

SHIELD Design Document

Sep 28, 2026 · Bryson Noble (revised Oct 23, 2026)

System for High Integrity Elimination of Low-Altitude Drones (SHIELD) — Software Design Document, Milestone 1

1. Project Title, Team Members

System for High Integrity Elimination of Low-Altitude Drones (SHIELD)

- Bryson Noble (SWE) - bnoble2023@my.fit.edu
- Sophia Versage (AEE) - sversage2024@my.fit.edu
- Edward Ledwith (AEE) - eledwith2023@my.fit.edu
- Nick Jacobson (AEE) - njacobson2023@my.fit.edu
- Aryaa Vijay (AEE) - avijay2024@my.fit.edu
- Miguel Mohammed-Mueses (AEE) - mmohammedmue2023@my.fit.edu
- Isabelle Chamberlain (AEE) - ichamberlain2023@my.fit.edu
- Joshua Huser (AEE) - jhuser2023@my.fit.edu
- Will Emerson (AEE) - wemerson2023@my.fit.edu
- Tharushi Pelendagama (AEE) - tpelendagama2023@my.fit.edu

AEE Faculty Advisor: Madhur Tiwari, PhD - mtiwari@fit.edu

Client: Firat Irmak, PhD - Aerospace Engineering Department, Florida Institute of Technology

2. Introduction

Purpose. This document describes how the SHIELD avionics software is built to meet the requirements in the SHIELD Requirements Document (AVS.01 through AVS.05.01): the hardware it runs on, the structure of the onboard pipeline, the design of each component, the interfaces between them, and the operator interface.

Scope. The document covers only the Avionics & Software (AVS) subsystem, which is my individual scope for the SWE senior design course: onboard target detection, tracking, ranging, the Ground Control System (GCS) software, and the test tooling. Guidance, navigation and control (GNC), structures and aerodynamics are designed by the AEE team and appear here only where AVS connects to them. Initial detection of an incoming drone is assumed to come from a separate ground system; SHIELD's mission starts once the interceptor has lifted off facing the target.

Revision note. This revision updates the Milestone 1 design with decisions made since then: the move from a single camera to a stereo camera for ranging, the selected carrier board, flight controller and altitude sensor, and the duplicate-box suppression stage added to the pipeline.

Design goals.

1. Real time: every frame processed within 33.33 ms (AVS.04), so the pipeline sustains at least 30 fps.
2. One code path: the same Python pipeline runs on a PC in emulation and on the Jetson Orin Nano, so everything tested in emulation carries over to hardware.

3. Fail safe: a lost track, a degraded confidence or a dead video link is always visible to the operator, and the operator can abort at any time.
4. Testable: every requirement can be verified by an automated STE test script (see the SHIELD Test Plan).

3. System Architecture

SHIELD is an interceptor drone plus a ground control station. On the drone, the Jetson Orin Nano runs the AVS pipeline: it reads frames from the stereo camera, finds and tracks targets, measures their position relative to the drone, and passes that target state to the flight controller, where GNC's guidance law turns it into flight commands. The GCS shows the operator what the drone sees and sends Engage and Abort commands back. A safety pilot with an RC transmitter can take manual control at any time.

SHIELD system architecture

Avionics & Software (this document) is highlighted; GNC and structures belong to the AEE team.

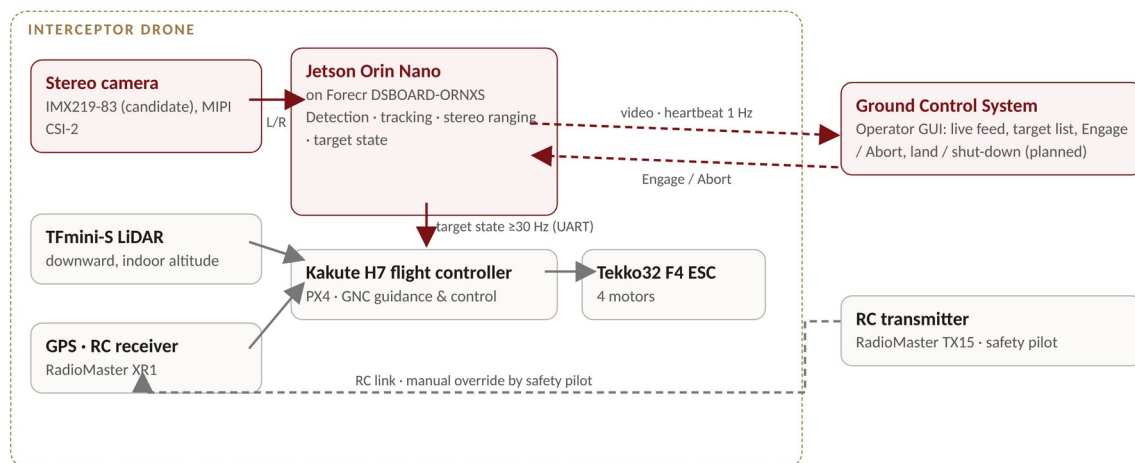


Figure 1. SHIELD system architecture

4. Hardware Design

Each component was chosen with a weighted decision matrix or Pugh matrix, presented at the weekly team meetings. Weight, power draw and compatibility drove most choices, since the airframe must stay small and maneuverable.

Component	Selection	Why
Edge computer	NVIDIA Jetson Orin Nano (Super)	Scored 3.55/4.00 against the Jetson Nano and Raspberry Pi 5 + AI HAT; GPU compute for YOLO at 30+ fps. Checked out from the HSDC.
Carrier board	Forecr DSBOARD-ORNXS	Scored 3.60/4.00; 50 g lighter and a third smaller than the dev-kit carrier, with the CSI ports the camera needs.
Camera	Stereo camera (IMX219-83 leading candidate)	Stereo ranging scored 3.20/4.00 for depth detection; dedicated depth cameras scored higher (3.50) but are over budget. Fixed mount: a gimbal's stability did not outweigh its mass and complexity (2.95 vs 2.65).

Flight controller	Holybro Kakute H7 V1.5	Scored 3.60/4.00: full PX4 support in a much smaller and lighter board than Pixhawk options.
Altitude sensor	Benewake TFmini-S micro LiDAR	Pugh matrix favored a downward LiDAR (the only option accurate indoors, where flights happen); TFmini-S scored 3.75/4.00.
Storage	SanDisk Extreme 128 GB microSD	Boot and log storage for the Orin.

Wiring. The full wiring is captured in three Scheme-it schematics: the flight controller connections (Orin carrier on UART3, TFmini-S on I²C, RC receiver on UART4, GPS on UART6, ESC on the motor header), the carrier board to stereo camera connections through 22-to-15-pin MIPI CSI-2 adapters, and the 260-pin Orin Nano to DSBOARD-ORNXS mapping. The Orin and DSBOARD CSI pinouts were checked against each other so the camera connects without custom wiring.

5. Software Architecture

The software is a single Python process built on OpenCV and Ultralytics, organized as a pipeline that handles one frame at a time. Frames are pulled from the source only when the pipeline is ready for the next one, so a slow frame never builds up a queue of stale images.

Onboard software pipeline

One frame at a time; every stage runs on the Orin Nano (or the PC in emulation).

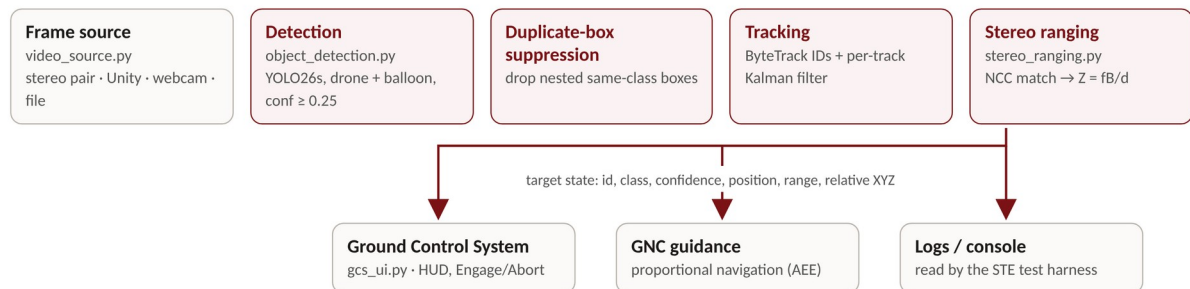


Figure 2. Onboard software pipeline

Module	Responsibility
__main__.py	Entry point. Parses the platform mode (0 Emulation, 1 Hardware, 2 Prototype) and frame source, builds the pipeline, and runs the main loop. Supports a clean shutdown on a QUIT line from the STE harness.
video_source.py	Frame sources behind one interface: Unity virtual camera (mono or stereo, over TCP), webcam, or video file. The stereo camera joins in Hardware mode.
object_detection.py	YOLO detection, duplicate-box suppression, ByteTrack tracking and the per-track Kalman filter. Produces one detection record per tracked target per frame.
stereo_ranging.py	Matches each tracked box into the right image and converts disparity to range and relative position.
gcs_ui.py	Operator interface: live annotated feed, target list, Engage/Abort, warning banners.
config.py	Every threshold, timeout and path in one place, shared by emulation and hardware.

Libraries. The Ultralytics YOLO and ByteTrack implementations, OpenCV for imaging and the Kalman filter, NumPy, and Tkinter for the GCS window. Models are trained with Ultralytics on a local GPU for experiments and on the AI.Panther HPC cluster for the final model.

6. Component Design

6.1 Frame acquisition

Every source returns the same frame object, so the rest of the pipeline does not know or care where frames come from. Detection runs on the left (reference) image; the right image is only used for ranging. In emulation the Unity virtual camera renders a rectified stereo pair with known baseline and focal length and sends both with every frame. On hardware the pair will need a stereo calibration (OpenCV stereoCalibrate and stereoRectify) and synchronized shutters before matching.

6.2 Detection

The detector is YOLO26s, chosen from a weighted matrix of YOLOv8, v11 and v26 in nano, small and medium sizes (3.35/4.00), after a first matrix showed the YOLO family ahead of EfficientDet and SSD models. It is fine-tuned on a two-class dataset (drone and balloon); balloons serve as the safe first test target the client asked for. Until the final YOLO26s model is trained on AI.Panther, a YOLO11n model trained on the same data stands in. Detections below a 0.25 confidence floor are discarded.

Duplicate-box suppression. YOLO sometimes draws looser, lower-confidence boxes that fully contain the real box on the same drone, mostly near the frame edge or at mid range. Those overlap too little for the model's own non-maximum suppression to merge, so before tracking, any same-class box that is at least 90% inside a higher-confidence box and centered within 30% of its size is dropped. This keeps one track per drone and does not depend on the model.

6.3 Tracking

ByteTrack assigns each target a persistent ID, using low-confidence detections to keep a track alive through brief occlusion. Each track also gets its own constant-velocity Kalman filter for smoothing and short-term prediction. A new filter is initialized from the track's first detection, with position known and velocity unknown, so a track reports its true position from its first frame. A filter is dropped after 30 frames without a matching detection, consistent with GNC.04's 30-frame line-of-sight limit.

6.4 Stereo ranging

For each tracked box, the matcher searches the same rows of the right image for the best normalized cross-correlation match, refines the peak to sub-pixel accuracy, and converts the disparity d into depth with $Z = f \cdot B / d$, where f is the focal length in pixels and B the baseline. X and Y follow from the box center: $X = (u - cx) \cdot Z / f$ and $Y = (cy - v) \cdot Z / f$. A match below 0.6 correlation or under 1 px of disparity is treated as no fix and the last good position is kept; positions are smoothed per track. Matching only the target's own box instead of computing a full depth map keeps the cost at about 0.27 ms per frame and keeps the background from deciding the answer. In emulation the median range error is 0.6%, rising to about 5–6% under 2 m and beyond 25 m.

6.5 Target state output

Each frame produces, for every tracked target: track ID, class, confidence, smoothed image position, range, and position relative to the camera (right, up, forward) in meters. Today this goes to the GCS and the console, where the STE harness reads it. For integration, the same record will be sent to the GNC software at 30 Hz or faster (GNC.01) over the Orin-to-flight-controller UART link; the message format will be agreed with the GNC team in Milestone 3, following the advisor's direction to develop detection and GNC separately and integrate at the end.

7. Interfaces

Interface	Between	Design
Virtual camera	Unity ↔ pipeline (emulation)	TCP, pull-based. The pipeline sends one request byte (0x01 mono, 0x02 stereo); Unity replies with a 4-byte length and the payload: a JPEG, or for stereo the baseline, focal length and both JPEGs. A 2 s read timeout declares the link dead.
Camera	Stereo camera → Orin	MIPI CSI-2 through the DSBOARD-ORNXS camera connectors.
Target state	Orin → flight controller (GNC)	UART3 on the Kakute H7. Target state at ≥30 Hz; format to be agreed with GNC (Milestone 3).
Commands	GCS → drone	Engage (with selected track ID) and Abort; land and shut-down modes planned (AVS.05).
Heartbeat	Drone → GCS	Status at 1 Hz (AVS.05.01). In emulation every frame request doubles as a link check.
Test control	STE harness ↔ Unity and pipeline	Scene commands over TCP (spawn hovering or moving drones, switch day/night, clear); pipeline output read from stdout; QUIT on stdin.

8. User Interface Design

The GCS is the operator's only view of the engagement, so it is designed to be read at a glance during long sessions and while things move. Color choices follow research on chromatic vibration and eye strain: cyan/teal for tracked targets, amber only for degraded-track warnings, and red reserved for the engaged target and the Abort control (with a white label). Targets are marked with a corner reticle rather than a solid box or crosshair, so the drone itself stays visible.

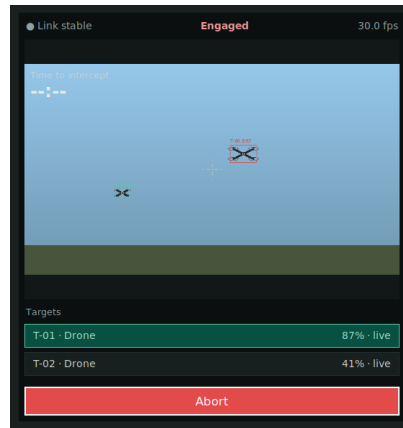


Figure 3. GCS window in emulation, with one target engaged

- **Status bar:** link state, mode (Standby or Engaged) and live frame rate.
- **Video:** annotated feed with reticles, track IDs and a time-to-intercept readout.
- **Target list:** each track's ID, class, confidence and range; clicking a row selects it.
- **Engage / Abort:** one button. Engage is disabled while the selected target is under 25% confidence (AVS.03); once engaged it becomes Abort.
- **Warnings:** a banner when the engaged track's confidence degrades, and a full-screen banner when the video link is dead, so stale video is never mistaken for live.
- **Planned:** two disengage modes requested by the client, Land Now and Shut Down, and the 1 Hz heartbeat indicator.

9. Data Design

- **Training data:** a combined two-class set of 9,738 images: two Kaggle drone datasets, a Kaggle balloon dataset, and 2,000 synthetic drone images from a Unity data-collector tool that randomizes placement, camera rotation and day/night lighting and labels images automatically. The Seraphim (83,483 images) and DUT Anti-UAV (10,000 images) sets will be merged for the final model, without letting the same image land in both training and validation.
- **Model weights:** the trained best.pt is committed to the repository so every machine runs the same model; an INT8 export script prepares it for the Orin.
- **Configuration:** config.py holds every tunable value with its rationale and the requirement it serves.
- **Logs:** each STE run writes a test log and a timestamped copy of the pipeline's output, so every verdict can be audited.

10. Platform Modes

Mode	Runs on	Frame source	Status
0 Emulation	Windows PC	Unity virtual camera (stereo), webcam or video file	Implemented
1 Hardware	Jetson Orin Nano on the bench	Stereo camera	Milestone 3
2 Prototype	Orin Nano mounted on the drone	Stereo camera, live GNC link	Second semester

11. Requirements Traceability

Requirement	Satisfied by
AVS.01 Detect a target within 20 m	Detection (6.2) with the custom model; stereo ranging (6.4) reports the range.
AVS.02 Track a target with a span of at least 0.25 m	Detection (6.2), duplicate-box suppression and tracking (6.3).
AVS.03 Keep at least 25% confidence en route	Confidence in every target-state record (6.5); GCS engage gate and degraded-track warning (8).
AVS.04 Process detection within 33.33 ms	Pull-based single-frame pipeline (5); GPU inference on the Orin; INT8 export.
AVS.05 Receive Engage and Abort from the GCS	GCS Engage/Abort control (8); command interface (7).
AVS.05.01 Send a 1 Hz heartbeat to the GCS	Heartbeat interface (7); video-link watchdog in emulation.

12. Design Risks and Open Items

- **Thermal throttling and tail latency:** worst-case frame time over a 6-minute flight, not the average, decides AVS.04. Mitigations: cooling, a GCS frame-rate monitor, and benchmarking on the Orin.
- **Stereo calibration:** real lenses need calibration and synchronized capture before ranging matches the emulation results.
- **Final camera:** the IMX219-83 is the leading candidate; other stereo cameras are still being compared.
- **Small or distant targets:** the stand-in model misses the smallest airframe at 20 m; the AI.Panther training run on the full dataset is meant to close this.
- **GNC message format:** to be agreed with the GNC team before integration in Milestone 3.
- **Export control:** guidance and interception code is reviewed before being made public.