System Design: Embedded Hit Detector

Simulated Projectile Hit Detection

1 Overview

This document describes the system architecture for a standalone embedded hit-detection unit mounted on a simulated firearm. The unit determines, at the instant of trigger pull, whether a simulated projectile would have struck a tracked airborne target, using a camera for target tracking and an inertial measurement unit (IMU) for barrel-motion compensation.

The design is specified in terms of functional subsystems and interface requirements rather than specific components. This allows the architecture to be realised across a range of hardware platforms, from a single-board computer during development to a custom PCB in production.

2 System Block Diagram

The embedded unit comprises six subsystems:

1. **Processing unit** — executes all software and coordinates the other subsystems.
2. **Camera module** — captures image frames and delivers them to the processor.
3. **Inertial measurement unit** — measures the barrel's angular velocity and linear acceleration.
4. **Trigger sensor** — detects the trigger-pull event and delivers a timestamped interrupt.
5. **Feedback actuators** — communicate the hit/miss result to the shooter.
6. **Power supply** — provides regulated power to all subsystems from a battery.

3 Hardware Subsystems

3.1 Processing Unit

Functional Requirements

- Execute a lightweight neural-network or classical image-processing pipeline at $\geq$60 fps with $\leq$10 ms per-frame latency.
- Read and fuse IMU data at $\geq$200 Hz.
- Perform the ballistic hit calculation (vector arithmetic, quadratic solve) within 1 ms of trigger pull.
- Drive all peripheral interfaces (camera, IMU, trigger, feedback) concurrently.

Architecture Characteristics

The processor should include hardware acceleration for at least one of: neural-network inference (NPU/TPU), GPU compute (for parallel image operations), or DSP (for fixed-point signal process-

ing). A heterogeneous SoC combining a general-purpose CPU with one or more of these accelerators is the natural choice. Typical categories include:

- **Application-class SoC** with integrated GPU/NPU and a camera serial interface (CSI). These run a lightweight Linux or RTOS and offer the fastest path to a working system.
- **Microcontroller with ML accelerator**, suitable when power and cost constraints are severe. These typically run bare-metal or a minimal RTOS and require more optimisation of the detection pipeline.
- **FPGA or FPGA-SoC**, offering maximum flexibility for custom camera and IMU data paths, at the cost of higher development effort.

Interface Requirements

- MIPI CSI-2 or parallel camera interface for the camera module.
- SPI or I^2C bus for the IMU.
- GPIO with interrupt capability for the trigger sensor.
- PWM or I^2C for feedback actuators.
- UART or USB for development/debug console.

3.2 Camera Module

Functional Requirements

- Frame rate: $\geq$60 fps.
- Resolution: $\geq$640×480 pixels. Higher resolution improves target localisation at range but increases processing load; 1280×720 is a reasonable upper bound.
- Exposure control: programmable exposure time, ideally $\leq$1/500 s, to minimise motion blur during fast swings.
- Interface: MIPI CSI-2 (preferred for bandwidth) or parallel digital output.
- Lens mount: C-mount or M12 (S-mount) to allow interchangeable lenses for field-of-view adjustment.

Lens Selection

The lens field of view (FOV) controls the tradeoff between angular coverage and target resolution. A narrow lens (15–25° horizontal FOV) is recommended: the shooter aims with their eyes, not through the camera, so wide coverage is unnecessary, and a narrow FOV maximises the number of pixels on the target, improving both detection reliability and distance estimation accuracy.

The focal length f (in pixels) is determined by the sensor pixel pitch p and the lens focal length F (in mm):

$$f = \frac{F}{p}$$

This value, together with the principal point (u_0, v_0) at the sensor centre, constitutes the camera's intrinsic calibration. It is measured once (e.g. via a checkerboard calibration procedure) and stored in firmware.

3.3 Inertial Measurement Unit

Functional Requirements

- 6-axis: 3-axis gyroscope + 3-axis accelerometer (9-axis with magnetometer is acceptable but the magnetometer is not used).
- Gyroscope range: $\geq \pm 500\,^\circ/\mathrm{s}$ to capture fast swings without saturation.
- Sample rate: $\geq 200\,\mathrm{Hz}$ (higher is better for rotation integration accuracy; $500\,\mathrm{Hz}$–$1\,\mathrm{kHz}$ is ideal).
- Interface: SPI (preferred for speed and deterministic timing) or $\mathrm{I}^2\mathrm{C}$.
- Interrupt output: data-ready interrupt line to the processor, enabling precise timestamping of each sample.

Mounting

The IMU must be rigidly mounted to the gun's barrel or receiver, as close to the camera as practical, to minimise the effect of any structural flex between the IMU and the camera.

Orientation Tracking

The gyroscope angular velocity $\boldsymbol{\omega}(t)$ is integrated to maintain the rotation matrix $M(t) \in \mathrm{SO}(3)$ mapping barrel-frame coordinates to the inertial frame. Integration is performed in quaternion representation to avoid gimbal lock:

$$q(t + \Delta t) = q(t) \cdot \Delta q, \qquad \Delta q = \left(\cos \frac{\|\boldsymbol{\omega}\| \Delta t}{2},\ \frac{\boldsymbol{\omega}}{\|\boldsymbol{\omega}\|} \sin \frac{\|\boldsymbol{\omega}\| \Delta t}{2} \right)$$

The accelerometer provides a gravity reference for correcting gyroscope drift during quiescent periods (e.g. before the shooter begins the swing). During the active swing, the accelerometer reading is dominated by the swing's centripetal and tangential accelerations, so the gravity reference is deweighted. A complementary filter or a dedicated orientation Kalman filter manages this transition.

3.4 Trigger Sensor

Functional Requirements

- Detect the trigger pull as a digital edge (rising or falling).
- Latency from mechanical actuation to processor interrupt: $\leq 1\,\mathrm{ms}$.
- Interface: single GPIO line with hardware interrupt.

Implementation

A microswitch or Hall-effect sensor mounted on the trigger mechanism produces a clean digital signal. If the mechanical trigger produces contact bounce, a simple RC debounce circuit ($\tau \approx 1\,\mathrm{ms}$) or a firmware debounce filter eliminates false edges.

The trigger interrupt handler timestamps the event against the system clock shared with the camera and IMU, ensuring precise temporal alignment between the trigger pull and the most recent sensor data.

3.5 Feedback Actuators

The hit/miss result must be communicated to the shooter within approximately 100 ms of the trigger pull to feel instantaneous. Any combination of the following may be used:

- **Haptic motor:** a linear resonant actuator (LRA) or eccentric rotating mass (ERM) motor driven via PWM or a dedicated haptic driver IC. Distinct vibration patterns encode hit vs. miss.
- **Audio transducer:** a small speaker or piezoelectric buzzer producing distinct tones for hit and miss.
- **LED indicator:** one or more LEDs visible in the shooter's peripheral vision (e.g. mounted near the receiver). Green for hit, red for miss.

3.6 Power Supply

Requirements

- Voltage regulation: provide stable rails for the processor (typically 1.8 V core, 3.3 V I/O), camera (typically 1.8 V and 2.8 V), and IMU (typically 3.3 V).
- Source: a single rechargeable lithium-polymer (LiPo) cell (3.7 V nominal) is the simplest option. A switching regulator with multiple output rails converts to the required voltages.
- Runtime: target $\geq$2 hours of continuous operation. Actual consumption depends on the processor choice but will typically be 2–5 W for an application-class SoC, under 1 W for a microcontroller-based design.

4 Software Architecture

The software is structured as a pipeline of five stages, matching the mathematical model described in the companion document. The pipeline runs on the processing unit and may be implemented bare-metal, on an RTOS, or on a lightweight Linux distribution, depending on the processor.

4.1 Stage 1: Sensor Acquisition

Two independent data streams feed the pipeline:

Camera frames. A camera driver reads frames from the CSI or parallel interface into a frame buffer. Each frame is timestamped against the system clock at the moment of exposure start (or centre, if available). The driver delivers the frame to Stage 2 via a double-buffered or ring-buffered handoff to decouple capture from processing.

IMU samples. An IMU driver, triggered by the data-ready interrupt, reads the gyroscope and accelerometer values via SPI/I^{2}C. Each sample is timestamped against the same system clock. The driver immediately updates the orientation quaternion $q(t)$ and stores the current angular velocity $\boldsymbol{\omega}(t)$.

Temporal alignment between camera and IMU is achieved by interpolating the IMU-derived orientation to each camera frame's timestamp using the high-rate quaternion history.

4.2 Stage 2: Target Detection

Each camera frame is processed to locate the target and extract:

- centroid (u_i, v_i) in pixel coordinates,
- apparent diameter s_i in pixels,

- detection confidence $c_i \in [0, 1]$.

The detection method is selected based on the target and the processor's capabilities:

Colour segmentation. Convert the frame to HSV colour space, threshold on the target's hue/saturation range, find connected components, and fit a bounding circle. This is computationally cheap (sub-millisecond on most processors) and sufficient for high-contrast targets against sky.

Neural-network detector. A small single-shot detector (e.g. MobileNet-SSD, YOLOv8-nano, or a custom architecture) runs on the processor's hardware accelerator (NPU, GPU, or DSP). The model is trained on a dataset of the target and quantised to INT8 for efficient inference. Output is a bounding box from which (u_i, v_i, s_i) are computed.

If no target is detected in a frame, the Kalman filter (Stage 3) coasts on its prediction without an update.

4.3 Stage 3: State Estimation

The Kalman filter maintains the target's state $(\boldsymbol{r}, \boldsymbol{v}, \boldsymbol{a})$ in the inertial frame. Its per-frame operation is:

1. **Observation construction.** From (u_i, v_i, s_i) and the stored camera intrinsics (f, u_0, v_0), compute the distance $d_i = Sf/s_i$ and unit ray direction $\hat{\boldsymbol{e}}_i$. Form the barrel-frame position $\boldsymbol{r}_i^b = d_i\, \hat{\boldsymbol{e}}_i$.
2. **Inertial transform.** Rotate into the world frame using the IMU-derived rotation: $\boldsymbol{r}_i = M(t_i)\, \boldsymbol{r}_i^b$, where $M(t_i)$ is the rotation matrix corresponding to the interpolated quaternion $q(t_i)$.
3. **Predict.** Propagate the 9-dimensional state $(\boldsymbol{r}, \boldsymbol{v}, \boldsymbol{a})$ forward by Δt using the constant-acceleration process model.
4. **Update.** Incorporate the observation $\boldsymbol{r}_i$, weighted by the detection confidence c_i (via the measurement noise covariance), to refine the state estimate.

The filter's 9-state predict/update cycle involves 9×9 matrix operations. On a processor with hardware floating-point support, this executes in tens of microseconds. On a fixed-point microcontroller, the matrices can be implemented in Q16.16 or similar representation with no loss of practical accuracy.

4.4 Stage 4: Ballistics Engine

This stage executes once, triggered by the trigger-pull interrupt. It implements the hit-detection mathematics:

1. **Read state.** Snapshot the Kalman filter's current estimate $(\hat{\boldsymbol{r}}_0, \hat{\boldsymbol{v}}, \hat{\boldsymbol{a}})$.
2. **Barrel state.** Read the current barrel direction $\hat{\boldsymbol{b}}$ and angular velocity $\boldsymbol{\omega}$ from the IMU orientation tracker.
3. **Projectile initial conditions.** Compute $\boldsymbol{p}_0 = \boldsymbol{r}_m$ and $\boldsymbol{V} = v_p\, \hat{\boldsymbol{b}} + \boldsymbol{\omega} \times \boldsymbol{r}_m$.
4. **Relative state.** Compute $\boldsymbol{\delta}_0 = \hat{\boldsymbol{r}}_0 - \boldsymbol{p}_0$, $\boldsymbol{\delta}_v = \hat{\boldsymbol{v}} - \boldsymbol{V}$, $\boldsymbol{\delta}_a = \hat{\boldsymbol{a}} - \boldsymbol{g}$.
5. **Intercept time.** Compute $\alpha = \boldsymbol{\delta}_a \cdot \boldsymbol{V}$, $\beta = \boldsymbol{\delta}_v \cdot \boldsymbol{V}$, $\gamma = \boldsymbol{\delta}_0 \cdot \boldsymbol{V}$ and solve:

$$\tau^* = \frac{-\beta \pm \sqrt{\beta^2 - 2\alpha\gamma}}{\alpha}$$

   Take the smallest positive root.
6. **Miss distance.** Evaluate $\rho = \|\boldsymbol{\delta}_0 + \boldsymbol{\delta}_v\, \tau^* + \frac{1}{2}\boldsymbol{\delta}_a\, \tau^{*2}\|$.
7. **Hit test.** Compute $\ell = v_p\, \tau^*$. If ℓ is within the maximum effective range and $\rho \leq R(\ell)$, the result is HIT; otherwise MISS.

The entire computation is a handful of dot products, a cross product, a square root, and a comparison. It completes in microseconds on any processor with hardware floating-point support.

4.5 Stage 5: Feedback

On receiving the hit/miss result from Stage 4, the feedback controller drives the appropriate actuator pattern. The latency budget from trigger pull to feedback onset is approximately:

Step	**Time**
Trigger interrupt to Stage 4 entry	$< 0.1\,\text{ms}$
Ballistics computation	$< 0.1\,\text{ms}$
Actuator driver activation	$< 1\,\text{ms}$
Total	$< 2\,\text{ms}$

The shooter perceives the feedback as instantaneous.

5 Timing and Synchronisation

All subsystems share a single monotonic system clock. The critical timing relationships are:

- **Camera–IMU alignment.** The IMU samples at a rate $\geq 3\times$ the camera frame rate. The IMU driver maintains a short circular buffer of timestamped quaternions. When a camera frame arrives, the orientation is interpolated (via spherical linear interpolation, SLERP) to the frame's exposure timestamp.
- **Trigger–state alignment.** The trigger interrupt timestamps against the same clock. The Kalman filter's state at trigger time is obtained either by reading the most recent estimate (if the trigger falls between frames, the state is at most one frame period stale) or by running a predict step forward from the last update to the trigger timestamp.
- **Frame deadline.** The per-frame pipeline (detection + geometry + Kalman update) must complete within one frame period (16.7 ms at 60 fps). If a frame overruns, it is dropped and the Kalman filter coasts.

6 Calibration

The following parameters are determined before deployment and stored in non-volatile memory:

- **Camera intrinsics** (f, u_0, v_0): measured via a standard checkerboard calibration procedure. If the lens is changed, recalibration is required.
- **Camera–IMU extrinsics**: the rotation and translation between the camera's optical frame and the IMU's body frame. If both are rigidly mounted to the same bracket, this is measured once during assembly.
- **Muzzle offset** $\boldsymbol{r}_m$: the displacement from the IMU to the muzzle tip, expressed in the IMU's body frame. Measured once when the unit is installed on the gun.
- **IMU biases**: gyroscope and accelerometer zero-rate offsets, measured during a stationary power-on calibration routine (the shooter holds the gun still for 2–3 seconds at startup).

- **Target diameter** S: entered per target type via a configuration interface.
- **Projectile parameters**: muzzle speed v_p and spread function $R(\ell)$, entered per ammunition type.

7 Configuration Interface

The unit exposes a serial (UART or USB) command interface for setting calibration parameters, selecting target and ammunition profiles, and streaming diagnostic data during development. In a production unit, this may be supplemented or replaced by:

- A Bluetooth Low Energy (BLE) link to a companion smartphone app for configuration and session review.
- A small OLED display and rotary encoder or button cluster on the unit itself, for field-adjustable settings (target type, ammunition type).
- An SD card slot for logging session data (target tracks, hit/miss results, video clips) for later review.

8 Performance Budget

	Application SoC	**MCU + Accelerator**
Frame rate	60 fps	30–60 fps
Detection method	Neural net (NPU)	Colour segmentation
Detection latency	3–8 ms	<1 ms
IMU rate	500–1000 Hz	200–500 Hz
Kalman update	<0.1 ms (float)	<0.5 ms (fixed-point)
Power consumption	2–5 W	0.3–1 W

Both configurations comfortably meet the 16.7 ms frame budget. The application-class SoC offers more headroom for complex detection algorithms and higher resolution; the microcontroller option trades detection sophistication for lower power and cost.

9 Mechanical Integration

The embedded unit should be packaged as a single module mountable on a Picatinny rail or similar accessory rail system. Design considerations:

- **Camera window**: an optical-quality glass or polycarbonate window protects the lens. The window must be anti-reflection coated to minimise glare.
- **Enclosure**: weather-sealed (IP54 or better) to handle outdoor use in light rain and dust.
- **Thermal management**: the processor is the primary heat source. A thermally conductive enclosure or a small heat spreader provides passive cooling. Active cooling (fans) should be avoided for noise and reliability reasons.
- **Vibration tolerance**: all connectors and the camera/IMU mounts should be mechanically secured (thread-locked fasteners, strain-relieved cables) to withstand repeated handling and the snap of a simulated trigger mechanism.

- **Trigger cable**: a thin, flexible cable runs from the trigger sensor to the main unit. A quick-disconnect connector allows the unit to be removed from the gun without tools.

10 Limitations and Design Risks

- **Gyroscope drift.** Unlike the ARKit-based iPhone prototype, the embedded system has no visual-inertial SLAM to correct gyroscope drift. Drift over a multi-second swing is mitigated by the startup bias calibration and by the fact that only short-term orientation accuracy (over the $\sim$0.5 s tracking window) is needed. If drift proves problematic, a visual-inertial odometry algorithm can be added in software, using background features in the camera image.

- **Lens calibration sensitivity.** The distance estimate $d = Sf/s$ is proportional to the focal length f. An error in f produces a proportional error in range, which propagates to the intercept time and miss distance. Careful calibration and a fixed-focus lens (no autofocus) are important.

- **Target detection in clutter.** Colour segmentation fails if the background contains colours similar to the target. A neural-network detector is more robust but requires a processor with hardware acceleration. The modular software architecture allows swapping detection methods without changing downstream stages.

- **Camera–IMU time synchronisation.** If the camera and IMU do not share a hardware trigger or synchronisation signal, their timestamps may have a small relative offset. This introduces a rotation error proportional to $\boldsymbol{\omega} \cdot \Delta t_{\text{offset}}$. At typical swing rates ($\sim$1 rad/s) and offsets ($<$1 ms), the error is $<0.06^\circ$, which is negligible. If the system is scaled to higher angular rates, a hardware sync line between camera and IMU should be added.