{
        char arr[5],j=0;
        if(n==0)
                uart_tx('0');
        else
        {
                if(n<0)
```

```
		{
			uart_tx('-');
			n=-n;
		}
	while(n>0)
	{
		arr[j++]= n%10;
		n=n/10;
	}
	for(--j;arr[j];j--)
		uart_tx(arr[j]+48);
	}
}



void uart_float(float f)
{
	int t=f;
	uart_num(t);
	uart_tx('.');
	t=(f-t)*1000;
	uart_num(t);
}
```

DELAY:

```
#ifndef DELAY_H
#define DELAY_H



void delay_us(unsigned int us_count)
{
	while(us_count!=0)
	{
		us_count--;
	}
}

void delay_ms(unsigned int ms_count)
{
	while(ms_count!=0)
	{
		delay_us(112);
		ms_count--;
	}
}

void delay_sec(unsigned char sec_count)
{
	while(sec_count!=0)
	{
		delay_ms(1000);
		sec_count--;
```

```
        }
}


#endif
```

LCD:

```
#include<reg51.h>
#include  "delay.c"
#include "lcd.h"

#define databus P2

sbit rs = P1^0;
sbit rw = P1^1;
sbit en = P1^2;

#define LCDMaxLines 2
#define LCDMaxChar 16
#define LineOne 0x80
#define LineTwo 0xC0

#define BlankSpace ' '

void LCD_EnablePulse()
{
        en = 0;
        en =1;
        delay_us(100);
        en =0;
        delay_us(10);
```

```
}

void LCD_Init(void)
{
        databus = 0x00;
        rs =0;
        rw =0;

        databus = 0x30;
        LCD_EnablePulse();
        delay_ms(100);

        databus = 0x30;
        LCD_EnablePulse();
        delay_us(500);

        databus = 0x30;
        LCD_EnablePulse();
        delay_us(500);

        databus = 0x20;
        LCD_EnablePulse();
        delay_us(200);

        /*LCD_CmdWrite(0x38);*/LCD_CmdWrite(0x2E);
        LCD_CmdWrite(0x01);
        LCD_CmdWrite(0x10);
        LCD_CmdWrite(0x0c);
```

```
	LCD_CmdWrite(0x81);
}

/*void LCD_CmdWrite(unsigned char cmd)
{
	databus = cmd;
	rs = 0;
	rw = 0;
	LCD_EnablePulse();

	delay_ms(1);
}

void LCD_DataWrite(unsigned char dat)
{
	databus = dat;
	rs = 1;
	rw = 0;
	LCD_EnablePulse();

	delay_ms(1);
}*/


void LCD_CmdWrite(char cmd)
{
	databus = (cmd &0xf0);
	rs =0;
	rw =0;
```

```
        LCD_EnablePulse();

        delay_us(500);

        databus = ((cmd<<4)&0xf0);
        rs =0;
        rw =0;
        LCD_EnablePulse();

        delay_us(500);
}

void LCD_DataWrite(char dat)
{
        databus = (dat &0xf0);
        rs =1;
        rw =0;
        LCD_EnablePulse();

        delay_us(500);

        databus = ((dat<<4)&0xf0);
        rs =1;
        rw =0;
        LCD_EnablePulse();

        delay_us(500);
}
```

```
void LCD_Clear()
{
	LCD_CmdWrite(0x01);
	LCD_CmdWrite(LineOne);
}

void LCD_GoToLineOne()
{
	LCD_CmdWrite(LineOne);
}

void LCD_GoToCmdTwo()
{
	LCD_CmdWrite(LineTwo);
}

void LCD_GoToXY(unsigned char row,unsigned char col)
{
	char pos;
	if(row<LCDMaxLines)
	{
		pos = LineOne | (row << 6);

		if(col<LCDMaxChar)
			pos = pos+col;

		LCD_CmdWrite(pos);
```

```
	}
}

void LCD_DisplayString( char *string_ptr)
{
	while(*string_ptr)
		LCD_DataWrite(*string_ptr++);
}

void LCD_DisplayNumber(int num)
{
	char arr[5], j=0;
	if(num == 0)
		LCD_DataWrite('0');
	else
	{

		if(num<0)
		{
			LCD_DataWrite('-');
			num =-num;
		}
		while(num>0)
		{
			arr[j++] = num%10;
			num = num/10;
		}
		for(--j;arr[j];j--)
		{
```

```
                LCD_DataWrite(arr[j]+48);
            }
        }
}

void LCD_DisplayFloatNumber(float float_num)
{
        int t = float_num;
        LCD_DisplayNumber(t);
        LCD_DataWrite('.');
        t= (float_num - t)*1000;
        LCD_DisplayNumber(t);
}


void LCD_ScrollMessage( char *msg_ptr)
{
        unsigned char i,j;
        LCD_CmdWrite(0x0c);
        for (i =0;msg_ptr[i];i++)
        {
            LCD_GoToLineTwo();

            for (j = 0;j<LCDMaxChar && msg_ptr[i+j];j++)
              LCD_DataWrite(msg_ptr[i+j]);

            for(j=j;j<LCDMaxChar;j++)
                LCD_DataWrite(BlankSpace);
```

```
                delay_ms(125);
        }
        LCD_CmdWrite(0x0E);
}
void LCD_DisplayRtcDate(char day, char month, char year)
{
        LCD_DataWrite(((day>>4) &0x0f)+ 0x30);
        LCD_DataWrite((day &0x0f)+ 0x30);
        LCD_DataWrite('/');

        LCD_DataWrite(((month>>4) &0x0f)+ 0x30);
        LCD_DataWrite((month &0x0f)+ 0x30);
        LCD_DataWrite('/');

        LCD_DataWrite(((year>4) &0x0f)+ 0x30);
        LCD_DataWrite((year &0x0f)+ 0x30);

}

void LCD_DisplayRtcTime(unsigned char hour,unsigned char min,unsigned char sec)
{
        LCD_DataWrite(((hour>>4) &0x0f)+ 0x30);
        LCD_DataWrite((hour &0x0f)+ 0x30);
        LCD_DataWrite(':');

        LCD_DataWrite(((min>>4) &0x0f)+ 0x30);
        LCD_DataWrite((min &0x0f)+ 0x30);
        LCD_DataWrite(':');
```

```
        LCD_DataWrite(((sec>>4) &0x0f)+ 0x30);
        LCD_DataWrite((sec &0x0f)+ 0x30);
}
```

I2C:

```
#ifndef _I2C_H
#define _I2C_H
void I2C_Clock(void);
void I2C_Start();
void I2C_Stop(void);
void I2C_write(unsigned char);
unsigned  char  I2C_read(void);
void I2C_Ack();
void I2C_NoAck();
#endif
#include<reg51.h>
#include"delay.c"

sbit SCL=P3^6;
sbit SDA=P3^7;
void I2C_Clock(void)
{
        delay_us(1);
        SCL=1;
        delay_us(1);
        SCL=0;
}
void I2C_Start()
{
        SCL=0;
        SDA=1;
```

```
        delay_us(1);
        SCL=1;
        delay_us(1);
        SDA=0;
        delay_us(1);
        SCL=0;
}
        void I2C_Stop(void)
        {
                SCL=0;
                delay_us(1);
                SDA=0;
                delay_us(1);
                SCL=1;
                delay_us(1);
                SDA=1;
        }
        void I2C_Write(unsigned char dat)
        {
                unsigned char i;
                for(i=0;i<8;i++)//loop and times
                {
                        SDA=dat & 0x80;//send bit by bit on SDA line
                        I2C_Clock();//generate Clock at SCL
                        dat=dat<<1;
                }
                SDA=1;//set SDa at last
        }
        unsigned char I2C_Read(void)
```

```
{
        unsigned char i,dat=0x00;
        SDA=1;//mark SDA as input
        for(i=0;i<8;i++)//loop and times
        {
                delay_us(1);
                SCL=1;//pull SCl high
                delay_us(1);
                dat=dat<<1;//dat is shifted each time
                dat=dat|SDA;
                SCL=0;
        }
        return dat;
}
void I2C_Ack()
{
        SDA=0;//pull SDA low to indicate
        I2C_Clock();
        SDA=1;//pull SDA back to high
}
void I2C_NoAck()
{
        SDA=1;
        I2C_Clock();
        SDA =1;
}
```

DS1307:

```
#define DS1307_ID 0xD0

#define SEC_ADDRESS 0x00
#define DATE_ADDRESS 0x04
#define CONTROL 0x07
#include<i2c.c>
#include<ds1307.h>

void DS1307_Init()
{
        I2C_Start();
        DS1307_Write(DS1307_ID);
        DS1307_Write(CONTROL);

        DS1307_Write(0x00);

        I2C_Stop();
}
void DS1307_Write(unsigned char dat)
{
        I2C_Write(dat);
        I2C_Clock();
}
unsigned char DS1307_Read()
{
```

```
	unsigned char dat;
	dat = I2C_Read();
	return(dat);
}

void DS1307_SetTime(unsigned char hh, unsigned char mm, unsigned char ss)
{
	I2C_Start();

	DS1307_Write(DS1307_ID);
	DS1307_Write(SEC_ADDRESS);

	DS1307_Write(ss);
	DS1307_Write(mm);
	DS1307_Write(hh);

	I2C_Stop();
}

void DS1307_SetData(unsigned char dd, unsigned char mm, unsigned char yy)
{
	I2C_Start();

	DS1307_Write(DS1307_ID);
	DS1307_Write(SEC_ADDRESS);

	DS1307_Write(dd);
	DS1307_Write(mm);
	DS1307_Write(yy);
```

```
        I2C_Stop();
}

void DS1307_GetTime(unsigned char *h_ptr, unsigned char *m_ptr, unsigned char *s_ptr)
{
        I2C_Start();

        DS1307_Write(DS1307_ID);
        DS1307_Write(SEC_ADDRESS);

        I2C_Stop();

        I2C_Start();
        DS1307_Write(0xD1);

        *s_ptr = DS1307_Read();         I2C_Ack();
        *m_ptr = DS1307_Read();         I2C_Ack();
        *h_ptr = DS1307_Read(); I2C_NoAck();

        I2C_Stop();
}

 void DS1307_GetDate(unsigned char *d_ptr, unsigned char *m_ptr, unsigned char *y_ptr)
{
        I2C_Start();

        DS1307_Write(DS1307_ID);
```

```
        DS1307_Write(DATE_ADDRESS);

        I2C_Stop();

        I2C_Start();
        DS1307_Write(0xD1);

        *d_ptr = DS1307_Read();         I2C_Ack();
        *m_ptr = DS1307_Read();         I2C_Ack();
        *y_ptr = DS1307_Read(); I2C_NoAck();

        I2C_Stop();
}
```

MCP3204:

```
#include<reg51.h>
sbit cs = P1^3;
sbit mosi = P1^4;
sbit miso = P1^5;
sbit clk = P1^6;
float spi_adc_read(bit d1,bit d0)
{
        unsigned int adc_val = 0;
        char j;
        cs = 0;
        clk = 0;mosi = 1;clk = 1;
        clk =0;mosi =1; clk = 1;
        clk =0;clk = 1;
        clk = 0; mosi = d1; clk =1;
        clk = 0; mosi = d0; clk = 1;
        clk = 0; clk = 1;
        clk = 0; clk = 1;

        for (j = 11;j>=0;j--)
        {
                clk = 0;
                adc_val = adc_val | miso;
                adc_val =  adc_val<<1;
                clk = 1;
```

```
        }
        cs = 1;
        return ((adc_val * 5.0)/4096.0);
}
```

MAIN:

```
#include<reg51.h>
#include<lcd.c>
#include <ds1307.c>
#include <mcp3204.c>
#include <uart.c>
void main()
{
        unsigned char sec , min, hour, day, month, year;

        LCD_Init();

        DS1307_Init();

        uart_init();

        /*DS1307_SetTime(0x12,0x19,0x40);
        DS1307_SetDate(0x27, 0x06, 0x20);*/

        LCD_DisplayString("Time: ");

        LCD_GoToXY(1,0);
        LCD_DisplayString("volt: ");
```

```
        while(1)
        {
                DS1307_GetTime(&hour, &sec, &min);

                LCD_GoToXY(0,6);
                LCD_DisplayRtcTime(hour, sec, min);

                DS1307_GetDate(&day, &month, &year);

                LCD_GoToXY(1,6);
                LCD_DisplayFloatNumber(spi_adc_read(0,0));

                uart_tx('Time:');
                uart_num(hour);
                uart_tx(':')
                uart_num(sec);
                uart_tx(':')
                uart_num(min);
                uart_tx('\n');
                uart_float(spi_adc_read(0,0));
                uart_str("Volt");
                uart_tx("\n");
                uart_float(spi_adc_read(0,1);
                uart_str('C');



        }
}
```