| Document Title: Verification and Validation Test Plan | | |
|---|---|---|
| Document #: VVP 01 | Revision #: 02 | Previous Doc/Rev #: N/A |

# Introduction

## Overview

This UAV Sensor Spoofer project aims to develop a system capable of simulating and detecting sensor spoofing attacks using camera, LiDAR, and radar/GPS signals. One control station generates spoofed data, while the UAV analyzes the input with the aid of a neural network to detect falsified information in real-time. This verification and validation plan lays out a framework to ensure that all subsystems meet their functionality, performance, and compliance requirements as defined in the Requirements Specification (RS-01).

## Objectives

**Spoofing Simulation**: generate distinct spoofed signal data via MATLAB/Python

**Detection Models**: Light weight neural network that can distinguish a spoofed signal from a real signal

**Integration**: deploy model on UAV hardware with logging & failsafe. Control station signal generation with sensor array(s)

**Validation**: simulation testing/controlled flight demo

**Deliverables**: datasets, documentation, IEEE aligned standards

# Subsystems and Logical Module Identification

| Module | Requirement ID(s) | Verification Method | Validation Method | Planned Test Type | Success Criteria | Notes / Dependencies |
|---|---|---|---|---|---|---|
| Spoofing Module | FUN-1, FUN-3, FUN-4, FUN-5, FUN-7, FUN-8, FUN-9, CON-2 | Flight test, vary parameters, isolation testing, exposure to different scenarios | Wireless confirmation, compare data, export logs, confirm spoofing parameters | MATLAB Simulink, Unreal Engine simulation, test videos | 10 minutes | Correct data |
| Neural Network Module | PER-1, PER-2, PER-3, PER-4, PER-5, I/O-1, CON-3 | Continuous collection, latency testing | Compare consistency of data after 10 minute run, measure latency in ms | Python, Jetson Nano, datasets | 50 ms, 85% Accuracy, 30 seconds | Edge device |
| Control System | COM-1, COM-2, ENV-1, I-O3, TEST-1, CON-1 | Documentation comparison, Relocation, Hardware, Chamber for flight | Confirm signal isolation, FAA paperwork, flight logs | Drone Hardware, Simulation, Edge device | Test environment while spoofing | Code of Virginia 4VAC5-30-400, U.S.C 4480 |
| Ground Control Station | I/O-2, TEST-2 | Identify reliable product, Measure distance and test | Confirm at Propper distance | Laptop | miles | Communications Act of 1934, 18 U.S.C. 1367(a) |
| Power Module | POW-1, ME-1 | Measure in stable state | Compare endurance (power draw under max load) | Power monitor | Wattage | Dedicated hardware |
| Failsafe Module | REL-1, REL-2 | Observe flight | Induce failure, confirm stability | Battery log data, Simulation fault | Lifespan, I/O | UAV/simulation |

VV_Plan_Template.xlsx

## Detailed Test Procedure

### Spoofing Module

The spoofing module is a multi-component system that serves as a system to create spoofed and real signals. This module will be located on the ground station side which is

the other half of the project entirety. Below is a current up to date procedure that will be followed for testing the subsystem. It is important to note that due to this project being a work in progress, test procedures are subject to change.

**Test 1: BASELINE – Send verified real signal**

Objective: Confirm module can transmit a real (reference) signal properly and a receiver can recognize it as authentic/expected.

Constraints

- Use appropriate chambers for signal generation
- If no chamber available simulate in appropriate virtual environments (Simulink, Unreal Engine5, etc)

Setup

- Configure station to only generate real signals for radar/GPS, lidar, etc.
- Start logging scripts for data collection and post processing
- If environment virtual, ensure that all dependencies installed and updated

Safety Considerations

- Ensure that signal is in an anechoic chamber or some kind of faraday cage
- Ensure that signal will not be damaging if leaks occur
- Limit signal power

Procedure

1. Reset module and all receiver clocks, ensures test starts from a baseline condition
2. Configure and load **real** signal profiles on control station. Note, profile names and versions not yet defined.
3. Start continuous transmission for 10 minutes.
4. Start receiver to confirm and listen for all signals in the environment.
5. Stop transmission and save transmit/receive data logs to respective devices.
6. Run post processing scripts for data output and visualization. Will return verification and identification of sent signals

Expected Outcomes & Evaluation

- **Pass:** Receiver can view and record signals; proof of transmission. Post processing of signal evaluation & equality yields >90% score on **real** signal identification
- **Fail:** No lock for reception of signals, mismatch on signal parameters, accuracy score of <80%, or failure of any signal generation

**Test 2: VARIANT – Send spoofed signal**

Objective: Confirm module can transmit a spoofed signal properly, and a receiver can recognize it as spoofed/expected.

Constraints

- Use appropriate chambers for signal generation
- If no chamber available simulate in appropriate virtual environments (Simulink, Unreal Engine5, etc)

Setup

- Configure station to only generate spoofed signals for radar/GPS, lidar, etc.
- Start logging scripts for data collection and post processing
- If environment virtual, ensure that all dependencies installed and updated

Safety Considerations

- Ensure that signal is in an anechoic chamber or some kind of faraday cage
- Ensure that signal will not be damaging if leaks occur
- Limit signal power

Procedure

1. Reset module and all receiver clocks, ensures test starts from a baseline condition
2. Configure and load **spoofed** signal profiles on control station. Note, profile names and versions not yet defined.
3. Start continuous transmission for 10 minutes.
4. Start receiver to confirm and listen for all signals in the environment.
5. Stop transmission and save transmit/receive data logs to respective devices.
6. Run post processing scripts for data output and visualization. Will return verification and identification of sent signals

Expected Outcomes & Evaluation

- **Pass:** Receiver can view and record signals; proof of transmission. Post processing of fake signal evaluation & equality yields >85% score on **real** signal identification
- **Fail:** No lock for reception of signals, lack of spoof identification, accuracy score of <70%, or failure of any signal generation

**Test 3: VARIANT – Send spoofed and real signals**

Objective: Confirm module can transmit a spoofed AND real signal properly for a receiver can recognize it as authentic/expected.

Constraints

- Use appropriate chambers for signal generation
- If no chamber available simulate in appropriate virtual environments (Simulink, Unreal Engine5, etc)

Setup

- Configure station to only generate real & spoofed signals for radar/GPS, lidar, etc.
- Start logging scripts for data collection and post processing
- If environment virtual, ensure that all dependencies installed and updated

Safety Considerations

- Ensure that signal is in an anechoic chamber or some kind of faraday cage
- Ensure that signal will not be damaging if leaks occur
- Limit signal power

Procedure

1. Reset module and all receiver clocks, ensures test starts from a baseline condition
2. Configure and load **spoofed and real** signal profiles on control station. Note, profile names and versions not yet defined.
3. Start continuous transmission for 10 minutes.
4. Start receiver to confirm and listen for all signals in the environment.
5. Stop transmission and save transmit/receive data logs to respective devices.
6. Run post processing scripts for data output and visualization. Will return verification and identification of sent signals

Expected Outcomes & Evaluation

- **Pass:** Receiver can view and record signals; proof of transmission. Post processing of fake signal evaluation & equality yields >85% score on **real** signal identification. Post processing of signal evaluation & equality yields >90% score on **real** signal identification
- **Fail:** No lock for reception of signals, lack of spoof identification, accuracy score of <70%, or failure of any signal generation

## Neural Network Module

The neural network module is not a physical component of the system; rather, it is a software framework that has access to hardware inputs for "viewing" signals. This module will locate on the edge device (computer) located on the UAV. It will have access to the entire sensor array (GPS, LiDAR, Radar, etc.) and the drone navigation system for evaluation of all relevant signals.

**Test 1: BASELINE – Recognition & classification of real signals**

Constraints

- Carefully define and set standard for "**real**" signals. Includes signal parameters, formats, etc.
- Use appropriate chambers for signal generation
- If no chamber available, feed raw data to Neural Network or inject data stream to sensor array (Simulink, Unreal Engine).

Setup

- Choose appropriately tuned neural network for type of simulated environment
- Start logging scripts for signal generation/injected data stream, received signal data for post processing/identification
- Isolate sensor array or use direct injection of data stream for proper test conditions

Safety Considerations

- Ensure failsafe mechanism for misinterpreted signals
- Ensure testing system is either virtual or in a closed environment

Procedure

1. Clean boot edge device with chosen neural network, boot and initialize sensor array
2. If using data injection method, ensure that the platform for data injection is reset and contains simulated **real** signal data. If in chamber, ensure that all potential sources of interference are shut down or put away.
3. Start continuous transmission for 10 minutes.
4. Start receiver & neural network/sensory array to listen and identify all signals.
5. Stop transmission and save organized/signal identification data to edge device
6. Run post processing scripts for data output and visualization. Will return verification and identification of sent signals

Expected Outcomes & Evaluation

- **Pass:** Neural network is able to identify signal being real and classifies specific parameters and identifies signal type. Post processing scripts should yield accuracy of >92% score.
- **Fail:** Neural network is an overfitted model, accuracy of model only remains true for trained dataset and not experimental; therefore, network fails test.

**Test 2: VARIANT – Recognition & classification of spoofed signals**

Constraints

- Carefully define and set standard for "**spoofed**" signals. Includes signal parameters, formats, etc.
- Use appropriate chambers for signal generation
- If no chamber available, feed raw data to Neural Network or inject data stream to sensor array (Simulink, Unreal Engine).

Setup

- Choose appropriately tuned neural network for type of simulated environment
- Start logging scripts for signal generation/injected data stream, received signal data for post processing/identification
- Isolate sensor array or use direct injection of data stream for proper test conditions

Safety Considerations

- Ensure failsafe mechanism for misinterpreted signals
- Ensure testing system is either virtual or in a closed environment

Procedure

1. Clean boot edge device with chosen neural network, boot and initialize sensor array
2. If using a data injection method, ensure that the platform for data injection is reset and contains simulated **spoofed** signal data. If in a chamber, ensure that all potential sources of interference are shut down or put away.
3. Start continuous transmission for 10 minutes.
4. Start receiver & neural network/sensory array to listen and identify all signals.
5. Stop transmission and save organized/signal identification data to edge device
6. Run post processing scripts for data output and visualization. Will return verification and identification of sent signals

Expected Outcomes & Evaluation

- **Pass:** Neural networks are able to identify signals being spoofed and classify specific parameters and original signal parameters. Post processing scripts should yield accuracy of >92% score.
- **Fail:** Neural network is an overfitted model, accuracy of model only remains true for trained dataset and not experimental; therefore, network fails test. Network falsely identifies 25% of signals as real, highlights identification issues in neural network

**Test 3: VARIANT – Recognition & classification of spoofed & real signals**

Constraints

- Carefully define and set standard for "**spoofed**" signals. Includes signal parameters, formats, etc.
- Carefully define and set standard for "**real**" signals. Includes signal parameters, formats, etc.
- Use appropriate chambers for signal generation
- If no chamber available, feed raw data to Neural Network or inject data stream to sensor array (Simulink, Unreal Engine).

Setup

- Choose appropriately tuned neural network for type of simulated environment
- Start logging scripts for signal generation/injected data stream, received signal data for post processing/identification
- Isolate sensor array or use direct injection of data stream for proper test conditions

Safety Considerations

- Ensure failsafe mechanism for misinterpreted signals
- Ensure testing system is either virtual or in a closed environment

Procedure

1. Clean boot edge device with chosen neural network, boot and initialize sensor array
2. If using data injection method, ensure that the platform for data injection is reset and contains simulated **real** and **spoofed** signal data. If in chamber, ensure that all potential sources of interference are shut down or put away.
3. Start continuous transmission for 10 minutes.
4. Start receiver & neural network/sensory array to listen and identify all signals.
5. Stop transmission and save organized/signal identification data to edge device

6. Run post processing scripts for data output and visualization. Will return verification and identification of sent signals

Expected Outcomes & Evaluation

- **Pass:** Neural network is able to distinguish signal being spoofed or real and classifies specific parameters and original signal parameters. Post processing scripts should yield accuracy of >90% score.
- **Fail:** Neural network is an overfitted model, accuracy of model only remains true for trained dataset and not experimental; therefore, network fails test. Network fails for identification and classification of signals with score of 50% (50% of signals falsely classified), highlights identification issues in neural network.

## Control System Module

The Control System Module is responsible for maintaining UAV stability, compensating for discrepancies between real and spoofed sensor data, triggering failsafe mechanisms during anomalies, and logging all sensor and control data for post-flight analysis. These tests verify compliance with core control and navigation requirements defined in the system specification.

**Test 1: BASELINE – Flight Stability**

Objective: Verify that the UAV maintains stable flight and proper navigation under good conditions with real sensor inputs.

Constraints

- UAV must maintain flight at a minimum altitude of 40 ft and a minimum speed of 15 mph during testing
- All primary sensors (GPS, LiDAR, camera, radar) must be active and calibrated.
- Communication links with the ground station must remain stable throughout the test.

Setup

- Configure UAV control software with verified sensor profiles and no spoofed data
- Start logging scripts for data collection and post processing

Safety Considerations

- Ensure that signal is in an anechoic chamber or some kind of faraday cage
- Ensure that signal will not be damaging if leaks occur
- Limit signal power

Procedure

1. Initialize the control system and calibrate all onboard sensors.
2. Launch the UAV and begin autonomous waypoint navigation.
3. Monitor flight path tracking, altitude, and speed throughout the mission.
4. Land the UAV and retrieve all control and sensor logs for analysis.

Expected Outcomes & Evaluation

- **Pass:** UAV completes mission at stable altitude and speed within operational thresholds.
- **Fail:** UAV shows significant deviation, instability, or control system malfunction during flight.

Test 2: Recalibration Testing

**Constraints**

- Define baseline sensor parameters for normal operation, including expected GPS drift tolerance, LiDAR distance accuracy, and radar detection ranges.
- Ensure that spoofed signals are introduced gradually, simulating a realistic slow drift or evolving anomaly rather than an abrupt change.
- Maintain all standard environmental conditions (wind, altitude, temperature) within acceptable operational ranges to isolate the effect of spoofed data.
- If physical flight testing is not possible, use a virtual simulation environment (Simulink, Gazebo, or Unreal Engine) to reproduce controlled spoofing behavior.

**Setup**

- Load the control system firmware with all adaptive flight and compensation algorithms enabled.
- Configure the ground control station to inject gradually changing spoofed data, starting with a 0.5% deviation and increasing to 5–10% over the course of the flight.
- Enable full logging of control commands, sensor inputs, compensation outputs, and trajectory data for post-flight analysis.

**Safety Considerations**

- Conduct the test in a closed or simulated environment to prevent unsafe flight conditions.
- Implement a manual override or emergency stop mechanism in case adaptive compensation fails or the UAV becomes unstable.

**Procedure**

1) Boot and initialize the UAV control system under normal sensor conditions.
2) Begin a standard navigation mission and verify stable flight with baseline data.
3) Introduce gradually spoofed data into the GPS and LiDAR inputs while the UAV is in flight, increasing spoof intensity in defined increments every 2 minutes.
4) Monitor and record how the control system compensates for the increasing deviation and whether it adjusts the flight path in real time.

5) At the end, stop transmission, land the UAV safely, and save all recorded data for post-processing.

**Expected Outcomes & Evaluation**

Pass: The control system continuously recalibrates and maintains a stable flight path as spoofing intensity increases. Positional differences remain below 10% of the planned trajectory and system can be seen compensating

Fail: The control system fails to adjust to gradual spoofing, deviates more than 20% from the planned flight path, or becomes unstable due to compensation errors.

## Ground Control Station Module

The Ground Control Station module is responsible for transmitting real and spoofed sensor data (LIDAR/Camera/Radar/GPS). It should be able to collect the incoming data streams from the sensor, process them in real time, and provide some interface to observe the system performance and the spoofing parameters. The following tests are designed to verify the functionality, performance, and how well the module complies with the project requirements.

**Test 1: BASELINE – Communication Link/ Data Integrity**

Objective: Confirm that the GCS can create a stable communication link with the UAV over the specified 3-mile range and successfully transmit sensor data packets

Constraints

- Communication must remain stable over the full operational range (≥ 3 miles)
- If physical field testing is not feasible, a virtual simulation environment (MATLAB/Simulink, Unreal Engine) can be used to replicate conditions.

Setup

- Deploy GCS configured for real sensor data transmission only.
- UAV receiver initialized with standard data logging and integrity-check scripts.
- Network monitoring tools for post processing

Safety Considerations

- Ensure that signal is in an anechoic chamber or some kind of faraday cage
- Ensure that signal will not be damaging if leaks occur

- Limit signal power

Procedure

1. Initialize the GCS and UAV receiver and synchronize system clocks.
2. Establish a communication link at 0.5 miles, then extend range to 3 miles.
3. Transmit continuous real sensor data (e.g., LiDAR, GPS, camera telemetry) for 10 minutes.
4. Record packet latency and data integrity at each distance increment.
5. analyze receiver logs to compare transmitted vs. received data.

Expected Outcomes & Evaluation

- **Pass:** Communication remains stable across the full range with >95% packet delivery and >98% data integrity
- **Fail: Significant** packet loss (>5%), connection dropouts, corrupted data, or receiver unable to decode transmitted signals.

**Test 2: Selective Spoofing**

Constraints

- Carefully define standards for each sensor type's "real" data profile (GPS, LiDAR, camera, radar). Include expected data format, frequency, and parameters.
- Carefully define and implement spoofed variants for each sensor stream with controlled parameter deviations (e.g., GPS offset, altered LiDAR point cloud).
- If physical sensors are unavailable, simulate sensor data streams using MATLAB, Simulink, or Unreal Engine.
- Ensure the test environment is isolated to prevent interference with actual navigation or communication systems.

**Setup**

- Configure GCS with multiple sensor profiles (GPS + LiDAR minimum).
- Enable selective spoofing functionality (ability to spoof one sensor stream while transmitting real data on others).
- Activate logging scripts on both GCS and UAV to capture transmission type, parameters, and timestamps.

**Safety Considerations**

- Ensure testing occurs in a closed virtual or controlled environment to prevent unintended GPS or sensor disruptions.
- Implement emergency stop or termination command in case spoofed data causes unsafe UAV behavior.

**Procedure**

- Clean boot the GCS system and load verified sensor profiles.
- Begin continuous transmission of real data for all sensor types for 5 minutes.
- Enable spoofing only on the GPS stream and continue transmitting for 5 minutes while logging data.
- Disable GPS spoofing and enable spoofing only on the LiDAR stream for 5 minutes.
- Save all transmission logs and run post-processing scripts to identify spoofed vs. real streams and their parameter differences.

**Expected Outcomes & Evaluation**

Pass: UAV or receiver correctly identifies which sensor streams are spoofed vs real and logs data with accurate timestamps.

Fail:Datastreams show unexpected modifications/spoofed streams fail to transmit/ classification accuracy falls below 85%.

## Schedule

| Week | Test | Subsystem | Resources |
|---|---|---|---|
| Week 1 | Training/Evaluation | Neural Network | Jetson, Python, Availible data sets |
| Week 2-3 | Simulation | Spoofing | MATLAB, dataset, Simulink |
| Week 4 | Integration testing | Neural Network + Spoofing | MATLAB, Jetson |
| Week 5 | Logging testing | UAV simulation | UAV hardware, drone environment |
| Week 6 | Range/latency testing | Ground Sation | Laptop |
| Week 7 | Endurance/Failsafe testing | Full system | UAV, control system |
| Week 8 | Data comparison and reporting | Data analysis | MATLAB |

# Conclusion

No validation/verification has been done at this point, but this document will ensure components meet early requirements. Any issues uncovered will feed directly into a revision of the design and updated RS-01 document. Continuous integration between MATLAB, Python, and UAV firmware will streamline testing in later phases. Additional hardware components will be accounted for after identification and budgeting.