VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

School of Electronics Engineering (SENSE)

B. Tech – Electronics & Communication Engineering

BECE403E – EMBEDDED SYSTEMS DESIGN

LAB RECORD

(lab slot L27+L28)

Submitted By

22BEC1205 – Jayakrishnan Menon

Submitted To

Dr. S.Muthulakshmi

DATE: 20/02/2025

Slot: L27+L28

Date: 20/02/2025

LAB – 07: Working with an Ultrasonic Sensor

AIM: To understand the interfacing of an ultrasonic sensor with Nucleo64-STM32L152RE Board for applications involving the measure of distances and proximity. We will also perform the following tasks:

Lab Task-1: Write a program to read distance value from HC-SR04 ultrasonic sensor module in cm and print it on the serial monitor. Implement and verify this logic on the STM32 Nucleo 64 board using Keil Studio Cloud IDE.

Lab Task-2: Write a program to design a reverse parking sensor module. This module consist of HC-SR04 ultrasonic sensor, LCD and buzzer interfaced with Nucleo. The ultrasonic sensor continuously measure the distance (in cm) between the car and obstacle, then display it on the first row of the LCD. Whenever the measured distance is lesser than 30cm generate warning signal to driver using buzzer also display a message "Obstacle !!!" on the second row of the LCD display. Implement and verify this logic on the STM32 Nucleo-64 board using Keil Studio Cloud IDE.

Lab Task-3: Write a program to design smart parking system using HC-SR04 ultrasonic sensor, servo motor, buzzer, LCD and STM32 Nucleo-64 board. (Challenging Task)

SOFTWARE REQUIRED: ARM Keil Studio (Mbed Online Compiler), Tera Term

HARDWARE REQUIRED: Micro USB cable, NUCLEO64-STM32L152 Board, Potentiometer, Jumper Wires (M-F and M-M), Breadboard, LCD, Servo Motor and an Ultrasonic Sensor (HCSR04)

PROCEDURE:

1. Go to ARM Keil Studio (https://studio.keil.arm.com) and log in
2. Select File → New → Mbed Project
3. Click the Example project drop-down list and select "mbed2-example-blinky"
4. In Project name field, provide the name of the new project and click Add project
5. Double click on the "main.cpp" file from the newly created project folder
6. Modify the code in the editor window as per the logic of your application
7. Check for any errors in the program under the "Problems" tab of the panels window
8. If no errors, connect the Nucleo Board to the computer using Micro USB Cable
9. Click Play icon (Run project) to upload and start the code execution on the board.

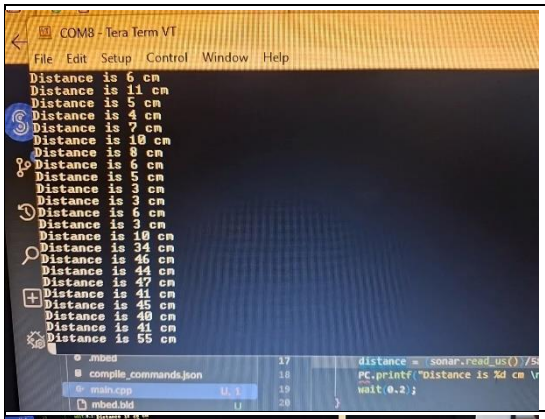
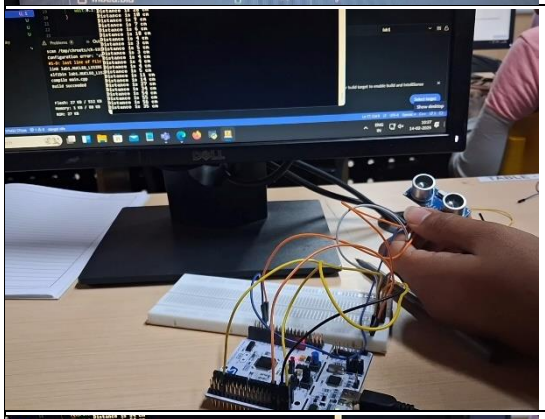
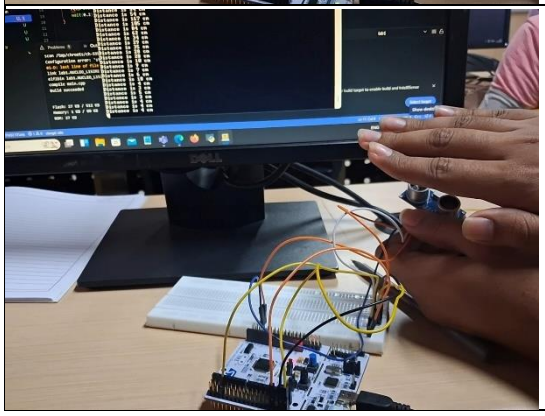
PROGRAMS:

Lab Task 1: Write a program to read distance value from HC-SR04 ultrasonic sensor module in cm and print it on the serial monitor. Implement and verify this logic on the STM32 Nucleo 64 board using Keil Studio Cloud IDE.

Code:

```
#include "mbed.h"
Serial PC(USBTX, USBRX);
DigitalOut trigger(PC_8);
DigitalIn echo(PC_6);
int distance = 0;
Timer sonar;
int main(){
    while(1) {
        trigger = 1;
        sonar.reset();
        wait_us(10.0);
        trigger = 0;
        while (echo == 0);
        sonar.start();
        while (echo == 1);
        sonar.stop();
        distance = (sonar.read_us()) / 58.0;
        PC.printf(" Distance is %d cm \n\r", distance);
        wait(0.2);
    }
}
```

Output:

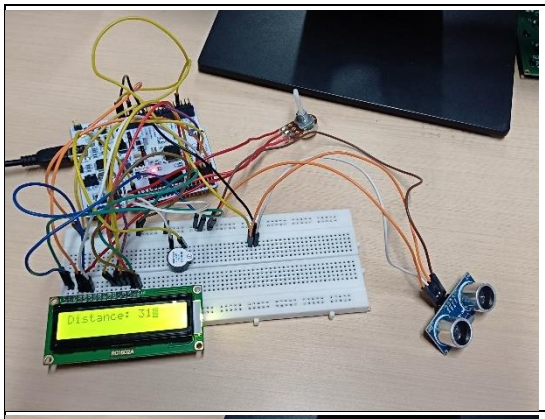
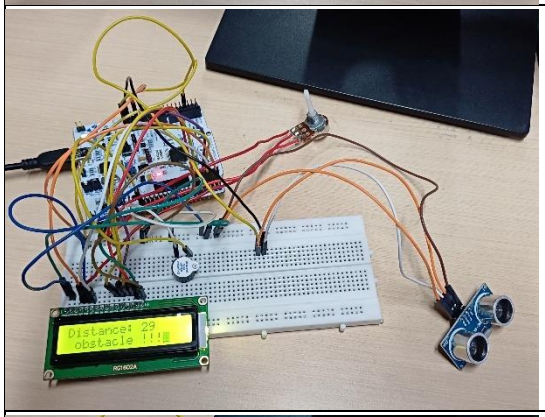
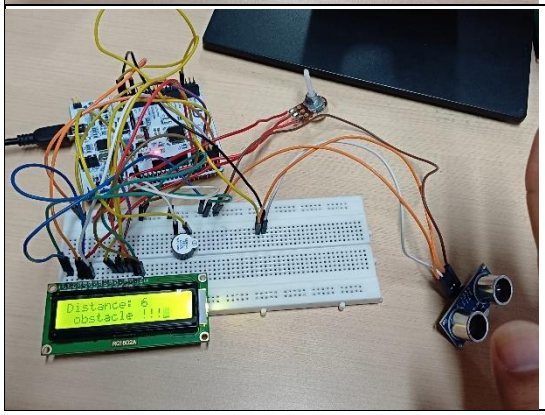
	**Fig. 1.1:** Distances are printed on the Serial Window within intervals of approximately 0.2 seconds.
	**Fig. 1.2:** When the obstacle is far away, larger distance values are printed on the Serial Window.
	**Fig. 1.3:** When the obstacle is near, smaller distance values are printed on the Serial Window.

Lab Task 2: Write a program to design a reverse parking sensor module. This module consist of HC-SR04 ultrasonic sensor, LCD and buzzer interfaced with Nucleo. The ultrasonic sensor continuously measure the distance (in cm) between the car and obstacle, then display it on the first row of the LCD. Whenever the measured distance is lesser than 30cm generate warning signal to driver using buzzer also display a message "Obstacle !!!" on the second row of the LCD display. Implement and verify this logic on the STM32 Nucleo-64 board using Keil Studio Cloud IDE.

Code:

```
#include "mbed.h"
#include "TextLCD.h"
TextLCD lcd(PC_0,PC_1,PB_0,PA_4,PA_1,PA_0); // rs,e,d4-d7
Digitalout trigger(PC_8);
DigitalIn echo(PC_6);
DigitalOut buzzer(PC_10);
int distance = 0;
Timer sonar;
int main(){
    lcd.cls();
    while(1) {
        trigger = 1;
        sonar.reset();
        wait_us(10.0);
        trigger = 0;
        while (echo == 0);
        sonar. start();
        while (echo == 1);
        sonar.stop();
        distance = (sonar.read_us())/58.0;
        lcd.locate(0,0);
        lcd.printf("Distance: ",distance);
        wait(0.2);
        if (distance<30){
            buzzer=1;
            lcd.locate(1,1);
            lcd.printf("obstacle !!!");
        }
        else{
            buzzer=0;
        }
    }
}
```

Output:

	**Fig. 2.1:** Distance from the Ultrasonic distance sensor is greater than 30 cm. This value is displayed on the LCD.
	**Fig. 2.2:** Distance from the Ultrasonic distance sensor is slightly lesser than 30 cm. The obstacle is 29 cm away from the sensor. This value is displayed on the LCD. Since the proximity is less than 30 cm. a warning message, "obstacle !!!" is displayed. The buzzer also rings indicating the same.
	**Fig. 2.3:** Distance from the Ultrasonic distance sensor is slightly lesser than 30 cm. The obstacle is 6 cm away from the sensor. This value is displayed on the LCD. Since the proximity is less than 30 cm. a warning message, "obstacle !!!" is displayed. The buzzer also rings indicating the same.

Lab Task 3: Write a program to design smart parking system using HC-SR04 ultrasonic sensor, servo motor, buzzer, LCD and STM32 Nucleo-64 board.

- The ultrasonic sensor module place near the gate entrance continuously check for the incoming vehicles. The LCD display "Smart Parking" on the first row and "Avail. slot: XY" in second row of the display.
- When a vehicle comes closer to the ultrasonic sensor detection area and parking slot is available then the system open a gate barrier to 90° (close after 10 seconds) to allow the vehicle enter into the parking slot and decrement parking slot by 1.
- If no parking slot available then display a message "No Parking slot" on LCD (2nd line) and switch on the buzzer (for 5 Seconds). Have a similar system on the exit and increment the free slot by 1 for every vehicle leaves the parking slot.

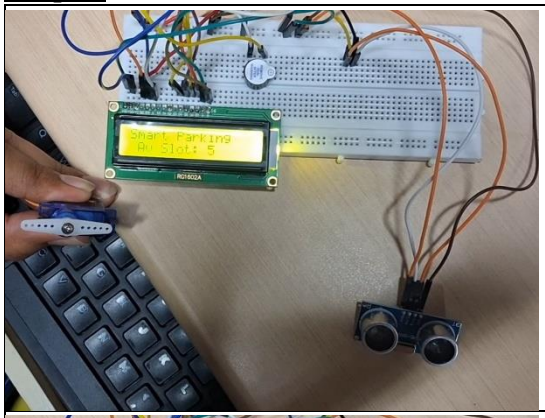
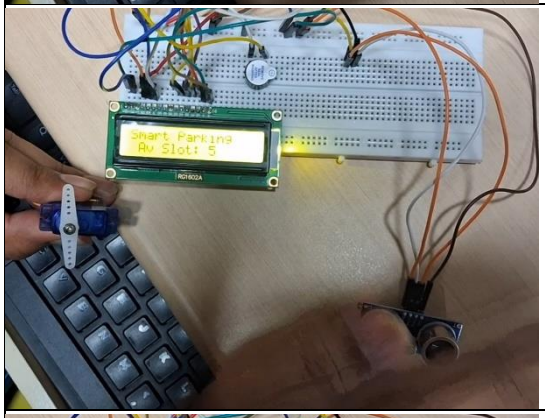
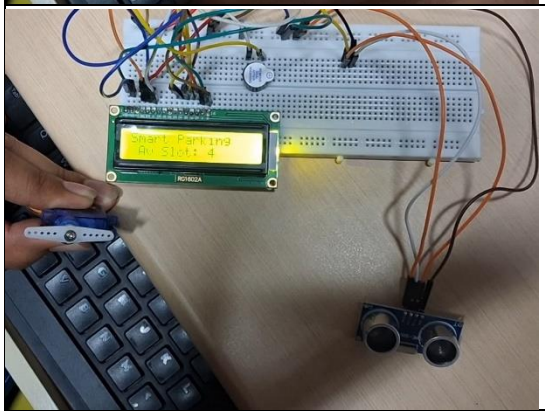
Simulate and verify this logic on Nucleo using Tinkercad circuits simulator. Note: XY is number of available slot and initially assume total available parking slot is 5.

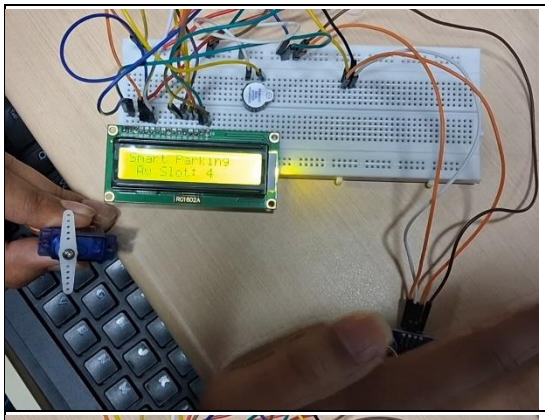
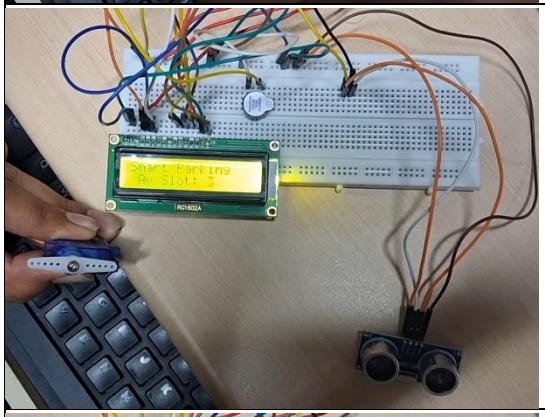
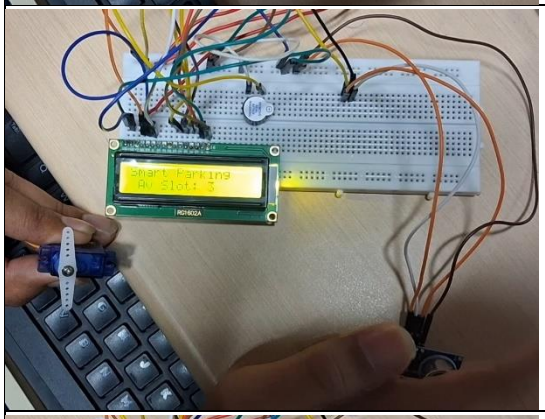
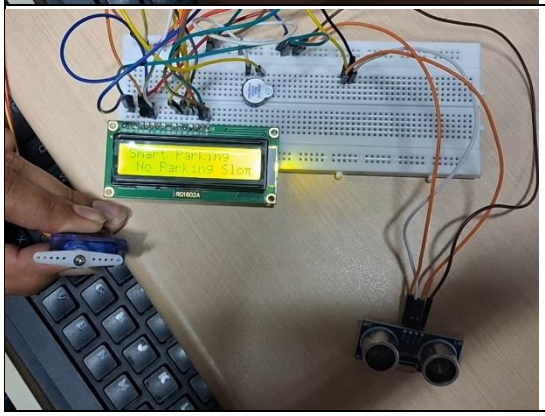
Code:

```
#include "mbed.h"
#include "TextLCD.h"
PwmOut PW(PC_9);
TextLCD lcd(PC_0,PC_1,PB_0,PA_4,PA_1,PA_0);
DigitalOut trigger(PC_8);
DigitalIn echo(PC_6);
DigitalOut buzzer(PC_10);
int distance =0;
Timer sonar;
int count = 5;
int main(){
    lcd.cls();
    while(1){
        lcd.locate(0,0);
        lcd.printf("Smart Parking");
        lcd.locate(1,1);
        lcd.printf("Av Slot: %d",count);
        wait(0.2);
        trigger = 1;
        sonar.reset();
        wait_us(10.0);
        trigger =0;
        while(echo==0);
        sonar.start();
        while(echo==1);
        sonar.stop();
        distance=(sonar.read_us())/58.0;
        if(distance<10 && count<=5){
            PW.pulsewidth_us(1500);
            count = count-1;
            wait(1.0);
            PW.pulsewidth_us(500);
            if(count==-1)count=0;
        }
```

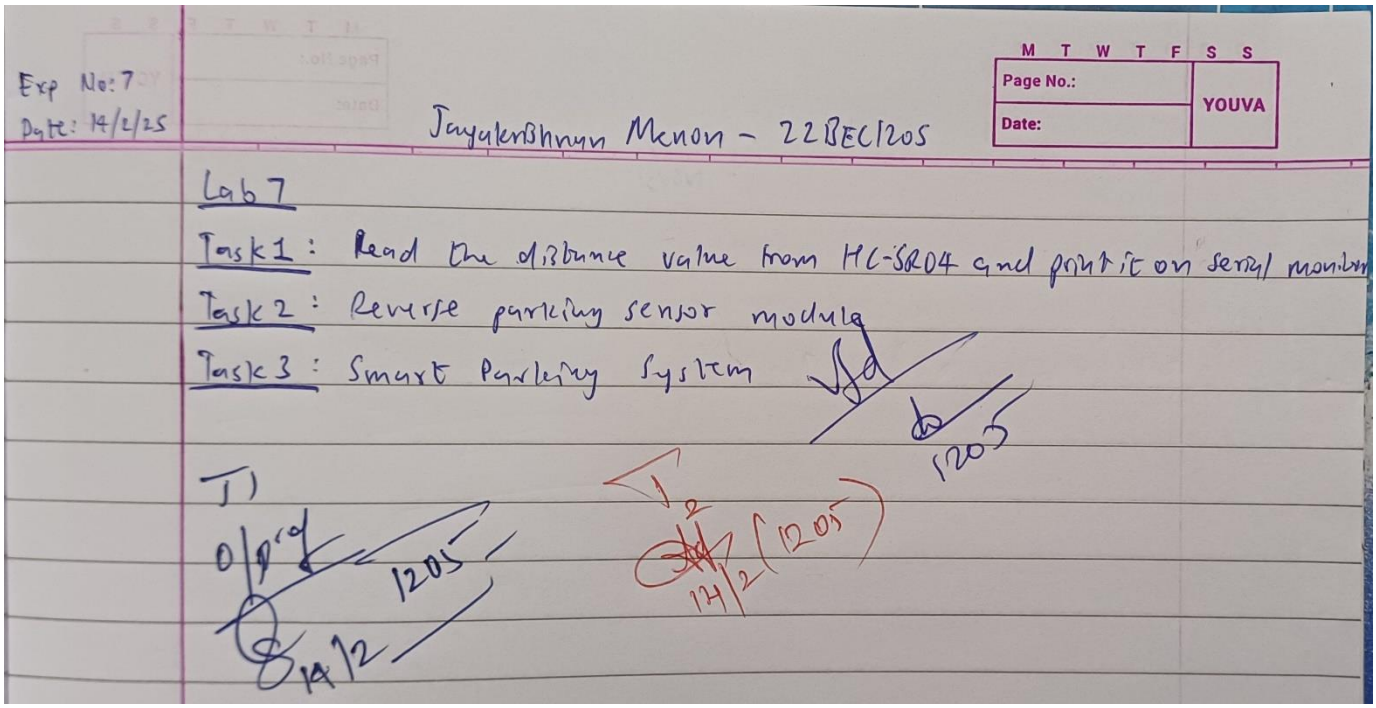
```
        else if(count==0){
            buzzer=1;
            lcd.locate(1,1);
            lcd.printf("No Parking Slot");
            wait(1.0);
        }
        wait(1.0);
        lcd.cls();
    }
}
```

Output:

	**Fig. 3.1:** Initially, the number of available Parking slots is 5 and there are no cars occupying the slots.
	**Fig. 3.2:** The gate (Servo Motor) opens, when a car is detected and then closed after a few seconds.
	**Fig. 3.3:** After the 1st car occupies a slot, there will only be 4 more slots left. The number of slots is updated on the LCD.

	**Fig. 3.4:** The gate (Servo Motor) opens, when a car is detected and then closed after a few seconds.
	**Fig. 3.5:** After the 2nd car occupies a slot, there will only be 3 more slots left. The number of slots is updated on the LCD.
	**Fig. 3.6:** The gate (Servo Motor) opens, when a car is detected and then closed after a few seconds.
	**Fig. 3.7:** After the 5th car occupies the final slot, slots won't be available anymore. At this point, whenever a car tries to enter, the LCD displays, "No Parking Slot". A buzzer also sounds to indicate the same.

OUTPUT VERIFICATION:



INFERENCE:

1. The Timer API in mbed allows users to measure time intervals with microsecond precision. It provides functions to start, stop, reset, and read the elapsed time. This is useful for performance measurement, timing control, and event scheduling in embedded applications.
2. The start() function initiates the timer, allowing it to begin counting time from its current state. If the timer was previously stopped, it resumes from the last recorded value rather than resetting.
3. The stop() function halts the timer without resetting its value. This is useful when measuring time intervals, as it allows reading the elapsed time without affecting future measurements.
4. The reset() function resets the timer to zero without stopping or starting it. If the timer is running, it continues counting from zero; otherwise, it remains at zero until started.
5. The read_us() function returns the elapsed time in microseconds since the timer was started. It is useful for obtaining precise time measurements in performance monitoring and delay implementations.
6. Using the “DigitalIn” API, we can read the value of a digital pin. And by using the “DigitalOut” API, we can write logical high or low values to a digital pin
7. The PwmOut API in mbed allows you to generate Pulse Width Modulated (PWM) signals on pins that support PWM functionality. This is particularly useful for controlling devices such as LEDs, Buzzers, and servos. Its Syntax is: “PwmOut Identifier(PinName)”.

8. The PwmOut API provides two methods, period and period_ms, to configure the period of the Pulse Width Modulation (PWM) signal. These methods are critical for setting the frequency of the PWM signal.
9. The write method in the PwmOut API sets the duty cycle of the PWM signal. The duty cycle determines the proportion of the PWM period for which the signal is HIGH (ON) versus LOW (OFF). This is essential for controlling the intensity, speed, or position of devices like LEDs, motors, and servos.
10. The TextLCD library allows a Nucleo board to control Liquid Crystal Displays (LCDs) based on the Hitachi HD44780 chipset in 4-bit mode. It can be used by including the header file, "TextLCD.h". The mbed TextLCD library performs the laborious LCD setup routines and also tells the LCD object which pins are used for which functions.
11. The Syntax for using this API is: “TextLCD lcd(rs, enable, d4, d5, d6, d7)”.
12. It also provides 4 main useful functions. The cls() function clears the screen and locates the cursor to 0,0. The locate() function can be used to move the cursor to any of the 16x2 positions available on the LCD. The putc() function writes a character to the LCD. The printf() function writes a formatted string to the LCD.
13. Programmable Delays can be implemented through the use of the “wait()” function.
14. A program can be set to run indefinitely using the “while(1)” loop.

RESULT:

Thus, the interfacing of an ultrasonic sensor with Nucleo64-STM32L152RE Board for applications involving the measure of distances and proximity was understood and the tasks were also performed successfully.