Politecnico di Torino

N-MON

PRELIMINARY DESIGN REVIEW

Presented by Group - G

Braia Lorenzo Eustachio - s346316@studenti.polito.it

Cavallaro Lorenzo - s346742@studenti.polito.it

Peradotto Simone - s343420@studenti.polito.it

OVERVIEW

- Objectives
- Requirements
- Architecture
- WBS
- Gantt Chart

OBJECTIVES

Main Goal

- To develop a compact and cost-effective monitoring system that detects and classifies GNSS jamming interference in real-time to protect high-value assets during transit.
- To provide an autonomous system that enhances the security of fleet management by immediately alerting stakeholders of jamming events, allowing for rapid response.

Specific Objectives

- Develop a portable monitoring station deployable on vehicles to continuously analyze GNSS frequency bands.
- Integrate advanced signal processing and a Artificial Intelligence Algorithms to accurately identify and classify different types of jamming signals
- Create a user-friendly interface to provide clear, immediate visualization of the GNSS signal status
- Ensure an automatic alert to a central control station the moment jamming is detected

USER REQUIREMENTS

UREQ_1	The user shall be able to deploy and operate the system simply by connecting power and network, without complex configuration procedures
UREQ_2 *	The Control Center shall have a simple, intuitive way to see if GNSS signals are clean or compromised, and what type of interference is disrupting them
UREQ_3 *	Automatic warning shall be delivered to the user when jamming is detected

* minor changes from the SRR, and already corrected in the SysReq document

FUNCTIONAL REQUIREMENTS

The system shall be able to perform the following operations:

FREQ_1	continuously monitor GNSS signals (L band) multi-constellation
FREQ_2	acquire and then process GNSS signals to extract time-frequency representations
FREQ_3	integrate Artificial Intelligence to correctly identify and classify jamming interference across multiple interference types with different power of jamming signals JSR
FREQ_4	provide the user a simple UI to visualize the output of the AI algorithm and signal status
FREQ_5	automatically send an immediate warning if jamming is present to both the UI and a control center

TECHNICAL REQUIREMENTS

The system will need the following technologies and tools to be implemented :

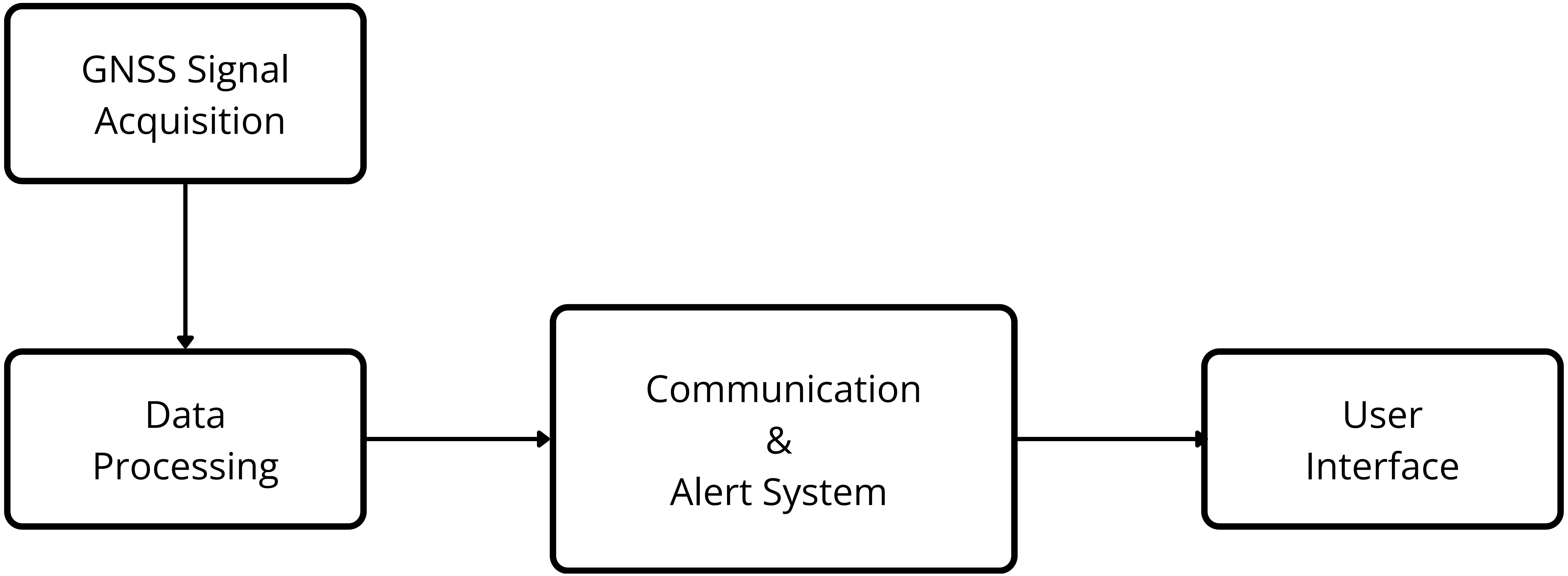
TREQ_1	Multi-band GNSS patch antenna suitable for vehicle-mounted applications, providing reliable signal reception in variable conditions
TREQ_2	An SDR-based RF front-end capable of capturing GNSS signals, with an ADC that provides sufficient sampling rate and resolution for high-quality GNSS signal processing
TREQ_3	computational unit supporting real time AI inference and signal processing operations
TREQ_4	signal processing pipeline to elaborate input data using STFT

TECHNICAL REQUIREMENTS

TREQ_5	test environment with capability to generate synthetic jamming signals with different power of jamming signals JSR
TREQ_6	communication protocols to reliably transmit alerts from the device to remote control station
TREQ_7	web based UI providing real-time status updates
TREQ_8	target cost lower than current laboratory setup
TREQ_9	reduced dimensions compared to existing laboratory setup

SysReq document here

HIGH-LEVEL ARCHITECTURE



PATCH ANTENNA

Model GPS P1MAM

Technical Specification

- Operating Frequency: T1 1575.42 ± 1.023 MHz
- Bandwidth: 10 MHz with S11 ≤ -10 dB (90% of signal is acquired)
- Gain: 28 dB
- Filter: DR Filter to avoid interference from signals in the near spectrum
- Noise Filter: 1.5dB power of the LNA
- Power Supply: 2.3 - 5.5 V ideal for trucks
- Operating Temperature: -30°C to +85°C

Satisfied requirements:
UREQ_1 , UREQ_2 , UREQ_3 , FREQ_1 , FREQ_2 , FREQ_3 , FREQ_4 , FREQ_5 , TREQ_1 , TREQ_2 , TREQ_3 , TREQ_4 , TREQ_5 , TREQ_6 , TREQ_7 , TREQ_8 , TREQ_9

HACKRF ONE

Open-Source SDR Platform

- Full documentation and active community support and compatible with GNU Radio ecosystem. Extensive software libraries available

Wide Frequency Coverage

- Operating range: 1 MHz to 6 GHz , covers al GNSS bands so in the future can be extended to monitor L2/L5

Technical Specifications

- Sampling rate: Up to 20 MS/s and ADC resolution of 8-bit

Cost-Effective

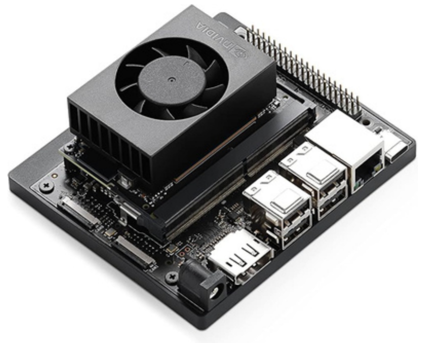
- Price: ~ 300€ (2000€+ for professional SDR equipment)

Low Power Consumption

USB powered: ~2.5W maximum consumption, so suitable for vehicle deployment without dedicated power supply

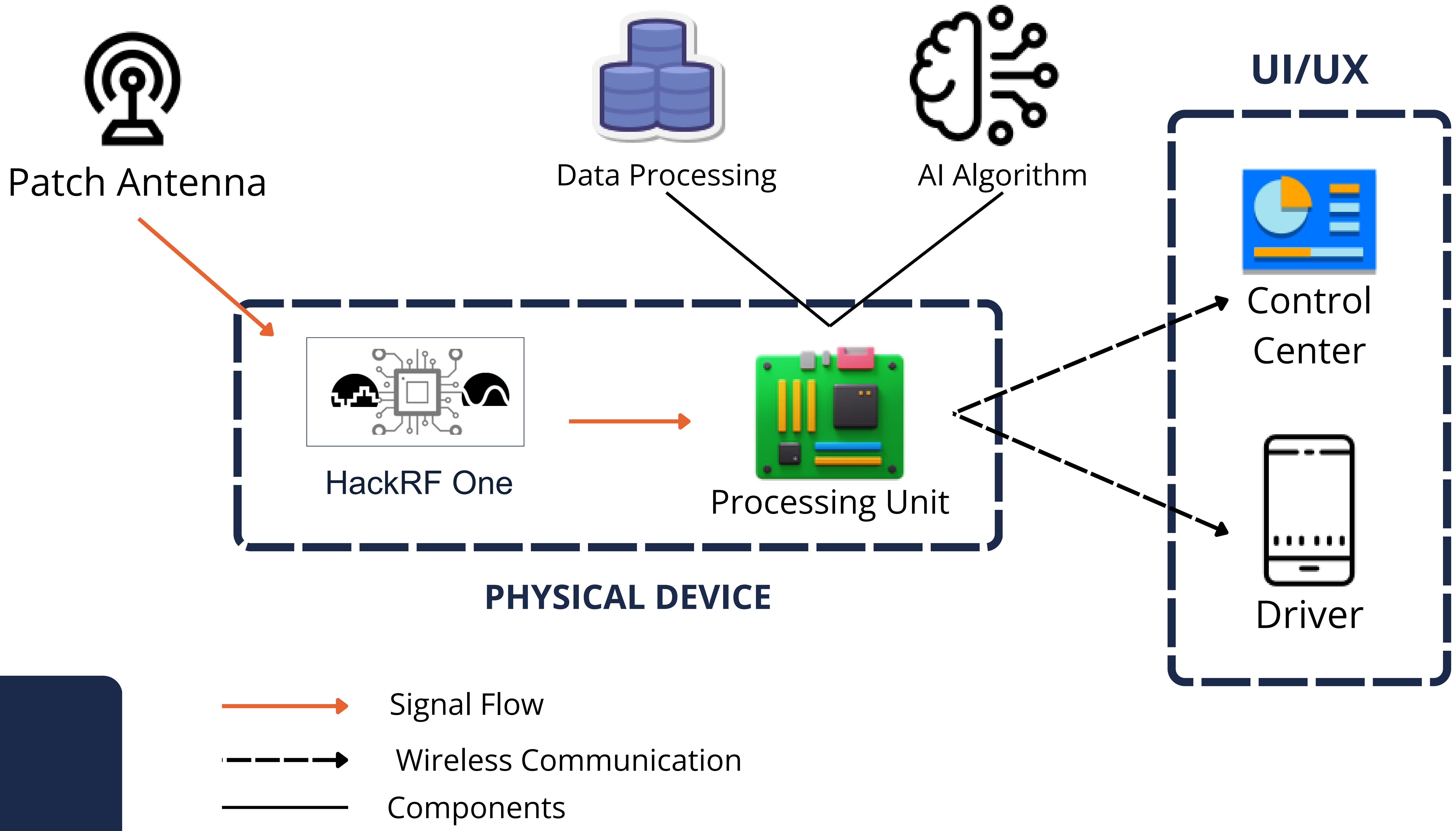
Satisfied requirements:
UREQ_1 , UREQ_2 , UREQ_3 , FREQ_1 , FREQ_2 , FREQ_3 , FREQ_4 , FREQ_5 , TREQ_1 , TREQ_2 , TREQ_3 , TREQ_4 , TREQ_5 , TREQ_6 , TREQ_7 , TREQ_8 , TREQ_9

PROCESSING UNIT

	NVIDIA Jetson Orin Nano	Raspberry Pi 5 - 8GB
		
Architecture	SoM + Carrier Board	SBC
CPU	6-Core	4-Core
NPU/GPU	GPU (40/67 TOPS)	NPU (13/26 TOPS)
RAM	8 GB	8 GB
Storage	NVMe	MicroSD
Size	100mm x 79mm x 21mm	86mm x 56mm x ~35mm
Price	~350€	~250€

Satisfied requirements:
UREQ_1 , UREQ_2 , UREQ_3 , FREQ_1 , FREQ_2 , FREQ_3 , FREQ_4 , FREQ_5 , TREQ_1 , TREQ_2 , TREQ_3 , TREQ_4 , TREQ_5 , TREQ_6 , TREQ_7 , TREQ_8 , TREQ_9

LOW-LEVEL ARCHITECTURE



DATA PROCESSING

1) Signal Acquisition

- HackRF One continuously captures raw IQ samples from L1 GNSS band
- Signal segments: 100 μs snapshots

2) Short-Time Fourier Transform (STFT)

- Converts time-domain IQ samples into time-frequency representation (spectrogram)
- Output: 2D matrix capturing spectral evolution over time

3) AI Model Inference

- Normalized spectrogram fed directly into AI Algorithm network
- Output: classification across 6 classes (5 jamming types [LWF, LN, TRI, TRIW, TICK] + clean signal)

4) Decision & Alert

- If jamming detected trigger MQTT alert

Satisfied requirements:
UREQ_1 , UREQ_2 , UREQ_3 , FREQ_1 , FREQ_2 , FREQ_3 , FREQ_4 , FREQ_5 , TREQ_1 , TREQ_2 , TREQ_3 , TREQ_4 , TREQ_5 , TREQ_6 , TREQ_7 , TREQ_8 , TREQ_9

AI ALGORITHM

	CNN	GRU
Accuracy	99.94%	99.98%
Trainable Parameters	~Millions	~Hundred Thousands
Model Size	~100MB	~ 10 MB
Inference Time	~300ms	~ 60 ms
Required Power	Higher Power Consumption	Low Power
Input Processing	Image Conversion and Normalization	Direct Spectrogram Processing

** Numerica Data taken from Tutor Thesis

AI ALGORITHM

	CNN	GRU
Accuracy	99.94%	99.98%
Trainable Parameters	~Millions	~Hundred Thousands
Model Size	~100MB	~ 10 MB
Inference Time	~300ms	~ 60 ms
Required Power	Higher Power Consumption	Low Power
Input Processing	Image Conversion and Normalization	Direct Spectrogram Processing

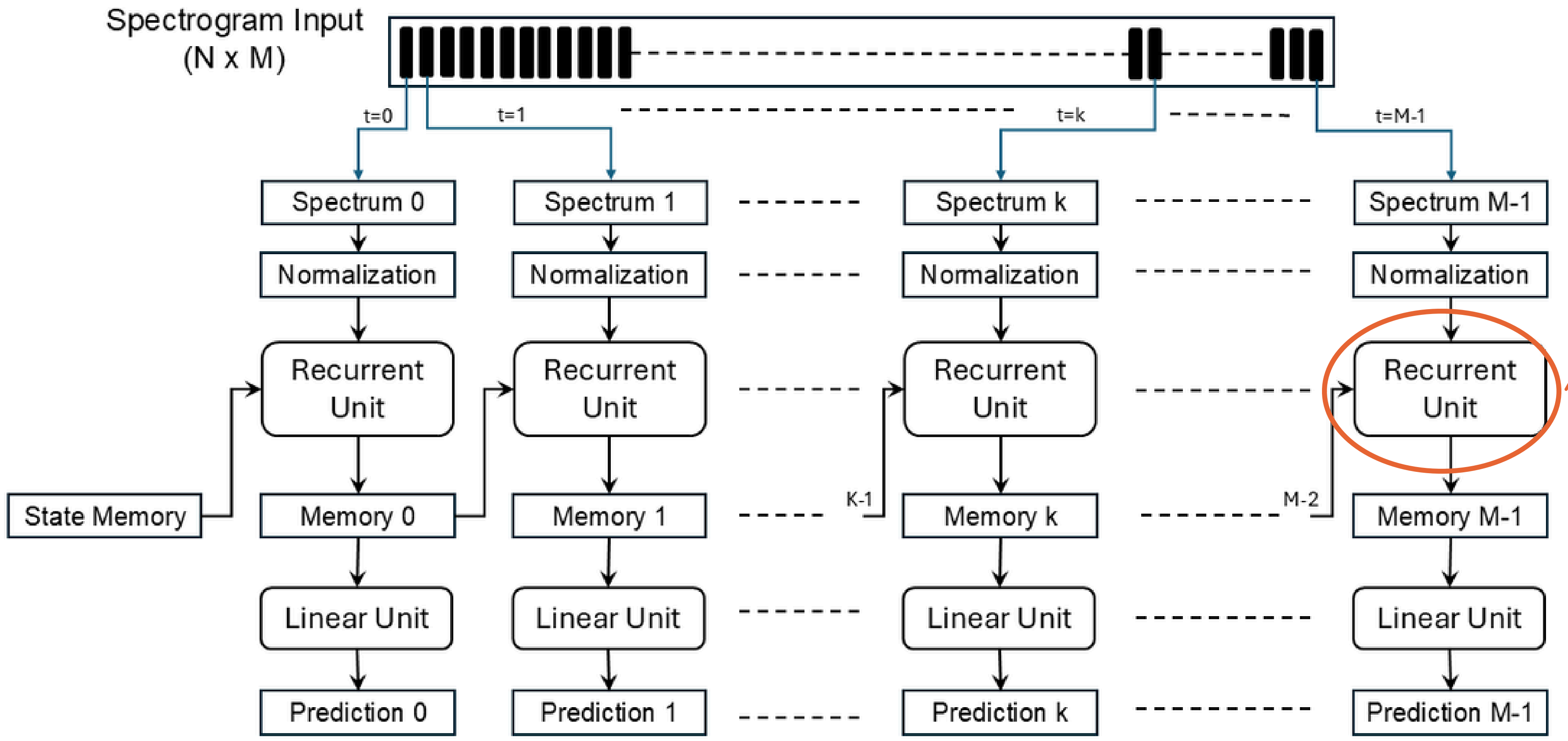
Satisfied requirements:
UREQ_1 , UREQ_2 , UREQ_3 , FREQ_1 , FREQ_2 , FREQ_3 , FREQ_4 , FREQ_5 , TREQ_1 , TREQ_2 , TREQ_3 , TREQ_4 , TREQ_5 , TREQ_6 , TREQ_7 , TREQ_8 , TREQ_9

GRU MODEL

GRU is a type of Recurrent Neural Network (RNN) designed to process sequential data by maintaining "memory" of previous inputs while processing current input. GRU learns how does the spectrum change over time

Spectrograms are inherently sequential: Each column represents frequency content at time t and jamming patterns evolve over time. GRU naturally models this temporal evolution

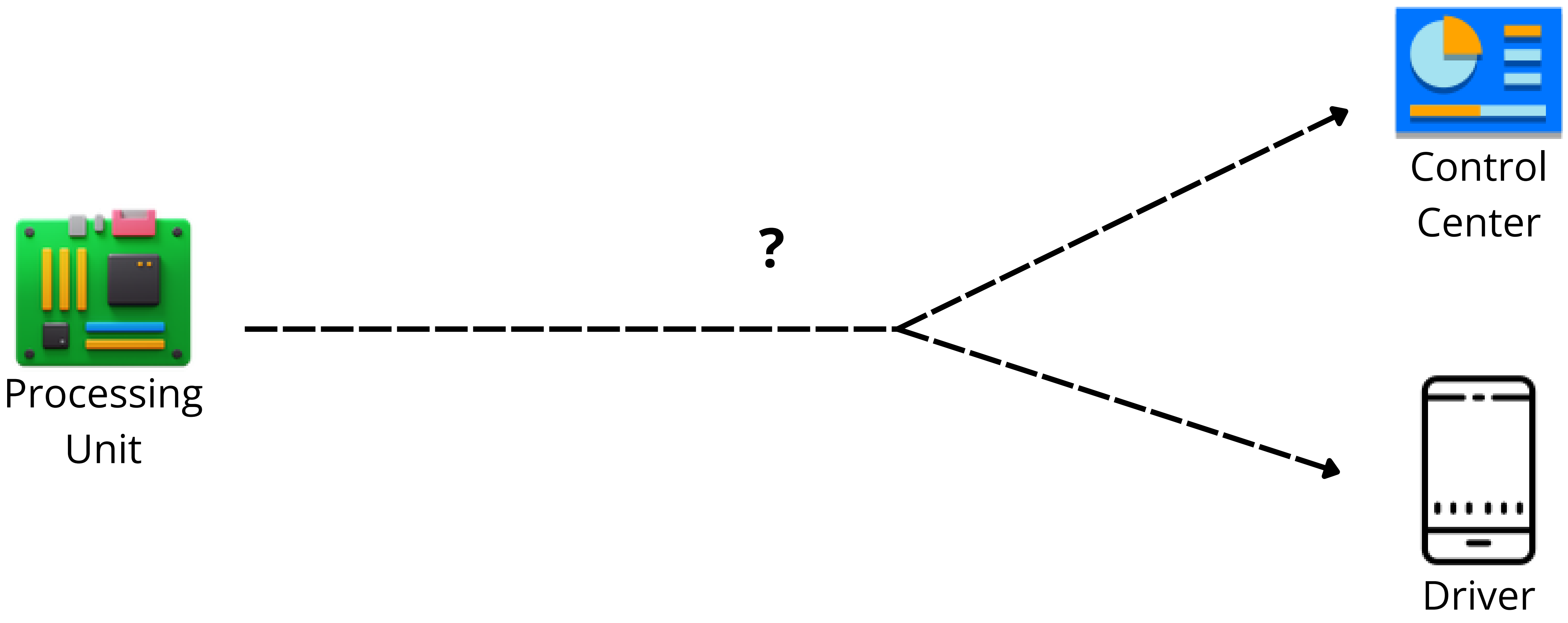
Key Advantage: GRU processes spectrograms as time series while CNN requires conversion to static images



Gated Recurrent Unit (GRU)

- What information to keep.
- What information to update.
- What information to pass forward to the next step.

COMMUNICATION



COMMUNICATION PROTOCOL

	HTTP	MQTT
Paradigm	Request/Response	Publish/Subscribe
Connection Type	Stateless (new connection per request)	Persistent Connection
Overhead	Low (Heavy Headers)	High (Small Headers)
Battery Efficiency	Poor (due to constant polling)	Excellent (event-driven)
Real Time Capability	Polling required	Push-based (instantaneous)
Bandwidth Usage	High	Very Low

MQTT

1) Network Resilience

- Trucks go through tunnels, rural areas, and underground parking
- HTTP: requires full TCP handshake for every packet so fails in zones with poor coverage
- MQTT: maintains lightweight session and auto-reconnects when signal returns

2) Quality of Service (QoS) Levels

- QoS 0: "Fire and forget" for periodic status updates
- QoS 1: "At least once" for Jamming alerts (retry until confirmed)
- QoS 2: "Exactly once"

3) Last Will & Testament (LWT)

- If device loses power/connection
- MQTT broker automatically publishes predefined "DEVICE OFFLINE" message
- Control center immediately knows truck is compromised

Satisfied requirements:

UREQ_1 , UREQ_2 , UREQ_3 , FREQ_1 , FREQ_2 , FREQ_3 , FREQ_4 , FREQ_5 , TREQ_1 , TREQ_2 , TREQ_3 , TREQ_4 , TREQ_5 , TREQ_6 , TREQ_7 , TREQ_8 , TREQ_9

BROKER

HIVEMQ CLOUD

Problem: modern web browsers (Chrome, Safari, Edge) are strictly designed for security. They don't allow web pages to open raw TCP sockets. If we try to connect the UI directly to the standard MQTT port (8883), the browser would block it.
Solution: we wrap the MQTT message inside a WebSocket envelope so that the UI can receive it using HiveMQ that bridges this gap

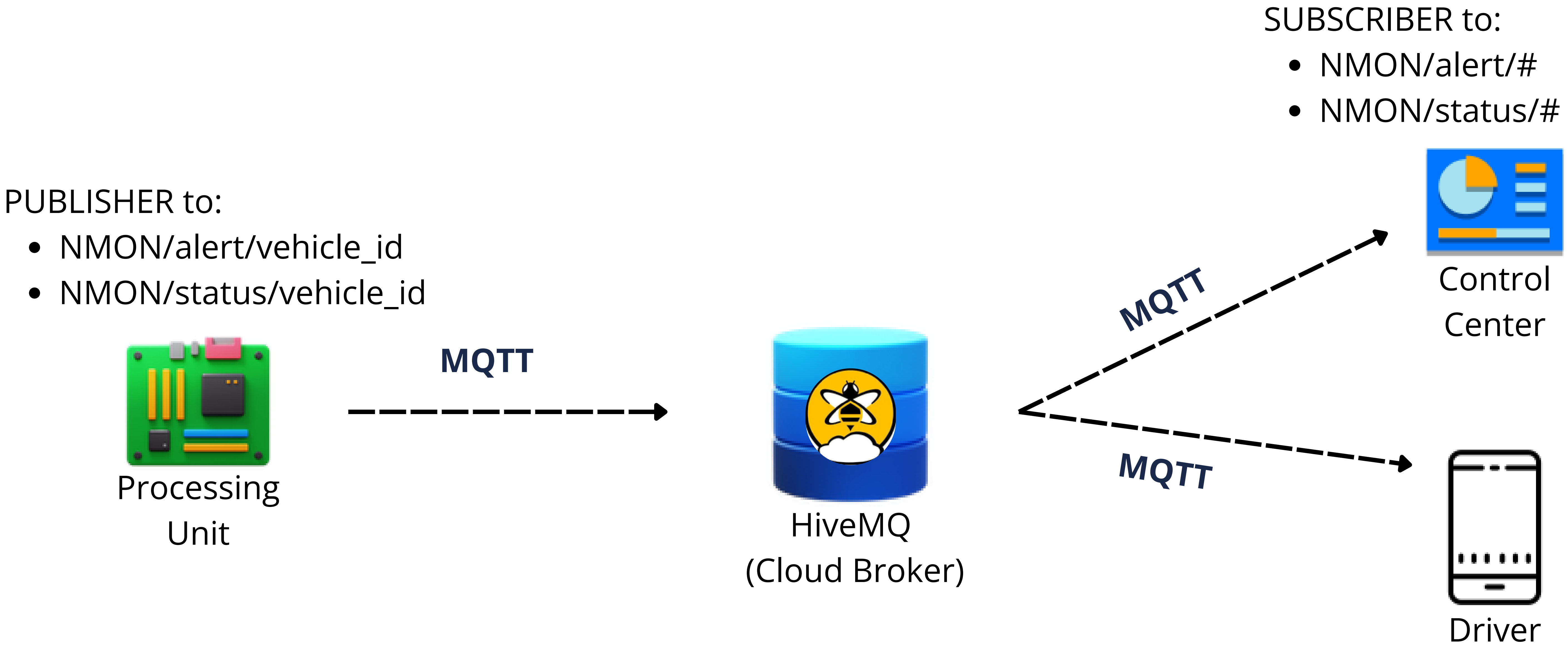
HiveMQ acts as a Translator:

- The Truck sends a packet via TCP to port 8883.
- HiveMQ receives the TCP packet. It removes the TCP headers to expose the raw MQTT payload (JSON data).
- HiveMQ checks who is subscribed. It sees the UIs are subscribed but are connected via WebSockets.
- HiveMQ takes that same MQTT payload, wraps it inside a WebSocket Frame, and pushes it out port 8884 to the Browser

Advantages:

- No need to maintain our own MQTT broker + realistic scenario
- TLS/SSL encryption + username/password authentication
- Handles multiple vehicles without infrastructure changes

COMMUNICATION



vehicle_id is a unique identifier of the vehicle our device is mounted on. In this architecture it's the plate

SUBSCRIBER to:

- NMON/alert/vehicle_id

USER INTERFACE

Streamlit

Control Center UI

- **Technology:** Streamlit (Python) , a framework specifically designed for rapidly building data-focused web applications. It's popular in the data science and machine learning communities because it allows to create interactive dashboards with minimal code
- **Pro:**
  - Fast development
  - Provides a real web server (Tornado) that can be deployed like any production application
  - Designed for analytics, includes built-in components for metrics, charts, and real-time data visualization
  - Can use the more efficient native MQTT protocol over TCP (port 8883)

Mobile UI

HTML CSS JS

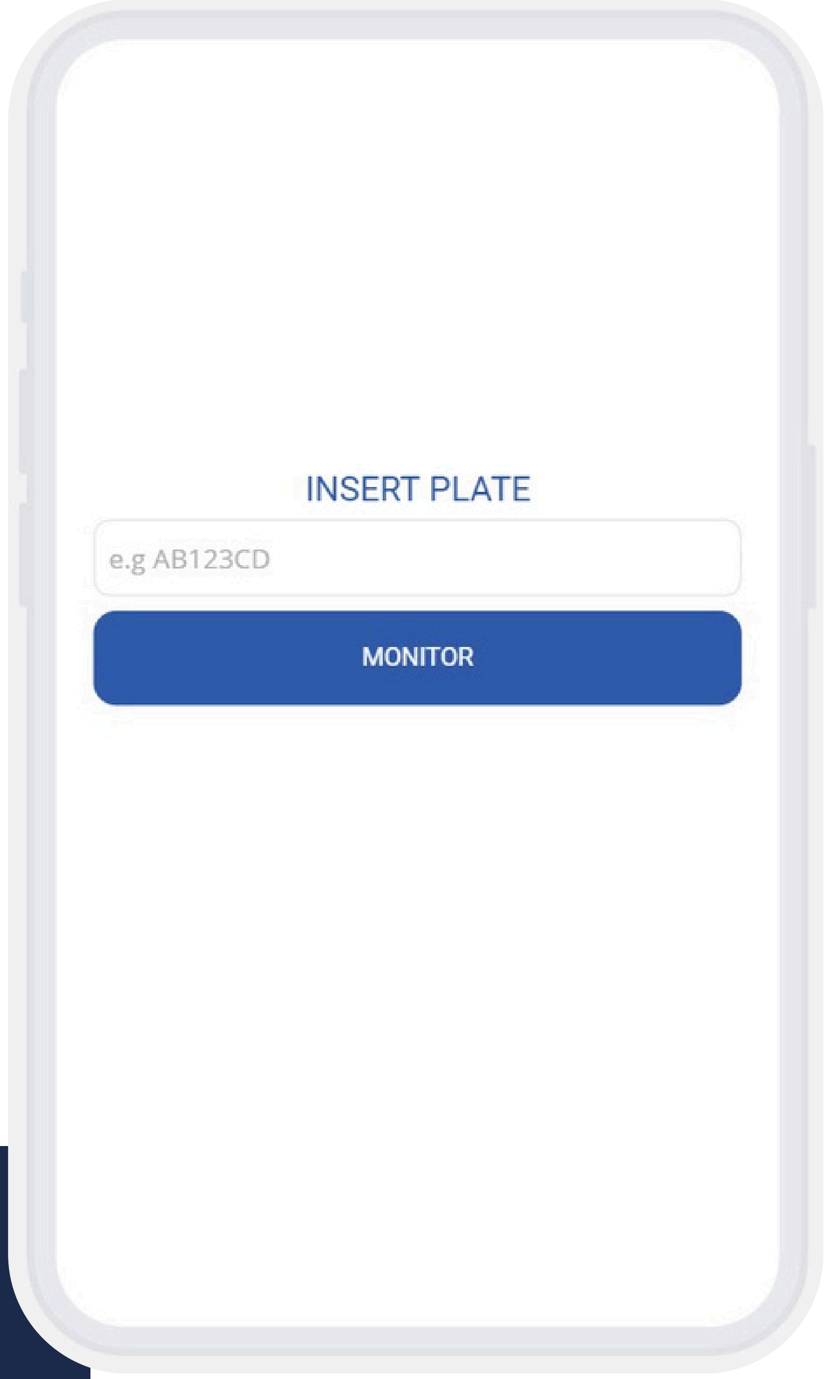
- **Technology:** HTML/CSS + MQTT.js
- **Pro:**
  - Simplicity , no installation required ,works on any phone/tablet
- **Cons:**
  - Web browsers can’t open raw TCP,but can open WebSocket for receiving real-time alerts via HiveMQ (port 8884)

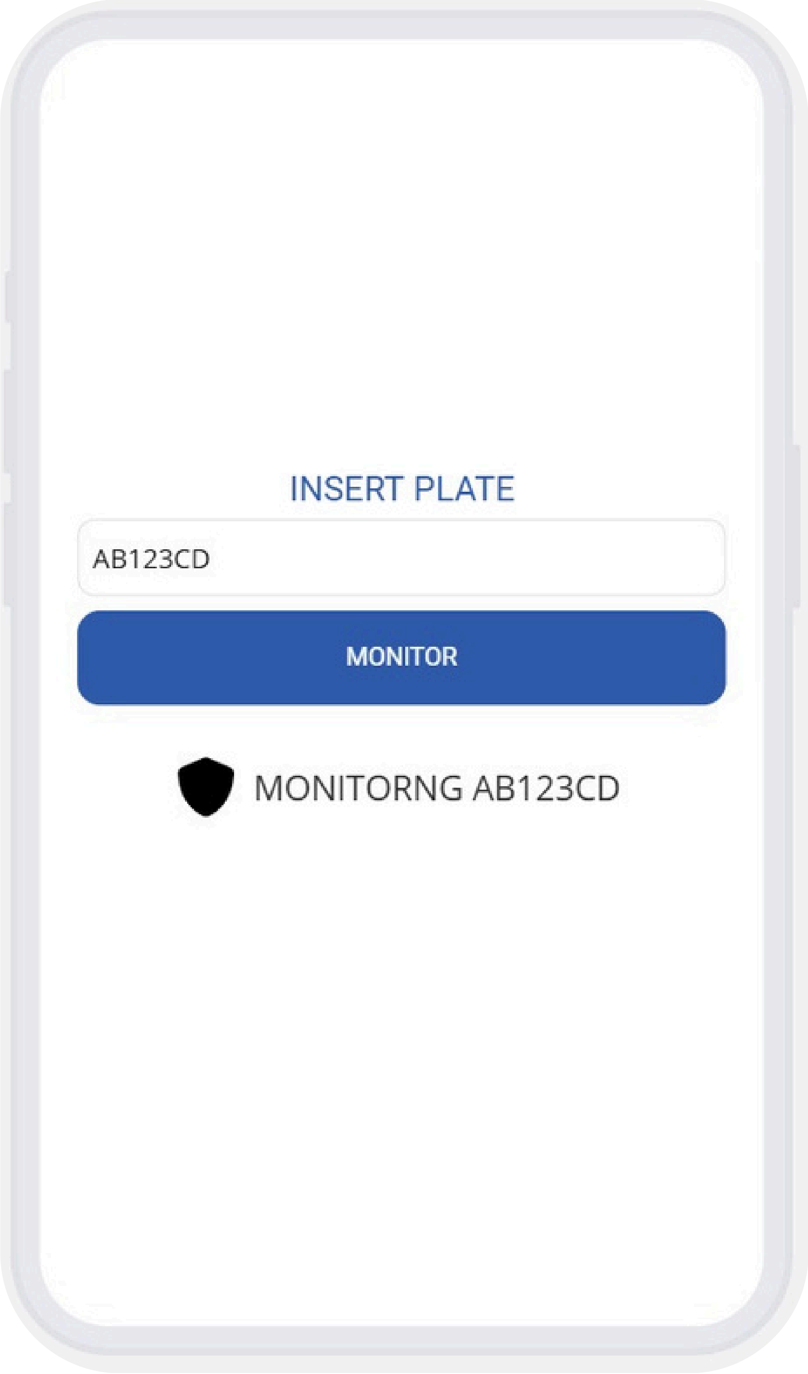
Satisfied requirements:

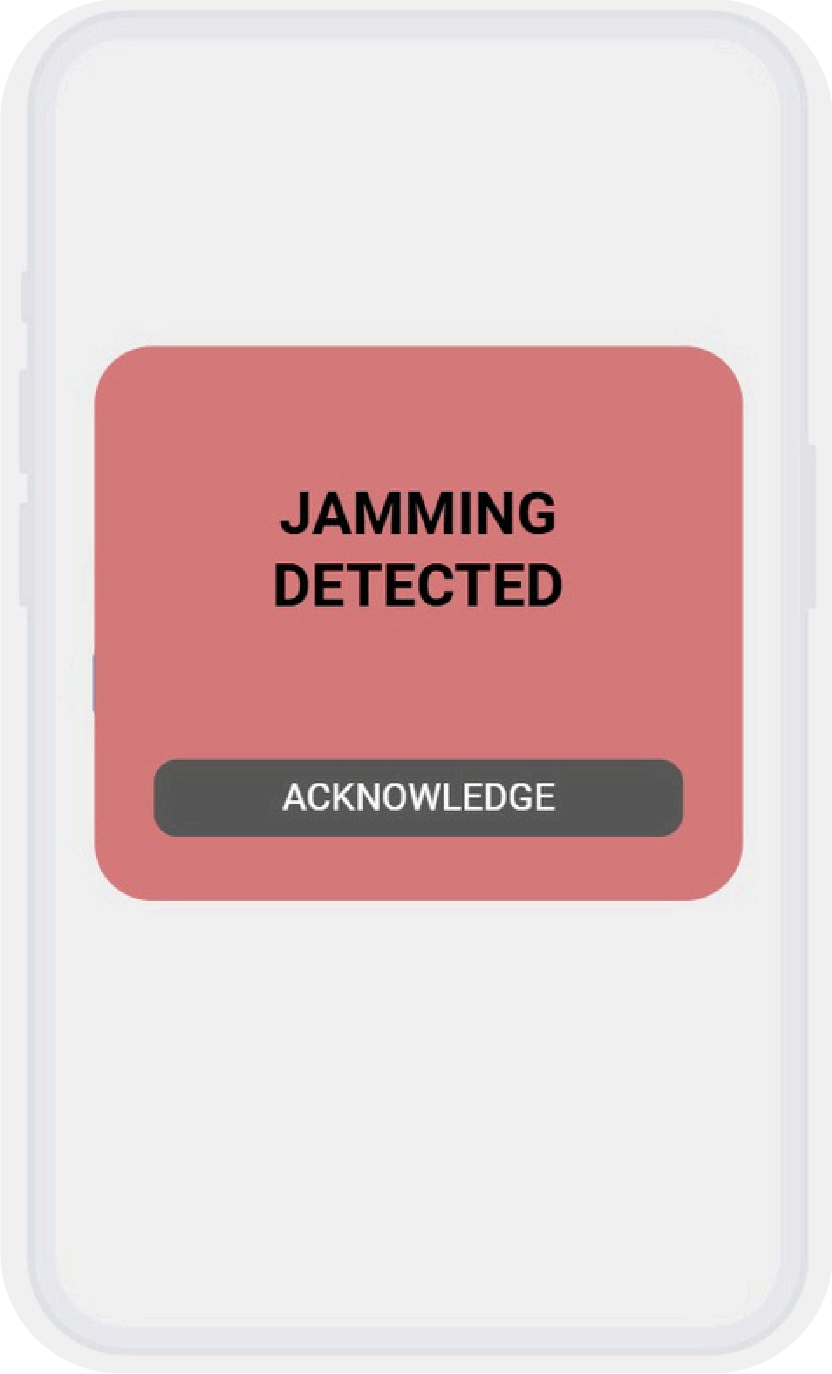
UREQ_1 , UREQ_2 , UREQ_3 , FREQ_1 , FREQ_2 , FREQ_3 , FREQ_4 , FREQ_5 , TREQ_1 , TREQ_2 , TREQ_3 , TREQ_4 , TREQ_5 , TREQ_6 , TREQ_7 , TREQ_8 , TREQ_9

USER INTERFACE

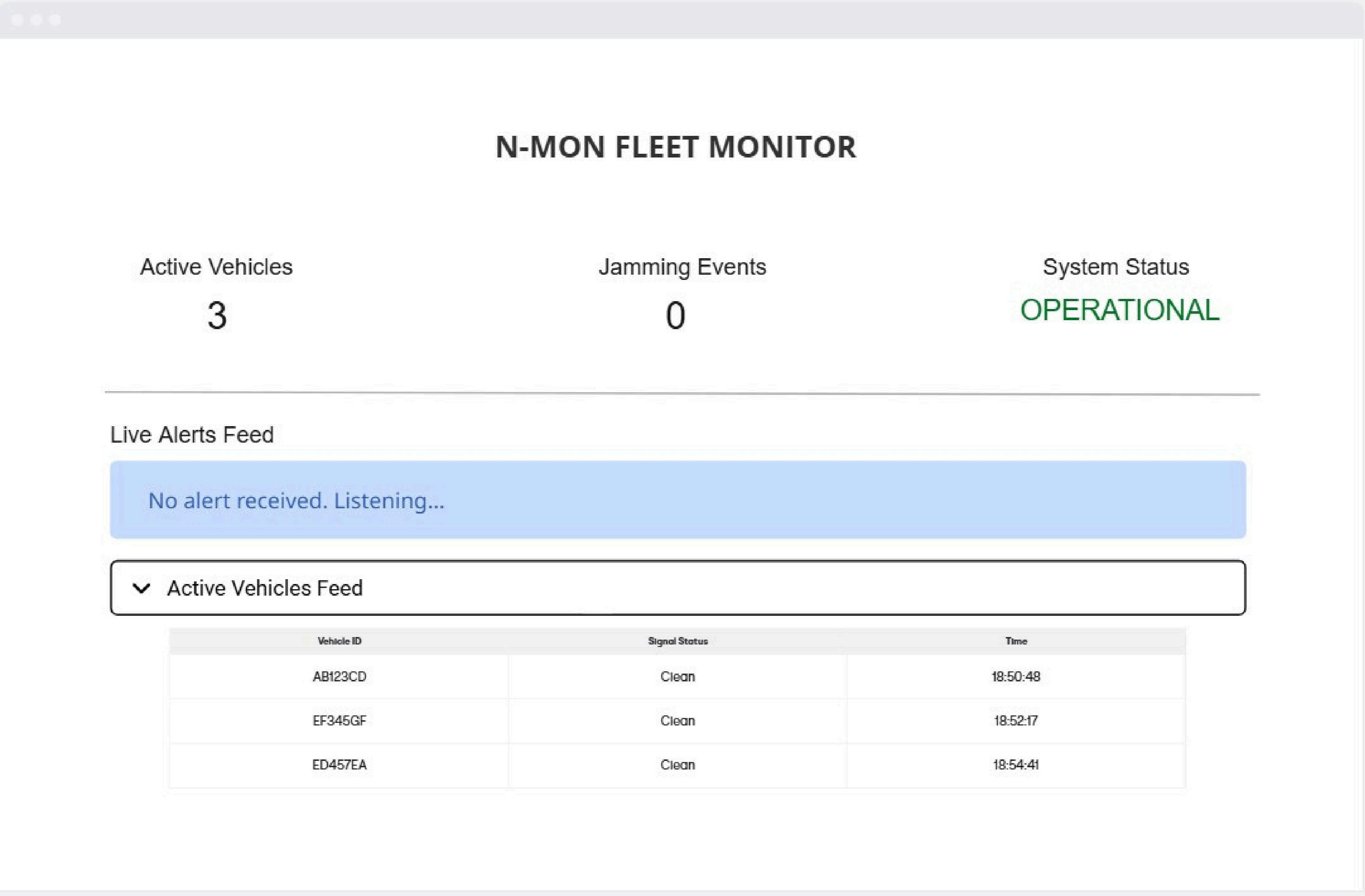
The only action performed by the user is insert the plate

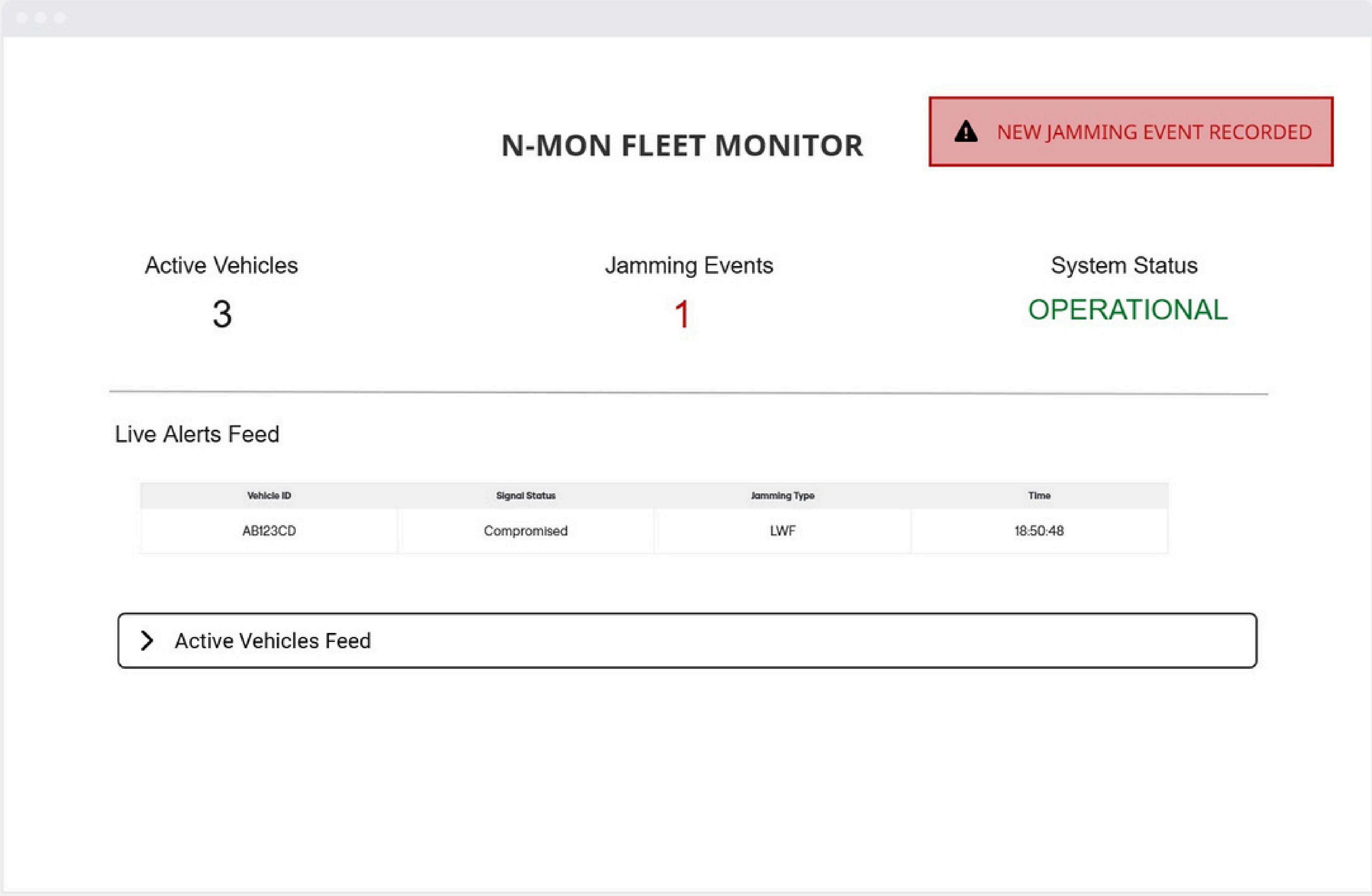




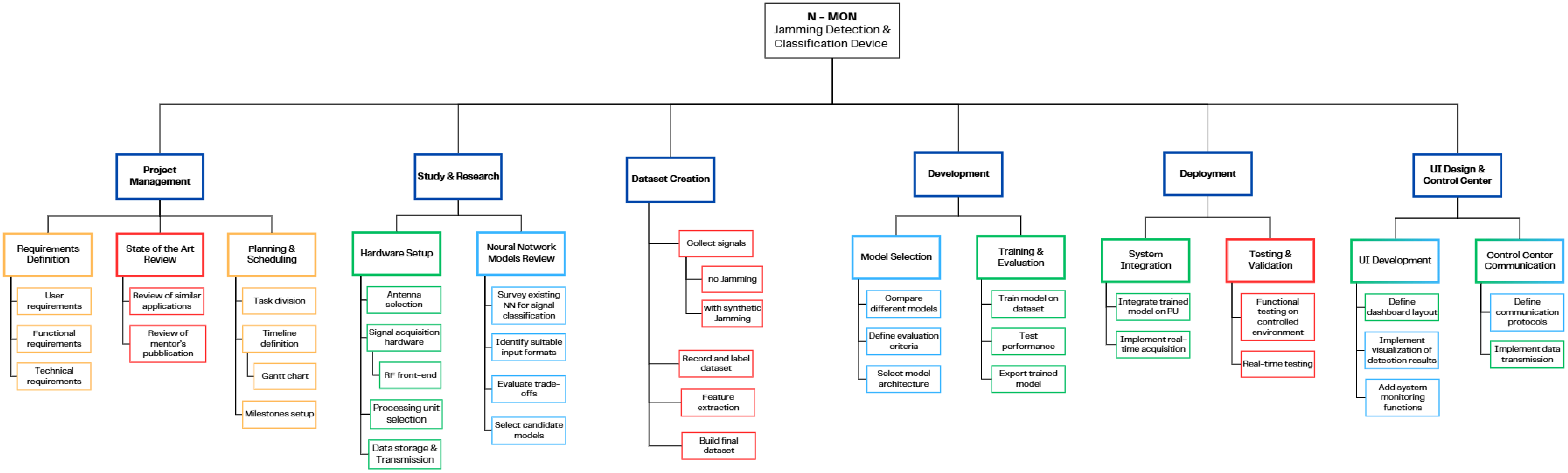


USER INTERFACE



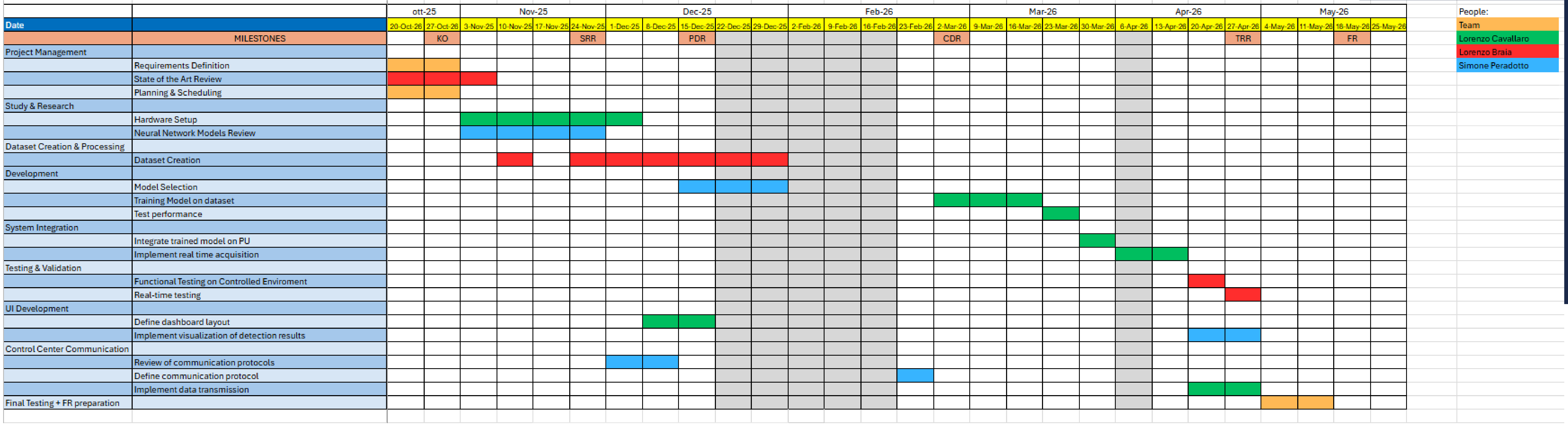


WBS



https://www.canva.com/design/DAG14EVzFeY/AZdbRfanr0YUQmWhZWHVmg/edit

GANTT CHART



Link accessible with Polito credentials :

https://politoit-my.sharepoint.com/:x:/g/personal/s343420_studenti_polito_it/EVHoF8byxhZOuSA_gHzWz6ABPUtiU81AzwQXv3SZYcAgyA?e=vecgTL

THANK YOU!

For your attention