Embedded Systems

CC461

FREQUENCY GENERATOR PROJECT

Alaa Hossam Abu-hashima	6750
Habiba Hossam Bakr	6794
Yasmin Ashraf Eshra	6958
Youmna Ahmed Ali	7161
Logine Magdi Hassan	7163

Team (24)

Eng/Mona Eltelaiti

Dr/Hossam Eldin Mostafa

Overview:

The project is to take the number from the keypad and display it into Quad seven segment then, the number we entered must be generated as frequency with minimal loss in frequency and best accuracy when we use frequency meter or hardware frequency meter.

Problems we faced:

We had some problems in proteus simulation due to the overhead in some values so when we calculate the delay before putting it in TH1 and TL1, we subtract the number of cycles which makes the overhead in the result. So, we have tried to compensate the code as much as possible to make the error does not exceed 0.05KHz.

Code:

```
#include <reg51.h>
#include <stdio.h>
#include <stdlib.h>
sbit c3 =P1^7;
sbit c2 =P1^6;
sbit c1 =P1^5;
sbit c0 =P1^4;

sbit r3 =P1^3;
sbit r2 =P1^2;
sbit r1 =P1^1;
sbit r0 =P1^0;

sbit s3 =P0^3;
sbit s2 =P0^2;
sbit s1 =P0^1;
sbit s0 =P0^0;
sbit fout=P3^7;
sbit sw=P3^2;
void seg(unsigned int);
unsigned char digit[] ={0,0,0,0};
unsigned char chr;
unsigned char display [] = {0xC0,0xF9,0x94,0xB0,0xA9,0xA2,0x82,0xF8,0x80,0xA0};
void debounce(i)
{
        int j = 255;
        for(;i>0;i--)
        for(;j>0;j--)
                                ;
}
void debounce1()
{
        unsigned char i = 0;
          while( sw == 0 );
        for( i = 255; i > 0; i--)
        if ( sw == 0 )
                i = 255;
}
unsigned char i = 0;

#define KEYPAD_NO_NEW_DATA '-'
static char Last_valid_Key_G = KEYPAD_NO_NEW_DATA;
char pKey;
char KeyPad_Scan()
{
        static char Old_Key;
        char Key = KEYPAD_NO_NEW_DATA;
        r0=0;
```

```
if(c3 ==0) {Key= 'A';}
        else if(c2==0) {Key= '3'; chr=3;}
        else if(c1==0) {Key= '2'; chr=2;}
        else if(c0==0) {Key= '1'; chr=1;}
        r0=1;
        r1=0;
if(c3 ==0) {Key= 'B';}
        else if(c2==0) {Key= '6'; chr=6;}
        else if(c1==0) {Key= '5'; chr=5;}
        else if(c0==0) {Key= '4'; chr=4;}
        r1=1;
        r2=0;
if(c3 ==0) {Key= 'C';}
        else if(c2==0) {Key= '9'; chr=9;}
        else if(c1==0) {Key= '8'; chr=8;}
        else if(c0==0) {Key= '7'; chr=7;}
        r2=1;
        r3=0;
if(c3 ==0) {Key= 'D';}
        else if(c2==0) {Key= '#';}
        else if(c1==0) {Key= '0'; chr=0;}
        else if(c0==0) {Key= '*';}
        r3=1;
        debounce();

        if(Key == KEYPAD_NO_NEW_DATA)
        {
 Old_Key = KEYPAD_NO_NEW_DATA;
                Last_valid_Key_G = KEYPAD_NO_NEW_DATA;
                return 0;
        }
```

```
            if(Key == Old_Key)
            {
   if(Key !=Last_valid_Key_G)
                        {
                                    pKey=Key;
                                    Last_valid_Key_G = Key;
                                    return 1;
                        }
            }



  Old_Key = Key;
  return 0;
}

bit result;

void timer0_isr (void) interrupt 1
{
            TH0 = 0xEE;    // 9 ms
            TL0 = 0x00;
            result = KeyPad_Scan();
            if(result == 1)
                        {
                                    seg(chr);
                        }

            if (i==0){
                        s0=0;
                        s1=1;
                        s2=1;
                        s3=1;
                        P2=display[digit[0]];
            }

            else if (i==1){
                  s0=1;
                  s1=0;
                  s2=1;
                  s3=1;
                  P2=display[digit[1]];
                  }
            else if (i==2){
            s0=1;
            s1=1;
            s2=0;
            s3=1;
            P2=display[digit[2]];
            }
            else if (i==3)
            {
            s0=1;
            s1=1;
            s2=1;
            s3=0;
            P2=display[digit[3]];
            }
            i++;
            if (i>3) i=0;
      }
      int q=0;
```

```
unsigned int freq ;
float temp;
unsigned int t;
int low,high;
unsigned char l,m;
void freqGenerator(void){
        freq=digit[0]+digit[1]*10+digit[2]*100+digit[3]*1000;
        time=1.0/(freq*2);
        if(time>0.0711065){
                time=time/20.0;
         l=20;
        }
        else{
                l=1;
        }
        t=(time/(0.000001085))-7;
        t=65536-t;
 low=t&0xff;
 high=(t>>8)&0xff;
        m=l;
}

void EXT_INT0 (void) interrupt 0
{
        debounce1();
        EA=0;
        freqGenerator();
        TL1=low;
 TH1=high;
        ET1=1;
        TR1=1;
```

```
            EA=1;
    }
    void timer1_isr (void) interrupt 3
    {
            TL1=low;
      TH1=high;
            if(m==0)
            {
                    m=l;
            }
            if(m==l){
                    fout=~fout;
                    m--;
             }
      else
            {
                    fout=fout;
       m--;
            }
    }

    void main()
    {
            P1= 0xF0;
            r0 = 1;
            r1 = 1;
            r2 = 1;
            r3 = 1;
            P2=0x00;
            P0=0x00;
            TMOD = (TMOD & 0xF0) | 0x11; //0001 0001
```

```
        TH1=high;
        TL1=low;
        fout=1;
        IT0=1;
        ET0=1;
        TH0=0xEE;
        TL0=0x00;
        EX0=1;
        TR0=1;
        EA=1;
        while(1)
        {       }
}
void seg(unsigned char ch)
{
        if (digit[3] == 0){
                digit[3]=digit[2];
        digit[2]=digit[1];
                digit[1]=digit[0];
                digit[0]=ch;
        }
else {
          digit[3]=0;
        digit[2]=0;
                digit[1]=0;
                digit[0]=ch;

}
}
```

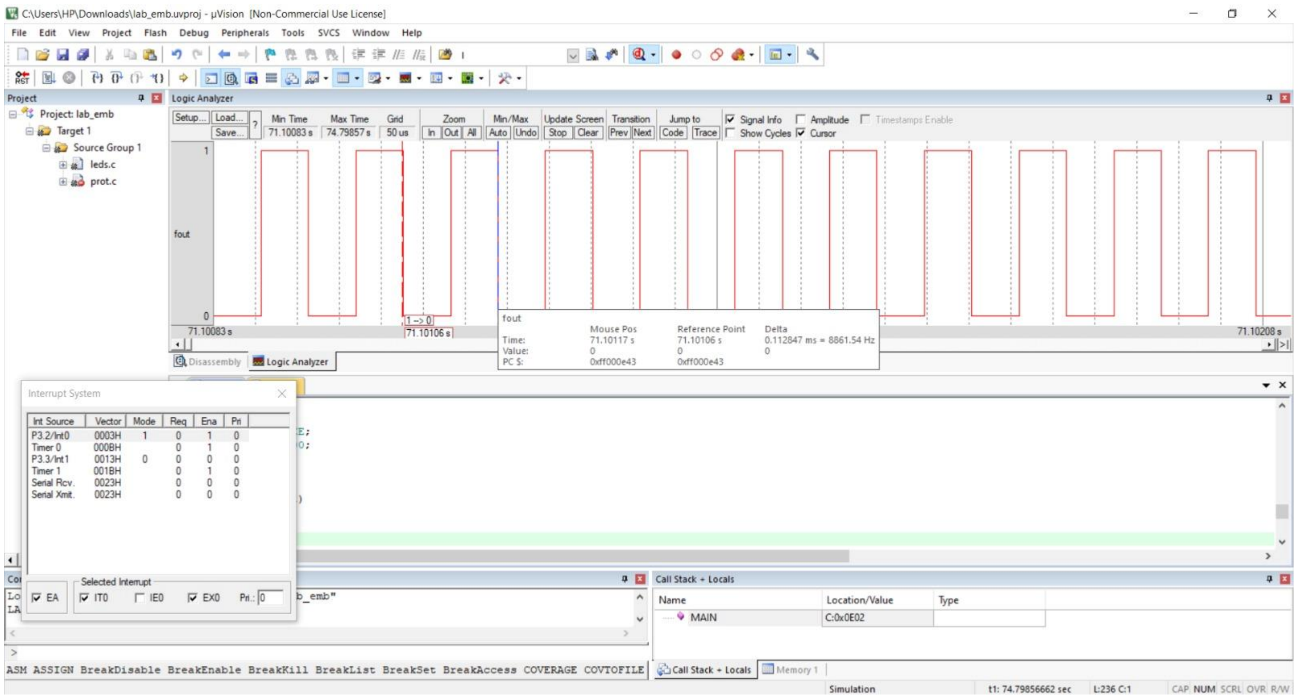
Code observation:

1. We make Port 1 as input for the keypad, port 2 as output in quad seven segment and from P0.0to P0.3 for 2N3906 transistors.
2. We divide our code into functions:
    - ❖ Debounce function for delay purposes.
    - ❖ Keypad scan function: to detect the key pressed by rows and columns positions and convert each key we pressed into its equivalent number.
    - ❖ Segment: to shift the digit into the array of digits.
    - ❖ Timer 0 ISR: to display the number into the seven segment and shifting the digits to the right position and refresh the display every specific delay put into TL0 and TH0.
    - ❖ EXT interrupt 0 ISR: to enable timer 1 for the generation of the square wave on pin P3.7.
    - ❖ Frequency Generator: to take the required frequency and convert it to the equivalent delay then putting it into TH1 and TL1 by doing some calculations as follows:

      ```
      temp=1.0/(freq*2);
      t=(temp/(0.000001085))-7;
      t=65536-t;
      TL1=t&0xff;
       TH1=(t>>8)&0xff;
      ```
    - ❖ Timer 1 ISR: just toggling the wave every amount of time (delay) to generate a real square wave.
    - ❖ Main: to Initialize the flags for timers and interrupts such as EA, IT0, ET1, ER1, TR0, TH0, TL0 and enabling timer 0 and timer 1 both in mode 1. (TMOD 0x11)

Keil Simulation:

- ✓ The figure below shows a Keil simulation when we put frequency = 9000Hz and generating square wave with this frequency.



fout

	Mouse Pos	Reference Point	Delta
Time:	71.10117 s	71.10106 s	0.112847 ms = 8861.54 Hz
Value:	0	0	0
PC $:	0xff000e43	0xff000e43	

Interrupt System

Int Source	Vector	Mode	Req	Ena	Pri
P3.2/Int0	0003H	1	0	1	0
Timer 0	000BH		0	1	0
P3.3/Int1	0013H	0	0	0	0
Timer 1	001BH		0	1	0
Serial Rcv.	0023H		0	0	0
Serial Xmit.	0023H		0	0	0

[✓ EA]

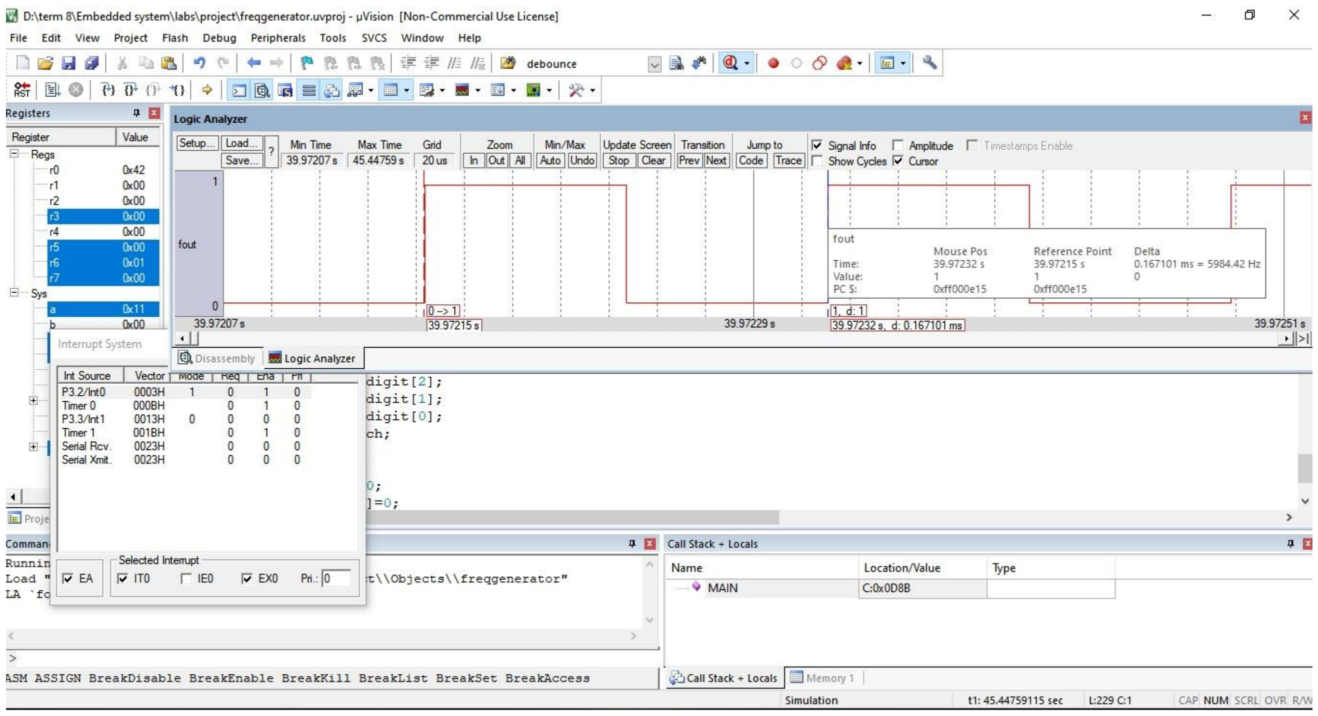
Selected Interrupt: [✓ IT0] IE0 [✓ EX0] Pri.: 0

Name	Location/Value	Type
MAIN	C:0x0E02	

```
ASM ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet BreakAccess COVERAGE COVTOFILE
```

Figure (1) screenshot from keil simulation when frequency = 9000Hz

- ✓ The next shown figure is a simulation for the generated square wave for frequency = 6000Hz.



Register	Value
Regs	
r0	0x42
r1	0x00
r2	0x00
r3	0x00
r4	0x00
r5	0x00
r6	0x01
r7	0x00
Sys	
a	0x11
b	0x00

Min Time	Max Time	Grid
39.97207 s	45.44759 s	20 us

```
39.97207 s    0 -> 1  39.97215 s    39.97229 s    1, d: 1  39.97232 s, d: 0.167101 ms    39.97251 s
```

```
fout
              Mouse Pos       Reference Point    Delta
Time:         39.97232 s      39.97215 s         0.167101 ms = 5984.42 Hz
Value:        1               1                  0
PC $:         0xff000e15      0xff000e15
```

Int Source	Vector	Mode	Req	Ena	Pri
P3.2/Int0	0003H	1	0	1	0
Timer 0	000BH		0	1	0
P3.3/Int1	0013H	0	0	0	0
Timer 1	001BH		0	1	0
Serial Rcv.	0023H		0	0	0
Serial Xmit.	0023H		0	0	0

Selected Interrupt: ☑ EA ☑ IT0 ☐ IE0 ☑ EX0 Pri.: 0

```
digit[2];
digit[1];
digit[0];
ch;

0;
]=0;
```

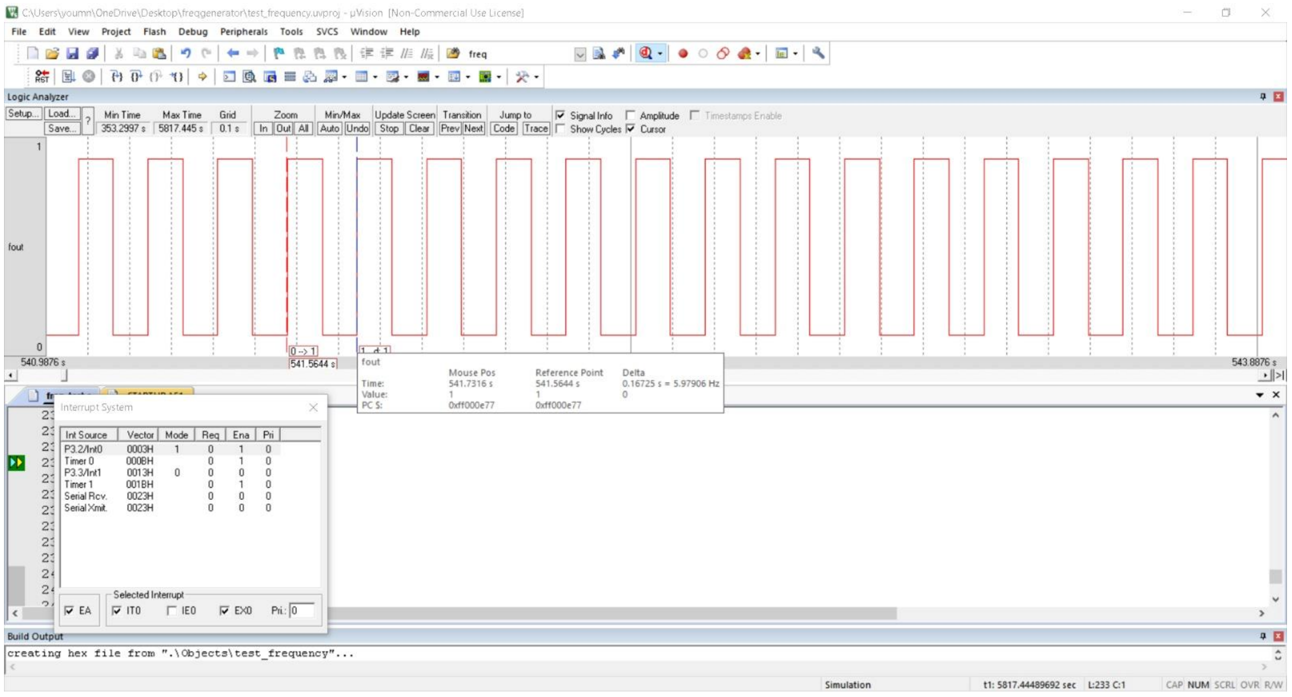
```
Runnin
Load "                    t\\Objects\\freqgenerator"
LA `fo
>
ASM ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet BreakAccess
```

Name	Location/Value	Type
MAIN	C:0x0D8B	

```
Simulation    t1: 45.44759115 sec    L:229 C:1    CAP NUM SCRL OVR R/W
```

Figure (2) screenshot from keil simulation when frequency = 6000Hz

- ✓ We are also handling the cases from 1 to 7 Hz to be generated correctly



	Mouse Pos	Reference Point	Delta
Time:	541.7316 s	541.5644 s	0.16725 s = 5.97906 Hz
Value:	1	1	0
PC $:	0xff000e77	0xff000e77	

Int Source	Vector	Mode	Req	Ena	Pri
P3.2/Int0	0003H	1	0	1	0
Timer 0	000BH		0	1	0
P3.3/Int1	0013H	0	0	0	0
Timer 1	001BH		0	1	0
Serial Rcv.	0023H		0	0	0
Serial Xmit.	0023H		0	0	0

```
creating hex file from ".\Objects\test_frequency"...
```

Figure (2) screenshot from keil simulation when frequency = 6Hz

Proteus simulation:

The figure shown below is a screenshot for simulation in proteus for frequency = 5500Hz.

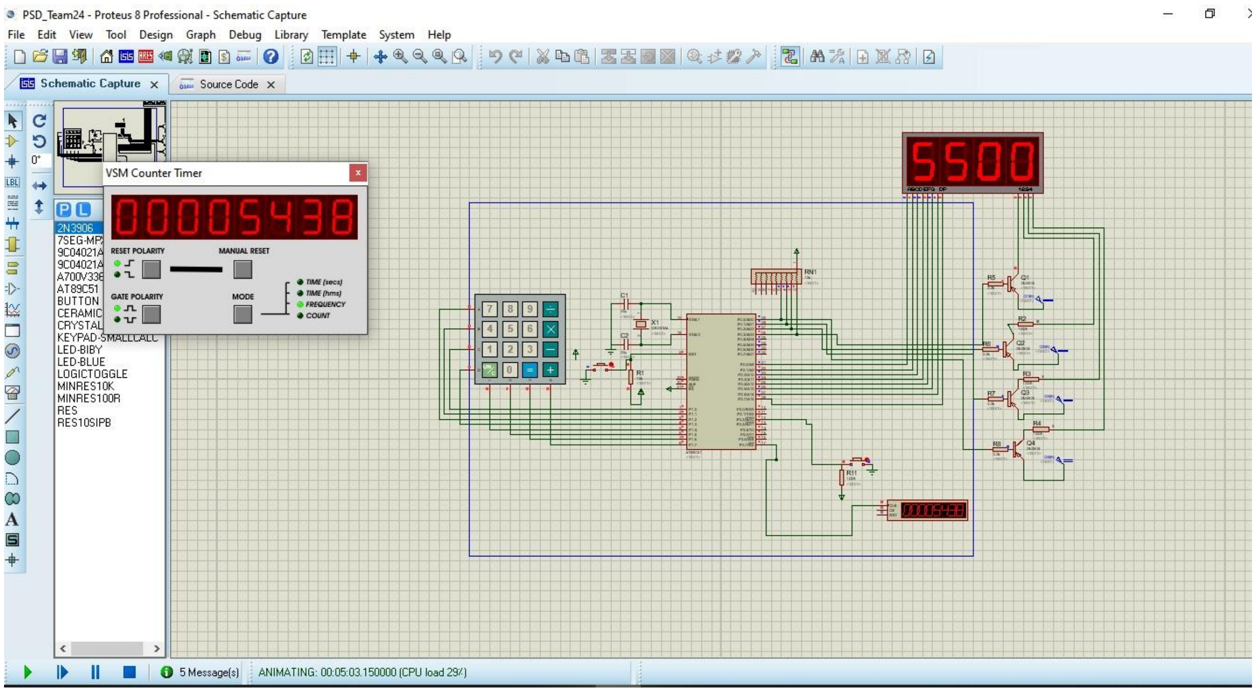


Figure (3) screenshot for frequency = 5500Hz.

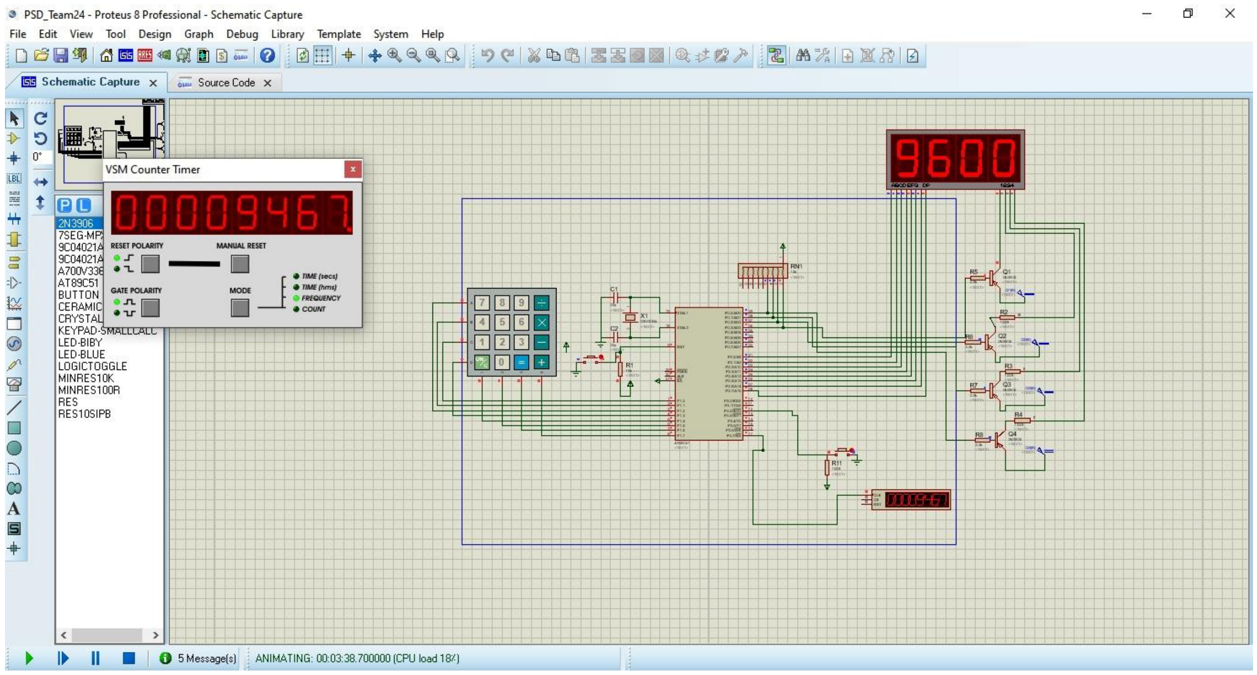
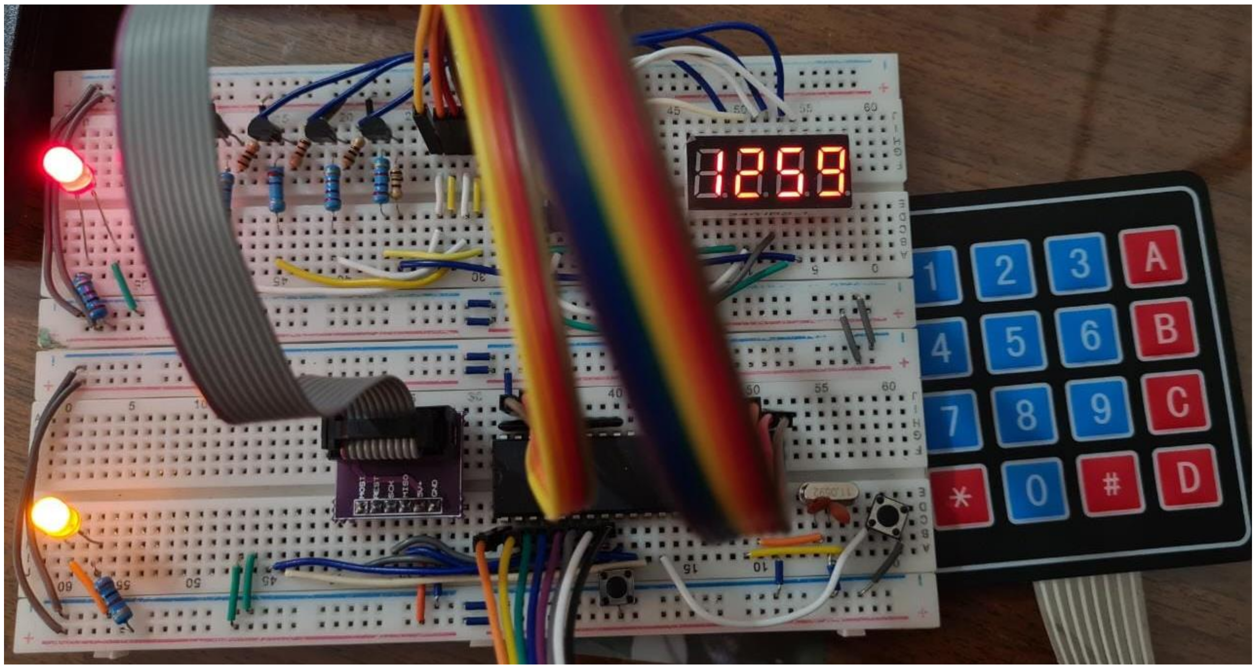
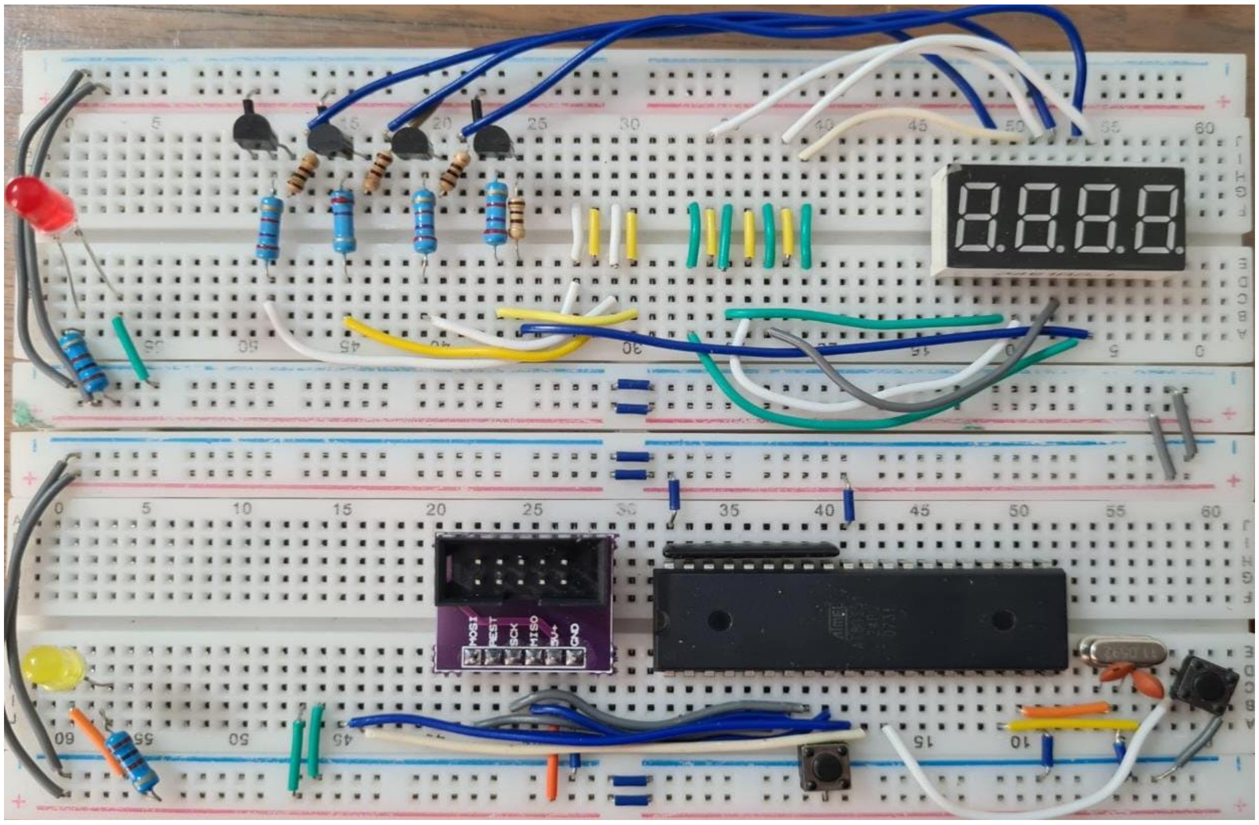


Figure (3) screenshot for frequency = 9600Hz.

Hardware circuit:

The following figure shown our hardware circuit.


1259
1
2
3
A
4
5
6
B
7
8
9
C
*
0
#
D
MOSI
REST
SCK
MISO
5V+
GND


MOSI
REST
SCK
MISO
5V+
GND