SYSTEM_JERICHO

Subsystem: SYS_SURVEILLANCE_CAM

Version: 1.0.2

📁 Folder Structure

```
SYSTEM_JERICHO/
├── src/
│   └── sys_surveillance_cam/
│       ├── __init__.py
│       ├── main.py                    # Entry point for launching all
camera configs
│       ├── router.py                  # Optional API endpoints (future
dashboard/API hook)
│       ├── exceptions.py              # Subsystem-specific exception
classes
│       ├── constants.py               # Codec constants, path templates,
thresholds
│       ├── schema.py                  # (Optional) Pydantic/YAML schema
validators
│       ├── src/                       # Core logic components
│       │   ├── camera.py              # Camera capture thread and
controller
│       │   ├── config_loader.py       # Validates and loads YAML camera
configs
│       │   ├── event_logger.py        # Event logging as JSON for alerts
│       │   └── utils.py               # UUID, timestamps, safe paths
│       ├── movement_detection/        # Motion detection logic
│       │   └── motion_detector.py     # Frame differencing and movement
analysis
│       └── cam_control/               # Webcam control, drivers, audits
│           ├── video_writer.py        # Handles dynamic MP4 video writing
│           └── camera_driver.py       # (Optional) abstraction over OpenCV
or OS-level control
├── config/
│   └── sys_surveillance_cam/
│       ├── camera1.yaml
│       └── camera2.yaml
├── data/
│   └── sys_surveillance_cam/
│       ├── recordings/<camera_id>/<YYYY-MM-DD>/motion_<HH-MM-SS>.mp4
│       ├── events/<camera_id>/event_<YYYY-MM-DD>_<HH-MM-SS>.json
│       └── logs/<camera_id>/surveillance.log
├── scripts/
│   └── sys_surveillance_cam/
│       ├── disable_device_windows.py
│       ├── enable_device_windows.py
│       ├── disable_device_linux.py
│       ├── enable_device_linux.py
│       └── audit_camera_processes.py
├── tests/
│   └── sys_surveillance_cam/
│       ├── test_utils.py
│       ├── test_motion_detector.py
│       ├── test_event_logger.py
│       ├── test_config_loader.py
│       └── test_camera.py
└── utilities/
    ├── logging_setup.py
    ├── notification_client.py
    └── deployment_helpers.py
```

🔧 Modules

`main.py`

Entry point. Discovers camera configs and spawns threads.

`camera.py`

Captures frames, detects motion, writes video, logs events.

`motion_detector.py`

Frame differencing-based motion detection with Gaussian blur.

`video_writer.py`

Encapsulated OpenCV video writer with `.mp4` output.

`event_logger.py`

Logs events as structured JSON for downstream systems.

`config_loader.py`

Loads YAML configs. Validates schema. Raises on error.

`utils.py`

Helpers: ISO timestamps, UUID, safe path joins.

`exceptions.py`

Custom error types like `CameraLoadError`, `InvalidConfigError`, etc.

`constants.py`

Centralized settings: default codec, directory structures, FPS range.

`router.py`

Optional: can expose FastAPI endpoints (future monitoring dashboard).

`schema.py`

(Optional) Pydantic models for strict config and event validation.

📜 YAML Camera Config

Each camera has a dedicated YAML:

```
camera_id: "camera1"
source: 0
frame_width: 1280
frame_height: 720
fps: 20
codec: "mp4v"
```

🛠 Scripts

`disable_device_windows.py` / `enable_device_windows.py`

Uses `pnputil` to toggle webcam at Windows driver level.

`disable_device_linux.py` / `enable_device_linux.py`

Uses `modprobe` to remove/reinsert webcam kernel module.

`audit_camera_processes.py`

Audits `/dev/video*` or uses `handle.exe` on Windows. Outputs to `data/audit/`.

✅ Compliance Checklist

Feature	Status
JerichoFormat	✅
Rotating Loguru logs	✅
ISO 8601 timestamps	✅
YAML + Schema Validation	✅
Modular components	✅
Event JSON export	✅
System Control Scripts	✅

📤 Integration Ready

This subsystem can now:

- Send events to SYS_ENCRYPTED_NOTIFICATION
- Be watched by SYS_WATCHDOG
- Feed honeypot context to SYS_HONEYPOT

📥 To Regenerate

Use the command:

```
SubsysDocPDF++ SYS_SURVEILLANCE_CAM
```

To regenerate this PDF from any chat.

Generated by SYSTEM_JERICHO — Powered by CarWant++, OmniSync, SovereignPurge, and MemoryWeave protocols.