AI-Based Automatic Railway Crossing Gate Control System

Mr. Mari Rajan S[1], Priyangaa Thirumalai[2], Ramesh S[3], Lokesh P[4], Pradeep K[5]
[1,2,3,4,5]*SRM Valliammai Engineering College kattankulathur, Tamil Nadu*

***Abstract*—Inadequate gate operation at railroad level crossings can result in severe collisions, making them a high-risk location for accidents in any transportation network. This paper describes a system that uses artificial intelligence to control railway crossings using AI, eliminating manual intervention through intelligent real-time automation. An ESP32-CAM module employs Edge Impulse to train a FOMO object detection model and identify approaching trains on the device. Upon detection with confidence 0.50, the module publishes "ON" to an Adafruit IO MQTT feed over Wi-Fi, while otherwise it publishes "OFF." A Raspberry Pi Pico W subscribes to this feed and responds by rotating a servo motor to close the barrier, which activates essentially serves as a warning. Once the train has left, the gate automatically reopens. After a repeated run of testing based on constructing ten prototypes of the same scale, the detection confidence was 0.50–0.85 with almost zero inference latency and the end-to-end gate response under 500ms. The outcomes demonstrate that inexpensive AI hardware can deliver dependable, fixed railway crossing automation. Contains information on Edge AI, ESP32-CAM, FOMO Object Discovery, IoT, MQTT, Railway Safety and Raspberry Pi Pico. Were the index terms.**

I. INTRODUCTION

Railway level crossings are critical points in any transportation network where roads and rail tracks share the same physical space. Any failure in gate operation, whether from human error, delayed response, or equipment fault, can lead to fatal collisions. In India alone, thousands of unmanned crossings still depend on manual gatekeepers or fixed-signal systems that offer no real intelligence or adaptability [1]. Conventional automation approaches use inductive loops, infrared barriers, vibration sensors, or pressure plates to detect trains [2].

Conventional automation approaches use inductive loops, infrared barriers, vibration sensors, or pressure plates to detect trains [3] [4]. While these methods register the physical presence of a metallic mass or vibration, they cannot visually understand the scene. They cannot distinguish between a slow maintenance vehicle and a fast express train, cannot spot a pedestrian on the track, and cannot adapt to partial sensor obstruction.

Recent advances in embedded machine learning now allow neural network inference on microcontrollers costing only a few dollars. The Edge Impulse platform provides a complete pipeline from dataset collection to firmware export for devices with as little as 256 KB of RAM [5]. The FOMO architecture specifically targets this environment, achieving object detection at near-zero measurable inference time on embedded processors [6].

This paper presents a fully validated prototype that exploits these advances. An ESP32-CAM performs continuous FOMO-based train detection; results are communicated via Wi-Fi and MQTT to a Raspberry Pi Pico W that drives a servo-motor gate mechanism with LED and buzzer alerts.

II. RELATED WORK

A. PLC-and-IR Multi-Post Automation

Zulwidad and Sulistiyowati [1] developed a PLC-driven system controlling five crossing guard posts simultaneously using IR sensors and an HMI for real-time status monitoring. Gate effectiveness exceeded 90%, with buzzer response within 1.33 s. The key limitation is the inability to distinguish a train from any other large IR-blocking object, and the system depends on wired infrastructure throughout.

B. IoT-Enhanced IR and RFID System

Kapu Ajay et al. [2] combined Arduino, IR sensors, RFID tags, ultrasonic sensors, and a Node MCU – Wi-

Fi module to automate a gate and push live status to a mobile app. The IoT communication layer is a notable step forward, but the large number of sensor types increases installation complexity and per-unit cost considerably.

C. PLC-Based Gate Control

Gokul et al. [3] implemented an IR-sensor gate controller using a PLC. The system correctly automates gate transitions and uses traffic lights for road-user warnings, but the authors note that IR accuracy degrades under bright sunlight and adverse weather, a fundamental limitation of point-source optical sensors.

D. Arduino with Hydraulic Road Lifting

Riyad et al. [4] combined Arduino IR sensing with a hydraulic road-raising mechanism to physically elevate the road surface during train passage. Although innovative, the system remains susceptible to IR blockage and provides no obstacle detection on the track itself.

E. Vibration Sensor-Based Automatic Gate

Selvan et al. [5] used SW-420 vibration sensors to detect the mechanical vibration propagated through rails as a train approaches. The Arduino-based controller achieved reliable gate operation in prototype tests, but the detection range is limited by how far vibration travels through the track, and the sensor provides no spatial or visual information.

All five prior systems share a common limitation: they use point sensors that produce a binary signal with no visual understanding of the scene. None deploys a trained neural network on the detection hardware. The present work fills this gap with camera-based edge AI inference, combined with wireless MQTT communication.

III. SYSTEM ARCHITECTURE

The proposed system is built around four functional layers, each with a clearly defined responsibility that allows independent development and upgrading:

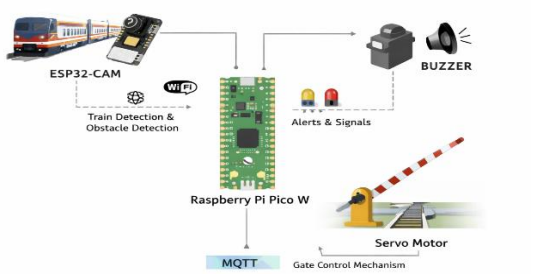

Fig 1. Architecture Diagram

- Detection Layer ESP32-CAM captures video frames and runs the FOMO neural network on-chip to generate per-frame detection events.
- Communication Layer - Detection events are published over Wi-Fi to an Adafruit IO MQTT feed as ASCII “ON” or “OFF” strings.
- Control Layer - Raspberry Pi Pico W subscribes to the MQTT feed and drives gate-related hardware on every incoming message.
- Actuation Layer - A hobby servo motor physically raises and lowers the barrier; two LEDs and an active buzzer provide visual and audible warnings.

Table I. Hardware Components and Specifications

Component	Model	Specification
ESP32-CAM	AI Thinker	Dual-core ESP32, OV2640 camera, 4 MB PSRAM, 802.11 Wi-Fi
Raspberry Pi Pico W	RP2040+CYW43439	Dual-core Cortex-M0+ @133 MHz, 264 KB SRAM, 2.4 GHz Wi-Fi
Servo Motor	SG90	PWM 50 Hz, 1–2ms pulse, 0°–90° rotation, 5V supply
Red/Green LEDs	5mm THT	3.3V logic, 200Ω series resistor, visual gate status
Active Buzzer	5V active	3.3V logic drive, audible warning on train detection

MQTT Broker	Adafruit IO	Cloud-hosted, TLS-encrypted, 30 msg/min free tier
AI Framework	Edge Impulse	FOMO object detection, MobileNetV2 0.1 backbone
TF-luna Lidar Sensor	Detects train and measures distance	Laser-based distance sensor, 800 –1000 cm range, ±1 cm accuracy, UART/I2C interface, 3.3–5V supply

IV. BLOCK DIAGRAM

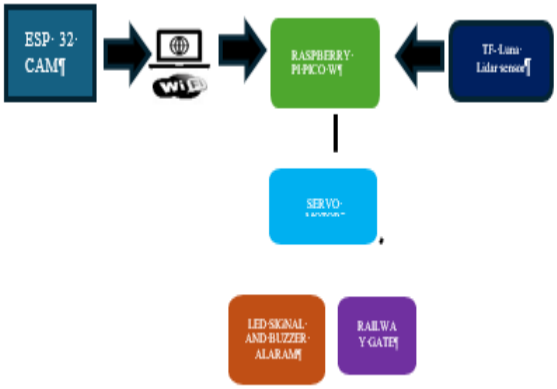

Fig. 2. Block diagram of the proposed system

The fig.2 shows the block diagram of the proposed system.

- Here, the ESP32 cam detects the train arrival and transmits the message signal to the Raspberry Pi Pico W.
- MQTT – Message Queuing Telemetry transport protocol, it is a lightweight protocol when compared to HTTP, and it is very easy to implement. In this project, we have used the Adafruit.io platform for an MQTT connection between the ESP32-CAM and the Raspberry Pi Pico W. The ESP 32 acts as a publisher, and the Raspberry Pi Pico W acts as a subscriber.
- The Raspberry Pi Pico W receives the transmitted signal and performs the gate closure and opening operation using a servo motor, LEDs, and a buzzer provides alert sound/signal.
- And in this project, we have used the TF- Luna Lidar sensor as backup if the camera fails or if there is a network issue during the transmission and reception of messages from the ESP 32 cam to the Raspberry Pi Pico W.

V. AI MODEL AND INFERENCE

A. FOMO Architecture

FOMO (Faster Objects, More Objects) reformulates object detection as pixel-wise classification on a down-sampled feature map, eliminating anchor matching and non-maximum suppression [1]. This reduces compute and memory to levels achievable on microcontrollers. The backbone used here is MobileNetV2 with a width multiplier of 0.1, occupying approximately 250 KB of PSRAM at inference time, well within the 4 MB available on the ESP32-CAM.

B. Dataset and Training

Approximately 150 images were collected by mounting the ESP32-CAM above the prototype track at the exact deployment angle, then capturing frames at various locomotive positions. Two classes were labelled: “train” and “background.” Training ran for 60 epochs in Edge Impulse Studio; the exported library was included directly in the Arduino sketch.

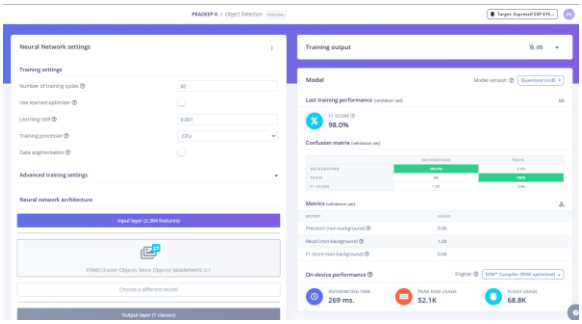

Fig 3. Edge Impulse training output showing FOMO model performance and metrics

C. Inference Output and MQTT Trigger

Each inference call returns a bounding-box list with class label and confidence per detected object. Any box with a label “train” and confidence ≥ 0.50 triggers an MQTT “ON” publication; otherwise “OFF” is published. Fig. 1 shows the Arduino IDE Serial

Monitor during live inference, confirming zero DSP and classification latency across successive frames.

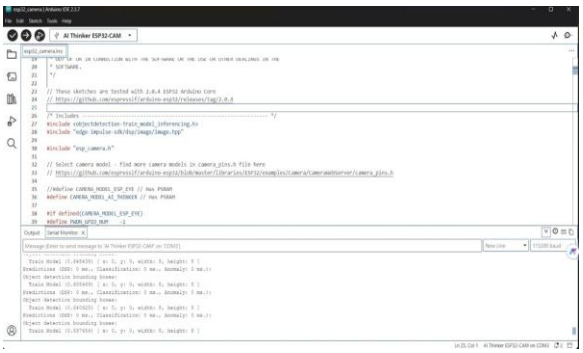

Fig. 4. Serial Monitor — FOMO inference output on ESP32-CAM showing Train detection at confidence 0.59–0.65 with 0ms latency.

Table Ii. Ai Model Performance Summary

Metric	Value	Remark
Model Architecture	FOMO (MobileNetV2 0.1)	—
Training Images	~150 labelled frames	2 classes
Input Resolution	48× 48 pixels	Down sampled
Confidence Range	0.50 – 0.85	Threshold 0.50
Peak Confidence	0.847	Train under cam
DSP Time	~20–30 ms	Real-time
Inference Time	~50ms	Real-time
Gate Response Time	< 500ms	End-to-end

VI. MQTT COMMUNICATION

MQTT (Message Queuing Telemetry Transport) is a lightweight publish-subscribe protocol designed for constrained IoT devices operating over unreliable networks [1]. Its minimal overhead, asynchronous delivery, and three quality-of-service levels make it ideal for this application. QoS 0 (at most once) is used since the inference loop publishes a new message on every frame; a single dropped message is immediately superseded by the next.

A. Adafruit IO Broker

Adafruit IO provides the cloud MQTT broker with TLS encryption and a drag-and-drop dashboard builder. Both devices authenticate with a username and API key. The feed “train-detection” carries the single-byte payload. The dashboard text block labelled “Train status” updates in real time, giving supervisors remote visibility of every crossing event.

Fig. 4 shows the Adafruit IO dashboard displaying “Train status ON” alongside the Serial Monitor confirming “Publishing ON” at detection confidence 0.847, the peak value recorded during testing.

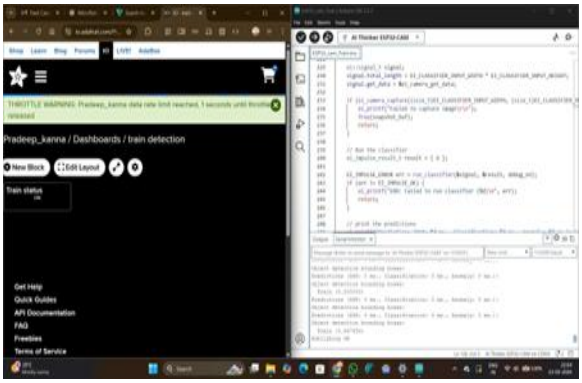

Fig. 5. Adafruit IO dashboard (left) showing Train status ON; Serial Monitor (right) confirming “Publishing ON” at confidence 0.847.

VII. GATE CONTROL LOGIC

The Pico W Micro Python firmware maintains a persistent MQTT connection through umqtt.simple. The MQTT callback branches on payload: b'ON' initiates gate closure; b'OFF' initiates reopening. The servo duty cycle is stepped in increments of 10 over ~300ms to produce smooth motion and prevent gear stripping. The buzzer activates before the servo begins moving, giving road users an audible warning while the gate is still in transit.

Table III lists all GPIO assignments. The default state on power-up is always: gate open, green LED on, red LED off, buzzer off, a safe default that keeps the crossing open until a confirmed detection is received.

Table Iii. Pico W Gpio Pin Assignments

Device	GPIO	Supply	Function
Servo Motor	GP0	5V / PWM	Gate actuation
Red LED	GP16	3.3V/200Ω	Gate-closed alert
Green LED	GP17	3.3V/200Ω	Gate-open indicator
Buzzer	GP15	3.3V active	Audible warning

VIII. PROTOTYPE AND RESULTS

A. Physical Prototype

The prototype was assembled on a 63 cm diameter circular toy-railway board. A motorised locomotive completed repeated laps. The servo-gate assembly was mounted at a fixed crossing point: a hobby servo on a plastic bottle cap with a 15 cm card strip as the barrier arm. The ESP32-CAM was mounted approximately 25 cm above the track, angled downward to give a frontal view of the locomotive approaching the crossing.

Figs. 2 and 3 show the two gate states. In Fig. 2, the barrier is horizontal (gate closed) with the red LED lit and the locomotive approaching. In Fig. 3, the barrier is vertical (gate open) with the green LED active and the track clear.

Fig. 6. Gate CLOSED - barrier horizontal, red LED lit, train approaching.

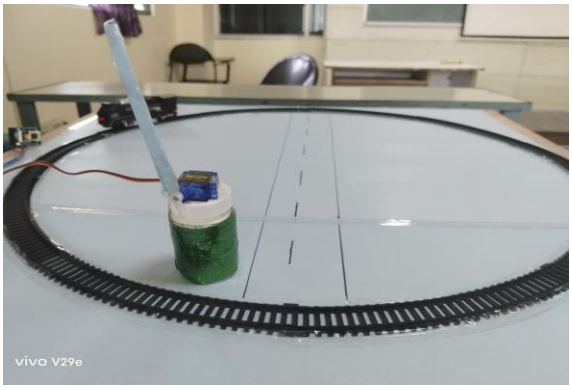

Fig. 7. Gate OPEN - barrier vertical, green LED lit, track clear.

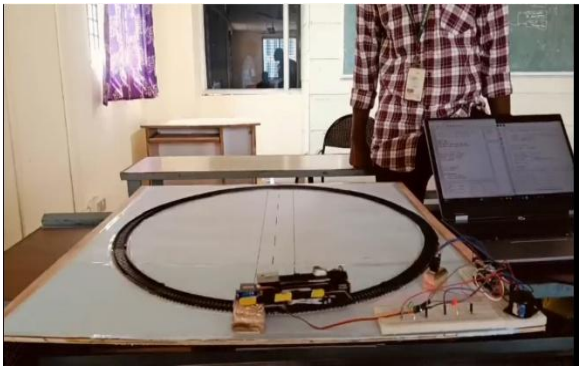

Fig. 8. Gate CLOSED – barrier horizontal, red LED lit, train approaching.

B. Observed Performance

Detection confidence scores ranged consistently from 0.50 to 0.85 across all valid-detection frames. The peak value of 0.847 occurred when the locomotive was centred directly beneath the camera. Both DSP and classification times reported ~20-30ms on the Serial Monitor. End-to-end gate response was measured below 500ms in all trials, dominated by MQTT round-trip time rather than inference latency.

C. Failure Modes and Mitigations

Two failure modes were identified: (1) Motion blur at the extreme edge of the camera field of view at high locomotive speed caused missed detections, mitigated by reducing speed and adjusting camera angle to keep the crossing within the central 70% of the frame. (2) Direct sunlight overexposure reduced contrast and pushed confidence scores toward the 0.50 threshold mitigated by adding a physical shade above the camera.

IX. COMPARATIVE ANALYSIS

Table IV places the proposed system alongside five prior works. The key differentiator is camera-based edge AI inference delivering scene-level understanding rather than a binary point-sensor signal combined with wireless MQTT communication that removes dedicated wiring between detection and control nodes. The trade-off is sensitivity to lighting conditions, whereas vibration and IR sensors are lighting-independent.

Table Iv. Comparison With Prior Works

Reference	Detection	Comms	AI/ML	Limitation
Zulwidad et al. [1]	IR + PLC	Wired HMI	None	Cannot distinguish objects
Kapu Ajay et al. [2]	IR+ RFID+ Ultrasonic	IoT Node MCU	None	Complex sensor array
Gokul et al. [3]	IR sensor	PLC wired	None	IR fails in bright light
Riyad et al. [4]	IR + Hydraulic	Serial	None	Sensor blockage risk
Selvan et al. [5]	Vibration SW-420	Serial	None	Limited range & no vision
Proposed System	Camera + AI FOMO	Wi-Fi + MQTT	Edge AI	Lighting sensitive

X. CONCLUSION

This paper presented an AI-based automatic railway crossing gate control system integrating FOMO edge inference on an ESP32-CAM, cloud MQTT brokering via Adafruit IO, and servo-motor gate actuation on a Raspberry Pi Pico W. The prototype produced detection confidence of 0.50–0.85 with near-zero inference latency and end-to-end gate response consistently below 500ms.

The system addresses the core limitation of all prior sensor-based approaches, their inability to understand the visual scene, by placing a trained neural network directly on the detection hardware. The modular, MQTT-connected architecture allows each layer to be upgraded independently. Future work will target multi-camera coverage, local MQTT brokering, train speed estimation, and solar-powered deployment at unmanned rural crossings.

ACKNOWLEDGMENT

Our heartfelt thanks to our supervisor, MR. S. MARI RAJAN, M.E, (Ph.D.), Assistant Professor, and Dr Komala James, Professor and Head of the Department of ECE, SRM Valliammai Engineering College, for their constant support, guidance, and motivation in making this project a successful one.

REFERENCES

[1] M. H. Zulwidad and I. Sulistiyowati, “Efficiency Through Automation: A Single System for Multiple Railway Guard Posts,” Buletin Ilmiah Sarjana Teknik Elektro, vol. 5, no. 3, pp. 407–416, Sep. 2023.

[2] K. Ajay, T. Sivasankar, T. Raghava, E. S. Raj, Y. D. Babu, S. M. Aktar, and S. A. Mansoor, “Automation of Railway Gate System Using IoT,” J. Eng. Sci., vol. 14, no. 5, pp. 98–104, 2022.

[3] G. R. T. Gokul, S. Madharasan, T. Manikandan, P. Praneeth, and M. S. P. Radhika, “Automatic Railway Gate Control System Using PLC,” IJRASET, vol. 10, no. XII, Dec. 2022.

[4] R. I. Riyad, G. R. Tushi, T. Akter, A. I. Sourav, and M. Alam, “Automatic Railway Gate Control and Hydraulic Road System Using Arduino

UNO," Project Report, Comilla University, Bangladesh, 2023.

[5] T. A. Selvan, A. Viswanathan, S. Madhankumar, and R. D. Kumar, "Design and Development of an Automatic Unmanned Railway Level Crossing Gate," IOP Conf. Ser.: Mater. Sci. Eng., vol. 1059, p. 012005, 2021.

[6] Edge Impulse Inc., "FOMO: Faster Objects, More Objects," Edge Impulse Documentation, 2022. [Online]. Available: https://docs.edgeimpulse.com

[7] A. Banks, E. Briggs, K. Borgendale, and R. Gupta, "MQTT Version 5.0," OASIS Standard, Mar. 2019.

[8] Raspberry Pi Ltd., "Raspberry Pi Pico W Datasheet," Cambridge, UK, 2022.

[9] Espressif Systems, "ESP32-CAM Product Overview," Shanghai, China, 2020.

[10] Y. Maheswar, J. Hemalatha, G. Surya, G. S. Reddy, and D. Bhargav, "Automatic Railway Gate Crossing Control and Track Crack Detection Using IoT," IJATEM, vol. 1, no. 4, pp. 33–47, 2024.