CPE301 – SPRING 2022

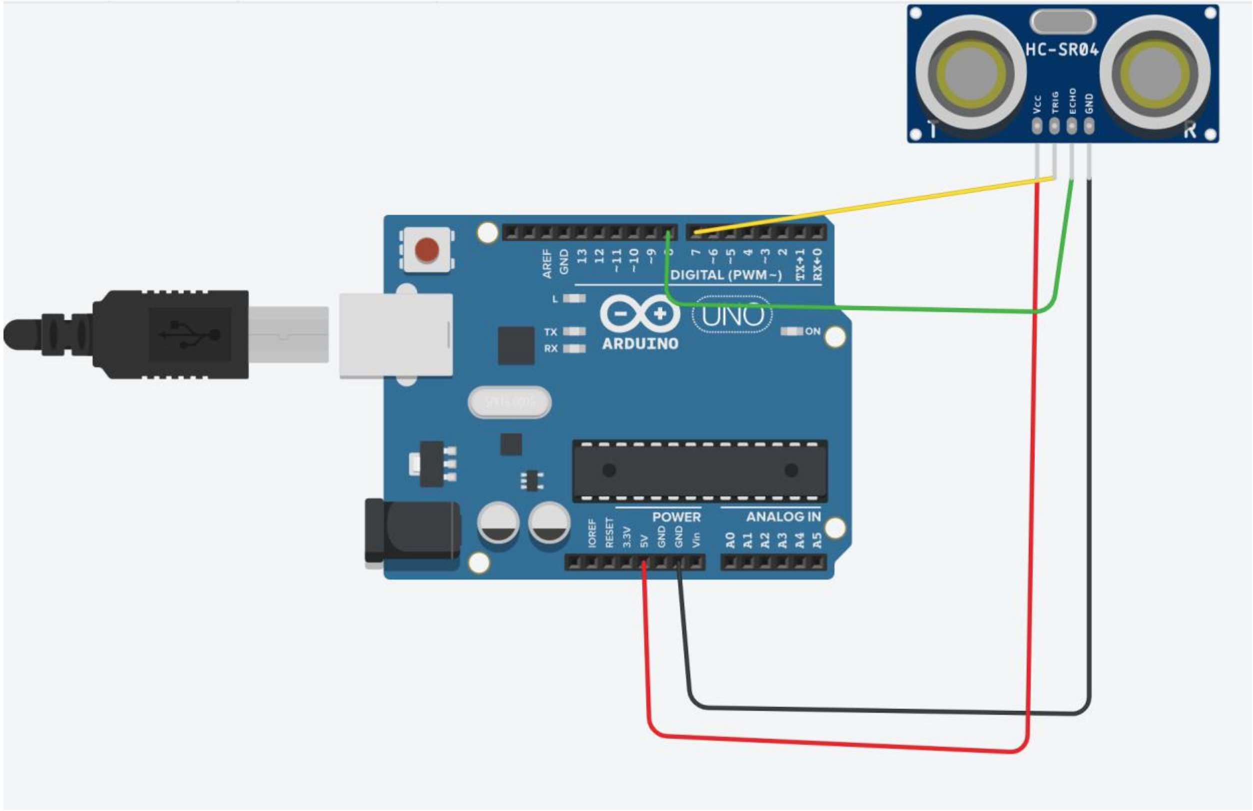
Design Assignment 4

Student Name: David Nakasone
Student #: 2001646072
Student Email: nakasd3@unlv.nevada.edu
Primary Github address: https://github.com/davenakasone
Directory: https://github.com/davenakasone/cpe301_David_Nakasone

1. COMPONENTS LIST AND CONNECTION BLOCK DIAGRAM w/ PINS

List of Components used: Atmega328pb, HC-SR04 ultra-sonic sensor

Block diagram with pins used in the Atmega328P:
UART on PD0/PD1
Trigger on PD7
Echo on PB0



2. INITIAL/MODIFIED/DEVELOPED CODE, all tasks

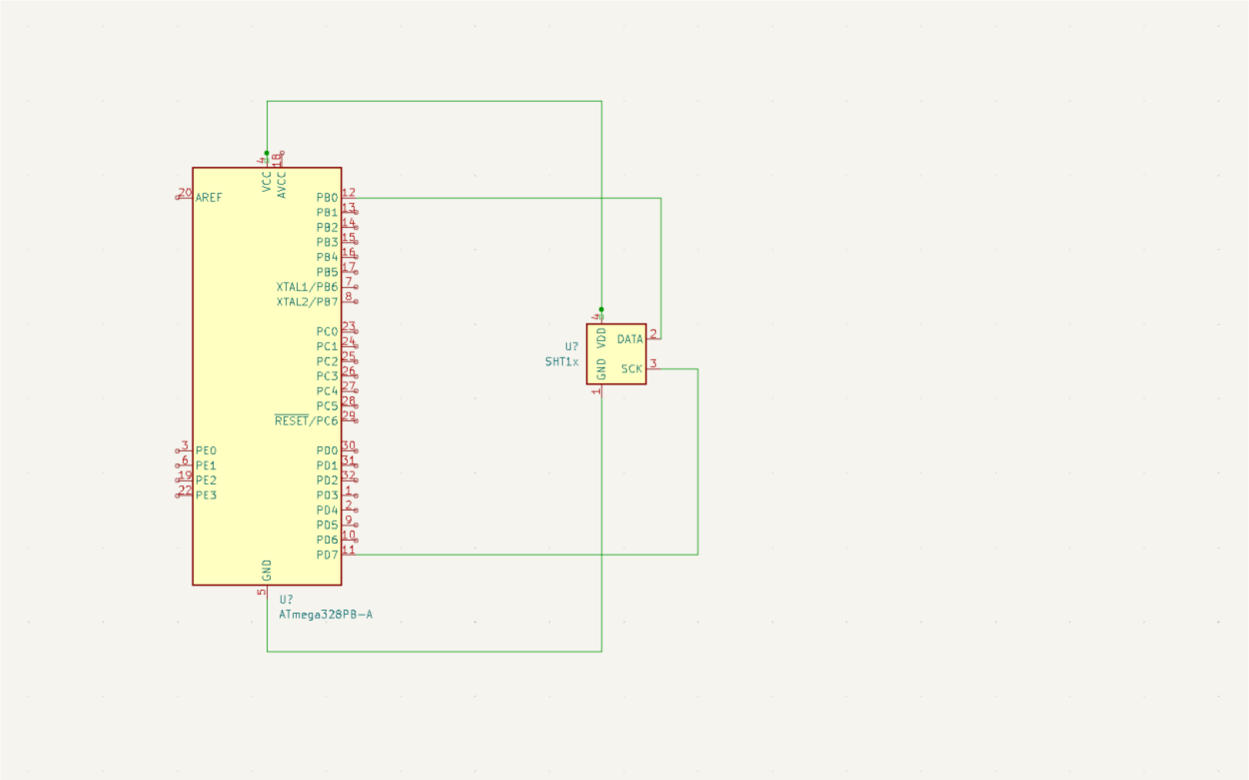
C code:

```
/*
    cpe301, da4

    interfacing with the ultra-sonic sensor

    PD7 for trigger
    PB0 for echo, must be use since timer1 has ICP1 on this pin

    preparations >>>
    [hammer] -> toolchain -> AVR/GNU Linker -> General -> check "Use vprintf library(-Wl, -u, vprintf)
    [hammer] -> toolchain -> AVR/GNU Linker -> Miscellaneous -> Other Linker Flags -> "-lprintf_flt"
    tools -> Data Visualizer -> Configuration -> External Connection -> Serial Port ->
    set the terminal's BAUD to 9600, open a terminal, add \r\n, COM3 "mEDBG"
*/

#define F_CPU 16000000UL
#define BAUD 9600
#define BAUD_PRESCALE (((F_CPU / (BAUD * 16UL))) - 1)
#define HELP_BUF 128
#define ECHO_PIN 0      // on PORTB, must be on PB0 for ICR1
#define TRIGGER_PIN 7   // on PORTD

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

volatile char helper[HELP_BUF];
volatile int timer_overflows;
volatile double distance;
volatile long tik_count;

void read_distance (void);
void usart_putc (char send_char);
void usart_puts (const char* send_str);


int main ()
{
    memset(helper, '\0', HELP_BUF);
    timer_overflows = 0;
    distance = 0;
    tik_count = 0;

    DDRD |= (1 << TRIGGER_PIN);                    // for outputting trigger signal
    PORTB = 0xFF;                                  // turn on pull up resistors for echo capture

    UCSR0B |= (1 << RXEN0) | (1 << TXEN0);       // turn on USART module, receive and transmit enabled
    UCSR0C |= (1 << UCSZ00) | (1 << UCSZ01);     // configure: asynchronous, 8-bit data, 1-bit stop
    UBRR0H = (BAUD_PRESCALE >> 8);               // sets baud rate
    UBRR0L = BAUD_PRESCALE;

    TIMSK1 = (1 << TOIE1);                       // enable timer1 overflow interrupt
    TCCR1A = 0;                                  // normal operation

    usart_puts("\r\n");
    usart_puts("initialized, program begins...\r\n");
    sei();

    while (1)
    {
        read_distance();
        _delay_ms(1000);
        usart_puts("\r\n");
        snprintf(helper, HELP_BUF-1, "%0.3lf  cm\r\n", distance);
        usart_puts(helper);
    }
    return EXIT_FAILURE;
}


////~~~~


void read_distance (void)
{
```

```
    timer_overflows = 0;                            // reset overflow counter
    // give 10us pulse to the trigger pin
    PORTD |= (1 << TRIGGER_PIN);                    // pulse begin
    _delay_us(10);                                  // wait 10us
    PORTD &= ~(1 << TRIGGER_PIN);                   // pulse end
    TCNT1 = 0;                                      // reset timer1 counter
    if (TIFR1 & (1 << ICF1))
    {
        TIFR1 = 1<<ICF1;                            // clear input capture flag
    }
    if (TIFR1 & (1 << TOV1))
    {
        TIFR1 = 1<<TOV1;                            // clear overflow flag
    }
    TCCR1B = 0x41;                                  // capture on rising edge, no prescaler
    // calculate echo width by input capture
    while ((TIFR1 & (1 << ICF1)) == 0) {}           // wait for rising edge
    if (TIFR1 & (1 << ICF1))
    {
        TIFR1 = (1 << ICF1);                        // clear input capture flag
    }
    if (TIFR1 & (1 << TOV1))
    {
        TIFR1 = (1 << TOV1);                        // clear overflow flag
    }
    TCCR1B = 0x01;                                  // capture on falling edge, no prescaler
    while ((TIFR1 & (1 << ICF1)) == 0) {}           // wait for falling edge
    // the distance is ready
    tik_count = ICR1 + (65535 * timer_overflows);   // get total tiks
    distance = (double)tik_count / 933;             // 16 MHz timer, 343 m/s speed of sound
}


////~~~~


void usart_putc (char send_char)
{
    while ((UCSR0A & (1 << UDRE0)) == 0) {}
    UDR0 = send_char;
}
```

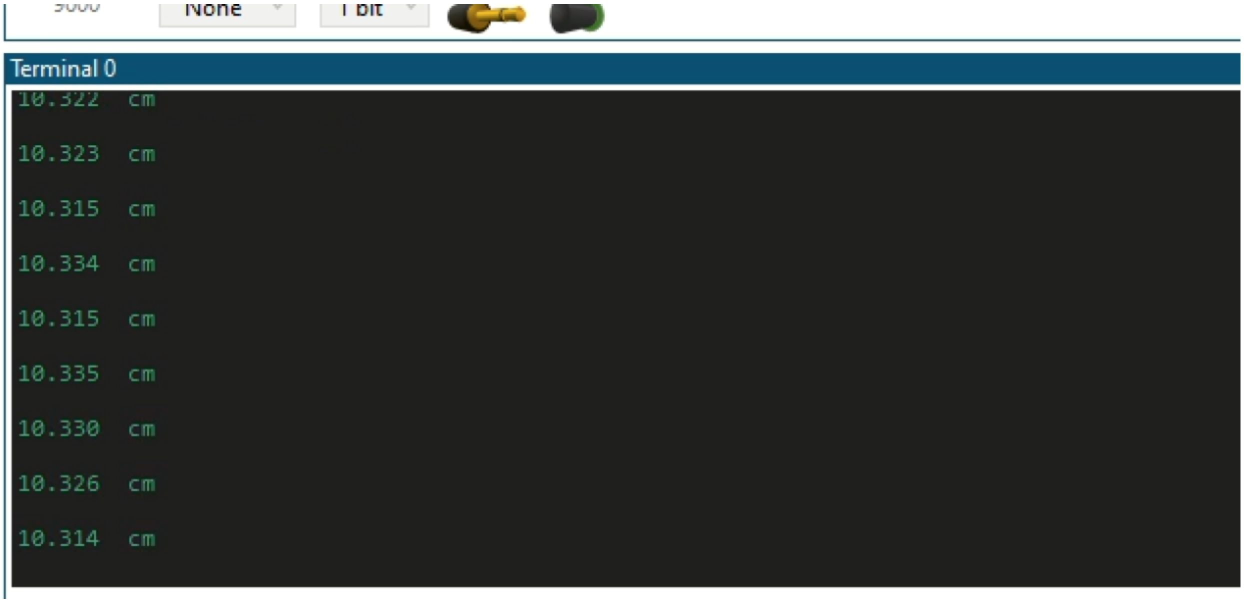
```
////~~~~


void usart_puts (const char* send_str)
{
    while (*send_str)
    {
        usart_putc(*send_str++);
    }
}


////~~~~


ISR(TIMER1_OVF_vect)
{
    timer_overflows++;
}


////////~~~~~~~~END>  da4.c
```

3. SCHEMATICS



4. SCREENSHOTS OF EACH TASK OUTPUT (ATMEL STUDIO OUTPUT)

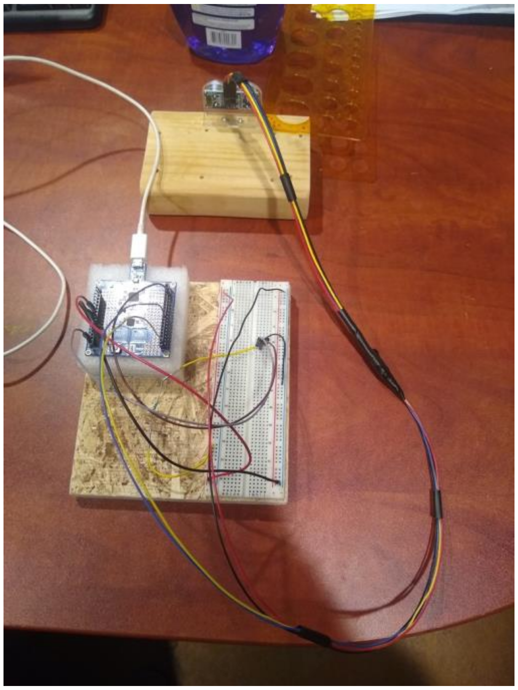
Terminal outputting distance in cm:



```
10.322  cm

10.323  cm

10.315  cm

10.334  cm

10.315  cm

10.335  cm

10.330  cm

10.326  cm

10.314  cm
```

5. SCREENSHOT OF EACH DEMO (BOARD SETUP)

The board and measurement:



Distance measurement, for verification:

6. VIDEO LINKS OF EACH DEMO

All tasks: https://youtu.be/CQrUfICaX4M

7. GITHUB LINK OF THIS DA

https://github.com/davenakasone/cpe301_David_Nakasone/tree/main/Design_Assignmentz/DA4

Student Academic Misconduct Policy

http://studentconduct.unlv.edu/misconduct/policy.html

"This assignment submission is my own, original work".

David Nakasone