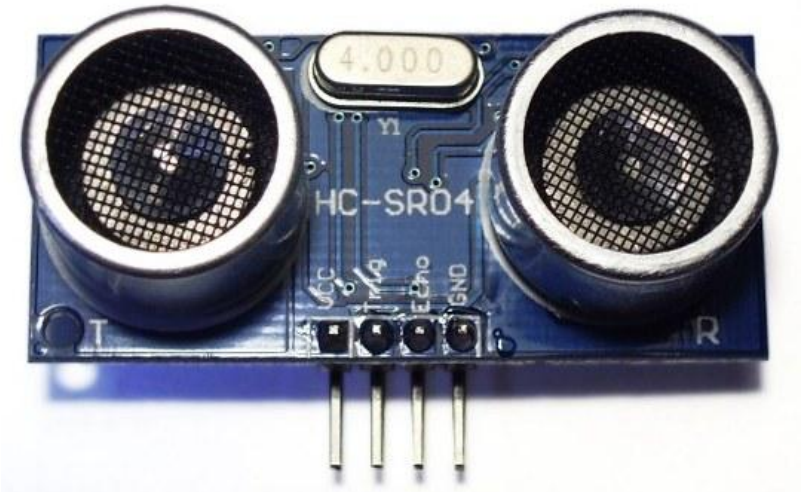


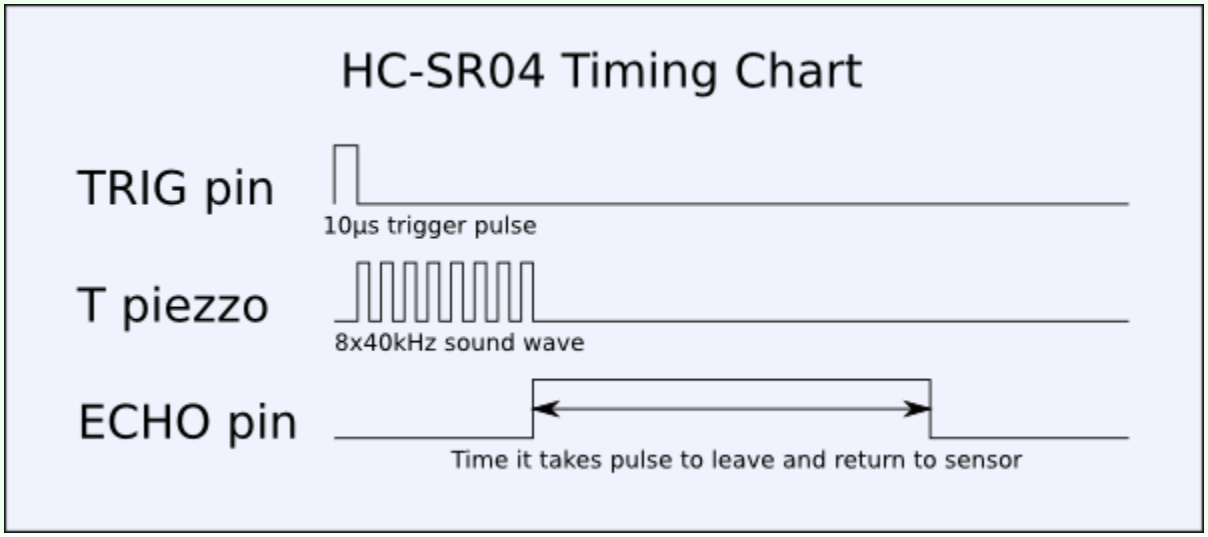
The HC-SR04 Ultrasonic Sensor
+
Atmel ATtiny13
+
AVR Assembly Language

The HC-SR04 (pictured above) is an inexpensive ultrasonic sensor that can sense not only if an object presents itself, like a PIR sensor, but can also sense and relay the distance to that object.

There is quite a lot of information on the web regarding this sensor coupled with AVR microcontrollers, but basically they are all done in C with hefty AVRs, like those found on Arduinos. I really wanted to see if I could get this sensor to work on my favorite AVR, the Tiny13, and do it in AVR assembly language. Happily, I've discovered that this sensor can be used quite easily on a Tiny13 in assembly language. And obviously, if it works on a Tiny13, it'll work on other AVR chips just as well.

A good starting point is the sensor datasheet from iteadstudio.com. Download it, print it out, and get familiar with page 2 of the document. There you will find a diagram of the pulse timing. What follows is pretty much that same information in somewhat more detail.

Basic Operation and Timing of the HC-SRO4 Ultrasonic Sensor



1. Make "Trig" (pin 2) on the sensor high for 10µs. This initiates a sensor cycle.
2. 8x40kHz pulses will be sent from the "T" transmitting piezzo transducer of the sensor, after which time the "Echo" pin on the sensor will go from low to high.
3. The 40kHz sound wave will bounce off the nearest object and return to the sensor.
4. When the sensor detects the reflected sound wave, the the Echo pin will go low again.
5. The distance between the sensor and the detected object can be calculated based on the length of time the Echo pin is high.
6. If no object is detected, the Echo pin will stay high for 38ms and then go low.

What follows is the mathmatics behind the code of the project. If you are only interested in the nuts and bolts of getting your sensor working then please feel free to jump directly to the hardware and software sections.

Basic Distance Calculation for Ultrasonic Sensing

At first glance there may seem to be a lot of math involved. In fact it's just a few very simple calculations to convert between speed, time, and distance.

The speed of sound: 340.29 m/s (meters per second).

We will be measuring distance to an object by the time it takes a sound wave to make a round trip to the object and back again, so the the useful number is actually:

The speed of sound to an object and back: 170.15 m/s.

Since our sensor can only detect items relatively nearby, let's change to more useful units by converting from m/s to µs/cm:

$$\frac{s}{170.15m} \times \frac{m}{100cm} \times \frac{1x10^{6}\mu s}{s} = \frac{58.772\mu s}{cm}$$

Time for pulse to travel 1cm to an object and then return to the sensor: 58.772µs.

Time/Distance Quiz

Question 1: A ping takes 150µs to hit an object and return. How far away is the object?

Answer:

$$150\mu s \times \frac{cm}{58.772\mu s} = \text{approx. } 2.55cm$$

Question 2: How long for a ping to hit and return from an object 30cm away?

Answer:

$$30cm \times \frac{58.772\mu s}{cm} = \text{approx. } 1{,}763\mu s \text{ or } 1.763ms$$

AVR Timings

The main piece of hardware used in this project (besides the sensor itself) is an Atmel ATtiny13(A) operating at the default fuse setting, which means it's running at 9.6MHz

with a /8 prescaler, which comes out to a 1.2MHz clock on the chip. This is the speed at which instructions on the chip are run. Note that some instructions may require 2 or more clock cycles to complete. See the ATtiny13 Datasheet for details.

AVR Clock Cycles (clks) to Time Conversions

```
==============  1.2MHz clock timings  ==============
1,200,000 clks  = 1s        1,000,000 clks = 833.3ms
  120,000 clks  = 100ms       100,000 clks =  83.3ms
   12,000 clks  = 10ms         10,000 clks =   8.3ms
    1,200 clks  = 1ms           1,000 clks = 833.3µs
      120 clks  = 100µs           100 clks =  83.3µs
       12 clks  = 10µs             10 clks =   8.3µs
      1.2 clks  = 1µs               1 clk  = 833.3ns
```

NOTE: The Tiny13 does not directly accept an external crystal ceramic resonator like other AVRs. Therefore the timings/measurements in this particular project are only accurate to ±10%. An external clock source *could* be used to drive an input pin, but it would hardly be worth the effort. If you really need more accuracy, I would suggest using an AVR that accepts an external crystal and, if you decide to go that route, the calculations that follow will change depending on the frequency of the crystal used.

Distance to AVR Clock Cycle Conversions

What we really need is a quick conversion between the AVR clock cycles and distance. We get that by combining the ultrasound timings with the AVR timings like this:

Distance to AVR Clock Cycles (clks)

```
   58.772µs   1.2 clks
   -------- X -------- =  70.526clks/cm
      cm         µs
```

Or to shed more light on what this means, the inverse:

```
      cm         µs
   -------- X -------- =  0.014179 cm/clks
   58.772µs   1.2 clks
```

So the time for a pulse to shoot out, hit something a centimeter away, and return to the sensor would be about 71 clock cycles. Likewise, in a single clock cycle, a distance of 0.14mm can be reckoned. (This is a theoretical value and cannot be attained by this project because the resolution of the sensor itself is only 3mm.)

Clock-Cycle/Distance Quiz

Question 1: A ping takes 2,110 clock cycles (clks) to hit an object and return. How far away is the object?

Answer:

$$2{,}110 \text{ clks} \times \frac{\text{cm}}{70.526 \text{ clks}} = \text{approx. } 30\text{cm}$$

Question 2: How many clock cycles will it take to sense an object 7cm away?

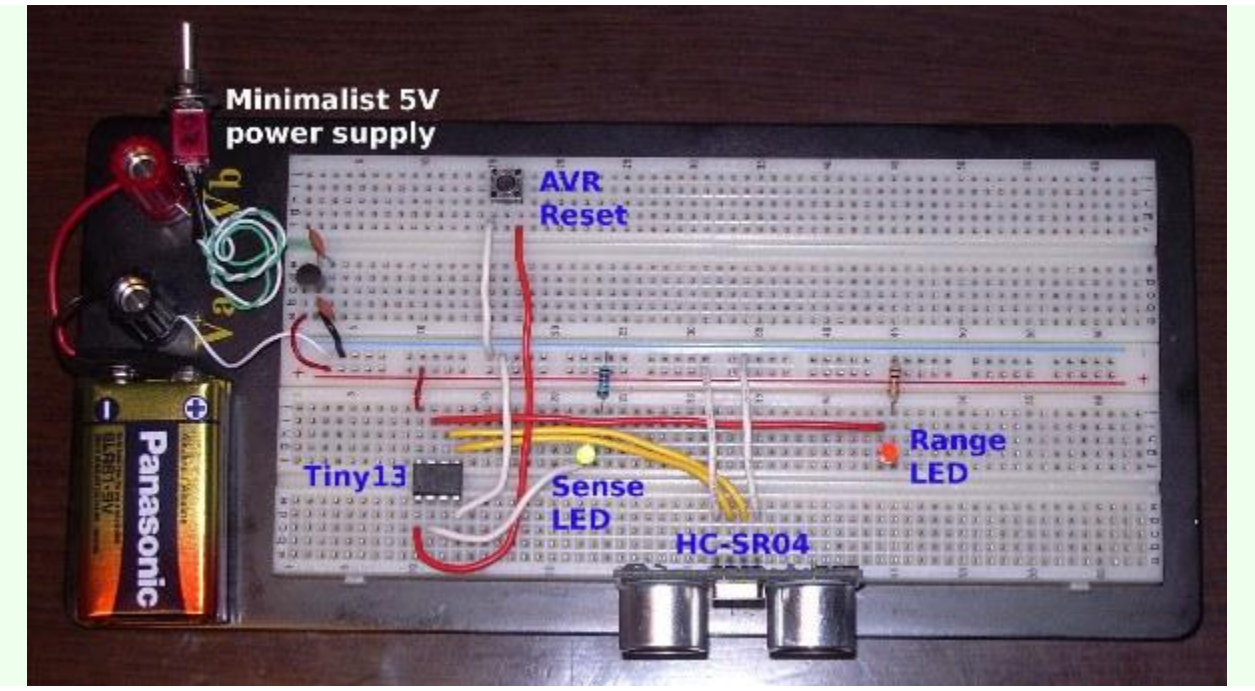
Answer:

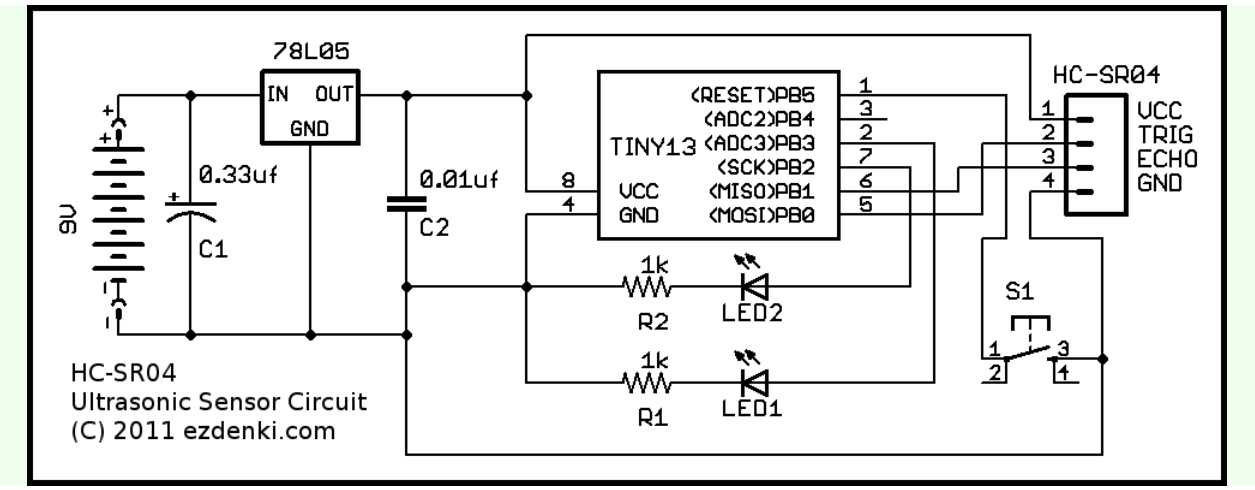
$$7\text{cm} \times \frac{70.526 \text{ clks}}{\text{cm}} = \text{approx. } 496 \text{ clks}$$

You can see that the clock-cycle to distance conversions won't end up in exact numbers of clock cycles. This isn't a huge issue since the sensor itself is only rated for a resolution of 0.3cm, so the 1.2MHz clock speed is fast enough to achieve that resolution, albeit with a 10% error factor based on the accuracy of the AVR's internal clock. In practice, without a crystal, the best we can do is probably closer to about 5mm~1cm of resolution -- YMMV.

The Hardware

Here is all that is needed to run the HC-SR04 on a Tiny13.





The sensor itself needs 5V to operate, so a simple 5V regulated supply running off of a 9V battery is used. Other than that, you have the sensor itself, the Tiny13, and 2 LEDs and their current limiting resistors. The "Sense LED" (LED1 on the schematic)

indicates if an object is detected, and the "Range LED" (LED2) tells us if the object is within the range specified in the program.

The Code

At this point, it's just a matter of counting clock cycles in the code. The code is heavily documented and includes most of the information on this page. It lists the number of clock cycles per instruction where it matters. There are 2 timing loops, a single-byte loop (BDELAY) and a word-length loop (WDELAY), to burn through clock cycles.

```
;---------------------------;
;  Ultrasonic Sensor Trial  ;
;---------------------------;
;  MikeShegedin
;  20.November.2011
;
;  Ver1.2
;     Cleanup of code and documentation
;
;  AVR:  ATTiny13A
;     AVR PIN  FUNCTION
;     =======  ======================
;      Pin2    PB3 SENSELED (sense LED) (optional)
;      Pin3    PB4 BUTTON (optional)
;      Pin5    PB0 TRIG pin on ultrasonic sensor
;      Pin6    PB1 ECHO pin on ultrasonic sensor
;      Pin7    PB2 RANGELED (range LED) (optional)
;
;  Ultrasonic Sensor:
;        HC-SR04 Ultrasonic ranging module
;
;  Operation:
;     Apply a 10µs signal on TRIG to start the sensing cycle. The HC-SR04
sensor will
;     send out a 40kHz pulse for 200µs. If something is in the path of the
ultrasonic
;     pulse, this pulse will bounch off that and return to the sensor. The
ECHO line on
;     the sensor will go high for 150µs~25ms depending on distance to
detected object.
;     If the ECHO line is high for 38ms it indicates that no return pulse was
detected.
;     There should be an interval of at least 50ms between subsequent TRIG
pulses
;     otherwise a lingering ultrasonic pulse may be detected during the next
ping cycle.
```


```
;
;
;================================================================================
;
;  Distance Calculation for ultrasonic sensing:
;     Speed of sound is 340.29 m/s. Pulse must be sent and then return.
;     Therefore time for a sound pulse (ping) to hit the target and return to
the sensor
;     is 170.15 m/s.
;
;     Convert to µs/s:
;
;           s        m     1x10^6(1E6)µs   58.772µs
;        ------- X ----- X ------------- = --------
;        170.15m   100cm        s             cm
;
;     This means it takes about 59µs for a sound pulse to travel 1cm to an
object and
;     then return to the sensor.
;
;  Compute distance based on time:
;     Example 1: A ping takes 150µs to hit an object and return. How far away
is the
;                object?
;
;                   cm
;        150µs X -------- = approx. 2.55cm
;                58.772µs
;
;     Example 2: How long for a ping to hit and return from an object 30cm
away?
;
;               58.772µs
;        30cm X -------- = approx. 1,763µs or 1.763ms
;                  cm
;
;
;================================================================================
;
;  AVR TIMINGS
;     Use default ATTINY13 timings (9.6MHz with /8 prescaler) for a 1.2MHz
clock.
;     Fuse setting: -U lfuse:w:0x6a:m -U hfuse:w:0xff:m
;
;     =============  1.2MHz clock timings  =============
;
;     1,200,000 clk  = 1s        1,000,000 clk = 833.3ms
;       120,000 clk  = 100ms       100,000 clk =  83.3ms
;        12,000 clk  = 10ms         10,000 clk =   8.3ms
```

```
;          1,200 clk  = 1ms           1,000 clk = 833.3µs
;            120 clk  = 100µs           100 clk =  83.3µs
;             12 clk  = 10µs             10 clk =   8.3µs
;            1.2 clk  = 1µs               1 clk = 833.3ns
;
;  NOTE: Timing accuracy of the AVR clock without a crystal is only about
±10%. Much
;        higher accuracy can be achieved by using an external crystal if
desired. Of
;        course, if an external crystal is used, clock timings will be based
on the
;        value of the crystal.
;
;
================================================================================
========
;
;     DELAY: Double Byte Timing Loop Routine
;        By counting the number of clock cycles for the routine, including
loading n
;        into YH and YL, calling the delay routine, and returning back, the
number of
;        clock cycles (clks) was found to be as follows:
;
;        Delay in clock pulses (clks) based on n: Time(clks) = 4n + 8
;
;        Time delay in µs based on n:
;                                        10µs
;           Time(µs) = (4n+8)clks X ------
;                                      12clks
;
;           Therefore, simplified:
;
;                        10n + 20
;           Time(µs) = --------
;                           3
;
;        Compute n for the desired time T(µs) (solve for n):
;
;                3T(µs) - 20
;           n = ===========
;                     10
;
;     Example: What should n be for a 100µs delay?
;
;                (3 x 100) - 20
;           n = --------------
;                     10
;
;           n = 28
;
```


```
;     Example 2: What should n be for a 50ms delay?
;
;                                                1000µs
;        First, convert ms to µs --> 50ms x ------ = 50,000µs
;                                                 1ms
;
;               (3 x 50,000) - 20              (3 x 50,000)
;           n = -----------------      or  n = ------------
;                        10                          10
;
;           n = 14,998                 or  n = 15,000
;
;           Note that the "- 20" term can probably be dropped for millisecond
;           calculations since doing so results in a less than 1% error in
these cases.
;
;
================================================================================
========
;
;  Conversion Summary
;
;  Time/Distance Ratio from start of sending ping to the time it is received
back:
;
;     58.772µs
;     --------
;        cm
;
;  AVR Clock/Time Ratio (for 1.2MHz AVR clock)
;
;     1.2clks
;     -------
;       µs
;
;  AVR Clock clks/Distance Formulae
;
;     70.526clks      0.014179cm
;     ----------  or  ----------
;         cm              clks
;
;  Timing of pulse indicating no object sensed (according to sensor
datasheet):
;
;     38ms  which is the equivilant of 31,667clks.
;
;
================================================================================
========
;
```


```
.INCLUDE "TN13DEF.INC"     ;  ATTINY13 DEFINITIONS


   .EQU  SENSELED=PORTB3   ;  SENSELED pin (Output on AVR)
   .EQU  RANGELED=PORTB2   ;  RANGELED pin (Output on AVR)
   .EQU  BUTT=PORTB4       ;  Button pin (Input on AVR)
   .EQU  TRIG=PORTB0       ;  Sensor TRIG pin (Output on AVR, input on
sensor)
   .EQU  ECHO=PORTB1       ;  Sensor ECHO pin (Input on AVR, output on
sensor)

   .EQU  BDV10usCNT=1      ; Number of iterations for (approx.) 10µs delay
   .EQU  WDV50msCNT=10415  ; WDELAY count for 50ms delay
   .EQU  BDV1cmCNT=19      ; Number of iterations for BDELAY loop to get 1cm
                           ; resolution.
   .EQU  BDV2mmCNT=6       ; Number of iterations for BDELAY loop to get 2mm
                           ; resolution.


   .DEF A = R16            ;  GENERAL PURPOSE ACCUMULATOR
   .DEF BDV = R17          ;  Counter used in BDELAY routine.
   .DEF CNT = R18          ;  CNT will be the no. of 150µs intervals of the
ECHO
;  ECHO pulse, each of which indicates 2.55cm. CNT=0
;  indicates object not detected.
   .DEF DTEST=R19          ;  DTEST is used to test for particular distances.
;  Will be 1 if distance test passes or 0 if failed
   .DEF DRNG1=R20          ;  Distance in increments of CNT for DNEARER and
the shorter
;  distance for DBETWEEN.
   .DEF DRNG2=R21          ;  Longer distance (in increments of CNT) for
DBETWEEN.

        .ORG 0000
RJMP  ON_RESET       ;  GO HERE WHEN CHIP IS TURNED ON OR RESET


        ON_RESET:
        ldi     A, low(RAMEND) ; 1   Set Stack Pointer to end of RAM (1 BYTE
ON TINY13)
        out     SPL,A

ldi   A, (1<<SENSELED) | (1<<RANGELED) | (1<<TRIG)
out   DDRB, A         ; Set output pins
sbi   PORTB, BUTT     ; Enable pullup on button
cbi   PORTB, ECHO     ; Disable pullup on ECHO line


;   rcall WAIT4BUTTON

        MAIN_LOOP:
```

```
rcall ONTRIG             ; Start 10µs trigger pulse by setting TRIG high.
ldi   BDV, BDV10usCNT    ; Set DELAY count for delay of approx. 10µs.
rcall BDELAY             ; Run the delay.
rcall OFFTRIG            ; Turn off the trigger.

rcall WAIT4ONECHO    ; Wait for start of ECHO pulse
rcall GETECHOPULSE   ; Get length of echo pulse in CNT.
                         ; At this point, CNT will have either the number
of
                         ; 2.55cm intervals to the object or will be zero,
                         ; indicating the object was not detected.

tst   CNT            ; Check CNT for zero, indicating object not detected.
breq  NOOBJECT
rcall ONSENSELED     ; Turn on SENSELED indicating object was detected.

ldi   DRNG1,30
ldi   DRNG2,31
rcall DBETWEEN       ; Test for object within certain distance.
tst   DTEST          ; DTEST will be 1 if test passed (object is near).
breq  NOTNEARER
rcall ONRANGELED
rjmp  MAINWAIT
NOTNEARER:
rcall OFFRANGELED
rjmp  MAINWAIT


   NOOBJECT:
rcall OFFSENSELED          ; Turn OFF SENSELED indicating object not
detected.
rcall OFFRANGELED          ; Turn OFF RANGELED since distnace test must be
false.

   MAINWAIT:                     ; The total sensing cycle from start to
finish should
ldi YH, HIGH(WDV50msCNT)   ; take at least 50ms in order to allow existing
ldi YL, LOW(WDV50msCNT)    ; ultrasonic pulses in the room to completely fade
and
rcall WDELAY               ; not interfere with the subsequent sensing cycle.
                               ; Since the time from triggering the pulse
to minimum
                               ; ultrasonic pulses in the room to
completely fade and
                               ; not interfere with the subsequent sensing
cycle.
                               ; Since the time from triggering the pulse
to minimum
                               ; pulse detection cycle is about 50µs, a
delay of 50ms
                               ; here is probably appropriate.
```


```
  END:
       RJMP  MAIN_LOOP          ; MAIN LOOP DOES NOTHING (YET).


; -------------------------------------
;  DNEARER
;  Checks CNT to determine if object was detected closer than a particular
distance.
;  The distance will be determined by the delay in GETECHOPULSE.
;  DRNG1 must be loaded with the desired range value before this routine is
called.

DNEARER:
ldi    DTEST,1         ; Assume test passed
cp     CNT,DRNG1
brloDNret           ; If CNT lower (closer) then exit with 1.
ldi    DTEST,0         ; Otherwise, farther out, so exit with 0
DNret:
ret

; -------------------------------------
;  DBETWEEN
;  Checks CNT to determine if object was within a range of distances.
;  The distance will be determined by the delay in GETECHOPULSE.
;  DRNG1 and DRNG2 must be loaded with range values before this routine is
called.

DBETWEEN:
ldi    DTEST,0         ; Assume test does not pass
cp     CNT,DRNG1       ; Check lower distance
brloDBret           ; If CNT lower (closer) then exit with 0.
cp     CNT,DRNG2       ; Check upper distance
brplDBret           ; If CNT higher (farther) then exit with 0.
ldi    DTEST,1         ; Otherwise, farther out, so exit with 1
DBret:
ret

; -------------------------------------
;  WAIT4BUTTON
;  Stops until button is pressed (grounded) and released, then returns.

WAIT4BUTTON:
sbic      PINB, BUTT          ; Check button and skip if high (no press)
rjmp      WAIT4BUTTON
W4B1:
sbis      PINB, BUTT
rjmp      W4B1
ret

; -------------------------------------
;  WAIT4ONECHO
```


```
;  Waits for echo to start from ultrasonic sensor.

WAIT4ONECHO:
sbis     PINB, ECHO         ; Check ECHO line of sensor and skip if high
rjmp     WAIT4ONECHO
ret

; -------------------------------------
;  GETECHOPULSE
;  Loop timing is set up to generate one iteration/count per a fixed length
(ex. 1cm.)
;  According to the timings listed above, we'll want 70 or 71 clks for 1cm
increments.
;  The maximum pulse length for when an object is not detected is 38ms.
However a
;  shorter maximum distance can be regarded as an object not detected.
;
;  Everything between GEP1 and GEPret will be executed CNT times. The rest of
the
;  routine has a 1-time 4clks overhead including the time to call the
routine. These
;  clock cycles should be considered negligiable. A delay routine can be
inserted or
;  called between the breq and sbic instructions such that CNT can be
associated with
;  a set distance. Use NOP instructions to fine tune this delay section.
;
;  If CNT overflows to 0 on the 256th count, the routine will exit and this
should be
;  considered an object-not-detected condition.

GETECHOPULSE:
clr      CNT            ; 1   Clear CNT
GEP1:
inc      CNT            ; 1
tst      CNT            ; 1   Increment CNT (assume first pulse)
breqGEPret         ; 1/2 If CNT overflows to zero, exit routine.
                           ;     Timing: false:1clk, true:2clk
ldi      BDV,BDV2mmCNT  ; *   Load delay count here
rcall    BDELAY         ; *   *time is 7+3(BDVxxxCNT) incl. setting BDV and
this
                           ;      rcall.
sbic     PINB, ECHO     ; 1/2 Check ECHO pin. If no ECHO, skip over next
instruction.
                           ;     Timing: false:1clk, true:2clk
rjmp     GEP1           ; 2

GEPret:
ret                     ; 4
```


```
; -------------------------------------
; BDELAY
; A simple 1-byte loop delay
; See notes above for timing specifications
; BDV must be loaded with timing count before this routine is called!

BDELAY:                    ; 3   # of clks to call here
dec      BDV               ; 1   Decrement counter
brne     BDELAY            ; 1/2 Loop back up if not zero
                           ;     Timing: false:1clks, true:2clks
ret                        ; 4   Else return


; -------------------------------------
; WDELAY
; A simple 2-byte (Word) loop delay
; See notes above for timing specifications
; YH and YL must be pre-loaded with desired n *before* this routine is called
as follows:
;     ldi      YH, HIGH(n)
;     ldi      YL, LOW(n)
;     rcall    WDELAY

WDELAY:
sbiw     YH:YL,1           ; 2   Word-level decrement
brne     WDELAY            ; 1/2 Loop back up if not zero
                           ;     Timing: false:1clks, true:2clks
ret                        ; 4   Else return

; -------------------------------------
; TOGGLE_LEDx
; Toggles either SENSELED or RANGELED

TOGGLESENSELED:
sbi      PINB,SENSELED
ret

TOGGLERANGELED:
sbi      PINB,RANGELED
ret

; -------------------------------------
; ON/OFF outputports
; Switches output port ON or OFF

ONSENSELED:
   SBI   PORTB,SENSELED
ret
ONRANGELED:
   SBI   PORTB,RANGELED
ret
```


```
ONTRIG:
   SBI   PORTB,TRIG
ret
OFFSENSELED:
   CBI   PORTB,SENSELED
ret
OFFRANGELED:
   CBI   PORTB,RANGELED
ret
OFFTRIG:
   CBI   PORTB,TRIG
ret
;------------------;
;  END OF PROGRAM  ;
;------------------;
```