DATE OF CONDUCTION -: 8/2/24

DATE OF SUBMISSION -: 15/2/24

EXPERIMENT 4

NAME -: Arjunsingh Gautam

PRN -: 22070123043

ENTC A2 2022-26

Study and working of Ultrasonic Sensors

AIM

To understand the functionality of an Ultrasonic Sensor, design a signal conditioning circuit on a breadboard and implement on Arduino IDE Software.

HARDWARE USED

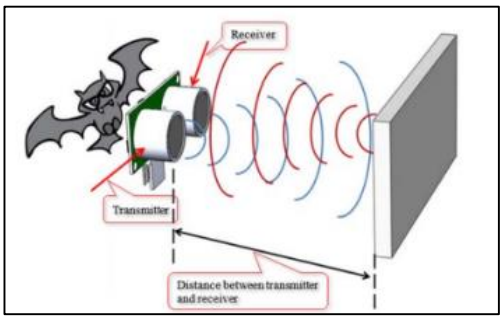
1. Arduino Board
2. Multiple Led
3. Breadboard
4. Jumper wires (male-to-female, male-to-male)
5. USB Cable
6. Computer with Arduino IDE
7. Arduino Code
8. Power Source (optional, e.g., battery pack)
9. Ultrasonic Sensor

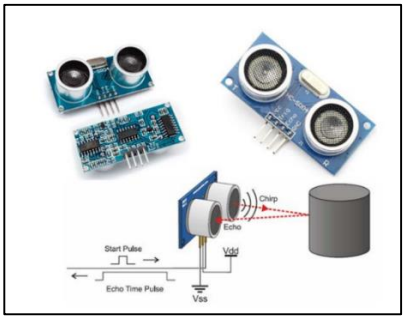
SOFTWARE USED

1. Arduino IDE for programming and microcontroller
2. Serial monitor for data visualization software for data analysis.

SYESETM INFORMATION

Ultrasonic is an excellent way of figuring out what is in the immediate vicinity of your Arduino. The basics of using ultrasound are like this: you shoot out a sound, wait to hear it echo back, and if you have your timing right, you will know if anything is out there and how far away it is. This is called echolocation and it ‘show bats and dolphins find objects in the dark and underwater, though they use lower frequencies than you can use with your Arduino.Figure-1 show the working principal of ultrasonic ranging concept



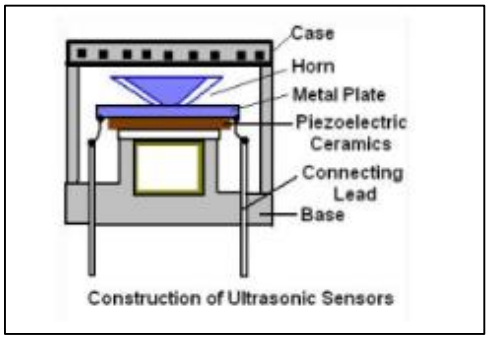


2. Module Specification

Electrical Parameters	Value
Operating Voltage	3.3Vdc ~ 5Vdc
Quiescent Current	<2mA
Operating Current	15mA
Operating Frequency	40KHz
Operating Range & Accuracy	2cm ~ 400cm (1in ~ 13ft) ± 3mm
Sensitivity	-65dB min
Sound Pressure	112dB
Effective Angle	15°
Connector	4-pins header with 2.54mm pitch
Dimension	45mm x 20mm x 15mm
Weight	9g

Sensor Construction

Piezoelectric crystals are used for sensor elements. Piezoelectric crystals will oscillate at high frequencies when electric energy is applied to it. The Piezoelectric crystals will generate electrical signal when ultrasound wave hit the sensor surface in reverse.



APPLICATIONS

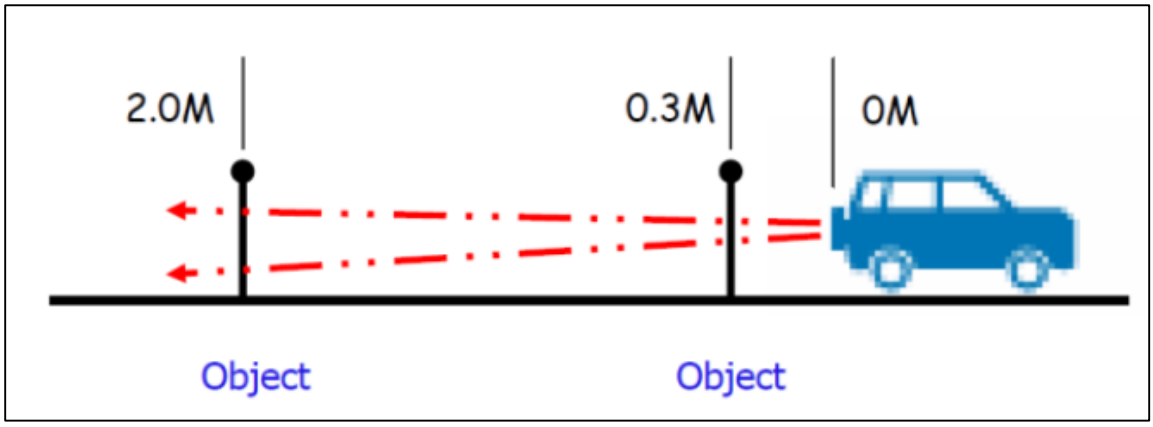
Ultrasonic sensors find diverse applications across industries, including distance measurement in parking assistance systems, obstacle detection in automotive collision avoidance systems, liquid level measurement in industrial tanks and silos, object detection in robotics and drones, medical imaging in ultrasound technology, motion detection in security systems. Some other applications include-:

- Car Parking Reverse Sensors

The main purpose is the distance range detection, which is widely used parking sensor for car. The sensor is

used for calculating the distance, or direction of an object from the time it takes for a sound wave to travel to

the object and echo back. The effective detective range is 0.3m ~ 3.0m.

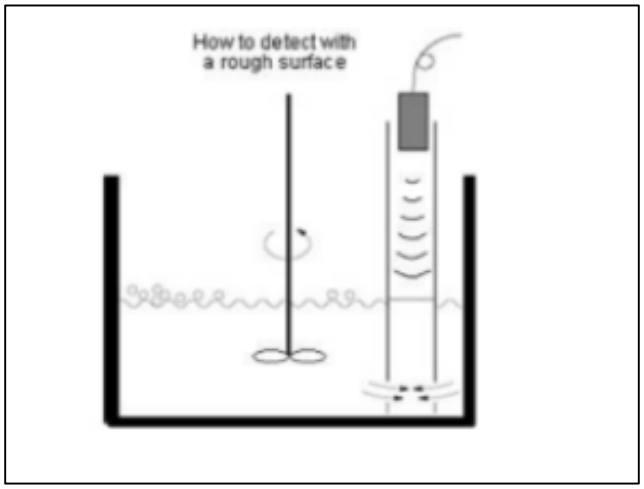


- Liquid Level Detection

Ultrasonic sensors are widely used for liquid level detection. In such cases, place a pipe on top of the sensor

head as shown Figure-3. By detecting the liquid level inside the pipe, a wavy surface or bubbles which can

disturb stable reading can be prevented.



FUNCTIONS USED IN CODE

1. `void setup()`: Initializes the serial communication and sets pin modes for trig, echo, and LED pins.

2. `void loop()`: Repeatedly triggers ultrasonic sensor pulses, measures the duration of the echo signal, calculates the distance based on the duration, and controls an LED based on the calculated distance.

3. `Serial.begin(9600)`: Initializes serial communication at a baud rate of 9600 bits per second.

4. `pinMode(TRIG_PIN, OUTPUT)`: Sets the trig pin as an output pin to trigger the ultrasonic sensor.

5. `pinMode(ECHO_PIN, INPUT)`: Sets the echo pin as an input pin to receive the echo signal from the ultrasonic sensor.

6. `pinMode(LED_PIN, OUTPUT)`: Sets the LED pin as an output pin to control the LED.

7. `digitalWrite(TRIG_PIN, HIGH)`: Sets the trig pin to a high state to trigger the ultrasonic sensor.

8. `delayMicroseconds(10)`: Delays the program execution for 10 microseconds to ensure a stable trigger pulse.

9. `digitalWrite(TRIG_PIN, LOW)`: Sets the trig pin back to a low state after triggering the sensor.

10. `duration_us = pulseIn(ECHO_PIN, HIGH)`: Measures the duration of the echo signal (time taken for the signal to return) and stores it in the variable `duration_us`.

11. `distance_cm = 0.017 * duration_us;`: Calculates the distance in centimeters based on the duration of the echo signal.

12. `if (distance_cm < DISTANCE_T) { ... } else { ... }`: Checks if the measured distance is less than a predefined threshold (`DISTANCE_T`) and controls the LED accordingly.

13. `digitalWrite(LED_PIN, HIGH)`: Turns on the LED if the measured distance is less than the threshold.

14. `digitalWrite(LED_PIN, LOW)`: Turns off the LED if the measured distance is greater than or equal to the threshold.

15. `Serial.print("distance: ")`: Prints the label "distance: " to the serial monitor.

16. `Serial.print(distance_cm)`: Prints the calculated distance in centimeters to the serial monitor.

17. `delay(500)`: Delays the program execution for 500 milliseconds before the next iteration of the loop.

CODE: TYPED CODE

```
//Abhishek Verma

//22070123004

int TRIG_PIN = 6;

int ECHO_PIN  = 7;

int LED_PIN = 3;

int DISTANCE_T = 20;


float duration_us, distance_cm;

void setup()

{
Serial.begin(9600);

pinMode(TRIG_PIN, OUTPUT);

pinMode(ECHO_PIN, INPUT);

pinMode(LED_PIN, OUTPUT);

}

Void loop()

{

digitalWrite(TRIG_PIN, HIGH);

delayMicroseconds(10);

digitalWrite(TRIG_PIN, LOW);


duration_us = pulseIn(ECHO_PIN, HIGH);


distance_cm = 0.017*duration_us;

if(distance_cm < DISTANCE_T)

digitalWrite(LED_PIN, HIGH);

else

digitalWrite(LED_PIN, LOW);

Serial.print(“distance: “);

Serial.print(“distance_cm);

Serial.print(“CM “);

delay(500);

}
```

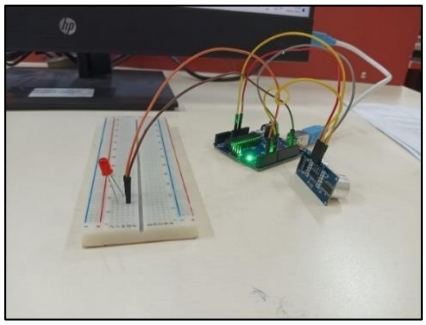
```
//Abhishek Verma

//22070123004

int TRIG_PIN = 6;

int ECHO_PIN  = 7;

int BUZZ_PIN = 3;

int LED_PIN = 3;

int DISTANCE_T = 20;

float duration_us, distance_cm;

void setup()

{

pinmode(8, OUPTUT);

pinmode(9,OUTPUT);
Serial.begin(9600);

pinMode(TRIG_PIN, OUTPUT);

pinMode(ECHO_PIN, INPUT);

pinMode(LED_PIN, OUTPUT);

}

Void loop()

{

digitalWrite(TRIG_PIN, HIGH);

delayMicroseconds(10);

digitalWrite(TRIG_PIN, LOW);


duration_us = pulseIn(ECHO_PIN, HIGH);


distance_cm = 0.017*duration_us;

if(distance_cm < DISTANCE_T)

digitalWrite(LED_PIN, HIGH);

else

digitalWrite(LED_PIN, LOW);

if(distance_cm > 0 & distance_cm <=10);{
digitalWrite(8,HIGH);

digitalWrite(buzz,LOW);{

else{digitalWrite(8, LOW);

Serial.print(“distance: “);

Serial.print(“distance_cm);

Serial.print(“CM “);

delay(500);
```

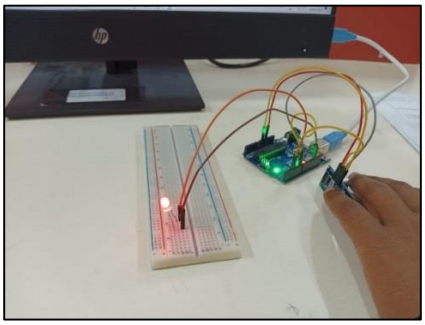
CODE: SCREENSHOT

```
int TRIG_PIN = 6;
 int ECHO_PIN = 7;
 int LED_PIN  = 3;
 int DISTANCE_T= 20;

float duration_us, distance_cm;

void setup()
{
  Serial.begin (9600);
  pinMode(TRIG_PIN, OUTPUT);
  pinMode(ECHO_PIN, INPUT);
  pinMode(LED_PIN, OUTPUT);
}

void loop()
{

  digitalWrite(TRIG_PIN, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIG_PIN, LOW);

  duration_us = pulseIn(ECHO_PIN, HIGH);

  distance_cm = 0.017 * duration_us;

  if(distance_cm < DISTANCE_T)
    digitalWrite(LED_PIN, HIGH);
  else
    digitalWrite(LED_PIN, LOW);

  Serial.print("distance: ");
  Serial.print(distance_cm);
  Serial.println(" CM");

  delay(500);
}
```

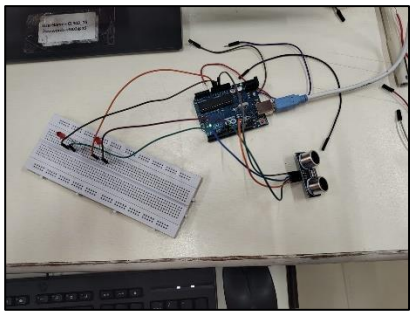
```
const int trig   = 6;
const int echo   = 7;
const int buzz = 3;
const int DISTANCE_THRESHOLD = 50;
float duration_us, distance_cm;
void setup() {
  pinMode(8, OUTPUT);
  pinMode(9, OUTPUT);
  Serial.begin (9600);
  pinMode(trig, OUTPUT);
  pinMode(echo, INPUT);
  pinMode(buzz, OUTPUT);
}
void loop() {
  digitalWrite(trig, HIGH);
  delayMicroseconds(10);
  digitalWrite(trig, LOW);
  duration_us = pulseIn(echo, HIGH);

  distance_cm = 0.017 * duration_us;

  if(distance_cm < DISTANCE_THRESHOLD)
    digitalWrite(buzz, HIGH);
  else
    digitalWrite(buzz, LOW);

  if(distance_cm > 0 && distance_cm <= 10) {
    digitalWrite(8, HIGH);
    digitalWrite(buzz, LOW);
  }
  else{
    digitalWrite(8, LOW);
  }
  Serial.print("distance: ");
  Serial.print(distance_cm);
  Serial.println(" cm");
  delay(500);
}
```

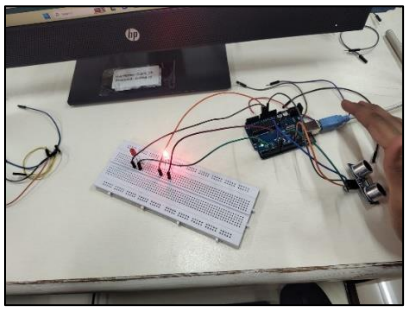
PICTURES OF THE EXPERIMENT

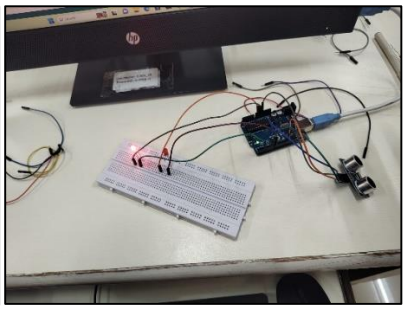


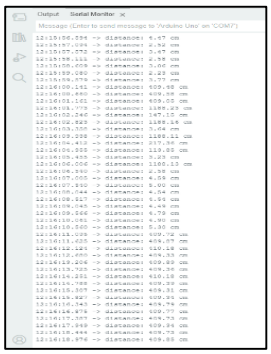


```
distance: 8.57CM
distance: 2.67CM
distance: 13.36CM
distance: 23.36CM
distance: 1.58CM
distance: 26.08CM
distance: 15.08CM
distance: 3.37CM
distance: 20.45CM
distance: 26.66CM
distance: 1.63CM
distance: 1.28CM
```









CONCLUSION

In this experiment, we interfaced an Ultrasonic sensor with help of breadboard, Arduino board and some electronic instruments. Implemented a simple program in Arduino IDE to read and display sensor values. The LED provided a visual indication when the sensor was active and working according to the functions and guidelines provided by us.