1.Develop a program using the Arduino IDE to interface an Ultrasonic sensor with an Arduino Uno and display the measured distance on the ThingSpeak cloud platform using the required communication interface.

```
#include <ESP8266WiFi.h>

// Wi-Fi credentials
const char* ssid = "Redmi Note 12 Pro 5G";
const char* password = "redmi1212";

// ThingSpeak API settings
const char* host = "api.thingspeak.com";
String apiKey = "K074D92MRK95XA8S";

// Ultrasonic sensor pins
#define TRIG_PIN 16  // Change based on your wiring
#define ECHO_PIN 17 // Change based on your wiring

WiFiClient client;

void setup() {
  Serial.begin(115200);

  // Setup ultrasonic sensor pins
  pinMode(TRIG_PIN, OUTPUT);
  pinMode(ECHO_PIN, INPUT);

  // Connect to Wi-Fi
  WiFi.begin(ssid, password);
  Serial.print("Connecting to Wi-Fi");
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.print(".");
  }
  Serial.println("\nConnected to Wi-Fi!");
}

void loop() {
  // Trigger ultrasonic pulse
  digitalWrite(TRIG_PIN, LOW);
  delayMicroseconds(2);
  digitalWrite(TRIG_PIN, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIG_PIN, LOW);

  // Measure echo duration
  long duration = pulseIn(ECHO_PIN, HIGH);
  // Calculate distance in cm
  float distance = duration * 0.034 / 2;

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

  // Send to ThingSpeak
  if (client.connect(host, 80)) {
```

```
    String url = "/update?api_key=" + apiKey + "&field1=" +
String(distance);
    client.print(String("GET ") + url + " HTTP/1.1\r\n" +
                 "Host: " + host + "\r\n" +
                 "Connection: close\r\n\r\n");
    client.stop();
  }

  delay(500); // ThingSpeak rate limit: 15 sec
}
```

2.Develop a program using the Arduino IDE to interface a DHT11 sensor with an Arduino Uno and display the measured data on an MQTT App. Use the required communication interface.

```
 #include <ESP8266WiFi.h>
#include <PubSubClient.h>
#include "DHT.h"


// DHT11 configuration
#define DHTPIN 14
#define DHTTYPE DHT11
DHT dht(DHTPIN, DHTTYPE);

// WiFi credentials
const char* ssid = "name";
const char* password = "password";

// MQTT broker details
const char* mqtt_server = "broker.emqx.io";
const int mqtt_port = 1883;
const char* client_id = "db032482-2f62-4ba8-9782-da6521020775";
const char* topic = "quantanics/industry/dht11_data";

WiFiClient espClient;
PubSubClient client(espClient);

void setup() {
  Serial.begin(115200);
  dht.begin();

  // Connect to WiFi
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("Connecting to WiFi...");
  }
  Serial.println("WiFi connected");

  // Setup MQTT server
  client.setServer(mqtt_server, mqtt_port);
}

void reconnect() {
  while (!client.connected()) {
    Serial.println("Connecting to MQTT...");
    if (client.connect(client_id)) {
      Serial.println("Connected to MQTT broker");
```

```
    } else {
      Serial.print("Failed, rc=");
      Serial.print(client.state());
      Serial.println(" Retrying in 5 seconds...");
      delay(5000);
    }
  }
}

void loop() {
  if (!client.connected()) {
    reconnect();
  }
  client.loop();

  // Read temperature and humidity
  float humidity = dht.readHumidity();
  float temperature = dht.readTemperature();

  if (isnan(humidity) || isnan(temperature)) {
    Serial.println("Failed to read from DHT sensor!");
    return;
  }

  // Create message
  String payload = "Temp: " + String(temperature) + " °C, Humidity: " + String(humidity) + " %";

  // Publish and print
  client.publish(topic, payload.c_str());
  Serial.println(payload);

  delay(2000); // Wait before next reading
}
```

3.Develop a program using the Arduino IDE to interface the DHT11 sensor with the ESP32 to measure temperature and humidity. Transmit the received data to the ThingSpeak cloud platform and visualize the readings in real time.

```
#include <WiFi.h>
#include "DHT.h"

// DHT11 configuration
#define DHTPIN 14
#define DHTTYPE DHT11
DHT dht(DHTPIN, DHTTYPE);

// Wi-Fi credentials
const char* ssid = "Your_WiFi_SSID";
const char* password = "Your_WiFi_Password";

// ThingSpeak settings
const char* host = "api.thingspeak.com";
String apiKey = "Your_ThingSpeak_API_Key";

WiFiClient client;

void setup() {
  Serial.begin(115200);
  dht.begin();
```

```
  // Connect to WiFi
  WiFi.begin(ssid, password);
  Serial.print("Connecting to WiFi");
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.print(".");
  }
  Serial.println("\nWiFi connected");
}

void loop() {
  // Read temperature and humidity
  float humidity = dht.readHumidity();
  float temperature = dht.readTemperature();

  if (isnan(humidity) || isnan(temperature)) {
    Serial.println("Failed to read from DHT sensor!");
    return;
  }

  Serial.print("Temperature: ");
  Serial.print(temperature);
  Serial.print(" °C, Humidity: ");
  Serial.print(humidity);
  Serial.println(" %");

  // Send data to ThingSpeak
  if (client.connect(host, 80)) {
    String url = "/update?api_key=" + apiKey +
                 "&field1=" + String(temperature) +
                 "&field2=" + String(humidity);
    client.print(String("GET ") + url + " HTTP/1.1\r\n" +
                 "Host: " + host + "\r\n" +
                 "Connection: close\r\n\r\n");
    client.stop();
  }

  delay(15000); // ThingSpeak requires 15 seconds delay between updates
}
```

4.Using Arduino IDE, develop a program to interface an Ultrasonic sensor with the ESP32 to measure the distance of an object. Display the distance on the ThingSpeak cloud platform.

```
#include <WiFi.h>

// Wi-Fi credentials
const char* ssid = "Redmi Note 12 Pro 5G";
const char* password = "redmi1212";

// ThingSpeak API settings
const char* host = "api.thingspeak.com";
String apiKey = "K074D92MRK95XA8S";

// Ultrasonic sensor pins
#define TRIG_PIN 16  // Change based on your wiring
#define ECHO_PIN 17 // Change based on your wiring

WiFiClient client;
```

```
void setup() {
  Serial.begin(115200);

  // Setup ultrasonic sensor pins
  pinMode(TRIG_PIN, OUTPUT);
  pinMode(ECHO_PIN, INPUT);

  // Connect to Wi-Fi
  WiFi.begin(ssid, password);
  Serial.print("Connecting to Wi-Fi");
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.print(".");
  }
  Serial.println("\nConnected to Wi-Fi!");
}

void loop() {
  // Trigger ultrasonic pulse
  digitalWrite(TRIG_PIN, LOW);
  delayMicroseconds(2);
  digitalWrite(TRIG_PIN, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIG_PIN, LOW);

  // Measure echo duration
  long duration = pulseIn(ECHO_PIN, HIGH);
  // Calculate distance in cm
  float distance = duration * 0.034 / 2;

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

  // Send to ThingSpeak
  if (client.connect(host, 80)) {
    String url = "/update?api_key=" + apiKey + "&field1=" + String(distance);
    client.print(String("GET ") + url + " HTTP/1.1\r\n" +
                 "Host: " + host + "\r\n" +
                 "Connection: close\r\n\r\n");
    client.stop();
  }

  delay(500); // ThingSpeak rate limit: 15 sec
}
```

5.Write a program to interface the DHT11 sensor with the ESP32 and display the measured temperature and humidity on an MQTT App.

```
#include <WiFi.h>
#include <PubSubClient.h>
#include "DHT.h"

// DHT11 configuration
#define DHTPIN 14
#define DHTTYPE DHT11
DHT dht(DHTPIN, DHTTYPE);

// WiFi credentials
const char* ssid = "name";
const char* password = "password";
```

```
// MQTT broker details
const char* mqtt_server = "broker.emqx.io";
const int mqtt_port = 1883;
const char* client_id = "db032482-2f62-4ba8-9782-da6521020775";
const char* topic = "quantanics/industry/dht11_data";

WiFiClient espClient;
PubSubClient client(espClient);

void setup() {
  Serial.begin(115200);
  dht.begin();

  // Connect to WiFi
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("Connecting to WiFi...");
  }
  Serial.println("WiFi connected");

  // Setup MQTT server
  client.setServer(mqtt_server, mqtt_port);
}

void reconnect() {
  while (!client.connected()) {
    Serial.println("Connecting to MQTT...");
    if (client.connect(client_id)) {
      Serial.println("Connected to MQTT broker");
    } else {
      Serial.print("Failed, rc=");
      Serial.print(client.state());
      Serial.println(" Retrying in 5 seconds...");
      delay(5000);
    }
  }
}

void loop() {
  if (!client.connected()) {
    reconnect();
  }
  client.loop();

  // Read temperature and humidity
  float humidity = dht.readHumidity();
  float temperature = dht.readTemperature();

  if (isnan(humidity) || isnan(temperature)) {
    Serial.println("Failed to read from DHT sensor!");
    return;
  }

  // Create message
  String payload = "Temp: " + String(temperature) + " °C, Humidity: " + String(humidity) + " %";

  // Publish and print
  client.publish(topic, payload.c_str());
  Serial.println(payload);

  delay(2000); // Wait before next reading
}
```

6.Write a program to interface an Ultrasonic sensor with the ESP32 to measure the distance of an obstacle. Implement the MQTT protocol to transmit and publish the measured distance to MQTTBox.

```
#include <WiFi.h>
#include <PubSubClient.h> // Define ultrasonic sensor pins
#define TRIGGER_PIN 16
#define ECHO_PIN 17 // WiFi credentials
const char* ssid = "Redmi Note 12 Pro 5G";
const char* password = "redmi1212"; // MQTT broker details
const char* mqtt_server = "broker.emqx.io";
const int mqtt_port = 1883; // MQTT Client ID (as shown in your MQTTBox)
const char* client_id = "5b81d50b-fd38-4764-8217-062336685962"; // MQTT topic to publish
distance data
const char* topic = "Sensors19";
WiFiClient espClient;
PubSubClient client(espClient);
void setup() {
  Serial.begin(115200);
    pinMode(TRIGGER_PIN, OUTPUT);
  pinMode(ECHO_PIN, INPUT);  // Connect to WiFi
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("Connecting to WiFi...");
  }
  Serial.println("WiFi connected");   // Setup MQTT server
  client.setServer(mqtt_server, mqtt_port);
}
void reconnect() {
  while (!client.connected()) {
    Serial.println("Connecting to MQTT...");
    if (client.connect(client_id)) {
      Serial.println("Connected to MQTT broker");
    } else {
      Serial.print("Failed, rc=");
      Serial.print(client.state());
      Serial.println(" Retrying in 5 seconds...");
      delay(5000);
    }
  }
}

void loop() {
  if (!client.connected()) {
    reconnect();
  }
  client.loop();

  // Measure distance
  digitalWrite(TRIGGER_PIN, LOW);
  delayMicroseconds(2);
  digitalWrite(TRIGGER_PIN, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIGGER_PIN, LOW);

  long duration = pulseIn(ECHO_PIN, HIGH);
```

```
  float distance = duration * 0.034 / 2;

  if (distance >= 0 && distance <= 400) {
    String payload = "Distance: " + String(distance) + " cm";
    client.publish(topic, payload.c_str());
    Serial.println(payload);
  } else {
    Serial.println("Out of range");
  }

  delay(1000); // Wait before next measurement
}
```

7.Write a program to interface a Soil Moisture sensor with the ESP32 to measure the moisture content. Implement the MQTT protocol to transmit the data to MQTTBox and display the readings in real time.

```
#include <WiFi.h>
#include <PubSubClient.h>

// Soil moisture sensor pin
#define SOIL_PIN 35

// WiFi credentials
const char* ssid = "name";
const char* password = "password";

// MQTT broker details
const char* mqtt_server = "broker.emqx.io";
const int mqtt_port = 1883;
const char* client_id = "db032482-2f62-4ba8-9782-da6521020775";
const char* topic = "quantanics/industry/testing1";

WiFiClient espClient;
PubSubClient client(espClient);

void setup() {
  Serial.begin(115200);
  pinMode(SOIL_PIN, INPUT);

  // Connect to WiFi
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("Connecting to WiFi...");
  }
  Serial.println("WiFi connected");

  // Setup MQTT server
  client.setServer(mqtt_server, mqtt_port);
}

void reconnect() {
  while (!client.connected()) {
    Serial.println("Connecting to MQTT...");
    if (client.connect(client_id)) {
      Serial.println("Connected to MQTT broker");
```

```
    } else {
      Serial.print("Failed, rc=");
      Serial.print(client.state());
      Serial.println(" Retrying in 5 seconds...");
      delay(5000);
    }
  }
}

void loop() {
  if (!client.connected()) {
    reconnect();
  }
  client.loop();

  // Read soil moisture
  int soilValue = analogRead(SOIL_PIN);

  // Create message string
  String payload = "Soil Moisture: " + String(soilValue);

  // Publish and print
  client.publish(topic, payload.c_str());
  Serial.println(payload);

  delay(1000); // Wait before next reading
}
```

8.Write a program to interface a Smoke/Gas sensor with the ESP32 to measure the quality of air and display the data on an MQTT App.

```
#include <WiFi.h>
#include <PubSubClient.h>


// Gas sensor pin
#define GAS_SENSOR_PIN 36

// WiFi credentials
const char* ssid = "name";
const char* password = "password";

// MQTT broker details
const char* mqtt_server = "broker.emqx.io";
const int mqtt_port = 1883;
const char* client_id = "db032482-2f62-4ba8-9782-da6521020775";
const char* topic = "quantanics/industry/gas_sensor_data";

WiFiClient espClient;
PubSubClient client(espClient);

void setup() {
  Serial.begin(115200);
  pinMode(GAS_SENSOR_PIN, INPUT);

  // Connect to WiFi
```

```
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("Connecting to WiFi...");
  }
  Serial.println("WiFi connected");

  // Setup MQTT server
  client.setServer(mqtt_server, mqtt_port);
}

void reconnect() {
  while (!client.connected()) {
    Serial.println("Connecting to MQTT...");
    if (client.connect(client_id)) {
      Serial.println("Connected to MQTT broker");
    } else {
      Serial.print("Failed, rc=");
      Serial.print(client.state());
      Serial.println(" Retrying in 5 seconds...");
      delay(5000);
    }
  }
}

void loop() {
  if (!client.connected()) {
    reconnect();
  }
  client.loop();

  // Read gas sensor value
  int gasValue = analogRead(GAS_SENSOR_PIN);

  // Create message string
  String payload = "Gas Sensor Value: " + String(gasValue);

  // Publish and print
  client.publish(topic, payload.c_str());
  Serial.println(payload);

  delay(1000); // Wait before next reading
}
```

9.Develop a program using the Arduino IDE to interface a Smoke/Gas sensor with the ESP32 to measure the quality of air. Transmit the data to the ThingSpeak cloud platform and publish it.

```
#include <WiFi.h>

// Gas sensor pin
#define GAS_SENSOR_PIN 36  // GPIO36 = VP (ADC1_CH0)

// WiFi credentials
const char* ssid = "Your_WiFi_SSID";
const char* password = "Your_WiFi_Password";
```

```
// ThingSpeak API settings
const char* host = "api.thingspeak.com";
String apiKey = "Your_ThingSpeak_API_Key";

WiFiClient client;

void setup() {
  Serial.begin(115200);
  pinMode(GAS_SENSOR_PIN, INPUT);

  // Connect to WiFi
  WiFi.begin(ssid, password);
  Serial.print("Connecting to Wi-Fi");
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.print(".");
  }
  Serial.println("\nWi-Fi connected");
}

void loop() {
  // Read gas sensor value
  int gasValue = analogRead(GAS_SENSOR_PIN);
  Serial.print("Gas Sensor Value: ");
  Serial.println(gasValue);

  // Send data to ThingSpeak
  if (client.connect(host, 80)) {
    String url = "/update?api_key=" + apiKey + "&field1=" + String(gasValue);
    client.print(String("GET ") + url + " HTTP/1.1\r\n" +
                 "Host: " + host + "\r\n" +
                 "Connection: close\r\n\r\n");
    client.stop();
  }

  delay(15000);  // ThingSpeak requires at least 15 seconds delay between updates
}
```

10.Develop a Python program to interface an Ultrasonic sensor with a Raspberry Pi to measure the distance of an object. Implement the MQTT protocol to transmit the data to MQTTBox and display the readings in real time.

```
import RPi.GPIO as GPIO
import time
import paho.mqtt.client as mqtt

# GPIO Pins
TRIG = 23
ECHO = 24

# MQTT Configuration
MQTT_BROKER = "mqtt.eclipseprojects.io"  # or your broker IP
MQTT_PORT = 1883
MQTT_TOPIC = "raspberrypi/ultrasonic"
```

```
# GPIO Setup
GPIO.setmode(GPIO.BCM)
GPIO.setup(TRIG, GPIO.OUT)
GPIO.setup(ECHO, GPIO.IN)

# MQTT Client Setup
client = mqtt.Client()
client.connect(MQTT_BROKER, MQTT_PORT, 60)

def get_distance():
    GPIO.output(TRIG, False)
    time.sleep(0.5)

    GPIO.output(TRIG, True)
    time.sleep(0.00001)
    GPIO.output(TRIG, False)

    while GPIO.input(ECHO) == 0:
        pulse_start = time.time()

    while GPIO.input(ECHO) == 1:
        pulse_end = time.time()

    pulse_duration = pulse_end - pulse_start
    distance = round(pulse_duration * 17150, 2)  # in cm

    return distance

try:
    while True:
        dist = get_distance()
        print(f"Distance: {dist} cm")
        client.publish(MQTT_TOPIC, f"{dist}")
        time.sleep(2)
except KeyboardInterrupt:
    print("Stopped by User")
    GPIO.cleanup()
```

11.Develop a Python program to interface a DHT11 sensor with a Raspberry Pi to measure temperature and humidity and display the data on an MQTT App.

```
import time
import paho.mqtt.client as mqtt
from gpiozero import DigitalInputDevice
import adafruit_dht
import board

# MQTT Broker Configuration
MQTT_BROKER = "broker.emqx.io"  # Change to your MQTT broker IP or URL
MQTT_PORT = 1883
MQTT_TOPIC = "raspberrypi5/dht11"

# Initialize MQTT Client
client = mqtt.Client()
client.connect(MQTT_BROKER, MQTT_PORT, 60)
```

```
SERIAL_PORT = "/dev/ttyAMA0"

# Define DHT11 Sensor (GPIO 4)
dht_device = adafruit_dht.DHT11(board.D23)

# Function to Read Data & Publish to MQTT
def read_and_send_data():
    try:
        temperature = dht_device.temperature
        humidity = dht_device.humidity

        if temperature is not None and humidity is not None:
            payload = f'{{"temperature": {temperature}, "humidity": {humidity}}}'
            client.publish(MQTT_TOPIC, payload)
            print("Publishing:", payload)
        else:
            print("Failed to get sensor data. Retrying...")

    except Exception as e:
        print("Error:", str(e))

# Main Loop
try:
    while True:
        read_and_send_data()
        time.sleep(5)  # Send data every 5 seconds

except KeyboardInterrupt:
    print("Script stopped by user")
    dht_device.exit()
    client.disconnect()
```