5G USE CASE LAB

CS 24502

M **Tech II Semester**

Lab Manual

(2024-25)

Department of Electronics and Communication Engineering

Maulana Azad National Institute of Technology Bhopal-462003

5G USE CASE LAB

LAB MANUAL

PROGRAM : M.TECH

SPECIALIZATION : ELECTRONICS AND COMMUNICATION ENGINEERING

SEMESTER : 2ND

COURSE CODE : CS 24502

Prepared by Dr. Lalita Gupta (Laboratory Coordinator)

Maulana Azad National Institute of Technology Bhopal-462003

Department of Electronics and Communication Engineering
5G Use Case Lab (CS-24502)
List of Experiments

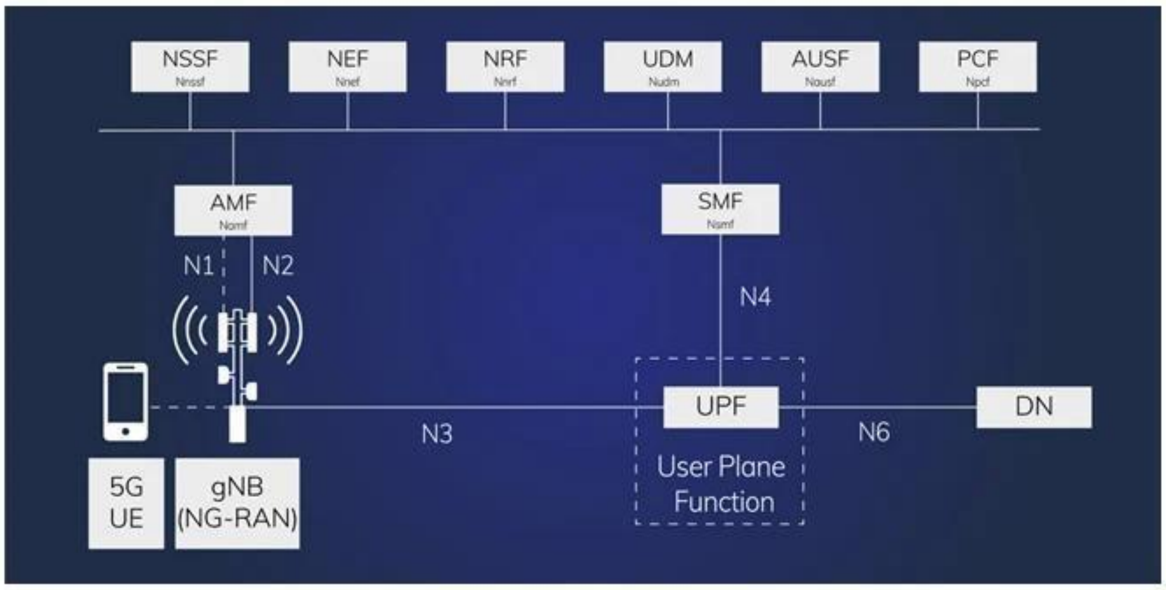
SR NO	EXPERIMENTS NAME
1	UNDERSTAND THE ARCHITECTURE OF 5G CORE NETWORK &ANALYZE ITS KEY NETWORK FUNCTIONS.
2	UNDERSTAND THE ARCHITECTURE OF 5G RADIO ACCESS NETWORK.
3	EXAMINE THE ROLE OF 5G ROUTERS & FIREWALLS.
4	MEASURE THE OUTPUT OF TEMPERATURE AND HUMIDITY SENSOR USING IOT GATEWAY.
5	MEASURE THE OUTPUT OF SOIL SENSOR (NPK) USING IOT GATEWAY
6	MEASURE THE OUTPUT OF LIGHT SENSOR USING IOT GATEWAY
7	MEASURE THE OUTPUT OF WATER SENSOR USING IOT GATEWAY.
8	EXAMINE THE WORKING OF 5G DRONE.
9	COMPARE THE PERFORMANCE OF AR-VR OVER 4G & 5G NETWORKS.
10	UNDERSTAND THE WORKING OF 5G EVOLUTION MODULE

EXPERIMENT NO- 1

Aim: Understand the architecture of 5G core network & analyze its key network functions.

Theory :

The 5G Core (5GC) architecture represents a fundamental shift from previous generations of mobile network architectures, offering improved flexibility, scalability, and functionality. It serves as the backbone of the 5G network and is responsible for managing traffic, mobility, security, and more. The 5G Core is designed to handle the increased demands of 5G, supporting a variety of use cases such as enhanced mobile broadband (eMBB), ultra-reliable low-latency communications (URLLC), and massive machine-type communications (mMTC).



5G Core Network Architecture

Key Components of 5G Core Architecture:

1. **Access and Mobility Management Function (AMF)**:
   - o Responsible for handling connection management, mobility management, and authentication of users in the 5G network.
   - o It manages the interaction with the User Equipment (UE) and ensures seamless handovers between different cells.
2. **Session Management Function (SMF)**:
   - o Manages the session life cycle, including the establishment, modification, and release of sessions. o It interacts with the User Plane Function (UPF) to configure data paths for user traffic.
3. **User Plane Function (UPF)**:
   - o Handles user data traffic (user plane). o Responsible for forwarding data between the user equipment and the data network.

   - It also manages traffic routing, packet filtering, and Quality of Service (QoS) enforcement.
4. **Authentication Server Function (AUSF)**:
   - Provides authentication services for the UE during registration and access requests.
   - Works alongside the AMF and interacts with the Subscription Permanent Identifier (SUPI) stored in the Home Subscriber Server (HSS).
5. **Unified Data Management (UDM)**:
   - Manages user-related information, such as subscription data, authentication data, and QoS profiles.
   - The UDM is a replacement for the Home Subscriber Server (HSS) in 4G.
6. **Policy Control Function (PCF)**:
   - Responsible for enforcing policies related to traffic, QoS, and charging. o Works in conjunction with the SMF and AMF to ensure that user sessions comply with the required policies.
7. **Network Slice Selection Function (NSSF)**:
   - Ensures the correct network slice is chosen based on the user's needs or use case.
   - It facilitates the selection of a network slice for specific services (e.g., for eMBB, URLLC, or mMTC).
8. **Service Communication Proxy (SCP)**:
   - It is used to handle service communication between different core network elements, such as the AMF, SMF, PCF, etc.
9. **Application Function (AF)**:
   - Works with the PCF to provide application-specific services. o It could be an application that requires QoS or other specific configurations in the network.

5G Core Network Functions:

The 5G Core is designed to be flexible and modular, based on a service-based architecture (SBA). The core network functions are defined as services that communicate over a servicebased interface (SBI). Key functions in the architecture include:

- **SBI (Service-Based Interface)**: Allows network functions (NFs) to interact with each other, providing the flexibility to scale and adapt to various services.
- **Service Functions**: These are the individual logical functions (like AMF, SMF, UPF, etc.) that can be virtualized and deployed independently for a more dynamic and efficient network.

Key Features of 5G Core:

- **Cloud-Native**: The 5G Core is designed to be deployed on cloud infrastructure, allowing for dynamic scaling and flexibility.
- **Service-Based Architecture (SBA)**: Instead of using traditional interfaces (like in 4G), SBA allows for more modular and dynamic communication between core network functions.

- **Network Slicing**: The core supports network slicing, where virtualized, isolated portions of the network are allocated for different types of services.
- **Edge Computing**: The 5G Core integrates edge computing to reduce latency by processing data closer to the user.
- **Support for IoT**: The 5G Core is designed to handle the massive number of IoT devices and their specific requirements, such as low power consumption and largescale connectivity.

5G Core Deployment:

The 5G Core can be deployed in different ways depending on the network operator's needs:

- **Standalone (SA) Deployment**: The core network operates independently, and 5G services are provided without relying on the 4G LTE core.
- **Non-Standalone (NSA) Deployment**: The 5G RAN works alongside the 4G LTE core for services like enhanced mobile broadband (eMBB), but the 5G Core is not yet fully deployed.

Key Benefits of 5G Core:

- **Flexibility**: The modular and cloud-native design allows operators to introduce new services and scale up or down based on demand.
- **High Performance**: The architecture is optimized for ultra-low latency, massive device connectivity, and high-speed data transfer.
- **Security**: The 5G Core introduces enhanced security features like end-to-end encryption, network slicing, and stronger user authentication.

Conclusion: the 5G Core is a highly sophisticated and flexible network architecture designed to support a diverse range of services and use cases, all while ensuring scalability, performance, and security for the next generation of wireless communication.

EXPERIMENT NO : 2

Aim: Understand the architecture of Radio Access Network.

Theory:

Studying the architecture of a **5G Radio Access Network (RAN)** through an experiment would involve understanding its components, functions, and performance characteristics. 5G RAN is essential for enabling 5G services, offering high throughput, low latency, and increased capacity compared to previous generations (like 4G LTE).

Here's an outline for an experiment that you can design to study and analyze the architecture of 5G RAN:

1. Objective of the Experiment:

- To understand the structure and working of 5G RAN.
- To analyze the various components, protocols, and interfaces involved in 5G RAN.
- To measure key performance indicators (KPIs) like latency, throughput, and mobility.

2. Key Components to Study:

The architecture of 5G RAN is divided into several main components:

- **User Equipment (UE):** The devices that connect to the 5G network (smartphones, IoT devices, etc.).
- **Next-Generation NodeB (gNB):** The base stations that provide connectivity between the user equipment and the 5G network.
- **Centralized Unit (CU):** Handles the control plane functionalities (e.g., signaling, resource management).
- **Distributed Unit (DU):** Handles the user plane, i.e., data traffic transmission.
- **Centralized RAN (C-RAN):** A virtualized and centralized base station architecture.
- **Radio Access Network (RAN):** The network that connects the gNB, CU, and DU.

3. Experiment Setup:

A. Software Tools:

- **NS3 (Network Simulator 3):** A popular network simulator for testing 5G networks.
- **OpenAirInterface (OAI):** Open-source software stack to simulate 5G network elements like eNB and gNB.
- **MATLAB or Simulink:** For modeling signal processing algorithms or channel models.
- **Omnet++ (optional):** Another network simulator that can be used to model and simulate 5G RAN.

B. Hardware Setup:

If you're planning a physical experiment, you could use:

- **5G capable devices**: Test UE like smartphones or IoT devices with 5G modems.
- **5G Base Station Equipment**: gNB or small cell base stations (this requires investment or partnership with service providers or research institutions).

4. Key Parameters to Monitor:

- **Throughput:** Measure the data rates at different points (UE to gNB, DU to CU, etc.).
- **Latency:** Measure round-trip time (RTT) and packet delays.
- **Signal-to-Noise Ratio (SNR):** Assess the quality of the radio channel between the UE and the gNB.
- **Coverage Area:** Study the cell coverage and the impact of gNB placement and density on coverage.
- **Handovers:** Analyze the mobility performance of 5G networks, including intra-RAN handovers and inter-RAN handovers.
- **QoS (Quality of Service):** Monitor service delivery based on different traffic types (e.g., URLLC, eMBB, mMTC).

5. Experiment Procedure:

A. Simulating 5G RAN Using NS3 or OAI:

1. **Setup Simulation:**
   - Define the network topology including gNB, CU, DU, and UE. ○ Set up the parameters (carrier frequencies, bandwidth, channel models, etc.).
   - Define the traffic types (e.g., web browsing, VoIP, video streaming).
2. **Run Tests:**
   - Measure the throughput and latency under different network conditions (e.g., different SNR, mobility scenarios).
   - Simulate various use cases like high data traffic (eMBB), low latency (URLLC), and massive IoT connections (mMTC).
3. **Collect Data:**
   - Monitor key metrics during the simulation (e.g., signal strength, data rate, delay, jitter). ○ Test different UE behaviors (e.g., moving through the coverage area). *B. Field Testing (if hardware is available):*

1. **Deploy a small test network** using commercial 5G hardware or use a testbed offered by institutions like OAI or testbed providers.
2. **Place multiple UEs** in different geographic locations to measure coverage and network behavior.
3. **Measure performance** using tools that can report network metrics such as drive tests for coverage or apps like iPerf for throughput.
4. **Evaluate Handover Scenarios:** Test mobility scenarios by having UEs move across different cells (from one gNB to another).

6. Data Analysis:

- **Graphical Representation:** Plot throughput vs. distance, latency vs. SNR, handover success rate, etc.

- **Compare with 4G RAN:** Benchmark the results against 4G LTE to highlight the improvements 5G offers in terms of latency, throughput, and capacity.
- **Examine Scalability:** Evaluate how the 5G RAN architecture can scale to handle large numbers of users or devices.

7. Conclusion:

- Summarize the findings related to performance improvements (e.g., reduced latency, better throughput).
- Discuss the challenges in implementing 5G RAN, such as network deployment complexity, power consumption, and spectrum management.
- Highlight the future potential of 5G networks in supporting massive IoT, ultra-reliable lowlatency communication, and high-speed internet services.

EXPERIMENT NO: 3

Aim: Examine the role of 5G Routers and Firewalls.

Theory:

To conduct an experiment studying 5G routers and firewalls, you can focus on testing their performance, security features, and how they handle the traffic flow in a 5G network environment. Here's an outline of what such an experiment could involve:

1. Objective of the Study:

- **Evaluate performance:** Assess the speed, latency, and overall performance of 5G routers.
- **Study security features:** Examine the security measures implemented in firewalls on 5G routers.
- **Traffic analysis:** Observe how 5G routers handle different types of network traffic and ensure that the firewall rules are effectively applied.

2. Equipment and Setup Required:

- **5G router(s):** Obtain 5G-enabled routers from different manufacturers (e.g., Cisco, Huawei, Netgear).
- **Firewall devices or software:** Look for firewalls integrated into the router or external firewall devices.
- **Test devices:** Computers or smartphones with 5G connectivity.
- **Network traffic generator:** Tools like Iperf or Wireshark to simulate traffic and measure network performance.
- **Traffic types:** A variety of traffic, including HTTP, video streaming, gaming, VoIP, and file transfers.
- **Measurement tools:** Tools like ping, traceroute, and bandwidth tests.

3. Experiment Phases:

A. Performance Testing

1. **Latency Measurement:**
    - Measure round-trip time (RTT) on the network.
    - Perform tests with minimal load and under heavy load conditions to see how latency changes.
2. **Bandwidth Testing:**
    - Use a tool like Iperf to measure the download and upload speeds under various conditions (e.g., high traffic vs. low traffic).
3. **Throughput Testing:**
    - Measure the maximum throughput (data transmission speed) a 5G router can sustain under different network load scenarios.
4. **Test on Different Traffic Types:**
    - Test with gaming, video streaming, file transfers, and VoIP traffic to see how well the router manages different types of traffic.

B. Security and Firewall Testing

1. **Firewall Configuration:**
    - Review the default security settings of the router/firewall.
    - Test custom configurations such as IP filtering, port blocking, and access control lists (ACLs).
2. **Penetration Testing:**
    - Attempt penetration testing or vulnerability scanning to identify potential weaknesses in the firewall or router security.
3. **Traffic Filtering:**
    - Verify how the firewall handles malicious traffic, like DDoS attacks or known attack signatures.
4. **Firewall Performance Under Load:**
    - Test how the firewall performs under heavy traffic conditions to ensure it doesn't degrade the network's performance while securing it.
5. **Intrusion Detection/Prevention:**
    - Check if the router's firewall has any Intrusion Detection or Prevention Systems (IDPS) and how well they work when under attack. *C. Real-World Testing:*

1. **Mobility Testing:**
    - Test the router's performance and security while moving between different 5G cells or network conditions (e.g., from urban to rural areas).
2. **QoS (Quality of Service) Testing:** o Analyze how the router prioritizes certain types of traffic (e.g., voice or video calls) over others.
3. **Simulated Network Failures:**
    - Simulate network outages, drops, or congestion and test how the router/firewall recovers or adapts.

4. Data Collection & Analysis:

- Record all performance metrics such as latency, throughput, packet loss, and CPU utilization of the router during tests.
- Capture security logs during firewall testing to see if there were any attempted intrusions or issues.
- Analyze how well the firewall rules are applied and whether they are bypassed or incorrectly configured under heavy load.
- Compare performance between different 5G routers and firewalls.

5. Expected Outcomes:

- **Performance:**
    - You should observe lower latency and higher throughput compared to previous wireless standards (e.g., 4G), especially for high-demand tasks.
    - The router should handle congestion without significantly affecting performance, but this will depend on the specific manufacturer and router model.
- **Security:**
    - A well-configured 5G router firewall should prevent unauthorized access, block malicious traffic, and be able to scale under load without reducing security.
- **Scalability:**

    - 5G routers should be scalable, able to handle a large number of devices simultaneously without major performance degradation.

6. Conclusion:

- Evaluate the overall effectiveness of the firewall in protecting the network while still maintaining high network performance.
- Consider areas for improvement in both router performance and firewall security.
- Provide recommendations for network administrators on how to best configure and maintain 5G routers and firewalls to ensure optimal performance and security.

This experiment can provide valuable insights into the capabilities and limitations of 5G routers and firewalls, helping to improve network design and security in 5G environments.

EXPERIMENT NO-4

Aim: Measure the output of temperature and humidity sensor using IoT gatway.

Theory:

Studying the practical of a temperature and humidity sensor using an IoT (Internet of Things) gateway involves setting up a sensor to measure environmental conditions, collecting data from the sensor, and transmitting that data to a centralized system via an IoT gateway. This kind of experiment is great for understanding how IoT works, especially in environments like smart homes, weather stations, and industrial monitoring.

Here's a general outline of what you'd need to do for the experiment:

Components Required:

1. **Temperature and Humidity Sensor**: Common options include:
   - **DHT11** or **DHT22** for basic use.
   - **BME280** for more advanced use with more accurate readings.
2. **IoT Gateway**: This could be a microcontroller or a development board like:
   - **Raspberry Pi** (with WiFi or Ethernet capabilities).
   - **Arduino** with a Wi-Fi module like **ESP8266** or **ESP32**.
3. **Software**:
   - **Arduino IDE** for writing the code (for Arduino-based setups).
   - **Node-RED** or **MQTT Broker** (for data management and visualization).
4. **Cloud Platform** (Optional for remote monitoring):
   - **ThingSpeak**, **Blynk**, or **Google Cloud IoT**.

Step-by-Step Procedure:

1. Set Up the Sensor:

- Connect the temperature and humidity sensor to the microcontroller (like an Arduino or Raspberry Pi).
- For example, for a **DHT22** sensor:
  - Connect the **VCC** to the 5V pin.
  - Connect **GND** to the GND pin.
  - Connect **DATA** to a digital input pin (e.g., pin 2 on Arduino).

2. Write Code to Read Sensor Data:

- In Arduino IDE (for an Arduino setup), use the **DHT sensor library** to interact with the sensor.
- The basic code to read temperature and humidity might look like this:

```
cpp
Copy
#include <DHT.h>

#define DHTPIN 2        // Pin connected to the sensor's data pin
```

```
#define DHTTYPE DHT22    // Define the sensor type (DHT11 or DHT22)
DHT dht(DHTPIN, DHTTYPE);

void   setup()   {
Serial.begin(9600);
dht.begin();      }
void loop() {
  float temperature = dht.readTemperature();
float humidity = dht.readHumidity();
   if (isnan(temperature) || isnan(humidity)) {
Serial.println("Failed to read from DHT sensor!");     return;
  }

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

  delay(2000); // Wait for 2 seconds before reading again
}
```

3. Set Up IoT Gateway:

- If you're using an **ESP32/ESP8266**, you can use **Wi-Fi** to transmit data.
- For **Raspberry Pi**, you can use Python and libraries like **MQTT** or **HTTP** to send data.

4. Sending Data to Cloud or Gateway:

- If using **MQTT** (a lightweight protocol for IoT), use a library like **PubSubClient** to publish the sensor data to an MQTT broker.
- For **Raspberry Pi**, you can use Python to publish the data to a cloud platform like **ThingSpeak**.

Example MQTT publishing code (for Arduino):

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

const char* ssid = "your_wifi_ssid"; const
char* password = "your_wifi_password"; const
char* mqtt_server = "mqtt_server_address";
WiFiClient espClient;
PubSubClient client(espClient);

void setup() {
  Serial.begin(9600);   WiFi.begin(ssid,
password);   while (WiFi.status() != WL_CONNECTED)
{     delay(1000);     Serial.println("Connecting
to WiFi...");
  }
  Serial.println("Connected to WiFi");

  client.setServer(mqtt_server, 1883); // Default MQTT port
}  void
loop() {
  if (!client.connected()) {
reconnect();
  }   client.loop();
float temperature =
dht.readTemperature();   float humidity =
dht.readHumidity();

  if (!isnan(temperature) && !isnan(humidity)) {
    String payload = "{\"temperature\":" + String(temperature) +
",\"humidity\":" + String(humidity) + "}";
client.publish("sensor/data", payload.c_str());
  }
  delay(2000); // Wait for 2 seconds before sending new data
}  void reconnect() {
while
(!client.connected()) {     if
(client.connect("ESP32Client")) {
client.subscribe("sensor/data");
    } else {
delay(5000);
    }
  }
}
```

5. Visualize Data:

- Once data is sent to the cloud (like **ThingSpeak**), you can visualize it on a dashboard in realtime.
- You can also use **Blynk** or **Node-RED** for more interactive visualization.

6. Troubleshooting and Calibration:

- Ensure the sensor is correctly connected.
- Check for any issues in the sensor's data output (e.g., incorrect voltage or wiring).
- Use serial debugging to track values and test the data.

Summary of the Experiment:

- **Objective**: Measure temperature and humidity using a sensor and send the data over an IoT gateway to a cloud platform or local server.
- **Outcome**: By the end of this experiment, you should understand how to set up sensors, connect them to an IoT device, and send data to the cloud or a server. You will also learn how to visualize and monitor the data remotely.

EXPERIMENT NO-5

Aim: Measure the output of soil sensor (NPK) using IoT gateway.

Theory:

To study the practical of a soil sensor (NPK) using an IoT gateway, you would typically be working with a system that monitors soil conditions and sends the data to a cloud or remote server for analysis. The NPK (Nitrogen, Phosphorus, and Potassium) sensor is a device used to measure the nutrient levels in the soil, which are crucial for plant growth.

Here's how you can go about the experiment and set it up:

Components Needed:

1. **NPK Sensor**: A sensor to measure the concentration of Nitrogen, Phosphorus, and Potassium in the soil. o Common types: Analog or Digital sensors.
2. **Microcontroller**: A device to process the data from the NPK sensor.
    - Common microcontrollers: Arduino, ESP8266, ESP32, Raspberry Pi, etc.
3. **IoT Gateway**: An IoT gateway is used to send data from the sensor to the cloud or remote server.
    - Example: ESP32, which has Wi-Fi or Bluetooth connectivity.
4. **Wireless Communication Module**: For communication with the IoT gateway (usually Wi-Fi or LoRa).
    - Example: Wi-Fi (ESP32), GSM, Zigbee, or LoRa.
5. **Cloud Service/Database**: A cloud platform or server to store and visualize the sensor data.
    - Example: Thingspeak, AWS IoT, Blynk, Firebase, etc.
6. **Power Supply**: To power the microcontroller and sensors.
    - Example: Batteries, USB power, or solar power (for remote applications).
7. **Software/Programming**: Code to read sensor data and communicate with the IoT gateway.

Steps to Conduct the Experiment:

1. Connect the NPK Sensor to the Microcontroller:

- Connect the NPK sensor's output pins (e.g., analog or digital) to the appropriate input pins on your microcontroller.
- For analog sensors, connect the analog output to an analog input pin on the microcontroller.
- For digital sensors, connect the output to a digital input pin.

2. Program the Microcontroller:

- If you're using Arduino, you will write a code to read the data from the NPK sensor.
- You may need to use libraries depending on the sensor you're using.
- Example for Arduino:

```
cpp
```

```
Copy
int sensorPin = A0; // Analog pin connected to the sensor
int sensorValue = 0;

void setup() {
  Serial.begin(9600); // Start serial communication
}  void
loop() {
  sensorValue = analogRead(sensorPin); // Read the sensor value
Serial.println(sensorValue); // Print the sensor value to the serial
monitor
  delay(1000); // Delay for 1 second
}
```

 This basic code will output sensor values to the serial monitor.

3. Set Up the IoT Gateway:

- If you're using an ESP32 or another Wi-Fi-enabled microcontroller, you can configure it to send data to a cloud platform.
- Install necessary libraries like `WiFi.h` and `ThingSpeak.h` (or any other IoT platform libraries).
- Example code to send data to ThingSpeak (for ESP32):

```
cpp
Copy
#include <WiFi.h>
#include "ThingSpeak.h"
 const char *ssid = "Your_SSID"; const
char *password = "Your_PASSWORD";  WiFiClient
client;
 unsigned long myChannelNumber = YOUR_CHANNEL_ID;
const char * myWriteAPIKey =
"YOUR_WRITE_API_KEY";

int sensorPin = A0; // Analog pin connected to the sensor

void setup() {
  Serial.begin(115200);
WiFi.begin(ssid, password);   while
(WiFi.status() != WL_CONNECTED) {
delay(1000);
    Serial.println("Connecting to WiFi...");
  }
  ThingSpeak.begin(client); // Initialize
ThingSpeak }  void loop() {   int
sensorValue = analogRead(sensorPin);
  ThingSpeak.setField(1, sensorValue); // Write the sensor data to
field 1
  ThingSpeak.writeFields(myChannelNumber, myWriteAPIKey);
delay(2000); // Wait for 2 seconds before next reading }
```

 Replace `YOUR_CHANNEL_ID` and `YOUR_WRITE_API_KEY` with your ThingSpeak credentials.

4. Set Up the Cloud Service:

- Sign up for a cloud platform like ThingSpeak, Blynk, or Firebase.
- Create a new project or channel, and configure it to receive data from your sensor.
- For example, on ThingSpeak, you will create a channel and use the API key to send the data to your cloud account.

5. Analyze Data:

- Once the data is being sent to the cloud, you can visualize the NPK levels (Nitrogen, Phosphorus, and Potassium) over time.
- You can use charts, graphs, or alerts based on predefined thresholds for optimal soil nutrient levels.

Conclusion:

By using an NPK sensor and an IoT gateway, you can monitor soil health remotely and analyze the data over time. This setup provides real-time insights into soil conditions and can be extended for agricultural applications, ensuring that plants get the right nutrients for optimal growth.

EXPERIMENT NO-6

Aim: Measure the output of Light sensor using IoT gateway **Theory:**

To study the practical of a light sensor using an IoT gateway, you'll typically be using an IoT platform to collect and monitor data from a light sensor, transmitting that data to a central system (IoT gateway), and possibly triggering actions based on light levels. Below is a general outline of the experiment setup and procedure:

Objective:

- Understand how a light sensor works and how it can be integrated with an IoT gateway for remote monitoring and automation.

Components Required:

1. **Light Sensor (LDR - Light Dependent Resistor or other types)**: A sensor that detects the intensity of light.
2. **IoT Gateway**: A device that connects sensors to the internet, such as Raspberry Pi, Arduino with Wi-Fi/Bluetooth, or dedicated IoT gateways.
3. **Microcontroller/Development Board (e.g., Arduino, ESP32, Raspberry Pi)**: To interface the sensor with the IoT gateway.
4. **Cloud Platform (e.g., ThingSpeak, AWS IoT, Google Cloud IoT)**: For collecting, storing, and analyzing the data.
5. **Jumper Wires**: For connecting the sensor and microcontroller.
6. **Breadboard**: For prototyping connections.
7. **Power Supply**: For the IoT gateway and sensor.

Procedure:

Step 1: Setting up the Light Sensor (LDR)

- **LDR Setup**: Connect the LDR to the analog input pin of your microcontroller. Typically, you will connect one leg of the LDR to a 5V supply, and the other leg to the analog input pin with a pull-down resistor (to ground) to form a voltage divider. The output voltage varies with the amount of light hitting the sensor.

Step 2: Connecting the IoT Gateway (Microcontroller)

- **Choose a microcontroller with Wi-Fi/Bluetooth (e.g., ESP32, Arduino with Wi-Fi Shield)**: This microcontroller will connect to the internet and send data to the cloud.
- **Programming**: Use the microcontroller's programming environment (Arduino IDE, PlatformIO) to write code that reads the sensor data and transmits it to the IoT gateway (WiFi or Bluetooth).
  - For Arduino/ESP32: You might use libraries like `WiFi.h`, `HTTPClient.h` to send data to the cloud via HTTP or MQTT.
  - **Example Code (Arduino):**

    ```
    cpp
    Copy
    ```

```
#include <WiFi.h>  #include
<HTTPClient.h>
 const char* ssid = "your_SSID"; const
char* password = "your_PASSWORD";
const char* serverName = "http://your-server-endpoint.com";
const int lightSensorPin = A0;

void setup() {
Serial.begin(115200);
  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {     delay(1000);
    Serial.println("Connecting to WiFi...");
  }
  Serial.println("Connected to WiFi");
}  void loop()
{
  int lightValue = analogRead(lightSensorPin);
float voltage = lightValue * (5.0 / 1023.0);
  Serial.println("Light sensor value: " + String(voltage));

  HTTPClient http;   http.begin(serverName);
  http.addHeader("Content-Type", "application/x-www-
formurlencoded");

  String postData = "light=" + String(voltage);   int
httpResponseCode = http.POST(postData);

  if (httpResponseCode > 0) {
    Serial.println("Data sent to server, Response: " +
String(httpResponseCode));
  } else {
    Serial.println("Error in sending data");
  }    http.end();
  delay(5000);  // Delay before sending the next reading }
```

Step 3: Setting up the IoT Gateway

- **Cloud Integration**: Set up the cloud platform or local IoT server to receive and process data from the IoT gateway. You can use services like **ThingSpeak**, **AWS IoT Core**, or **Google Cloud IoT** to collect and visualize the data.
    - For **ThingSpeak**: You will need to create a ThingSpeak channel, generate an API key, and send the data using HTTP requests.
    - For **AWS IoT**: You can use MQTT protocol to send data from the IoT device to the AWS cloud.

Step 4: Data Visualization and Automation

- **Visualization**: On your cloud platform, set up data visualization tools like dashboards to display real-time light intensity readings.
- **Automation**: You can also set up actions based on the light sensor's readings. For example, if the light intensity falls below a certain threshold, it could trigger an alert or turn on an actuator (e.g., a light bulb or fan).

Step 5: Testing and Observations

- **Test**: Test your setup by varying the light intensity (e.g., using a flashlight or by covering the sensor). Observe how the data changes on the cloud platform.
- **Observe**: The cloud platform should receive the sensor readings and update them in real time. You can graph the data to understand the changes in light intensity over time.

Expected Results:

- **Data Sent to Cloud**: You should see the sensor data (light intensity) sent to the cloud platform at regular intervals.
- **Visualization**: A graph showing the changes in light levels based on the environment's lighting.
- **Automation (Optional)**: If using automation, check if actions (e.g., turning on/off a device) are triggered when the light intensity falls below or exceeds a certain threshold.

Conclusion:

This experiment demonstrates the practical use of a light sensor in an IoT environment. By connecting the sensor to an IoT gateway and cloud platform, you can monitor light intensity remotely and even trigger automation based on sensor data.

EXPERIMENT NO-7

Aim: Measure the output of water sensor using IoT gateway.

Theory:

To study the practical application of a water sensor using an IoT gateway, you'll need to set up a simple experiment that involves connecting a water sensor to an IoT device (like a microcontroller or IoT gateway) and monitoring water levels or detecting the presence of water in a given location. Below is a step-by-step guide to help you understand and implement this experiment:

Objective:

- To monitor water levels or detect water presence using a water sensor.
- To send data from the sensor to an IoT gateway and view the data on a cloud-based platform or locally.

Components Required:

1. **Water Sensor** (e.g., YL-69 or FC-28 soil moisture sensor, or a water level detection sensor)
2. **Microcontroller/IoT Gateway** (e.g., Arduino, ESP8266, ESP32, Raspberry Pi)
3. **Wi-Fi Module** (if your microcontroller doesn't have built-in Wi-Fi, e.g., ESP8266) 4. **Cloud Service** for IoT (e.g., ThingSpeak, Blynk, or Firebase)
5. **Jumper Wires and Breadboard** for making connections.
6. **Power Supply** for the microcontroller and sensors.
7. **Resistors** (for voltage regulation, depending on sensor requirements).
8. **Water Source** (for testing).

Steps to Perform the Experiment:

Step 1: Connect the Water Sensor to the Microcontroller

1. **Water Sensor:**
   - The water sensor typically has three pins: VCC (Power), GND (Ground), and DO (Digital Output) or AO (Analog Output).
     - Connect VCC to 5V or 3.3V (depending on your microcontroller).
     - Connect GND to ground.
     - Connect the output pin (either DO or AO) to a digital or analog pin on your microcontroller.
2. **Microcontroller:**
   - Use an Arduino or ESP32/ESP8266 as the IoT gateway.
   - If you're using an Arduino, connect the sensor's DO or AO pin to one of the digital or analog pins of the Arduino, respectively.

Step 2: Write Code to Read Sensor Data

1. **Arduino Code Example**: Here's an example code to read from a water sensor (Digital Output):

```
cpp
Copy
```

```
int
waterSe
nsorPin
= 2;
//
Digital
Pin
connect
ed to
the
sensor'
s DO
pin
int sensorState = 0;      // Variable to store sensor state

void setup() {
  Serial.begin(9600);  // Start serial communication for monitoring
pinMode(waterSensorPin, INPUT);
}  void
loop() {
  sensorState = digitalRead(waterSensorPin);  // Read the
sensor's state
  if (sensorState == HIGH) {
    Serial.println("Water Detected");
  } else {
    Serial.println("No Water Detected");
  }   delay(1000);  // Wait for 1 second before checking
again }
```

Step 3: Set Up IoT Gateway for Data Transmission

1. **Using Wi-Fi to Send Data:**
   - If you're using an ESP8266 or ESP32, you can send the data to a cloud platform like ThingSpeak, Blynk, or Firebase.
   - **Example with ThingSpeak (ESP8266)**:
     1. Install the ThingSpeak library in the Arduino IDE.
     2. Set up a ThingSpeak account and create a channel.
     3. Use the following code to send the sensor data to ThingSpeak:

     ```
     cpp
     Copy
     #include <ESP8266WiFi.h> #include
     <ThingSpeak.h>  const char* ssid =
     "yourSSID"; const
     char* password = "yourPASSWORD"; const char*
     host = "api.thingspeak.com";
     const unsigned long channelID = yourChannelID;  // Replace with
     your channel ID
     const char* writeAPIKey = "yourWriteAPIKey";  // Replace with
     your write API key

     WiFiClient client;

     int waterSensorPin = 2;  // Sensor pin
     int sensorState = 0;
     ```

```
void setup() {
  Serial.begin(115200);
WiFi.begin(ssid, password);
   while (WiFi.status() != WL_CONNECTED) {
delay(1000);
    Serial.println("Connecting to WiFi...");
}

  ThingSpeak.begin(client);
}
 void loop()
{   sensorState =
digitalRead(waterSensorPin);

  if (sensorState == HIGH) {
    ThingSpeak.setField(1, 1);  // 1 means water detected
  } else {
    ThingSpeak.setField(1, 0);  // 0 means no water
}

  ThingSpeak.writeFields(channelID, writeAPIKey);

  delay(2000);  // Upload data every 2 seconds
}
```

- After uploading the code, your sensor's data will be visible on ThingSpeak.

Step 4: Monitor and Visualize the Data

- **Using ThingSpeak:**
  - Log into your ThingSpeak account and navigate to your channel.
  - You'll be able to see the water sensor data in real time. o You can also create charts, alerts, and more based on the sensor data.

Step 5: Analyze the Data

- Check how the sensor reacts when exposed to water or when dry.
- You can set thresholds (e.g., water detected = 1, no water = 0) and program responses to take actions (like turning on a pump, sending a notification, etc.).

Step 6: Optional Improvements

1. **Water Level Monitoring:**
   - Use an analog water level sensor for more granular data.
   - Display the water level on a mobile app or website.
2. **Alert System:**
   - Set up email or SMS alerts when water is detected. o Integrate with other systems to trigger an action (e.g., turn on a pump or valve).

Conclusion:

By the end of this experiment, you'll have a working IoT system that can detect water presence or levels and communicate the data to a cloud platform using an IoT gateway. This type of setup can be extended to many applications, such as flood detection, irrigation systems, or water leakage monitoring.

EXPERIMENT NO- 8

Aim: Examine the working of 5G drone.

Theory:

An experiment to study the working of a 5G-enabled drone would focus on examining the interaction between the drone and the 5G network, highlighting key aspects like communication speeds, latency, and range, among other performance factors. Below is an outline for setting up and conducting this experiment:

Objective:

To study the working, advantages, and challenges of using 5G technology for controlling and operating drones.

Key Areas to Investigate:

1. **Communication Speed and Latency:**
   - Measure the data transfer speeds and latency between the drone and the 5G network.
   - Evaluate the time taken for commands to be received and executed by the drone.
2. **Range and Connectivity:**
   - Test the maximum range of the drone while maintaining a stable connection with the 5G network.
   - Explore how the drone performs at varying distances from the base station or 5G tower.
3. **Video Streaming Quality:**
   - Assess the quality of real-time video streaming from the drone back to the control station (i.e., low latency, high resolution).
   - Investigate how 5G enables high-quality live streaming compared to previous technologies (like 4G).
4. **Drone Control in Different Environments:**
   - Test drone performance in various environmental conditions like urban areas (with tall buildings and interference) vs open fields to see how 5G performs under different challenges. 5. **Interference and Reliability:**
   - Test the drone's behavior in areas with high interference or congestion on the network, such as in crowded urban environments or during a high-demand network scenario.

Equipment Needed:

- **Drone with 5G capabilities**: A drone that can connect to 5G networks (preferably with an integrated 5G module or support for 5G connectivity).
- **5G Base Station or 5G Mobile Hotspot**: A device capable of providing 5G connectivity to the drone. This can be a 5G-enabled mobile hotspot or a 5G base station.
- **Monitoring Tools**: Software or hardware to measure latency, network speed, and the quality of communication.
- **Control Station**: A laptop or remote control system capable of receiving the drone’s data and sending commands via the 5G network.

Steps of the Experiment:

1. Setup and Initialization:

- Set up the 5G base station/mobile hotspot at the starting point of the experiment.
- Power up the drone and ensure it is connected to the 5G network.
- Set up a control station for remote piloting and monitoring the drone's flight.
- Ensure the drone has a live video feed (camera feed) connected to the control station.

2. Measure Latency and Speed:

- Send control commands from the base station to the drone (e.g., take off, move left/right, altitude changes).
- Measure the time delay (latency) between issuing the command and the drone responding.
- Use network analysis tools to check data transmission rates and compare them to the expected speeds of 5G networks.

3. Range Testing:

- Begin flying the drone from a stationary position and gradually increase the distance while keeping an eye on the connection stability and quality.
- Record the maximum distance at which the drone still maintains a stable 5G connection.

4. Video Quality Testing:

- Begin a video streaming session from the drone back to the control station.
- Check video resolution, frame rate, and quality (i.e., whether buffering or lag occurs during transmission).
- Test in both ideal and challenging environments (e.g., dense city, open fields).

5. Stress Test (Simulate Interference or Network Congestion):

- Introduce simulated interference or network congestion by testing in high-density areas with many other devices connected to the same 5G network.
- Measure the drone's ability to maintain stable communication under stress (e.g., will the drone disconnect, suffer from lag, or experience delays?).

6. Collect Data and Results:

- Throughout the experiment, collect data on latency, speed, connectivity range, and video quality.
- Record instances of failure, disconnection, or unexpected behavior.

7. Analyze Results:

- Analyze the collected data to assess how well the 5G technology supports drone communication.

- Compare this data with what would be expected from lower-generation networks like 4G to observe the benefits (e.g., lower latency, faster speeds, improved reliability).

Expected Results:

- **Lower Latency:** You should observe lower latency in drone control, leading to more responsive flight maneuvers.
- **Improved Video Quality:** High-definition video should stream with little to no lag, even in real-time.
- **Greater Range:** The 5G connection should allow the drone to operate over a longer distance compared to 4G or Wi-Fi.
- **Enhanced Connectivity:** The drone should experience fewer connection drops and higher reliability when compared to older network technologies.

Challenges to Watch For:

- **Signal Obstruction:** 5G networks can be sensitive to physical obstructions, such as tall buildings, which may interfere with the signal in urban environments.
- **Battery Consumption:** Continuous 5G communication might put additional strain on the drone's battery life, which could be a limiting factor during extended flight times.
- **Network Congestion:** If the 5G network is overloaded or congested, performance could degrade, leading to issues with real-time control and video streaming.

Conclusion:

This experiment would provide valuable insights into the practical use of 5G networks for drone operations, helping to identify both the potential advantages and the current limitations of 5G technology in the field of unmanned aerial systems (UAS).

EXPERIMENT NO- 9

Aim: Compare the performance of AR-VR over 4G & 5G networks.

Theory:

To study the integration of AR-VR devices with 5G networks, you can conduct an experiment that evaluates both the technical feasibility and the performance of the system. The experiment should focus on the capabilities of 5G to handle the bandwidth, low latency, and high reliability required by AR-VR applications.

Here's an outline of the experiment you could conduct:

1. Objective

- To explore how 5G network characteristics (e.g., latency, speed, reliability) improve the performance of AR-VR devices.
- To compare the performance of AR-VR devices on 4G vs. 5G networks.

2. Materials

- **AR-VR Devices**: Choose a set of AR and VR devices like Oculus Quest, Microsoft HoloLens, or Magic Leap for AR and devices like HTC Vive or PlayStation VR for VR.
- **5G Connectivity**: Ensure you have access to a 5G network (this can be done through a local 5G provider or a 5G testbed).
- **4G Connectivity**: Use a 4G network as a baseline.
- **Network Performance Tools**: Tools for measuring bandwidth, latency, and packet loss (e.g., iPerf, NetFlow, or specific 5G tools).
- **AR/VR Content**: Pre-recorded or live AR/VR content to test the devices' performance (e.g., immersive environments, real-time multiplayer games, or 3D video streams).
- **Testing Environment**: A controlled space where both 5G and 4G networks are available.

3. Experiment Setup

- **Network Setup**:
  - Test the AR/VR devices on both 4G and 5G networks.
  - Measure network performance (e.g., speed, latency, jitter, packet loss) under both conditions.
- **AR-VR Testing**:
  - Use AR/VR applications that are sensitive to network performance (e.g., multiplayer games, live interactive experiences, or video streaming).
  - Monitor user experience in terms of responsiveness, visual quality, and immersion.
  - Test in a variety of environments (e.g., moving users, stationary users, dense signal areas, and weak signal areas).
- **Performance Metrics**:

- **Latency**: Measure the latency between the user input (e.g., head movement or controller input) and the response on the device.
- **Bandwidth**: Measure the bandwidth usage during heavy AR/VR content (e.g., high-resolution video streaming, complex graphical environments).
- **Reliability**: Track how often the connection drops or experiences glitches during the test.
- **User Experience**: Qualitative assessment of the experience in terms of immersion, quality of interaction, and comfort.

4. Procedure

- **Step 1**: Set up the AR/VR devices and connect them to the 4G and 5G networks one at a time.
- **Step 2**: Run multiple trials of the AR/VR content and ensure that the device is fully connected to the network.
- **Step 3**: Collect network performance data (e.g., speed, latency, jitter, and packet loss) using network monitoring tools.
- **Step 4**: While running AR/VR content, measure performance using subjective feedback and quantitative data (frame rate, latency, etc.).
- **Step 5**: Analyze how the AR/VR experience changes when switching between 4G and 5G networks.

5. Analysis

- **Network Comparison**: Compare the performance of 4G vs. 5G networks in terms of latency, bandwidth, and reliability. Evaluate how these differences impact the AR/VR experience.
- **Device Performance**: Assess the AR/VR devices' performance in terms of how smooth the content runs, the level of immersion, and how well the device responds to user input in each scenario.
- **Qualitative Feedback**: Gather user feedback about the immersion and fluidity of the AR/VR experience on both networks.
- **Network Limitations**: Investigate how 5G's low latency, increased bandwidth, and reliability improve real-time AR/VR applications compared to 4G.

6. Potential Improvements and Considerations

Explore possible improvements in 5G technology (e.g., edge computing) and how it can further enhance AR/VR applications by reducing latency even more.

- Consider the effect of different 5G deployment types (e.g., standalone vs. nonstandalone) on performance.
- Assess the impact of congestion in a 5G network (due to heavy user demand) on the AR/VR experience.

7. Conclusion

- Summarize how the integration of AR/VR devices with 5G networks impacts overall performance.

- Discuss any benefits or limitations observed, and the potential for future applications in AR/VR enhanced by 5G networks.

EXPERIMENT NO-10

Aim: Understand the working of 5G evolution module.

Theory:

To study the working of the 5G Evolution Module, an experiment can be conducted to observe the core components of 5G networks, how they work together, and how different technologies (like network slicing, beamforming, and ultra-low latency) improve communication speeds and capabilities. Here's an outline for an experiment:

Objective:

To understand the working of the 5G Evolution Module, its technologies, and its role in enhancing mobile network performance.

Required Equipment:

- 5G-capable mobile device (smartphone or testing hardware)
- 5G Evolution Network Infrastructure (e.g., 5G base stations, antennas, spectrum analyzers)
- 5G test module (which could be a simulator or actual 5G hardware)
- Software tools for network testing (e.g., network emulators, traffic generators)
- Signal measurement and monitoring tools
- Access to a 5G network or testbed **Steps for the Experiment:** 1. **Set Up the Environment:** ○ Ensure access to a 5G network or a dedicated 5G testbed, which is often provided by telecommunications companies or research institutions.

  ○ Set up testing equipment, including network analyzers, 5G signal generators, and measurement tools to monitor the performance of the network.

2. **Understand the 5G Network Components:**
   ○ **Radio Access Network (RAN):** Learn how the 5G RAN works, focusing on new technologies like massive MIMO (Multiple Input Multiple Output), beamforming, and the use of millimeter waves. ○ **Core Network (5GC):** Study the new 5G core architecture, which enables capabilities like network slicing, edge computing, and ultra-low latency communication.
3. **Connect to the 5G Network:**
   ○ Use a 5G-compatible device to connect to the 5G network. ○ Measure the connection speed, latency, and other key performance indicators (KPIs).
4. **Study Key Features of 5G Evolution:**
   ○ **Network Slicing:** Study how network slicing allows the creation of virtual networks tailored to different use cases (e.g., autonomous vehicles, IoT devices, high-speed data transfer).
   ○ **Beamforming and Massive MIMO:** Observe how these technologies improve signal strength, coverage, and throughput by focusing the radio waves toward the devices. ○ **Low Latency:** Measure the latency of various

operations (e.g., gaming, video streaming, remote operations) to see the improvements offered by 5G compared to 4G networks.

5. **Test the 5G Performance:**
    - **Throughput Test:** Conduct speed tests to measure download and upload speeds on the 5G network. ○ **Latency Test:** Measure the round-trip latency during high-demand activities like online gaming or video calling. ○ **Signal Strength Test:** Assess the signal quality using tools like a signal strength meter and evaluate how the device handles mobility between cells. ○ **Reliability Test:** Test the network's ability to maintain a stable connection during various realworld activities like walking, driving, or while moving between different cells of the network.
6. **Simulation/Use Case Testing:**
    - Run simulations for specific 5G applications, such as smart cities, autonomous driving, or healthcare services (e.g., remote surgery). ○ Monitor the behavior of different devices under various traffic conditions to see how the 5G network adapts to the demand.
7. **Data Analysis:**
    - Use software tools to analyze the collected data, such as speed, signal strength, and latency.
    - Compare the performance of 5G with previous technologies like 4G to highlight improvements.
8. **Conclusion:**
    - Summarize how the 5G Evolution Module works, focusing on its core components and technologies. ○ Discuss the advantages of 5G, such as higher speeds, lower latency, better connectivity, and support for a wide range of applications. ○ Analyze any challenges or limitations observed during the experiment.