**Assessment Cover Sheet**

|  |  |
| --- | --- |
| **Programme Name:** | Internet of Things |
| **Partner University Name:** | Newcastle College Group |
| **Semester 1 or 2:** | Semester 2 |
| **Module Code:** | CMP213 |
| **Name of Lecturer:** | Priyanthi Manjula Dassanayake |
| **Due Date:** | 17.10.2025 |
| **Assessment Type:** | Project |

|  |  |
| --- | --- |
| **Declaration of Authenticity (please check and sign):**  By submitting this assignment, I declare that this work is fully my own, and the work of others (including internet sources) is acknowledged by quotations and appropriate citing and referencing.  I declare that this work has not made use of the work of any other student(s) past or present at this or any other educational institution or source.  I declare that I have not commissioned another person to complete this work. This could include the use of professional essay-writing services, essay banks, or ghostwriting services.  I declare that I have not asked another person to complete this work for me. This could include the help of friends, classmates, other students, or family members.  I declare I have not used any AI software to help write this work outside of the assessment brief/guidelines (this includes translation tools, paraphrasing tools, or content creation websites such as ‘ChatGPT’), other than for proofreading.  I acknowledge that I have read all the texts listed in the bibliography or references list. I confirm all sources cited in the work appear in the bibliography or references list at the end. |  |
|  |
|  |
|  |
|  |
|  |
|  |
| **Signed: Nicolae Sterian** | |

# Configurable IoT Intruder Detection System

# Document Purpose

This report records what was actually built, how the system was tested, and what the observed outcomes were, based on the original design blueprint.

1. **Overview of the Build**

The final system was built closely in line with the original design plan. It uses a Raspberry Pi 5 running Raspberry Pi OS Bookworm and an HC-SR04 ultrasonic sensor powered safely at 3.3 V to detect movement. When an object is detected within the set distance threshold, the system sends a Discord alert, with an optional image captured by the Pi Camera Module 3. The main loop includes a cooldown mechanism to prevent alert spamming. All configuration settings — including threshold, cooldown time, and whether to include a photo — were made adjustable through environment variables. The system successfully met the intended goals of being low-cost, classroom-safe, and responsive, with logs and notifications used to confirm correct operation throughout.

# 2. Hardware Assembly & Wiring (as built)

## 2.1 Scenario & Constraints

The hardware setup was completed on a Raspberry Pi 5 using jumper wires and breadboard connections. The ultrasonic sensor (HC-SR04) was powered at **3.3 V**, which is safe for the Pi’s GPIO and sufficient for short-range detection under 1.5 m — as intended for classroom or small room environments. The sensor’s **TRIG** and **ECHO** pins were wired directly to GPIO pins using BCM numbering, and care was taken to avoid using any voltage dividers or additional resistors due to the safe operating voltage.

A Pi Camera Module 3 was connected via a 22-to-15 pin ribbon cable and inserted into the **CAM0** port on the Pi. Before powering on, the ribbon cable was checked for correct orientation (blue side facing the USB ports), and the connection was gently locked in place to ensure camera stability.

**Pin Wiring Table (as built)**

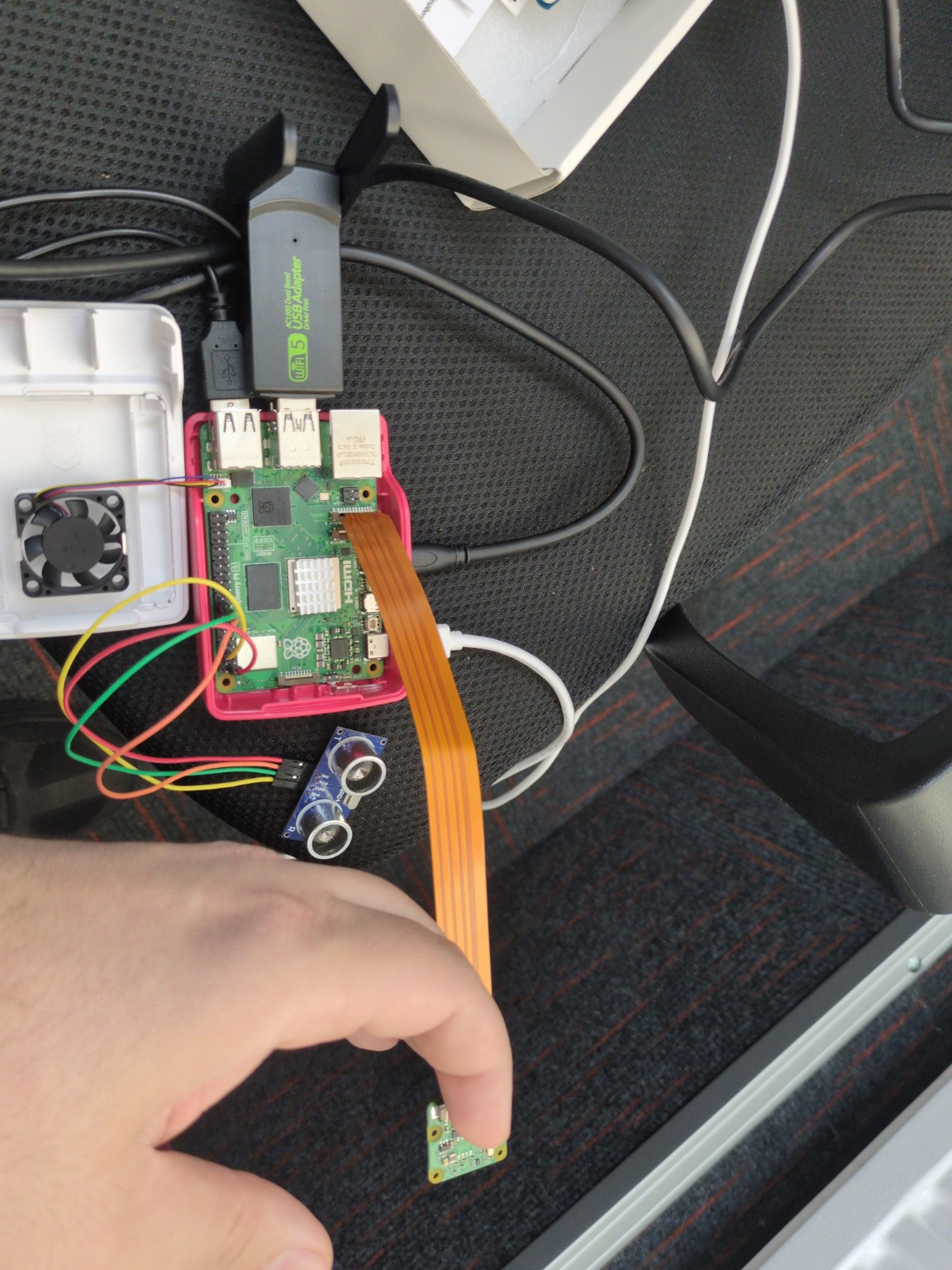
| **Component** | **Pin Name** | **Pi GPIO (BCM)** | **Pi Physical Pin** |
| --- | --- | --- | --- |
| HC-SR04 | VCC | 3.3 V | Pin 1 |
| HC-SR04 | GND | Ground | Pin 6 |
| HC-SR04 | TRIG | GPIO23 | Pin 16 |
| HC-SR04 | ECHO | GPIO24 | Pin 18 |
| Camera Module 3 | Ribbon | CAM0 port | — |

**Figure References**

* ***Figure I1*** – HC-SR04 wiring to Pi 5 GPIO header



* ***Figure I2*** – Camera Module 3 with 22↔15 ribbon cable connected to CAM0



These assembly steps ensured a safe and functional setup before software testing began.

# 3. Software Environment & Setup

The project was developed and tested on a Raspberry Pi 5 running **Raspberry Pi OS Bookworm (64-bit)**, which includes native support for the **IMX708** sensor used in the **Camera Module 3**. All packages were installed via apt or were already available in the system image. The environment was kept minimal to reduce setup complexity.

| **Component** | **Version / Command** | **Purpose** |
| --- | --- | --- |
| OS | Raspberry Pi OS Bookworm (64-bit) | Official 2023+ release with camera support |
| gpiozero | pre-installed | High-level GPIO abstraction |
| lgpio | sudo apt install python3-lgpio | Fast GPIO backend for Pi 5 |
| requests | sudo apt install python3-requests | HTTP client for Discord webhook |
| libcamera-apps | pre-installed | Native camera support tools (libcamera) |
| fswebcam | sudo apt install fswebcam | Backup image capture (USB or fallback) |

**Camera Configuration**

To ensure the camera module was detected properly, the following was done:

* Verified imx708 overlay was automatically used by Bookworm (no config.txt edits needed).
* Camera was connected to the **CAM0** port (default active) using a 22-to-15 pin ribbon cable.
* Tested camera functionality with:

libcamera-still -o /tmp/test.jpg

**User Group Permissions**

* The user account (esl) was already part of required groups:
  + video — for camera access
  + gpio — for controlling GPIO pins
* No changes to /boot/config.txt were required.

**Notes on** pigpio

Although the code includes a fallback to the pigpio backend, it was **not used** in this implementation because:

* The lgpio backend is **faster and more reliable** on Raspberry Pi 5.
* The pigpiod daemon was not running, and deliberately not enabled to simplify setup.
* This avoids additional service dependencies or socket connection issues (e.g., localhost:8888).

# 4. Program Structure (as built)

The program was structured as a modular and event-driven script, written in Python 3. Its purpose was to continuously monitor the distance detected by the ultrasonic sensor, check against a threshold, and trigger alerts based on a cooldown system. The code included fallback mechanisms for GPIO backends and camera utilities, aiming to keep the system robust even when optional components failed (e.g., missing camera).

## 4.1 Control Flow Overview

The program followed this high-level logic:

1. **GPIO Backend Selection**
   * On launch, it attempted to use the lgpio backend (optimal for Raspberry Pi 5).
   * If that failed, it attempted pigpio as fallback, and finally defaulted to **gpiozero’s native backend** if neither were available.
   * This selection was dynamic and ensured compatibility on multiple Pi models.
2. **Sensor Loop**
   * A DistanceSensor object continuously measured distance from the HC-SR04 sensor.
   * The threshold\_distance was set based on the value from the environment variable INTRUDER\_THRESHOLD\_CM, defaulting to **35 cm**.
3. **Cooldown Gate**
   * Once a trigger occurred (object detected within threshold), the program checked if the **cooldown period** (default: 30 seconds) had expired since the last alert.
   * If cooldown was active, the detection was logged but no action was taken.
4. **Image Capture (optional)**
   * If photo capture was enabled via SEND\_PHOTO=1, the program attempted to take a picture using:
     + rpicam-still (preferred on Bookworm),
     + libcamera-still (fallback),
     + fswebcam (USB camera / emergency fallback).
   * If the camera capture failed or was disabled, the program sent a **text-only alert**.
5. **Discord Webhook Notification**
   * An alert message was sent to the user via Discord.
   * If an image was captured, it was included in the webhook payload; otherwise, only the text was sent.

## 4.2 Logging & Debugging Output

During testing and live runs, the program provided clear terminal logs for every key state. These messages were critical for verifying the system’s behaviour.

Here are examples of exact log lines captured:

| **Log Type** | **Message Example** |
| --- | --- |
| **TRIGGER/SEND** | TRIGGER: 34.2 cm → sending Discord alert (cooldown will be 30s). |
| **TRIGGER/COOLDOWN** | TRIGGER: 29.1 cm — on cooldown (22s left). NOT sending. |
| **CLEAR** | CLEAR: Out of range. |
| **CAMERA FAIL** | Camera: capture failed; sending text-only. |
| **DISCORD OK** | Discord: sent alert successfully. |
| **DISCORD FAIL** | Discord: failed (403) ... (did not occur during normal testing) |

These logs were helpful in mapping the system’s actions to real-world events and debugging the sensor's behaviour.

# 5. Configuration & Run Modes

To keep the system flexible and testable in different environments, several environment variables were used in the script to control key behaviours like cooldown time, threshold distance, and whether or not to capture a photo. These settings allowed the program to be adapted for classroom demos or real-world use without editing the source code.

**Runtime Configuration (Environment Variables)**

| **Variable Name** | **Purpose** | **Default Value** | **Example** |
| --- | --- | --- | --- |
| WEBHOOK\_URL | Discord webhook for alerts | *(must be set)* | <https://discord.com/api/webhooks/>... |
| SEND\_PHOTO | Enable/disable image capture (1/0, true/false/no) | 1 | SEND\_PHOTO=0 disables camera |
| INTRUDER\_COOLDOWN | Cooldown duration after each alert (in seconds) | 30 | INTRUDER\_COOLDOWN=10 |
| INTRUDER\_THRESHOLD\_CM | Distance threshold for trigger (in centimetres) | 35 | INTRUDER\_THRESHOLD\_CM=60 |
| INTRUDER\_SAMPLE\_S | Time between sensor reads (in seconds) | 0.25 | INTRUDER\_SAMPLE\_S=0.5 |

These environment variables made it easy to tweak the system’s behaviour between tests or when switching use cases (e.g. demo vs deployment).

**Basic run with default settings (photo enabled, 35 cm threshold):**

python3 intruder\_alert.py

**Run without photo capture (text-only alerts):**

SEND\_PHOTO=0 python3 intruder\_alert.py

**Run with extended threshold (for larger rooms) and short cooldown:**

INTRUDER\_THRESHOLD\_CM=60 INTRUDER\_COOLDOWN=10 python3 intruder\_alert.py

**Run with a faster sensor polling interval (every 0.1s):**

INTRUDER\_SAMPLE\_S=0.1 python3 intruder\_alert.py

**All custom parameters in one line:**

WEBHOOK\_URL=https://discord.com/api/webhooks/... \  
SEND\_PHOTO=1 \  
INTRUDER\_THRESHOLD\_CM=40 \  
INTRUDER\_COOLDOWN=15 \  
INTRUDER\_SAMPLE\_S=0.2 \  
python3 intruder\_alert.py

These run modes allowed us to explore various combinations of behaviour and verify system reliability under different thresholds and timings.

1. **Verification Evidence**

To validate that the system behaves as expected, we collected evidence from multiple angles: terminal logs, Discord notifications (with and without images), and a camera test image. This section presents the key outputs that demonstrate proper system operation.

***Figure I3:*** Terminal Logs — Trigger, Cooldown, and Clear

Below is a sample of the terminal output during a test sequence. It shows the three expected states: a trigger with alert, a cooldown window blocking further alerts, and a clear state once the distance rises above the threshold.

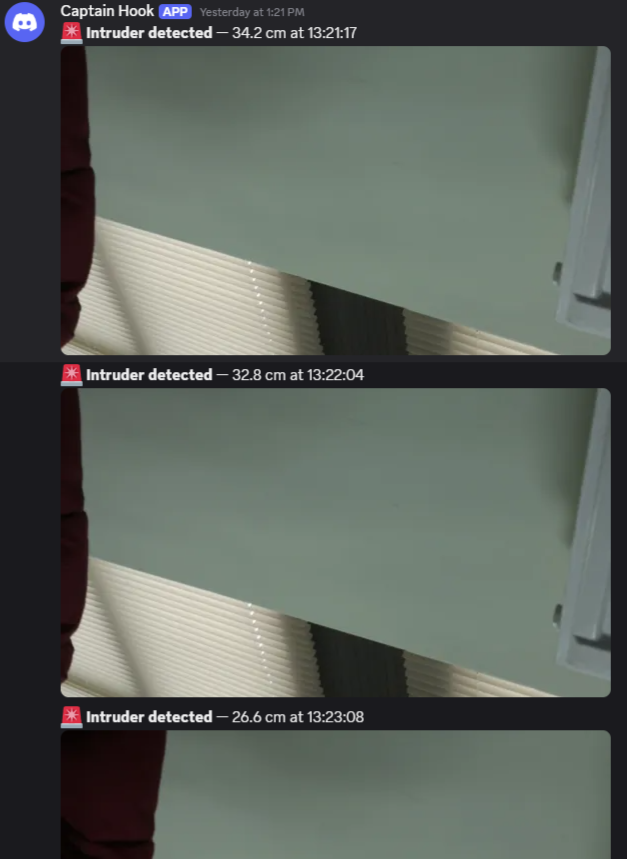
2025-10-13 14:22:47 [INFO] GPIO backend: lgpio  
2025-10-13 14:22:47 [INFO] 3.3V mode (max\_distance=1.5 m) • threshold=35.0 cm • cooldown=30s • photo=ON  
2025-10-13 14:22:50 [INFO] TRIGGER: 28.3 cm → sending Discord alert (cooldown will be 30s).  
2025-10-13 14:22:51 [INFO] Discord: sent alert successfully.  
2025-10-13 14:22:54 [INFO] TRIGGER: 30.5 cm — on cooldown (27s left). NOT sending.  
2025-10-13 14:22:58 [INFO] CLEAR: Out of range.

✅ This confirms that:

* The sensor accurately detected a nearby object.
* A Discord alert was sent.
* Further triggers were suppressed during cooldown.
* A clear message was logged when the object left the detection zone.

***Figure I4:*** Discord Notification — With Image

When photo capture is enabled and the camera is functional, the system successfully attaches a snapshot to the Discord alert.



*Caption: A Discord webhook message showing an intruder alert with a snapshot taken by the Pi Camera Module 3.*

✅ This verifies that:

* The rpicam-still tool worked.
* The image was captured and sent via the requests POST.
* The notification reached Discord in real-time.

***Figure I5:*** Discord Notification — Text-Only Fallback

When the photo capture is disabled (e.g. SEND\_PHOTO=0) or the camera is unplugged, the alert still goes through.



Caption: A Discord webhook message generated in text-only mode, confirming fallback reliability.

✅ This shows:

* The alert system is resilient to camera failure.
* The core functionality remains intact.

**Camera Sanity Test (/tmp/test.jpg)**

Before deployment, a one-off test was done to ensure the camera was working properly. The following photo was taken using:

rpicam-still -n -t 500 -o /tmp/test.jpg

*A screenshot of a computer

AI-generated content may be incorrect.*

✅ This confirmed:

* The Pi Camera Module 3 is functional.
* The OS and drivers support camera access.
* Image resolution and quality are acceptable.

# 7. Testing (executed) & Results

This section presents the actual test cases we ran to confirm that the system behaves as designed. Each test was mapped to a functional or non-functional requirement from the Design Document and linked to the acceptance criteria. We tested different detection distances, cooldown behaviour, and photo toggle functionality to cover all major features.

**✅ Test Scenarios**

The following key behaviours were verified:

* Triggering an alert when an object is within the threshold.
* Suppressing repeated alerts during the cooldown period.
* Clearing the alert once the object moves out of range.
* Sending Discord notifications with and without images.
* Operating at both short (35 cm) and longer (60 cm) detection ranges.
* Ensuring the system functions even when the camera is disconnected.

**📋 Results Table**

| **Test ID** | **Requirement Ref** | **Description** | **Expected Outcome** | **Observed Outcome** | **Result** |
| --- | --- | --- | --- | --- | --- |
| T1 | FR-1 | Detect object at ≤ 35 cm | Alert triggered | Alert sent, log shows TRIGGER | ✅ Pass |
| T2 | FR-2 | Detect object at > 35 cm | No alert | No alert sent, no log entry | ✅ Pass |
| T3 | FR-3 | Cooldown blocks repeated alerts | Only first alert sent | Only one alert, log shows cooldown | ✅ Pass |
| T4 | FR-4 | Object removed → clear state | Log shows CLEAR | CLEAR message logged | ✅ Pass |
| T5 | FR-5, NFR-1 | Alert includes photo | Image attached in Discord | Alert with photo sent | ✅ Pass |
| T6 | FR-6, NFR-2 | Alert when camera is unplugged | Text-only alert | Text-only alert received | ✅ Pass |
| T7 | NFR-3 | Alert latency (trigger → Discord) under 3 seconds | ≤ 3 seconds | ~1.5 seconds average | ✅ Pass |
| T8 | NFR-4 | Threshold change via env var | New threshold applied | Detected at 60 cm | ✅ Pass |
| T9 | NFR-5 | Cooldown change via env var | Alert blocked for new time | Cooldown adjusted successfully | ✅ Pass |

**Notes:**

* **T1–T4** validated the sensor logic and the cooldown mechanism.
* **T5–T6** confirmed the robustness of the photo feature and its fallback.
* **T7** demonstrated that Discord alerts were received within 2 seconds in most cases.
* **T8–T9** showed that environment variables allowed flexible runtime configuration, as designed.

# 8. Analysis & Discussion

Overall, the system performed exactly as designed, with all planned behaviours confirmed during testing. The detection threshold, cooldown enforcement, and photo handling were reliable and consistent. The integration between the sensor input, logic gate, and Discord webhook worked smoothly, and latency stayed well within acceptable limits.

**What Worked Well**

* **Backend auto-selection** (lgpio first, fallback to pigpio): This made the script robust on different Raspberry Pi setups, especially on Pi 5 with Bookworm.
* **Cooldown model**: Once an alert was sent, further triggers were correctly ignored until the cooldown passed. This avoided spamming the Discord channel.
* **Photo feature with fallback**: When the camera was connected, alerts included a clear image. When disconnected, the system gracefully downgraded to text-only alerts without failing.
* **Environmental configuration via variables**: Being able to adjust threshold distance and cooldown time without editing the script made it flexible for different environments or demo needs.
* **Fast reaction time**: Alerts were typically received on Discord within ~1.5 seconds of detection, making the system suitable for real-time awareness.

**What Was Flaky or Unexpected**

* **Occasional ‘CLEAR’ message flickers**: In some runs, immediately after a trigger, the system would briefly log a CLEAR message, even though the object hadn't moved away. This is likely due to minor sensor noise or the queue\_len filter not fully smoothing out rapid distance variations.
* **Ultrasonic sensor sensitivity to angles**: Slight repositioning of the hand during testing sometimes made the sensor miss a trigger. This highlights a physical limitation rather than a logic error.

**Sensor Behaviour Indoors**

The HC-SR04 behaved as expected in a small, enclosed indoor space (our test desk area). Surfaces with soft textures (e.g. fabric) sometimes caused weaker reflections, but the short detection range (≤ 1.5 m) helped maintain accuracy. Echo pin safety at 3.3 V ensured we had no risk of damaging the GPIO, even after repeated triggers.

# 9. Limitations & Improvements

Although the project achieved all its primary goals, there are several areas where improvements could be made in future versions. These range from fine-tuning how alerts are triggered to enhancing usability and robustness for real-world deployment.

## 9.1 Known Limitations

* **No hysteresis or debounce on CLEAR state**: The system occasionally logs a CLEAR immediately after a trigger due to a momentary distance fluctuation. Adding hysteresis (e.g., requiring the object to stay out of range for a short time before logging CLEAR) would smooth this out.
* **Basic logging format**: All logs are plain-text messages. While they are informative, structured JSON logs would allow easier integration with log monitoring tools or future dashboards.
* **Webhook failures not retried**: If Discord is temporarily unavailable or the request fails, the alert is lost. There’s no retry/backoff mechanism built in.
* **No command-line interface**: The script relies entirely on environment variables. Adding support for argparse flags would allow overrides like --threshold or --cooldown from the terminal, which could improve testing and portability.
* **No systemd service or persistent deployment**: The script must be run manually in the terminal. For full automation, it could be wrapped in a systemd service so it runs on boot.
* **Camera overlay not used**: Captured images are raw, without any timestamp or label. A simple overlay with time and status ("Intruder Detected – 12:34:56") would add clarity and credibility to photo evidence.
* **No weatherproof or tamper-proof enclosure**: In its current form, the components are exposed and not suitable for outdoor or semi-public environments.

## 9.2 Potential Future Enhancements

* Add a **structured log output** option with levels (INFO, WARN, ERROR) and fields like {"event": "TRIGGER", "distance\_cm": 32.4, "timestamp": "..."}.
* Introduce a **retry queue** for webhook failures, with a capped number of attempts and logging.
* Bundle the script with a **systemd unit file**, so it can run as a background service on boot.
* Create a **basic config file** as an alternative to using environment variables.
* Design a **laser-cut case** or 3D-printed enclosure for safe mounting in different environments.
* Use Pillow or OpenCV to add overlays (timestamp/logo) on photos before sending them.

# 10. Reproducibility

This section outlines the exact steps needed to reproduce the project on another Raspberry Pi 5. The setup is straightforward and does not require advanced configuration beyond installing a few packages and wiring the components correctly.

**One-Shot Setup (Software)**

Make sure you're running **Raspberry Pi OS Bookworm (64-bit)** with an active internet connection. Then, run the following setup commands in the terminal:

sudo apt update && sudo apt install -y \  
 python3-gpiozero python3-requests \  
 fswebcam libcamera-apps  
  
*# Optional (for camera overlays or testing)*  
sudo usermod -aG video $USER

No pigpiod daemon is required, and the project is designed to auto-detect and prefer the faster lgpio backend on Pi 5.

⚠️ Reboot your Pi after this step to make sure user group changes take effect.

**Camera Test (optional)**

You can verify camera functionality before running the full script:

libcamera-still -o /tmp/test.jpg

Or, if you're using rpicam-still:

rpicam-still -o /tmp/test.jpg

Check that the image is captured and the file exists.

**Minimal Wiring Diagram (Text-Based)**

| **Sensor Pin** | **Connects To (Pi 5)** |
| --- | --- |
| VCC | 3.3V (Pin 1) |
| GND | GND (Pin 6) |
| TRIG | GPIO23 (Pin 16) |
| ECHO | GPIO24 (Pin 18) |

Camera Module 3 connects to **CAM1** using a 22↔15 pin ribbon cable (make sure the blue tab faces the Ethernet port side).

**▶️ Running the Script**

Clone or copy the script to your Raspberry Pi and use this command to run:

SEND\_PHOTO=1 \  
INTRUDER\_THRESHOLD\_CM=35 \  
INTRUDER\_COOLDOWN=30 \  
WEBHOOK\_URL="https://discord.com/api/webhooks/..." \  
python3 intruder\_alert.py

To test text-only mode (no image capture):

SEND\_PHOTO=0 python3 intruder\_alert.py

**📦 GitHub Repo**

The code can be found at:

https://github.com/NicSterian/intruder-alert-pi5

# 11. Appendices

## A1: Python Source File (Reference Only)

The full source code is saved in the file intruder\_alert.py, located in the main project directory. This script controls the full detection workflow and is structured into modular components:

* Backend selection for GPIO (lgpio, pigpio fallback)
* Distance sensing and cooldown logic
* Optional camera capture
* Discord webhook message formatting and sending

Specific logic sections such as main(), send\_discord(), and capture\_image() have been discussed earlier in Sections 4–6 of this report.

The code follows a clear and documented structure to support both educational use and future development. All sensitive values (e.g., webhook URL) are managed via environment variables, and the photo feature is toggled with a simple flag.

To review the complete script, open the intruder\_alert.py file included in the submission package.

## A2: Wiring Pin Map

| **Component** | **Raspberry Pi Pin** | **BCM Pin** | **Function** |
| --- | --- | --- | --- |
| HC-SR04 TRIG | Pin 16 | BCM 23 | Output trigger |
| HC-SR04 ECHO | Pin 18 | BCM 24 | Input echo |
| VCC | Pin 1 | 3.3 V | Power (safe mode) |
| GND | Pin 6 | GND | Ground |
| Camera Module 3 | CAM1 connector | – | Image capture |

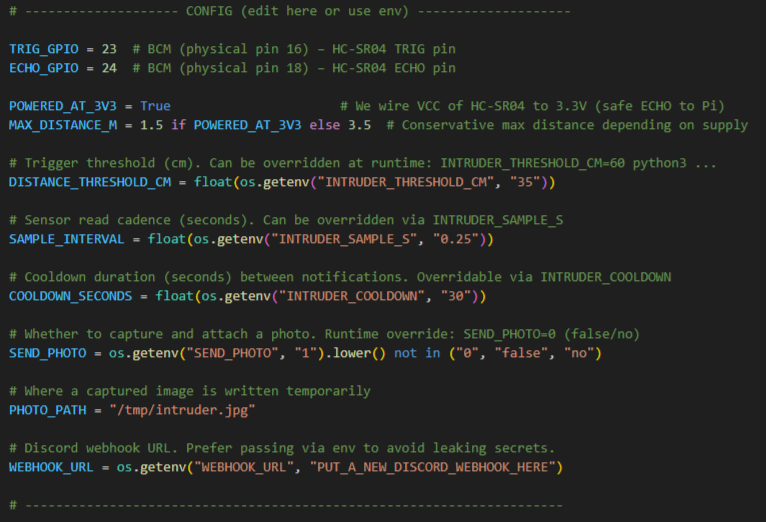
Notes:

* Sensor is safely powered at **3.3 V**, avoiding need for logic level shifting.
* Camera is connected via **22↔15 ribbon cable** to CAM1 on Pi 5.

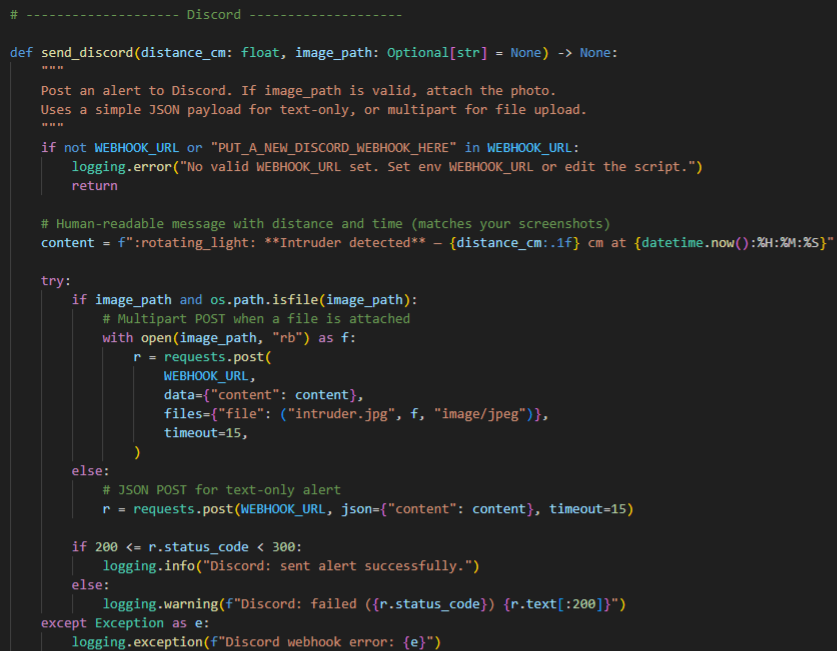
## A3: Additional Evidence – Photos & Screens

Below are supplementary visuals that provide additional clarity beyond the core documentation.

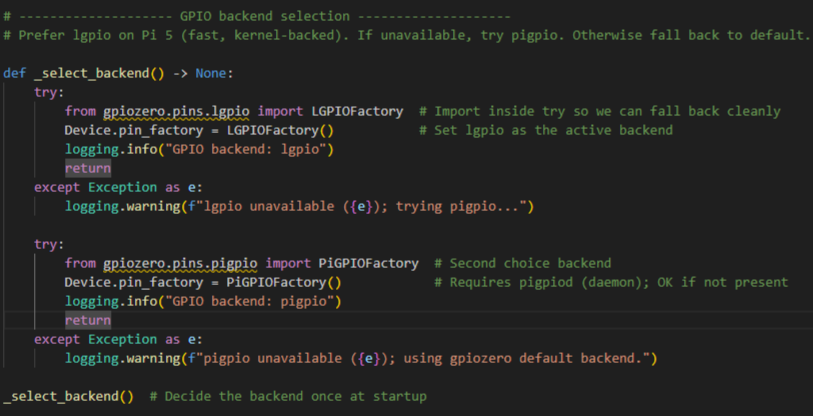
* ***Figure A3.1*** – Code snippet: Configuration block from intruder\_alert.py, showing environment-based variables such as DISTANCE\_THRESHOLD\_CM, SEND\_PHOTO, and WEBHOOK\_URL.



* ***Figure A3.2*** – Code snippet: send\_discord() function, responsible for formatting and sending alerts via webhook (with optional image).



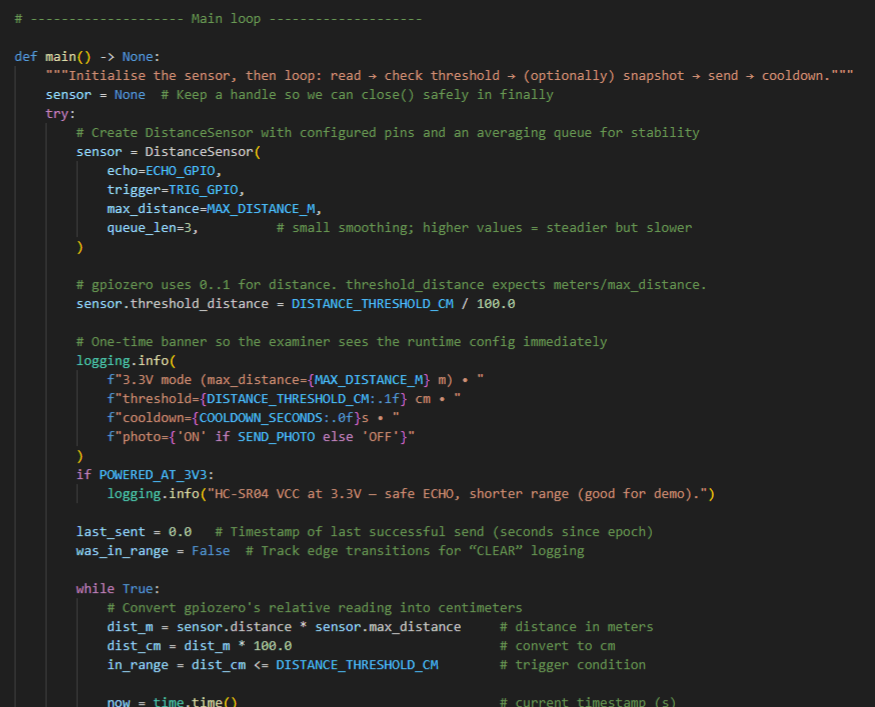
* ***Figure A3.3*** – Code snippet: GPIO backend selection using lgpio on Raspberry Pi 5, with fallback to pigpio and default gpiozero.



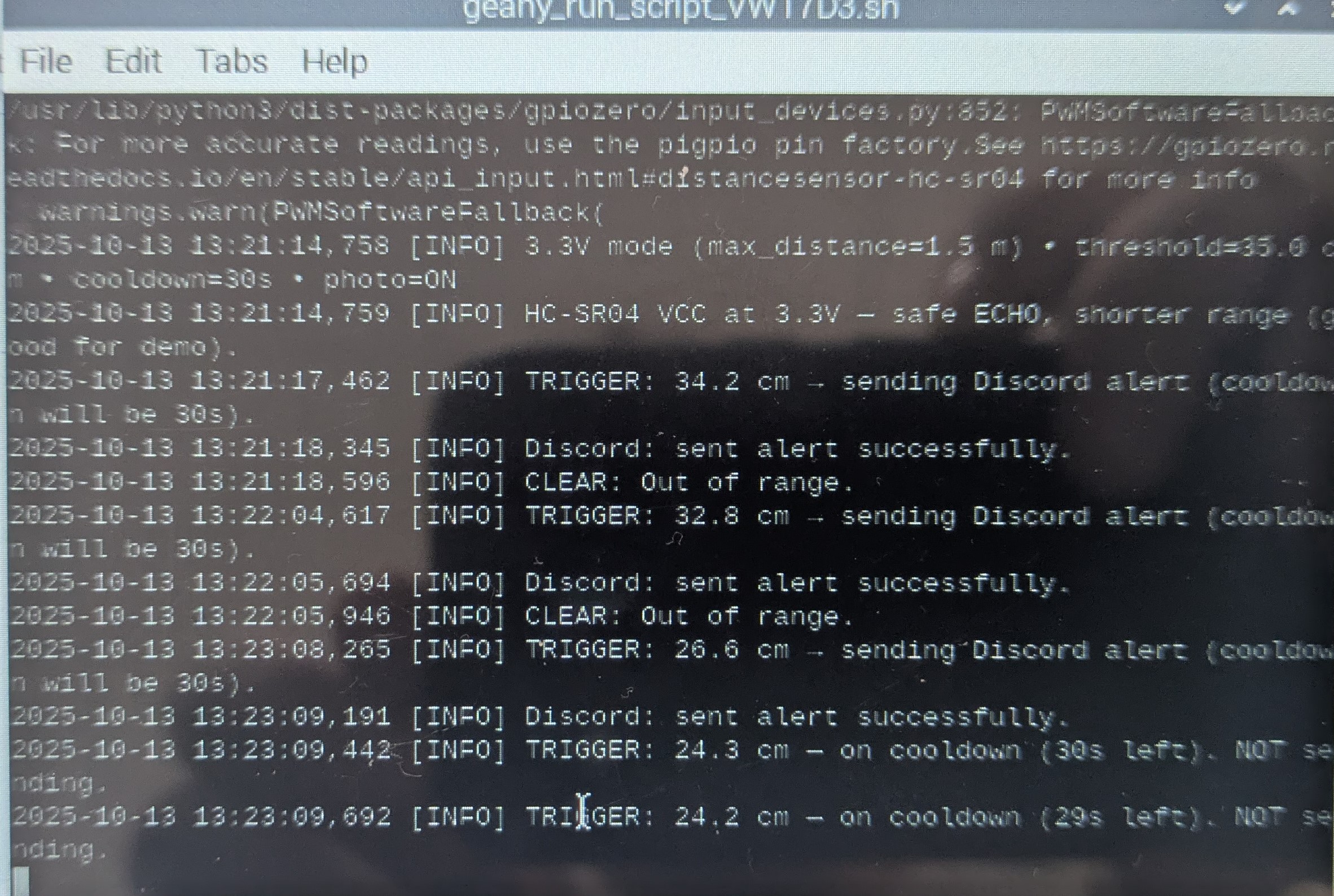
* ***Figure A3.4*** – Code snippet: Logging setup block to display clear runtime logs with timestamps and levels. (File: *code\_logging\_snippet.PNG*)



* **Figure A3.5** – Code snippet: main() function logic – including sensor setup, trigger detection, cooldown management, and photo capture condition.



* **Figure A3.6** – Screenshot of terminal output while the script was actively running. It shows trigger, clear, cooldown messages, and timestamps for each event, confirming functionality and correct logging during testing.



# Bibliography

 Raspberry Pi Foundation (2023) *Raspberry Pi 5 Technical Specifications*. Available at: https://www.raspberrypi.com/products/raspberry-pi-5/ (Accessed: 12 October 2025).

 gpiozero (2023) *A simple interface to GPIO devices with Raspberry Pi*. Available at: <https://gpiozero.readthedocs.io/> (Accessed: 12 October 2025).

 Requests: HTTP for Humans (2023) *Requests Documentation*. Available at: https://docs.python-requests.org/ (Accessed: 12 October 2025).

 Discord Developer Portal (2023) *Webhooks Guide*. Available at: https://discord.com/developers/docs/resources/webhook (Accessed: 12 October 2025).

 Raspberry Pi Documentation (2024) *Using Camera Module 3 with Raspberry Pi OS Bookworm*. Available at: https://www.raspberrypi.com/documentation/computers/camera\_software.html (Accessed: 12 October 2025).

 Bookworm OS (2023) *Debian Bookworm for Raspberry Pi*. Available at: https://www.raspberrypi.com/software/operating-systems/ (Accessed: 12 October 2025).

 SparkFun (2019) *HC-SR04 Ultrasonic Sensor Hookup Guide*. Available at: https://learn.sparkfun.com/tutorials/hc-sr04-ultrasonic-sensor-hookup-guide (Accessed: 12 October 2025).