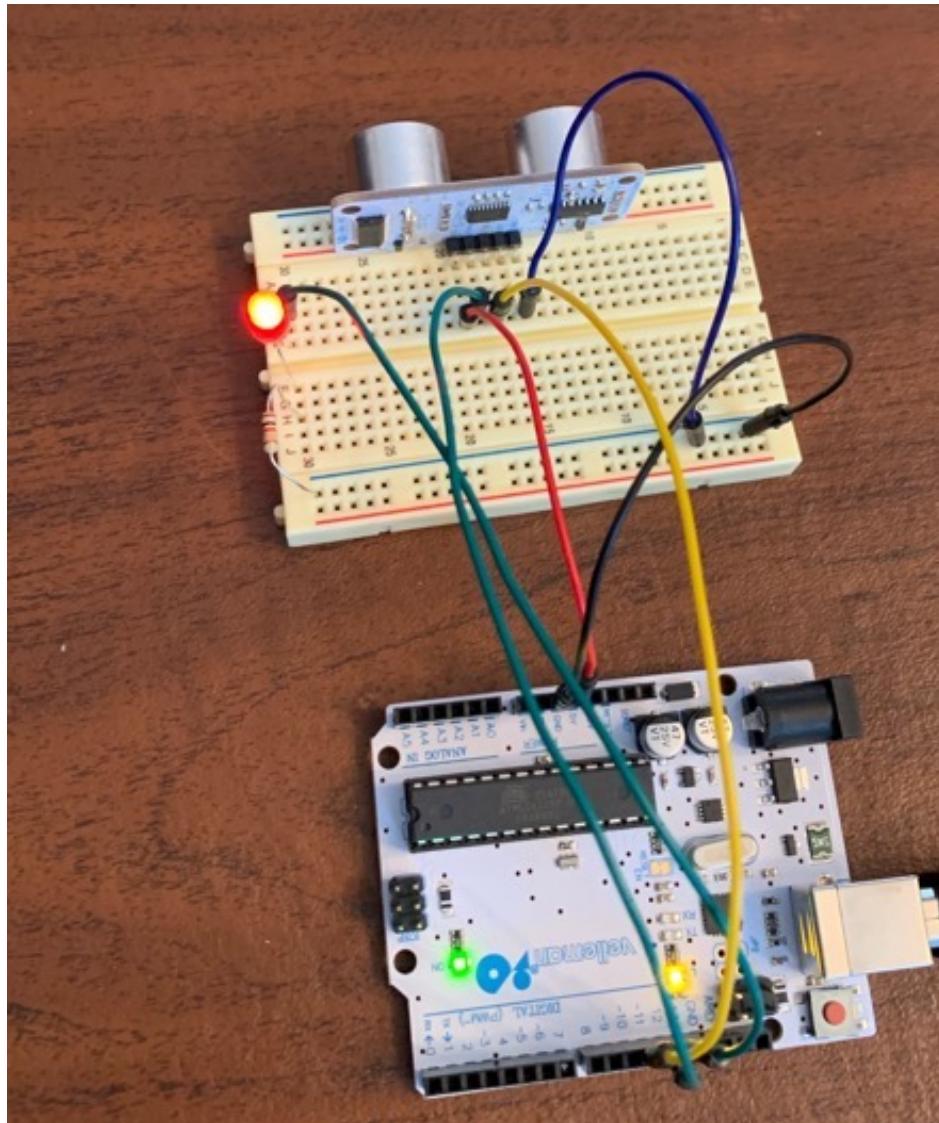


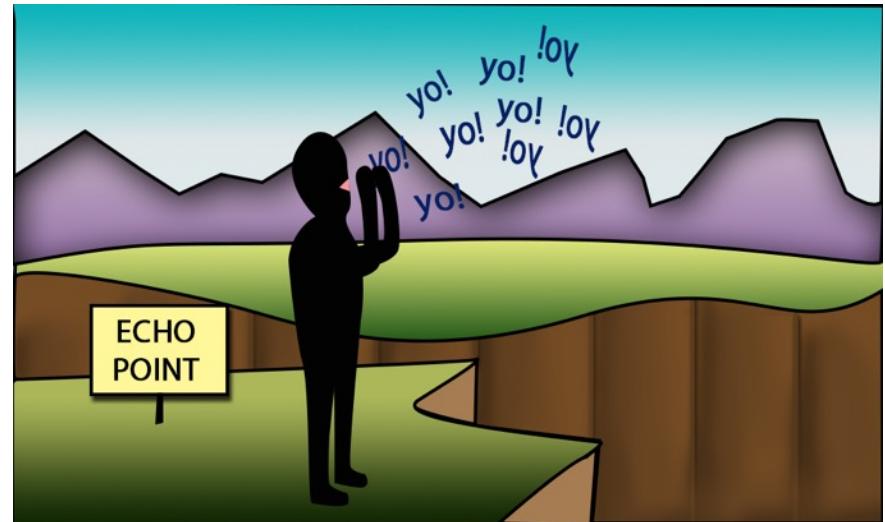
## L8.10 - Infrasound & Assignment 8.3



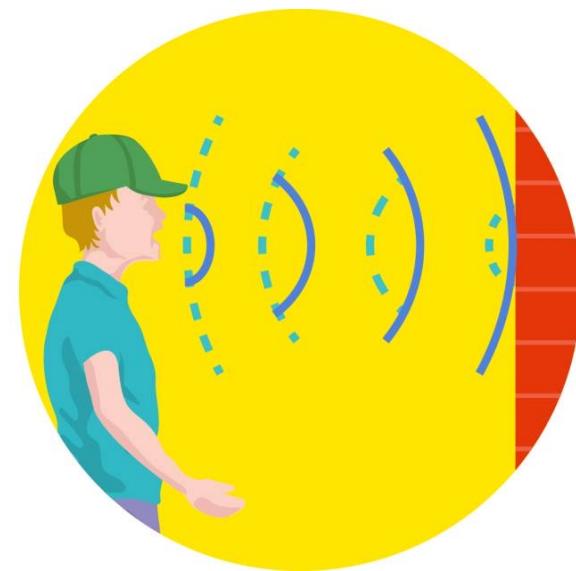
# Assignment #8.3

- Implement a distance measuring system using the ultrasound sensor & OLED display in your kit:
  - measure object distance from ~ 5 to 200 cm
  - Continuously display distance to detected object in cm using your 4-digit, 7-segment display with 1 cm resolution
- Due Saturday 5/10/25

# Echo



**human hearing range:**  
20 Hz – 20,000 Hz (20 kHz)



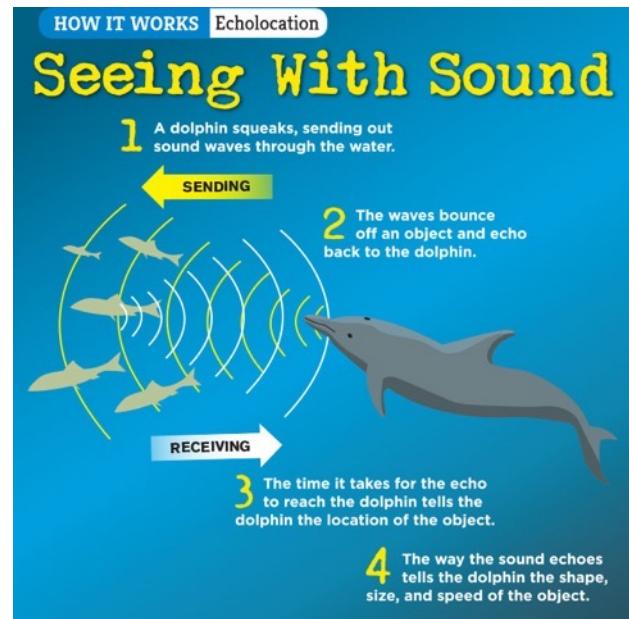
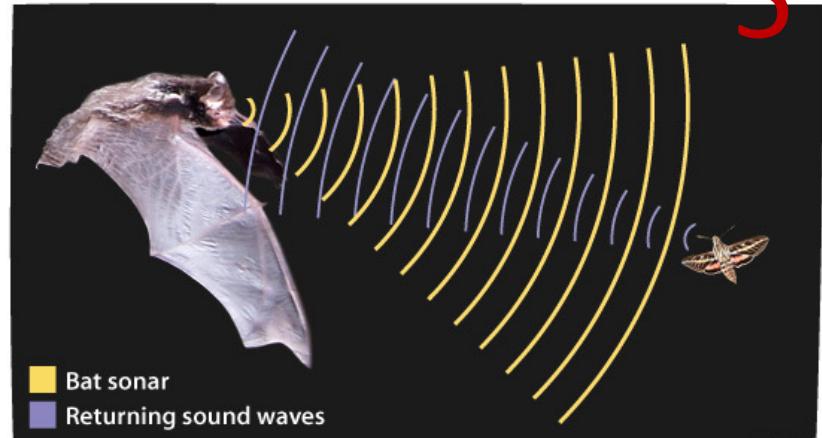


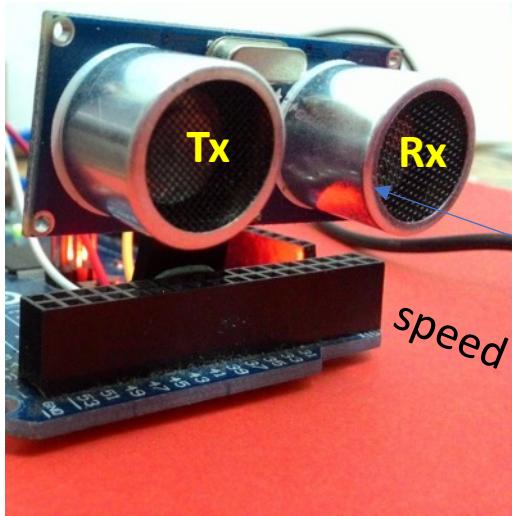
**Speed of sound:**  
**343 m/s**  
**741 mph**  
**1,125 ft/s**

speed of light:  
300,000,000 m/s  
983,571,056 ft/s

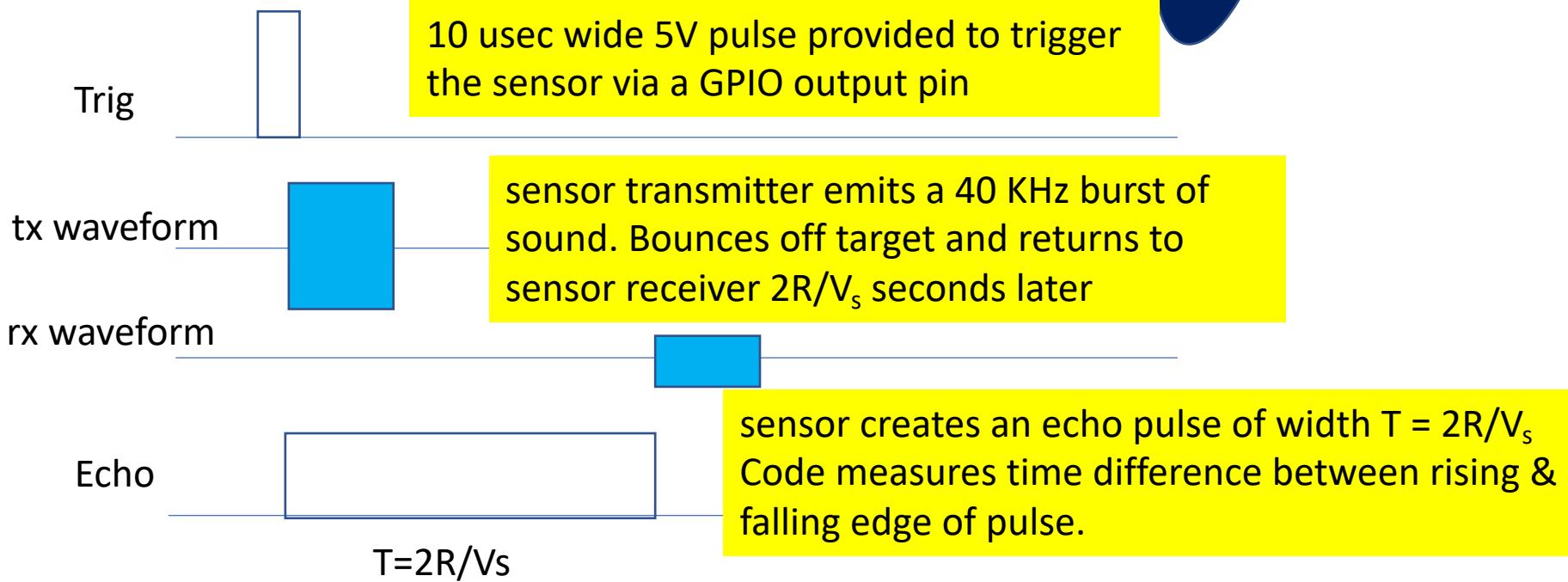
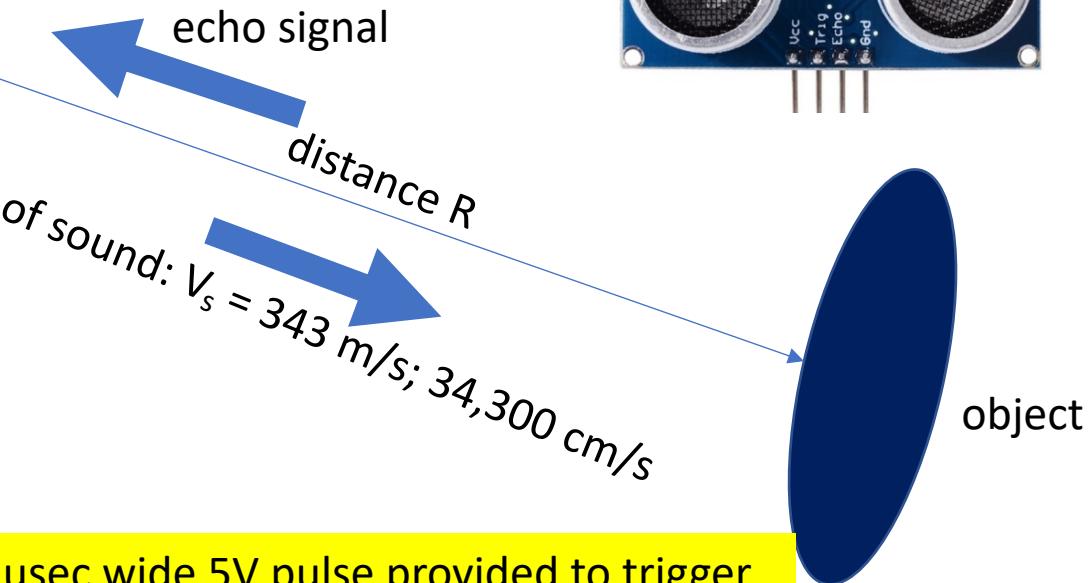
# Ultrasonic SONAR

- bats, dolphins “see” (navigate) using sound waves above 20KHz
- HC-SR04 is a 40 KHz SONAR

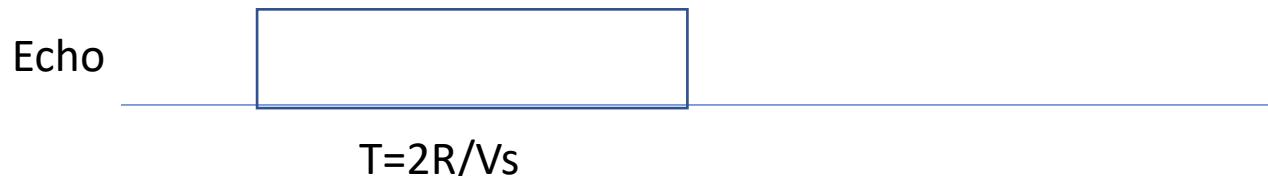




# Ultrasonic Distance Sensor HC-SR04 & ATmega328P



Speed of sound:  $V_s = 343 \text{ m/s} = 34300 \text{ cm/s} = 34300 \times 10^{-6} \text{ cm/uS} = 0.034 \text{ cm/uS}$



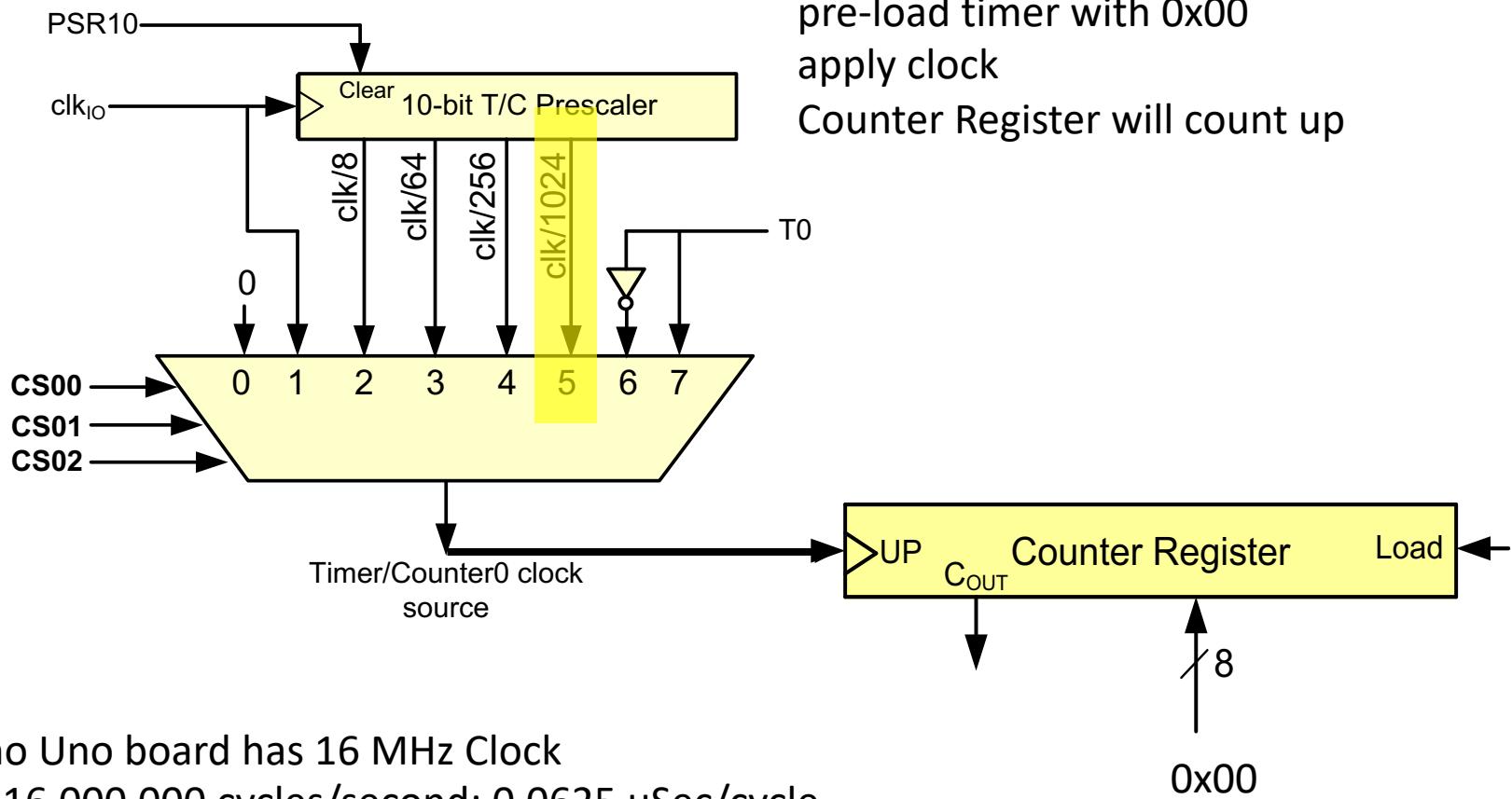
Example:  $R = 10 \text{ cm}; T = 2 * 10 / 0.034 = 588 \text{ uS}$

echo pulse width  $T = 2R/V_s$

distance (cm)	T (sec)	T (uSec)	Clock tics
0	0	0	0
10	0.00058309	583	9
20	0.00116618	1166	18
50	0.00291545	2915	46
75	0.00437318	4373	68
100	0.0058309	5831	91
150	0.00874636	8746	137
200	0.01166181	11662	182

# Timer clock  
tics at 64 uSec  
per tic

# Timer/Counter



Arduino Uno board has 16 MHz Clock

$\text{clk}_{IO} = 16,000,000 \text{ cycles/second}; 0.0625 \mu\text{Sec/cycle}$

$\text{clk}/8 = 2,000,000 \text{ cycles/second}, 0.5 \mu\text{Sec/cycle}$

$\text{clk}/64 = 250,000 \text{ cycles/second}, 4 \mu\text{Sec/cycle}$

$\text{clk}/256 = 62,500 \text{ cycles/second}, 16 \mu\text{Sec/cycle}$

$\text{clk}/1024 = 15,625 \text{ cycles/second}, 64 \mu\text{Sec/cycle}$

Let us measure the duration (width) of a pulse connected to a GPIO pin, for example PB1

start a timer counting (normal mode; 16 MHz clock/1024)

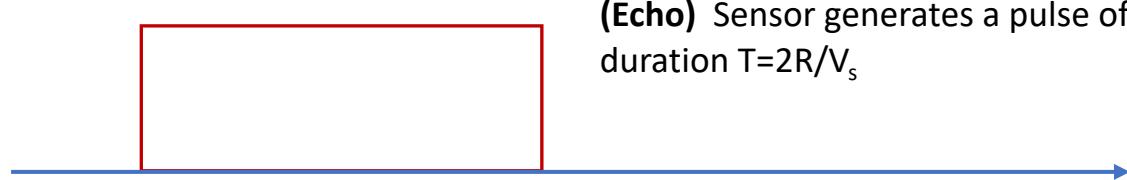
wait for the PB1 to go high

$T_r$  = # clock cycles to the rising edge (close to 0)

wait for PB1 to go low

$T_f$  = # clock cycles to the falling edge (between 0 and 182)

$$T = T_f - T_r$$



**(Echo)** Sensor generates a pulse of duration  $T=2R/V_s$

Speed of sound:  $V_s = 343 \text{ m/s} = 34300 \text{ cm/s} = 34300 \times 10^{-6} \text{ cm/uS} = 0.034 \text{ cm/uS}$

```
#define TRIG PB0
#define ECHO PB1
#include <avr/io.h> (and other includes)
int main(void){
```

```
    unsigned int timeToRisingEdge, timeToFallingEdge;
    unsigned int pulseWidth;
    float range;

    timer0_init();

    while(1){

        PORTB &= ~(1 << TRIG); // 5 usec pre-TRIG
        _delay_us(5);
        PORTB |= 1 << TRIG; // 10 us pulse
        _delay_us(10); // to ultrasound
        PORTB &= ~(1 << TRIG); // TRIG pin
        TCNT0 = 0;
        while ((PINB & (1<<ECHO))==0); //wait till ECHO goes high
        timeToRisingEdge = TCNT0;
        while (!(PINB & (1<<ECHO))==0); //wait till ECHO1 goes low
        timeToFallingEdge = TCNT0;
        pulseWidth = timeToFallingEdge - timeToRisingEdge;
        range = pulseWidth * 1.098; //one way distance to target in cm

        send_to_monitor();
    }
}
```

```
void timer0_init(void){
    TCCR0A = 0; //timer mode - normal
    TCCR0B = 0x05; //1024 prescaler
}
```

**(Trig)** 10 us pulse triggers the ultrasound sensor



transmitter emits a 40 kHz sound wave pulse

delay time is  $2R/V_s$

Pulse echo is detected by the receiver; clock is stopped

**(Echo)** Sensor generates a pulse of duration  $T=2R/V_s$

Speed of sound:  $V_s = 343 \text{ m/s} = 34300 \text{ cm/s} = 34300 \times 10^{-6} \text{ cm/uS} = 0.034 \text{ cm/uS}$

C ultrasound.c &gt; ...

```
1  /* ultrasound.c This code operates the HC-SR04 ultrasound sensor
2  and displays measured range on a serial monitor via UART.
3
4  Operation: code provides a 10us trigger pulse on PB1 then
5  waits for an echo pulse on PB0. Duration of echo pulse is
6  the round-trip time delay to the target. With Fcpu=16 MHz
7  and 1024 divider, Fclock = 16 MHz/1024 = 15.625 KHz. Each
8  clock pulse is therefore 1/15,625 = 64 us.
9  Speed of sound is 343 m/s or 34300 cm/s so each clock pulse
10 is 34300 cm/s x 64e-6 s = 2.195 cm round trip or 1.098 cm one way
11 8 bit counter with 256 states can measure distances 0 to 255*1.098
12 = 2.80 m or 9.2 feet with 1.1 cm resolution.
13 Timer0 will be set free-running, 0-255 then repeating
14 Note: objects >2.80m will have round-trip delay longer
15 than the clock count-up time; such echos are not printed to UART.
16 D. McLaughlin 4/14/22 Initial Code Writing for ECE--231 Spring 2022 */
17 #define TRIG PB1      //PB1 = pin 15
18 #define ECHO PB0      //PB0 = pin 14
19 #define RANGE_PER_CLOCK 1.098
20 #include <avr/io.h>
21 #include <util/delay.h>
22 #include <string.h> // Defines strlen() function
23 #include <stdlib.h> // Defines itoa() function
24 void uart_init(void);
25 void uart_send(unsigned char);
26 void send_string(char *stringAddress);
27 void timer0_init(void);
28 void send_to_monitor(unsigned char, unsigned char, float);
29
30 int main(void){
31     unsigned char rising_edge_clocks, falling_edge_clocks,
32     | | | | | | | | echo_width_clocks;
33     float target_range;
34     DDRB = 1<<TRIG;        // TRIG is output pin;
35     PORTB &= ~1<<TRIG;    // Set the TRIG pin low
36     uart_init();
37     timer0_init();         // Initialize timer0
38
39     while(1){
```

ultrasound.c &gt; ...

```
while(1){
    TCNT0 = 0;          // Load counter with 0
    PORTB |= 1<<TRIG; // These three lines of code
    _delay_us(10);     // Put a 10 usec pulse on the
    PORTB &= ~(1<<TRIG); // TRIG pin.

    // Wait till the ECHO pulse goes high
    while ((PINB & (1<<ECHO)) ==0);
    rising_edge_clocks = TCNT0; // Note the time
    // Now wait till the ECHO pulse goes low
    while (!(PINB & (1<<ECHO))==0);
    falling_edge_clocks = TCNT0;

    if (falling_edge_clocks > rising_edge_clocks){
        // Compute target range and send it to the serial monitor
        echo_width_clocks = falling_edge_clocks - rising_edge_clocks;
        target_range = echo_width_clocks * RANGE_PER_CLOCK;
        send_to_monitor(rising_edge_clocks, falling_edge_clocks, target_range);
    }
    _delay_ms(500); // Delay then go again
}

// Send info to the serial monitor via the UART
> void send_to_monitor(unsigned char t1, unsigned char t2, float range){...

// Initialize timer0: normal mode (count up), divide clock by 1024
> void timer0_init(){...

> // Send a string, char by char, to UART via uart_send() ...
> void send_string(char *stringAddress){...

// Initialize the UART
> void uart_init(void){...

// Sent a single character to serial monitor via UART
> void uart_send(unsigned char ch){...
```

C ultrasound.c &gt; ...

```
62 // Send info to the serial monitor via the UART
63 void send_to_monitor(unsigned char t1, unsigned char t2, float range){
64     char buffer[10];
65     send_string("Rising edge: ");
66     utoa(t1, buffer, 10);    //send the delay count to comm port
67     send_string(buffer);
68     //send_string(" clocks; ");
69     send_string(" Falling edge: ");
70     utoa(t2, buffer, 10);    //send the delay count to comm port
71     send_string(buffer);
72     //send_string(" clocks; ");

73
74     send_string(" Echo pulse width: ");
75     utoa(t2-t1, buffer, 10);
76     send_string(buffer);
77     send_string(" clock pulses.      Target Range = ");
78     dtostrf(range, 3, 0, buffer);
79     send_string(buffer);
80     send_string(" cm "); //PROBLEM?
81     dtostrf(range/2.54, 3, 0, buffer); //send the delay count to comm port
82     send_string(buffer);
83     send_string(" inch");
84     uart_send(13); //tx carriage return
85     uart_send(10); //tx new line
86 }
87
88 // Initialize timer0: normal mode (count up), divide clock by 1024
89 void timer0_init(){
90     TCCR0A = 0;          // Timer 1 Normal mode (count up)
91     TCCR0B = 5;          // Divide clock by 1024
92     TCNT0=0;            // Start the timer at 0
93
94 }
```

