Linux-based Modem communication system

System overview

The system establishes a two-way communication link with the Iridium Certus 9770 modem using MQTT (Message Queuing Telemetry Transport). It enables remote command execution, response handling, and error management via a structured MQTT-based architecture.

Key components:

1. systemd Service: Automates the execution of the setup script on boot and ensures restarts in case of failure.
2. Mosquitto Broker: The MQTT broker facilitating message exchange between the platform and the systemd.
3. Platform: Display Modem related information

Systemd Service

The systemd, which is a linux-based daemon system, is developed to unify the service configuration and system behavior across the linux distributions in order to activate the Iridium Certus Modem and ensure its availability for communication.

The core functionalities of the systemd are:

1. The modem manager ensures that the system script starts on boot as well as handling threads.
2. Set up MQTT to treat the messages as JSPR commands in order to send them to the Iridium through serial port A using the serial python library.

```
# MQTT Client Setup
client = mqtt.Client()
modem = ModemManager()

def on_connect(client, userdata, flags, rc):
      client.subscribe("modem/commands")

def on_message(client, userdata, msg):
      try:
….
```
Figure 1: Code snippet for setting up MQTT client

3. Structure JSPR commands using **paho-mqtt** library that handles a set of functions:
    a. Ensures that the system is connected to the MQTT broker, Mosquitto.
    b. Publishing/subscribing to the following topics:
        i. modem/commands
        ii. modem/responses<request-id>
        iii. modem/errors
    c. Handles message callbacks such as on_message()

```
payload = json.loads(msg.payload)
      command = {
            'request_id': payload.get('request_id'),
            'action': payload['action'],
            'path': payload['path'],
            'params': payload.get('params', {}),
            'expected_code': payload.get('expected_code', 200)
      }
```
Figure 2: Code snippet for structuring the JSPR command request

Mosquitto Broker

MQTT is the mosquitto broker handling transferring requests to the system which is going to treat it and send it as JSPR commands to the Iridium. In return, the latter will send a response back to the system that is going to send it back with a MQTT request.

The main functionalities of the Broker:
1. Publishes ongoing messages through the topics:
    a. modem/commands

```
{
  "request_id": string,
  "action": "GET"/"PUT",
  "path": JSPR_FUNCTION_NAME,
  "params": {}

}
```
Figure 3: Code snippet for the commands message

2. Subscribes to the ingoing messages through the topic:
    a. modem/responses/<request-id>

```
{
  "type": "response",
  "request_id": string,
  "status": number,
  "path": JSPR_FUNCTION_NAME,
  "data": {}
}
```
Figure 4: Code snippet for the responses format

b. modem/errors

```
{
  "type": "response",
  "request_id": string,
  "status": number,
  "path": JSPR_FUNCTION_NAME,
  "data": {}
}
```
Figure 5: Code snippet for the errors format

Platform

The platform is the web-app that allows users specifically pilots, co-pilots, and the people who are interested in knowing about the skylink and Iridium Certus status. The platform is developed using React framework, and has MQTT services and hooks