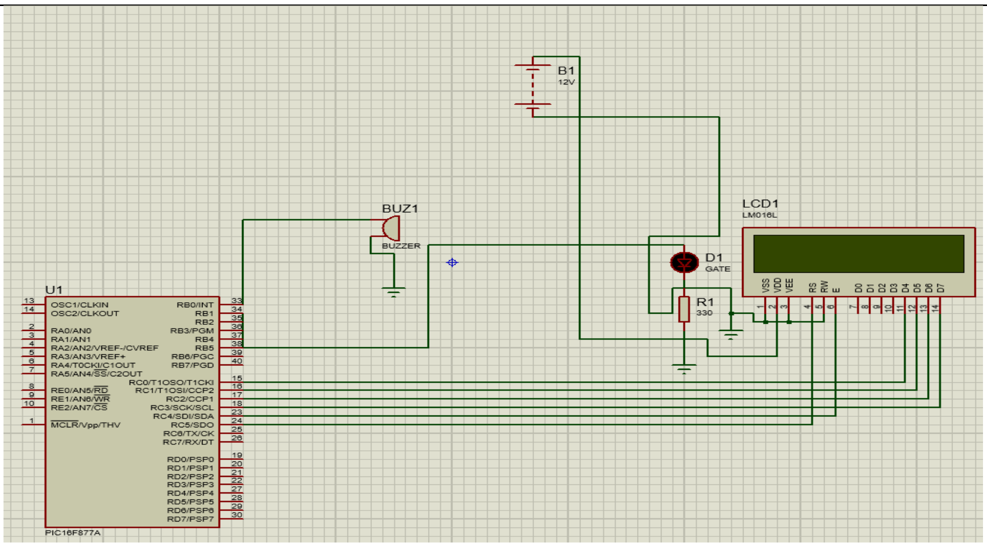
Suez Canal University

automated school bell



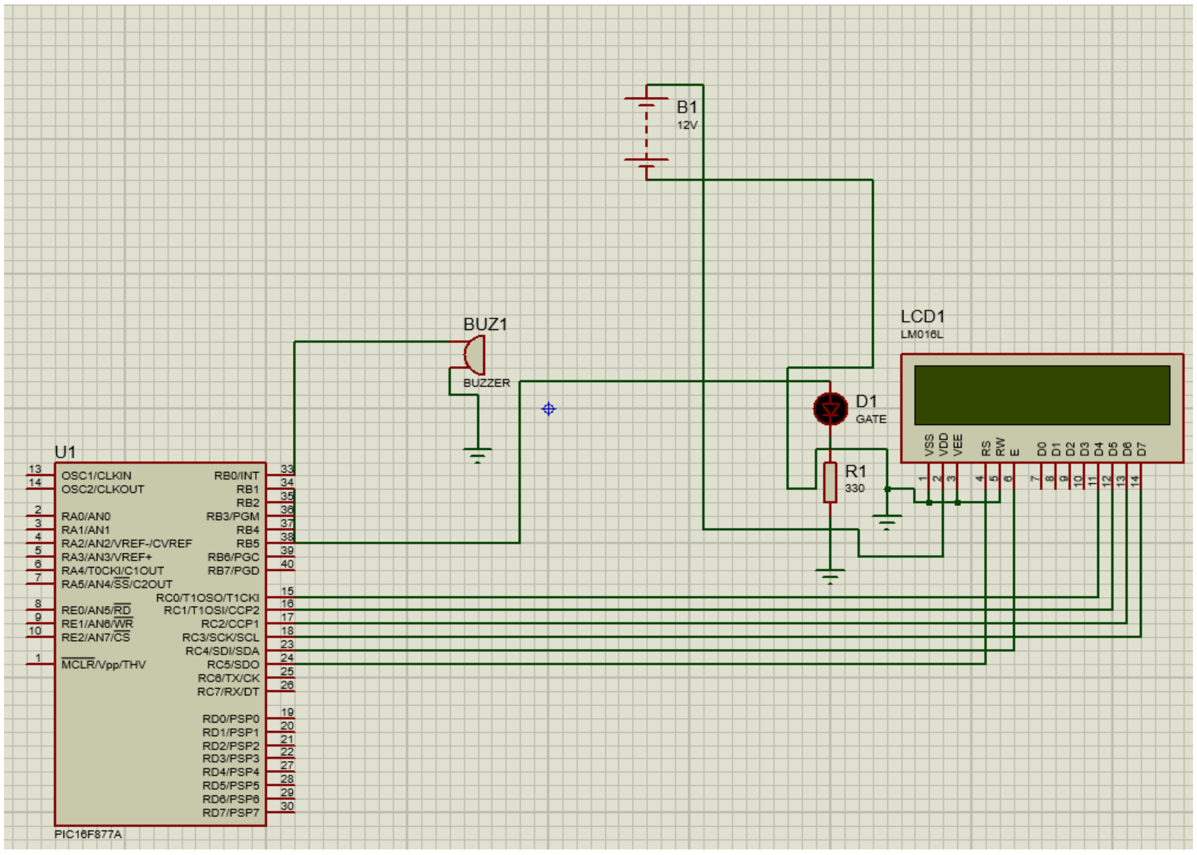
Student name	Ibrahim Mohamed Mohamed El Araby Abdullah Emad Abdel Maksoud Saifeldin ahmos kamal Mohamed Raafat Abdel Aziz Mohamed Mohamed Radwan
Level	4
Subject	Logic report

Ideology: -

In most schools, bells are manual and there is someone keeping track of the time (which waste time and resources and may cause errors), so we decided to build automated control system controlling the school gate and bell.

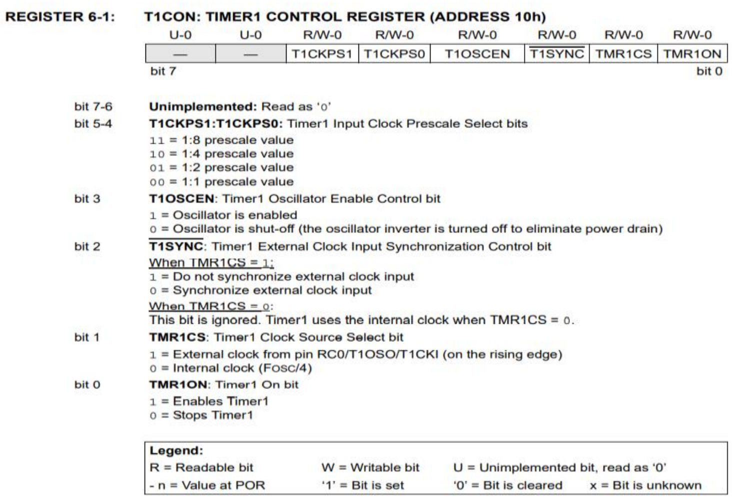
the gate it will be opened at the beginning of the day then the system will keep track of lectures time and after each lecture the bell will ring and increment the lecture counter which we will display on the LCD display, at the end of the day the display will show the message “goodbye student”.

For the main controller we used pic 16f877a (for its small size, low cost, easy to program and have a 16bit timer module which we need) and regular buzzer for the bell and led for the gate and lcd for the display



Code review: -

The pic 16f877a has three counter modules and we used TMR1 module because it is 16bit register



REGISTER 6-1: T1CON: TIMER1 CONTROL REGISTER (ADDRESS 10h)

U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	T1CKPS1	T1CKPS0	T1OSCEN	$\overline{\text{T1SYNC}}$	TMR1CS	TMR1ON
bit 7							bit 0

bit 7-6 **Unimplemented:** Read as '0'

bit 5-4 **T1CKPS1:T1CKPS0:** Timer1 Input Clock Prescale Select bits
11 = 1:8 prescale value
10 = 1:4 prescale value
01 = 1:2 prescale value
00 = 1:1 prescale value

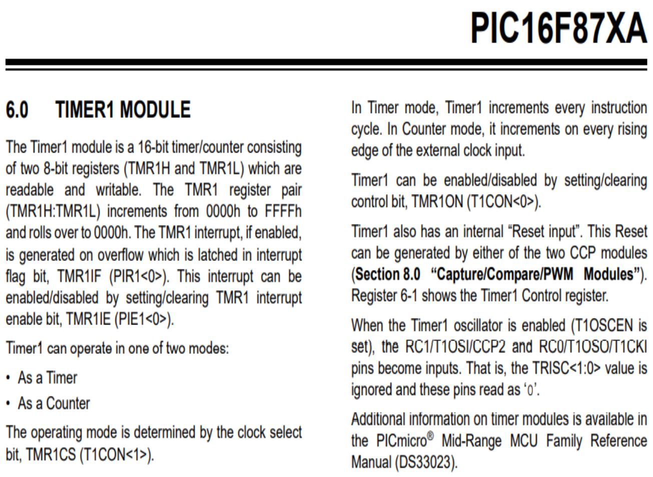
bit 3 **T1OSCEN**: Timer1 Oscillator Enable Control bit
1 = Oscillator is enabled
0 = Oscillator is shut-off (the oscillator inverter is turned off to eliminate power drain)

bit 2 **$\overline{\text{T1SYNC}}$**: Timer1 External Clock Input Synchronization Control bit
When TMR1CS = 1:
1 = Do not synchronize external clock input
0 = Synchronize external clock input
When TMR1CS = 0:
This bit is ignored. Timer1 uses the internal clock when TMR1CS = 0.

bit 1 **TMR1CS**: Timer1 Clock Source Select bit
1 = External clock from pin RC0/T1OSO/T1CKI (on the rising edge)
0 = Internal clock (FOSC/4)

bit 0 **TMR1ON**: Timer1 On bit
1 = Enables Timer1
0 = Stops Timer1

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown



PIC16F87XA

6.0 TIMER1 MODULE

The Timer1 module is a 16-bit timer/counter consisting of two 8-bit registers (TMR1H and TMR1L) which are readable and writable. The TMR1 register pair (TMR1H:TMR1L) increments from 0000h to FFFFh and rolls over to 0000h. The TMR1 interrupt, if enabled, is generated on overflow which is latched in interrupt flag bit, TMR1IF (PIR1<0>). This interrupt can be enabled/disabled by setting/clearing TMR1 interrupt enable bit, TMR1IE (PIE1<0>).

Timer1 can operate in one of two modes:

- As a Timer
- As a Counter

The operating mode is determined by the clock select bit, TMR1CS (T1CON<1>).

In Timer mode, Timer1 increments every instruction cycle. In Counter mode, it increments on every rising edge of the external clock input.

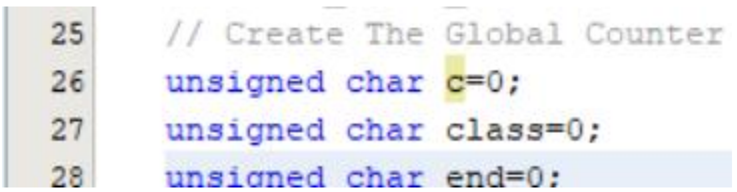
Timer1 can be enabled/disabled by setting/clearing control bit, TMR1ON (T1CON<0>).

Timer1 also has an internal "Reset input". This Reset can be generated by either of the two CCP modules (**Section 8.0 "Capture/Compare/PWM Modules"**). Register 6-1 shows the Timer1 Control register.

When the Timer1 oscillator is enabled (T1OSCEN is set), the RC1/T1OSI/CCP2 and RC0/T1OSO/T1CKI pins become inputs. That is, the TRISC<1:0> value is ignored and these pins read as '0'.

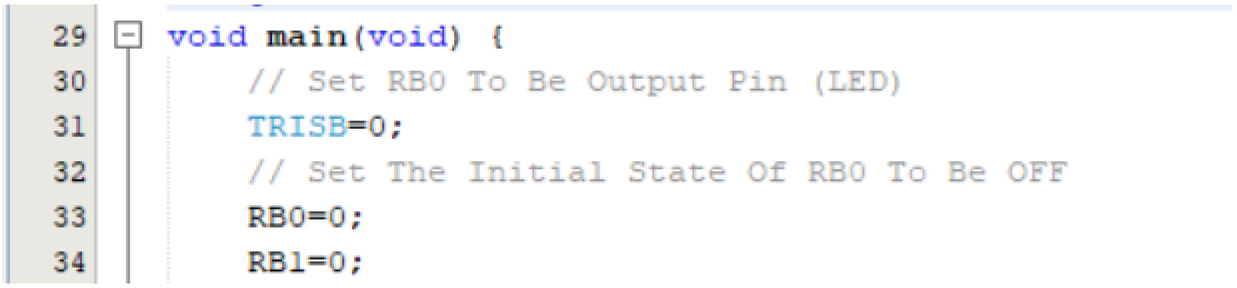
Additional information on timer modules is available in the PICmicro® Mid-Range MCU Family Reference Manual (DS33023).

We set global variable to count time overflow (c), number of lecture (class), end variable (end)



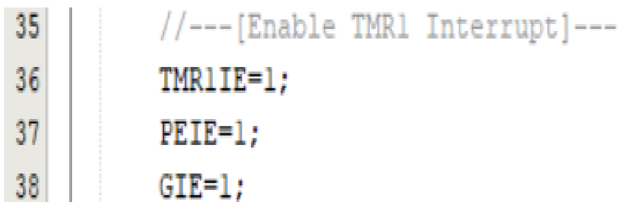
```
// Create The Global Counter
unsigned char c=0;
unsigned char class=0;
unsigned char end=0;
```

We set PORTB in the pic as output (GATE, BELL)

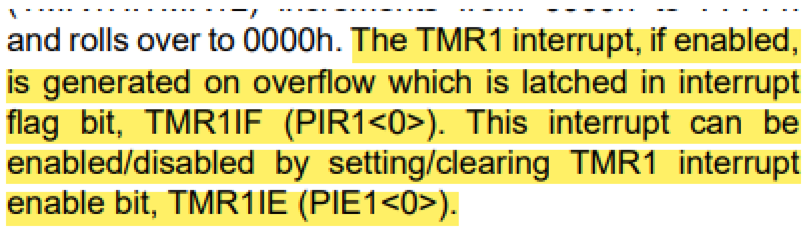


```
void main(void) {
    // Set RB0 To Be Output Pin (LED)
    TRISB=0;
    // Set The Initial State Of RB0 To Be OFF
    RB0=0;
    RB1=0;
```

Then we set the timer interrupt to on



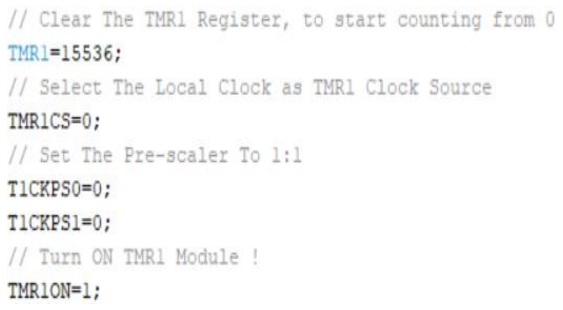
```
    //---[Enable TMR1 Interrupt]---
    TMR1IE=1;
    PEIE=1;
    GIE=1;
```



and rolls over to 0000h. The TMR1 interrupt, if enabled, is generated on overflow which is latched in interrupt flag bit, TMR1IF (PIR1<0>). This interrupt can be enabled/disabled by setting/clearing TMR1 interrupt enable bit, TMR1IE (PIE1<0>).

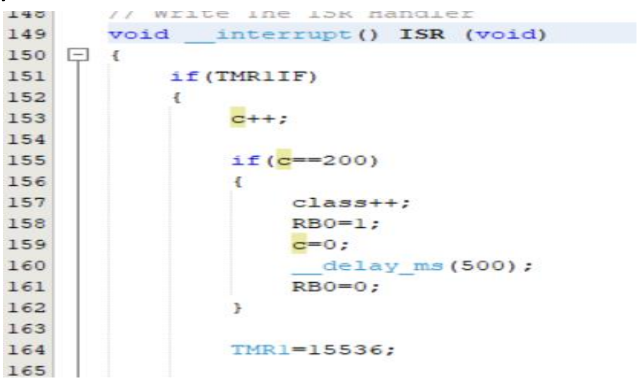
$$T_{out} = \frac{4 * Prescaler * (65536 - TMR1) * X}{F_{osc}}$$

We used this equation to calculate the overflow X
We set prescaler to 1 and the frequency to 4MH and we initialize TMR1 to 15536 to get integer value for the overflow and finally we set T out to 10 seconds (lecture time)



```
// Clear The TMR1 Register, to start counting from 0
TMR1=15536;
// Select The Local Clock as TMR1 Clock Source
TMR1CS=0;
// Set The Pre-scaler To 1:1
T1CKPS0=0;
T1CKPS1=0;
// Turn ON TMR1 Module !
TMR1ON=1;
```

In the interrupt function we will set the RB0(bell) to high every 10 seconds (c = 200)
We reset the TMR1 to 15536

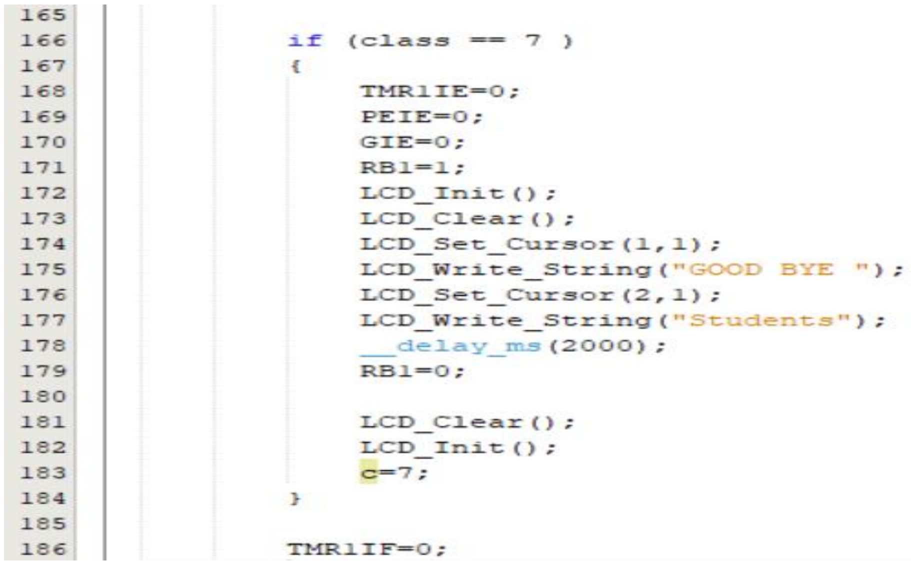


```
// Write The ISR Handler
void __interrupt() ISR (void)
{
        if(TMR1IF)
        {
            c++;

            if(c==200)
            {
                class++;
                RB0=1;
                c=0;
                __delay_ms(500);
                RB0=0;
            }

            TMR1=15536;
```

Then after 7 lectures the RB1(Gate Led) will be high and the lcd will print (“goodbye students”) and will turn off the interrupt flag



```
            if (class == 7 )
            {
                TMR1IE=0;
                PEIE=0;
                GIE=0;
                RB1=1;
                LCD_Init();
                LCD_Clear();
                LCD_Set_Cursor(1,1);
                LCD_Write_String("GOOD BYE ");
                LCD_Set_Cursor(2,1);
                LCD_Write_String("Students");
                __delay_ms(2000);
                RB1=0;

                LCD_Clear();
                LCD_Init();
                c=7;
            }

            TMR1IF=0;
```

*The lcd library has been imported to write to the lcd display