# **Satellite Communication System**

## **Table of Contents**

1. Overview
2. System Architecture
3. Communication Flow
4. Protocols Used
   1. MQTT
   2. CCSDS
   3. DVPS
5. Commands Overview
   1. System Setup Commands
   2. From NPK Sensor to Server
   3. From Server to Satellite
   4. From Satellite to Mobile App
6. Detailed Communication Flow
7. Example of Full Communication Flow
8. Conclusion

## **Overview**

This document outlines the communication architecture for a satellite monitoring system, detailing the flow of data from **NPK Sensors** to the **Mobile App**. The system leverages multiple protocols to ensure efficient communication across the entire system:

* **NPK Sensors** send data to the **Server** via **MQTT**.
* The **Server** communicates with the **Satellite** using **CCSDS** to issue commands.
* The **Satellite** sends telemetry and images back to the **Mobile App** using **DVPS**.

This system architecture ensures a seamless, real-time data exchange between the sensors, satellite, server, and mobile application.

## **System Architecture**

### **Key Components:**

1. **NPK Sensor**: Collects real-time sensor data (telemetry, environment, etc.) and sends it to the **Server** via **MQTT**.
2. **Server**: Receives the sensor data, processes it, and sends **CCSDS** commands to the **Satellite** for operations (e.g., telemetry collection, image capture).
3. **Satellite**: Executes the received commands, collects telemetry, and sends the data back to the **Mobile App** via **DVPS**.
4. **Mobile App**: Receives telemetry and images from the **Satellite** and displays them to the user interface.

### **Communication Flow:**

1. **NPK Sensor → Server**: **MQTT** protocol is used to transmit real-time data from the sensor to the server.
2. **Server → Satellite**: **CCSDS** protocol is used for communication between the server and the satellite for operations like telemetry collection or image capture.
3. **Satellite → Mobile App**: **DVPS** protocol handles the transmission of telemetry data and captured images to the mobile app for display.

## **Communication Flow**

The communication process can be broken down into the following steps:

1. **Sensor to Server**:
   * The **NPK Sensor** collects telemetry data and sends it to the **Server** using MQTT.
2. **Server to Satellite**:
   * The **Server** processes the received sensor data and sends **CCSDS** commands to the **Satellite** to trigger specific operations, like collecting telemetry or capturing images.
3. **Satellite to Mobile App**:
   * The **Satellite** transmits telemetry data and images to the **Mobile App** using **DVPS**. The app receives and displays the data.

## **Protocols Used**

### **MQTT**

**MQTT** (Message Queuing Telemetry Transport) is a lightweight messaging protocol used for real-time sensor data transmission. It ensures low-latency and low-overhead communication.

* **Usage**:
  + **Sensor → Server**: Transmitting telemetry data (e.g., temperature, altitude).
  + **Example**: sensor\_data = {"temperature": 22.5, "altitude": 3500}

### **CCSDS**

**CCSDS** (Consultative Committee for Space Data Systems) is used for satellite communication. It ensures robust and reliable command transmissions to the satellite for operations such as telemetry collection and image capturing.

* **Usage**:
  + **Server → Satellite**: Commands for operations like telemetry or image capture.
  + **Example Command**: START\_TELEMETRY\_COLLECTION

### **DVPS**

**DVPS** (Digital Video and Packetized Service) is used for satellite-to-mobile communication. It handles telemetry data and image/video transmission back to the mobile app.

* **Usage**:
  + **Satellite → Mobile App**: Sending telemetry data and images.
  + **Example Data**: telemetry\_data = {"temperature": 23.0, "altitude": 3495}, image\_data = base64\_encoded\_image

## **Commands Overview**

### **System Setup Commands**

To set up the server and environment, the following commands are necessary:

**Install MongoDB**:  
bash  
Copier le code  
sudo apt install -y mongodb-org

**Install Python 3 and PIP**:  
bash  
Copier le code  
sudo apt install python3 python3-pip

**Install Required Python Libraries**:  
bash  
Copier le code  
pip3 install pandas numpy paho-mqtt

**Install OpenSSH Server**:  
bash  
Copier le code  
sudo apt install openssh-server

sudo systemctl enable ssh

**Install Mosquitto and MQTT Clients**:  
bash  
Copier le code  
sudo apt install mosquitto mosquitto-clients

**Install Flask and FastAPI for Web API**:  
bash  
Copier le code  
pip3 install flask fastapi

**Start MongoDB**:  
bash  
Copier le code  
sudo systemctl start mongodb

**Install OpenCV for Image Processing**:  
bash  
Copier le code  
sudo apt install python3-opencv

### **From NPK Sensor to Server (via MQTT)**

* The **NPK Sensor** sends real-time telemetry data to the **Server** using MQTT.
* **Command**: SEND\_SENSOR\_DATA\_TO\_SERVER
  + **Example**:
    - **MQTT Message**: sensor\_data = {"temperature": 22.5, "altitude": 3500}

### **From Server to Satellite (via CCSDS)**

The **Server** sends **CCSDS** commands to the **Satellite** for performing operations like telemetry collection or image capture.

* **Commands**:
  + **Start Telemetry Collection**: START\_TELEMETRY\_COLLECTION
  + **Capture Image**: CAPTURE\_IMAGE
* **Example Command**:
  + **CCSDS Command**: START\_TELEMETRY\_COLLECTION
  + **CCSDS Command**: CAPTURE\_IMAGE

### **From Satellite to Mobile App (via DVPS)**

The **Satellite** transmits telemetry data and images to the **Mobile App** using **DVPS**.

* **Command**: TRANSMIT\_TELEMETRY\_AND\_IMAGES
  + **Telemetry Data**: { "temperature": 23.0, "altitude": 3495 }
  + **Image Data**: Base64-encoded image
* **Example Command**:
  + **DVPS Data**: telemetry\_data = {"temperature": 23.0, "altitude": 3495}
  + **DVPS Data**: image\_data = base64\_encoded\_image

## **Detailed Communication Flow**

### **1. NPK Sensor → Server**

* **Command**: SEND\_SENSOR\_DATA\_TO\_SERVER
* **Protocol**: MQTT
* **Data Sent**: Real-time sensor data (e.g., temperature, altitude)

### **2. Server → Satellite**

* **Command**: START\_TELEMETRY\_COLLECTION / CAPTURE\_IMAGE
* **Protocol**: CCSDS
* **Action**: Starts telemetry collection or captures images.

### **3. Satellite → Mobile App**

* **Command**: TRANSMIT\_TELEMETRY\_AND\_IMAGES
* **Protocol**: DVPS
* **Data Sent**: Telemetry data and images.

### **4. Mobile App**

* **Command**: DISPLAY\_RECEIVED\_DATA
* **Action**: Displays telemetry data and images on the user interface.

## **Example of Full Communication Flow**

1. **NPK Sensor → Server**:
   * **Sensor Data**: { "temperature": 22.5, "altitude": 3500 }
   * The **NPK Sensor** sends the data to the **Server** using **MQTT**.
2. **Server → Satellite**:
   * **Command**: START\_TELEMETRY\_COLLECTION
   * The **Server** sends a **CCSDS** command to the **Satellite** to start telemetry collection.
3. **Satellite → Mobile App**:
   * **Telemetry Data**: { "temperature": 23.0, "altitude": 3495 }
   * **Image Data**: Base64-encoded image
   * The **Satellite** sends the telemetry and image data to the **Mobile App** using **DVPS**.
4. **Mobile App**:
   * The **Mobile App** displays the received telemetry and image data on the user interface.

## **Conclusion**

This **Satellite Communication System** utilizes **MQTT**, **CCSDS**, and **DVPS** protocols to ensure effective, real-time communication between the \*\*NPK Sensor