Name: Von Kaukeano EE3612 Assignment name: Final Project Date:12/18/19

Place the problem statement here.

a) If your hand is greater than 20 inches from the ping sensor, turn the servo motor in **clockwise** direction and the LED is off.

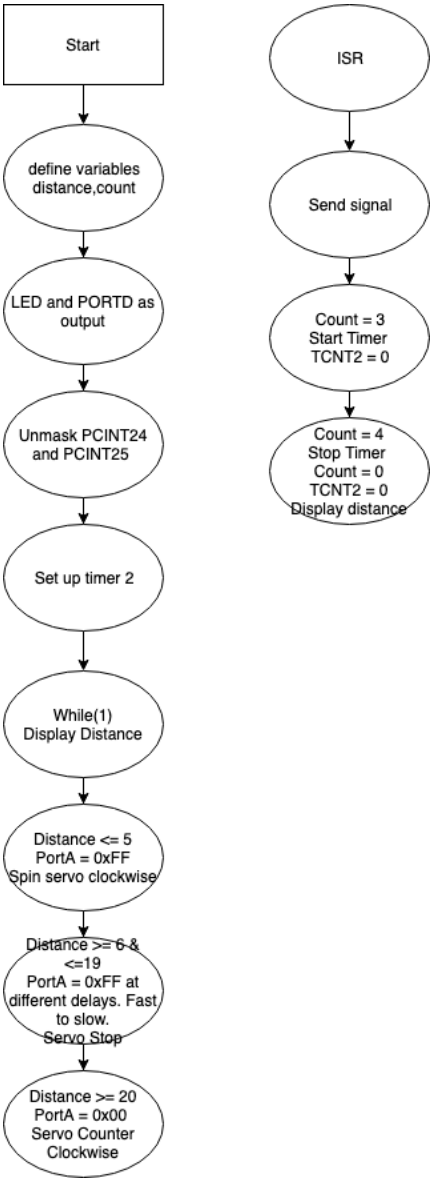
b) If your hand is at a distance between 5 and 20 inches, blink an LED (or all of them). The blinking speed of the LED is **determined by the distance** and should blink slowly starting at 20 inches and incrementally "speed up" as you reach 5 inches. The servo and stepper motors should **not turn on** in this distance range.

c) If your hand is 5 inches or closer towards the ping sensor, the LED should **light up solid** and both the servo motor should rotate **counter clockwise** with no delay and you should display in inches on the LCD the distance. The LED(s) should return back to a blink if it is between 5 inches and 20 inches and should turn off for distances greater than 20 inches.
Make sure that all the components are attached on a breadboard for ease of design. When performing the task, try to elevate the PING sensor by **placing something** under the breadboard. This will help the sensor detect objects in its path more efficiently (your hand in this case) and minimize false errors.

Describe your problem solution here using a few sentences to describe your flowchart below.

The solution to this problem starts with the code from the slides to determine distance from the ultrasonic sensor and display it on the LCD. Once it was determined that the ping sensor was broken and borrowed a fellow students, the code worked. After using the code I added a few if statements to section out the distances for the ranges specified in the problem. Using the delay function I successfully turned the servo motor and sped and slowed down the LED's.

Place Flowchart from IO.DRAW here. You may have to use a screenshot.



Code for your solution here.

```
#include <avr/io.h>
#include <avr/interrupt.h>
#define F_CPU 16000000UL
#include <util/delay.h>
#include "SSD1306.h"

#define PINGPin PD0
#define ECHOPin PD1
```

```
volatile int dist = 0;
char str[100];
volatile int count = 0;
char text[] = "TEMPLE";
int main(){

	OLED_Init();
	_delay_ms(1);
	OLED_Clear();

	DDRA = 0xFF;
	DDRD=0xFF;

	PORTD|= (1<<0);
	PORTD|= (1<<1);

	PCICR |= (1<<PCIE3);
	PCIFR |= (1<<PCIF3);
	PCMSK3 |= (1<<PCINT24)|(1<<PCINT25);
	TCCR2B =(1<<CS21);

	TIMSK2 = (1<<TOIE2);
	OCR2B = 255;
	sei();

	while (1)
	{
		OLED_SetCursor(0,0);
		OLED_SetCursor(1,0);
		OLED_DisplayString(text);
		OLED_SetCursor(4,0);
		OLED_Printf("Distance:");
		OLED_DisplayFloatNumber(dist);

		if(dist <= 5){
			PORTA=0xFF;
			PORTD = 0b00000100;
			_delay_ms(1.3);
			PORTD = 0b00000000;
			_delay_ms(20);
		}

		else if(dist >= 6 & dist <= 19){

			if(dist >= 6){
				PORTA = 0xFF;
```

```
                        _delay_ms(500);
                        PORTA = 0x00;
                        _delay_ms(200);
                }

                else if(dist >= 10){
                        PORTA = 0xFF;
                        _delay_ms(500);
                        PORTA = 0x00;
                        _delay_ms(100);
                }

                else if(dist >= 15){
                        PORTA = 0xFF;
                        _delay_ms(500);
                        PORTA = 0x00;
                        _delay_ms(50);
                }

                else if(dist >= 17){
                        PORTA = 0xFF;
                        _delay_ms(500);
                        PORTA = 0x00;
                        _delay_ms(20);
                }


        }

        else if(dist >= 20){
                PORTA = 0x00;
                PORTD = 0b00000100;
                _delay_ms(1.7);
                PORTD = 0b00000000;
                _delay_ms(20);
        }

    }

}

ISR(TIMER2_OVF_vect){
    if(OCR2B == 255){
        TCCR2B = (1<<CS21);
        DDRD |= (1<<PINGPin);
        PORTD |= (1<<PINGPin);
```

```
                OCR2B = 3;
                return;
        }
        else if(OCR2B == 3){
                PORTD &= ~(1<<PINGPin);
                OCR2B = 2;
                return;
        }

        else if(OCR2B == 2){
                DDRD &= ~(1<<ECHOPin);
                OCR2B = 255;
                TCCR2B = (1<<CS22)|(1<<CS21);
                return;
        }

}

ISR(PCINT3_vect){
        count++;
        if(count == 3)
        TCNT2 = 0;

        if(count == 4){
                dist=(TCNT2*.214);
                OCR2B = 255;
                count = 0;
                return;
        }

        return;
}
```

Discuss results and verification here.

I have used the ultrasonic sensor on a raspberry pi and it is interesting breaking down the timer and interrupt that this project consists of. The raspberry pi was configured similarly but used functions such as time to calculate time the number of pulses received. The delays could have been replaced with timers and interrupts, but it took longer than expected to come to the conclusion that my ping sensor actually did not work. The project still meets all specs of the assignment.

Screenshot of results here that are described above

https://drive.google.com/file/d/1v-O0SAWwsdSocBuOJBwvdP7QPirBs3vJ/view?usp=sharing