ReadMe (monitor.py - Field-Test Monitoring Script)

OBU RX PCAP Growth Monitoring + GNSS Sampling + Timestamped Outputs

Overview

This document describes monitor.py, a Python monitoring script used during V2X/C-V2X field tests to generate raw time-series data while driving an OBU-equipped vehicle through RSU coverage zones.

During the field test, an IEEE 1609 application (commonly example1609) is run on both the RSU and the OBU to enable packet exchange. While that application is running, monitor.py runs on the in-vehicle laptop and:

- connects to the OBU over Ethernet using SSH/SFTP
- monitors RX PCAP file growth on the OBU (rx.pcap) as a proxy for received bytes
- samples GNSS latitude/longitude from the OBU once per second
- computes speed (mph) from consecutive GNSS points
- generates timestamped raw outputs: metrics_<timestamp>.csv, events_<timestamp>.csv, and rsu_map_<timestamp>.html

1. What the Script Measures (Core Signals)

1.1 RX activity → delta_bytes (primary signal)

The OBU continuously writes received packets into an RX capture file:

- Remote RX file (monitored): /mnt/rw/log/current/rx.pcap

Each second, the script checks the remote file size and computes delta_bytes = current_size − previous_size (clamped to ≥ 0).

Interpretation:

- delta_bytes > 0 → RX file grew during the interval → reception observed
- delta_bytes = 0 → no growth during the interval → no reception observed

In downstream analysis/plots, coverage is typically treated as: coverage ⇔ (delta_bytes > 0).

1.2 GNSS sampling (once per second)

Each loop iteration, the script runs a GNSS sampling command on the OBU (over SSH) and parses latitude/longitude.

- Default command in the script:
  - cd /mnt/rw/example1609 && kinematics-sample-client -a -n1

1.3 Speed estimate (mph)

Speed is computed from consecutive GNSS samples using haversine distance and the polling interval, then converted to mph.

1.4 ENTRY/EXIT events (coverage boundary markers)

The script maintains an internal state machine (OUTSIDE → INSIDE and back). It records ENTRY when reception persists for several consecutive seconds (smoothed), and EXIT when reception is absent for several consecutive seconds (smoothed).

Note: monitor.py computes a smoothed pps-like estimate internally to stabilize ENTRY/EXIT transitions. This is mainly for event triggering; analysis typically emphasizes delta_bytes.

2. Recommended Repository / Folder Structure

Recommended structure after cloning the repo:

```
RSU_Range_Tester/
├── monitor.py
├── README.md
├── requirements.txt          (recommended)
├── outputs/                  (auto-created at runtime if missing)
│   ├── metrics_YYYYmmdd_HHMMSS.csv
│   ├── events_YYYYmmdd_HHMMSS.csv
│   └── rsu_map_YYYYmmdd_HHMMSS.html
```

3. Step-by-Step Procedure (Field Setup → Run Script)

This section follows the operational steps used in the field test.

Step 1 - Assign a static IP to the OBU

1. Power on the OBU.

2. Log into the OBU using PuTTY (serial interface / console access).

3. Check current Ethernet interface configuration:

```
ifconfig eth0
```

4. If no DHCP IPv4 address is assigned, set a temporary static IP on eth0 (example used in the field test):

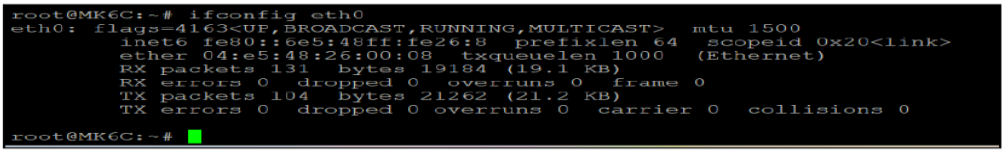
```
ifconfig eth0 192.168.52.80 netmask 255.255.255.0 up
```

This sets a temporary static IP (it can reset after reboot/power cycle).

4.2.1.4. Step-by-Step Implementation Procedure

Step-1: Assign Static IP to OBU

After powering on the OBU and logging into it via PuTTY (serial interface). Executing the command '***ifconfig eth0***' will display the network configuration details like IP address, subnet mask, default gateway and some other network information for the specific interface eth0 (which is the OBUs interface in this case). In the *figure 17*, observe there is no DHCP allocated dynamic IP address (which will generally like 192.X.X.X) that's why we are statically allocating an IP address to make communication using that allocated IP address by executing the command '***ifconfig eth0 192.168.52.80 netmask 255.255.255.0 up***' as illustrated in *figure 18*. This sets a temporary static IP. It will reset upon reboot/power cycle.



Step 2 - Assign a static IP to the laptop Ethernet adapter

On the laptop connected to the OBU via Ethernet:

- Go to Settings → Network & Internet → Ethernet.
- Select the Ethernet interface connected to the OBU.
- Edit IP assignment: set to Manual and enable IPv4.
- Assign the laptop IP (example used in the field test): IP 192.168.52.100, Subnet mask 255.255.255.0 (gateway can be blank).

Step-2: Assign Static IP to Laptop

Inside the laptop that is connected to the OBU, navigate to: *Settings* > *Network & Internet* > *Ethernet*. Select the ethernet interface connection of the laptop for the corresponding OBU. If you observe in the *figure 19*, even though the default setting is set to *Automatic (DHCP)* but no 192.X.X.X (proper IP address has been allocated) by DHCP. Edit IP assignment to *Manual* and set *IP address* as 192.168.52.100 and *Subnet mask* as 255.255.255.0 as shown in the *figure 20*. After saving these new settings, we could now observe the newly assigned IP address for the laptop as illustrated in the *figure 21*.

Figure 19: Navigation to the ethernet network interface of OBU from the laptop settings

Figure 20: Manual allocation of IP address and subnet mask to the laptop

Step 3 - Verify connectivity (must pass before running monitor.py)

From the laptop:

Ping the OBU:

```
ping 192.168.52.80
```

SSH test (if available in your environment):

```
ssh user@192.168.52.80
```

If ping/SSH fails, re-check static IP assignments, Ethernet link status, interface selection (eth0), and whether the OBU IP reset.

Step 4 - Create project folder + virtual environment (venv)

From a terminal (Windows PowerShell / Command Prompt):

Create a project folder and enter it:

```
mkdir RSU  Range  Tester
cd RSU_Range_Tester
```

Create and activate a virtual environment:

- Windows:

```
python -m venv venv
venv\Scripts\activate
```

- macOS/Linux:

```
python3 -m venv venv
source venv/bin/activate
```

Upgrade pip:

```
python -m pip install --upgrade pip
```

Step 5 - Install required libraries

Required packages:

- paramiko (SSH/SFTP)
- folium (HTML map generation)

Recommended requirements.txt content:

```
paramiko
folium
```

Install dependencies:

```
pip install -r requirements.txt
# or
pip install paramiko folium
```

Step 6 - Configure monitor.py for your OBU

Open monitor.py and update these configuration values near the top:

- OBU_HOST — OBU IP (example: 192.168.52.80)
- OBU_USER / OBU_PASSWORD — SSH credentials
- REMOTE_RX_FILE — path to RX capture file on the OBU
- KINEMATICS_CMD — GNSS sampling command (may vary by device)
- OUTPUT_DIR — output folder (default: outputs)

Example configuration:

```
OBU_HOST = "192.168.52.80"
OBU_USER = "user"
OBU_PASSWORD = "user"

REMOTE_RX_FILE = "/mnt/rw/log/current/rx.pcap"
KINEMATICS_CMD = "cd /mnt/rw/example1609 && kinematics-sample-client -a -n1"

OUTPUT_DIR = "outputs"
```

Step 7 - Start the RSU/OBU communication application (example1609)

Before starting monitoring, ensure packet exchange is enabled by starting the IEEE 1609 application (commonly example1609) on both the RSU and the OBU. Keep it running during the drive loop(s). monitor.py does not create the communication traffic; it monitors reception behavior while the application is active.

Step 8 - Run the monitoring script (start logging)

From the repo folder (with venv active):

```
python monitor.py
```

What you will see in the terminal:

- Connection logs (e.g., Connecting to <OBU_HOST>...)
- Periodic status/info logs
- EVENT logs when ENTRY/EXIT is detected
- Map update logs when events are recorded

To stop logging, press Ctrl + C to terminate safely.

4. Outputs

All outputs are stored in outputs/ with a timestamp suffix (YYYYmmdd_HHMMSS).

4.1 metrics_<timestamp>.csv (raw per-second log)

A time-series table sampled at approximately 1 Hz that contains:

- timestamp
- rx_size (remote rx.pcap file size in bytes)
- delta_bytes (file growth during last interval)
- pps (estimated/smoothed; not the primary analysis signal)
- pdr (binary indicator in this implementation)
- latitude
- longitude
- speed_mph

4.2 events_<timestamp>.csv (ENTRY/EXIT markers)

An event table created when coverage transitions are detected:

- timestamp
- event_type (ENTRY or EXIT)
- reason (why the event fired)
- latitude
- longitude

4.3 rsu_map_<timestamp>.html (event map)

An interactive map showing event markers:

- Green marker = ENTRY
- Red marker = EXIT

Open it in a browser by double-clicking the file or using Open with → Browser.

5. Common Troubleshooting

Cannot connect to the OBU

- Confirm laptop Ethernet IP is in the same subnet as the OBU.
- Confirm OBU static IP is set (temporary IP resets after reboot).
- Confirm you can ping the OBU IP.
- Confirm SSH credentials are correct.

GNSS latitude/longitude is empty

- Ensure GNSS antenna is connected and receiving.
- Confirm the command in KINEMATICS_CMD works when run manually over SSH.

rx.pcap does not grow (delta_bytes stays 0)

- Confirm RSU and OBU applications are running (example1609).
- Confirm RSU is deployed and powered.
- Confirm the OBU is within coverage and the comms mode is correct.

6. Notes on Reproducibility and Safety

- This script writes raw test artifacts (CSV and HTML) that may include sensitive location traces. Store and share outputs according to your project's data handling policy.
- Avoid committing passwords in public repositories. If needed, replace OBU_PASSWORD with an environment variable or prompt-based input.

7. Pipeline Context (Recommended)

This script is part of a field-test pipeline:

1. monitor.py generates raw metrics/events/map during the drive.

2. A post-processing script (e.g., analyze_loop.py) converts raw logs into metrics_enhanced.csv for plotting and analysis.