Lab 3: Theremin

ESE519/IPD519: Introduction to Embedded Systems
University of Pennsylvania

Please fill out your name and link your Github repository below to begin. Be sure that your code on the repo is up-to-date before submission!

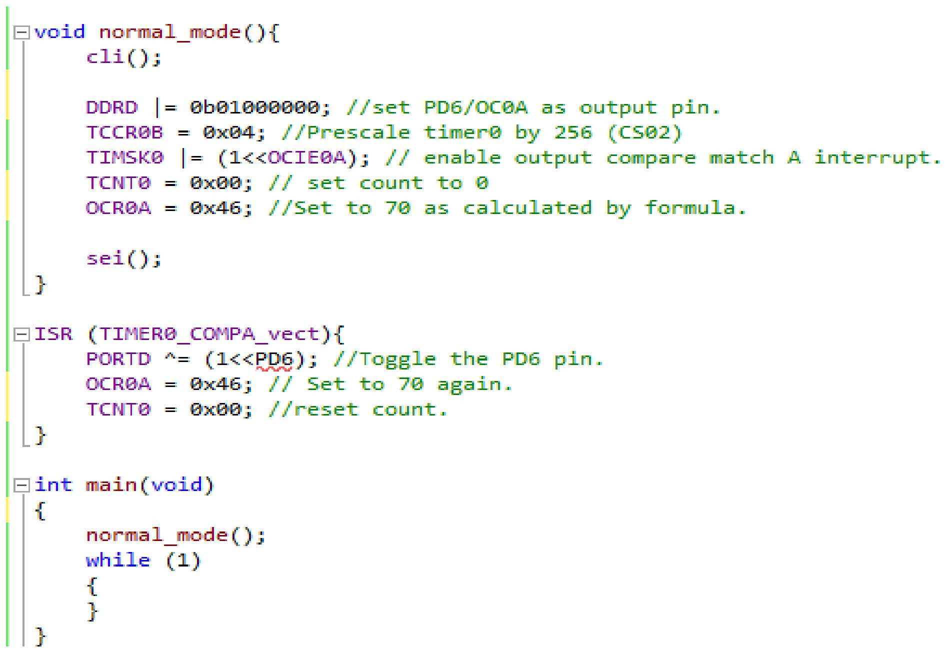
Student Name: Sumanth Marakini Shivshankar
Pennkey: sumanthm
GitHub Repository: https://github.com/SumanthMarakini/ESE519.git

1. The frequency is calculated using the formula

   $f_{OCnx} = \frac{f_{clkI/O}}{2*N*256}$ Where, N is the prescaler. N=8 in this case.

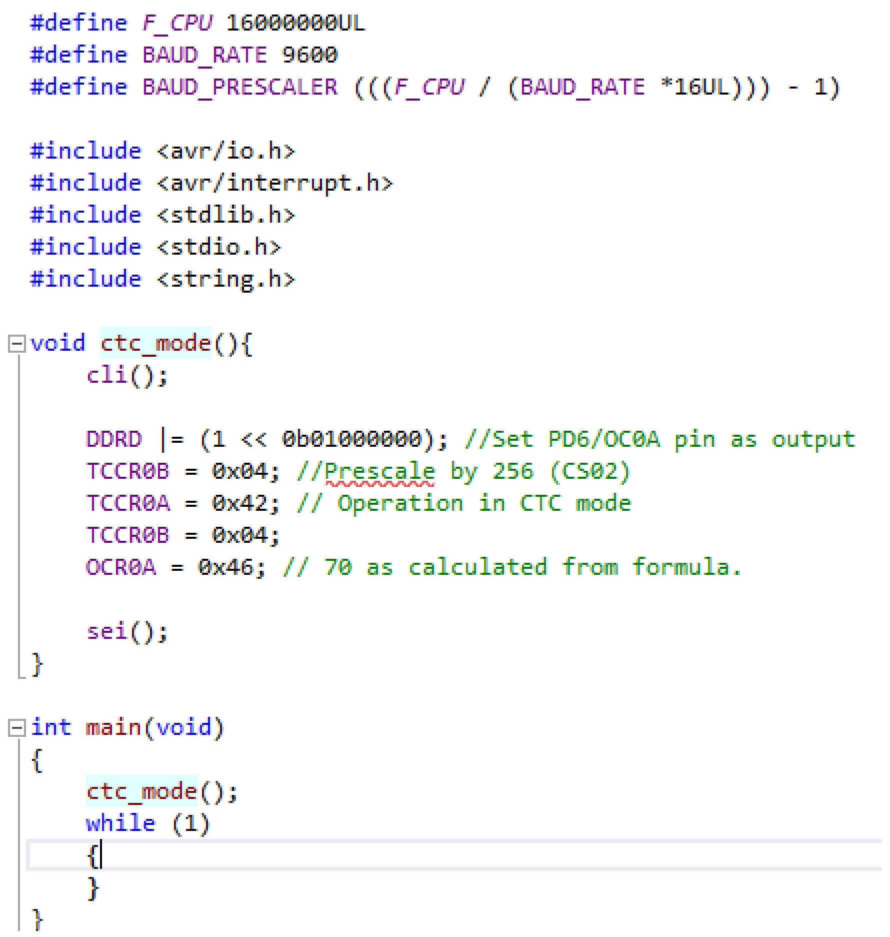
   Therefore, f = 16000000/2*8*256 = 3906.25 Hz. which is the expected frequency.

2. Yes, the clock has to be prescaled by 256 in order to obtain the desired frequency of 440Hz.
3. Using the formula $f_{OCnx} = \frac{f_{clkI/O}}{2*N*(1+OCnx)}$, where FOCnx is 440 Hz, we can obtain the value of OCnx to be used. The OCnx value used after prescaling the clock by 256 is 70 or 0x46.
4. The screenshot of the code is shown below:



```
void normal_mode(){
    cli();

    DDRD |= 0b01000000; //set PD6/OC0A as output pin.
    TCCR0B = 0x04; //Prescale timer0 by 256 (CS02)
    TIMSK0 |= (1<<OCIE0A); // enable output compare match A interrupt.
    TCNT0 = 0x00; // set count to 0
    OCR0A = 0x46; //Set to 70 as calculated by formula.

    sei();
}

ISR (TIMER0_COMPA_vect){
    PORTD ^= (1<<PD6); //Toggle the PD6 pin.
    OCR0A = 0x46; // Set to 70 again.
    TCNT0 = 0x00; //reset count.
}

int main(void)
{
    normal_mode();
    while (1)
    {
    }
}
```

5. The OCR0A should be set to 70 or 0x46. The formula mentioned above holds good for this as well.
6. The screenshot of the code is as shown below:

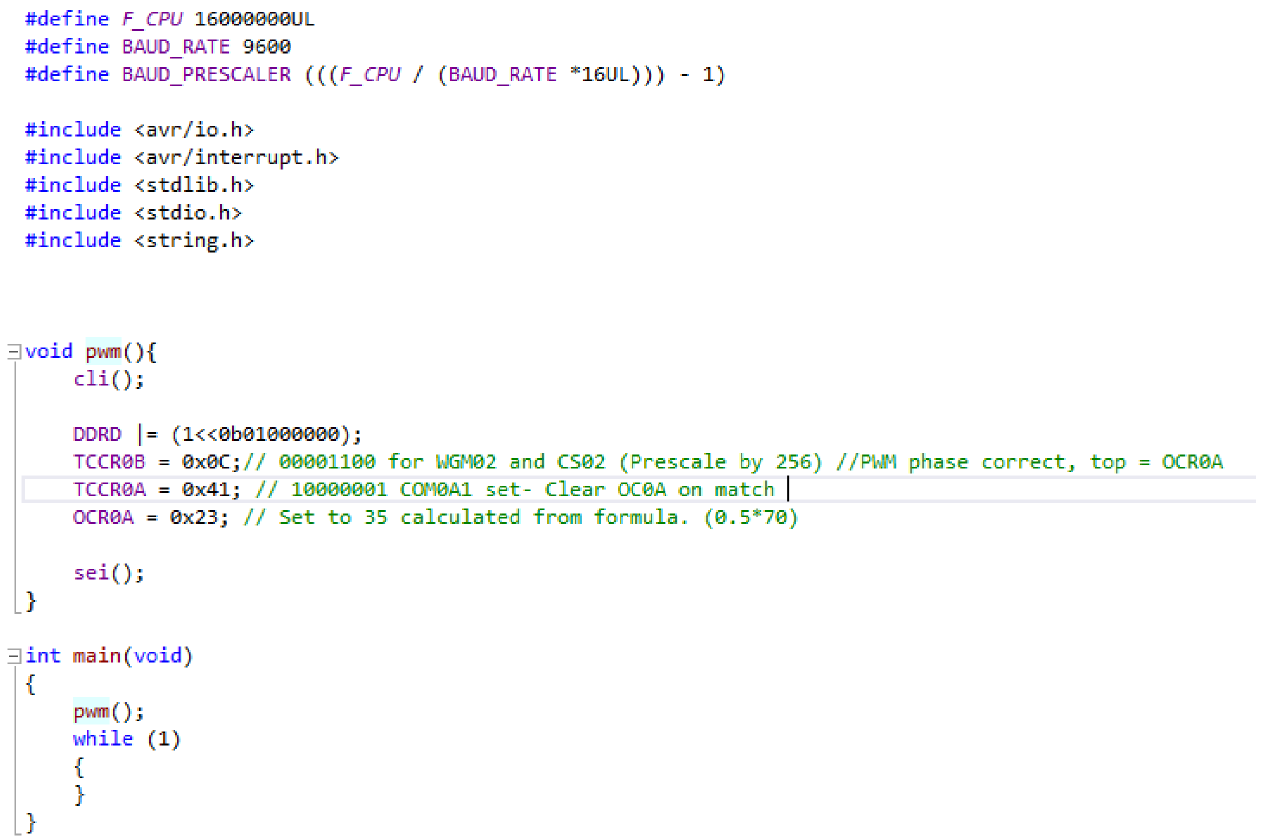


```
#define F_CPU 16000000UL
#define BAUD_RATE 9600
#define BAUD_PRESCALER (((F_CPU / (BAUD_RATE *16UL))) - 1)

#include <avr/io.h>
#include <avr/interrupt.h>
#include <stdlib.h>
#include <stdio.h>
#include <string.h>

void ctc_mode(){
    cli();

    DDRD |= (1 << 0b01000000); //Set PD6/OC0A pin as output
    TCCR0B = 0x04; //Prescale by 256 (CS02)
    TCCR0A = 0x42; // Operation in CTC mode
    TCCR0B = 0x04;
    OCR0A = 0x46; // 70 as calculated from formula.

    sei();
}

int main(void)
{
    ctc_mode();
    while (1)
    {
    }
}
```

7. OCR0A value must be 35 or 0x23 as calculated from the formula mentioned below.

PWM phase correct, top value = OCR0A. $f_{OCnx} = \frac{f_{clkI/O}}{N*(1+OCnx)}$.

8.



```
#define F_CPU 16000000UL
#define BAUD_RATE 9600
#define BAUD_PRESCALER (((F_CPU / (BAUD_RATE *16UL))) - 1)

#include <avr/io.h>
#include <avr/interrupt.h>
#include <stdlib.h>
#include <stdio.h>
#include <string.h>


void pwm(){
    cli();

    DDRD |= (1<<0b01000000);
    TCCR0B = 0x0C;// 00001100 for WGM02 and CS02 (Prescale by 256) //PWM phase correct, top = OCR0A
    TCCR0A = 0x41; // 10000001 COM0A1 set- Clear OC0A on match
    OCR0A = 0x23; // Set to 35 calculated from formula. (0.5*70)

    sei();
}

int main(void)
{
    pwm();
    while (1)
    {
    }
}
```

9. The duration of the input pulse is 10us.

10. The Trig pin is used to trigger the ultrasonic sound. It is an Input pin and has to be kept high for 10us to start the measurement by sending an ultrasonic wave.

The Echo pin is used to generate a pulse when the reflected signal is received. It is an Output pin and reads high for a period of time which will be equal to the time taken for the ultrasonic wave to return back to the sensor.

11. Largest distance measured is 257 cm.

12. Smallest distance measured is 6 cm.

13. Prescale the timer by 64. f_clkIO = 16MHz.

Note	**C6**	**D6**	**E6**	**F6**	**G6**	**A6**	**B6**	**C7**
Freq (Hz)	1046	1174	1318	1397	1568	1760	1975	2093
OCR0A	238	213	189	180	159	141	128	118

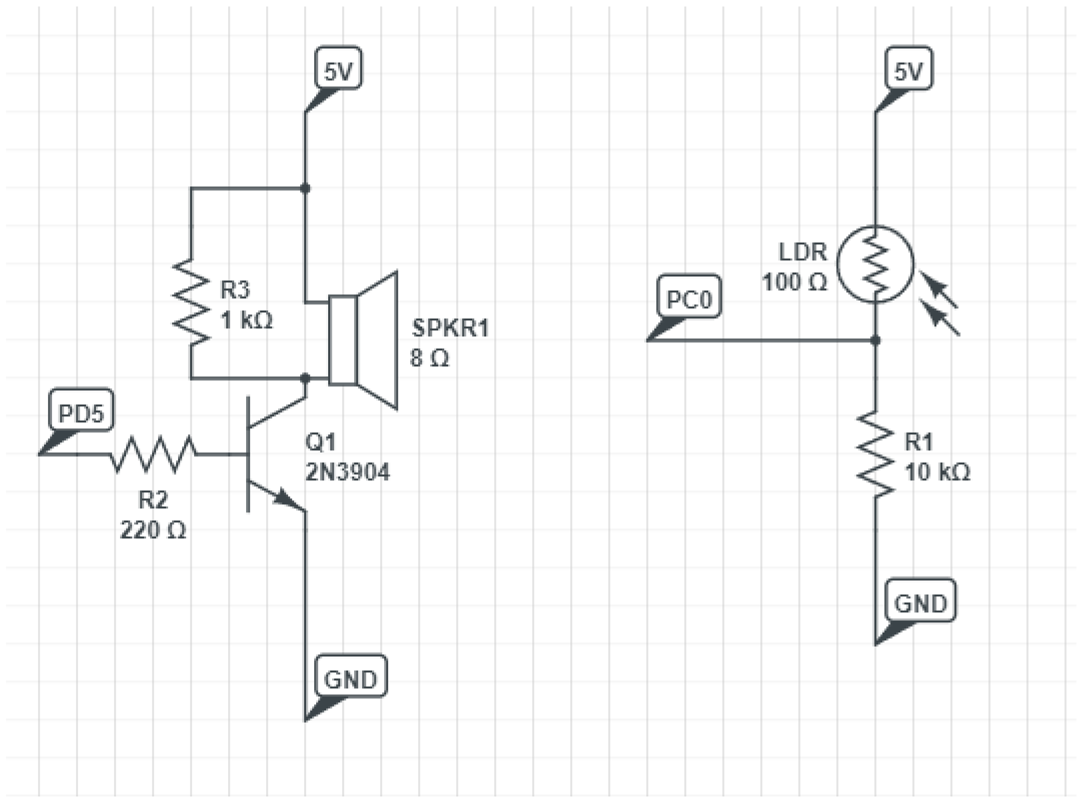
14. FREQ (or OCR0A) = SENSOR_VALUE * SOME_RATIO + SOME_NUMBER
OCR0A = (distance * 1.51) + 102.3

15. Minimum ADC value = 24
Maximum ADC value = 1006

16. Following is the range of ADC values:

ADC Ranges	Duty Cycle
100-190	5%
190-280	10%
280-370	15%
370-460	20%
460-550	25%
550-640	30%
640-730	35%
730-820	40%
820-910	45%
910-1000	50%

17. The Circuit is attached below:
Connect the PB4 to TRIG and PB0 to ECHO.



18. Video attached here :
https://drive.google.com/drive/folders/1MW0N4kdWnV5YDEGTDlEALE_ioQynXvE8?usp=sharing
Final Code is in the github link as mentioned above. Please refer to **Putting_together_lab3.c**

Extra Credit:

19. R1 is the base resistor. It is a current limiting resistor that limits the current flowing into the base of the transistor to prevent it from being damaged. However, it must also allow sufficient base current to flow to ensure that the transistor is fully saturated when switched on. So the R1 resistor also protects the output from the arduino.

20. BJTs are commonly used in low-current applications. The BJT allows the buzzer to be powered from a different voltage to the Arduino. BJTs have higher fidelity & better gain in the linear areas as evaluated with the MOSFETs which is why BJT is used instead of a MOSFET. Additionally, since BJTs operate better in high frequency operating devices (eg: buzzer), it is favourable to use a BJT. BJTs are faster because of the low capacitance on the control pin. A MOSFET can also be used here, but is not preferred because of the above mentioned drawbacks.