Remotely Controlled Car via LTE or Wi-Fi

• Christian Prieto, cprieto2023@my.fit.edu

• Joseph Digafe, jdigafe@my.fit.edu

• Nicholas Shenk, nshenk2023@my.fit.edu

• Donoven Nicolas, dnicolas2021@my.fit.edu

Faculty Advisor: Marius Silaghi, msilaghi@fit.edu

Progress of current Milestone (progress matrix)

Task	Completion %	Christian	Joseph	Nicholas	Donoven	To do
1. Investigate tools	100%	25%	25%	25%	25%	none
2. Hello World demos	100%	Video	UI harness	UDP ESP32	UDP Laptop	none
3. Requirement Document	100%	20%	20%	30%	30%	none
4. Design Document	100%	25%	25%	25%	25%	none
5. Test Plan	100%	25%	25%	25%	25%	none
6. Implement, test & demo feature/module (UI + telemetry)	50%	0%	50%	0%	0%	wire image-path input; refine metrics
7. Implement, test & demo feature/module (network/video)	30%	15%	0%	15%	0%	implement bitrate adapt; debug UDP jitter

Discussion of each accomplished task:

Task 1: Investigate tools
The team compared networking stacks and evaluated UI frameworks, ultimately selecting Electron with JavaScript for the operator console. We also confirmed the feasibility of both LTE and Wi-Fi transport for end-to-end communication.

Task 2: Hello World demos
We built minimal UDP sender/receiver tests to validate packet flow and connectivity. On the UI

side, an Electron shell was created to render a simple image feed and confirm the operator interface approach.

Task 3: Requirement Document
We documented system goals, user stories, and performance constraints, including the key latency requirement of keeping end-to-end delay under 300 ms.

Task 4: Design Document
An initial architecture was drafted with four main layers—capture, transport, secure channel, and UI. Work is ongoing to finalize the cryptography approach and refine interfaces between modules.

Task 5: Test Plan
Initial test cases were written to measure latency, reconnection handling, and safety mechanisms like the panic stop and dead-man timeout. These will be expanded as features mature.

Task 6: Implement UI + telemetry
We developed an Electron and JavaScript demo that displays video alongside telemetry metrics. The UI shows latency, jitter, loss, and the time-to-display image metric, validating the pipeline end-to-end.

Task 7: Implement network/video
The networking path was prototyped with bitrate control and jitter buffering on top of UDP. Early tests confirmed stability in local conditions, while LTE testing highlighted areas for improvement.

Contributions of each member:

Christian Prieto:
Christian led the video capture work, testing libCamera on the Raspberry Pi and validating adjustable resolution and frame rates. He also contributed to tool selection, helping finalize the choice of Electron and JavaScript for the operator UI. In addition, Christian supported the Requirements and Design Documents by outlining the video subsystem's role, its interfaces, and integration with other modules.

Joseph Digafe:
Joseph built the Electron + JavaScript demo, which displayed video feed and telemetry metrics such as latency, jitter, and delay-to-display. He also authored portions of the Requirements and Test Plan, defining UI needs like controller support, telemetry visibility, and safety features. In the Design Document, Joseph detailed how the UI integrates with networking and telemetry modules, linking hands-on coding to system design.

Nicholas Shenk:
Nicholas developed the ESP32 UDP sender and helped test socket receivers on Windows to confirm packet flow. He contributed to the Requirements Document with networking needs, to the Design Document with control path and motor integration, and to the Test Plan with latency

and packet-loss scenarios. His work grounded the project's communication layer in both practice and documentation.

Donoven Nicolas:
Donoven investigated the Windows UDP harness to validate cross-platform networking and test communication paths. He also supported documentation, helping define resilience and failover in the Requirements Document, reconnection logic in the Design Document, and safety scenarios in the Test Plan. His work connected practical networking tests with well-documented design goals.

Plan for next Milestone

Task	Nicholas	Christian	Joseph	Donoven
Implement a secure channel	Integrate DTLS/AEAD, replay window tests.	Integrate DTLS/AEAD, replay window tests.	Protocol integration tests.	Key/config loader.
Operator UI foundation + controls	Video integration & layout wiring.	Video integration & layout wiring.	Gamepad/wheel loop, dead-man stop.	Failover status; LTE/Wi-Fi toggle.

Discussion of planned tasks:

Task 1: Secure Channel: Select crypto path, integrate handshake, manage keys and nonces.

Task 2: Operator UI + Controls: Integrate live video, finalize control loop with dead-man stop, expand telemetry panel.

Meetings & Feedback

Client meeting: Sep 10, 2025.

Client feedback: see Faculty Advisor feedback.

Faculty Advisor meeting: Sep 2, Oct 1, 2025.

Faculty Advisor feedback:
Task 1: Improvement to JPEG is too hard for this project. Better some parallelization
Task 2: Requirements like 2.1, 4.3 with absolute peerfomence values cannot be
Task 3: theoretically guaranteed without clearly defined environments
Task 4:
Task 5:
Task 6:
Task 7:

Faculty Advisor Signature: [illegible] Date: 10/01/25

Evaluation by Faculty Advisor

Faculty Advisor: detach and return this page to Dr. Chan (HC 209) or email the scores to pkc@cs.fit.edu

Member	0	1	2	3	4	5	5.5	6	6.5	7	7.5	8	8.5	9	9.5	10
Nick Shenk																
Christian Prieto																
Joseph Digafe																
Donoven Nicolas																

Faculty Advisor Signature: ___________________________ Date: __________