Pak-Austria Fachhochschule: Institute of Applied Sciences and Technology, Haripur, Pakistan

Department of IT and Computer Science

Course:

COMP-261L Computer Organization & Assembly Language Lab

Project Report:

Range Finder using Ultrasonic Sensor and 8051 Microcontroller

Submitted By:

Saad Fahim	B21F0812CS064

Submitted To:

Dr.Taimur Ahmed

Table of Contents

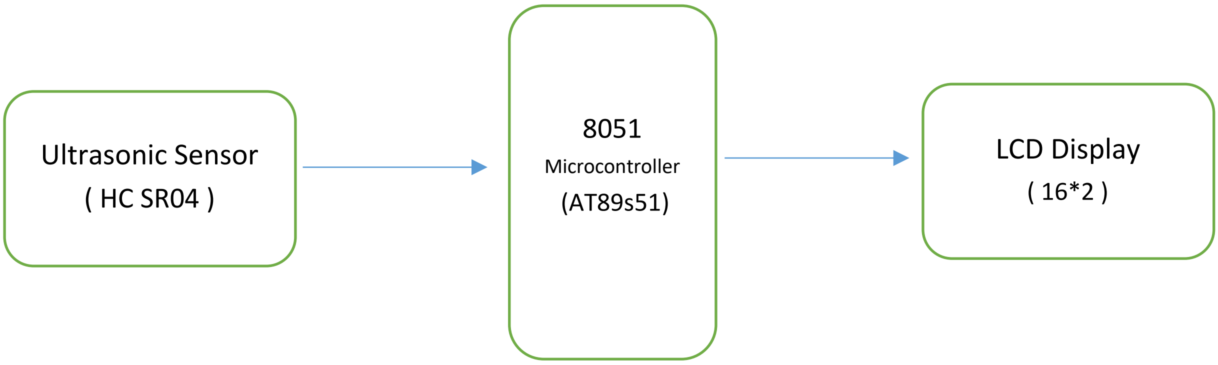
ABSTRACT

Ultrasonic Range Finder measures the distance by emitting a pulse of ultrasonic sound that travels through the air until it hits an object. When that pulse of sound hits an object, it's reflected off the object and travels back to the ultrasonic Range Finder. Ultrasonic Range finder can measure distance based on time. AT89S51 microcontroller and the ultrasonic sensor module HC-SR04 forms the basis of this circuit. The ultrasonic module sends a signal to the object, then picks up its Echo and outputs a wave forms whose time is proportional to the distance. The microcontroller accepts this signal, performs necessary processing, and displays the corresponding distance on the LCD display.

Methodology

In Range Finder, we wanted to detect the distances and display the result. Firstly, summarized the methods which we want to proceed. These are the main methods that used to build this project.

- Trigger the ultrasonic sensor using 8051.
- Calculate distance using internal timers of 8051.
- Display result on the LCD display.



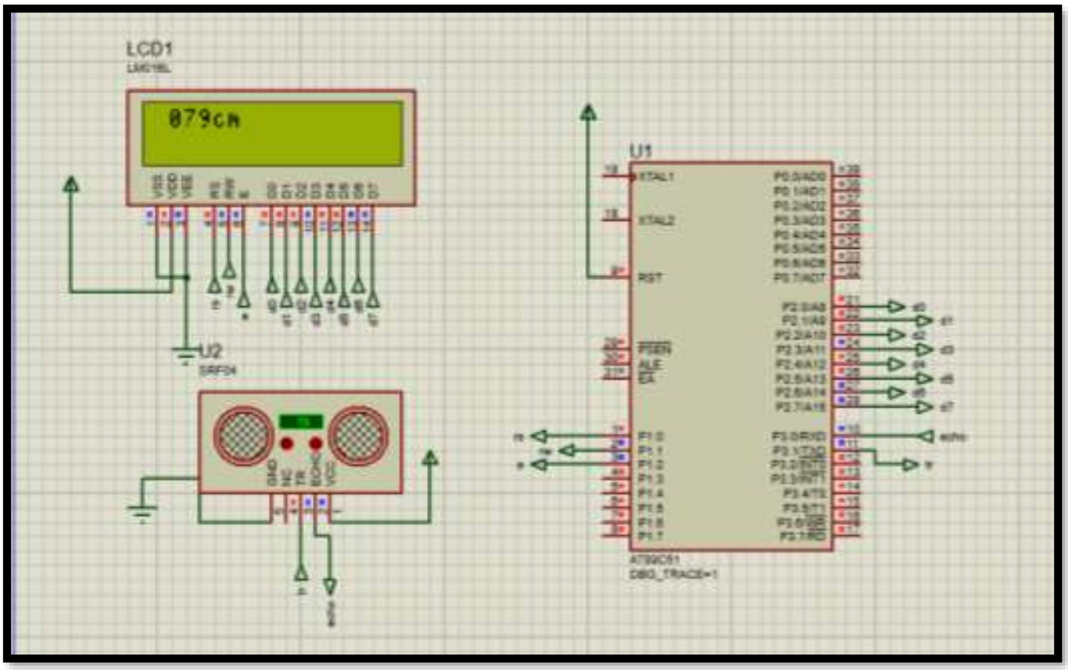
Block Diagram

The ultrasonic sensor has 2 pins (Trig, Echo) connected to the 8051. The Trig Pin is used to activate the sensor on demand by 8051, the Echo is the output of the sensor which has distance related information. Ultrasonic sensor provides 2cm - 400cm non-contact measurement function, the ranging accuracy can reach to 3mm. The modules include ultrasonic transmitters, receiver, and control circuit.

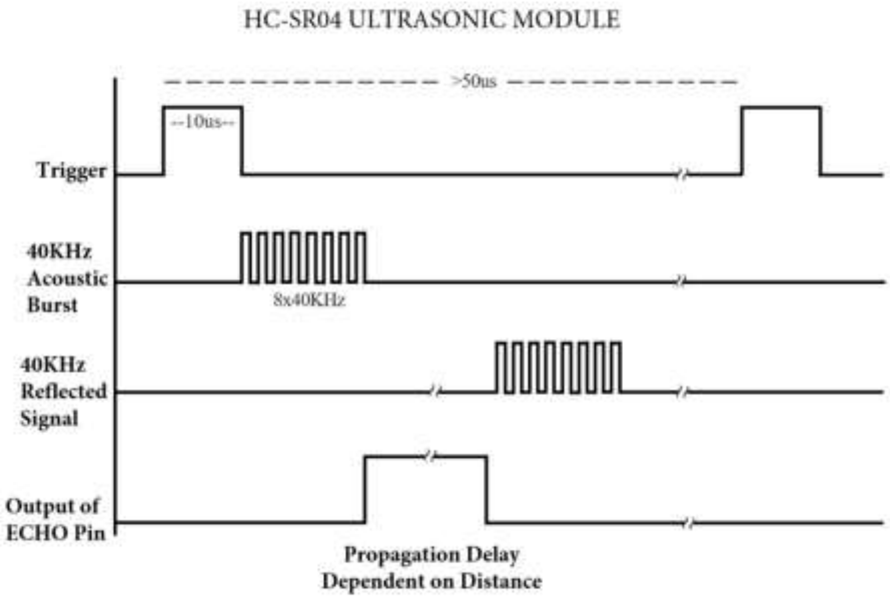
Following are the basic principle of work.

1. Using IO trigger for at least 10us high level signal
2. The Module automatically sends eight 40 kHz and detect whether there is a pulse signal back.
3. If the signal back, through high level, time of high output IO duration is the time from ending ultrasonic to returning.

Schematics



- ❖ Trigger the ultrasonic sensor using 8051.



The ultrasonic sensor needs a trigger pulse of minimum 10 µS to transmit the sound waves and to produce ECHO pulse. For the distance to be updated continuous stream of pulses are fed to trigger pin with required amount of delay. This process is repeated indefinitely in the project. The ECHO from the sensor is used to start and stop Internal TIMER 1 of 8051.

From the timing diagram it is noted that the cycle time for the sensor is 50 µS. Therefore any trigger pulse applied within 50 µS preceding a trigger will not be accounted by the sensor. It takes 58 µS for sound waves to return to sensor after reflecting from a surface 1 cm away. Assuming speed of sound to be 340 m/S time to transit 1cm will be,

$$time = (2 * distance) / \ speed$$

$$time = (\ 2 * 1\ ) \ / \ 34000$$

$$= 58.8 \ \mu S$$

The distance is considered twice since sound waves are detected after reflection from the surface. This is total transit time, to and for from the surface. The pulse width of the ECHO signal is the transit time.

❖ Display result on the LCD display.

To display output on LCD, it’s initialized as a screen with two rows and 5X7 (space to fully display a single letter or number) matrix by sending 38H as a command to it via 8051 microcontroller. Then the screen is cleared of previously stored values by sending 01H as another command. Now first address location of data to be printed on LCD is sent to it via a command i.e. 80H (points the cursor to first location in first row). In case we need to send data to first location in second row, the command to be sent is 0C0H.Then the outputs to be displayed are sent as Data, be it a string or and any other data type(manual conversion is required). Every time a new element is printed on the screen, the cursor moves to next location to print next element in the data.

Assembly Code:

```
trig  EQU  P3.1   ;  // trig pin on ultrasonic sensor
echo  EQU  P3.0   ; //echo pin on ultrasonic sensor
enable equ p1.2
rs equ p1.0
rw equ p1.1
LCD_dat equ p2
 ORG  0000
; EQUATES & VARIABLE DEFINITIONS
setb echo
clr trig
mov tmod, #02h
mov th0, #202
 acall LCD_init
 acall delay_2s
acall LCD_clear
loop1:
        ACALL get_level
        ACALL CONVERT
        acall cursr_home
                                ACALL display
        SJMP Loop1
LCD_init: mov dptr, #syntax
                                clr rs
                                clr rw
loop:       clr a
                                movc a, @a+dptr
                                jz LCD_logo
                                setb enable
                                mov LCD_dat, a
                                clr enable
                                acall delay1ms
                                inc dptr
                                sjmp loop

syntax: db 38h,0fh,01h,10h,00h
LCD_logo: mov dptr, #syntax1
                                setb rs
                                clr rw
loop4:    clr a
                                movc a, @a+dptr
                                jz new_command
                                setb enable
```

```
                mov LCD_dat, a
                clr enable
                acall delay1ms
                inc dptr
                sjmp loop4
syntax1: db ' Welcome  to ',0

new_command: mov dptr, #syntax2
                clr rs
                clr rw
loop5:    clr a
                movc a, @a+dptr
                jz LCD_logo_2
                setb enable
                mov LCD_dat, a
                clr enable
                acall delay1ms
                inc dptr
                sjmp loop5
syntax2: db 0c0h,14h,14h,14h,00h
LCD_logo_2: mov dptr, #syntax3
                setb rs
                clr rw
loop6:    clr a
                movc a, @a+dptr
                jz return
                setb enable
                mov LCD_dat, a
                clr enable
                acall delay1ms
                inc dptr
                sjmp loop6
syntax3: db "Range Finder",0
return:ret
cursr_home:
clr rs
setb enable
mov LCD_dat,#80h
clr enable
acall delay10ms
setb enable
mov LCD_dat,#0Ch
clr enable
ret
LCD_clear:
```

```
clr rs
setb enable
mov LCD_dat,#01h
clr enable
acall delay10ms
ret
get_level:
                        clr a
                        setb trig
                        acall delay_10us
                        clr trig
wait5:          jnb echo, wait5
                        setb tr0
wait6:          jnb tf0, wait6
                        inc A
                        clr tf0
                        jz return

                        jb echo, wait6
                        clr tr0
                        ret
delay_10us:
                        mov r7, #18
stay:           djnz r7, stay
                        ret
CONVERT:
    MOV    B,#10
    DIV    AB
    MOV     41,B     ; SAVE LOW(ONES) DIGIT IN 41 RAM ADDRESS
    MOV    B,#10
    DIV    AB
    MOV  42,B   ; save tenth place digit in 42 RAM ADDRESS
                MOV   43,A     ; SAVE HUNDREDTH PLACE DIGIT IN 43 RAM ADDRESS
                ACALL LOKUP
                MOV  43,A
                MOV  A,42
                ACALL LOKUP
                MOV  42,A
                MOV  A,41
                ACALL LOKUP
                MOV  41,A
                RET
LOKUP:
    CJNE A,#00H,ONE
    MOV    A,#'0'
    RET
ONE:   CJNE A,#01H,TWO
    MOV    A,#'1'
    RET
TWO:   CJNE A,#02H,THREE
    MOV    A,#'2'
    RET
THREE:  CJNE A,#03H,FOUR
    MOV   A,#'3'
    RET
FOUR:   CJNE A,#04H,FIVE
    MOV    A,#'4'
    RET
FIVE:   CJNE A,#05H,SIX
    MOV    A,#'5'
    RET
SIX:    CJNE A,#06H,SEVEN
    MOV    A,#'6'
    RET
SEVEN:  CJNE A,#07H,EIGHT
    MOV    A,#'7'
    RET
EIGHT:  CJNE A,#08H,NINE
    MOV    A,#'8'
    RET
NINE:   CJNE A,#09H,TEN
    MOV    A,#'9'
    RET
TEN:
    RET
display:   //display on LCD
    clr rw
                setb rs
                acall delay1ms

                SETB    enable
          MOV    LCD_dat,#' '
    clr enable
                acall delay1ms

                SETB    enable
          MOV    LCD_dat,43
    clr enable
```

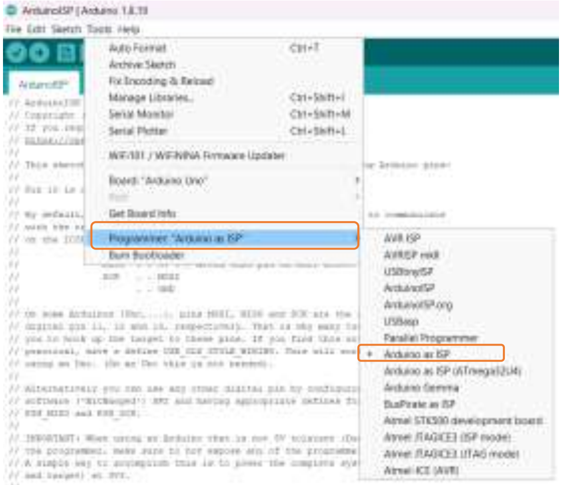
```
                acall delay1ms

                SETB   enable
        MOV    LCD_dat,42
    clr enable
                acall delay1ms

                SETB   enable
        MOV    LCD_dat,41
    clr enable
                acall delay1ms

                SETB   enable
    MOV    LCD_dat,#'c'
    clr enable
                acall delay1ms

                SETB   enable
    MOV    LCD_dat,#'m'
    clr enable
                acall delay1ms
    RET
delay10ms:      MOV   R3,#1
                  MOV   R2,#1
                MOV   R1,#19
TT1:            DJNZ  R1,TT1
                DJNZ  R2,TT1
                DJNZ  R3,TT1
                RET
delay1ms:       MOV   R2,#04
                MOV   R1,#18
TT2:            DJNZ  R1,TT2
                DJNZ  R2,TT2
                RET
delay_2s:MOV   R3,#50
                  MOV   R2,#10
                MOV   R1,#250
TT3:            DJNZ  R1,TT1
                DJNZ  R2,TT1
                DJNZ  R3,TT3
                RET

;DIST: DB "Distance:",0

   END
```

Burring HEX file into 8051 Microcontroller

After creating an assembly code we have to burn the code into 8051 Microcontroller. In this project, we used Arduino uno board to as ISP. Following steps shows how to burn HEX file into 8051 Microcontroller.

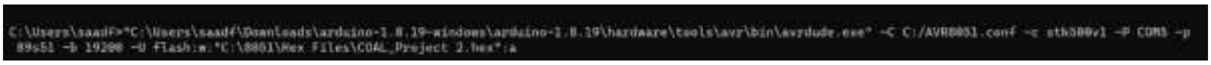
I. Download Arduino IDE and connect the circuit with Arduino uno

II.


ArduinoISP | Arduino 1.8.19
File Edit Sketch Tools Help
ArduinoISP
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 / WiFiNINA Firmware Updater
Board: "Arduino Uno"
Port
Get Board Info
Programmer: "Arduino as ISP"
Burn Bootloader
AVR ISP
AVRISP mkII
USBtinyISP
ArduinoISP
ArduinoISP.org
USBasp
Parallel Programmer
Arduino as ISP
Arduino as ISP (ATmega32U4)
Arduino Gemma
BusPirate as ISP
Atmel STK500 development board
Atmel JTAGICE3 (ISP mode)
Atmel JTAGICE3 (JTAG mode)
Atmel-ICE (AVR)

III. Open Arduino IDE and Upload the Arduino ISP code into board.

IV. Open the Command Prompt and type the AVRDUDE file location inside “ ”

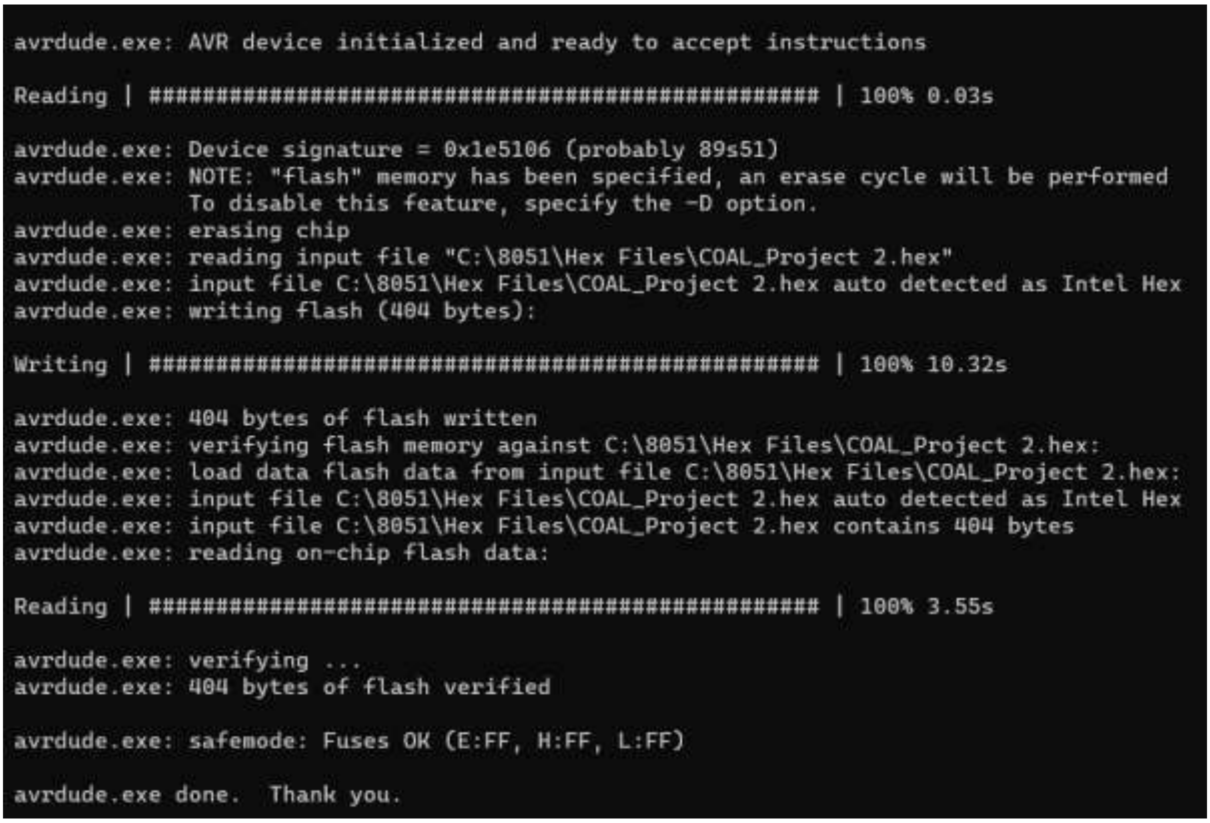


```
C:\Users\saadf>"C:\Users\saadf\Downloads\arduino-1.8.19-windows\arduino-1.8.19\hardware\tools\avr\bin\avrdude.exe" -C C:/AVR8051.conf -c stk500v1 -P COM5 -p
 89s51 -b 19200 -U flash:w:"C:\8051\Hex Files\COAL_Project 2.hex":a
```

(This is the entire code including file locations and instructions to flash)

After entering above code it will start to burning.

After completing burning it will show like below image. It shows **avrdude.exe done.**



```
avrdude.exe: AVR device initialized and ready to accept instructions

Reading | ################################################## | 100% 0.03s

avrdude.exe: Device signature = 0x1e5106 (probably 89s51)
avrdude.exe: NOTE: "flash" memory has been specified, an erase cycle will be performed
             To disable this feature, specify the -D option.
avrdude.exe: erasing chip
avrdude.exe: reading input file "C:\8051\Hex Files\COAL_Project 2.hex"
avrdude.exe: input file C:\8051\Hex Files\COAL_Project 2.hex auto detected as Intel Hex
avrdude.exe: writing flash (404 bytes):

Writing | ################################################## | 100% 10.32s

avrdude.exe: 404 bytes of flash written
avrdude.exe: verifying flash memory against C:\8051\Hex Files\COAL_Project 2.hex:
avrdude.exe: load data flash data from input file C:\8051\Hex Files\COAL_Project 2.hex:
avrdude.exe: input file C:\8051\Hex Files\COAL_Project 2.hex auto detected as Intel Hex
avrdude.exe: input file C:\8051\Hex Files\COAL_Project 2.hex contains 404 bytes
avrdude.exe: reading on-chip flash data:

Reading | ################################################## | 100% 3.55s

avrdude.exe: verifying ...
avrdude.exe: 404 bytes of flash verified

avrdude.exe: safemode: Fuses OK (E:FF, H:FF, L:FF)

avrdude.exe done.  Thank you.
```