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 & Benefits

Application: Remote FPV Rover for Dangerous Spaces

- **Use Case:** Police, firefighters, inspectors → explore unsafe/collapsed buildings without risk.
- **Feature Integration:**
  - Secure video provides situational awareness.
  - Operator app unifies control + safety features.
  - Connection resilience ensures usability under poor networks.
- **Benefit:** Safer exploration without risking personnel.
- **Secondary Uses:** Driver training simulators, FPV racing/entertainment.

Libraries and Software Tools

Languages & Tools

- **C++** for networking, crypto, control, UI.
- **Python** for testing/scripting.
- **libCamera** (video), SIM7600X library (LTE), ESP32 libraries (control).
- **Windows API (C++)** for operator UI.

Cryptography

- Custom secure channel in C++: handshake, key schedule, authenticated encryption.
- Alternatives under advisor review (DTLS, TLS/WireGuard).

Technical challenges

- **Low-Latency Video:** Integrating video capture, encoding, and UDP streaming with minimal buffering across unpredictable networks.

- **Cryptographic Implementation:** Designing and implementing secure handshakes, key schedules, and authenticated encryption correctly in C++. Requires careful use of standards and test vectors.

- **Cross-Platform Networking:** Supporting both Unix (car side) and Windows (operator side) introduces portability and debugging challenges with raw sockets and APIs.

Milestone 1 Tooling and Core Demos

- Video capture on car with libCamera, render on laptop.
- UDP send/receive demo between Windows ↔ ESP32.
- Initial Requirements, Design, Test Plan.
- Outcome: Basic end-to-end demo (unencrypted or temporary crypto).

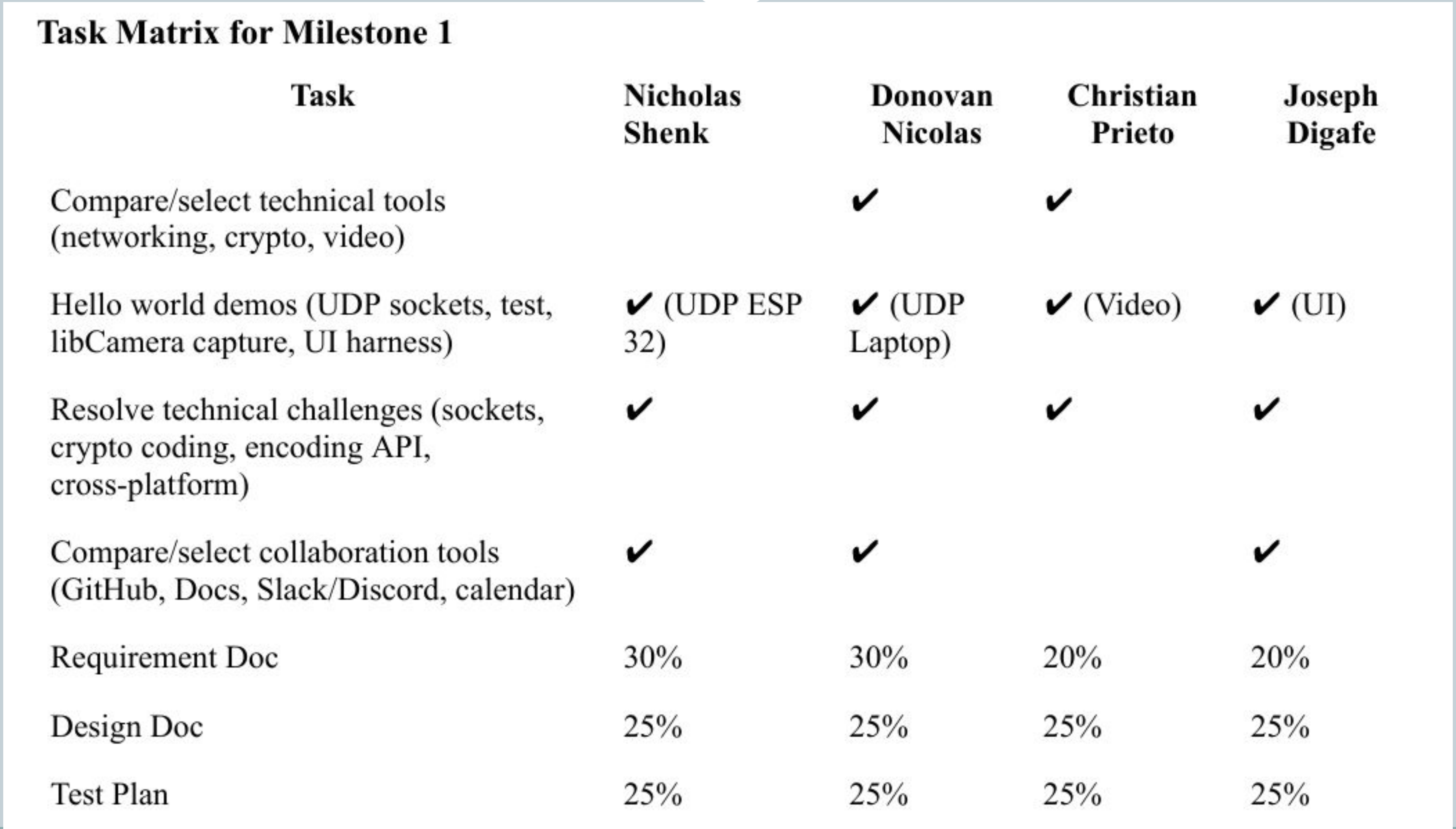
Milestone 2 Secure Channel and Control

- Cryptography decision point with advisor (Options A–D).
- Implement chosen path + document limits (key lifetimes, nonce rules).
- Operator UI foundation: live video + basic stats.
- Outcome: Secure alpha — operator can drive with live video + crypto path selected.

Milestone 3 Reliability & Demo

- Implement resilience: auto-reconnect, Wi-Fi → LTE fallback.
- Adaptive video bitrate + control rates based on link health.
- Telemetry UI: latency, jitter, packet loss.
- Outcome: Robust integrated demo, ready for Spring refinement.

Task Matrix



Task Matrix for Milestone 1

Task	Nicholas Shenk	Donovan Nicolas	Christian Prieto	Joseph Digafe
Compare/select technical tools (networking, crypto, video)		✔	✔	
Hello world demos (UDP sockets, test, libCamera capture, UI harness)	✔ (UDP ESP 32)	✔ (UDP Laptop)	✔ (Video)	✔ (UI)
Resolve technical challenges (sockets, crypto coding, encoding API, cross-platform)	✔	✔	✔	✔
Compare/select collaboration tools (GitHub, Docs, Slack/Discord, calendar)	✔	✔		✔
Requirement Doc	30%	30%	20%	20%
Design Doc	25%	25%	25%	25%
Test Plan	25%	25%	25%	25%

Next Steps (Spring Semester)

Advanced Resilience: Implement seamless NAT traversal, extended failover testing, and multi-network switching.

Optimization: Improve video pipeline performance, refine control responsiveness, and reduce overall system latency.

UI Enhancements: Polish operator application with richer telemetry, improved layout, and customizable controls.

Security Hardening: Validate cryptography design with extended testing; refine key management and replay protection.

Evaluation: Conduct end-to-end field testing; measure performance (latency, jitter, reliability).

Deliverables: Final demo, showcase poster, user/developer manuals, and demonstration video for Senior Design Showcase.