Experiment No. 02

Title: Sensor and actuator interfacing with Arduino

Batch: B–1 **Roll No.: 16010422234** **Experiment No.: 02**

Aim: Sensor and actuator interfacing with Arduino

Resources needed: Internet, Arduino Board, Sensors and Actuators

---

Theory:
Pre Lab/ Prior Concepts:

The Internet of Things (IoT) is the network of physical devices, vehicles, home appliances and other items embedded with electronics, software, sensors, actuators, and connectivity which enable these objects to connect and exchange data. Each thing is uniquely identifiable through its embedded computing system but is able to inter-operate within the existing Internet infrastructure.

Role of Sensor in IoT:
Sensors are now found in a wide variety of applications, such as smart mobile devices, automotive systems, industrial control, healthcare, oil exploration and climate monitoring. Sensors are used almost everywhere, and now sensor technology is beginning to closely mimic the ultimate sensing machine, the human being. The technology that allows this to happen is sensor fusion, which leverages a microcontroller to fuse the individual data collected from multiple sensors to get a more accurate and reliable view of the data than one would get by using the data from each discrete sensor on its own. Sensor fusion creates a situation in which the whole is much greater than the sum of its parts.

Role of Actuator in IoT:
An actuator is a component of a machine that is responsible for moving and controlling a mechanism or system, for example by opening a valve. In simple terms, it is a "mover".
An actuator requires a control device (controlled by control signal) and a source of energy. The control signal is relatively low energy and may be electric voltage or current, pneumatic, or hydraulic fluid pressure, or even human power. Its main energy source may be an electric current, hydraulic pressure, or pneumatic pressure. The control device is usually a valve. When it receives a control signal, an actuator responds by converting the source's energy into mechanical motion. In the electric, hydraulic, and pneumatic sense, it is a form of automation or automatic control.

The displacement achieved is commonly linear or rotational, as exemplified by linear motors and rotary motors, respectively. Rotary motion is more natural for small machines making large

displacements. By means of a leadscrew, rotary motion can be adapted to function as a linear actuator (a linear motion, but not a linear motor).

Activity:

1. List out sensors which can be used with Arduino.

1) Temperature Sensors: DHT11, LM35, DS18B20
2) Ultrasonic Sensors: HC-SR04
3) Light Sensors: LDR (Light Dependent Resistor), TSL2561
4) Motion Sensors: PIR Sensor, Accelerometer (ADXL345)
5) Proximity Sensors: IR Sensor, Capacitive Proximity Sensor
6) Gas Sensors: MQ2, MQ135
7) Humidity Sensors: DHT11, DHT22
8) Sound Sensors: Microphone Sensor Module
9) Pressure Sensors: BMP180, BMP280
10) Touch Sensors: Capacitive Touch Sensor (TTP223)

2. Integrate a variety of available sensors in the lab with Arduino, list out step performed, circuit, code, output and components used.

Steps Performed:

1) Hardware Setup:
   Connect HC-SR04 Ultrasonic Sensor to Arduino:
   - VCC to 5V
   - GND to GND
   - Trig to Pin 12
   - Echo to Pin 11

   Place the sensor on a stable surface.
2) Write Code:
   Use the provided program with the NewPing library to measure distances.
3) Compile and Upload Code:
   Connect Arduino to a computer and upload the code using the Arduino IDE.
4) Test and Observe Output:
   Open the serial monitor in the Arduino IDE to view the measured distance in centimeters.

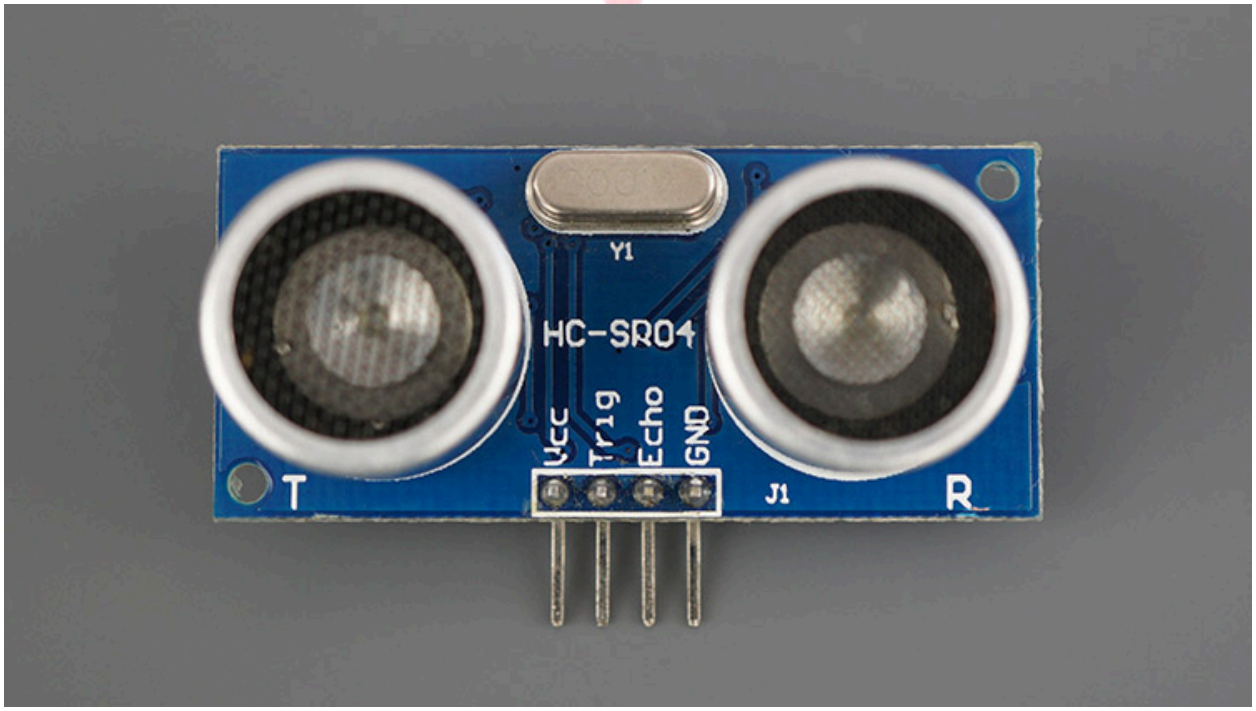
Results: (Program printout with output / Document printout as per the format)

Components Used:
Arduino Mega 2560 board
HC-SR04 ultrasonic sensor
Jumper wires

Sensor Used:
HC–SR04

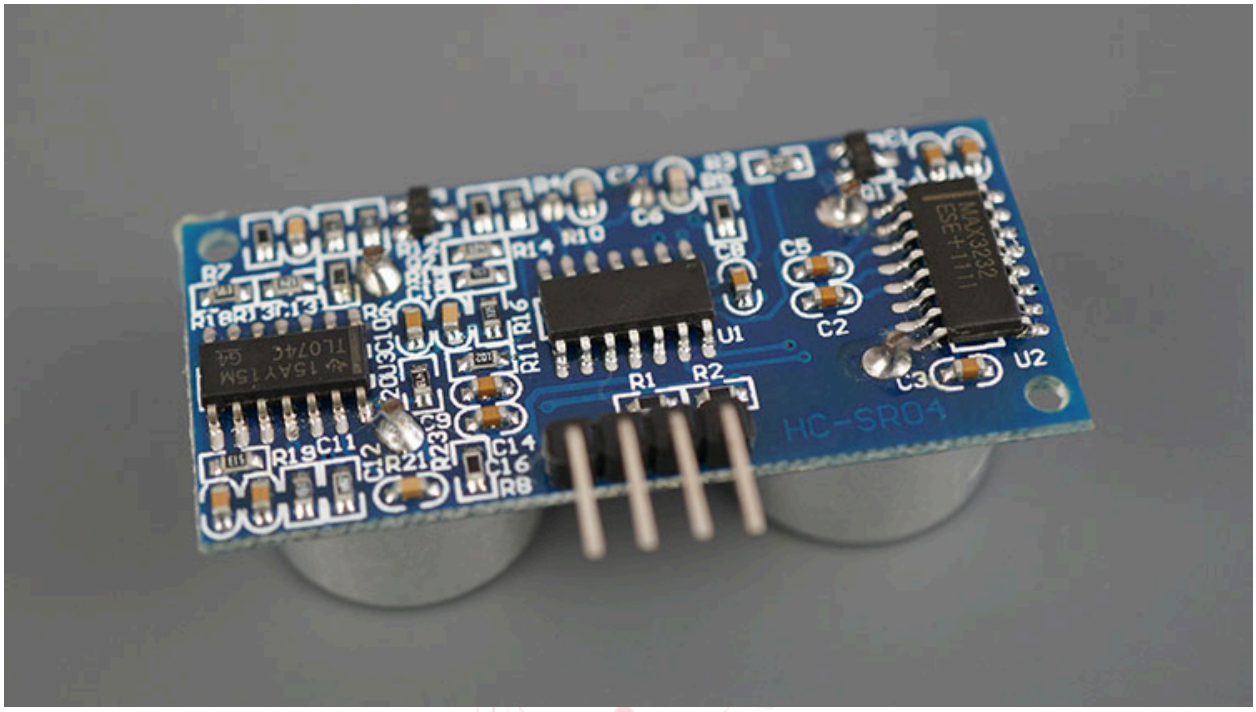
Description:
The HC-SR04 ultrasonic sensor uses sonar to determine the distance to an object. This sensor reads from 2cm to 400cm (0.8inch to 157inch) with an accuracy of 0.3cm (0.1inches), which is good for most hobbyist projects. In addition, this particular module comes with ultrasonic transmitter and receiver modules.

The following picture shows the HC-SR04 ultrasonic sensor.



HC-SR04 Ultrasonic Sensor Module Distance Measurement Component Part Front

The next picture shows the other side of the sensor.



HC-SR04 Ultrasonic Sensor Module Distance Measurement Component Part Back

Features

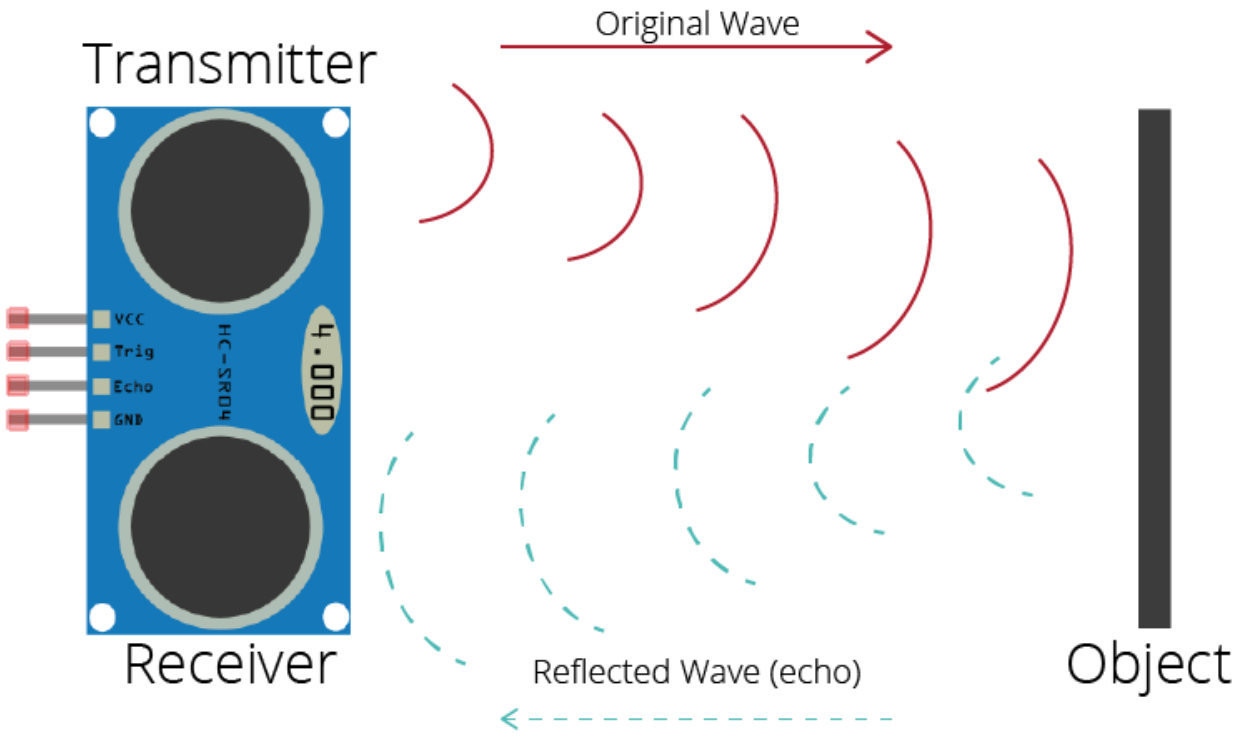
Here's a list of some of the HC-SR04 ultrasonic sensor features and specs—for more information, you should consult the sensor's datasheet:

1. Power Supply :+5V DC
2. Quiescent Current : <2mA
3. Working Current: 15mA
4. Effectual Angle: <15°
5. Ranging Distance : 2cm – 400 cm/1″ – 13ft
6. Resolution : 0.3 cm
7. Measuring Angle: 30 degree
8. Trigger Input Pulse width: 10uS TTL pulse
9. Echo Output Signal: TTL pulse proportional to the distance range
10. Dimension: 45mm x 20mm x 15mm

How Does it Work?

The ultrasonic sensor uses sonar to determine the distance to an object. Here's what happens:

1. The ultrasound transmitter (trig pin) emits a high-frequency sound (40 kHz).
2. The sound travels through the air. If it finds an object, it bounces back to the module.
3. The ultrasound receiver (echo pin) receives the reflected sound (echo).



How Ultrasonic Sensor Works

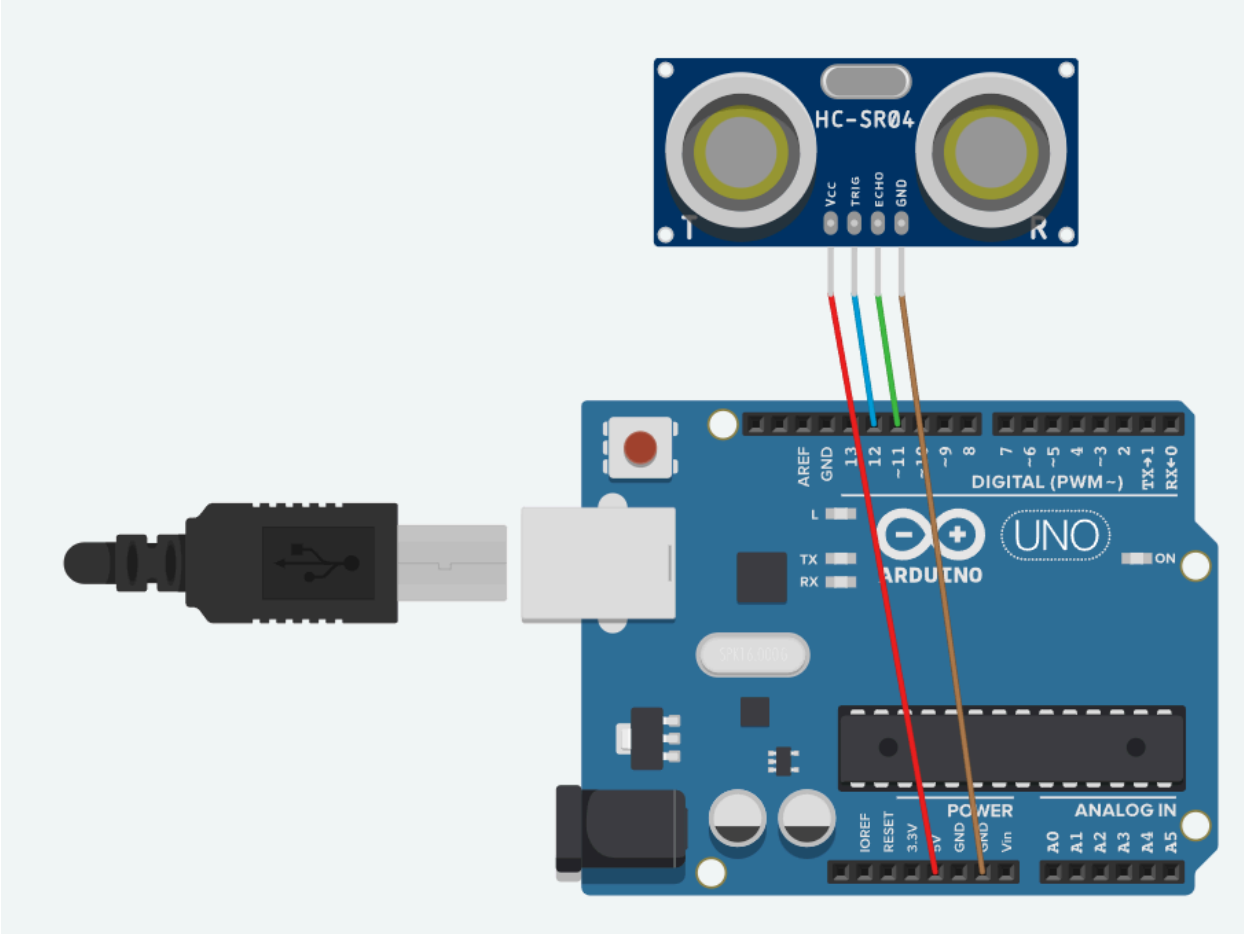
The time between the transmission and reception of the signal allows us to calculate the distance to an object. This is possible because we know the sound's velocity in the air. Here's the formula: distance to an object = ((speed of sound in the air)*time)/2
– speed of sound in the air at 20ºC (68ºF) = 343m/s

HC-SR04 Ultrasonic Sensor Pinout

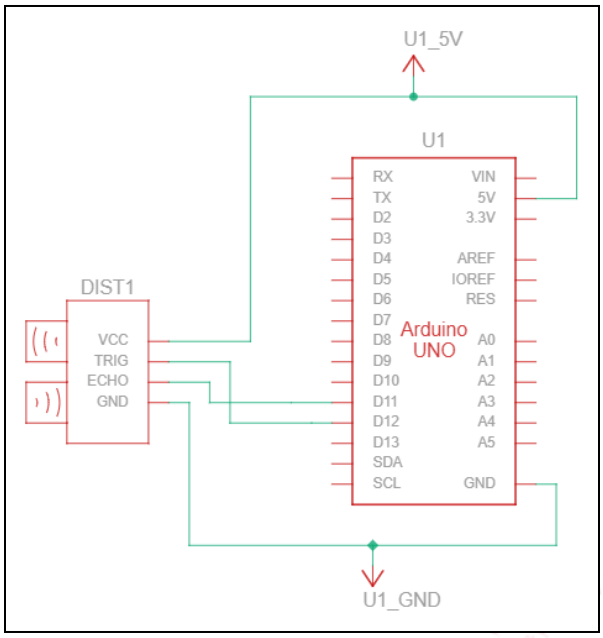
Here's the pinout of the HC-SR04 Ultrasonic Sensor.
VCC – Powers the sensor (5V)
Trig – Trigger Input Pin
Echo – Echo Output Pin
GND – Common GND

HC–SR04 connections:
VCC → 5V
GND → GND
Trig → Arduino Pin 12
Echo → Arduino Pin 11

Circuit View:

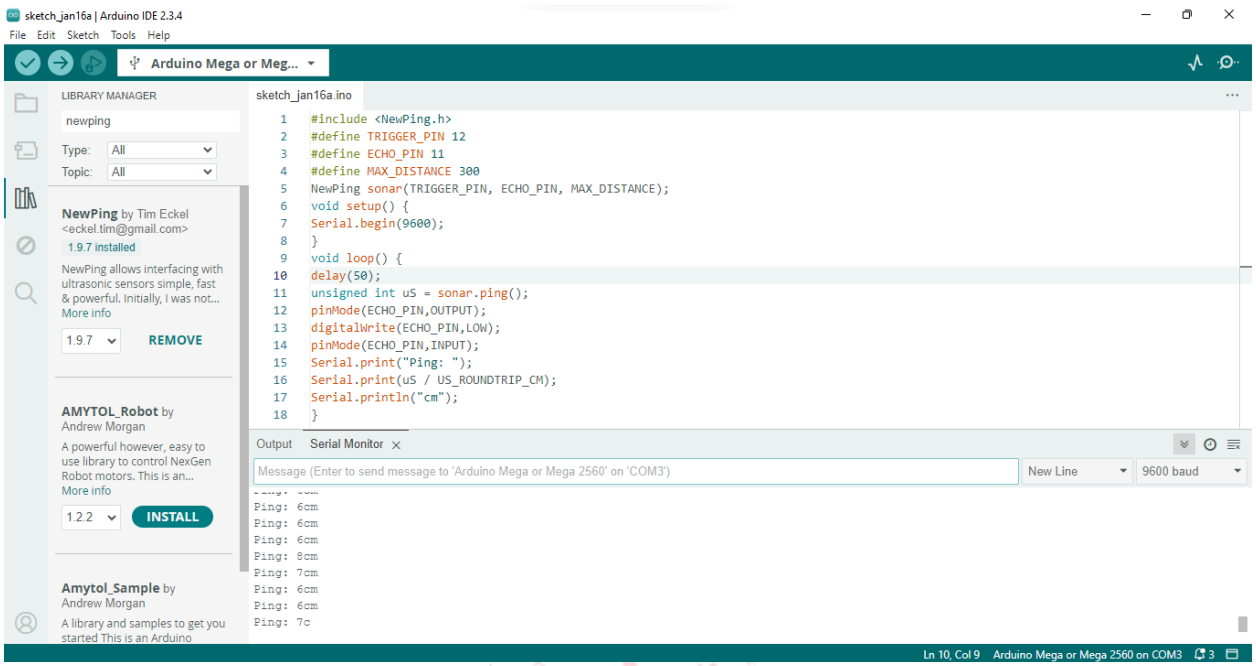


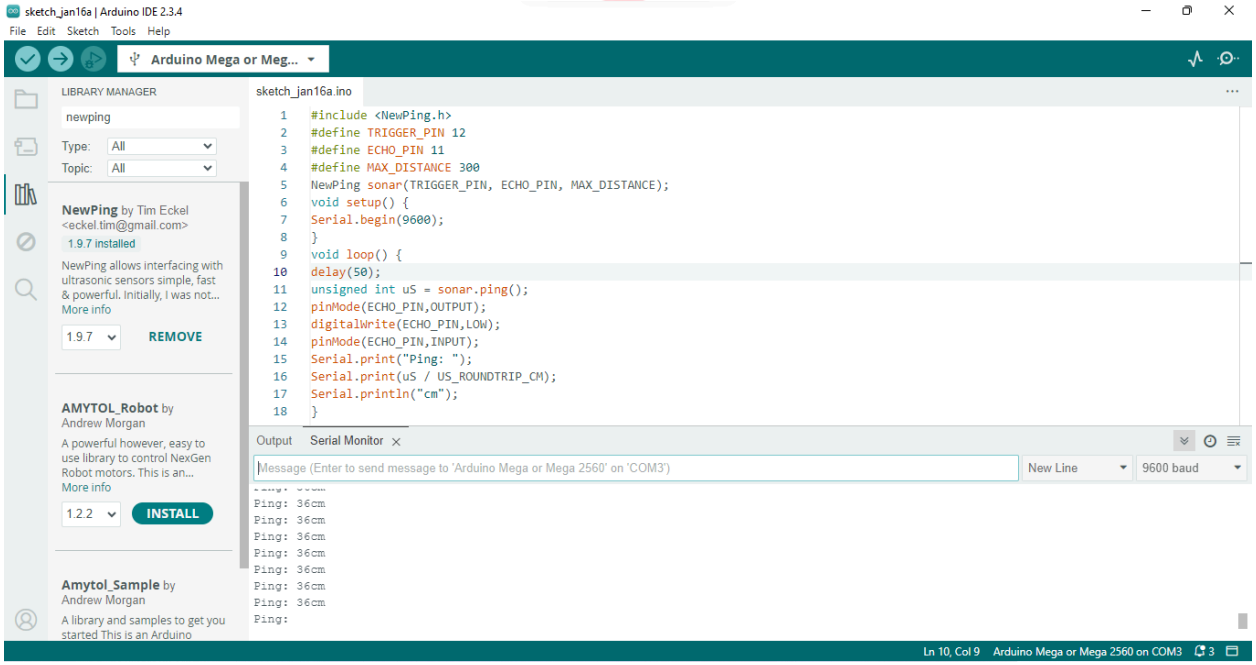
Schematic View:



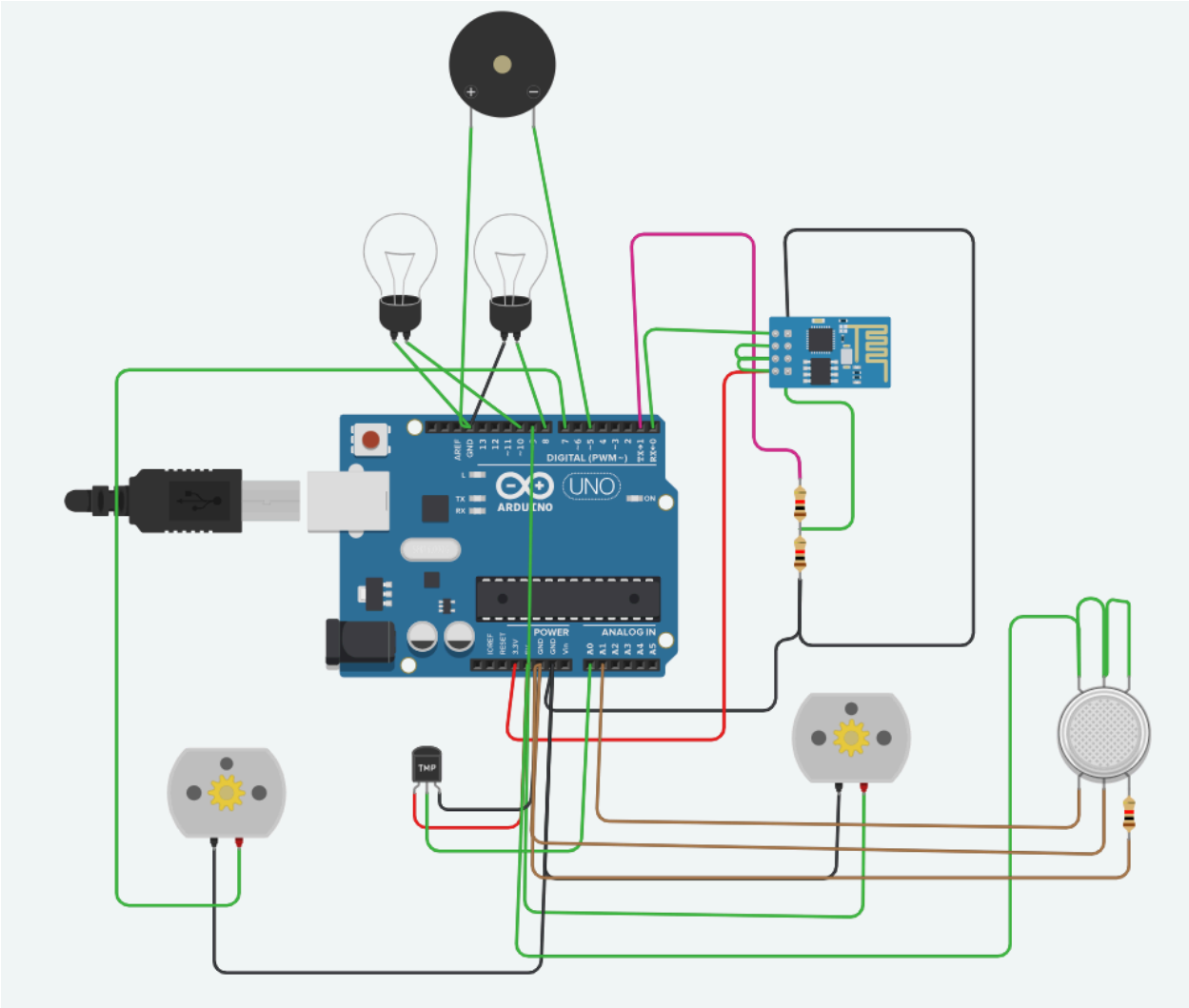
Arduino IDE Code:

```
#include <NewPing.h>
#define TRIGGER_PIN 12
#define ECHO_PIN 11
#define MAX_DISTANCE 300

NewPing sonar(TRIGGER_PIN, ECHO_PIN, MAX_DISTANCE);

void setup() {
Serial.begin(9600);
}

void loop() {
delay(50);
unsigned int uS = sonar.ping();
pinMode(ECHO_PIN,OUTPUT);
digitalWrite(ECHO_PIN,LOW);
pinMode(ECHO_PIN,INPUT);
Serial.print("Ping: ");
Serial.print(uS / US_ROUNDTRIP_CM);
Serial.println("cm");
}
```

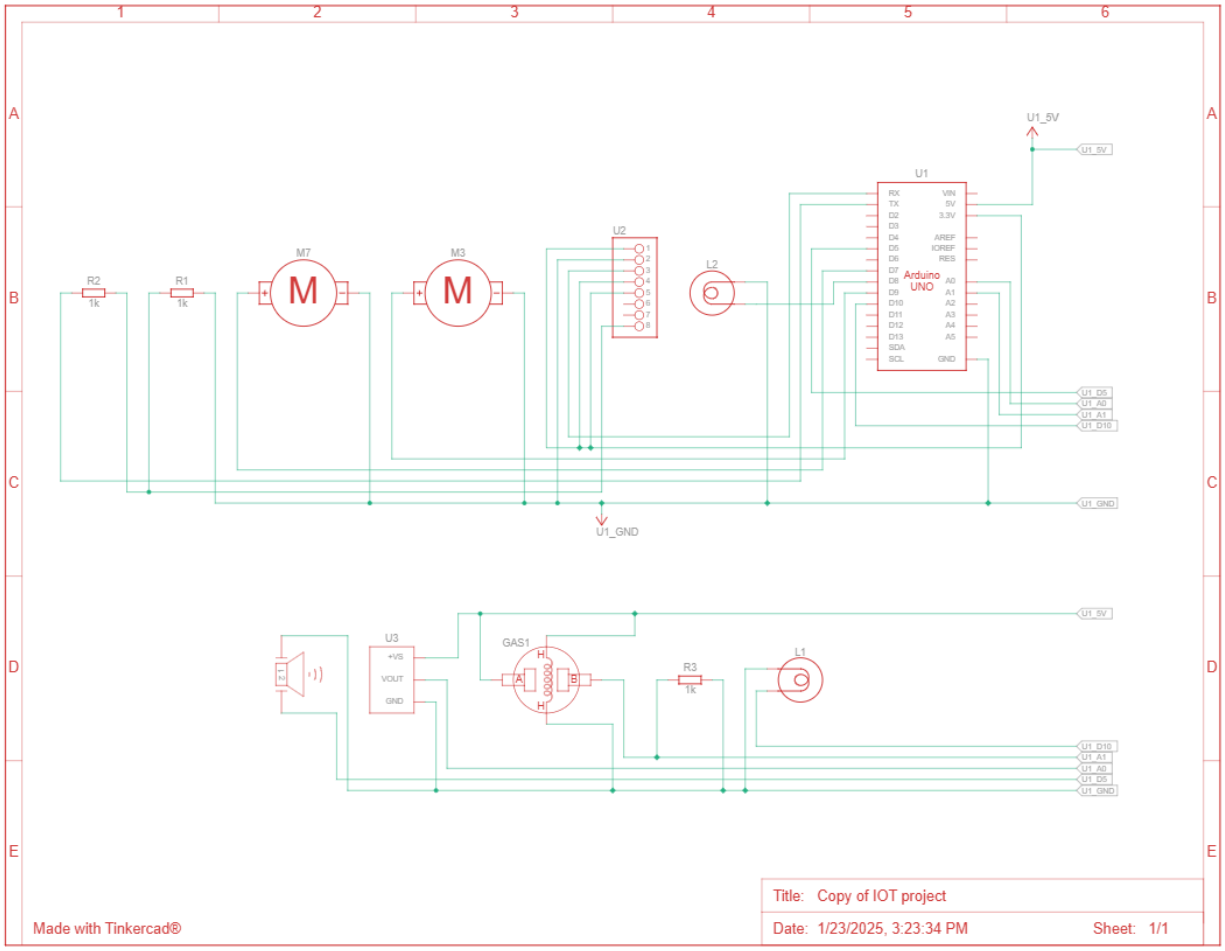

sketch_jan16a | Arduino IDE 2.3.4
File Edit Sketch Tools Help
Arduino Mega or Meg...
LIBRARY MANAGER
newping
Type: All
Topic: All
NewPing by Tim Eckel <eckel.tim@gmail.com>
1.9.7 installed
NewPing allows interfacing with ultrasonic sensors simple, fast & powerful. Initially, I was not...
More info
1.9.7 REMOVE
AMYTOL_Robot by Andrew Morgan
A powerful however, easy to use library to control NexGen Robot motors. This is an...
More info
1.2.2 INSTALL
Amytol_Sample by Andrew Morgan
A library and samples to get you started This is an Arduino
sketch_jan16a.ino
#include <NewPing.h>
#define TRIGGER_PIN 12
#define ECHO_PIN 11
#define MAX_DISTANCE 300
NewPing sonar(TRIGGER_PIN, ECHO_PIN, MAX_DISTANCE);
void setup() {
Serial.begin(9600);
}
void loop() {
delay(50);
unsigned int uS = sonar.ping();
pinMode(ECHO_PIN,OUTPUT);
digitalWrite(ECHO_PIN,LOW);
pinMode(ECHO_PIN,INPUT);
Serial.print("Ping: ");
Serial.print(uS / US_ROUNDTRIP_CM);
Serial.println("cm");
}
Output Serial Monitor
Message (Enter to send message to 'Arduino Mega or Mega 2560' on 'COM3')
New Line 9600 baud
Ping: 6cm
Ping: 6cm
Ping: 6cm
Ping: 8cm
Ping: 7cm
Ping: 6cm
Ping: 6cm
Ping: 7c
Ln 10, Col 9 Arduino Mega or Mega 2560 on COM3


sketch_jan16a | Arduino IDE 2.3.4
File Edit Sketch Tools Help
Arduino Mega or Meg...
LIBRARY MANAGER
newping
Type: All
Topic: All
NewPing by Tim Eckel <eckel.tim@gmail.com>
1.9.7 installed
NewPing allows interfacing with ultrasonic sensors simple, fast & powerful. Initially, I was not...
More info
1.9.7 REMOVE
AMYTOL_Robot by Andrew Morgan
A powerful however, easy to use library to control NexGen Robot motors. This is an...
More info
1.2.2 INSTALL
Amytol_Sample by Andrew Morgan
A library and samples to get you started This is an Arduino
sketch_jan16a.ino
#include <NewPing.h>
#define TRIGGER_PIN 12
#define ECHO_PIN 11
#define MAX_DISTANCE 300
NewPing sonar(TRIGGER_PIN, ECHO_PIN, MAX_DISTANCE);
void setup() {
Serial.begin(9600);
}
void loop() {
delay(50);
unsigned int uS = sonar.ping();
pinMode(ECHO_PIN,OUTPUT);
digitalWrite(ECHO_PIN,LOW);
pinMode(ECHO_PIN,INPUT);
Serial.print("Ping: ");
Serial.print(uS / US_ROUNDTRIP_CM);
Serial.println("cm");
}
Output Serial Monitor
Message (Enter to send message to 'Arduino Mega or Mega 2560' on 'COM3')
New Line 9600 baud
Ping: 36cm
Ping: 36cm
Ping: 36cm
Ping: 36cm
Ping: 36cm
Ping: 36cm
Ping: 36cm
Ping:
Ln 10, Col 9 Arduino Mega or Mega 2560 on COM3

Circuit View:



Schematic View:



Code:

```
void setup()
{
  pinMode(A0, INPUT);
  pinMode(A1,INPUT);
  pinMode(9, OUTPUT);
  pinMode(8, OUTPUT);
  pinMode(7, OUTPUT);
  pinMode(10, OUTPUT);
  Serial.begin(9600);
}

void loop()
{
  int melody = 150;
```

```
  int MQ2pin = A1;

  while (1 != 0) {

    int sensorValue = analogRead(MQ2pin);

    if(sensorValue >= 200){
      tone(5, melody) ;
      Serial.print(sensorValue);
    Serial.println(" |SMOKE DETECTED");
    }else{
      digitalWrite(5,LOW);
       Serial.print(sensorValue);
             Serial.println(" |NO SMOKE DETECTED");
    }


    if (-40 + 0.488155 * (analogRead(A0) - 20) < 30) {
      if (-40 + 0.488155 * (analogRead(A0) - 20) < 20) {
        digitalWrite(9, LOW);
        digitalWrite(8, HIGH);
        digitalWrite(7, LOW);
        digitalWrite(10, HIGH);
      } else {
        digitalWrite(9, LOW);
        digitalWrite(8, LOW);
        digitalWrite(10, HIGH);
        digitalWrite(7, LOW);
      }
    } else {
       if (-40 + 0.488155 * (analogRead(A0) - 20) > 30 && -40 + 0.488155 *
(analogRead(A0) - 20) < 40) {
        digitalWrite(9, HIGH);
        digitalWrite(10, LOW);
        digitalWrite(8, LOW);
        digitalWrite(7, LOW);
      } else {
        digitalWrite(9, HIGH);
        digitalWrite(8, LOW);
        digitalWrite(7, HIGH);
        digitalWrite(10, LOW);
      }
    }
  }

  -40 + 0.488155 * (analogRead(A0) - 20);
  delay(10); // Delay a little bit to improve simulation performance
}
```

Explanation of the Circuit and Code in TinkerCAD

This circuit is built around an Arduino Uno, integrating multiple sensors and actuators to detect temperature, smoke, and light levels, triggering outputs such as a buzzer and bulbs accordingly.

Hardware Components

Arduino Uno – Main microcontroller.
MQ-2 Gas Sensor – Detects smoke/gas.
LM35 Temperature Sensor – Measures ambient temperature.
LDR (Light Dependent Resistor) – Detects light intensity.
Buzzer – Provides an alarm when smoke is detected.
Bulbs (LEDs) – Indicate different temperature levels.
ESP8266 WiFi Module – Possible wireless communication (though not utilized in the given code).

Circuit Working

1. Smoke Detection (MQ-2 Sensor)
Analog pin A1 reads the smoke level.
If the smoke level ≥ 200, the buzzer (Pin 5) turns ON (alarm activated).
If smoke level < 200, the buzzer remains OFF.
The detection status is printed on the Serial Monitor.

2. Temperature Monitoring (LM35 Sensor)
Reads temperature from A0 and calculates it using:
$\text{Temperature} = -40 + 0.488155 \times (\text{analogRead}(A0) - 20)$
Based on temperature levels:
Below 20°C → Bulb 1 (Pin 8) ON, Bulb 2 (Pin 9) OFF.
Between 20°C and 30°C → Both bulbs OFF.
Between 30°C and 40°C → Bulb 2 (Pin 9) ON.
Above 40°C → Bulb 1 & 2 ON (Indicating High Temperature).

3. Additional Components
LDR Sensor (appears in the circuit but is not handled in the code).
ESP8266 WiFi Module (wiring suggests connectivity, but it's not used in the provided code).

Summary

Buzzer activates when smoke is detected.
Bulbs light up based on temperature levels.
Serial monitor logs real-time smoke and temperature data.

Questions:

1. How Does IoT Work?

Ans: 1) Devices and Sensors: IoT devices collect data using sensors and actuators embedded in physical objects.
2) Connectivity: Devices connect to a network via Wi-Fi, Bluetooth, Zigbee, or cellular communication.
3) Data Transmission: The collected data is transmitted to cloud servers or edge devices for processing.
4) Data Processing: The cloud processes the data, enabling insights and decisions through analytics and machine learning.
5) Action: Processed data triggers actions through actuators or provides feedback to users via applications.

Outcomes: CO3 – Realize the design process of IoT applications and IoT challenges.

Conclusion:

The experiment successfully demonstrated sensor and actuator interfacing with Arduino. The HC-SR04 ultrasonic sensor was integrated with Arduino, and the distance was measured accurately. This experiment highlighted the role of sensors and actuators in IoT systems, emphasizing their importance in automation and real-time monitoring.

Grade: AA / AB / BB / BC / CC / CD / DD

Signature of faculty in–charge with date

References:

Links:

1. https://en.wikipedia.org/wiki/Internet_of_things
2. https://www.tinkercad.com/things/aSBH9ShfJ80-iot-project-

Books:

1. Jan Holler, Vlasios Tsiatsis, Catherine Mulligan, Stefan Avesand, Stamatis Karnouskos, David Boyle, "From Machine-to-Machine to the Internet of Things: Introduction to a New Age of Intelligence", 1st Edition, Academic Press, 2014.
2. Vijay Madisetti and Arshdeep Bahga, "Internet of Things (A Hands-on-Approach)", 1st Edition, VPT, 2014.
3. Dr. Ovidiu Vermesan, Dr. Peter Friess, "Internet of Things - From Research and Innovation to Market Deployment", River Publisher, 2014