Here’s the complete, ready-to-run Python code for the RTK-GPS rover navigation software (simulated and hardware-ready versions). It includes:

Mock GPS Server (for testing without hardware).

GPS Data Ingestion Client (works with real Emlid Reach M2 or mock server).

Data Logging & Visualization.

File 1: mock\_gps\_server.py

Simulates the Emlid Reach M2 GNSS receiver (run this first for testing).

import socket

import time

import random

def generate\_nmea(lat, lon, fix\_quality=2):

"""Generate a fake GNGGA NMEA sentence with RTK status."""

nmea\_template = (

f"$GNGGA,{time.strftime('%H%M%S')},{abs(lat):09.4f},{'N' if lat >= 0 else 'S'},"

f"{abs(lon):010.4f},{'E' if lon >= 0 else 'W'},{fix\_quality},"

f"{random.randint(8, 12)},1.2,100.0,M,0.0,M,,\*"

)

checksum = calculate\_nmea\_checksum(nmea\_template)

return f"{nmea\_template}{checksum:02X}\r\n"

def calculate\_nmea\_checksum(sentence):

"""Compute NMEA checksum (XOR of bytes between $ and \*)."""

checksum = 0

for char in sentence[sentence.find('$')+1:sentence.find('\*')]:

checksum ^= ord(char)

return checksum

def start\_mock\_server(host='0.0.0.0', port=9001):

"""Run a TCP server to simulate Reach M2."""

with socket.socket(socket.AF\_INET, socket.SOCK\_STREAM) as s:

s.bind((host, port))

s.listen()

print(f"Mock GPS server listening on {host}:{port}")

conn, addr = s.accept()

with conn:

print(f"Client connected: {addr}")

try:

while True:

# Simulate movement in a 10m box around a fixed point

lat = 37.7749 + random.uniform(-0.0001, 0.0001) # ~10m variation

lon = -122.4194 + random.uniform(-0.0001, 0.0001)

fix\_quality = random.choice([1, 2, 4]) # 1=GPS, 2=RTK FIX, 4=RTK FLOAT

nmea = generate\_nmea(lat, lon, fix\_quality)

conn.sendall(nmea.encode())

time.sleep(1) # 1Hz update rate

except KeyboardInterrupt:

print("Server stopped.")

if \_\_name\_\_ == "\_\_main\_\_":

start\_mock\_server()

File 2: gps\_ingestion\_client.py

Connects to either the mock server or real Emlid Reach M2, parses data, and logs it.

import socket

import time

from datetime import datetime

import pandas as pd

import matplotlib.pyplot as plt

class GPSReceiver:

def \_\_init\_\_(self, host='localhost', port=9001):

self.host = host # Use 'localhost' for mock server or Reach M2's IP

self.port = port

self.socket = None

self.log\_data = []

def connect(self):

"""Establish TCP connection."""

try:

self.socket = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

self.socket.connect((self.host, self.port))

print(f"Connected to {self.host}:{self.port}")

except Exception as e:

print(f"Connection failed: {e}")

raise

def parse\_nmea(self, nmea\_sentence):

"""Parse NMEA sentence (e.g., GNGGA for LLH and fix status)."""

if not nmea\_sentence.startswith('$'):

return None

try:

parts = nmea\_sentence.split(',')

if parts[0] == '$GNGGA':

lat = float(parts[2]) / 100

lat\_dir = parts[3]

lon = float(parts[4]) / 100

lon\_dir = parts[5]

fix\_quality = int(parts[6]) # 0=Invalid, 1=GPS, 2=RTK FIX, 4=RTK FLOAT

satellites = int(parts[7])

altitude = float(parts[9])

# Convert to decimal degrees

lat\_dec = lat if lat\_dir == 'N' else -lat

lon\_dec = lon if lon\_dir == 'E' else -lon

return {

'timestamp': datetime.now().strftime('%Y-%m-%d %H:%M:%S'),

'latitude': lat\_dec,

'longitude': lon\_dec,

'altitude': altitude,

'fix\_quality': fix\_quality,

'satellites': satellites,

'raw\_sentence': nmea\_sentence.strip()

}

except Exception as e:

print(f"Failed to parse NMEA: {e}")

return None

def read\_data(self):

"""Continuously read and parse data."""

buffer = ""

while True:

try:

data = self.socket.recv(1024).decode('ascii', errors='ignore')

if not data:

print("Connection lost. Reconnecting...")

self.connect()

continue

buffer += data

lines = buffer.split('\n')

for line in lines[:-1]:

parsed = self.parse\_nmea(line)

if parsed:

self.log\_data.append(parsed)

self.monitor\_rtk\_status(parsed['fix\_quality'])

print(f"LLH: {parsed['latitude']:.6f}, {parsed['longitude']:.6f} | Fix: {parsed['fix\_quality']}")

buffer = lines[-1]

except (socket.error, KeyboardInterrupt) as e:

print(f"Error: {e}. Stopping...")

break

def monitor\_rtk\_status(self, fix\_quality):

"""Alert if RTK status degrades."""

if fix\_quality == 2:

print("RTK FIX: High precision (<5 cm)")

elif fix\_quality == 4:

print("RTK FLOAT: Medium precision (~20 cm)")

else:

print(f"WARNING: Low precision (Fix quality: {fix\_quality})")

def save\_logs(self, filename='gps\_log.csv'):

"""Save logs to CSV and plot trajectory."""

df = pd.DataFrame(self.log\_data)

df.to\_csv(filename, index=False)

print(f"Logs saved to {filename}")

# Plot trajectory

plt.scatter(df['longitude'], df['latitude'], c=df['fix\_quality'], cmap='viridis')

plt.xlabel("Longitude"); plt.ylabel("Latitude")

plt.colorbar(label='Fix Quality (2=RTK FIX)')

plt.title("Rover Trajectory")

plt.savefig('trajectory\_plot.png')

plt.show()

if \_\_name\_\_ == "\_\_main\_\_":

gps = GPSReceiver(host='localhost') # Change to Reach M2's IP for real hardware

gps.connect()

try:

gps.read\_data()

finally:

gps.save\_logs()

How to Use:

For Simulation:

Run mock\_gps\_server.py in one terminal.

Run gps\_ingestion\_client.py in another terminal.

The client will connect to the mock server and log synthetic data.

For Real Hardware:

Replace host='localhost' in gps\_ingestion\_client.py with the Emlid Reach M2’s IP.

Ensure the Reach M2 is streaming NMEA data over TCP (port 9001).

Outputs:

gps\_log.csv: Timestamped GPS data.

trajectory\_plot.png: Visualization of coordinates (color-coded by RTK status).

Dependencies:

bash

Copy

pip install pandas matplotlib

Let me know if you'd like to extend this (e.g., add motor control or QGIS path integration)!