Published by :
https://www.ijert.org/
An International Peer-Reviewed Journal

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 10 , October - 2026

IoT-Based SmartRack and Inventory Monitoring System

Project Guide: Mrs. R. S. Ghat

Sayali Shantinath Chougule, Soundarya Bipin Narade, Tanushka Rahul Power, Shreya Mukesh Raval

Department of Electronics and Telecommunication Engineering

D.K.T.E. Society's Yashwantrao Chavan Polytechnic, Ichalkaranji

***Abstract* - Inventory management is essential in retail shops, warehouses, laboratories, and manufacturing units. Manual stock checking requires time and effort and may lead to inaccurate inventory records. The proposed Smart Rack System combines IoT sensors and camera-based machine learning to monitor stock levels. An ESP32 will collect sensor data, while a camera and an object-detection model will identify products and estimate their quantities. A monitoring dashboard will display stock levels, low-stock alerts, and mismatch warnings. Sensor readings and camera-based estimates will be compared to identify possible discrepancies. The system will be tested under different lighting conditions and product arrangements. Its performance will be evaluated using actual product counts, detection accuracy, and response time to demonstrate an economical solution for intelligent inventory monitoring.**

***Index Terms* - SmartRack, Internet of Things (IoT), ESP32, machine learning, computer vision, object detection, inventory monitoring, sensor fusion.**

I. INTRODUCTION / BACKGROUND

Inventory management is essential in retail shops, warehouses, supermarkets, laboratories, and manufacturing industries. Traditional methods such as manual counting, registers, and spreadsheets require time and effort and may lead to inaccurate stock records. Proper inventory monitoring helps prevent shortages, reduce unnecessary purchases, and improve operational efficiency.

The Internet of Things (IoT) enables sensors and microcontrollers to collect and transmit real-time data. In the proposed system, an ESP32 will collect sensor readings, while a load cell connected through an HX711 module will measure changes in rack weight. However, weight measurements alone cannot accurately identify products or determine their quantities in all situations.

The proposed Smart Rack System integrates IoT-based sensing with camera-based machine learning. A camera captures product images, and an object-detection model identifies supported products and estimates their visible quantities. Sensor readings and visual estimates will be compared to identify possible stock discrepancies and generate mismatch warnings when required.

A monitoring dashboard will display estimated stock levels, sensor readings, low-stock alerts, and the latest update time. The ESP32 will handle sensor data collection, while a laptop or compatible edge computer will process images. The prototype will be tested under different conditions to evaluate its accuracy, performance, limitations, and suitability for intelligent inventory monitoring.

II. LITERATURE REVIEW

TABLE I. LITERATURE REVIEW

Sr. No.	Literature	Major Findings	Relevance to Proposed System
1	ESP32 Camera Driver Documentation – Espressif Systems [1]	Explains camera interfacing, image capturing and image frame processing using ESP32-compatible camera modules.	Supports camera interfacing and image acquisition for the proposed system.
2	Object Detection Documentation – Ultralytics YOLO [2]	Explains object detection using image and video streams with bounding boxes, class labels and confidence scores.	Supports automatic object identification and detection in the proposed system.

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 10 , October - 2026

Sr. No.	Literature	Major Findings	Relevance to Proposed System
3	OpenCV Computer Vision Resources – OpenCV [3]	Provides image processing, object tracking, image analysis and computer vision algorithms.	Supports image preprocessing and computer vision operations.
4	HX711: 24-Bit Analog-to-Digital Converter for Weigh Scales – Avia Semiconductor [4]	Describes a 24-bit ADC designed to convert load cell signals into digital weight measurements.	Supports accurate weight measurement using a load cell and HX711 module.
5	ESP32 Technical Documentation – Espressif Systems [5]	Describes ESP32 microcontroller features, GPIO interfaces, Wi-Fi and Bluetooth connectivity.	Supports hardware interfacing, sensor communication and wireless data transmission.
6	Python Software Foundation – Python Documentation [6]	Provides programming libraries and tools for image processing, data handling and application development.	Supports software development, processing and integration of system components.

Research Gap

The proposed system integrates ESP32-CAM-based image capturing, YOLO-based object detection, OpenCV image processing and load-cell-based weight measurement using the HX711 module. Python is used for software development and processing, while ESP32 supports hardware interfacing and communication. The main objective is to combine object identification and weight measurement into a single integrated system. The overall accuracy, detection performance and weight measurement reliability must be evaluated through prototype testing.

III. PROBLEM STATEMENT

Traditional retail stores face difficulties in manually counting products and managing stock accurately. Manual checking takes time and may cause errors or unexpected stock shortages. Therefore, a smart system using ESP32-CAM and AI is needed to identify products, estimate stock quantity, and generate low-stock alerts automatically.

IV. OBJECTIVES

1. To develop an IoT-based SmartRack system for real-time stock monitoring and management.
2. To identify and count products using ESP32-CAM and AI-based image processing.
3. To update stock details automatically and generate low-stock alerts when the quantity falls below the minimum level.
4. To display product details and stock information on a web dashboard, reducing manual work and inventory errors.

V. PROPOSED METHODOLOGY, TOOLS AND TECHNIQUES

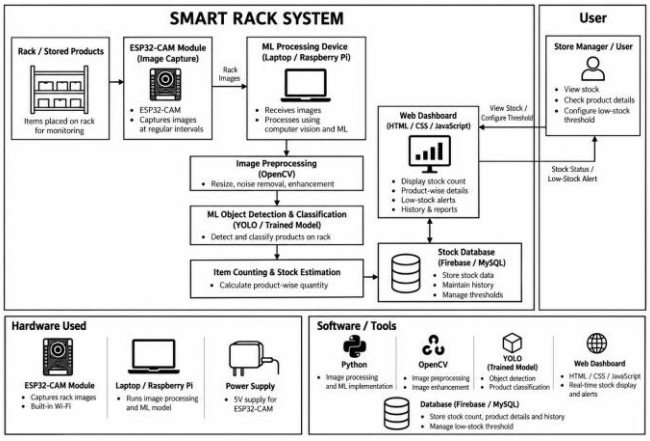


Fig. 1. Smart Rack System.

Working

1. Rack / Stored Products
   - Products are placed on the rack.
   - The system checks the products using a camera.
2. ESP32-CAM Module
   - ESP32-CAM captures images of the products.
   - It sends the images to a laptop or Raspberry Pi using Wi-Fi.
3. ML Processing Device
   - A laptop or Raspberry Pi processes the images.
   - It uses AI to identify different products.
4. Image Preprocessing
   - OpenCV improves the image quality.
   - It removes noise and resizes the images.

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 10 , October - 2026

5. Object Detection and Classification
   - The trained AI model identifies the products.
   - It recognises different products on the rack.
6. Item Counting and Stock Estimation
   - The system counts the products.
   - It calculates the stock quantity for each product.
7. Stock Database
   - The database stores product details and stock quantity.
   - It also stores stock history and minimum stock levels.
8. Web Dashboard
   - The dashboard displays stock information.
   - It shows product details and stock quantity.
   - It displays low-stock alerts and reports.
9. Store Manager / User
   - The store manager can check stock details.
   - The manager can set the minimum stock level.
   - The manager receives low-stock notifications.
10. Hardware Used
    - ESP32-CAM: Captures product images.
    - Laptop / Raspberry Pi: Processes images using AI.
    - Power Supply: Provides power to the ESP32-CAM.
11. Software Used
    - Python: Runs the program.
    - OpenCV: Processes images.
    - YOLO: Detects and identifies products.
    - Web Dashboard: Displays stock details and alerts.

VI. PROPOSED SYSTEM ARCHITECTURE AND WORKING

The proposed SmartRack System will operate through a sequence of sensing, image capture, processing, comparison, and display operations. The main functional blocks are described below.

TABLE II. STAGES OF THE PROPOSED SYSTEM

No.	Stage	Description
1	Product storage	Products are placed on or removed from the monitored rack section.
2	Sensor measurement	The connected sensor measures a physical parameter, such as the load applied to the rack.
3	Image capture	The camera captures an image of the products within its field of view.
4	Image processing	The ML model identifies supported product classes and estimates the count of visible items.
5	Data transmission	Sensor measurements, detection results, and time stamps are transmitted to the software application.
6	Result comparison	The application compares available sensor evidence with camera results and identifies potential inconsistencies.
7	Stock status generation	The system updates estimated stock information and checks configured low-stock thresholds.
8	Dashboard display	The user can view the latest monitoring results and receive warnings when stock is low or a discrepancy is detected.
9	Verification and record keeping	Uncertain results can be manually checked, and monitoring records can be retained for future analysis.

System Design Consideration

A single camera can count only the products visible within its field of view. Products hidden behind other items may not be detected. Load cells measure total load and cannot independently identify product types. Therefore, sensor and camera outputs should be treated as complementary evidence. A significant mismatch should trigger verification instead of automatically changing confirmed inventory records.

VII. HARDWARE AND SOFTWARE REQUIREMENTS

TABLE III. HARDWARE REQUIREMENTS

Sr. No.	Component	Quantity	Purpose
1	ESP32 development board	1	Sensor acquisition and communication
2	Compatible camera module, such as ESP32-CAM	1	Capturing images of rack products
3	Load cell	As required per monitored rack section	Measuring rack load

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 10 , October - 2026

Sr. No.	Component	Quantity	Purpose
4	HX711 module	1 per applicable load-cell interface	Converting load-cell signals into digital readings
5	Laptop or compatible edge computer	1	Running the ML model and monitoring software
6	Regulated power supply	As required	Supplying suitable voltage to components
7	Connecting wires and connectors	1 set	Electrical connections
8	Rack mounting and camera holder	1 set	Maintaining stable component placement
9	LED lighting, if required	1 set	Improving image capture consistency
10	Prototype rack and mounting materials	1 set	Holding products and hardware

The number and type of sensors will depend on the physical design of the rack. A single load cell may monitor an entire section, but it will not necessarily measure the quantity of every product individually.

TABLE IV. SOFTWARE REQUIREMENTS

Sr. No.	Software / Platform	Purpose
1	Arduino IDE	Developing and uploading ESP32 firmware
2	Python	Running the ML model and processing data
3	OpenCV	Image capture and image-processing functions
4	Suitable object-detection framework	Detecting and classifying products
5	HTML, CSS and JavaScript	Developing the web dashboard
6	Database or cloud platform, if required	Storing stock data and monitoring history
7	Serial monitor and debugging tools	Testing communication and sensor readings

VIII. APPLICATIONS

- **Retail Shops and Super markets:** Monitoring selected products and identifying potential low-stock conditions.
- **Warehouses and Storerooms:** Improving visibility of inventory in defined rack sections.
- **College Laboratories:** Monitoring the presence of electronic components, tools, and laboratory materials.
- **Small Manufacturing Units:** Tracking selected parts and consumables stored on racks.
- **Libraries and Equipment Rooms:** Monitoring supported item categories where visual recognition is suitable.
- **Small Distribution Centres:** Supporting periodic stock checks and identifying potential differences between physical stock and stored records.

IX. ESTIMATED PROJECT COST

The following is a preliminary prototype estimate. The actual cost will depend on the selected components, the number of monitored rack sections, and whether a laptop is already available.

TABLE V. ESTIMATED PROJECT COST

Sr. No.	Component	Quantity	Unit Cost (₹)	Total (₹)
1	ESP32 development board	1	500	500
2	Load cell with HX711 module	1 set	350	350
3	ESP32-CAM or compatible camera module	1	700	700
4	Regulated power supply and connectors	1 set	500	500
5	Camera mounting, rack materials and lighting	1 set	800	800
6	Miscellaneous prototype materials	1 set	500	500
	Estimated Total			₹ 3,350

Cost Note

This estimate excludes a laptop or edge computer on the assumption that one is already available. It also assumes a basic

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 10 , October - 2026

prototype with limited monitoring coverage. Additional load cells, a more capable edge device, or specialised rack construction would increase the cost. Component prices should be verified before purchase.

X. PROJECT TIMELINE

The following schedule is a proposed plan for completing the prototype and documentation. The duration may change depending on component availability, dataset preparation, model performance, and troubleshooting.

TABLE VI. PROJECT TIMELINE

Sr. No.	Activity	Duration	Status / Remarks
1	Topic finalization and requirement analysis	1 week	Proposed
2	Literature review and component selection	1 week	Proposed
3	System architecture and hardware design	1 week	Proposed
4	Hardware assembly and sensor calibration	1 week	Proposed
5	Image data set collection and annotation	1 week	Proposed
6	ML model configuration and dashboard development	2 weeks	Proposed
7	Hardware-software integration and testing	1 week	Proposed
8	Result analysis and performance evaluation	1 week	Proposed
9	Final documentation and presentation	1 week	Proposed

XI. CONCLUSION

This paper presents the proposed design of an IoT-based SmartRack and inventory monitoring system that combines ESP32-based sensing, load-cell measurement through the HX711 module, camera-based product detection, and a web dashboard. The system is intended to support stock estimation, low-stock alerts, and the identification of possible differences between sensor readings and visual counts. Since prototype evaluation is planned as future work in this paper, detection accuracy, response time, and measurement reliability must be established through implementation and experimental testing.

ACKNOWLEDGMENT

We express our sincere gratitude to our project guide, Mrs. R. S. Ghat, for her guidance, encouragement, and valuable suggestions during the planning and preparation of this project synopsis. Her support has helped us understand the technical requirements and organise the proposed work systematically. We also thank the Head of Department, faculty members, and D.K.T.E. Society's Yashwantrao Chavan Polytechnic, Ichalkaranji, for providing academic support and the opportunity to develop this project.

We acknowledge the developers and maintainers of the technical tools and documentation used in this synopsis, including the ESP32 ecosystem, Python, OpenCV, and object-detection frameworks. Finally, we extend our thanks to everyone who contributes to the design, implementation, testing, and documentation of the proposed Smart Rack System.

REFERENCES

[1] Espressif Systems, "ESP32 Camera Driver Documentation and Source Code." Available: https://github.com/espressif/esp32-camera
[2] Ultralytics, "Object Detection Documentation." Available: https://docs.ultralytics.com/tasks/detect/
[3] OpenCV, "OpenCV Computer Vision Resources." Available: https://opencv.org/
[4] Avia Semiconductor, "HX711: 24-Bit Analog-to-Digital Converter for Weigh Scales," datasheet.
[5] Espressif Systems, "ESP32 Technical Documentation." Available: https://docs.espressif.com/
[6] Python Software Foundation, "Python Documentation."

(This work is licensed under a Creative Commons Attribution 4.0 International License.)