Lab 2c: Ultrasonic sensor

Aim

The aim of this exercise is to measure distance using the SRF04 ultrasound transceiver and an AVR ATmega128 controller. The level of difficulty of the lab is medium.

Theory

The SRF04 transceiver from Devantech is used here to measure distance. It is very easy to interface, as only two signals need to be connected (as well as power and ground).

It is used by first giving it a logic high starting pulse from the AVR with a duration of >10 µs. The transceiver then transmits a 40 kHz wave, which is subsequently reflected from a surface and bounces back to the transceiver. When the AVR finds an echo, it emits a logic high pulse in its output, the duration of which (between 100 µs and 18 ms) depends on the measured distance. Figure 1 shows the SRF04 transceiver module.

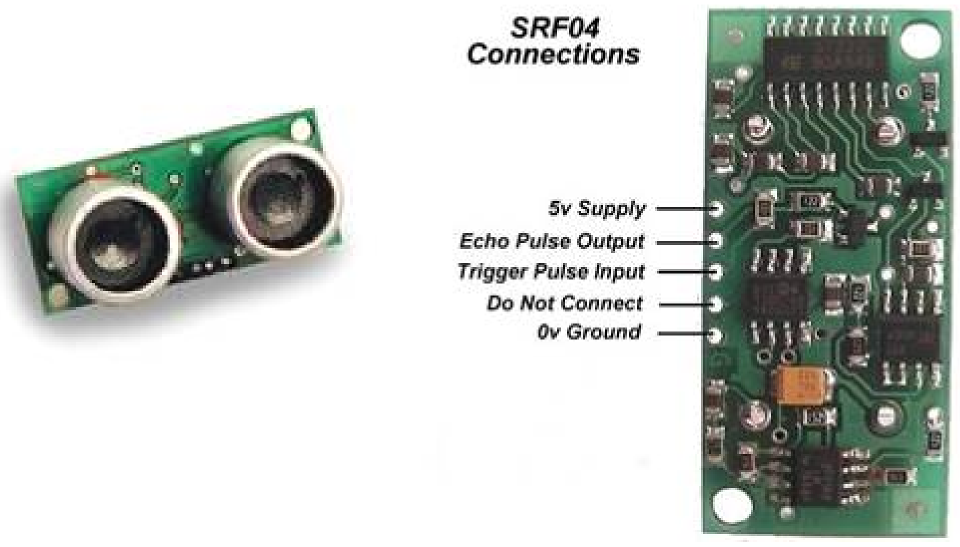

Figure 1. SRF04 transceiver module.

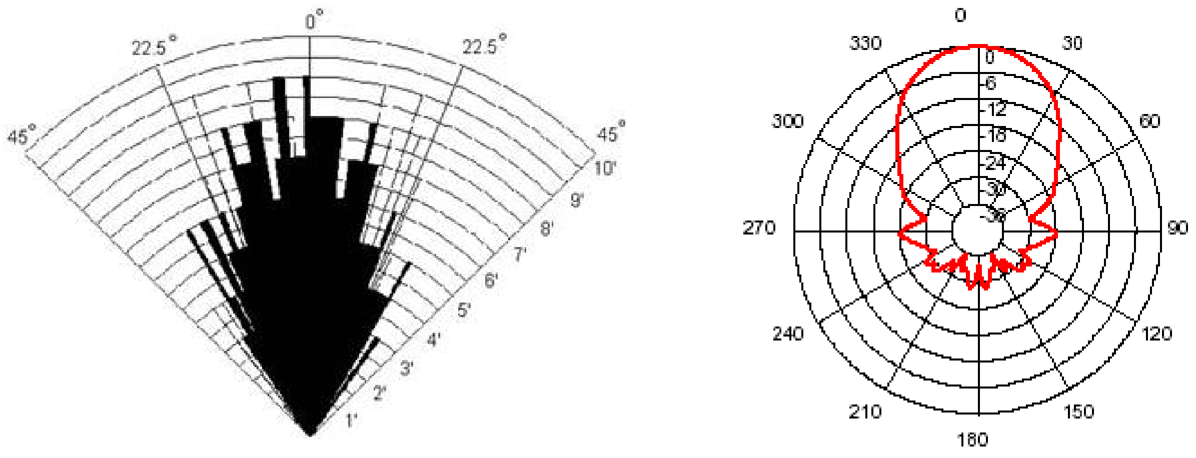

Figure 2. Range diagram and beam pattern.

Applications

Ultrasound is used in medicine to study the foetus and discover injuries. Different materials absorb ultrasound differently, which is why it is rather easy to create an image of the foetus or injury. In industry, ultrasound is used to monitor cracks and other damage in structures. Cars use ultrasound in alarm sensors and as a parking aid. We use it here in measuring the distance to objects.

Ultrasound module

The SRF04 ultrasound module used is from Devantech. A description of the module is found at www.parallax.com.

The SRF04 timing diagram is shown below. You only need to supply a short 10 μs pulse to trigger the input to start the ranging operation. The SRF04 will emit an 8-cycle burst of ultrasound at 40 kHz and raise its echo line high. It then listens for an echo, and as soon as it detects one, lowers the echo line again. The echo line is therefore a pulse the width of which is proportional to the distance to the object. By timing the pulse, it is possible to calculate the distance in inches, centimetres, or any other unit. If nothing is detected, the SRF04 will lower its echo line in any case after about 36 ms.

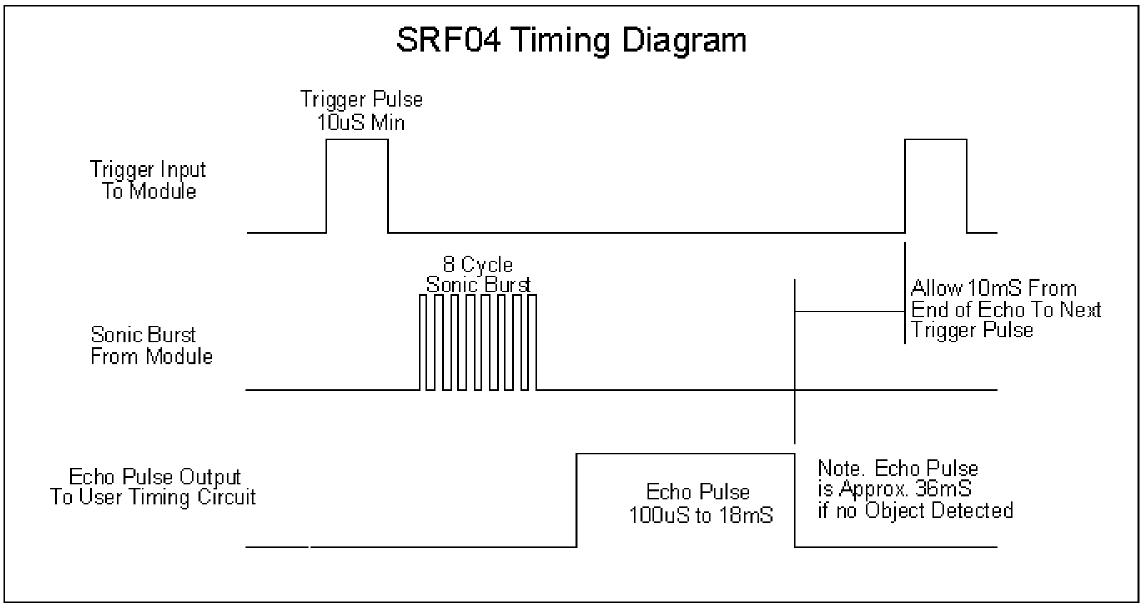
SRF04 Timing Diagram
Trigger Pulse 10uS Min
Trigger Input To Module
8 Cycle Sonic Burst
Allow 10mS From End of Echo To Next Trigger Pulse
Sonic Burst From Module
Echo Pulse Output To User Timing Circuit
Echo Pulse 100uS to 18mS
Note. Echo Pulse is Approx. 36mS if no Object Detected

Figure 3. Diagram of module timing. Source Devantech datasheet.

Calculating the Distance

The SRF04 provides an echo pulse proportional to distance to and from the object, using the formula
$C_{air} = 331.5 + (0.6 \times T_c)\ {}^{m}/_{s}$
where T_c is the actual temperature in degrees centigrade.

Exercise

Connect the SRF04, JTAGICE, and the LCD modules to the ATmega128 board according to the following schematics and image.

Materials needed

ATmega128 board
JTAGICE (USB-JTAG converter from Olimex)
LCD module
SRF04 ultrasound module

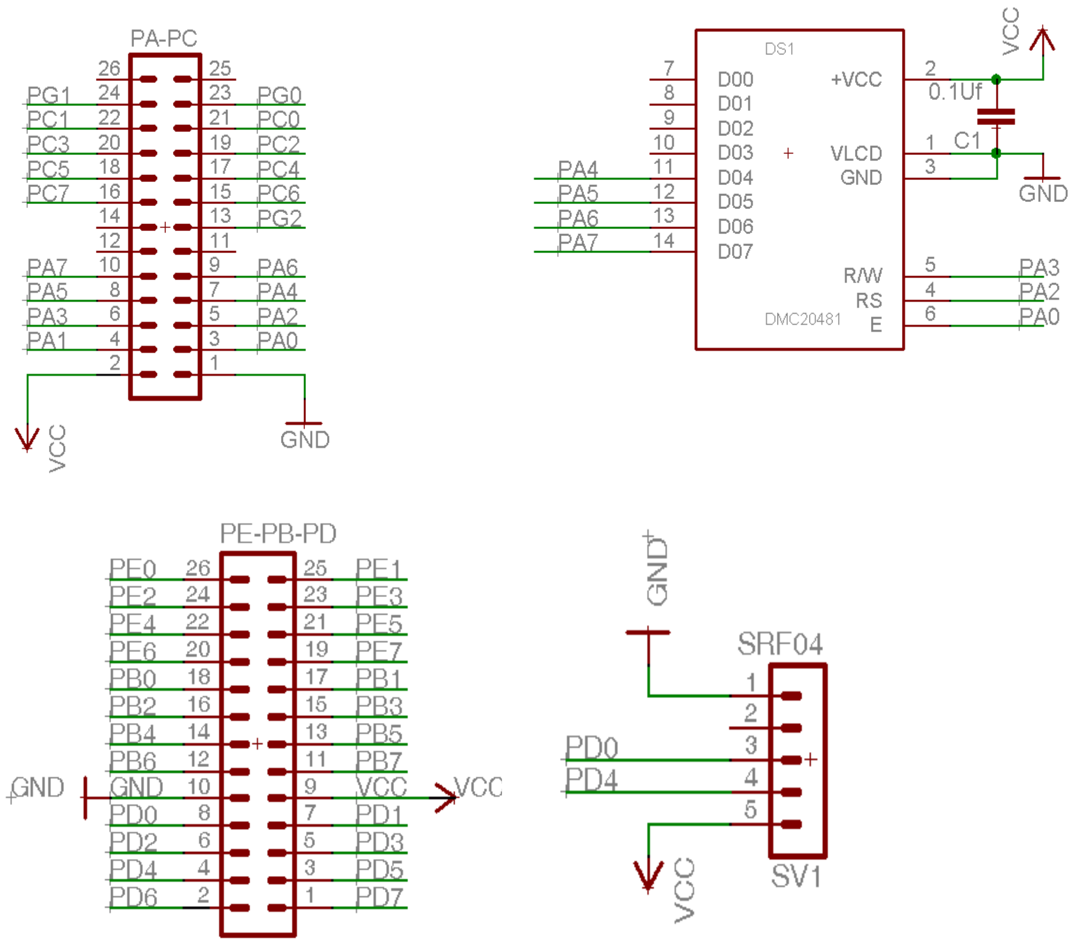

Figure 4. LCD connection (top) and SRF04 connection (bottom).

Experimental setup

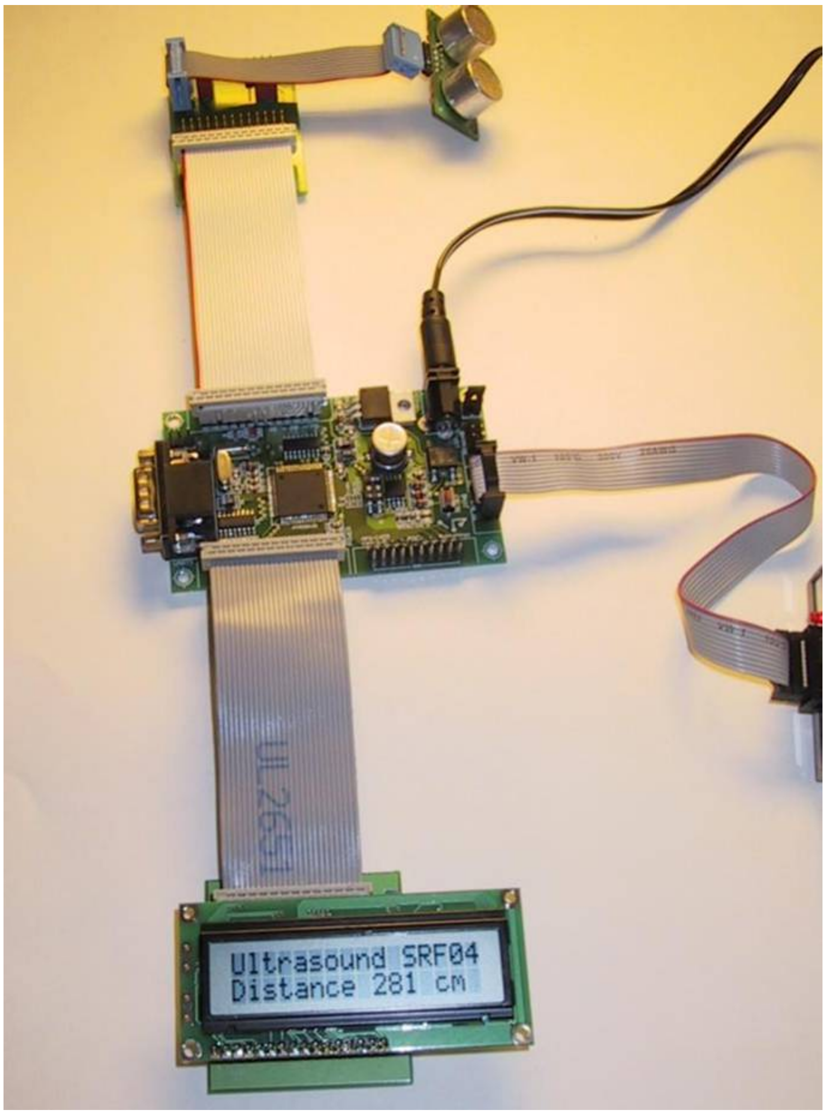

Figure 5. Experimental setup with flat cable for JTAGICE (right), SRF04 module (top), ATmega128 board (middle), and LCD (bottom).

Flowchart

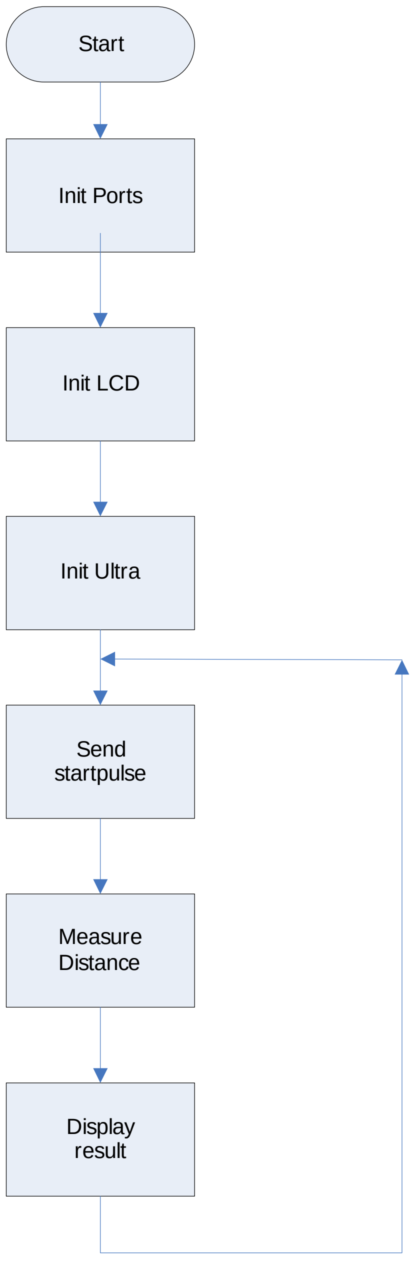

Figure 6. Flowchart of the SRF04 program.

The code

```
// This program measures distance to an object with SRF04
// ATmega128 with PORTD4 as input, PORTD0 as output
// Displays the result on PORTB LED display

// Ultrasound SRF04 with ATmega128 board
// Input  PORTD pin 4
// Output PORTD pin 0

#define F_CPU 14745600UL

#include <avr/interrupt.h>
#include <util/delay.h>

// Declarations

unsigned int rising=0, falling=0,i; // start and end time of signal
unsigned int led=0;                 // display values on LED

// Functions
ISR(TIMER1_CAPT_vect)
{
    // Check if start of pulse (rising edge) or end (falling edge)

    if ((bit_is_set(PIND, 4)))      // if high level detected
    {
        rising=ICR1;                // save start time
        TCCR1B = 0x01;              // trig next on falling edge
    }

else

    {
        falling=ICR1;               // save falling time
        TCCR1B = 0x41;              // trig rising edge next
        led = falling - rising;     // calculate the pulse width
        PORTB = ~led;               // output distance on active low LEDs
    }
}

void init_ports(void)
{
    DDRD |= 0x20;                   // b00100000 Pulse output
    DDRB  = 0xFF;                   // LEDS port output
}

void enable_interrupts(void)
{
    TIMSK = 0x20;                   // Input Capture Interrupt Enable
    sei();                          // Enable Global interrupt
}

void pulse()                        // send pulse throught PORTD pin 0
{
    PORTD|= 0x20;                   // PORTD pin 0 high
    _delay_us(15);                  // 15 mikrosec delay
    PORTD&= ~0x20;                  // PORTD pin 0 low and end of pulse
    TCCR1B = 0x41;                  // next trig rising edge
}
```

```
int main()
{
    init_ports();
    enable_interrupts();

    while(1)
    {
        pulse();                        //Send pulse to SRF04 ultrasonic sensor
        _delay_ms(50);                  // wait 50 millisec for LEDdisplay
    }
}   // end main
```

2c.1

Modify the code above to display the distance on the LEDs.
Hint: The speed of sound is $C_{air} = 331.5 + (0.6 \times T_c)\ ^m/_s$

where T_c is the actual temperature in degrees centigrade.

2c.2

Modify the program to show the distance on the LCD display.
Hint: The function "itoa(….)" is useful when displaying integer numbers. Take a look in the AVR-libc-user-manual (included with WinAVR).

2c.3

Modify exercise 1c above by taking 10 samples and using the average value.

Theoretical exercises

Questions

1. What happens when mixing the following variable types?

```
uc = unsigned char, ui=unsigned int, sc=signed char,
f=float

 uc1 = ui1 + ui2;   /* ui1=123, ui2=134 */
 uc2 = ui1 - ui2;
 sc1 = ui1 - ui2;
 ui3 = 1000 * uc1;
 uc3 = f1;          /* f1=4.53? */
 f2  = ui1 / 2;
 f3  = (float)ui1 / 2;
```

2. What is being done in the statement below? Illustrate with an example.

```
a +=++b/c--+2;
/* An example in bad C-programming */
```

3. What is actually being done in the sequence below? What is the result in the variables number and control?

```
number = 0;
control = 0;
    do
    {
        number ++;
        control += 10;
    }
    while(control < 80);
```