Experiment No 2

AIM: To simulate sensor data generation and collection.

Objective: To simulate sensor data generation and collection for IoT applications. We will:

1. Generate random sensor data for parameters like temperature and humidity.
2. Collect the generated data in real-time.
3. Save the data to a file for further analysis.

Tools Used: Google Colab, Python libraries (random)

Theory:
The Internet of Things (IoT) is a transformative concept, involving a network of interconnected devices that communicate and exchange data to enable smart environments. Sensor data generation is a fundamental aspect of IoT systems, where physical parameters such as temperature, humidity, and pressure are measured by sensors and transmitted to processing units or the cloud. This data is essential for analysis, decision-making, and actuation in applications ranging from home automation to industrial monitoring.

Simulating sensor data enables learners to replicate the behavior of IoT devices without the need for physical sensors. Using Python, we can generate pseudo-random data mimicking real-world sensor readings, which helps students understand IoT data streams and their processing. This approach is highly advantageous for prototyping and learning as it eliminates the dependency on hardware.

The use of the `random` module in Python allows us to generate realistic, variable readings within a defined range. For example, temperature readings can be emulated as floating-point values within a typical range for environmental conditions. Simulated data is stored and processed for tasks such as visualization, statistical analysis, or transmission over IoT communication protocols like MQTT or HTTP.

This experiment demonstrates the basics of creating a simulated IoT environment, emphasizing the importance of data generation, collection, and understanding real-world constraints like data variability and trends.

Program Code:

```
import random
import time
import json
# Function to simulate sensor data generation

def generate_sensor_data():

   """
   Simulates data for temperature and humidity sensors.
   Returns a dictionary containing the sensor readings.
   """
   temperature = round(random.uniform(20.0, 30.0), 2)  # Random
temperature between 20°C and 30°C

   humidity = round(random.uniform(40.0, 60.0), 2)     # Random humidity
between 40% and 60%

   timestamp = time.strftime("%Y-%m-%d %H:%M:%S")      # Current timestamp

   return {

       "timestamp": timestamp,

       "temperature": temperature,

       "humidity": humidity

   }

# Function to collect sensor data and save to a file

def collect_sensor_data(duration, interval, output_file):

   """

   Collects sensor data for a given duration and interval, and saves it to
a file.

   Args:

   - duration: Total duration in seconds for data collection.
   - interval: Time interval (in seconds) between data points.
   - output_file: File to save the collected data in JSON format.
   """
   collected_data = []
```

```
    start_time = time.time()
    print("Starting sensor data collection...\n")
    while time.time()-start_time < duration:
        # Generate sensor data
        sensor_data = generate_sensor_data()

        # Display the generated data
        print(f"Collected Data: {sensor_data}")
                # Append to the collection
        collected_data.append(sensor_data)
                # Wait for the specified interval
        time.sleep(interval)
    # Save the data to a file
    with open(output_file, 'w') as file:
        json.dump(collected_data, file, indent=4)
        print(f"\nSensor data collection completed. Data saved to
{output_file}")
# Main program
if __name__ == "__main__":
    # Define simulation parameters
    duration = 30  # Collect data for 30 seconds

    interval = 5   # Collect data every 5 seconds

    output_file = "sensor_data.json"

    # Start the data collection

    collect_sensor_data(duration, interval, output_file)
```

Explanation and Logic of Code

1. **Sensor Data Simulation**:
    - Randomly generates values for **temperature** (20.0–30.0 °C) and **humidity** (40.0–60.0%) to mimic real-world sensor readings.
    - Includes a timestamp for real-time tracking.
2. **Data Collection**:
    - Runs a loop for a specified duration.
    - Collects data at defined intervals using `time.sleep(interval)` to simulate real-time behavior.
    - Accumulates the data in a list for structured storage.
3. **Data Storage**:
    - At the end of data collection, the sensor readings are saved in a **JSON file** for analysis and visualization.
4. **Dynamic Behavior**:

- The program can adapt to different durations and intervals, making it versatile for various simulation needs.

Message Flow

Below is the message flow describing the interaction between system components:

Flowchart:

Start Program
→ Initialize Parameters (duration, interval, output file).
→ Begin Data Collection Loop:

- Generate sensor data.
- Append data to collection.
- Print and store data. → End Loop After Duration.
  → Save Data to JSON File.
  → Stop Program.

Observation Table:

Timestamp	Temperature (°C)	Humidity (%)	Remarks
2025-02-01 06:01:22	26.35	46.17	Moderate temperature.
2025-02-01 06:01:27	20.7	52.77	Temperature Drops.
2025-02-01 06:01:32	29.09	43.33	Highest Temperature Recorded
2025-02-01 06:01:37	20.69	40.25	Lowest Temperature
2025-02-01 06:01:42	28.29	47.21	Temperature Rise Again
2025-02-01 06:01:47	20.87	57.78	Temperature Decrease Again

Outcome:

```
Starting sensor data collection...

Collected Data: {'timestamp': '2025-02-01 06:01:22', 'temperature': 26.35, 'humidity': 46.17}
Collected Data: {'timestamp': '2025-02-01 06:01:27', 'temperature': 20.7, 'humidity': 52.77}
Collected Data: {'timestamp': '2025-02-01 06:01:32', 'temperature': 29.09, 'humidity': 43.33}
Collected Data: {'timestamp': '2025-02-01 06:01:37', 'temperature': 20.69, 'humidity': 40.25}
Collected Data: {'timestamp': '2025-02-01 06:01:42', 'temperature': 28.29, 'humidity': 47.21}
Collected Data: {'timestamp': '2025-02-01 06:01:47', 'temperature': 20.87, 'humidity': 57.78}

Sensor data collection completed. Data saved to sensor_data.json
```

Conclusion:

Simulating sensor data generation and collection is a foundational exercise in understanding IoT systems. This practical approach helps to grasp the concepts of real-time data acquisition, storage, and processing. By extending the simulation with tasks like visualization, anomaly detection, or cloud integration, We can bridge the gap between theory and real-world applications, preparing them for advanced IoT challenges

Homework Assigned:

Extend the Simulation

- **Task**: Add another sensor, such as a pressure and light intensity sensor, and simulate its data along with temperature and humidity.
- **Hint**: Use the same `random.uniform()` function to generate random values for the new sensor.

Program Code :

```
import random
import time
import datetime

def generate_sensor_data():
  # Simulate temperature data between 20 and 30 degrees Celsius
  temperature = random.uniform(20, 30)

  # Simulate humidity data between 40% and 60%
  humidity = random.uniform(40, 60)

  # Simulate pressure data between 980 and 1030 hPa
  pressure = random.uniform(980, 1030)

  # Simulate light intensity data between 0 and 1000 lux
  light_intensity = random.uniform(0, 1000)

  timestamp = datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S")
  return timestamp, temperature, humidity, pressure, light_intensity

# Example usage: Generate and print sensor data every second for 5
iterations

for _ in range(5):
```

```
  timestamp, temperature, humidity, pressure, light_intensity =
generate_sensor_data()
  print(f"Timestamp: {timestamp}, Temperature: {temperature:.2f}°C,
Humidity: {humidity:.2f}%, Pressure: {pressure:.2f} hPa, Light Intensity:
{light_intensity:.2f} lux")
  time.sleep(1)
```

Observation Table :

Timestamps	Temperature	Humidity	Pressure	Light Intensity	Remarks
2025-02-01 06:20:18	20.60°C	57.18%	1012.22 hPa	766.93 lux	Low Temperature
2025-02-01 06:20:19	26.35°C	55.48%	988.97 hPa	897.92 lux	Moderate Inc in Temp.
2025-02-01 06:20:20	25.82°C	44.90%	1010.55 hPa	894.88 lux	Stable & Normal Temp.
2025-02-01 06:20:21	29.68°C	44.71%	982.10 hPa	634.59 lux	Highest Temp. Recorded
2025-02-01 06:20:22	25.42°C	50.76%	1010.49 hPa	288.27 lux	Slightly Decrease in Temp

Outcome :

```
Timestamp: 2025-02-01 06:20:18, Temperature: 20.60°C, Humidity: 57.18%, Pressure: 1012.22 hPa, Light Intensity: 766.93 lux
Timestamp: 2025-02-01 06:20:19, Temperature: 26.35°C, Humidity: 55.48%, Pressure: 988.97 hPa, Light Intensity: 897.92 lux
Timestamp: 2025-02-01 06:20:20, Temperature: 25.82°C, Humidity: 44.90%, Pressure: 1010.55 hPa, Light Intensity: 894.88 lux
Timestamp: 2025-02-01 06:20:21, Temperature: 29.68°C, Humidity: 44.71%, Pressure: 982.10 hPa, Light Intensity: 634.59 lux
Timestamp: 2025-02-01 06:20:22, Temperature: 25.42°C, Humidity: 50.76%, Pressure: 1010.49 hPa, Light Intensity: 288.27 lux
```

Conclusion : This experiment successfully demonstrates the use of MQTT for real-time sensor data publishing and subscribing. The system simulates temperature, pressure, humidity, and light intensity values and transmits them to an MQTT broker.