Progress Evaluation 6
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
Client: Marius Silaghi, Computer Sciences & Cybersecurity, Florida Institute of Technology

Progress of Current Milestone (Progress Matrix)

Task	Comp %	Christian	Joseph	Nicholas	Donoven	To Do
1. Complete Wi-Fi/LTE Failover	100%	30%	10%	50%	10%	None
2. Adaptive Video Quality	100%	40%	20%	40%	0%	None
3. Senior Design Poster	100%	30%	20%	30%	20%	None
4. Bug Fixes & Finishing Touches	100%	35%	35%	30%	0%	None
5. Formal Evaluation & Testing	100%	30%	20%	20%	30%	None
6. Final System Polish & Presentation Prep	100%	25%	25%	25%	25%	None

Discussion of Accomplished Tasks

Task 1: Complete Wi-Fi/LTE Failover

The Pi-side network handling established in Milestone 5 was extended and stabilized this milestone. The vehicle can communicate through either direct Wi-Fi (192.168.4.20) or through the relay server over LTE, with the Pi automatically attempting reconnection to the relay server every five seconds via a persistent loop in Main.cc. On the server side, the Node.js relay was hardened to handle peer reconnection cleanly, closing stale WebRTC peer connections before establishing new ones to ensure port 43682 remains available..

Task 2: Adaptive Video Quality

Video streaming quality and reliability were improved substantially this milestone. On the Pi, the C++ camera pipeline (Main.cc) uses libcamera with YUV420 capture and TurboJPEG compression across eight worker threads, with a frame-drop mechanism that discards stale frames in favor of the latest available. A five-second inactivity timeout pauses the camera pipeline when no control input is received, reducing unnecessary bandwidth usage. On the browser side, a persistent flicker issue caused by the browser's asynchronous image decode pipeline was resolved by switching from img.src blob URL assignment to ImageBitmap-based rendering via ctx.drawImage, which draws only fully decoded frames. Real-time metrics including FPS, delay, and jitter are collected client-side and fed into live graph widgets in the diagnostics panel.

Task 3: Senior Design Poster

The senior design poster was completed and submitted before the Showcase deadline. The poster covers the full system architecture including LTE and Wi-Fi communication paths, the WebRTC video streaming pipeline, GPS integration and autonomous path retracing, motor control via PWM, and safety features including the kill switch and controller timeout. Each team member contributed content for their respective subsystem.

Task 4: Bug Fixes & Finishing Touches

Several stability and correctness issues were addressed this milestone. The motor controller (controller.py) was validated to correctly stop all motors within one second of losing control input, preventing runaway behavior. GPS data is logged to a CSV file every five seconds during operation for post-session route review. On the UI side, the canvas engine's graph widget was connected to live client metrics, the map widget was corrected to properly resize and reposition tiles after window scaling events, and the diagnostics panel slide animation was stabilized. The client API was also cleaned up to remove broken placeholder functions.

Task 5: Formal Evaluation & Testing

End-to-end system testing was conducted covering video latency, motor responsiveness, and connection stability under both Wi-Fi and LTE conditions. FPS, delay, and jitter were measured using the client-side rolling metric buffers and verified against the real-time graph display. The kill switch behavior was validated by intentionally dropping the control connection and confirming motors halt within the expected timeout window. GPS accuracy was verified through outdoor field testing and cross-checked against the logged CSV coordinates. Remaining evaluation work centered on structured latency measurement methodology as advised by Dr. Silaghi.

Task 6: Final System Polish & Presentation Prep

The system was prepared for the senior design presentation and Showcase. A complete end-to-end demo was rehearsed covering live video streaming, real-time control, GPS map display, and diagnostics. The web UI was tested across multiple devices to verify layout correctness. Documentation was finalized across the three project repositories (raspi, CarUI, UI_Template). The team conducted a long-duration session test to validate stability and confirmed the system operates reliably across the full demo scenario.

Contributions of Each Team Member

Christian Prieto

Refactored the WebRTC browser client into a clean API (connect_Stream, disconnect_Stream, get_stream_url, get_stream_bitmap, send_inputs, get_fps, get_delay, get_jitter, start_client) to integrate with Nick's canvas engine. Diagnosed and resolved the video flicker issue by switching rendering to ImageBitmap-based drawing. Implemented rolling metrics history for the diagnostics graphs. Coordinated relay server stability improvements and assisted with GPS field testing and milestone documentation.

Joseph Digafe

Developed the WebRTC Node.js relay server (server.js) including WebRTC peer connection management, UDP packet forwarding from the Pi to the browser, and control packet routing back to the vehicle. Implemented the initial web UI layout and contributed to the diagnostics panel graph wiring and live metrics display. Performed system-level testing to verify end-to-end command and video delivery.

Nicholas Shenk

Developed the full Raspberry Pi camera and communication stack in C++ (Main.cc) using libcamera, TurboJPEG, and UDP with an 8-thread JPEG compression pipeline. Implemented GPS integration via serial AT commands, chunked UDP frame transmission with sequence and reassembly protocol, and the five-second inactivity timeout. Built the canvas UI engine (script.js) including the graph, map widget, button, switch, and slider widgets, and the full CarUI application layout.

Donoven Nicolas

Continued development of the connection-loss traceback algorithm, refining failure detection logic and improving diagnostic output to surface the root cause of LTE and Wi-Fi disconnection events. Contributed to end-to-end integrated testing and hardware validation, including motor control verification and GPS logging validation during field sessions.

Lessons Learned

Lesson 1: WebRTC is not plug-and-play — signaling state must be managed carefully

Getting WebRTC DataChannels working reliably between a Node.js server and a browser required careful attention to ICE gathering state, offer/answer timing, and which side creates the data channel. The server must create the DataChannel before generating the offer so the SDP includes data channel negotiation; the browser picks it up via ondatachannel. Rushing past this understanding early in the project cost significant debugging time.

Lesson 2: UDP packet reassembly must account for frame interleaving and stale packets

Because UDP delivers packets out of order and across multiple frames simultaneously, the reassembly logic on both the Pi sender and browser receiver must explicitly track frame IDs and drop packets from older frames when a newer frame begins arriving. Without this, corrupted frames accumulate silently, causing visual artifacts and wasted bandwidth.

Lesson 3: Browser image rendering has hidden latency — ImageBitmap eliminates it

Setting img.src to a new blob URL causes the browser to decode the image asynchronously. During that decode window, img.complete is false and the canvas engine skips drawing, producing a blank frame every time a new video frame arrives. Pre-decoding with createImageBitmap before storing the frame ensures ctx.drawImage is always instantaneous, eliminating the flicker entirely.

Lesson 4: Separating subsystem APIs early enables parallel development

Defining a clean boundary between the networking layer (client.js) and the rendering layer (script.js / index.html) early in the milestone allowed the UI and networking code to be developed and debugged independently. The explicit API surface (send_inputs, get_stream_bitmap, get_fps, etc.) meant changes on one side did not break the other, and integration was straightforward.

Lesson 5: Hardware safety features must be implemented at the lowest level possible

The motor kill switch is implemented directly in controller.py on the Pi, not in the web UI or relay server. This ensures motors stop within one second of losing control input regardless of what happens to the network connection, browser, or relay. Relying on higher-level software for safety would create a single point of failure in a system with an LTE hop between the operator and the vehicle.

Lesson 6: Version coordination across multiple active contributors requires discipline

With three active code repositories (raspi, CarUI, UI_Template as a submodule) and multiple team members pushing simultaneously, merge conflicts and integration breaks were a recurring cost. Establishing clearer branch conventions and communication around push timing would have reduced the overhead of repeatedly re-applying changes on top of each other's updates across the final milestone.

Meetings & Feedback

Date(s) of meeting(s) with Client during the current milestone: _____

Client feedback on the current milestone: See Faculty Advisor feedback below.

Date(s) of meeting(s) with Faculty Advisor during the current milestone: _____

Faculty Advisor feedback on each task for the current milestone: Marius Silaghi

Task 1:

Task 2:

Task 3:

Task 4:

Task 5:

Task 6:

Faculty Advisor Signature: Marius Silaghi Date: _____

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

Score (0-10) for each member: circle a score (or circle two adjacent scores for .25 or write down a real number between 0 and 10)

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

Faculty Advisor Signature: _____ Date: _____