# Introduction to IOT Lab-3

# Spring 2025

**Overview**

In this Lab session you will learn to use the HC-SR04 Ultrasonic Sensor to detect the distance between an object and the sensor with the help of the ESP32 Dev Module. You will also learn to use the Bluetooth Classic wireless protocol to get the Ultrasonic Sensor readings on your Android Device.

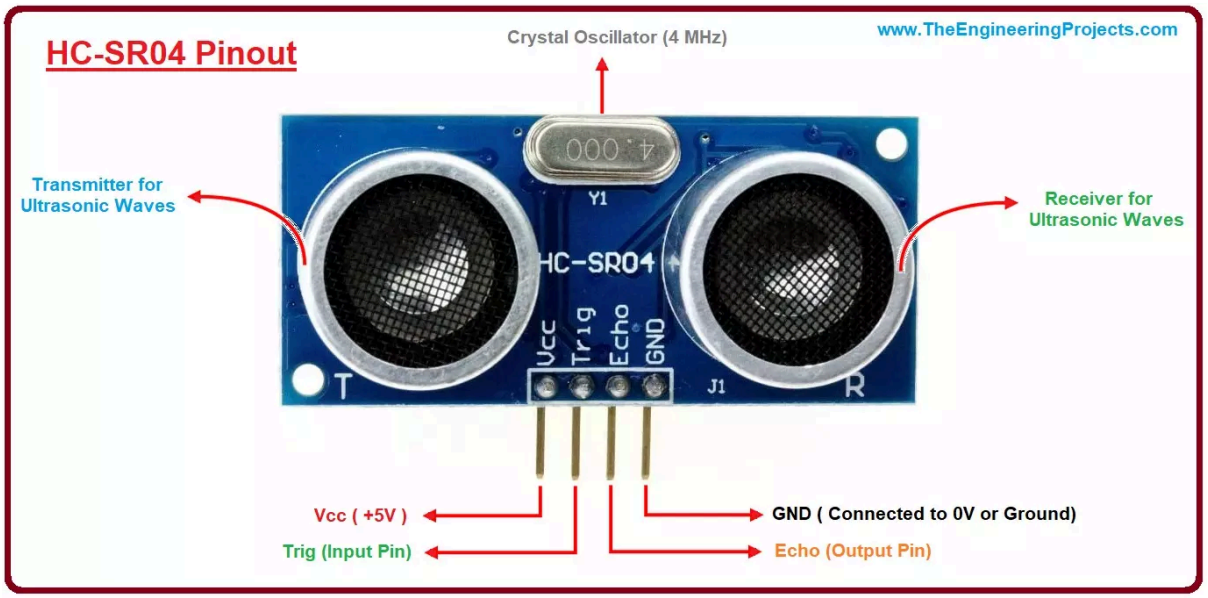
**HC-SR04 Ultrasonic Sensor**

The HC-SR04 Ultrasonic Distance Sensor is a sensor used for detecting the distance to an object using sonar. It's ideal for any robotics projects you have which require you to avoid objects, by detecting how close they are you can steer away from them!

The HC-SR04 uses non-contact ultrasound sonar to measure the distance to an object, and consists of two ultrasonic transmitters (basically speakers), a receiver, and a control circuit. The transmitters emit a high frequency ultrasonic sound, which bounce off any nearby solid objects, and the receiver listens for any return echo. That echo is then processed by the control circuit to calculate the time difference between the signal being transmitted and received. This time can subsequently be used, along with some clever math, to calculate the distance between the sensor and the reflecting objects.

**Specifications**

- Input Voltage: 5V
- Current Draw: 20mA (Max)
- Digital Output: 5V
- Digital Output: 0V (Low)
- Working Temperature: -15°C to 70°C
- Sensing Angle: 30° Cone
- Angle of Effect: 15° Cone
- Ultrasonic Frequency: 40kHz
- Range: 2cm - 400cm




## Pinout

- 5V Vcc
- Trigger Pulse Input
- Echo Pulse Output
- Ground

## Getting Started with Bluetooth Classic

Bluetooth is a great wireless communication technology that has been popular for quite a few years. Operating in the unlicensed 2.4 GHz ISM (Industrial, Scientific and Medical) frequency band, Bluetooth is a short□range wireless communication technology with range up to 100 m.
ESP32 SoC integrates both the Bluetooth Link Controller (or Link Manager) and the Baseband into its silicon. Physically, only an external antenna is needed for proper Bluetooth Communication.

Since both Wi-Fi and Bluetooth operate at the same 2.4 GHz ISM frequency, the Wi-Fi Radio and the Bluetooth Radio share the same antenna in ESP32. If you take a look at the pinout of ESP32 SoC, there is only one pin for connecting to the antenna (LNA_IN).
ESP32 supports both the Classic Bluetooth (Classic BT) and Bluetooth Low Energy (BLE) which can be configured with BLUEDROID Bluetooth Stack.
ESP32 Bluetooth supports three types of Host Controller Interface (HCI): UART, SPI and VHCI (Virtual HCI) interfaces (only one can be used at a time and UART is the default).

### Classic Bluetooth

The Classic Bluetooth also known as Bluetooth Base Rate / Enhanced Data Rate, is the original point-to-point network topology designed for one-to-one wireless communication between a

master and a slave. Even though multiple slave devices can be connected to a single master, only one slave can be actively communicating with the master. Our Bluetooth keyboards and mouse work with Classic Bluetooth technology. Another simple example is file transfer between two devices (like two mobile phones or a laptop and a mobile phone) over Bluetooth is based on Classic Bluetooth functionality.

You can have a closer look at the various Bluetooth Profiles and Protocols here

The communication between ESP32's Processor and Bluetooth Controller is based on Serial Interface. Let us explore more about ESP32 Bluetooth by using the 'BluetoothSerial' library for Classic Bluetooth.
The BluetoothSerial library works similar to the Serial library but it is just within ESP32. Some of the frequently used functions offered by BluetoothSerial library are:

- begin()
- available()
- write()
- read()

Use BluetoothSerial library to instantiate an object of class BluetoothSerial.

```
#include "BluetoothSerial.h"  // Include the Bluetooth library for ESP32

// Initialize Bluetooth
BluetoothSerial SerialBT;
```

Use the begin() function while passing the name which you want to give the device to initialise the setup.

```
SerialBT.begin("ESP32 Arduino");
```

The write() and read() functions can be used as follows.

```
SerialBT.write("Hi!")
char command = SerialBT.read();
```

SerialBT.available() works just like the Serial.available() function it checks for characters available on the serial monitor, in case of SerialBT.available it checks characters in the Bluetooth serial monitor.

```
SerialBT.available()
```

Sample Source codes and more documentation can be found here.

To view the received data of ESP32, we will print the data on the serial port. Coming to the mobile phone, in order to send and receive data over Bluetooth, we have to use an application.

**Installation**
Install “Serial Bluetooth Terminal” from the Google Play Store.
Serial Bluetooth terminal app will help you communicate to your esp and also will help you view the transmitted data from the ESP32.

# Experiments

**Experiment-1 : Use the HC-SR04 Ultrasonic sensor to get the distance value of the object kept in front and display the readings on the serial monitor.**

Required hardware:-
- Breadboard
- ESP32 Dev Module
- HC-SR04 sensor
- Jumper Cables

**Pseudocode**
- Define the trigger and echo pins
- Define the speed of sound and your conversion measurements
- Start the serial communication in the setup block
- Define trigger pin as output and echo pin as input.
- In the loop block set trigger pin high for 10 microseconds and set it back to low.
- Use pulseIn() function to measure the duration of sound wave travel.

**Required Output**
Show the reading of the Ultrasonic sensor on the serial monitor and try to use at least 2 different units of measurement for this given experiment.

**Experiment-2A : Use the bluetooth functionality of the ESP32 Dev module to communicate with the Serial Bluetooth Terminal on the Phone.**

Required Hardware:-

- Breadboard
- ESP32 Dev Module
- HC-SR04 sensor
- Jumper Cables

**Pseudocode**

- Follow the same steps as experiment 1 to obtain the distance values
- Write this sensor data to the Bluetooth serial monitor on the phone using SerialBT.print()

**Required Output**

The sensor data should be updated every few seconds on the Bluetooth serial monitor on the phone.

**Experiment-2B : Now that you have established a connection from the ESP board to the Bluetooth Serial Monitor on the phone, create a proximity notifier using the same apparatus. Enter the proximity threshold on the serial monitor of the phone, If the sensor distance reading is less than that of the threshold, light an LED notifying the user of a proximity alert.**

Required Hardware:-

- Breadboard
- ESP32 Dev Module
- HC-SR04 sensor
- Jumper Cables
- LED

**Pseudocode**

- In continuation of 2A now read the value sent from the Bluetooth serial monitor on the phone using SerialBT.available() check the availability of any character.
- Read the character, and since reading the character yields in it's ASCII code look for ways to parse the set of characters.
- Once you've read the set of characters, implement your use case using the now obtained threshold.
- Give out an indication to the user on the phone about a proximity object using the SerialBT.write() function.
- Light the LED in case of an object being in proximity.

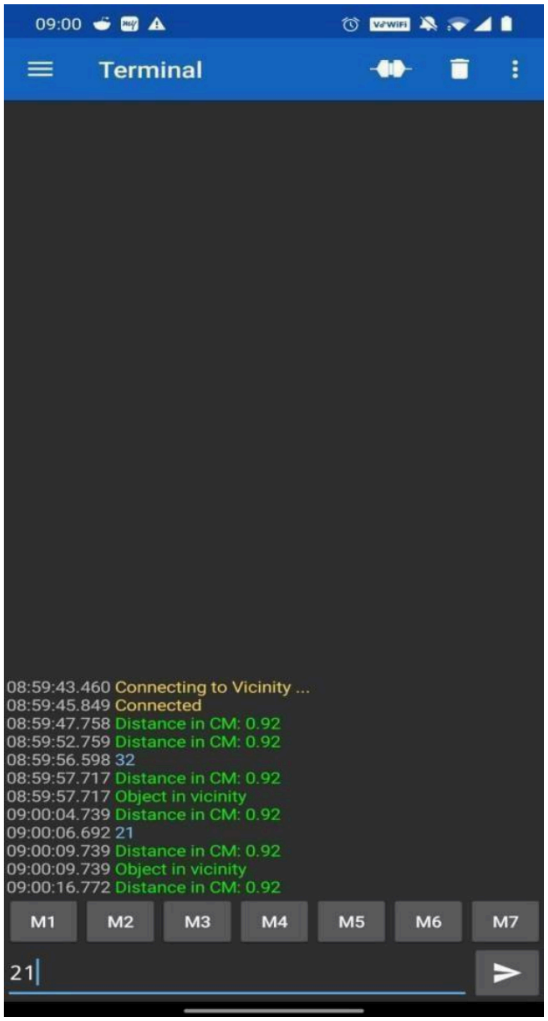
**Required Output**

Users should get the sensor readings on the Serial Bluetooth Terminal and

once the user enters a threshold he should be notified if any object lies within that threshold distance. In case if any object lies within the threshold value, light the LED to notify the user.

**Output of the serial monitor on phone**

The lines in blue represent the input of the phone and lines in green represent the transmitted information from the ESP32 dev module.



**Experiment-3 : Use the HC-SR04 Ultrasonic sensor to get the speed of the object in front of it upon command given through the serial monitor on the phone.**

**Required hardware**

- Breadboard
- ESP32 Dev Module
- HC-SR04 sensor
- Jumper Cables

**Guidelines**

- Continuing to use the same setup as before, write code to look for a specific character in the serial monitor from the phone.
- Use this special character as a trigger to start measuring speed of the object.
- Measurement of speed can be achieved in any way you deem fit (try to keep it simple).
- Output the speed value back to the serial monitor on the phone.

**Required Output**

Display the speed value calculated using the ultrasonic sensor upon given a command from the serial monitor on the phone.