End-to-end pseudocode for the Heron Deterrent Solution

1. High-Level Application Flow

```
None
MAIN()
 ├── load_configuration()
 ├── initialize_hardware()
 ├── load_ai_model()
 ├── start_web_ui()
 ├── start_video_pipeline()
 └── monitor_system_health()
```

---

2. Configuration & Initialization

Configuration Loader

```
None
FUNCTION load_configuration():
    config.motion.sensitivity
    config.motion.cooldown_seconds
    config.ai.confidence_threshold
    config.ai.model_path
    config.alert.sms_enabled
    config.alert.sms_throttle_minutes
    config.audio.volume
    config.storage.retention_days
    config.system.active_hours
```

Hardware Initialization

```
None
FUNCTION initialize_hardware():
    camera = open_camera(device_id)
    speaker = open_audio_device()
    storage = initialize_filesystem()
    database = connect_sqlite()
    mqtt = connect_mqtt_if_enabled()
```

AI Model Initialization (Edge TPU)

```
None
FUNCTION load_ai_model():
    model = load_yolo_model(config.ai.model_path)
    bind_model_to_edge_tpu(model)
    RETURN model
```

---

3. Video Capture & Motion Detection

Video Pipeline Loop

```
None
FUNCTION start_video_pipeline():
    background_model = initialize_background_subtractor()
    last_trigger_time = 0

    WHILE system_running:
        frame = camera.read()

        IF outside_active_hours():
            CONTINUE
```

```
        motion_detected = detect_motion(frame, background_model)

        IF motion_detected AND
cooldown_elapsed(last_trigger_time):
            clip = capture_video_clip(seconds=5)
            last_trigger_time = current_time()

            process_detection_event(clip)
```

Motion Detection

```
None
FUNCTION detect_motion(frame, background_model):
    foreground_mask = background_model.apply(frame)
    motion_area = calculate_contours(foreground_mask)

    IF motion_area > sensitivity_threshold:
        RETURN TRUE
    ELSE:
        RETURN FALSE
```

---

4. Detection Event Processing

```
None
FUNCTION process_detection_event(video_clip):
    frames = extract_key_frames(video_clip)

    detections = []
    FOR frame IN frames:
        detections.append(run_object_detection(frame))
```

```
    final_detection = aggregate_detections(detections)

    store_media(video_clip, final_detection)
    handle_detection_result(final_detection, video_clip)
```

Object Detection

```
None
FUNCTION run_object_detection(frame):
    results = yolo_model.infer(frame)

    FOR each detection IN results:
        IF detection.confidence >=
config.ai.confidence_threshold:
            RETURN detection

    RETURN detection(label="unknown", confidence=0)
```

Detection Aggregation

```
None
FUNCTION aggregate_detections(detections):
    IF any detection.label == "heron":
        RETURN heron_detection_with_highest_confidence

    IF any detection.label != "unknown":
        RETURN most_common_non_unknown_detection

    RETURN unknown_detection
```

---

5. Deterrent & Alert Logic

Detection Handler

```
None
FUNCTION handle_detection_result(detection, video_clip):

    IF detection.label == "heron":
        trigger_deterrent()
        create_alert(type="CRITICAL", detection)

    ELSE IF detection.label == "unknown":
        create_alert(type="WARNING", detection)
        enqueue_for_manual_review(video_clip)

    ELSE:
        log_event(type="INFO", detection)
```

Deterrent Sound

```
None
FUNCTION trigger_deterrent():
    sound = select_random_wav()
    speaker.set_volume(config.audio.volume)
    speaker.play(sound)
```

Alerting

```
None
FUNCTION create_alert(type, detection):
    save_alert_to_database(type, detection)

    IF type == "CRITICAL":
        IF sms_throttle_allows():
            send_sms_alert(detection)

    publish_mqtt_event_if_enabled(type, detection)
```

---

6. Storage & Retention

Media Storage

```
None
FUNCTION store_media(video_clip, detection):
    path = build_storage_path(detection.label)
    save_video(video_clip, path)
    save_metadata_to_db(detection, path)
```

Retention Job

```
None
FUNCTION retention_cleanup_job():
    DAILY:
        delete_media_older_than(config.storage.retention_days)

delete_db_records_older_than(config.storage.retention_days)
```

---

7. Manual Labeling & Dataset Export

Unclassified Queue

```
None
FUNCTION enqueue_for_manual_review(video_clip):
    mark_clip_as_unclassified_in_db(video_clip)
```

UI Label Submission

```
None
FUNCTION on_user_label_submission(image, bounding_boxes, label):
    save_label_in_yolo_format(image, bounding_boxes, label)
    update_db_label(image, label)
```

YOLO Export Structure

```
None
DATASET/
 ├── images/
 │    └── image_001.jpg
 └── labels/
      └── image_001.txt
```

---

8. Web UI (Logical Endpoints)

```
None
GET /live-stream
GET /alerts
GET /detections
GET /unclassified
POST /label
POST /trigger-deterrent
POST /upload-test-video
GET /system-status
POST /config
```

---

9. System Health & Fault Handling

```
None
FUNCTION monitor_system_health():
    EVERY 30 seconds:
        IF camera_disconnected():
            restart_camera()

        IF disk_space_low():
            raise_warning_alert()

        IF model_not_loaded():
            restart_ai_service()
```

---

10. MQTT (Optional)

```
None
TOPICS:
 heron/detection
 heron/alert
 heron/unclassified
 heron/system/status
```

```
None
FUNCTION publish_mqtt_event(type, payload):
    mqtt.publish(topic_for(type), payload)
```

---

11. Phase Transition Notes (Coral → Raspberry Pi)

```
None
IF running_on_raspberry_pi:
    reduce_frame_rate()
    lower_resolution()
    use_tflite_cpu_or_usb_tpu()
```

---

12. PoC Success Criteria (Implicit in Code)

- Detection latency < 500ms
- Deterrent triggers immediately
- False positives logged but non-blocking
- Manual labeling produces valid YOLO datasets
- No cloud dependency

What I can do next (high value)

- Convert this pseudocode into **actual Python module skeletons**
- Create a **repo structure** (`/capture`, `/ai`, `/ui`, `/storage`)
- Write **Edge TPU–optimized inference code**
- Define **unit + field test scenarios**
- Produce a **sequence diagram** from this flow

If you want code next, tell me:
Python + OpenCV + YOLOv8?