Milestone 1

REMOTELY CONTROLLED CAR VIA LTE OR WI-FI

TEAM MEMBERS: CHRISTIAN PRIETO, JOSEPH DIGAFE, NICHOLAS SHENK, DONOVEN NICOLAS

ADVISOR: MARIUS SILAGHI

Overview

- Project: Remotely Controlled Car via LTE or Wi-Fi
- Milestone 1 focuses on tool investigation, initial demos, requirements, and design
- Built Electron + JavaScript UI demo with latency/time-to-display metrics
- Made UDP client and server connection

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	90%	25%	22%	22%	21%	Finalize Designs and add missing diagrams
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

Task 1: Tool Investigation

- Decided on github and google docs for collaborative work
- Evaluated UI frameworks → selected Electron + JavaScript
- Confirmed Wi-Fi and LTE feasibility

Task 2: Hello World Demos

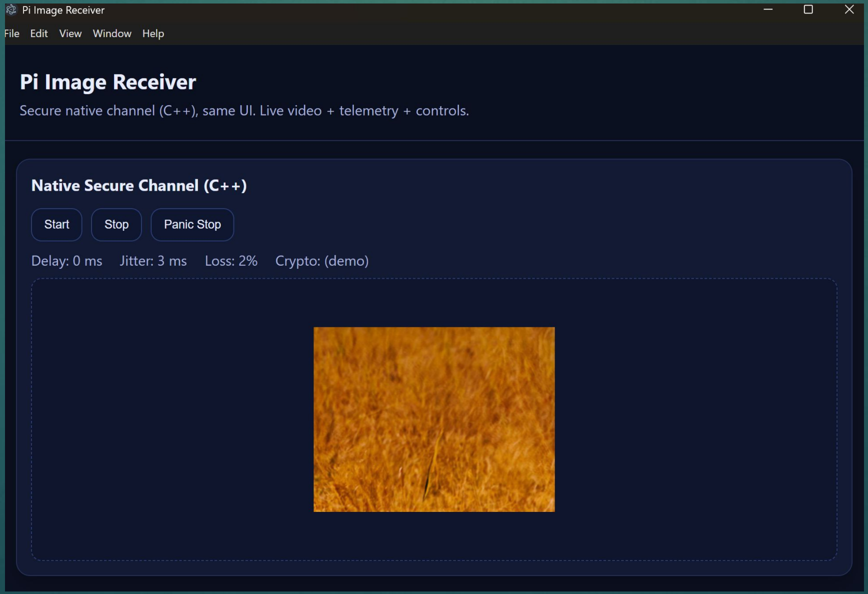
- UDP sender/receiver validated between Windows ↔ ESP32
- Electron shell displayed image feed
- Confirmed baseline operator interface & packet flow

Task 3-5: Docs

- Defined system goals & user stories
- Key latency constraint: <300ms end-to-end
- Captured functional + non-functional requirements
- Drafted 4-layer architecture: Video Capture, Transport, Secure Channel, UI + Telemetry
- Documented module interactions & design rationale
- Defined test cases for latency measurement, reconnection handling, safety mechanisms

Task 6: UI + Telemetry

- Built Electron demo showing live video placeholder
- Displayed metrics: latency, jitter, packet loss, delay-to-display



Pi Image Receiver

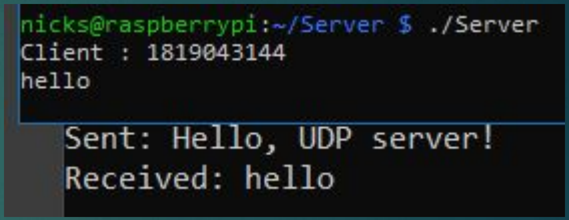
Secure native channel (C++), same UI. Live video + telemetry + controls.

Native Secure Channel (C++)

Delay: 0 ms Jitter: 3 ms Loss: 2% Crypto: (demo)

Task 7: Networking + Video Integration

- Captured image from camera
- Set up UDP connection



```
nicks@raspberrypi:~/Server $ ./Server
Client : 1819043144
hello
```

```
Sent: Hello, UDP server!
Received: hello
```

Milestone 2 Task Matrix

Task	Nicholas Shenk	Christian Prieto	Joseph Digafe	Donoven Nicolas
Secure Channel Implementation	Integrate replay window tests	Integrate DTLS/AEAD, replay window tests	-	-
UI + Raspberry Pi Camera Integration	-	-	Telemetry expansion, controller loop (dead-man stop)	Failover status, LTE/Wi-Fi toggle
Telemetry Expansion	Latency & jitter metrics validation	Bitrate and video stats integration	Encryption status, delay meter in UI	Logging hooks, failover telemetry
Documentation Updates	Update networking/control path sections in Design Doc	Update video subsystem in Design Doc	Update UI + telemetry sections in SRD/Test Plan	Update failover + resilience sections in SRD/Test Plan

Task 1: Secure Channel

- Advisor-guided crypto choice (Custom AEAD, DTLS, TLS/WireGuard)
- Implement handshake, key schedule, replay protection
- Document nonce rules & rekey policy

Task 2: UI + Raspberry Pi Camera Integration

- Connect UI directly to live video stream from Pi camera
- Render encrypted video feed in operator console
- Validate latency and video quality in real conditions

Task 3: Telemetry Expansion

- Extend telemetry panel with latency, bitrate, encryption status
- Prepare data hooks for future failover metrics

Task 4: Documentation Updates

Update Requirements, Design, and Test Plan with:

- Final crypto decision
- Secure channel implementation details
- UI–camera integration flow

Meetings & Feedback

- Advisor meeting: Sep 02, oct 1, 2025 + follow-ups
- Feedback: Improvement to JPEG is too hard for this project. Better some parallelization, requirements and design draft adequate, but Requirements like 2.1, 4.3 with absolute performance values cannot be theoretically guaranteed without clearly defined environments.