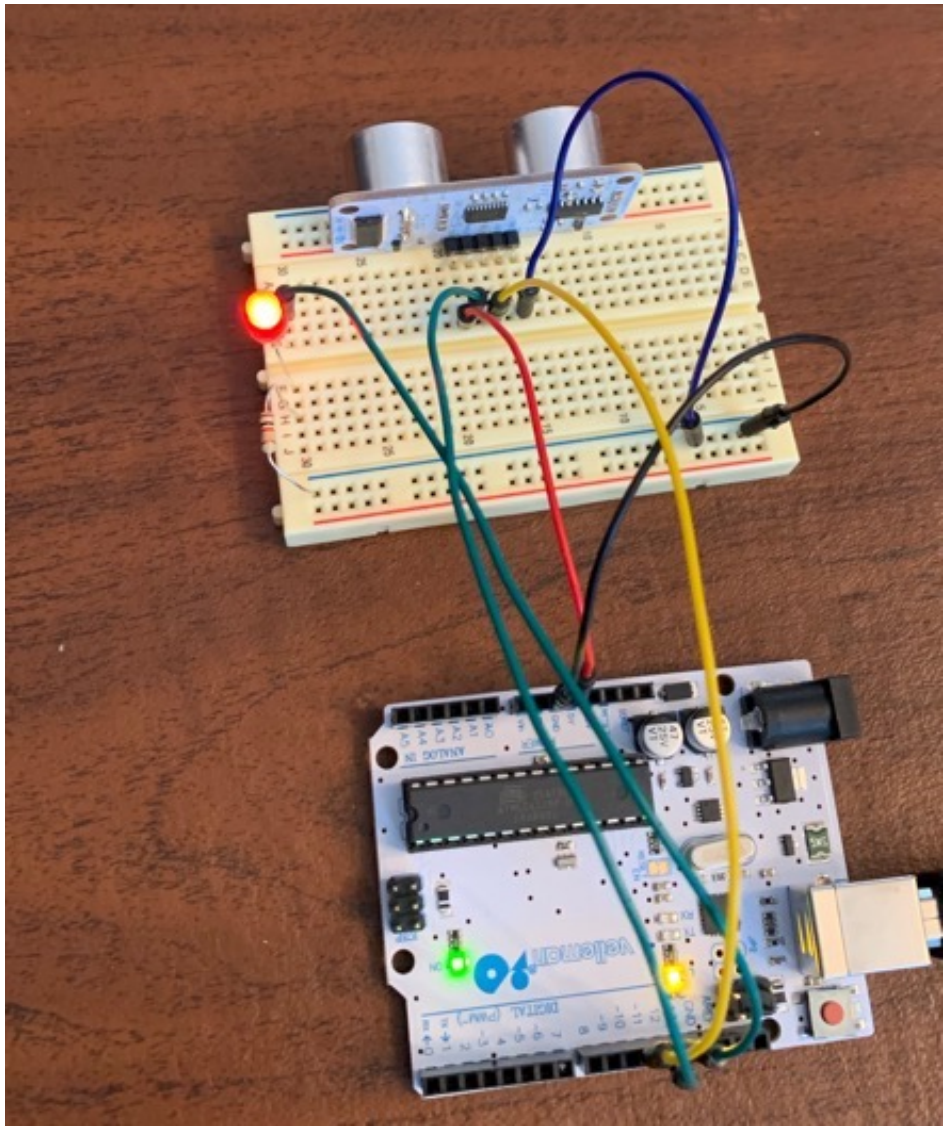
L8.10 - Infrasound & Assignment 8.3

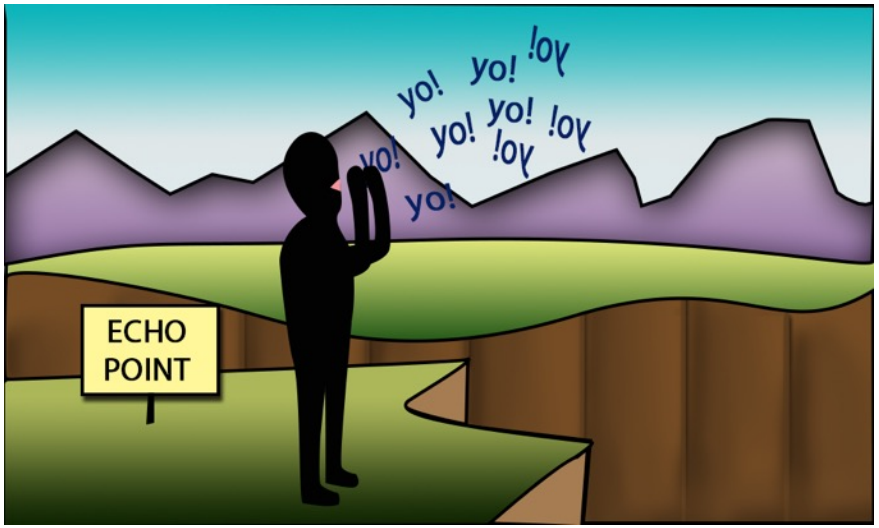


Assignment #8.3

- 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 5/11/24

Echo

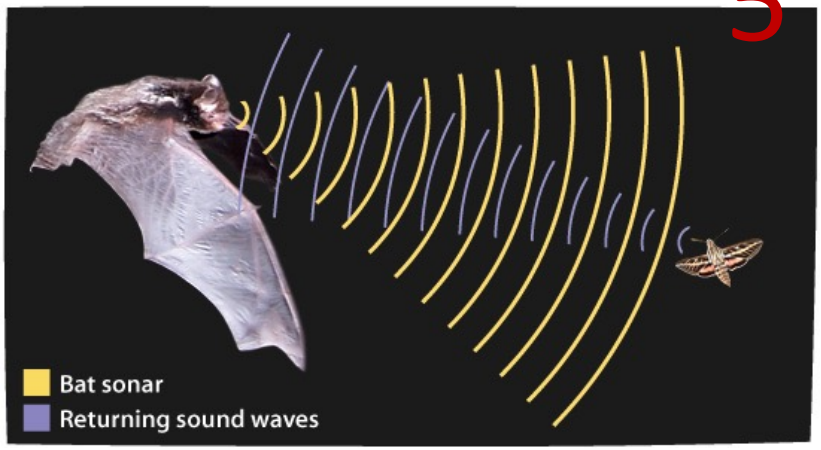


human hearing range:
20 Hz – 20,000 Hz (20 kHz)

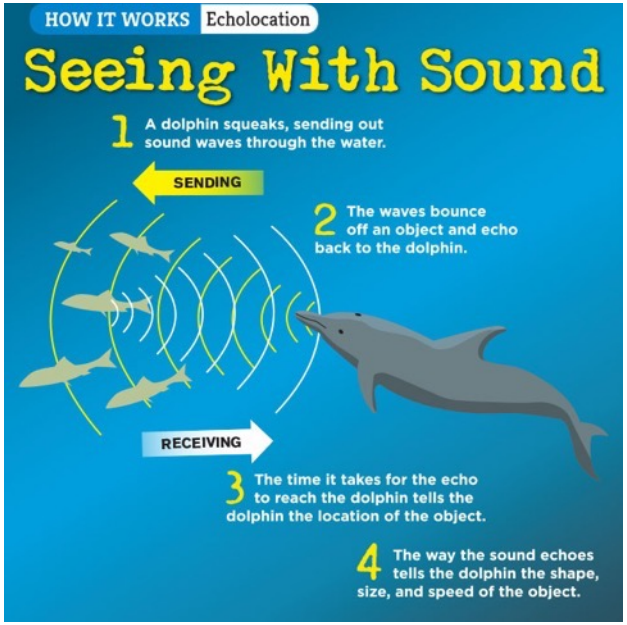
Speed of sound:
343 m/s
741 mph
1,125 ft/s

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

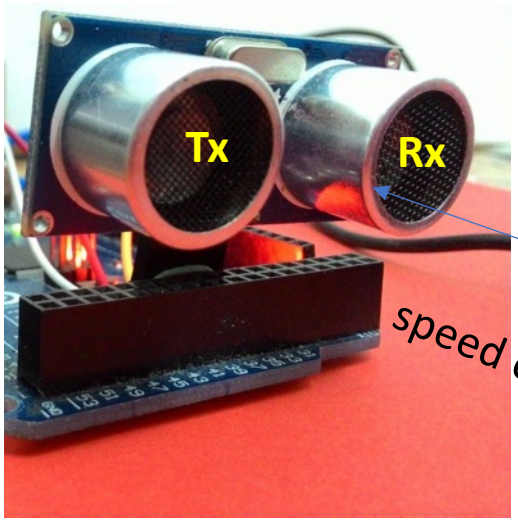
Ultrasonic SONAR

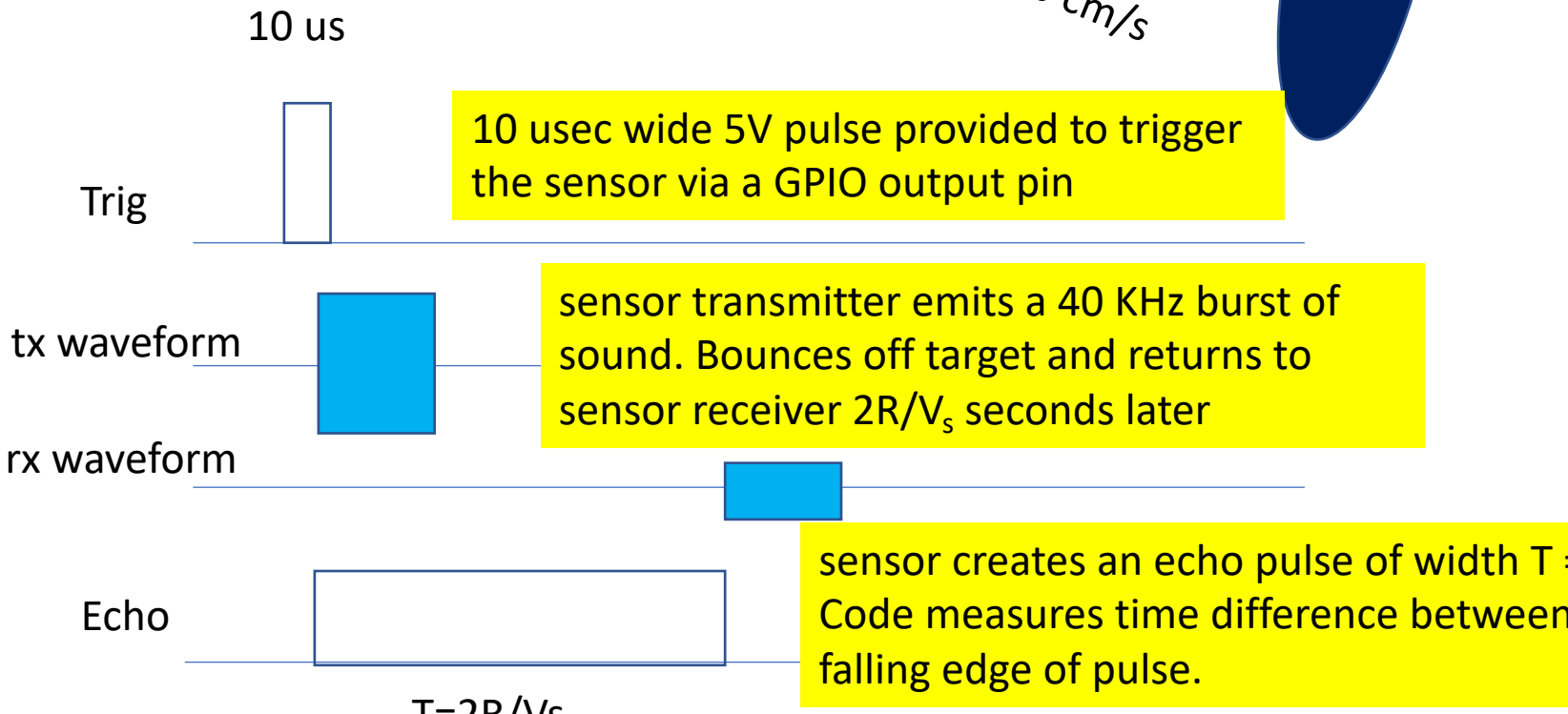


- bats, dolphins “see” (navigate) using sound waves above 20KHz
- HC-SRO4 is a 40 KHz SONAR

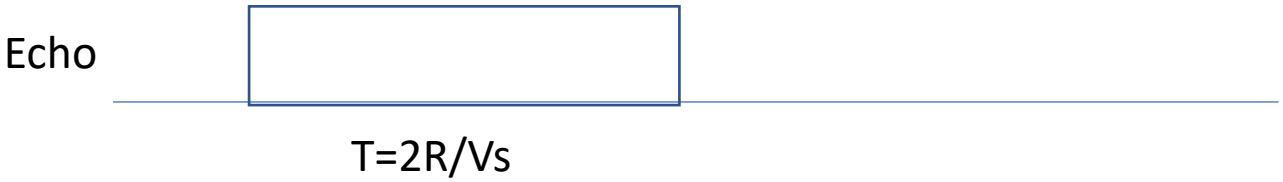


Ultrasonic Distance Sensor HC-SR04 & ATmega328P





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



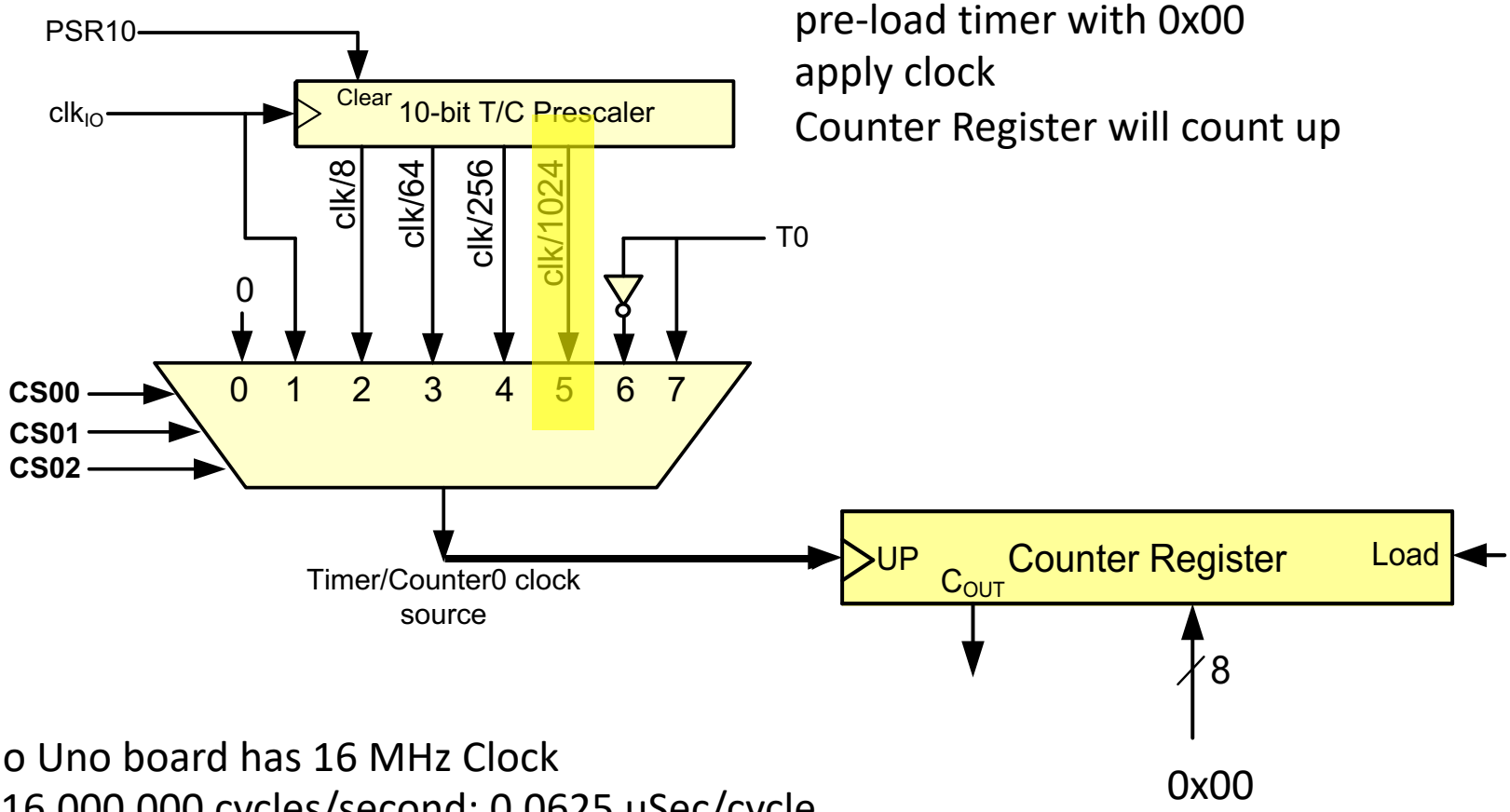
Example: R = 10 cm; T = 2*10/0.034 = 588 uS

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

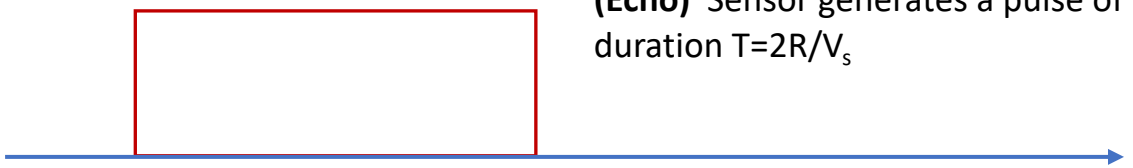


pre-load timer with 0x00
apply clock
Counter Register will count up

Arduino Uno board has 16 MHz Clock
clk$_{IO}$ = 16,000,000 cycles/second; 0.0625 uSec/cycle
clk/8 = 2,000,000 cycles/second, 0.5 uSec/cycle
clk/64 = 250,000 cycles/second, 4 uSec/cycle
clk/256 = 62,500 cycles/second, 16 uSec/cycle
clk/1024 = 15,625 cycles/second, 64 uSec/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
Tr = # clock cycles to the rising edge (close to 0)
wait for PB1 to go low
Tf= # clock cycles to the falling edge (between 0 and 182)
T = Tf - Tr



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

Speed of sound: V_s = 343 m/s = 34300 cm/s = 34300 x 10^{-6} cm/uS = 0.034 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
}
```

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

ultrasound.c > ...

```c
/* ultrasound.c This code operates the HC-SR04 ultrasound sensor
and displays measured range on a serial monitor via UART.

Operation: code provides a 10us trigger pulse on PB1 then
waits for an echo pulse on PB0. Duration of echo pulse is
the round-trip time delay to the target. With Fcpu=16 MHz
and 1024 divider, Fclock = 16 MHz/1024 = 15.625 KHz. Each
clock pulse is therefore 1/15,625 = 64 us.
Speed of sound is 343 m/s or 34300 cm/s so each clock pulse
is 34300 cm/s x 64e-6 s = 2.195 cm round trip or 1.098 cm one way
8 bit counter with 256 states can measure distances 0 to 255*1.098
= 2.80 m or 9.2 feet with 1.1 cm resolution.
Timer0 will be set free-running, 0-255 then repeating
Note: objects >2.80m will have round-trip delay longer
than the clock count-up time; such echos are not printed to UART.
D. McLaughlin 4/14/22 Initial Code Writing for ECE--231 Spring 2022 */
#define TRIG PB1    //PB1 = pin 15
#define ECHO PB0    //PB0 = pin 14
#define RANGE_PER_CLOCK 1.098
#include <avr/io.h>
#include <util/delay.h>
#include <string.h>  // Defines strlen() function
#include <stdlib.h>  // Defines itoa() function
void uart_init(void);
void uart_send(unsigned char);
void send_string(char *stringAddress);
void timer0_init(void);
void send_to_monitor(unsigned char, unsigned char, float);

int main(void){
    unsigned char rising_edge_clocks, falling_edge_clocks,
                                      echo_width_clocks;
    float target_range;
    DDRB = 1<<TRIG;         // TRIG is output pin;
    PORTB &= ~(1<<TRIG);    // Set the TRIG pin low
    uart_init();
    timer0_init();          // Initialize timer0

    while(1){
```

ultrasound.c > ...

```
    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){…
```

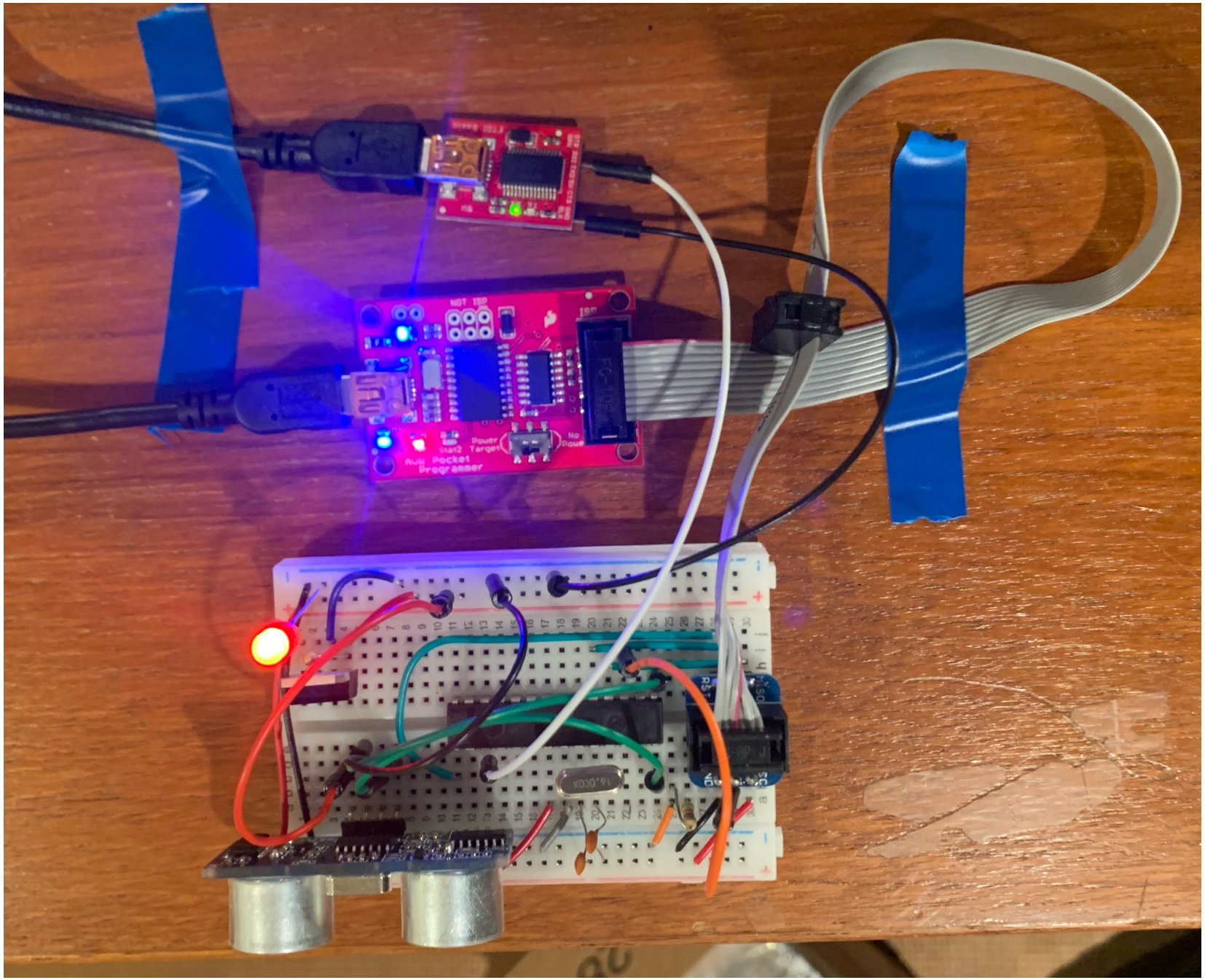
ultrasound.c > ...

```c
// Send info to the serial monitor via the UART
void send_to_monitor(unsigned char t1, unsigned char t2, float range){
        char buffer[10];
        send_string("Rising edge: ");
        utoa(t1, buffer, 10);   //send the delay count to comm port
        send_string(buffer);
        //send_string(" clocks; ");
        send_string(" Falling edge: ");
        utoa(t2, buffer, 10);   //send the delay count to comm port
        send_string(buffer);
        //send_string(" clocks; ");

        send_string(" Echo pulse width: ");
        utoa(t2-t1, buffer, 10);
        send_string(buffer);
        send_string(" clock pulses.     Target Range = ");
        dtostrf(range, 3, 0, buffer);
        send_string(buffer);
        send_string(" cm "); //PROBLEM?
        dtostrf(range/2.54, 3,0,buffer); //send the delay count to comm port
        send_string(buffer);
        send_string(" inch");
        uart_send(13); //tx carriage return
        uart_send(10); //tx new line
}

// Initialize timer0: normal mode (count up), divide clock by 1024
void timer0_init(){
    TCCR0A = 0;         // Timer 1 Normal mode (count up)
    TCCR0B = 5;         // Divide clock by 1024
    TCNT0=0;            // Start the timer at 0

}
```


Pocket
Programmer