Step-by-Step Guide for Managing the SIM8200EA-M2 5G Module in the PHIL Lab

May 19, 2025

1 Introduction

This guide provides detailed instructions for setting up and managing the SIM8200EA-M2 5G HAT on Raspberry Pi 4 modules for the Power Hardware-in-the-Loop (PHIL) lab's 5G-connected sensing project. The objective is to enable reliable MQTT-based data exchange of experiment parameters (e.g., voltage, frequency, power) over a 5G network. Due to the unavailability of Single Network Slice Selection Assistance Information (S-NSSAI), a hybrid prioritization framework replaces network slicing, combining MQTT QoS levels, Linux `tc` traffic shaping, and a custom MQTT client with prioritization logic. The guide is based on:

- Project documents: `installation_simcomdriver_matt_ros_udp.sh`, `Guidelines 5G modules.sh`, `Instructions for operating the PHIL system_V3_final.pdf`.
- Waveshare SIM8200EA-M2 5G HAT wiki: https://www.waveshare.com/wiki/SIM8200EA-M2_5G_HAT.

Assumptions: The Raspberry Pi 4 runs Ubuntu 20.04, the SIM8200EA-M2 is connected via USB, and the project uses Mosquitto (MQTT broker) and Paho MQTT (client) with ROS 2 nodes (`mqtt_offered`, `udp_server`). Python code uses the 5G module's IP address, with 5G configuration limited to AT commands via `minicom` during installation.

2 Step-by-Step Instructions

2.1 Step 1: Prepare the Hardware

Objective: Set up the SIM8200EA-M2 5G HAT and Raspberry Pi 4.

1. **Connect the SIM8200EA-M2**:
   - Attach the SIM8200EA-M2 to the Raspberry Pi 4 via USB.
   - Insert a 5G SIM card into the SIM card slot.
   - Connect 5G antennas to the ANT ports (MAIN and AUX).
   - Power on the Raspberry Pi (5V, 3A power supply).
2. **Verify Connections**:
   - Ensure USB cable, SIM card, and antennas are secure.
   - Position antennas for optimal signal.

3. **Log In**:

```
ssh ubuntu@172.16.11.10  # Verify credentials
```

4. **Update System**:

```
sudo apt-get update
sudo apt-get upgrade -y
```

2.2 Step 2: Install SIM8200EA-M2 Drivers

Objective: Install drivers to enable the `wwan0` interface.

1. **Install Dependencies**:

```
sudo apt-get install -y net-tools build-essential p7zip-full
```

2. **Download Driver**: Follow project documents or Waveshare wiki.

3. **Replace `install.sh`**:

```
nano install.sh
#!/bin/bash
linuxheaders="linux-headers-"
uname_r=$(uname -r)
headerdir="/usr/src/$linuxheaders$uname_r"
if [ ! -d "$headerdir" ]; then
    echo "Kernel headers not found, installing..."
    sudo apt-get update
    sudo apt-get install -y linux-headers-$(uname -r)
else
    echo "Kernel headers directory found at: $headerdir"
fi
cd option
make
mv /lib/modules/$(uname -r)/kernel/drivers/usb/serial/option.ko /lib/
   modules/$(uname -r)/kernel/drivers/usb/serial/option_bk.ko
cp option.ko /lib/modules/$(uname -r)/kernel/drivers/usb/serial/
cd ..
cd qmi_wwan_simcom/
make
cp qmi_wwan_simcom.ko /lib/modules/$(uname -r)/kernel/drivers/net/usb
cd ..
depmod
modprobe option
modprobe qmi_wwan_simcom
dmesg | grep "ttyUSB"
dmesg | grep "qmi_wwan_simcom"
mkdir -p /usr/share/udhcpc
sudo chmod 777 default.script
sudo cp default.script /usr/share/udhcpc
```

```
chmod +x install.sh
sudo ./install.sh
```

4. **Verify Drivers**:

```
dmesg | grep "ttyUSB"  # Should show ttyUSB0, ttyUSB1, ttyUSB2
dmesg | grep "qmi_wwan_simcom"  # Should show driver registration
```

2.3 Step 3: Configure SIM8200EA-M2 with AT Commands

Objective: Set NDIS mode and verify 5G connectivity.

1. **Install `minicom`**:

```
sudo apt-get install -y minicom
```

2. **Set NDIS Mode**:

```
sudo minicom -D /dev/ttyUSB2
AT+CUSBCFG=USBID,1E0E,9001
```

   Exit: `Ctrl+A, Q, 'Yes'`.

3. **Verify SIM and Network**:

```
sudo minicom -D /dev/ttyUSB2
AT  # Should return OK
ATI  # Module info
AT+CPIN?  # +CPIN: READY
AT+CSQ  # Signal quality (e.g., +CSQ: 20,99)
AT+COPS?  # Operator (e.g., +COPS: 0,0,"Operator",9)
AT+CGREG?  # +CGREG: 0,1 or 0,5
AT+C5GREG?  # +C5GREG: 0,1
AT+CNMP=71  # Prefer 5G
AT+CNSMOD?  # +CNSMOD: 0,9 for 5G
```

   Exit `minicom`.

4. **Create Configuration Script (optional)**:

```
nano configure_sim8200.sh
#!/bin/bash
echo "AT" | sudo minicom -D /dev/ttyUSB2 -b 115200
echo "ATI" | sudo minicom -D /dev/ttyUSB2 -b 115200
echo "AT+CPIN?" | sudo minicom -D /dev/ttyUSB2 -b 115200
echo "AT+CSQ" | sudo minicom -D /dev/ttyUSB2 -b 115200
echo "AT+COPS?" | sudo minicom -D /dev/ttyUSB2 -b 115200
echo "AT+CGREG?" | sudo minicom -D /dev/ttyUSB2 -b 115200
echo "AT+CNMP=71" | sudo minicom -D /dev/ttyUSB2 -b 115200
echo "AT+CNSMOD?" | sudo minicom -D /dev/ttyUSB2 -b 115200
```

```
chmod +x configure_sim8200.sh
./configure_sim8200.sh
```

2.4 Step 4: Establish 5G Network Connection

Objective: Connect to the 5G network via `wwan0`.

1. **Run Dial-Up**:

```
cd SIM8200_for_RPI/Goonline
make
sudo ./simcom-cm
```

2. **Verify `wwan0`**:

```
ip a  # Should show wwan0 with 192.168.6.21/28
```

3. **Automate Dial-Up**:

```
sudo nano /etc/rc.local
#!/bin/bash
sudo /home/ubuntu/SIM8200_for_RPI/Goonline/simcom-cm
exit 0
```

```
sudo chmod +x /etc/rc.local
```

4. **Test Connectivity**:

```
ping 192.168.88.10 -I wwan0
ping 192.168.6.21 -I wwan0
ping 8.8.8.8 -I wwan0
```

2.5 Step 5: Configure Traffic Prioritization

Objective: Implement a hybrid prioritization framework to replace network slicing, ensuring low latency for critical data (frequency, voltage, current), sufficient bandwidth for power data, and efficient handling of non-critical data (mode).

Since Single Network Slice Selection Assistance Information (S-NSSAI) is unavailable, a hybrid framework combines:

- **MQTT QoS Levels**: Assign QoS 1 for reliable delivery of critical topics (frequency, voltage, current, power) and QoS 0 for non-critical mode, mimicking URLLC and mMTC.
- **Linux `tc` Traffic Shaping**: Prioritize MQTT traffic (port 1883) on `wwan0`, allocating bandwidth to replicate URLLC and eMBB.
- **Custom MQTT Client**: Use a priority queue to process critical topics first, enhancing URLLC and eMBB prioritization.

1. **Configure MQTT QoS**:
   - Update the MQTT configuration to assign QoS levels:

```
nano ~/ros2_ws/src/mqtt_offered_client/config/mqtt_config.yaml
mqtt:
  broker: "172.16.11.10"
  port: 1883
  topics:
    phil/grid/frequency/measured: {ros_topic: "/ros2mqtt/grid/
       frequency/measured", qos: 1}
    phil/grid/frequency/reference: {ros_topic: "/ros2mqtt/grid/
       frequency/reference", qos: 1}
    phil/supercap/voltage/phase1: {ros_topic: "/ros2mqtt/supercap/
       voltage/phase1", qos: 1}
```

```
    phil/battery/current/measured: {ros_topic: "/ros2mqtt/battery/
        current/measured", qos: 1}
    phil/fuelcell/power/active: {ros_topic: "/ros2mqtt/fuelcell/
        power/active", qos: 1}
    phil/battery/mode: {ros_topic: "/ros2mqtt/battery/mode", qos:
        0}
    phil/supercap/mode: {ros_topic: "/ros2mqtt/supercap/mode", qos
        : 0}
```

2. **Set Up `tc` Traffic Shaping**:
   - Install `iproute2`:

```
sudo apt-get install -y iproute2
```

   - Create and run a `tc` script:

```
nano tc_setup.sh
#!/bin/bash
# Clear existing rules
sudo tc qdisc del dev wwan0 root 2>/dev/null
# Add HTB queue
sudo tc qdisc add dev wwan0 root handle 1: htb default 30
# Parent class: 10mbit total
sudo tc class add dev wwan0 parent 1: classid 1:1 htb rate 10mbit
# Critical MQTT (frequency, voltage, current): 6mbit, high
    priority
sudo tc class add dev wwan0 parent 1:1 classid 1:10 htb rate 6mbit
     ceil 8mbit prio 1
# Power MQTT: 3mbit, medium priority
sudo tc class add dev wwan0 parent 1:1 classid 1:20 htb rate 3mbit
     ceil 8mbit prio 2
# Other traffic: 1mbit, low priority
sudo tc class add dev wwan0 parent 1:1 classid 1:30 htb rate 1mbit
     ceil 2mbit prio 3
# Filter MQTT traffic to critical class
sudo tc filter add dev wwan0 protocol ip parent 1:0 prio 1 u32
    match ip dport 1883 0xffff match ip dst 172.16.11.10 flowid
    1:10
sudo tc filter add dev wwan0 protocol ip parent 1:0 prio 1 u32
    match ip sport 1883 0xffff match ip dst 172.16.11.10 flowid
    1:10
```

```
chmod +x tc_setup.sh
sudo ./tc_setup.sh
```

3. **Preview Custom MQTT Client**:
   - A custom MQTT client with a priority queue will be set up in Step 6 to process critical topics (e.g., frequency) before others.

2.6 Step 6: Set Up MQTT and ROS 2

Objective: Configure Mosquitto, Paho MQTT, and a custom MQTT client for prioritized data exchange.

1. **Install Mosquitto**:

```
sudo apt-get install -y mosquitto mosquitto-clients
sudo systemctl enable mosquitto
sudo systemctl start mosquitto
```

2. **Configure Mosquitto**:

```
sudo nano /etc/mosquitto/conf.d/01-allow-anonymous.conf
listener 1883 0.0.0.0
allow_anonymous true
max_queued_messages 300
max_inflight_messages 30
retry_interval 5
```

```
sudo systemctl restart mosquitto
```

3. **Install Paho MQTT**:

```
sudo apt install -y python3-pip
pip3 install paho-mqtt==1.6.1
```

4. **Set Up Custom MQTT Client**:

   - Create a prioritized MQTT client:

```
nano ~/ros2_ws/src/mqtt_offered_client/mqtt_offered_client/
    mqtt_python_client.py
import paho.mqtt.client as mqtt
import queue
import threading
import time

class PriorityMQTTClient:
    def __init__(self, broker="172.16.11.10", port=1883):
        self.client = mqtt.Client()
        self.client.connect(broker, port)
        self.priority_queue = queue.PriorityQueue()
        self.running = True
        self.thread = threading.Thread(target=self._publish_loop)
        self.thread.start()

    def publish(self, topic, message, qos=0, priority=10):
        if "frequency" in topic:
            priority = 1
        elif "voltage" in topic or "current" in topic:
            priority = 2
        elif "power" in topic:
            priority = 3
        self.priority_queue.put((priority, (topic, message, qos)))

    def _publish_loop(self):
        while self.running:
            if not self.priority_queue.empty():
                priority, (topic, message, qos) = self.
                    priority_queue.get()
                self.client.publish(topic, message, qos=qos)
            time.sleep(0.005)

```

```
    def stop(self):
        self.running = False
        self.thread.join()
```

2.7 Step 7: Configure UDP Server

Objective: Update udp_server to handle PHIL data with prioritized MQTT publishing.

1. **Create Package**:

```
cd ~/ros2_ws/src
ros2 pkg create --build-type ament_python udp_server
cd udp_server/udp_server
nano udp_node.py
```

2. **Update udp_node.py**:

```
from std_msgs.msg import Float32, String
from rclpy.node import Node
from mqtt_offered_client.mqtt_python_client import PriorityMQTTClient

class UDPServerNode(Node):
    def __init__(self):
        super().__init__('udp_server')
        self.mqtt_client = PriorityMQTTClient()
        self.subscriptions = {
            '/ros2mqtt/grid/frequency/measured': self.
               create_subscription(
                Float32, '/ros2mqtt/grid/frequency/measured', self.
                   freq_callback, 10),
            '/ros2mqtt/supercap/voltage/phase1': self.
               create_subscription(
                Float32, '/ros2mqtt/supercap/voltage/phase1', self.
                   volt_callback, 10),
            '/ros2mqtt/battery/mode': self.create_subscription(
                String, '/ros2mqtt/battery/mode', self.mode_callback,
                   10)
        }

    def freq_callback(self, msg):
        self.mqtt_client.publish("phil/grid/frequency/measured", str(
           msg.data), qos=1)
    def volt_callback(self, msg):
        self.mqtt_client.publish("phil/supercap/voltage/phase1", str(
           msg.data), qos=1)
    def mode_callback(self, msg):
        self.mqtt_client.publish("phil/battery/mode", str(msg.data),
           qos=0)
```

2.8 Step 8: Optimize Network Settings

Objective: Ensure stable IP configuration and network prioritization.

1. **Set Static IP**:

```
sudo nano /etc/netplan/50-cloud-init.yaml
network:
```

```
  version: 2
  renderer: networkd
  ethernets:
    eth0:
      dhcp4: no
      addresses:
        - 172.16.11.10/24
      gateway4: 172.16.11.1
      nameservers:
        addresses:
          - 8.8.8.8
          - 8.8.4.4
```

```
sudo netplan apply
```

2. **Configure `wwan0`**:

```
sudo ip addr del 192.168.6.21/30 dev wwan0
sudo ip addr add 192.168.6.21/28 dev wwan0
sudo ip route add default via 192.168.6.20 dev wwan0
sudo nano /etc/rc.local
while ! ip link show wwan0 >/dev/null 2>&1; do sleep 2; done
IP=$(ip -4 addr show wwan0 | grep -oP '(?<=inet\s)\d+(\.\d+){3}/\d+'
    || echo "")
[ -n "$IP" ] && [ "$(echo "$IP" | cut -d'/' -f2)" != "28" ] && sudo ip
    addr del "$IP" dev wwan0
sudo ip addr add 192.168.6.21/28 dev wwan0
/home/ubuntu/tc_setup.sh
```

3. **Open Ports**:

```
sudo ufw allow 1883/tcp
sudo ufw allow 7400:7600/udp
```

2.9 Step 9: Test and Validate

Objective: Verify 5G, MQTT, and prioritization.

1. **Test 5G**:

```
ip a
ping 192.168.6.21 -I wwan0
sudo minicom -D /dev/ttyUSB2
AT+CSQ
AT+C5GREG?
```

2. **Test MQTT**:

```
# Broker
mosquitto_pub -h 172.16.11.10 -t "phil/grid/frequency/measured" -m
    "50.0" -q 1
mosquitto_pub -h 172.16.11.10 -t "phil/battery/mode" -m "auto" -q 0
# Client
mosquitto_sub -h 172.16.11.10 -t "phil/#" -v
ros2 topic echo /ros2mqtt/grid/frequency/measured
```

3. **Test UDP-to-MQTT**:

- Send voltage from Simulink (172.16.11.70:7000).
-
```
ros2 topic echo /ros2mqtt/supercap/voltage/phase1
ros2 topic echo /mqtt2ros/supercap/voltage/phase1
```

4. **Test Prioritization**:
   - Simulate load:

```
for i in {1..100}; do mosquitto_pub -h 172.16.11.10 -t "phil/
   fuelcell/power/active" -m "1000" -q 1; done
```

   - Verify frequency data is prioritized:

```
ros2 topic echo /ros2mqtt/grid/frequency/measured
tc -s qdisc show dev wwan0
```

5. **PHIL Test**:
   - Run nodes on broker (172.16.11.10) and client (192.168.6.22).
   - Send voltage from Simulink.
   -

```
ros2 topic echo /mqtt2ros/supercap/voltage/phase1
```

2.10 Step 10: Monitor and Troubleshoot

Objective: Ensure reliability with dynamic monitoring.

1. **Monitor**:

```
sudo systemctl status mosquitto
tail -f /var/log/mosquitto/mosquitto.log
dmesg | grep "qmi_wwan_simcom"
ros2 topic list
```

   - Create a monitoring script:

```
nano monitor_mqtt.sh
#!/bin/bash
LOG_FILE="/var/log/mqtt_monitor.log"
while true; do
    PING=$(ping -c 1 172.16.11.10 -I wwan0 | grep time= | awk '{
       print $7}' | cut -d'=' -f2)
    QUEUE=$(mosquitto_sub -h 172.16.11.10 -t "phil/#" -C 1 -W 1
       2>/dev/null | wc -l)
    echo "$(date): Latency=${PING}ms, Queue=${QUEUE}" >> $LOG_FILE
    if (( $(echo "$PING > 50" | bc -l) )); then
        sudo tc class change dev wwan0 parent 1:1 classid 1:10 htb
            rate 7mbit ceil 9mbit prio 1
    fi
    sleep 10
done
```

```
chmod +x monitor_mqtt.sh
./monitor_mqtt.sh &
```

2. **Troubleshoot**:

- **No `wwan0`**: Reconnect USB, rerun `./simcom-cm`.
- **Poor signal**: Check `AT+CSQ`, reposition antennas.
- **MQTT issues**: Verify port 1883, check `mosquitto.log`.

- **Prioritization issues**: Check `tc -s qdisc`, monitor $\texttt{mqtt}_{monitor.log}$.

3 References

- Waveshare SIM8200EA-M2 5G HAT Wiki: https://www.waveshare.com/wiki/SIM8200EA-M2_5G_HAT
- Project Documents: `installation_simcomdriver_matt_ros_udp.sh`, `Guidelines 5G modules.sh`, `Instructions for operating the PHIL system_V3_final.pdf`