VIT Chennai

FOSSEE INTERNSHIP

Interfacing Ultrasonic Sensor with Arduino -Python

By

Sam Prince Franklin K

Submitted To

FOSSEE CLUB – VIT, Chennai

Under The Guidance Of

Dr. Chanthini Baskar

Abstract

The goal of the project is to create a distance measurement system that uses ultrasonic waves and is interfaced with an Arduino. The human hearing range is 20hz to 20khz, as we know. We may use the ultrasonic sensor HC-SR04 to use this frequency range waves. When this sensor is connected to an Arduino, which is a control and sensing device, a precise distance measurement may be achieved using innovative approaches. As large amounts are spent for hundreds of inflexible circuit boards, the Arduino will allow business to bring many more unique devices. This distance measurement system can be widely used as range meters and as proximity detectors in industries. The ultrasonic sensor's hardware is connected to an Arduino. This method of measurement is a quick and accurate approach to measure short distances. An ultrasonic sensor is used to determine the distance between an obstruction and the sensor. The distance may be determined after the sound speed is known. The project can be further continued by interfacing Ultrasonic Sensor with LED Display or 7 Segment display and getting outputs according to the distance. Further research can be made to make a human detection system for a house surveillance system.

Contents

Interfacing Ultrasonic Sensor with Arduino - Python

0.1 Ultrasonic Sensor:

The HC-SR04 ultrasonic sensor uses SONAR (Sound Navigation and Ranging) to determine the distance of an object just like the bats do. It offers excellent non-contact range detection with high accuracy and stable readings in an easy-to-use package from 2 cm to 400 cm or 1” to 13 feet. The operation is not affected by sunlight or black material, although acoustically, soft materials like cloth can be difficult to detect. It comes complete with ultrasonic transmitter and receiver module.

0.2 Technical Specifications:

- Power Supply +5V DC
- Quiescent Current <2mA
- Working Current 15mA
- Effectual Angle <15
- Ranging Distance 2cm – 400 cm/1 – 13ft
- Resolution 0.3 cm
- Measuring Angle 30 degree

0.3 Interfacing Ultrasonic Senor With Arduino:

Components Required:

- 1 Arduino Uno R3
- 1 ULTRASONIC Sensor (HC-SR04)
- 4 x Jumper Wires

0.4 Circuit Diagram

Here is a illustration of the circuit which has to make for this experiment. This is simulated using TINKERCAD , an online platform to simulate circuits.

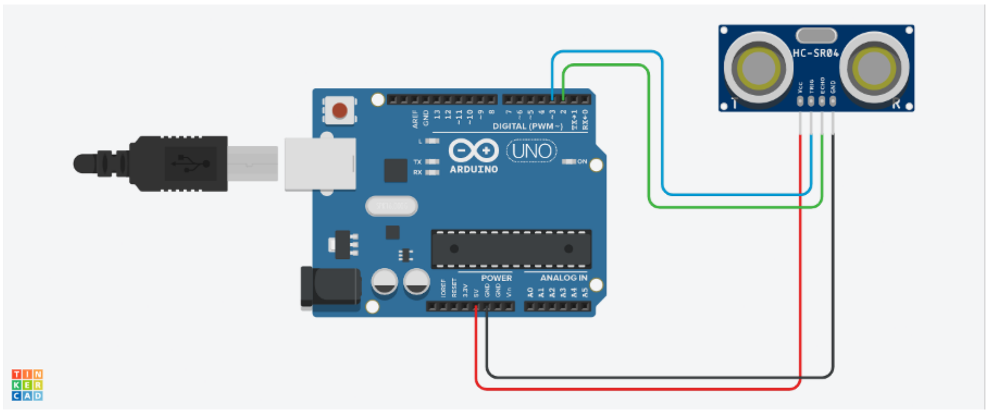
1.png 

Figure 1: *Circuit Diagram*

0.5 Arduino Code:

Listing 1 – Arduino Code

```
#define echoPin 2
#define trigPin 3
long duration;
int distance;
void setup(){
Serial.begin(9600);
pinMode(trigPin,OUTPUT);
pinMode(echoPin,INPUT);
}
void loop(){
digitalWrite(trigPin,LOW);
delayMicroseconds(2);
digitalWrite(trigPin,HIGH);
delayMicroseconds(10);
digitalWrite(trigPin,LOW);

duration=pulseIn(echoPin,HIGH);
distance=(duration*0.034/2);
Serial.print("Distance : ");
Serial.print(distance);
Serial.println(" cm ");
delay(1000);
}
```

0.6 Code Explanation:

Step 1: Open the Arduino IDE and we need define two pins, echoPin on digital pin 2 and trigPin on digital pin 3. by using the keyword "define'. Next declare two variables, one is "duration". This is for store the duration of sound wave travelled. Other is "distance" for store the distance calculated.

Listing 2 – Arduino Code

```
#define echoPin 2
        #define trigPin 3
        long duration;
        int distance;
```

Step 2: In the void setup() function we need to begin the serial communication with baurd rate as 9600. It is done by the keyword "Serial.begin(9600)". Then set the trigPin as "OUTPUT", by the keyword "pinMode(trigPin, OUTPUT)". Because the trigPin is the input pin of transmitter of sensor module. Now we need to set the echoPin as "INPUT". By the keyword "pinMode(echoPin, INPUT)".

Listing 3 – Arduino Code

```
void setup(){
                        Serial.begin(9600);
                        pinMode(trigPin,OUTPUT);
                        pinMode(echoPin,INPUT);
                }
```

Step 3: Now the trigPin state is in a float condition. We need to set it as "LOW". for this purpose we use the keyword "digitalWrite(trigPin, LOW)". Then hold this state for 2 microseconds by the keyword "delayMicroseconds(2)".

Listing 4 – Arduino Code

```
    digitalWrite(trigPin,LOW);
        delayMicroseconds(2);
```

Now we need to set the trigPin "HIGH" for 10 seconds, with the same keyword mentioned above. Only change the parameter.Then set the trigPin as "LOW" state.

Listing 5 – Arduino Code

```
digitalWrite(trigpin,HIGH);
delayMicroseconds(10);

digitalWrite(trigpin,LOW);
```

Now read the echoPin and put it to the function "pulseIn(echoPin, HIGH)". This returns the total travel time. So we need to store this return value to the variable "duration".

Listing 6 – Arduino Code

```
duration=pulseIn(echoPin,HIGH);
```

The total travel time is now stored in the variable "duration" Now we can calculate the distance from this duration by using the equation. And store calculated value(distance) to the variable "distance". The equation is explained above

distance=(duration*0.034/2);

The distance from the sensor to the object is now stored in the variable "distance". Then we need to display it to screen. For this purpose, here we using the serial communication. You can also use LCD, Sven Segment Display, OLED Display, etc...(The will change). First print a headingor a message. Here I am going to print "Distance". by using "Serial.print("Distance : ")". After that print the distance to the serial monitor, we use the keyword "Serial.println(distance)". Then print the unit by "Serial.println(" cm ")". Here I used a "ln" with "Serial.print()". It's for start a new line. The code is like,then add Delay.

Listing 7 – Arduino Code

```
    Serial.print("Distance␣:␣" );
    Serial.print(distance)";
␣␣␣␣Serial.println(" cm ")";

        delay(1000);
```

0.7 Python Code:

Listing 8 – Python Code

```
import os
import sys
import time

cwd = os.getcwd()
(setpath,Examples) = os.path.split(cwd)
sys.path.append(setpath)

from Arduino import Arduino
from time import sleep

class HCSR04:
    def __init__(self, baudrate):
        self.baudrate = baudrate
        self.setup()
        self.run()
        self.exit()

    def setup(self):
        self.obj_arduino = Arduino()
        self.port = self.obj_arduino.locateport()
        self.obj_arduino.open_serial(1, self.port, self.baudrate)

    def run(self):
        self.echo = 8
        self.trig = 9
        while True:
            st=self.obj_arduino.cmd_dist(1,self.trig,self.echo)
            print(st)
            sleep(1)

    def exit(self):
        self.obj_arduino.close_serial()

def main():
    obj_hcsr04 = HCSR04(115200)

if __name__=='__main__':
    main()
```