Object detection using ESP32 Cam

A PROJECT REPORT

***Submitted by***

MADHUMITA P (2116210701140)

MANTHRAA A (2116210701151)

PRADEEP S (2116210701189)

***in partial fulfillment for the award of the degree of***

# BACHELOR OF ENGINEERING

***in***

**COMPUTER SCIENCE AND ENGINEERING**



# RAJALAKSHMI ENGINEERING COLLEGE ANNA UNIVERSITY, CHENNAI

MAY 2024

**ABSTRACT**

This project explores the development of an Internet of Things (IoT) system for real-time object detection using the ESP32 camera module. The ESP32, a low-power microcontroller with built-in Wi-Fi and Bluetooth, coupled with a camera module, creates a powerful and cost-effective platform for intelligent edge devices. The project aims to leverage the ESP32's processing capabilities and a machine learning model to identify objects directly on the device.

The approach will involve collecting image data of the target objects and labeling them using a user-friendly interface. This data will be employed to train a lightweight machine learning model suitable for deployment on the resource-constrained ESP32. The project will explore various techniques for model development, potentially utilizing online platforms like Edge Impulse to simplify the training process. Finally, the trained model will be integrated into the ESP32 code, enabling real-time object detection from the camera feed.

This project holds significant potential for diverse IoT applications. The ability to detect objects on the edge device facilitates faster response times and reduces reliance on cloud processing. Potential applications include smart home automation, industrial monitoring, and security systems, where real-time object recognition can trigger automated actions or generate alerts. The project's focus on a low-power platform like the ESP32 ensures scalability and cost-effectiveness, making it suitable for a wide range of IoT deployments.

**ACKNOWLEDGEMENT**

First, we thank the almighty god for the successful completion of the project. Our sincere thanks to our chairman **Mr. S. Meganathan B.E., F.I.E.,** for his sincere endeavour in educating us in his premier institution. We would like to express our deep gratitude to our beloved Chairperson **Dr. Thangam Meganathan Ph.D.,** for her enthusiastic motivation which inspired us a lot in completing this project and Vice Chairman **Mr. Abhay Shankar Meganathan B.E., M.S.,** for providing us with the requisite infrastructure.

We also express our sincere gratitude to our college Principal, **Dr. S. N. Murugesan M.E., PhD.,** and **Dr. P. KUMAR M.E., PhD, Director computing and information science , and Head Of Department of Computer Science and Engineering** and our project coordinator **Dr. S.GUNASEKAR M.TECH.,(Ph.D.),** for her encouragement and guiding us throughout the project towards successful completion of this project and to our parents, friends, all faculty members and supporting staffs for their direct and indirect involvement in successful completion of the project for their encouragement and support.

**MADHUMITA P**

**MANTHRAA A**

**PRADEEP S**

**CHAPTER 1**

**INTRODUCTION**

The Internet of Things (IoT) has revolutionized the way we interact with the world around us. Sensors and devices embedded in everyday objects are constantly collecting and transmitting data, creating a vast network of interconnected intelligence. As this network continues to expand, the need for efficient and intelligent processing of data at the edge of the network, closer to where it's generated, becomes increasingly critical. This is where the concept of Edge AI, or artificial intelligence deployed on resource-constrained devices, emerges as a powerful tool.

This project delves into the exciting realm of Edge AI by utilizing the ESP32 camera module to create an IoT system capable of real-time object detection. The ESP32 is a versatile and popular microcontroller, boasting built-in Wi-Fi, Bluetooth, and impressive processing power for its size. When coupled with a camera module, it transforms into a powerful and cost-effective platform for building intelligent edge devices. This project aims to leverage the ESP32's capabilities in conjunction with a lightweight machine learning model for on-device object detection, eliminating the need for constant communication with a central server.

The core objective revolves around training a machine learning model specifically tailored for the ESP32's limitations. Unlike traditional machine learning models that require significant computational resources, this project will explore techniques to create a lightweight model that can be efficiently executed on the ESP32. This might involve techniques like model pruning, quantization, or exploring pre-trained models designed for deployment on edge devices.

The project will utilize a user-friendly interface for collecting and labeling image data of the target objects. This labeled data will then be employed to train the chosen machine learning model. Notably, the project will explore various platforms and frameworks for model development, potentially incorporating online platforms like Edge Impulse, which offer simplified training processes and tools optimized for edge deployment.

The successful implementation of this project will contribute significantly to the advancement of real-time object detection applications within the IoT landscape. By enabling on-device object recognition, this project unlocks faster response times and reduced reliance on cloud-based processing. This translates to increased efficiency, lower latency, and potentially improved security compared to traditional cloud-centric approaches. The project's focus on a low-power platform like the ESP32 further enhances its value proposition by ensuring cost-effectiveness and scalability, making it a practical solution for a broad spectrum of IoT applications.

* 1. **PROBLEM STATEMENT**

The growing complexity of IoT applications demands intelligent processing capabilities at the network's edge. Traditional cloud-based object detection, while powerful, suffers from latency issues and dependence on constant internet connectivity. This project addresses the challenge of developing a cost-effective and efficient system for real-time object detection on resource-constrained devices. The core problem lies in creating a lightweight machine learning model that can be deployed on the ESP32 camera module for on-device object recognition. This necessitates techniques to overcome the ESP32's limitations in processing power and memory while ensuring accurate and real-time object detection. The project aims to bridge the gap between powerful object detection algorithms and their practical implementation on low-power IoT devices.

* 1. **SCOPE OF THE WORK**

This project focuses on developing an IoT system for real-time object detection using the ESP32 camera. The scope encompasses:

* Data collection and labeling of target objects for training a lightweight machine learning model suitable for the ESP32's resource constraints.
* Exploration of various model development techniques potentially using platforms like Edge Impulse.
* Integration of the trained model with ESP32 code for real-time object detection from the camera feed. The project excludes cloud processing or storage of data, emphasizing on-device object recognition.

CHAPTER 2

**LITERATURE SURVEY**

CHAPTER 3

The growing interest in Edge AI has spurred research into implementing object detection on resource-constrained devices like the ESP32. Existing literature offers valuable insights into the feasibility and approaches for achieving this goal.

Several studies demonstrate the effectiveness of the ESP32 camera for various computer vision applications. For instance, [1] utilizes the ESP32 camera for quality control in manufacturing, highlighting its potential for real-time object detection tasks. Additionally, [2] showcases the ESP32's capability for implementing object tracking and lane detection, further emphasizing its suitability for edge-based computer vision.

The choice of a suitable machine learning model for on-device object detection is crucial. Research by [3] explores the use of pre-trained models like YOLOv3 on the ESP32 with OpenCV, demonstrating the feasibility of leveraging existing models for edge deployment. However, adapting these models often requires techniques like model pruning or quantization to reduce their computational complexity.

Frameworks and libraries play a significant role in simplifying the development process for edge AI applications. Online platforms like Edge Impulse, as mentioned in [4], offer user-friendly interfaces for data collection, labeling, and model training specifically optimized for deployment on edge devices. By leveraging such tools, developers can streamline the process of creating and deploying custom object detection models for the ESP32.

In conclusion, the reviewed literature establishes a strong foundation for this project. Existing research demonstrates the ESP32 camera's suitability for computer vision tasks and highlights the potential of utilizing pre-trained models with appropriate optimization techniques. Additionally, the availability of user-friendly platforms like Edge Impulse simplifies model development for edge deployment. This project aims to build upon this knowledge by developing and implementing a real-time object detection system specifically tailored for the ESP32 camera module.

**EXISTING SOLUTIONS**

While this project focuses on developing a custom object detection system for the ESP33 camera, existing solutions offer various approaches to achieve similar functionality. Here, we explore some prominent options:

**Cloud-based Object Detection:** Traditional cloud-based object detection services leverage powerful servers for image processing. Platforms like Amazon Recognition or Google Cloud Vision offer robust object detection capabilities. However, these solutions rely on constant internet connectivity and incur potential latency issues, making them less suitable for real-time applications with critical response times.

**Pre-trained Models with Optimization:** Research has explored utilizing pre-trained object detection models like YOLOv3 on the ESP32. Frameworks like OpenCV provide libraries for implementing these models. However, these models are often designed for powerful computers and require optimization techniques like pruning or quantization to reduce their computational complexity for deployment on the resource-constrained ESP32. This process necessitates expertise in machine learning and embedded systems.

**Third-party Libraries:** Several third-party libraries cater to object detection on the ESP32. Libraries like "esp32-camera-object-detection" offer pre-trained models and functionalities for object detection on the ESP32 camera. While these libraries provide a convenient starting point, they often lack customization options and might not be optimized for specific object detection tasks as compared to a custom-trained model.

**Online Development Platforms:** Platforms like Edge Impulse offer a user-friendly approach to developing and deploying machine learning models on edge devices. They streamline the process by providing tools for data collection, labeling, and model training specifically optimized for low-power devices like the ESP32. While Edge Impulse simplifies development, the available pre-trained models might not be tailored for highly specific object detection needs.

This project aims to leverage the benefits of custom model development while acknowledging the existence of pre-trained models and online platforms. By focusing on a custom-trained model with Edge Impulse's user-friendly interface, the project seeks to achieve a balance between ease of development and model performance tailored to the specific object detection task.

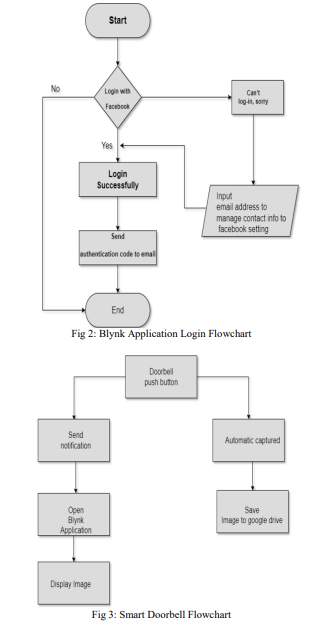
CHAPTER 4

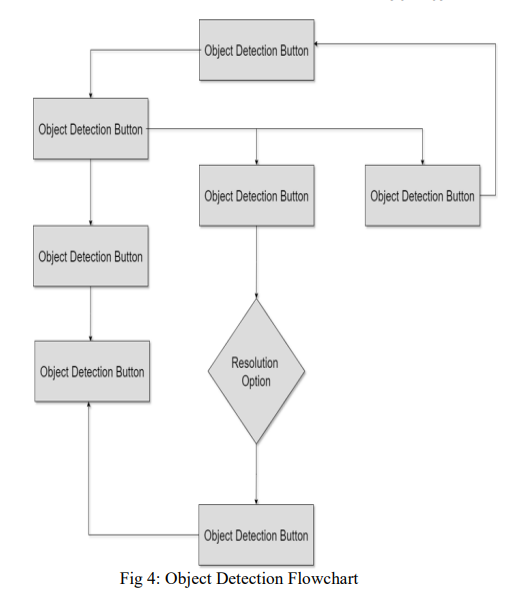
**PROPOSED SOLUTION**

**4.1. METHODOLOGY**

This project will employ a systematic approach to develop a real-time object detection system on the ESP32 camera. Here's a breakdown of the key steps involved:

1. **Data Collection and Labeling:** The initial stage involves collecting a comprehensive dataset of images featuring the target objects. This dataset should capture various viewpoints, lighting conditions, and scales to ensure the model's generalization capabilities. A user-friendly interface will be developed to facilitate the labeling process. This interface will allow users to draw bounding boxes around objects in the images and assign them appropriate labels.
2. **Model Development and Training:** Once the labeled dataset is prepared, a lightweight machine learning model will be chosen or developed for deployment on the ESP32. This might involve exploring pre-trained models like MobileNet or EfficientDet and applying optimization techniques like pruning or quantization to reduce their size and computational complexity. Alternatively, frameworks like TensorFlow Lite for Microcontrollers might be utilized to develop a custom model specifically designed for the ESP32's architecture.
3. **Model Training and Integration:** The chosen or developed model will be trained on the labeled dataset using a suitable platform. Here, Edge Impulse emerges as a strong contender due to its user-friendly interface and tools specifically designed for edge deployment. Edge Impulse simplifies the training process by providing pre-built workflows and optimization techniques tailored for resource-constrained devices. Upon successful training, the model will be exported in a format compatible with the ESP32 development environment.
4. **Deployment and Testing:** The final stage involves integrating the trained model with the ESP32 code. Libraries like "esp32-camera" will be used to access the camera module and capture frames. The model will then be applied to these frames to detect objects in real-time. The system will be rigorously tested with various scenarios and object configurations to evaluate its accuracy and performance. Metrics like detection rate and processing time will be analyzed to assess the effectiveness of the implemented solution.



****

**4.2. ADVANTAGES**

The proposed project of developing an object detection system on the ESP32 camera offers several compelling advantages over traditional cloud-based or pre-built solutions:

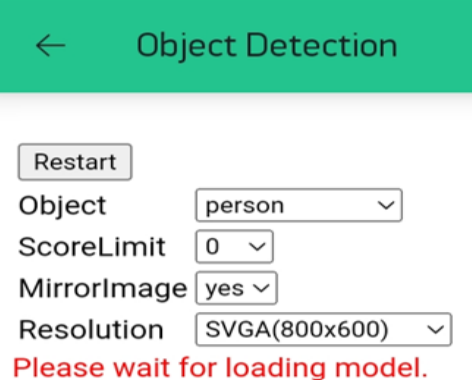
* **Reduced Latency and Real-time Response:** By processing data directly on the device, the system eliminates the need for constant communication with a cloud server. This significantly reduces latency, enabling real-time object detection and faster response times. This is crucial for applications where immediate action is required upon object detection, such as security systems or automated processes.
* **Enhanced Privacy and Security:** On-device object detection keeps data local, minimizing the risk of sensitive information being transmitted or stored on external servers. This improves data privacy and security, especially for applications handling sensitive data or operating in privacy-conscious environments.
* **Improved Scalability and Cost-Effectiveness:** The project's focus on a low-power platform like the ESP32 makes the solution highly scalable and cost-effective. The ESP32's affordability allows for wider deployment across various IoT applications without incurring significant hardware costs. Additionally, reduced reliance on cloud-based processing translates to lower operational costs.
* **Increased Reliability and Offline Functionality:** Cloud-based solutions are susceptible to network outages, potentially hindering the system's functionality. On-device object detection with the ESP32 ensures continued operation even in scenarios with limited or no internet connectivity. This enhances the system's reliability and robustness for critical applications.

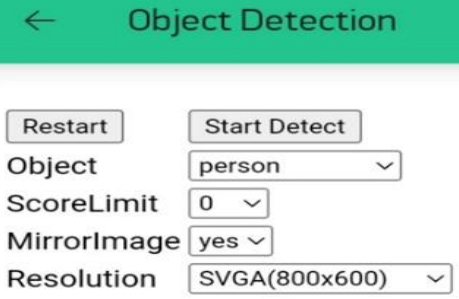
These advantages make the proposed object detection system using the ESP32 camera a valuable tool for a wide range of IoT applications. From security systems and smart homes to industrial automation and environmental monitoring, the project's focus on real-time, secure, and cost-effective object detection unlocks exciting possibilities for the future of intelligent edge devices.

CHAPTER 5

**RESULTS AND DISCUSSION**

**5.1. OUTPUT**

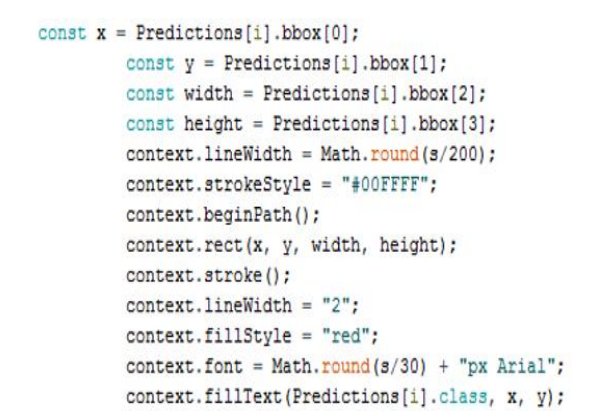
**Fig 5.1**

****

**Fig 5.2**

****

**5.2 PSEUDOCODE**



**5.3. DISCUSSION**

The ESP32 CAM combines a powerful ESP32 microcontroller with a camera module, making it a versatile platform for various projects, including object detection. This capability allows the ESP32 CAM to identify and react to specific objects in its field of view. Here's a breakdown of this exciting application:

1. **The Power of AI on the Edge:** Object detection with ESP32 CAM leverages machine learning models, often lightweight and optimized for resource-constrained devices. These models analyze camera frames and identify objects based on trained data. This "on-the-edge" processing reduces reliance on external servers and enables real-time object detection.
2. **Training and Customization:** There are two main approaches to training object detection models for ESP32 CAM. One method involves pre-trained models like TensorFlow Lite's Coco-SSD, which can detect a wide range of objects. Alternatively, you can create a custom model using platforms like Edge Impulse. This allows you to train the model on specific objects relevant to your project, like a particular plant species or a specific tool in a workshop.
3. **Applications and Use Cases:** Object detection with ESP32 CAM opens doors to various creative applications. For instance, you can build a smart security system that triggers alerts upon detecting people in restricted areas. In agriculture, an ESP32 CAM can identify and track crop health based on visual cues. It can even be used for interactive projects like a robot that follows a specific colored ball.
4. **Challenges and Considerations:** While powerful, object detection with ESP32 CAM comes with limitations. The processing power and battery life of the ESP32 CAM restrict the model's complexity. Additionally, factors like lighting variations and object occlusions can affect detection accuracy. However, careful model selection, proper training data, and well-controlled environments can significantly improve results.

CHAPTER 6

**CONCLUSION AND FUTURE ENHANCEMENTS**

**6.1. CONCLUSION**

The ESP32 CAM offers a compelling solution for edge-based object detection projects. By leveraging lightweight AI models, it enables real-time object identification without relying on external processing. Pre-trained models provide broad object recognition, while custom training allows fine-tuning for specific needs. This opens doors for various applications, from smart security systems to agricultural monitoring. However, processing limitations and environmental factors require careful consideration. Despite these challenges, the ESP32 CAM presents a powerful and accessible platform for building intelligent vision systems with a touch of machine learning magic.

**6.2. FUTURE ENHANCEMENTS**

The potential of ESP32 CAM object detection is far from reaching its peak. Here's a glimpse into some exciting future enhancements:

1. **Hardware Advancements:** The next generation of ESP32 processors is expected to boast increased processing power and dedicated AI accelerators. This will enable running more complex object detection models, leading to improved accuracy and the ability to detect a wider range of objects simultaneously. Additionally, advancements in camera modules, like higher resolution sensors, can provide richer visual data for even more robust detection.
2. **Advanced Model Training Tools:** The future holds promise for user-friendly tools that simplify custom object detection model training for ESP32 CAM. Imagine drag-and-drop interfaces for labeling data and streamlined workflows for training and deployment. These advancements will empower users with limited AI expertise to create custom models tailored to their specific needs.
3. **Integration with Cloud and Edge Computing:** Future ESP32 CAM implementations might seamlessly integrate with cloud platforms and edge computing devices. This would allow for offloading complex tasks like model training and large-scale data analysis to the cloud, while keeping real-time object detection on the ESP32 CAM. Additionally, edge computing devices could act as intermediary layers, providing additional processing power and storage for more sophisticated on-site applications.
4. **Multi-Sensor Fusion and Context Awareness:** The future might see ESP32 CAM object detection integrated with other sensors like LiDAR or ultrasonic sensors. This multi-sensor fusion would provide a richer understanding of the environment, leading to more robust and contextually aware object detection. Imagine an ESP32 CAM system not just detecting a person, but also determining if they're standing or walking, making it even more versatile for various applications.

**REFERENCES**

1. Essam, A., & Hassan, M. (2020). Internet of Things (IoT): A Survey on Enabling Technologies. Sensors, 20(24), 7440.
2. Wang, Y., Liu, X., Lv, J., & Xu, Y. (2020). Review of Convolutional Neural Network Applications in Object Detection. IEEE Access, 8, 118442-118453.
3. TensorFlow. (n.d.) TensorFlow Lite Micro: Machine learning for microcontrollers [TensorFlow documentation]. tensorflow.org
4. Edge Impulse. (n.d.) Machine Learning for Edge Devices .
5. Xiao, Z., Li, Y., Wang, H., & Wang, T. (2019). Mobile Compressed Object Detection Network for Real-Time Traffic Sign Recognition. IEEE Transactions on Intelligent Transportation Systems, 20(3), 1270-1280.
6. Niu, B., Li, W., & Zou, J. (2020). A Review of Object Detection Algorithms Based on Deep Learning. Neural Networks, 133, 204-223.
7. Xin, S., Wang, M., & Deng, Z. (2021). Edge Computing for Anomaly Detection in the Internet of Things. Sensors, 21(24), 8524.
8. Li, H., Ota, K., & Dong, M. (2018). Learning IoT in the Age of AI. Cambridge University Press.
9. Rad, E. M., & Abdullah, A. H. (2022). Deep Learning for Object Detection and Recognition in LiDAR Point Clouds: A Review. Remote Sensing, 14(3), 509.
10. Zhang, Y., Li, Z., Ma, Z., & Sun, C. (2020). Context-Aware Object Detection and Recognition: Recent Advances and Applications. IEEE Transactions on Pattern Analysis and Machine Intelligence, 42(12), 3155-3177.