Remotely Controlled Car via LTE or Wifi

Group Members: Nicholas Shenk, Joseph Digafe, Christian Prieto, Donoven Nicolas

Four Core Features (CS/SE Focus) RC CAR

1. Secure, Low-Latency Video + Control Channel

- Real-time FPV video with adaptive bitrate and < 300ms latency.
  Encrypted, authenticated commands and telemetry using a custom lightweight protocol (with standard crypto primitives).
  Resilience against packet loss, replay, and disconnects.

2. Operator Application (Laptop UI)

- Unified interface for driving: live video feed + gamepad/wheel inputs.Connection stats (latency, jitter, loss, encryption status).“Panic stop” button and dead-man timeout for safety.

3. Connection Resilience & Failover Logic

- Automatic reconnect after network dropouts (cellular/Wi-Fi). Adaptive control rates + video quality adjustment based on link conditions. The system is designed to work over any IP network (Wi-Fi, LTE,). In real deployments, LTE data costs may be a consideration for some users, but alternative setup like wifi or other networks is fine. Optional fallback behaviors (stop, return to last safe waypoint). *(RC car hardware = minimal — one FPV camera + LTE module. All complexity is in networking, security, UI, and resilience logic.)*

4. Can be LTE or Wifi

- The choice of LTE or WIFI can be chosen by the user. The performance is measured by a delay that is shown to the user. When using WIFI if connection is lost or range is exceeded the program will try to connect with LTE incase the device has lost wifi connection or range has been exceeded.

Application Using the 3 Features

Application: Remote FPV Rover for Dangerous or Inaccessible Spaces

- **Use Case:** Police, firefighters, or inspectors remotely pilot the rover into unsafe buildings or tight areas (e.g., collapsed structures, hazardous sites).
- **Feature Integration:**
  - Secure, low-latency video gives the operator situational awareness.
  - The operator desktop app provides control, video, and safety tools in one place.
  - Connection resilience ensures the rover maintains usability even under poor network conditions; fails safe if the link is lost.
- **Benefit:** Safer exploration without risking personnel; same software framework could also be used for driver training simulators or entertainment racing platforms.

Libraries and Software Tools

Languages: Python(possible scripting), C++ (for control and networking)

Networking/Streaming: Unix and Windows API for sockets and UDP transmission

Cryptography: We will implement our own secure channel protocol (handshake, key schedule, and authenticated encryption) in C++ for educational purposes, using published algorithms and test vectors

Hardware Libraries: libCamera for Raspberry Pi camera and https://github.com/Xinyuan-LilyGO/T-SIM7600X for cellular capabilities as well as arduino libraries and ESP32 open source libraries like https://github.com/espressif/arduino-esp32

UI: Windows API (C++)

Libraries/APIs: (if sim integration Python), C++ api for video encoding