**DC Ammeter**

Cherla Pradyumna Reddy (U101116FEC279)

Archana Gujar(U101116FEC158)

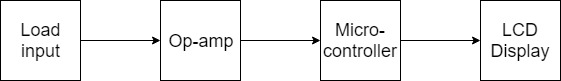
**Range:** 0-1024 mA, resolution 1mA

**Abstract:** In this project we have aimed to build a digital ammeter with the help of PIC microcontroller. There are no direct methods available to measure current. One of the simplest way to measure is using Ohm’s Law. By finding out the voltage applied and resistance between two nodes we can calculate the amount of current flowing through it. The same methodology is used in this project. We have used PIC16F877a for computational purpose and 8-bit LCD display to view the current value.

**Project Requirements:**

1. Microcontroller: PIC16F877a
2. Op-amp: LM324
3. 8 Bit LCD Display
4. Resistors: 1k (2), 3k (1)
5. Software: PCW.exe
6. Debugger: MPLAB ICE2

**Project Layout:**



**Steps to Follow:**

1. Sense voltage across load resistor and give it as input to the op-amp.
2. Amplify the voltage by op-amp.
3. Provide output of op-amp to the microcontroller.
4. Microcontroller will compute current with the help of known load resistance and divide it by gain factor of op-amp.
5. Display the value on LCD.

**Diagram:**

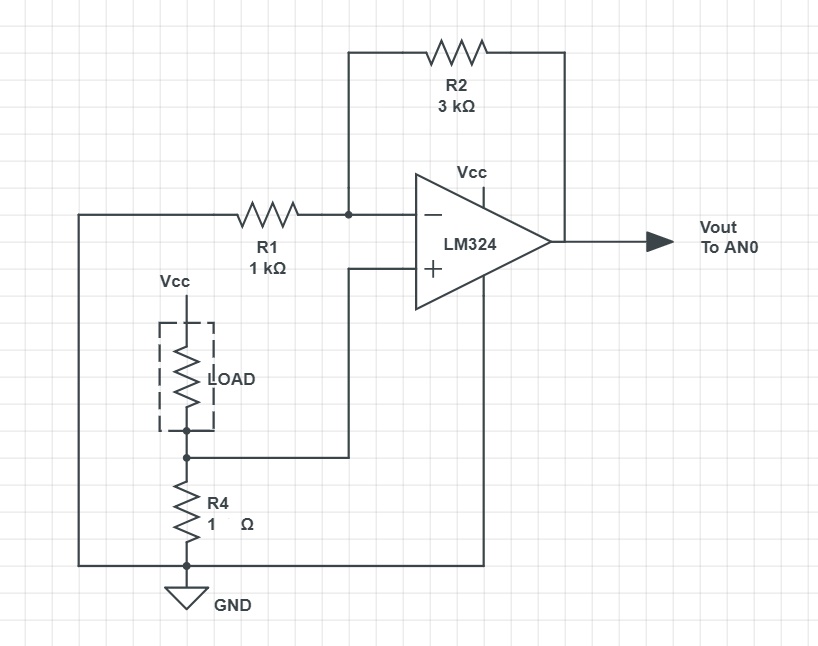


Fig. Op-amp

**Calculations:**

Vout = Vin()

= Vin(1+)

Vout = 4(Vin) ……… (equ.1)

Op-Amp is used in order to amplify the input voltage as it is very low to measure. The output of op-amp LM324 is connected to the input line AN0 of pic. Voltage value is read at this pin and based on the comparator present inside pic, compared and converted into digital format. But this value is then divided by 4 times to calculate the input voltage (refer equ.1).

**Code:**

/\*=========================================

8-BIT LCD DRIVER FOR PIC16F73 CCSC

==========================================\*/

///////////////////////////////////////////////////////////////////

// LCD 16x2

//

//////////////////////////////////////////////////////////////////

#include<16f877a.h>

#device ADC=10

#include <stdlib.h>

#use delay(clock=20000000)

#fuses HS,NOWDT,PROTECT

#zero\_ram

#use rs232(baud=9600,xmit=PIN\_B7,rcv=PIN\_B5)

#define PORT\_D                  3

#define NCHAR\_PER\_LINE          16              // max char numbers per line

#define LCD\_RS                  PIN\_B4

#define LCD\_E                   PIN\_B2

#define LCD\_DAT                 PORT\_D

///////////////////////////////////////////////// output()

//lcd data bus output

void output(int8 port, int8 dat)

{

                output\_D(dat);

}//end output()

//////////////////////////////////////////////// lcd\_write\_cmd()

//

void lcd\_write\_cmd(int8 cmd)

{

         delay\_us(400);

        output\_low(LCD\_RS);

        output(LCD\_DAT, cmd);

        output\_high(LCD\_E);

        delay\_us(400);

        output\_low(LCD\_E);

   ////getc();

}//end lcd\_write\_cmd()

/////////////////////////////////////////////// lcd\_write\_dat()

//

void lcd\_write\_dat(int8 dat)

{

         delay\_us(400);

        output\_high(LCD\_RS);

        output(LCD\_DAT, dat);

        output\_high(LCD\_E);

        delay\_us(400);

        output\_low(LCD\_E);

}//end lcd\_write\_dat()

//////////////////////////////////////////// lcd\_init()

//

void lcd\_init(void)

{

        output\_low(LCD\_E);              // Let LCD E line low

        lcd\_write\_cmd(0x38);            // LCD 16x2, 5x7, 8bits data

        delay\_ms(15);

        lcd\_write\_cmd(0x01);            // Clear LCD display

        delay\_ms(10);

        lcd\_write\_cmd(0x0f);            // Open display & current

        delay\_ms(10);

        lcd\_write\_cmd(0x06);            // Window fixed

        delay\_ms(10);

}//end lcd\_init()

///////////////////////////////////////// lcd\_display\_char()

//

void lcd\_display\_char(int8 line, int8 pos, int8 ch)

{

        int8 tmp;

        line = (line==0) ? 0 : 1;

        pos  = (pos >NCHAR\_PER\_LINE) ? NCHAR\_PER\_LINE : pos;

        tmp = 0x80 + 0x40\*line + pos;

   //printf("tmp=%xReady?\n\r",tmp);

   //printf("ch=%xReady?\n\r",ch);

   ////getc();

        lcd\_write\_cmd(tmp);

        lcd\_write\_dat(ch);

   //delay\_ms(40); //delay removed

}//end lcd\_display\_char()

/////////////////////////////////////////// lcd\_display\_str()

//

void lcd\_display\_str(int8 line, char str[])

{

     int8 i;

        for(i=0; i<NCHAR\_PER\_LINE; i++)

            {

                lcd\_display\_char(line, i, ' ');

                }

        for(i=0; i<NCHAR\_PER\_LINE; i++)

                {

                if(str[i] == '\0') break;

                lcd\_display\_char(line, i, str[i]);

                }

   delay\_ms(500);

}//end lcd\_display\_str()

/\*=======================================\*/

void main(void)

{

   setup\_adc(ADC\_CLOCK\_INTERNAL);

   setup\_adc\_ports(ALL\_ANALOG);

   set\_adc\_channel(0);

   while(1)

   {

      int8 i;

      int num;

      char string1[5]="mA";

      char string[10];

      char string2[16]= "DIGITAL AMMETER";

      int32 x;

      x = x/4;

      x= read\_adc();

      itoa(x,10,string);

      strcat(string,string1);

     lcd\_display\_str(0, string2);

     lcd\_display\_str(1, string);

     }

}//end main()

**Output:**

