Relative Humidity Meter

Report

Microcontrollers Project

CPEN213

Presented by Nicolas Ghandour A2211342

to Dr. Nicolas Haddad

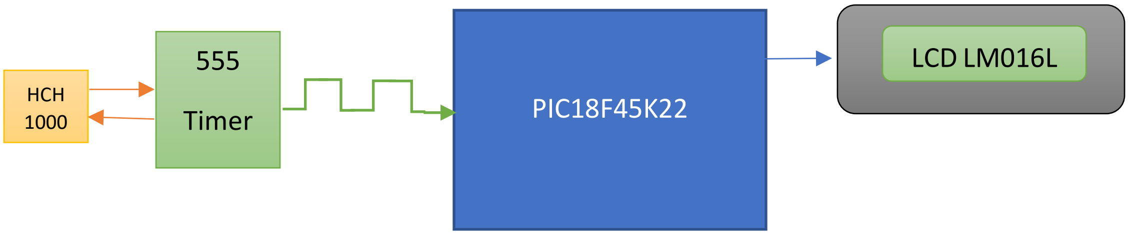
Faculty of Engineering

University of Balamand

20- 5- 2024

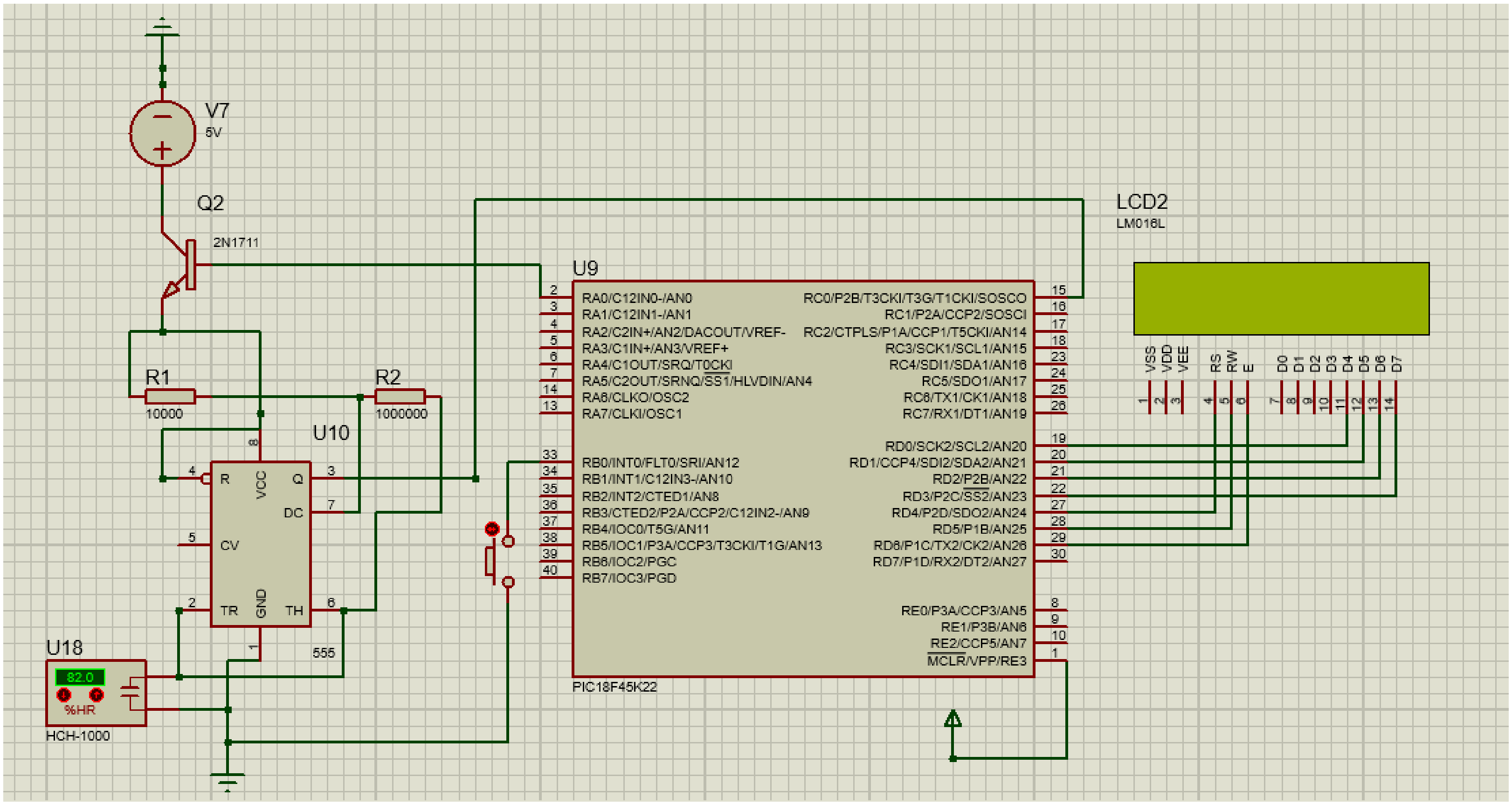
I - Abstract:

The objective of this project was to design a device that can read and display relative humidity. The approach implemented in this project was to build a circuit containing the 555 Timer and the HCH-1000 sensor that outputs a square signal. Then, the microcontroller (PIC18K45F22) will first record the frequency of that signal and then calculate the relative humidity and display it on the LCD.



This device can widely be used in meteorology, pharmaceuticals, microelectronics, food processing, and many other applications.

II – Hardware Implementation:



The first step was to make the 555 Timer work in the astable mode that generates a square wave by connecting to the 555 Timer the HCH-1000 capacitor and some resistors in a specific manner[1]. Then, having fed the output of the 555 Timer the external clock of Timer1 (RC0) of the microcontroller and having set Timer0 to count one second, Timer1 will hold the value of the frequency of the square signal. Finally, the microcontroller will use the frequency obtained along with other parameters (discussed in the next Section) to calculate the relative humidity and display it on the LCD. The pushbutton connected to RB0 serves as an ON/OFF pushbutton of the device. The BJT whose base is connected to RA0 serves as a switch that connects Vcc to the 555 Timer when the device is ON and breaks this connection otherwise in order to minimize power consumption.

[1] https://www.ti.com/lit/ds/symlink/lm555.pdf?ts=1715379285914&ref_url=https%253A%252F%252Fwww.google.com%252F page 10

III – Basic Theory:

Having calculated the frequency of the square signal, we use a formula found in the datasheet of the 555 Timer[2] relating the frequency to the measured capacitance of the capacitor:

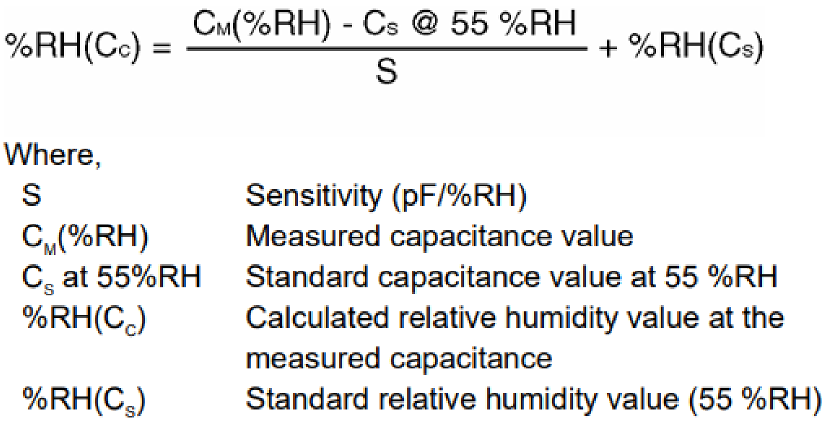
F= 1.44 / (Cm.(2.R2+R1)) (F in Hz, Cm in F, R1 R2 in Ω)

So Cm= 1.44 / (F.(2.R2+R1))

For the values of $R1 = 10K\Omega$ and $R2 = 1\ M\Omega$, the simplified formula will be:

Cm= 716.417 / F (Cm in nF, F in Hz)

Now we use another equation found in the datasheet of the HCH-1000[3] relating its capacitance to the relative humidity:



$$\%RH(C_C) = \frac{C_M(\%RH) - C_S \text{ @ } 55\ \%RH}{S} + \%RH(C_S)$$

Where,

S	Sensitivity (pF/%RH)
$C_M(\%RH)$	Measured capacitance value
C_S at 55%RH	Standard capacitance value at 55 %RH
$\%RH(C_C)$	Calculated relative humidity value at the measured capacitance
$\%RH(C_S)$	Standard relative humidity value (55 %RH)

For the values of $\%RC(Cs) = 50$, $S = 0.6$, Cs at $55\%RH = 330$, the final simplified formula is:

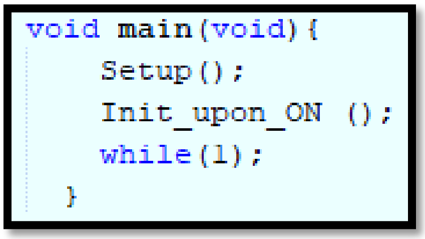
%RH (Cc)= Cm*1000 /0.6 – 495 (Cm in nF)

[2] https://www.ti.com/lit/ds/symlink/lm555.pdf?ts=1715379285914&ref_url=https%253A%252F%252Fwww.google.com%252F page 11

[3] https://www.covionline.it/wp-content/uploads/2016/12/HCH-1000-Foglio-di-applicazione.pdf page 2

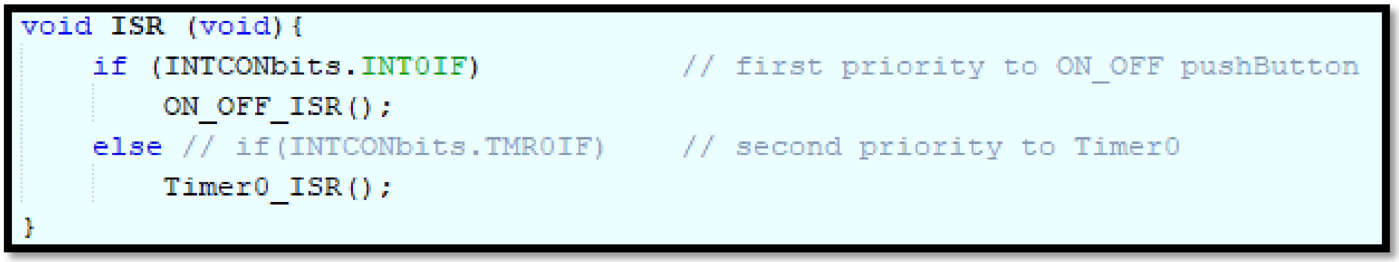
IV – Software implementation:

The **main()** function of the code is the following:



```
void main(void){
    Setup();
    Init_upon_ON ();
    while(1);
  }
```

The **Setup()** subroutine is responsible for the initializations needed to be done for once and for all whereas the **Init_upon_ON ()** function is responsible for the initializations needed to be done every time we turn ON the device. After finishing the initializations, we will wait for interrupts.



```
void ISR (void){
    if (INTCONbits.INT0IF)          // first priority to ON_OFF pushButton
        ON_OFF_ISR();
    else // if(INTCONbits.TMR0IF)   // second priority to Timer0
        Timer0_ISR();
}
```

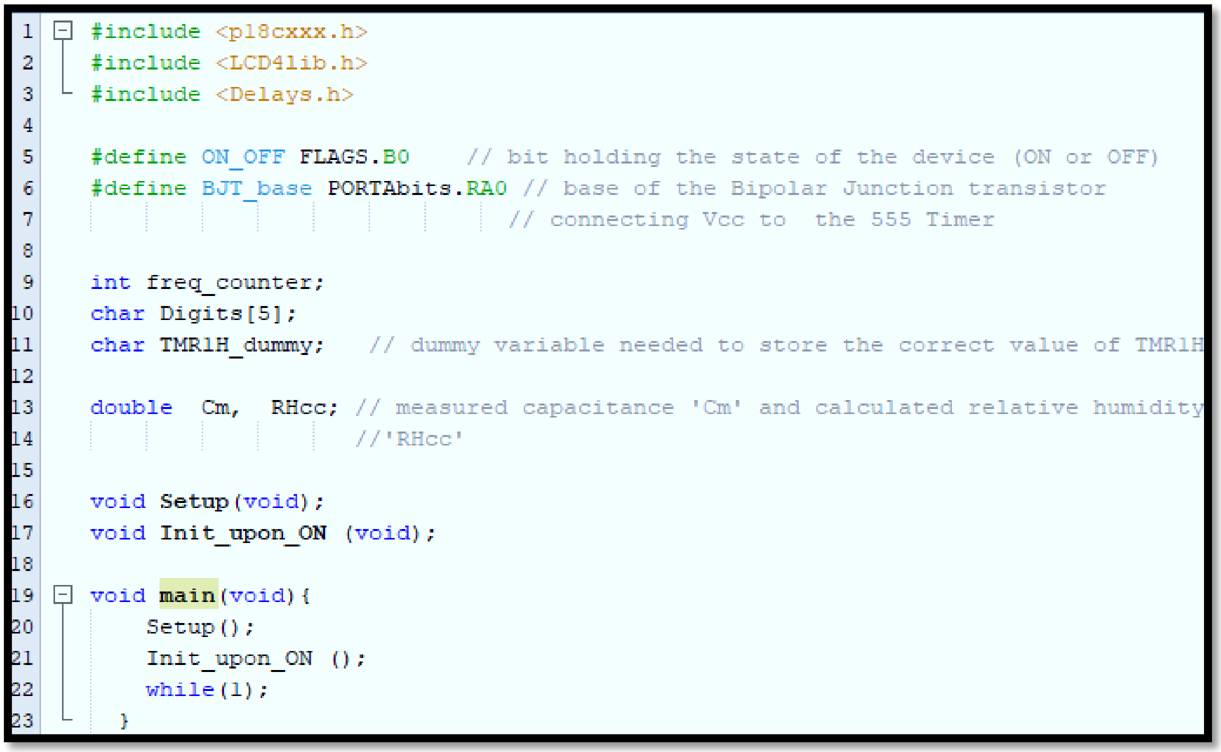
The first interrupt service routine **ON_OFF_ISR ()** will display an exit message, disconnect the 555 Timer from Vcc and put the microcontroller in a Sleep mode when the device is OFF; When the device is reactivated, **ON_OFF_ISR()** will call **Init_upon_ON()** and then program will proceed to where it left off in the "**while**(1);" in the **main().**

The second interrupt service routine is **Timer0_ISR ()**. Having set Timer0 in 16-bit mode, with a prescale of 64 and a load of 65536 – 15625, Timer0 will interrupt the **main()** every one second, which is the duration needed by Timer1 to determine the frequency; therefore, the calculation of RH and its display on the LCD is done in **Timer0_ISR ()**.

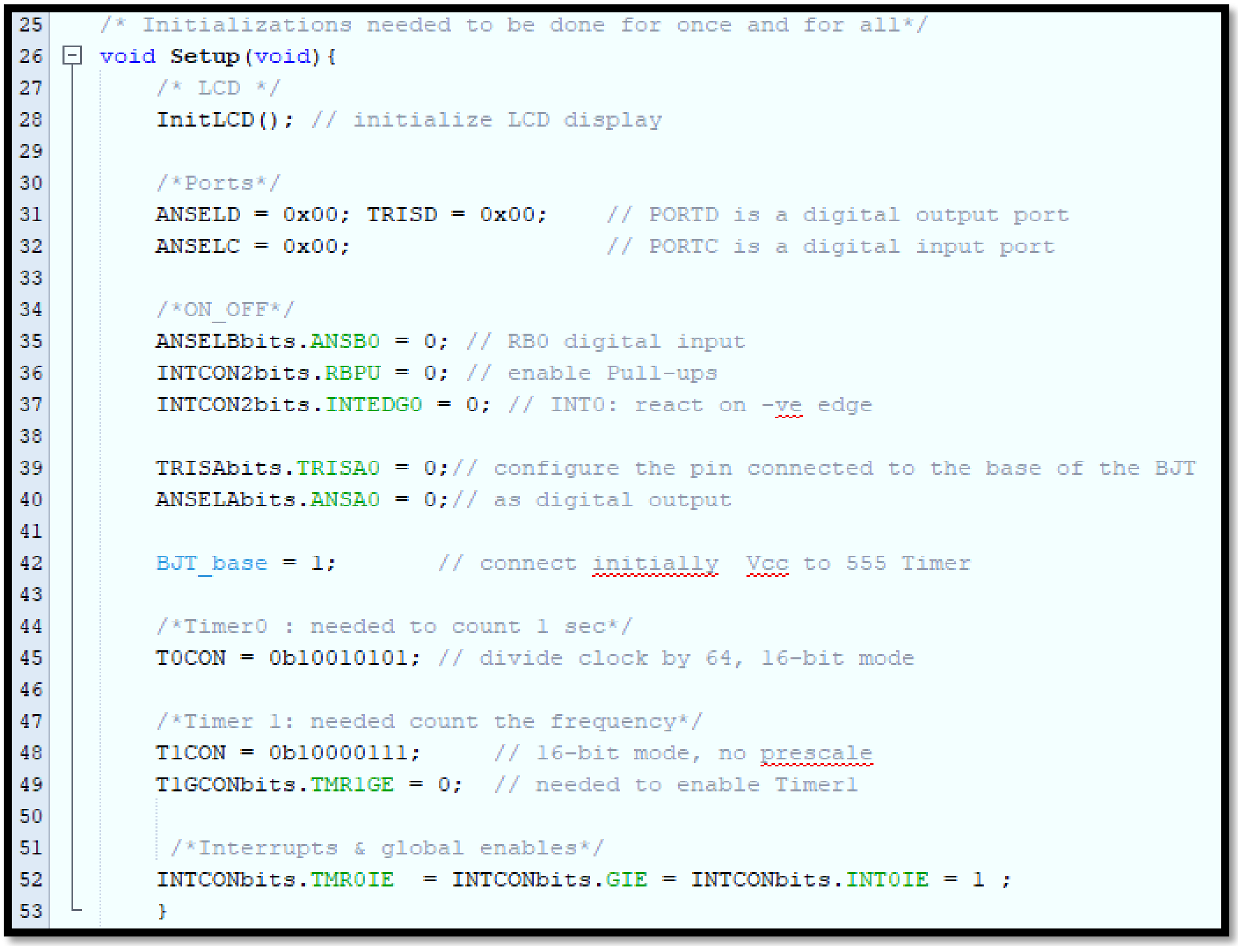
Moreover, having also done some calculations over the range of possible frequencies[4] given the values of R1, R2 and the parameters of the HCH-1000, we could deduce that Timer1 in 16-bit mode with no prescale is enough to store any possible frequency; thus, Timer1 doesn't need to enable its interrupt since it will never exceed the frequency 2364.4 Hz (which is less than the max value that can be written in 16 bits (65536)).

[4] This is done by writing the frequency F as a function of %RH using the equations found in **Section III.** The equation becomes F = 1 194028.3/(%RH + 495). For the minimum value of %RH (=10), we get the maximum value of F (which is 2364.4 Hz)

C Code[5]

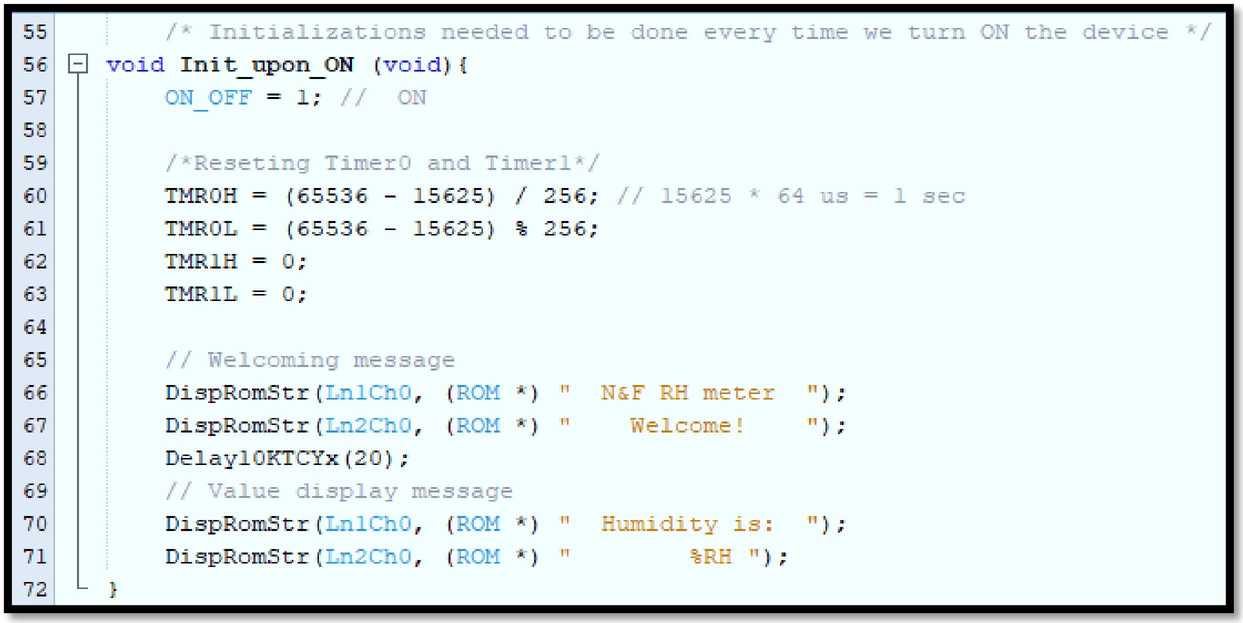


```c
#include <p18cxxx.h>
#include <LCD4lib.h>
#include <Delays.h>

#define ON_OFF FLAGS.B0    // bit holding the state of the device (ON or OFF)
#define BJT_base PORTAbits.RA0 // base of the Bipolar Junction transistor
                              // connecting Vcc to  the 555 Timer

int freq_counter;
char Digits[5];
char TMR1H_dummy;   // dummy variable needed to store the correct value of TMR1H

double  Cm,  RHcc; // measured capacitance 'Cm' and calculated relative humidity
                   //'RHcc'

void Setup(void);
void Init_upon_ON (void);

void main(void){
    Setup();
    Init_upon_ON ();
    while(1);
  }
```

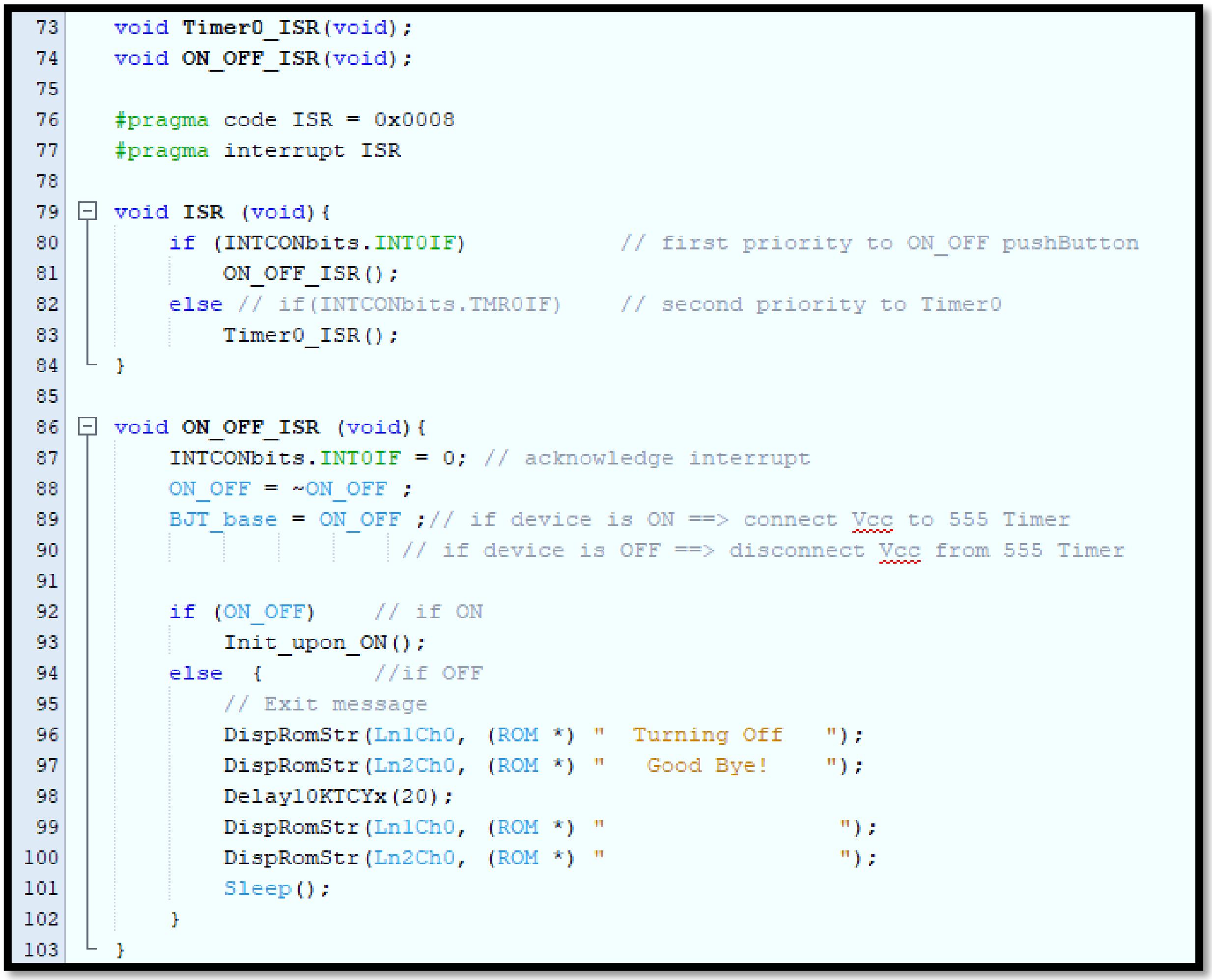


```c
/* Initializations needed to be done for once and for all*/
void Setup(void){
    /* LCD */
    InitLCD(); // initialize LCD display

    /*Ports*/
    ANSELD = 0x00; TRISD = 0x00;    // PORTD is a digital output port
    ANSELC = 0x00;                  // PORTC is a digital input port

    /*ON_OFF*/
    ANSELBbits.ANSB0 = 0; // RB0 digital input
    INTCON2bits.RBPU = 0; // enable Pull-ups
    INTCON2bits.INTEDG0 = 0; // INT0: react on -ve edge

    TRISAbits.TRISA0 = 0;// configure the pin connected to the base of the BJT
    ANSELAbits.ANSA0 = 0;// as digital output

    BJT_base = 1;       // connect initially  Vcc to 555 Timer

    /*Timer0 : needed to count 1 sec*/
    T0CON = 0b10010101; // divide clock by 64, 16-bit mode

    /*Timer 1: needed count the frequency*/
    T1CON = 0b10000111;     // 16-bit mode, no prescale
    T1GCONbits.TMR1GE = 0;  // needed to enable Timer1

     /*Interrupts & global enables*/
    INTCONbits.TMR0IE  = INTCONbits.GIE = INTCONbits.INT0IE = 1 ;
    }
```

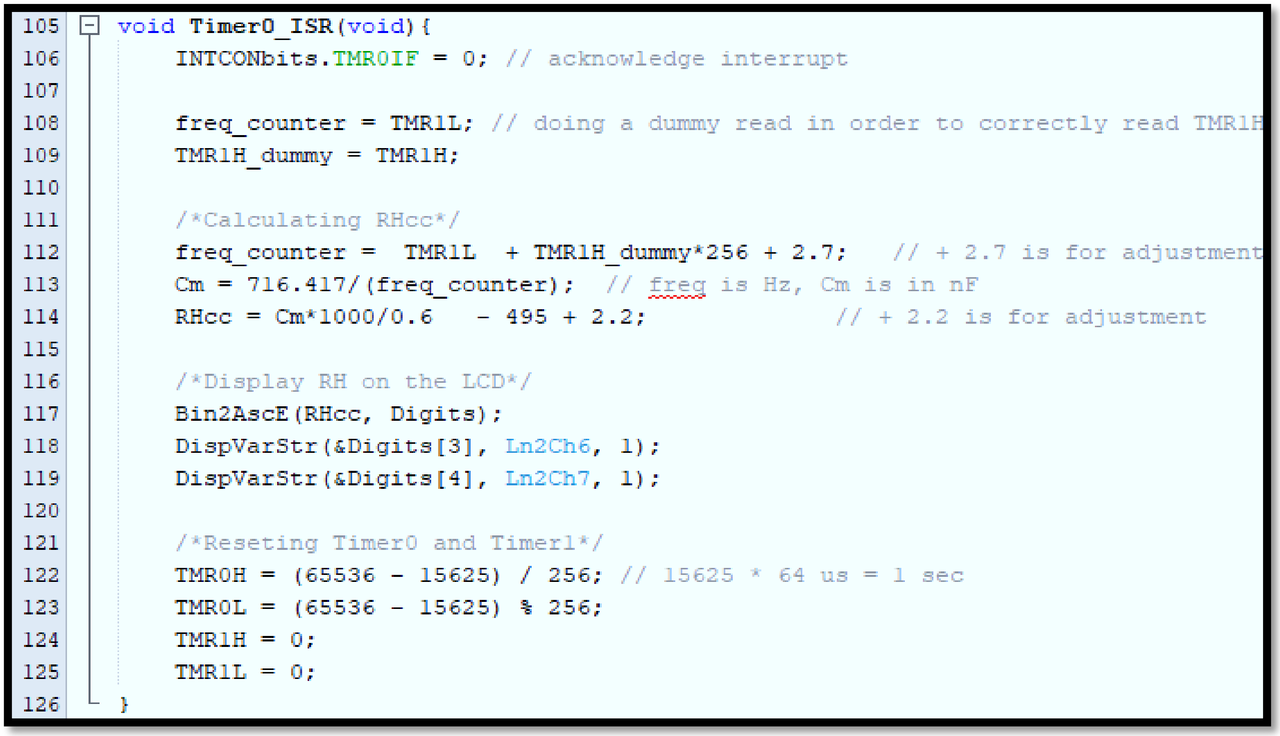
[5] See **Appendix B** if you want to copy the code and try it



```
    /* Initializations needed to be done every time we turn ON the device */
void Init_upon_ON (void){
    ON_OFF = 1; //  ON

    /*Reseting Timer0 and Timer1*/
    TMR0H = (65536 - 15625) / 256; // 15625 * 64 us = 1 sec
    TMR0L = (65536 - 15625) % 256;
    TMR1H = 0;
    TMR1L = 0;

    // Welcoming message
    DispRomStr(Ln1Ch0, (ROM *) "  N&F RH meter  ");
    DispRomStr(Ln2Ch0, (ROM *) "    Welcome!    ");
    Delay10KTCYx(20);
    // Value display message
    DispRomStr(Ln1Ch0, (ROM *) "  Humidity is:  ");
    DispRomStr(Ln2Ch0, (ROM *) "        %RH ");
}
```



```
void Timer0_ISR(void);
void ON_OFF_ISR(void);

#pragma code ISR = 0x0008
#pragma interrupt ISR

void ISR (void){
    if (INTCONbits.INT0IF)          // first priority to ON_OFF pushButton
        ON_OFF_ISR();
    else // if(INTCONbits.TMR0IF)   // second priority to Timer0
        Timer0_ISR();
}

void ON_OFF_ISR (void){
    INTCONbits.INT0IF = 0; // acknowledge interrupt
    ON_OFF = ~ON_OFF ;
    BJT_base = ON_OFF ;// if device is ON ==> connect Vcc to 555 Timer
                     // if device is OFF ==> disconnect Vcc from 555 Timer

    if (ON_OFF)     // if ON
        Init_upon_ON();
    else   {        //if OFF
        // Exit message
        DispRomStr(Ln1Ch0, (ROM *) "  Turning Off   ");
        DispRomStr(Ln2Ch0, (ROM *) "   Good Bye!    ");
        Delay10KTCYx(20);
        DispRomStr(Ln1Ch0, (ROM *) "                ");
        DispRomStr(Ln2Ch0, (ROM *) "                ");
        Sleep();
    }
}
```



```
void Timer0_ISR(void){
    INTCONbits.TMR0IF = 0; // acknowledge interrupt

    freq_counter = TMR1L; // doing a dummy read in order to correctly read TMR1H
    TMR1H_dummy = TMR1H;

    /*Calculating RHcc*/
    freq_counter =  TMR1L  + TMR1H_dummy*256 + 2.7;   // + 2.7 is for adjustment
    Cm = 716.417/(freq_counter);  // freq is Hz, Cm is in nF
    RHcc = Cm*1000/0.6   - 495 + 2.2;              // + 2.2 is for adjustment

    /*Display RH on the LCD*/
    Bin2AscE(RHcc, Digits);
    DispVarStr(&Digits[3], Ln2Ch6, 1);
    DispVarStr(&Digits[4], Ln2Ch7, 1);

    /*Reseting Timer0 and Timer1*/
    TMR0H = (65536 - 15625) / 256; // 15625 * 64 us = 1 sec
    TMR0L = (65536 - 15625) % 256;
    TMR1H = 0;
    TMR1L = 0;
}
```

V – Conclusion:

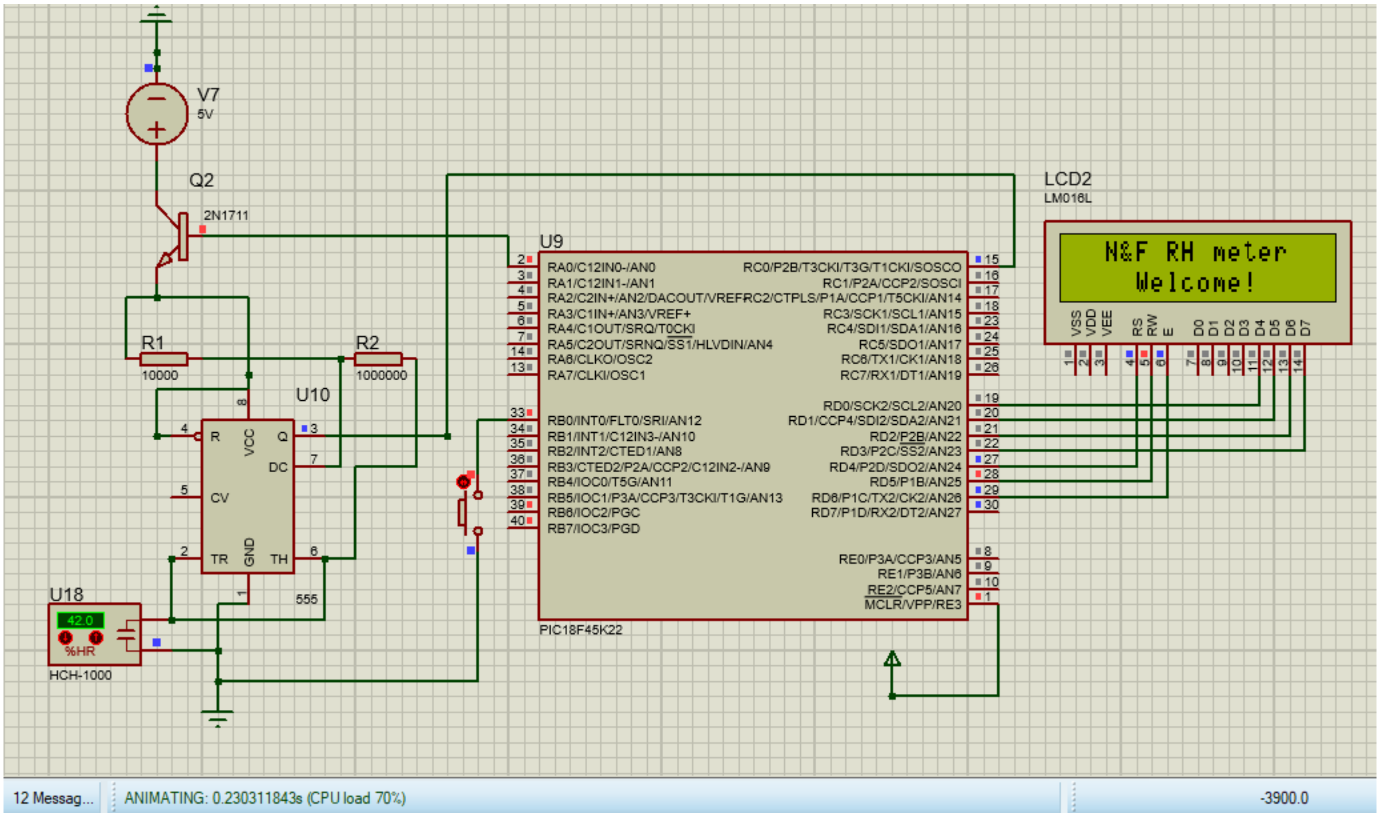
Eventually, the design worked successfully as carefully planned. Some improvements can be done in the future to improve the precision of the relative humidity by considering floating number calculations in the design.

References

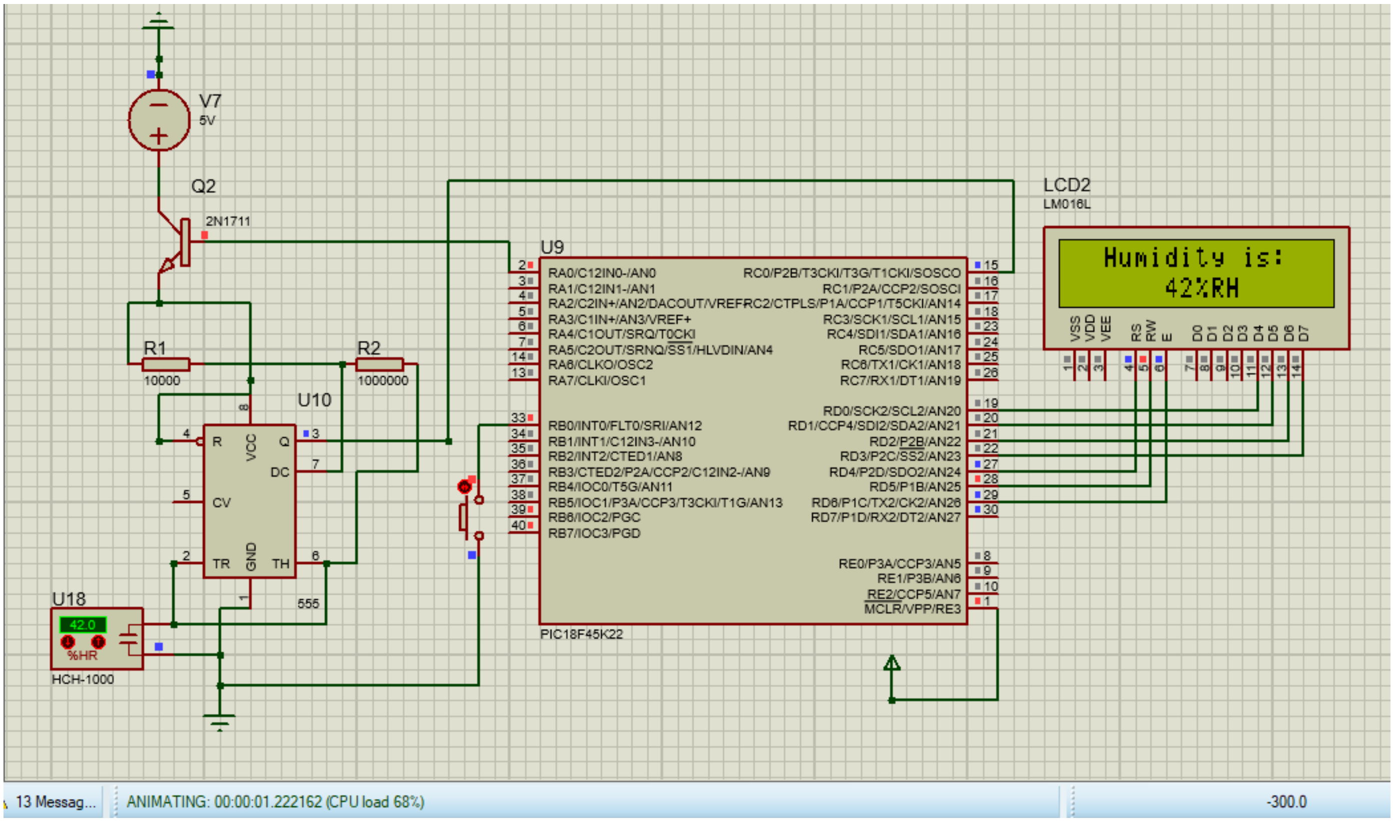
https://www.ti.com/lit/ds/symlink/lm555.pdf?ts=1715379285914&ref_url=https%253A%252F%252Fwww.google.com%252F **555-Timer Datasheet**

https://www.covionline.it/wp-content/uploads/2016/12/HCH-1000-Foglio-di-applicazione.pdf **HCH-1000 Datasheet**

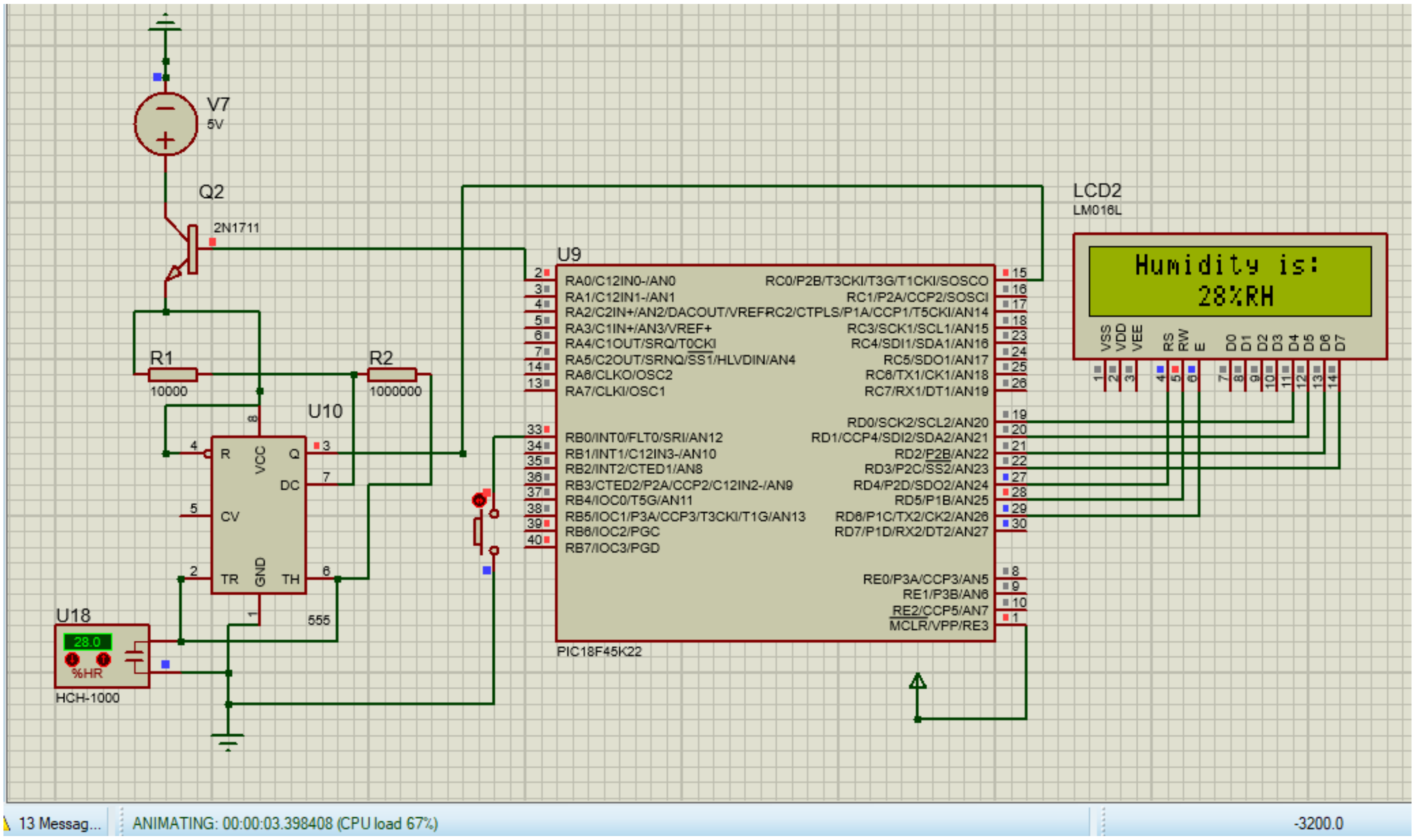
Appendix A: Simulations



12 Messag... | ANIMATING: 0.230311843s (CPU load 70%) | -3900.0

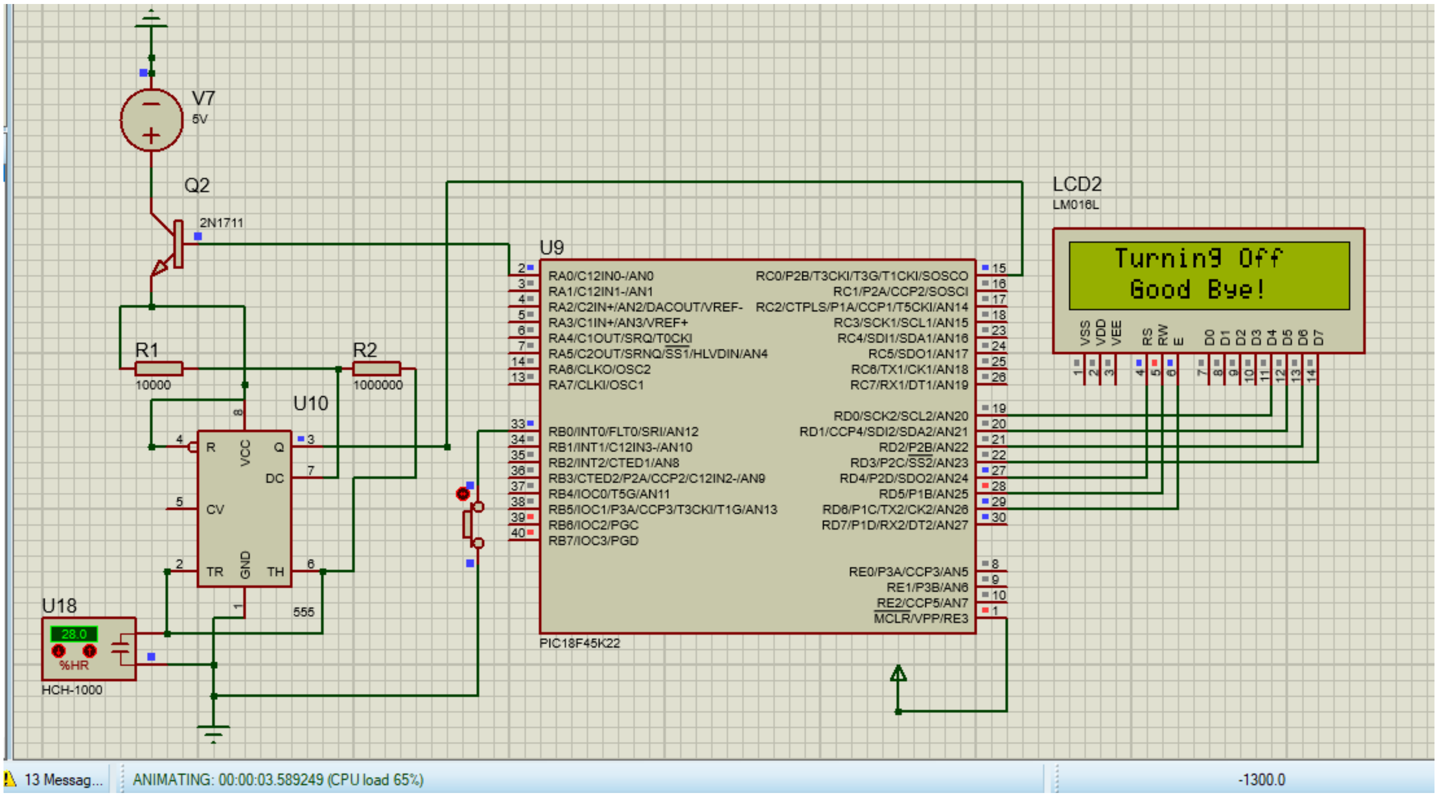


13 Messag... | ANIMATING: 00:00:01.222162 (CPU load 68%) | -300.0



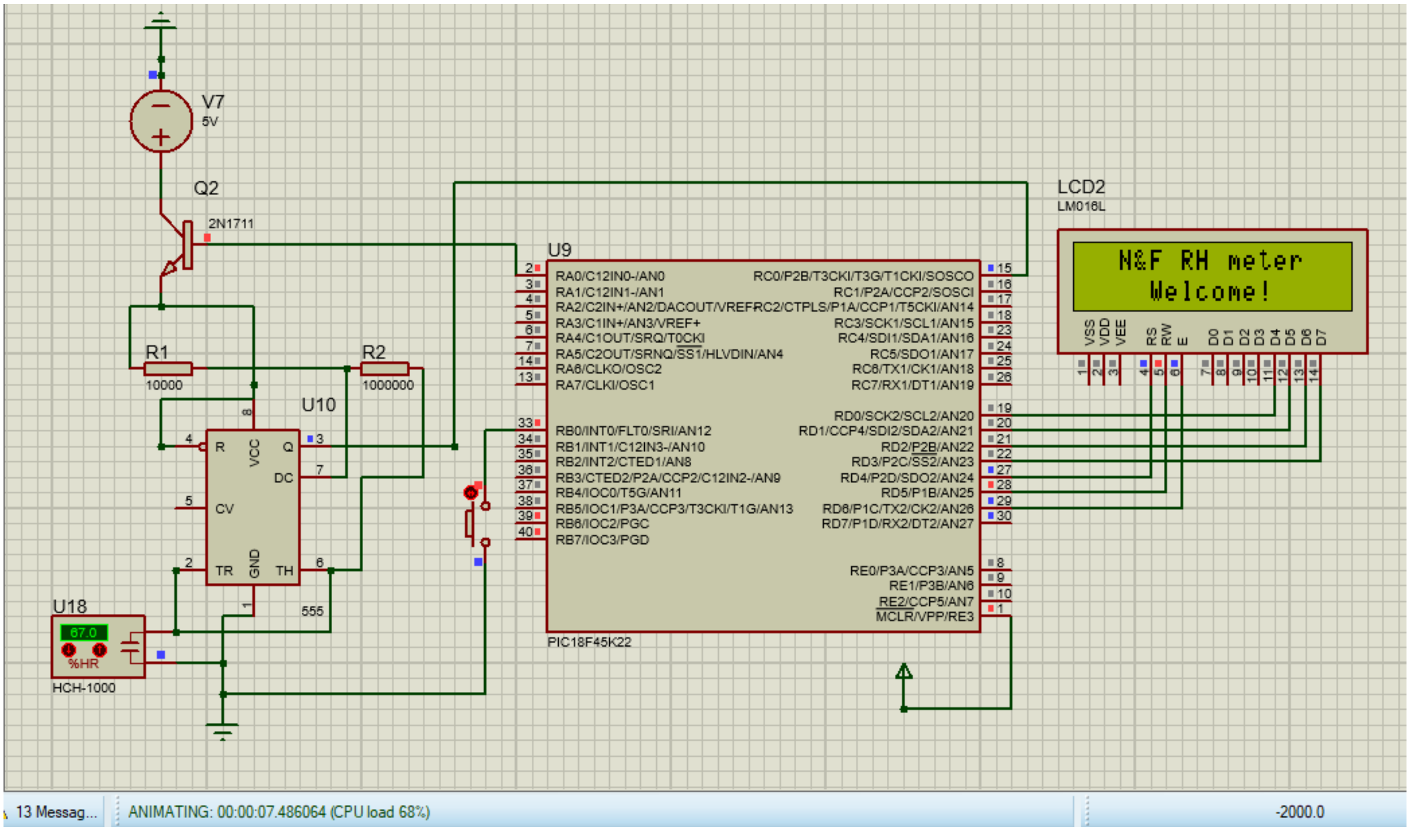
Humidity is:
28%RH

13 Messag... ANIMATING: 00:00:03.398408 (CPU load 67%) -3200.0



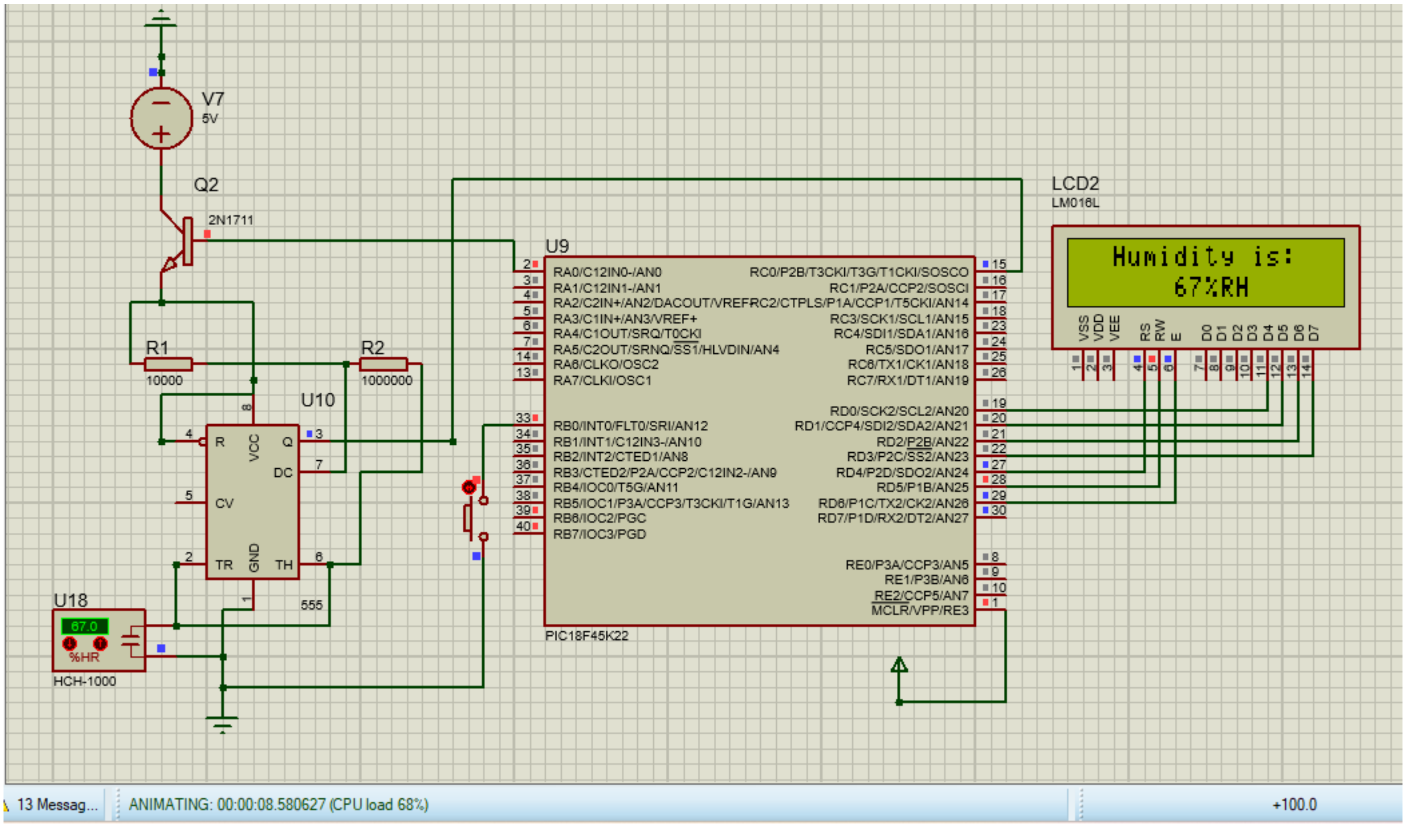
Turning Off
Good Bye!

13 Messag... ANIMATING: 00:00:03.589249 (CPU load 65%) -1300.0



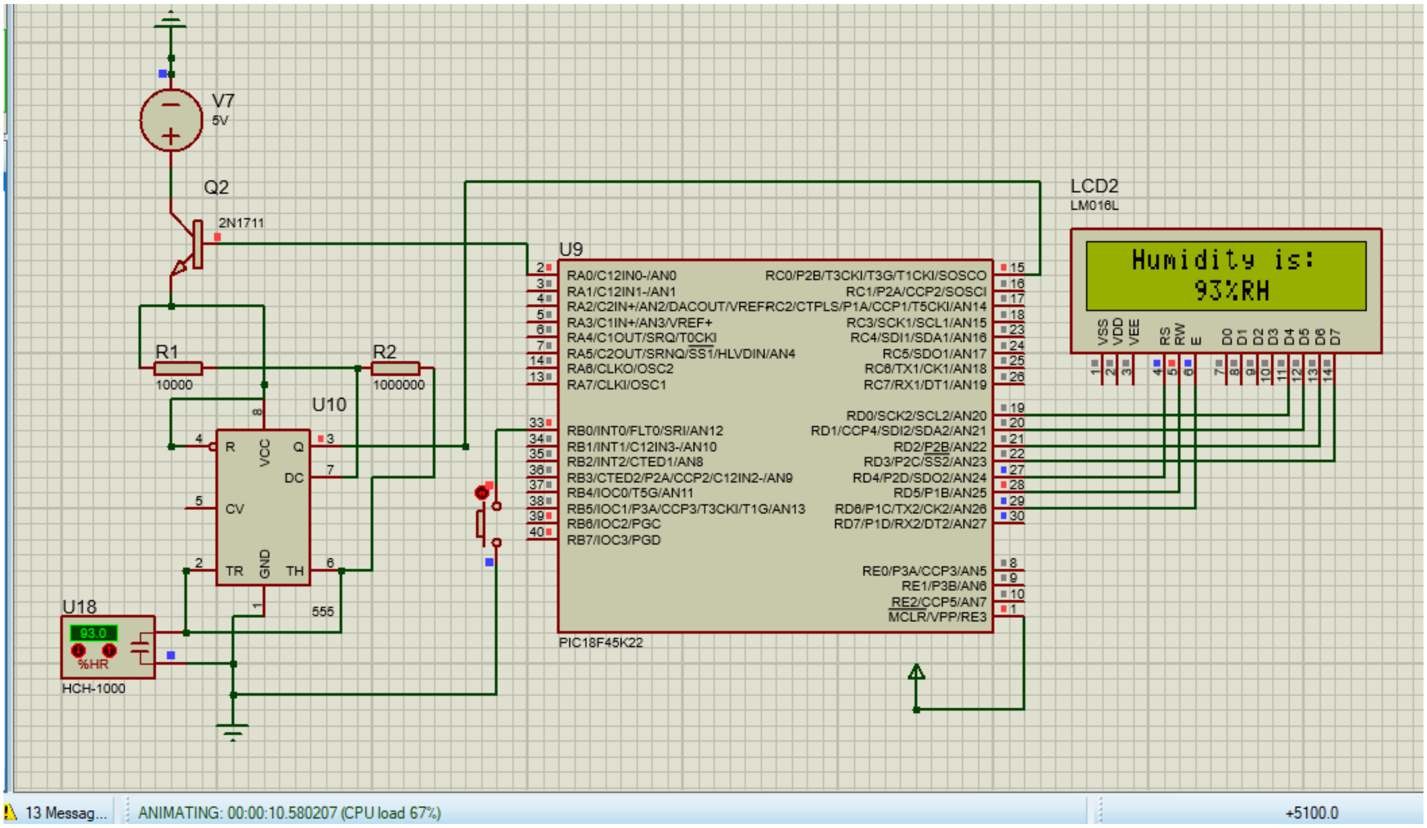
N&F RH meter
Welcome!

13 Messag... ANIMATING: 00:00:07.486064 (CPU load 68%) -2000.0



Humidity is:
67%RH

13 Messag... ANIMATING: 00:00:08.580627 (CPU load 68%) +100.0



```
Humidity is:
   93%RH
```

13 Messag...

ANIMATING: 00:00:10.580207 (CPU load 67%)

+5100.0

Appendix B: C Code

```
#include <p18cxxx.h>
#include <LCD4lib.h>
#include <Delays.h>

#define ON_OFF FLAGS.B0    // bit holding the state of the device (ON or OFF)
#define BJT_base PORTAbits.RA0 // base of the Bipolar Junction transistor
                           // connecting Vcc to  the 555 Timer

int freq_counter;
char Digits[5];
char TMR1H_dummy;   // dummy variable needed to store the correct value of
TMR1H

double  Cm,  RHcc; // measured capacitance 'Cm' and calculated relative humidity
                              //'RHcc'

void Setup(void);
void Init_upon_ON (void);

void main(void){
   Setup();
   Init_upon_ON ();
   while(1);
  }

/* Initializations needed to be done for once and for all*/
void Setup(void){
   /* LCD */
   InitLCD();    // initialize LCD display

   /*Ports*/
   ANSELD = 0x00; TRISD = 0x00;    // PORTD is a digital output port
   ANSELC = 0x00;                  // PORTC is a digital input port

   /*ON_OFF*/
   ANSELBbits.ANSB0 = 0; // RB0 digital input
   INTCON2bits.RBPU = 0; // enable Pull-ups
   INTCON2bits.INTEDG0 = 0; // INT0: react on -ve edge

   TRISAbits.TRISA0 = 0;// configure the pin connected to the base of the BJT
   ANSELAbits.ANSA0 = 0;// as digital output

   BJT_base = 1;       // connect initially  Vcc to 555 Timer

   /*Timer0 : needed to count 1 sec*/
   T0CON = 0b10010101; // divide clock by 64, 16-bit mode
   /*Timer 1: needed count the frequency*/
   T1CON = 0b10000111;     // 16-bit mode, no prescale
   T1GCONbits.TMR1GE = 0;  // needed to enable Timer1

    /*Interrupts & global enables*/
   INTCONbits.TMR0IE  = INTCONbits.GIE = INTCONbits.INT0IE = 1 ;
   }

   /* Initializations needed to be done every time we turn ON the device */
void Init_upon_ON (void){
   ON_OFF = 1; //  ON

   /*Reseting Timer0 and Timer1*/
   TMR0H = (65536 - 15625) / 256; // 15625 * 64 us = 1 sec
   TMR0L = (65536 - 15625) % 256;
   TMR1H = 0;
   TMR1L = 0;

   // Welcoming message
   DispRomStr(Ln1Ch0, (ROM *) "  N&F RH meter  ");
   DispRomStr(Ln2Ch0, (ROM *) "    Welcome!    ");
   Delay10KTCYx(20);
   // Value display message
   DispRomStr(Ln1Ch0, (ROM *) "  Humidity is:  ");
   DispRomStr(Ln2Ch0, (ROM *) "        %RH ");
}
void Timer0_ISR(void);
void ON_OFF_ISR(void);

#pragma code ISR = 0x0008
#pragma interrupt ISR

void ISR (void){
   if (INTCONbits.INT0IF)           // first priority to ON_OFF pushButton
      ON_OFF_ISR();
   else // if(INTCONbits.TMR0IF)    // second priority to Timer0
      Timer0_ISR();
}

void ON_OFF_ISR (void){
   INTCONbits.INT0IF = 0; // acknowledge interrupt
   ON_OFF = ~ON_OFF ;
   BJT_base = ON_OFF ;// if device is ON ==> connect Vcc to 555 Timer
                 // if device is OFF ==> disconnect Vcc from 555 Timer

   if (ON_OFF)    // if ON
      Init_upon_ON();
   else  {        //if OFF
      // Exit message
      DispRomStr(Ln1Ch0, (ROM *) "  Turning Off   ");
      DispRomStr(Ln2Ch0, (ROM *) "   Good Bye!    ");
      Delay10KTCYx(20);
      DispRomStr(Ln1Ch0, (ROM *) "                ");
```

```
        DispRomStr(Ln2Ch0, (ROM *) "                ");
        Sleep();
    }
}

void Timer0_ISR(void){
    INTCONbits.TMR0IF = 0; // acknowledge interrupt

    freq_counter = TMR1L; // doing a dummy read in order to correctly read TMR1H
    TMR1H_dummy = TMR1H;

    /*Calculating RHcc*/
    freq_counter =  TMR1L  + TMR1H_dummy*256 + 2.7;   // + 2.7 is for adjustment
    Cm = 716.417/(freq_counter);                       // freq is Hz, Cm is in nF
    RHcc = Cm*1000/0.6   - 495 + 2.2;                  // + 2.2 is for adjustment

    /*Display RH on the LCD*/
    Bin2AscE(RHcc, Digits);
    DispVarStr(&Digits[3], Ln2Ch6, 1);
    DispVarStr(&Digits[4], Ln2Ch7, 1);

    /*Reseting Timer0 and Timer1*/
    TMR0H = (65536 - 15625) / 256; // 15625 * 64 us = 1 sec
    TMR0L = (65536 - 15625) % 256;
    TMR1H = 0;
    TMR1L = 0;
}
```