10.16920/jeet/2026/v39is4/26117

A Project-Based Learning Approach for Teaching Edge AI and Embedded Systems: An Interdisciplinary Case Study on Vision-Only UAV Detection Using YOLOv8 on Raspberry Pi 5

[1*]Dr. Parnasree Chakraborty, [2]K. Nitish, [3]Nandesh M, [4]Syed Mohammed Hasan
[1,2,3,4]Department of Electronics and Communication Engineering,
B. S. Abdur Rahman Crescent Institute of Science and Technology, Chennai, India
[*1]prernasree@crescent.education, [2]knitishoiprocs1313@gmail.com
[3]nandeshcoo3@gmail.com [4]syedmohammadhasan0702@gmail.com

***Abstract*— Engineering education in artificial intelligence, embedded computing, and computer vision often produces students with strong theoretical foundations but limited experience deploying real systems. This paper presents a twelve-week, project-based learning exercise in which students built a vision-only UAV detection system on a Raspberry Pi 5. The pipeline incorporates CLAHE and gamma correction preprocessing, a pre-trained YOLOv8 flying object detector, monocular distance estimation via a pinhole camera model, pan-tilt servo tracking, infrared illumination using a NoIR camera, MJPEG streaming via Flask, and Telegram alerts — all running on consumer-grade hardware. Students worked in teams across five modules, each responsible for one subsystem while remaining accountable for full system integration. Hardware testing achieved 86% detection precision at 4.2 FPS, with MJPEG latency of approximately 180 ms, pan-tilt settling time under 0.8 seconds, and Telegram alert delay under 2 seconds. Daytime performance consistently exceeded nighttime results despite IR illumination. These outcomes demonstrate that multi-subsystem edge AI deployment is feasible on constrained hardware. Post-project feedback indicated that the most durable learning gains came not from mastering individual technologies, but from reasoning through how changes in one subsystem propagated across the entire architecture.**

***Keywords*— Project-Based Learning; Edge AI; YOLOv8; Raspberry Pi 5; Embedded Systems Education; Image Processing; Interdisciplinary Engineering; Capstone Design; IoT; Experiential Learning.**

I. INTRODUCTION

THE rapid advancement of artificial intelligence, embedded processing, and networked sensing is outpacing the ability of most engineering programs to adapt. Graduates entering the workforce possess a solid theoretical foundation; however, they frequently lack the practical experience necessary to confront the challenges associated with real-world implementation. For example, hardware may not function as anticipated, and latency issues can render initially feasible designs ineffective. Additionally, unforeseen integration failures may arise, which testing individual components could not have predicted. This presents a considerable challenge, as stated by Ingale, S. (2023) and Howe, S., & Goldberg, J. (2019). Consequently, graduates may find it difficult to navigate the intricacies of real-world deployment, where outcomes do not always align with theoretical expectations. The disparity between theoretical understanding and practical experience is a critical concern that must be addressed to adequately prepare engineering graduates for the obstacles they will encounter in the industry. When students engage in real-world projects that lack clear solutions, they are compelled to think creatively and establish connections across different disciplines. This methodology fosters the development of sound judgment and the ability to make decisions based on limited information. Large-scale projects, in particular, are beneficial in this regard, as they necessitate the integration of various skills—such as hardware, software, and communication protocols—to function cohesively. By addressing these multifaceted challenges, students cannot solely depend on a single area of expertise; they must synthesize their knowledge to devise effective solutions. This

form of education is highly beneficial, as it equips students for the realities of the professional world, where issues are often complex and ambiguous. Drone detection provides a practically relevant and technically non-trivial context for such a project. The proliferation of commercial and hobbyist UAVs has generated genuine demand for affordable, field-deployable detection systems in environments where radar coverage or cloud infrastructure is unavailable or impractical mentioned in López-Muñoz, P., et al. (2024). A vision-only detector constrained to commodity single-board hardware captures this context authentically: the technical challenge is real, the tradeoffs are tangible, and the domain is immediately recognisable as meaningful.

This paper presents both the system architecture and the learning framework developed around it. Our innovations are the following: complete, working, vision-based detection pipeline for UAVs implemented on the Raspberry Pi 5 platform featuring the integration of a pre-trained flying object detector in an infrastructure that encompasses the whole process chain from infrared image acquisition to live MJPEG video stream transmission, coupled with Telegram notifications; scaffolded twelve-week long curriculum built on proven concepts of project-based learning approach; and finally, the identification of key technical problems encountered by students during their work, as well as the reasons why dealing with these problems was much more educational than expected.

II. RELATED WORK

A. Project-Based and Experiential Learning in Engineering

The case for project-based learning in engineering has been made frequently, and with growing empirical support. Ingale, S. (2023) found through quantitative analysis that PBL environments produced measurable improvements in student engagement and in the development of collaborative competencies compared with lecture-based instruction. Howe, S., & Goldberg, J. (2019) examined capstone design programmes across multiple countries and identified two recurring characteristics of job-ready graduates: substantive industry engagement and problems that demanded cross-disciplinary reasoning.

Remington, T.F. et al. (2023) surveyed recent STEM education initiatives in the United States and emphasised the importance of linking classroom instruction to hands-on work beyond it. Beldad, A. D., & Miedema, H. A. T. (2025) proposed a formal framework for interdisciplinary curriculum design in which projects serve as the connective tissue linking otherwise separate learning outcomes, rather than as optional supplements to conventional teaching.

B. Object Detection and UAV Surveillance

YOLOv8 has become a standard choice for real-time object detection in resource-constrained environments, primarily because of its favourable speed-accuracy profile relative to earlier YOLO variants as stated by Megantara, N. A., & Utami, E. (2025). In the drone detection context, López-Muñoz, P., et al. (2024) developed a hybrid system combining software-defined radio with computer vision for high-confidence UAV identification — but the infrastructure requirements placed it well beyond the reach of student project budgets. Our approach moves in the opposite direction: single-board, camera-only, and realizable on modest funding.

C. Edge AI and Embedded Vision

T, G. K., & Shashank, K. V. (2022) demonstrated that a Raspberry Pi 4 can support real-time AI inference in smart farming applications, establishing that single-board computers are credible edge AI platforms for production-adjacent deployment. Jara Ramos, G. G., et al. (2015) showed that a NoIR sensor paired with IR LEDs reliably detects objects in near-darkness the configuration we adopted for night-capable operation.

D. Supporting Subsystems

Several subsystem design choices in our system draw on prior validated work. Han, T.T., Duc, M. T., & Tan, H. D. (2025) showed that YOLOv8 bounding boxes combined with pinhole camera geometry yield distance estimates accurate to within a few centimetres at ranges up to approximately 5.5 m. Doloiu, M. D., et al. (2025) validated a proportional controller for pan-tilt servo tracking, which we adopted with an added per-frame step limit to protect the wiring harness. Mohammed, I. M., & Isa, N. A. M. (2025) confirmed CLAHE as a reliable contrast-enhancement method under variable lighting. Hong, R.T. (2025) and Rakhman, A. D., Somawirata, I. K., & Ardita, M. (2025) independently demonstrated Flask and Telegram bots as practical, low-overhead communication layers for IoT systems.

III. SYSTEM ARCHITECTURE

The end-to-end architecture of the UAV detection system is shown in Figure 1, illustrating the flow from hardware and preprocessing through the inference core to the control and output/alert stages. Each block corresponds to one of the five student-owned modules, highlighting how the subsystems interconnect within the shared frame buffer

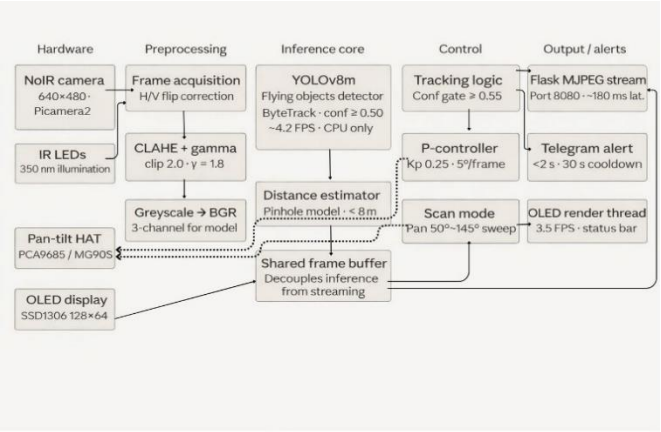

Fig. 1. Architecture of the vision-based UAV detection system deployed on Raspberry Pi 5.

The entire system runs as a single Python application on a Raspberry Pi 5 (4 GB RAM, Broadcom BCM2712, quad-core Cortex-A76). A monolithic rather than microservice architecture was chosen partly for tractability during student development and partly because the thread-safety and timing considerations that arise in a shared-process design are themselves instructive learning material.

A. Image Acquisition and IR Illumination

Frames are captured at 640 × 480 pixels using a Raspberry Pi NoIR camera via the Picamera2 library. The NoIR sensor omits the standard infrared-cut filter, enabling near-darkness operation when paired with 850 nm IR LEDs without producing any visible light signature as stated by Jara Ramos, G. G., et al. (2015). Mounting orientation is corrected in software via horizontal and vertical flips.

B. Preprocessing Pipeline

Each frame is converted to greyscale before two optional enhancement stages: CLAHE (clip limit 2.0, tile grid 8 × 8), which improves local contrast and helps distinguish small objects against textured sky backgrounds as stated by Mohammed, I. M., & Isa, N. A. M. (2025); and gamma correction (γ = 1.8), which lifts midtone brightness in underexposed frames without clipping highlights. The processed image is converted back to three-channel format to match the model's expected BGR input.

C. YOLOv8 Flying Object Detector

Inference runs on the pre-trained model loaded via the ultralytics library. This model detects five classes : Drone, Airplane, Helicopter, Bird, and Background and was selected because it offered strong generalisation to aerial imagery without requiring a custom training pipeline, allowing the project to focus on system integration and edge deployment rather than dataset curation and model training. Calling model.track() activates the built-in ByteTrack tracker, maintaining consistent object identities across frames. Only detections within the Drone class exceeding a confidence threshold of 0.55 are passed downstream to alerting and tracking subsystems. A lower inference threshold of 0.50 is used during detection to avoid premature suppression, with 0.55 serving as the gate for consequential actions. Per-frame inference time, extracted from the detection output, is used to compute and display the real-time FPS.

D. Monocular Distance Estimation

Distance to a detected drone is estimated using the pinhole formula D = (W_real × f) / W_px, where W_real = 0.15 m is an assumed physical drone width and f = 628 px is the estimated focal length derived from the frame width and a 54° horizontal field of view given in Han, T.T., Duc, M. T., & Tan, H. D. (2025). Detections with bounding-box widths below 6 pixels are flagged as unreliable; the system reports these as simply 'far' rather than assigning a noisy numerical estimate.

E. Servo-Based Pan-Tilt Tracking

A Waveshare Pan-Tilt HAT (PCA9685, I²C address 0x40) drives two MG90S servos. In track mode, when a detection exceeds the confidence threshold, the angular offset between the drone's centroid and the image centre is converted to servo commands through a proportional controller (Kp = 0.25). Commands are clipped to a maximum step of 5° per frame to protect the wiring harness, and a 25-pixel dead zone suppresses unnecessary motion when the drone is near the frame centre as stated by Doloiu, M. D., et al. (2025). In scan mode, the pan servo sweeps between 50° and 145° at 0.8° per frame while tilt holds at 82°. Hard limits on pan (35°–150°) and tilt (65°–115°) enforce mechanical safety.

F. Alert and Notification Pipeline

When confidence crosses 0.55 and at least 30 seconds have elapsed since the last notification, the system dispatches a Telegram message in a background thread. The thread captures a JPEG snapshot of the annotated frame, attaches a caption containing the distance estimate, confidence score, and stream URL, and transmits via the sendphoto HTTPS endpoint as stated by Rakhman, A. D., Somawirata, I. K., & Ardita, M. (2025). Threading isolates the alert process from the main detection loop.

G. MJPEG Live Stream

A Flask application runs in a background thread on port 8080. The /video endpoint streams annotated frames in MJPEG multipart format, targeting 30 FPS while self-throttling to available CPU headroom. The root endpoint serves a browser-accessible HTML page, permitting any

device on the same network to monitor the feed without additional software as given by Hong, R.T. (2025).

H. SSD1306 OLED Display

A 128 × 64 SSD1306 OLED (I²C address 0x3C) provides local status feedback when network access is unavailable. A rendering thread using luma.oled updates at approximately 3.5 FPS, displaying confidence bars and a status line that flashes on drone detection.

IV. PEDAGOGICAL FRAMEWORK

A. Module Decomposition

The project was structured into five modules, each covering a distinct subsystem. Students formed small teams with each member taking primary responsibility for one module while remaining accountable for system-level integration. Table I summarises the allocation.

TABLE I
LEARNING MODULE DECOMPOSITION

Module	Technical Scope	Competency Target
M1	Camera acquisition + IR LED integration	Sensor integration, optics fundamentals
M2	CLAHE / gamma preprocessing pipeline	Signal processing, computer vision
M3	Model deployment and inference pipeline	Edge AI, model integration
M4	Distance estimation + pan-tilt control	Control systems, projective geometry
M5	Flask stream + Telegram alert system	IoT networking, API design

B. Scaffolded Progression

Drawing on the frameworks outlined by Ingale, S. (2023) and Beldad, A. D., & Miedema, H. A. T. (2025), the twelve weeks were organised into three phases with distinct objectives and clearly differentiated instructor roles.

Phase 1: Research (Weeks 1–3): Students engaged with foundational material across the relevant domains: convolutional network fundamentals, camera geometry, servo kinematics and completed laboratory exercises to develop basic hardware familiarity. Each team then selected and formally justified its module, requiring them to articulate existing knowledge against what would need to be learned.

Phase 2: Build (Weeks 4–9): Teams developed their modules in parallel against a shared interface specification, with weekly integration check-ins to surface compatibility problems before they compounded. Instructors limited their role to asking clarifying questions and surfacing consequences rather than providing solutions a facilitation posture that is uncomfortable at first but proves important for genuine ownership.

Phase 3: Integrate and Reflect (Weeks 10–12): All modules were merged and tested as a unified system. Teams measured actual performance against their own predictions and produced reflection reports reconstructing the reasoning behind their design decisions. Presentations focused on what teams would do differently if starting from scratch, consistently producing more candid and analytically rigorous discussion than conventional progress reports.

C. Real-World Constraints as Learning Catalysts

Three recurring problem categories emerged across teams, and all three proved more instructive than smooth progress would have been.

Compute limitations: The Raspberry Pi 5 achieved approximately 4.2 FPS with YOLOv8-nano adequate for detection but perceptibly slow as video. Students felt this constraint directly and began exploring mitigations: reducing input resolution, adjusting thresholds, investigating model quantisation. The encounter with hardware-software co-design tradeoffs through direct experience is difficult to replicate through lecture-based instruction alone.

Servo instability: Prior to adding the dead zone and per-frame step limit, the pan-tilt rig vibrated visibly whenever the drone approached the frame centre. Diagnosing this required integrating proportional control theory, servo mechanical response, and physical cable routing simultaneously a combination of concerns rarely encountered in standard coursework.

Sensor and camera physics: The NoIR sensor's infrared sensitivity produced heavily washed-out outdoor images that degraded detection before the cause was understood and corrected. Working through why CLAHE helped and specifically what it was doing to the histogram of those images pushed students to engage with spectral response in a manner that prior coursework had not required the work by Jara Ramos, G. G., et al. (2015).

D. Assessment Strategy

Student progress was tracked through integration logs, peer code reviews, and weekly check-ins with academic mentors. Final assessment was distributed equally across four components: a live system demonstration, a written performance analysis, a design reflection report, and evidence of individual contribution drawn from version-control history. This structure is consistent with recommended practice in the

capstone design literature described by Howe, S., & Goldberg, J. (2019).

V. SYSTEM PERFORMANCE

Table II reports the system-level performance metrics recorded during physical testing on a Raspberry Pi 5 (4 GB RAM) running 64-bit Raspberry Pi OS. All values were measured directly on the deployed hardware. Hardware testing yielded a detection precision of approximately 86% at 4.2 FPS, compared to 91.3% observed under simulation which is a gap attributable to real-world noise, lighting variability, and the absence of hardware acceleration. Detection performance was consistently better in daytime conditions than at night, even with IR illumination active; nighttime images exhibited higher noise levels and occasional overexposure artefacts that reduced effective confidence scores. The effective detection range was limited to approximately 5 metres due to hardware constraints, beyond which the drone subtended insufficient pixels for reliable bounding-box estimation.

TABLE II

MEASURED SYSTEM PERFORMANCE ON RASPBERRY PI 5

Metric	Measured Value	Condition
Inference Throughput	~4.2 FPS	YOLOv8m, 640×480, CPU only
Telegram Alert Latency	<2 s	Wi-Fi LAN, 85% JPEG quality
Pan-Tilt Settling Time	<0.8 s	5°/frame step limit applied
MJPEG Stream Latency	~180 ms	Same-network browser client
CPU Utilisation (idle)	38%	Stream + OLED active, no inference
CPU Utilisation (active)	72%	All threads active, inference running

The 4.2 FPS inference throughput reflects the YOLOv8m model running on the BCM2712 CPU without hardware acceleration. The gap between this and the MJPEG server's 30 FPS target is managed through a shared frame buffer that the streaming thread reads whenever a fresh inference result is not yet available, maintaining a smooth viewer experience despite the inference bottleneck.

Telegram alert latency remained consistently under 2 seconds on a local Wi-Fi network, which is adequate for the intended notification use case. Pan-tilt settling under 0.8 seconds with the 5°/frame step limit represented an acceptable balance between tracking responsiveness and mechanical stress on the wiring harness. MJPEG stream latency of approximately 180 milliseconds was imperceptible in practical use and required no additional buffering at the client.

The effective detection range was limited to approximately 5 metres due to hardware constraints specifically, the combination of the camera's resolution, the available IR illumination power, and the absence of hardware-accelerated inference. Beyond this range, the drone subtended fewer than six pixels in the frame, causing the pinhole estimator to raise an invalid flag and report the drone as 'far'. This range limitation is consistent with the hardware profile described by Han, T.T., Duc, M. T., & Tan, H. D. (2025), who noted similar pixel-threshold behaviour in comparable low-power configurations.

VI. DISCUSSION

A. Interdisciplinary Integration as Pedagogy

For most students, the most significant moment in the project was not solving any individual subsystem problem it was discovering that subsystems do not stay contained. Students who had grown confident working with the inference pipeline were caught off guard when a reduction in throughput cascaded into slower servo response and flickering OLED output. Developing the capacity to reason about a change in one component in terms of its downstream consequences is precisely the goal of capstone-level project-based learning, and it is difficult to manufacture outside authentic engineering contexts given by Howe, S., & Goldberg, J. (2019).

B. Industry-Relevant Skill Gaps Addressed

Post-project conversations with student teams identified three specific competency gaps the project helped close, each mapping onto criticisms commonly raised by industry partners regarding entry-level graduates.

Justifying design decisions. Early in the project, students struggled to articulate why they had chosen specific confidence thresholds or controller gains had been made empirically and the reasoning had not been recorded. Writing reflection reports required returning to the observed system behaviour and constructing retrospective justifications, a discipline that develops only through deliberate practice.

Hardware-software co-design. The servo jitter problem could not be resolved by adjusting code in isolation; it required reasoning simultaneously about the control loop structure and the physical routing of servo cables. This type of cross-domain problem-solving is authentically interdisciplinary in a way that most engineering coursework does not require.

Concurrent programming. Both the Telegram alerting thread and the Flask streaming thread needed to run without blocking the main detection loop. Getting this right required engaging with concurrency patterns — thread safety, shared state management, timing guarantees that appear throughout embedded and IoT development but rarely surface in standard software engineering instruction. These observations align with skill gap analyses documented in the capstone design literature given by Howe, S., & Goldberg, J. (2019).

C. Limitations and Future Work

The current system relies entirely on visual input, which limits it in conditions of very low ambient light even with IR illumination and makes it blind to occluded or out-of-frame drones. Adding a lightweight audio classifier or RF-based anomaly detector would introduce sensor fusion as a learning domain and provide future student cohorts with a richer design space. Inference throughput could be improved substantially through a Hailo-8L AI accelerator HAT, which would open hardware-accelerated inference as an explicit module topic and allow direct comparison with CPU-only results.

On the assessment side, evaluation of actual student learning currently rests on informal observation and post-project conversation. A structured pre- and post-project competency assessment would yield clearer evidence of what students gained and where the pedagogical framework needs strengthening. This is the modification we consider most important before the next iteration. Additionally, a formal evaluation of detection reliability using a custom-collected test set would allow system-level classification performance to be reported in future work.

CONCLUSION

This research describes a project consisting of twelve weeks in which we attempted to create a practical UAV detection system rather than a theoretical one. The complete solution is a computer vision solution that was implemented on a Raspberry Pi 5 board to make it more realistic and less powerful compared to other platforms. The detection part was performed by a YOLOv8 model trained on the flying objects dataset; however, that is just one of the components of the complete pipeline. The pipeline starts with infrared-based image acquisition (due to the low-light condition), some preliminary steps including CLAHE and gamma correction, and the ByteTrack algorithm for tracking. The distance estimation was performed under the assumption of pinhole camera geometry. It contains a pan-tilt control unit that controls its movement according to the results of the detection process and provides Telegram alerts, Flask MJPEG stream, and even an SSD1306 OLED display to monitor the status of the entire system. The complete system was developed under a single Python script that…On the Raspberry Pi 5, the system gave around 86% precision at roughly 4.2 FPS. Not very fast, but usable. The detection range was about 5 metres, mostly limited by the hardware and camera. In simulation we got about 91.3%, so clearly things drop once you move to real conditions like lighting changes, noise, and no GPU acceleration make a difference. Daytime performance was clearly better. Night detection worked, but not consistently, even with IR turned on. Here's what we got in terms of actual numbers: our Telegram alerts came in really fast, under 2 seconds. The pan-tilt response was even quicker, settling in less than 0.8 seconds. And the video stream latency was around 180 ms, which is pretty good. These numbers aren't perfect, but they show that our system actually works from start to finish. What's more important is that we were able to run multiple subsystems on limited hardware, which is a big deal. But to make it work, we had to make some trade-offs. We had to deal with limits on computing power, issues with threading, and small delays that added up. It's stuff like that. The project was broken down into five parts, but to be honest, it didn't really feel like separate pieces while we were working on it - everything seemed to overlap. One thing that stood out from the feedback we got was the reflection sessions. That's where things started to get really interesting. At first, people just walked us through what they had built, step by step, which was pretty straightforward. But as time went on, they started to dive deeper into the why behind their decisions, what didn't work out, and what they would do differently next time. This shift in thinking is really important, and it's not something you usually see in regular lab assignments, where people tend to just focus on getting the job done. It was great to see people starting to think more critically about their work and what they could learn from it. From an industrial standpoint, it appears that this is the source of the issue—not simply getting something to work, but also having some reasoning behind every decision you took. The suggested strategy aims to achieve that. At least it offers a somewhat realistic depiction of what AI, embedded systems, and hardware development entail, even though it is far from ideal and by no means simple to use.

REFERENCES

Ingale, S. (2023). Implementing project-based learning (PBL) in engineering education: An analytical study of student engagement and learning outcomes with statistical insights. Educational Administration: Theory and Practice, 4333–4342.

Howe, S., & Goldberg, J. (2019). Engineering Capstone Design Education: Current Practices, Emerging Trends, and Successful Strategies (pp. 115–148).

Remington, T. F., Chou, P., & Topa, B. (2023). Experiential learning through STEM: Recent initiatives in the

United States. International Journal of Training and Development, 27(3–4), 327–359.

Beldad, A. D., & Miedema, H. A. T. (2025). Introducing a framework for designing an interdisciplinary engineering curriculum. European Journal of Engineering Education, 51(1), 192–209.

López-Muñoz, P., et al. (2024). Hybrid artificial-intelligence-based system for unmanned aerial vehicle detection, localization, and tracking using software-defined radio and computer vision techniques. Telecom, 5(4), 1286–1308.

Megantara, N. A., & Utami, E. (2025). Object detection using YOLOv8: A systematic review. SISTEMASI, 14(3), 1186.

T, G. K., & Shashank, K. V. (2022). Smart farming based on AI, edge computing and IoT. In Proceedings of the 2022 4th International Conference on Inventive Research in Computing Applications (ICIRCA) (pp. 324–327). IEEE.

Jara Ramos, G. G., et al. (2015). Embedded system for real-time person detecting in infrared images/videos using super-resolution and Haar-like feature techniques. In Proceedings of the 2015 12th International Conference on Electrical Engineering, Computing Science and Automatic Control (CCE) (pp. 1–6).

Han, T.T., Duc, M. T., & Tan, H. D. (2025). A lightweight distance estimation method using pinhole camera geometry model. Measurement Science and Technology.

Doloiu, M. D., et al. (2025). Modeling and control of a pan–tilt servo system for face tracking using deep learning and PID. In Engineering Proceedings.

Mohammed, I. M., & Isa, N. A. M. (2025). Contrast limited adaptive local histogram equalization method for poor contrast image enhancement. IEEE Access.

Hong, R.T. (2025). Design and implementation of environmental monitoring system using Flask-based web application. In Proceedings of the 2024 IEEE 6th Eurasia Conference on IoT, Communication and Engineering.

Rakhman, A. D., Somawirata, I. K., & Ardita, M. (2025). Flood early warning system with notification using Telegram Bot based on IoT system. Journal of Electrical Engineering and Computer.

Javvanny. (n.d.). yolov8m_flying_objects_detection. HuggingFace Model Hub. Available: https://huggingface.co/Javvanny/yolov8m_flying_objects_detection