Lab 2
Playing with ultrasonic sensor

Objective

The main objective of this lab is to introduce the Ultrasonic sensor. It works by sending sound waves from the transmitter, which then bounce off of an object and then return to the receiver. You can determine how far away something is by the time it takes for the sound waves to get back to the sensor.

The components

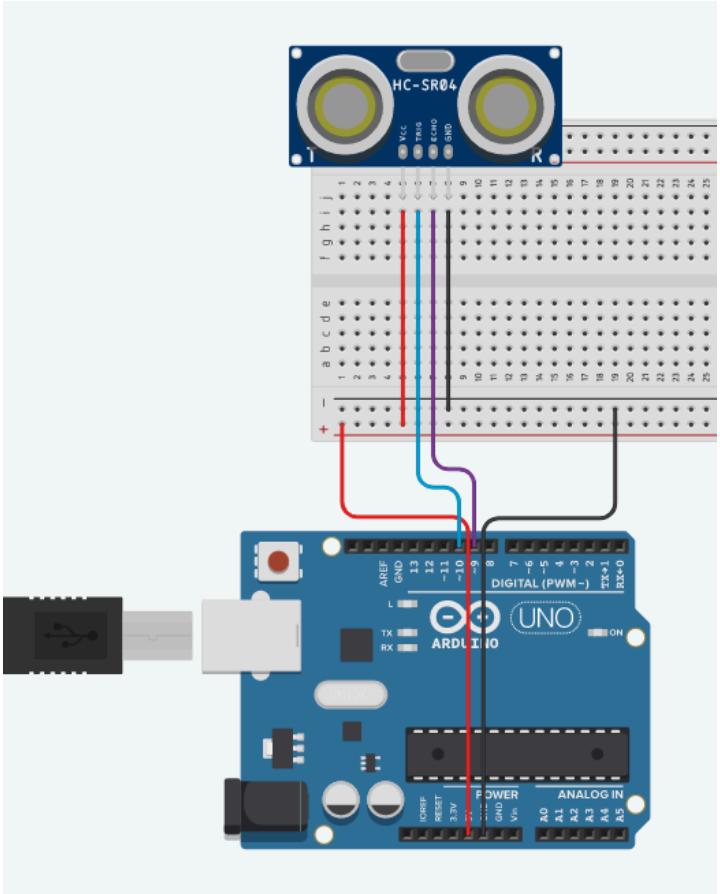
- Arduino
- Ultrasonic sensor
- Jumper wires
- Breadboard
- Servo motor
- LEDs
- Resistors

Implementation 1

In this lab, you will learn how to connect and use an ultrasonic sensor with Arduino. First, you will setup your circuit with an Arduino board and the sensor, and then discover how to calculate the distance.

Create the Arduino circuit

We must create this circuit:



How to build the circuit

1. VCC to 5V
2. GND to GND
3. Trig to pin 9
4. Echo to pin 10

Then, we write the following source code:

```
1   const int trigPin = 9;
2   const int echoPin = 10;
3   float duration, distance;
4   void setup() {
5     pinMode(trigPin, OUTPUT);
```

```
6      pinMode(echoPin, INPUT);
7      Serial.begin(9600);
8    }
9    void loop() {
10      digitalWrite(trigPin, LOW);
11      delayMicroseconds(2);
12      digitalWrite(trigPin, HIGH);
13      delayMicroseconds(10);
14      digitalWrite(trigPin, LOW);
15      duration = pulseIn(echoPin, HIGH);
16      distance = (duration*.0343)/2;
17      Serial.print("Distance: ");
18      Serial.println(distance);
19      delay(100);
20    }
```

How the sensor works

In order to generate the ultrasound we need to set the **Trigger Pin** on a **High** State for **10 µs**. That will send out an 8 cycle sonic burst which will travel at the speed sound and it will be received in the Echo Pin. The Echo Pin will **output** the **time** in microseconds the sound wave traveled.

For example, if the object is 20 cm away from the sensor, and the speed of the sound is **340 m/s** or 0.034 cm/µs the sound wave will need to travel about 588 microseconds. But what you will get from the Echo pin will be **double** that number because the sound wave needs to **travel forward** and **bounce backward**. So in order to get the distance in cm we need to multiply the received travel time value from the echo pin by 0.034 and divide it by 2.

In the loop first you have to make sure that the trigPin is clear so we have to set that pin on a **LOW State** for just **2 µs**. Now for generating the **ultrasound** wave we have to set the **trigPin** on **HIGH State** for **10 µs**. Using the **`pulseIn()`** function you have to read the travel time and put that value into the variable “duration”.

This function has 2 parameters, the first one is the name of the echo pin and for the second one you can write either HIGH or LOW. In this case, HIGH means that the **`pulseIn()`** function will wait for the pin to go HIGH caused by the bounced sound wave and it will start timing, then it will wait for the pin to go LOW when the sound wave will end which will stop the timing. At the end the function will return the length of the pulse in microseconds. For getting the distance we will multiply the duration by 0.034 and divide it by 2 as we explained this equation previously. At the end we will print the value of the distance on the Serial Monitor.

Steps :

1. First do the wiring as shown in the picture
2. Open Arduino IDE Software and write down your code, or download the code below and open it
3. Choose your own Arduino board (in this case Arduino Uno), by selecting **Tools** > **Board** > **Arduino/Geniuno Uno**
4. Choose your COM Port (usually it appears only one existing port), **Tools** > **Port** > **COM..** (If there are more than one ports, try it one by one)
5. Upload your code by pressing **Ctrl + U** or **Sketch** > **Upload**
6. To display the measurement data you can use Serial Monitor by pressing **Ctrl + Shift + M** (make sure that the baud rate speed is 9600)

Implementation 2

Now that we know how an ultrasonic sensor works, let's combine it with a set of LEDs.

This is **challenge number 1**. You have to realize the following objectives:

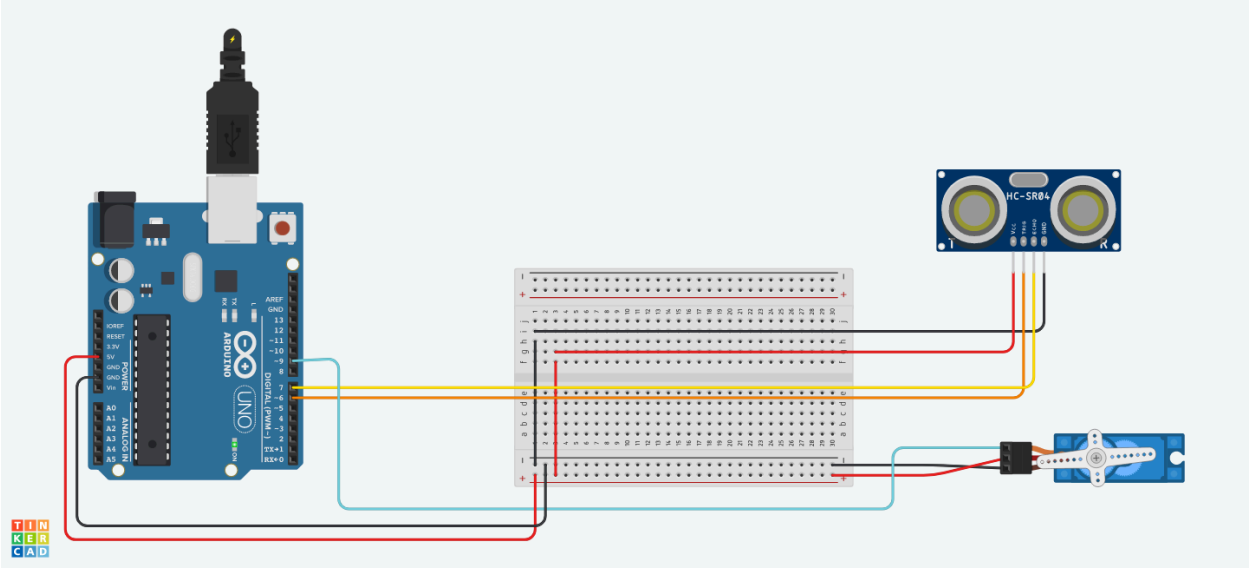
1. Build the **ultrasonic sensor** as seen previously
2. Connect 2 LEDs an seen in the previous lab (please ask your instructor to provide you two different colors, **color#1** and **color#2**)
3. **Color#1** should be **switched on** by default
4. If the distance between **10 cm** and **20 cm**, **color#1** is switched off but **color#2** is blinking every 1 sec
5. If the distance is **less than 10cm**, blink **color#1** and **color#2** alternately every 500 msec.
6. If the distance is **more that 20cm**, **switch off** the **color#2** and **switch on** the **color#1**

Implementation 3

Now that you've mastered the ultrasonic sensor input and the LEDs output, we'll move on to the servo motor. The latter will move once the capture has been sensed.

Create the Arduino circuit

We must create this circuit:



The servo motor should be connected using the following colors:

- Yellow to Pin 9
- Red to 5V
- Brown to GND

After that, use the following code:

```
1.    #include <Servo.h>
2.    const int TRIG_PIN  = 6;  // This pin is connected to TRIG pin
3.    const int ECHO_PIN  = 7;  // This pin is connected to Echo pin
4.    const int SERVO_PIN = 9; // This pin is connected to Servo pin
5.    const int DISTANCE_THRESHOLD = 50; // distance in centimeters
6.    Servo servo; // create servo object to control a servo
7.    float duration_us, distance_cm; //variables to verify the
distance
8.    void setup() {
9.      Serial.begin (9600);     // initialize serial port
10.      pinMode(TRIG_PIN, OUTPUT); // set arduino pin to output
mode
11.      pinMode(ECHO_PIN, INPUT);  // set arduino pin to input mode
12.      servo.attach(SERVO_PIN);   // attaches the servo on pin 9
to the servo object
13.      servo.write(0);
14.    }
15.    void loop() {
16.      // generate 10-microsecond pulse to TRIG pin
```

```
17.      digitalWrite(TRIG_PIN, HIGH);
18.      delayMicroseconds(10);
19.      digitalWrite(TRIG_PIN, LOW);
20.      // measure duration of pulse from ECHO pin
21.      duration_us = pulseIn(ECHO_PIN, HIGH);
22.      // calculate the distance
23.      distance_cm = 0.017 * duration_us;
24.      if(distance_cm < DISTANCE_THRESHOLD)
25.         servo.write(90); // rotate servo motor to 90 degree
26.      else
27.         servo.write(0);  // rotate servo motor to 0 degree
28.      Serial.print("distance: ");
29.      Serial.print(distance_cm);
30.      Serial.println(" cm");
31.      delay(500);
32.   }
```

This source code shows that at a distance of 50 cm or less (**line 24**), the servo motor rotates 90 degrees, otherwise it returns to its initial position.

Implementation 4

This is **challenge number 2**. You have to control the rotation of the servo motor according to the distance. The **closer** you are to the sensor (from 180 cm), the **greater the angle** of the servo motor.