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:Ziyi Yang
Pennkey:ziyiyang
GitHub Repository: https://github.com/Zakiyanggg/Intro-to-Embedded-system/tree/main/Lab3

1. Main clock frequency = 16MHz
   Timer0：8bit timer，count from 0~255，256 steps
   Prescaler：1/8
   Ovf trigger：1/2
   16000000/256/8/2 = 3906.25Hz
2. Normal mode
   2. Yes, timer0 is prescaled by 256
   3. 1600000/440/2/256 = 71 since it counts from 0 so OCR0A should be 70
   4.

```
 * GccApplication2.c
 *
 * Created: 10/6/2021 2:51:53 PM
 * Author : Zaki
 */
#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 <util/delay.h>
#include <stdio.h>
#include <string.h>
#include "inc/uart.h"


ISR(TIMER0_COMPA_vect)//OVF register trigger
{
    PORTD ^= (1<<DDD6);
    TCNT0 = 0;
}

int main(void)
{
    cli();
    //PORTD &= ~(1<<PORTD6);
    DDRD &= ~(1<<DDD6);//PD6 setup as output pin
    OCR0A = 70;
    TCCR0B |= (1<<CS02);//prescaler 256
    TIMSK0 |= (1<<OCIE0A);//Enable Timer Overflow Interrupt
    sei();
    while (1)
    {
    }
}
```

3. CTC mode
5. OCR0A = 70
6.

```c
/*
 * GccApplication2.c
 *
 * Created: 10/6/2021 2:51:53 PM
 * Author : Zaki
 */
#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 <util/delay.h>
#include <stdio.h>
#include <string.h>
#include "inc/uart.h"



int main(void)
{
    cli();
    OCR0A = 70;
    PORTD &= ~(1<<PORTD6);
    DDRD |=(1<<DDD6);//PD6 setup as output pin
    TCCR0B |= (1<<CS02);//prescaler 256
    TCCR0A |= (1<<WGM01);//set mode up to OCR0A
    TCCR0A |= (1<<COM0A0);//PD6/OC0A toggle on compare match
    sei();
    while (1)
    {
    }
}
```

4. PWM mode

7. Here the PD6 is toggled when TCNT0 is up to OCR0A and down to 0, So I divided the OCR0A by 2 compare to the previous value to keep the output frequency to 440Hz.

8.

```c
/*
 * GccApplication2.c
 *
 * Created: 10/6/2021 2:51:53 PM
 * Author : Zaki
 */
#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 <util/delay.h>
#include <stdio.h>
#include <string.h>
#include "inc/uart.h"


int main(void)
{
    cli();
    DDRD |=(1<<DDD6);
    PORTD &=~(1<<PORTD6);

    TCCR0B |= (1<<CS02);//prescaler 256

    TCCR0A |=(1<<WGM00);
    TCCR0A &=~(1<<WGM01);
    TCCR0B |= (1<<WGM02);//PWM correct mode

    TCCR0A |= (1<<COM0A0);//PD6/OC0A toggle on compare match

    OCR0A = 35;

    sei();
    while (1)
    {
    }
}
```

Part C:

9. More than 10us. I used 32us.

10. Trig is used to receive pulse from the microcontroller to start ranging
Echo is used to output the pulse that its pulse width is proportional to the range.

11. I tested it at my bedroom, it was around 350-360cm, it's hard to aim the reflection object towards the small sensor. I'm sure it can go higher than that.

12. 1cm is not stable and 2cm is ensured.

PartD:

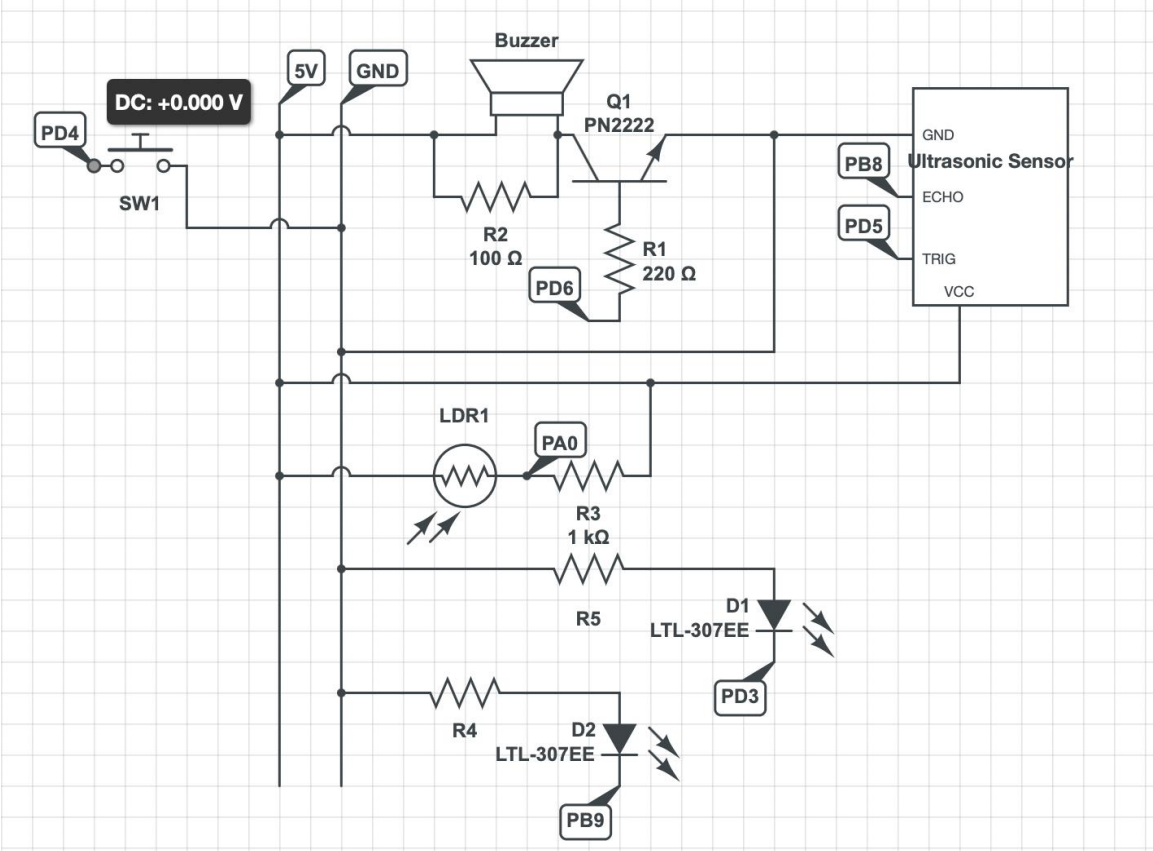
13.

Note	**C6**	**D6**	**E6**	**F6**	**G6**	**A6**	**B6**	**C7**
Freq (Hz)	1046	1174	1318	1397	1568	1760	1975	2093
OCR0A	29	26	23	22	20	18	16	15

1. $\hat{y} = -71.88292X + 3068.88705$
1. Min:10 when covered by a black face mask Max:1017 when direct exposure under flash light
1.

ADC Ranges	Duty Cycle
10-110	5%
110-210	10%
210-310	15%
310-410	20%
410-510	25%
510-610	30%
610-710	35%
710-810	40%
810-910	45%
910-1010	50%

17.



18. Uploaded on git

Part G:

19. The resistor is used to ensure that not too much current can flow through the base and cause damage to the transistor, at the same time allows enough current to make sure the transistor is saturated when turned on.

20. Since using current control is easier in our case, BJT is a better option. Yes, we can replace it with a MOSFET.

21. Two LEDs to indicate which frequency output mode is being used. Red is for continuous mode and green is for discrete mode.

22.