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

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

Federated Learning Based Intrusion Detection System for IoT Devices using Dual ESP32 Nodes

Mahadev Prasad S, Gowtham Naik M, Kyatanagouda V Patil, Shreedharagouda B G
Department of Computer Science and Engineering
Maharaja Institute of Technology, Thandavapura, Mysore, India

***Abstract*—Internet of Things (IoT) devices are being deployed in access-control, building-automation and remote-monitoring systems, yet most of them have very limited processing and memory resources, and sending all raw sensor data to a central server adds communication overhead and privacy concerns. This paper presents a low-cost prototype of a Federated Learning Based Intrusion Detection System (FL-IDS) built on two ESP32 development boards. ESP32 #1 acts as a local Wi-Fi access point, web-dashboard host and federated-learning coordinator. ESP32 #2 acts as the sensing and local-learning node and interfaces with an MFRC522 RFID reader, a PIR motion sensor, status LEDs, a buzzer, an I2C LCD and a servo-operated gate. The two boards exchange events and model parameters over UART. A logistic-regression model with three event features (recent unauthorised RFID activity, motion while security is enabled, and a combined indicator) is trained on the sensor node; its weights and bias are sent to the coordinator, aggregated, returned as global parameters, validated and applied. A browser dashboard provides login, security control, connected-station monitoring and five rule-based suspicious-activity indicators. The prototype demonstrates the complete embedded workflow of sensing, local training, parameter exchange and alerting. Quantitative detection accuracy is not claimed and must be established separately on representative labelled data.**

***Index Terms*—Internet of Things, federated learning, intrusion detection, ESP32, logistic regression, model aggregation, RFID, PIR sensor, UART, web dashboard.**

I. Introduction

The Internet of Things (IoT) connects sensors, controllers and physical devices so that systems can monitor conditions and respond automatically. As the number of connected devices grows, security monitoring becomes important, because unauthorised access or abnormal behaviour can affect the safety and reliability of an installation. An intrusion detec-tion system (IDS) observes selected events or traffic patterns and flags behaviour that may indicate an attack or a policy violation.

Conventional learning-based IDS designs collect records at a central location for training. This needs substantial communication and exposes raw records to additional systems. Federated learning (FL) offers a distributed alternative: each client trains a model locally, shares only model parameters with a coordinator, and receives a combined global model [1]. Raw data movement is reduced, although model updates can still leak information and must be protected [2].

This work combines the FL idea with physical sensors and an embedded dashboard to build an end-to-end IoT security prototype using two ESP32 boards. The main contributions are:

1) A two-node architecture that separates dashboard and FL coordination (ESP32 #1) from sensing, actuation and local learning (ESP32 #2).
2) A lightweight logistic-regression intrusion score computed on the microcontroller from RFID and PIR event features.
3) A UART message protocol for events, scores, local parameters (FLW) and global parameters (FLM), with validation of received parameters.
4) A local Wi-Fi web dashboard with login, security control, connected-station monitoring and five rule-based suspicious-activity indicators.
5) A working hardware prototype with measured firmware resource usage and an honest statement of what remains to be evaluated.

II. Related Work

IDS techniques are usually grouped as signature-based, anomaly-based and rule-based. Signature systems compare observations with known attack patterns, anomaly systems learn normal behaviour and flag deviations, and rule-based monitoring applies explicit conditions such as a threshold on failed logins. Practical systems often combine these. The scale of IoT-specific threats, such as large botnets built from compromised devices, has been documented by Kolias *et al.* [5].

Machine-learning IDS work for IoT has moved towards lightweight models. Kitsune [3] uses an ensemble of small autoencoders for online network intrusion detection, and D¨IoT [4] applies federated self-learning to detect anomalous behaviour of IoT devices without sharing raw traffic. These systems operate on network traffic and typically assume gateway-class hardware.

Edge computing processes data near its source instead of sending every observation to a remote server [6]. Microcontrollers such as the ESP32 can perform feature extraction, threshold checks and compact model inference, but memory and processing limits restrict model size and update frequency. Logistic regression [7] is attractive in this setting because its parameter set is one weight per feature plus a bias, which is easy to train, transmit and inspect.

(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 11 , November - 2026

TABLE I
COMPARISON OF APPROACHES (TYPICAL FEATURES)

Approach	ML model	On-device training	Param. sharing	Local web UI
Rule-based monitoring	No	No	No	Partial
Centralised ML IDS [3]	Yes	No	No	No
Federated IoT IDS [4]	Yes	Partial	Yes	No
Proposed system	Yes	Yes	Yes	Yes

Federated Averaging (FedAvg) [1] combines client models by averaging parameters, optionally weighted by the number of local training samples. Our prototype follows this idea at a very small scale, using a four-element parameter vector exchanged over a serial link, and pairs it with physical sensing and a local operator interface.

III. RESEARCH GAPS

The reviewed work points to several issues relevant to an embedded FL-IDS prototype.

A. Hardware-Level Demonstration

Most FL-IDS studies are evaluated in simulation or on traffic datasets. Few show the complete loop of physical sensing, on-device training, parameter exchange and a visible alert on low-cost microcontrollers.

B. Resource Constraints

Models for IoT nodes must fit within the flash and RAM of a microcontroller. A compact parameter vector makes training, transmission and inspection practical.

C. Separation of Model Score and Alert State

A learned score and a rule-based alert are different things. They should be shown separately so that an alert flag is not mistaken for a calibrated probability.

D. Weak Local Interfaces

Embedded systems often lack an offline operator interface. A local access-point dashboard allows monitoring and control without internet access.

E. Unsupported Performance Claims

Prototype papers often report accuracy without a labelled test set. Performance claims should follow only from measured predictions on held-out data.

Table I compares typical features of earlier approaches with the proposed system.

IV. PROPOSED SYSTEM

The system is organised into the six stages shown in Fig. 1. A phone or laptop joins the Wi-Fi access point created by ESP32 #1 and opens the dashboard in a browser (stage 1). ESP32 #2 reads the RFID and PIR sensors (stage 2), converts recent events into a feature vector (stage 3), and runs the local logistic-regression model (stage 4). Model parameters and events are exchanged with the coordinator over UART (stage 5), and results appear as dashboard values, LCD text, LEDs, buzzer and gate movement (stage 6).

TABLE II
HARDWARE COMPONENTS AND CONNECTIONS

Component	Function	Connection
ESP32 #1	SoftAP, dashboard, aggregation	UART RX 16 / TX 17
ESP32 #2	Sensing, local model	UART RX 16 / TX 17
MFRC522	RFID UID reading	SS 5, RST 4, SCK 18, MISO 19, MOSI 23
PIR sensor	Motion detection	GPIO 27
Green / Red LED	Normal / alert	GPIO 25 / GPIO 26
Buzzer	Audible alert	GPIO 14
Servo motor	Gate demonstration	GPIO 15
I2C LCD 16×2	Status, dashboard IP	SDA 21, SCL 22, 0x27

V. HARDWARE AND SOFTWARE COMPONENTS

The hardware consists of two ESP32 boards and the peripherals in Table II. The block diagram is given in Fig. 2. The UART link is point-to-point: TX of one board goes to RX of the other (GPIO 17→GPIO 16 in both directions), the grounds are common, and the baud rate is 115200. A servo can draw current peaks beyond the board regulator, so a suitably rated supply with common ground is recommended, and 5 V signals must not be applied directly to ESP32 GPIO pins.

Firmware is written in the Arduino IDE using the ESP32 Arduino core and the MFRC522, LiquidCrystal_I2C and ESP32Servo libraries. The dashboard is served over the coordinator's SoftAP network and works without internet access.

VI. METHODOLOGY

A. Event Feature Extraction

ESP32 #2 identifies RFID cards by reading the UID, formatting it as a colon-separated hexadecimal string, and comparing it with a configured authorised-UID list. An authorised card toggles the security state; an unrecognised card records an unauthorised event. The PIR input is processed on its rising edge so that a continuously high signal is not counted as repeated events. Three features are derived over a 60 s recency window:

$$x_1 = 1 \text{ if unauthorised RFID within last 60 s, else } 0, \quad (1)$$

$$x_2 = 1 \text{ if motion while armed within last 60 s, else } 0, \quad (2)$$

$$x_3 = (x_1 + x_2)/2. \quad (3)$$

Once an event is older than the window its feature returns to zero, so this is a simple recency representation rather than a time-series model.

(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 11 , November - 2026

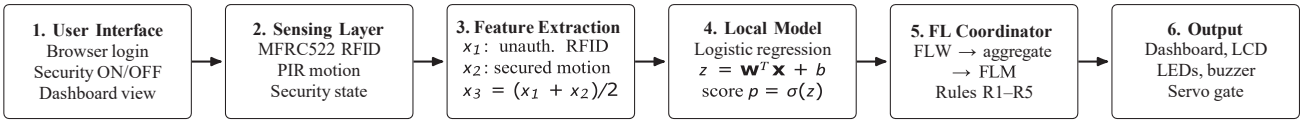


Fig. 1. Architecture of the proposed FL-IDS.

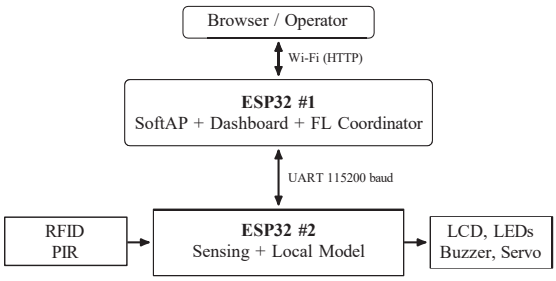


Fig. 2. Hardware block diagram of the two-node system.

TABLE III
DASHBOARD SUSPICIOUS-ACTIVITY RULES

Rule	Condition	Purpose
R1	All three configured accounts active at once	Unusual session state
R2	More than 20 tracked commands in 60 s	High command rate
R3	More than 3 failed logins in 60 s	Repeated failures
R4	More than 3 connected Wi-Fi stations	Station count
R5	Station MAC not in allowlist	Unknown station

B. Logistic-Regression Scoring

The model parameters are the weights w_1, w_2, w_3 and the bias b. The linear score and the model output are

$$z = w_1x_1 + w_2x_2 + w_3x_3 + b, \qquad p = \sigma(z) = \frac{1}{1 + e^{-z}}. \quad (4)$$

Each weight sets how strongly its feature contributes to z, with the sign giving the direction. The bias is a learned offset that moves the decision boundary independently of the inputs. The sigmoid maps z to $(0, 1)$; p is used as a score and should be read as a calibrated probability only after validation.

C. Local Training

For each labelled sample the firmware computes p, the error $p - y$, and accumulates gradients. The parameters are updated by gradient descent:

$$w_i \leftarrow w_i - \eta\,\overline{(p - y)\,x_i}, \qquad b \leftarrow b - \eta\,\overline{(p - y)}, \quad (5)$$

where the overbar denotes the average over the training samples. In the firmware $\eta = 0.15$ and training runs for 80 epochs over six labelled demonstration samples. These settings describe the implementation; they do not show that the model generalises to real intrusions.

D. Federated Parameter Exchange

The parameter vector is $\boldsymbol{\theta} = [w_1, w_2, w_3, b]$. The workflow in one round is:

1) ESP32 #2 initialises and trains the local model.
2) ESP32 #2 sends `FLW,w1,w2,w3,bias` to ESP32 #1.
3) ESP32 #1 parses the values and aggregates them.
4) ESP32 #1 returns `FLM,w1,w2,w3,bias`.
5) ESP32 #2 checks that every value is finite and within configured bounds, then replaces its parameters and recomputes the score.
6) ESP32 #2 periodically resends its local parameters.

A basic aggregation is the element-wise mean of the parameter vectors of two clients A and B,

$$\boldsymbol{\theta}_{global} = (\boldsymbol{\theta}_A + \boldsymbol{\theta}_B)/2. \quad (6)$$

Weighted FedAvg [1] generalises this by weighting client k with its sample count n_k:

$$\boldsymbol{\theta}_{global} = \sum_k \frac{n_k}{N}\boldsymbol{\theta}_k, \qquad N = \sum_k n_k. \quad (7)$$

The current prototype demonstrates the simple averaging case; it does not track per-client sample counts, round identifiers or model versions.

E. Rule-Based Dashboard Indicators

The coordinator evaluates the five rules in Table III. These are rule-based triggers and are *not* outputs of the logistic-regression model. When a rule fires, the coordinator can send `ALERT:SUSPICIOUS` to the sensor node, which switches on the red LED and buzzer for a fixed duration and then restores the normal indicator state. The alert state is therefore separate from the model score.

VII. SYSTEM IMPLEMENTATION

The sensor-node firmware is divided into functions for display, gate control, security state, alert timing, RFID and PIR processing, training and inference, parameter validation, parameter transmission and UART handling. The coordinator firmware hosts the HTTP server, login and session handling, connected-station monitoring, rule evaluation and aggregation. Login uses three configured accounts with application-level session state; every protected request should check the session. Connected stations are read from the SoftAP station list and compared with a configured MAC allowlist. This is an indicator only, because MAC addresses can be randomised or spoofed.

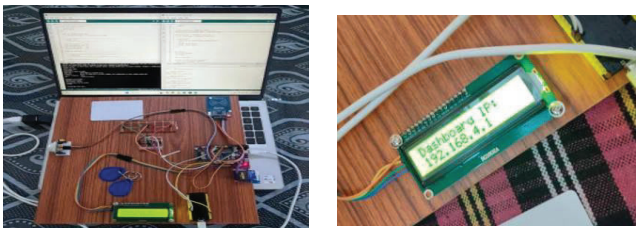
Messages are plain text lines terminated by a newline (Table IV). Receivers reject incomplete or malformed messages,

(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 11 , November - 2026

TABLE IV
UART MESSAGE FORMATS

Message	Direction	Purpose
SECURITY:1 / :0	#1→#2	Arm / disarm
ALERT:SUSPICIOUS	#1→#2	Timed alert
IP:⟨address⟩	#1→#2	LCD dashboard IP
EVENT:⟨text⟩	#2→#1	RFID / PIR / model event
SCORE:⟨value⟩	#2→#1	Current model score
FLW,w1,w2,w3,b	#2→#1	Local parameters
FLM,w1,w2,w3,b	#1→#2	Global parameters



(a) (b)

Fig. 3. Hardware prototype. (a) Assembled setup with ESP32 boards, RFID reader and tags, and LCD connected to the laptop for programming. (b) The I2C LCD showing the dashboard IP address 192.168.4.1 sent by the coordinator.

non-finite or out-of-range parameters, and clear any overlong line instead of appending indefinitely.

VIII. RESULTS AND DISCUSSION

A. Prototype Operation

Fig. 3(a) shows the assembled prototype. After start-up, the coordinator creates its SoftAP network and sends its address to the sensor node, which prints *Dashboard IP: 192.168.4.1* on the LCD (Fig. 3(b)). A browser connected to the access point can then open the dashboard at this address.

B. Firmware Deployment

Both firmwares were compiled and uploaded to ESP32 Dev Module boards (ESP32-D0WD-V3, dual-core, 240 MHz) through the Arduino IDE 2.3.10; the upload logs are shown in Fig. 4. Table V lists the resource usage reported by the compiler. The coordinator uses 73% of program storage because it contains the Wi-Fi stack and web server, while the sensor node, which runs sensing and training, uses 32%. Both leave ample dynamic memory, so the compact model and message protocol are comfortably within the microcontroller's limits.

C. Dashboard Output

Fig. 5 shows the coordinator dashboard. It displays the security state (*Armed*), the latest model score received from ESP32 #2 (0.78), the status of rules R1–R5, the list of connected Wi-Fi stations with a known/unknown label, and a log of recent events. In the displayed case rules R1–R4 are clear while R5 is triggered because one connected station is not in the allowlist, and the dashboard reports that

TABLE V
FIRMWARE RESOURCE USAGE REPORTED BY THE ARDUINO IDE

Resource	ESP32 #1 (coordinator)	ESP32 #2 (sensor node)
Program storage (bytes)	959,664	423,407
of 1,310,720	73%	32%
Global variables (bytes)	48,168	30,432
of 327,680	14%	9%

`ALERT:SUSPICIOUS` was sent to ESP32 #2. The example shows that the score and the alert flag are displayed as separate items, as intended.

D. Planned Evaluation Measures

Functional tests show that the hardware and message flow work, but they do not establish detection accuracy. A classifier evaluation needs labelled normal and attack cases, separate training and test data, a stated decision threshold and a confusion matrix with true/false positives and negatives (TP, FP, TN, FN):

$$\text{Precision} = \frac{TP}{TP+FP}, \quad \text{Recall} = \frac{TP}{TP+FN}, \tag{8}$$

$$F_1 = \frac{2\,\text{Precision}\cdot\text{Recall}}{\text{Precision}+\text{Recall}}, \quad \text{Acc} = \frac{TP+TN}{TP+TN+FP+FN}. \tag{9}$$

UART update latency, dashboard response time and model convergence over several rounds should also be recorded. No numerical detection-performance claim is made in this paper because verified measurements are not yet available.

E. Discussion

The prototype shows a useful integration of sensing, a local operator interface and a lightweight model-update protocol. Separating the two nodes makes it easy to add more clients later. The central research question, whether aggregation improves generalisation on representative data, remains to be answered; the present work provides the platform for that experiment.

IX. LIMITATIONS

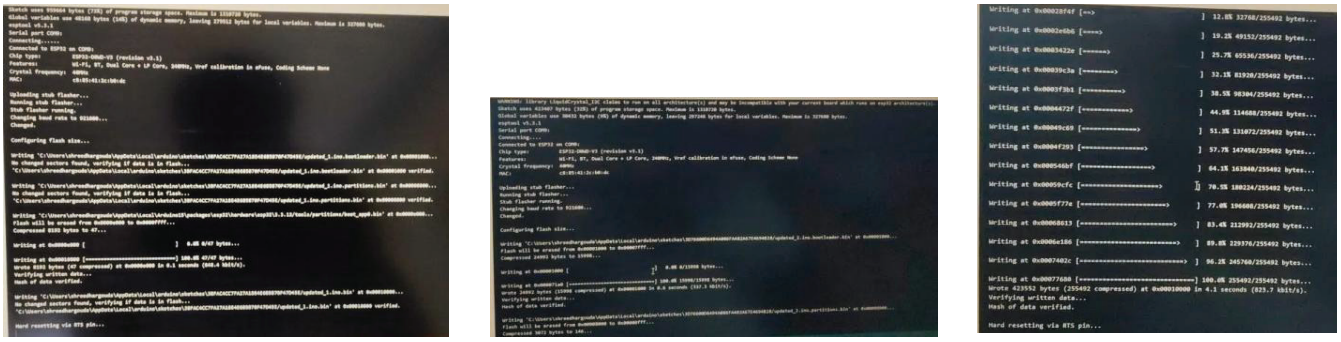
The feature vector covers only a few physical-access and motion indicators, and the six demonstration samples are not a comprehensive IoT intrusion dataset. The aggregation is a basic averaging demonstration and lacks client identification, round IDs and versioning. Passwords configured in firmware, MAC allowlists (which can be randomised or spoofed) and unauthenticated UART messages are unsuitable for production use. Accuracy, precision, recall, latency and convergence are not yet measured. The prototype is not a hardened security product.

X. CONCLUSION

This paper presented a two-node ESP32 prototype of a federated-learning based intrusion detection system for IoT devices. The coordinator hosts a local web dashboard and manages event and model exchange, while the sensor node integrates RFID, PIR, LEDs, a buzzer, an LCD and a servo

(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 11 , November - 2026



(a) (b) (c)

Fig. 4. Firmware upload logs from the Arduino IDE. (a) Coordinator firmware flashed to ESP32 #1 with "Hash of data verified". (b) Sensor-node firmware: compile summary and start of flashing. (c) Sensor-node flashing completed to 100% and verified.

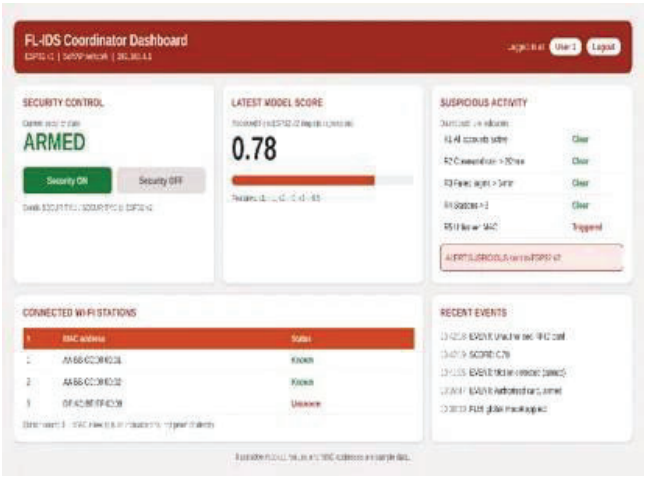


Fig. 5. FL-IDS coordinator dashboard served by ESP32 #1 at 192.168.4.1, showing security control, latest model score, suspicious-activity rules, connected stations and recent events.

gate with on-device logistic-regression training. The workflow of local training, parameter transmission, aggregation, validation and global-parameter use was implemented, and the firmware fits comfortably within the ESP32's memory. The work is presented as an embedded proof of concept and not as a validated accuracy claim.

XI. Future Work

Future work includes training and evaluating on a labelled, representative IoT intrusion dataset; implementing documented FedAvg with sample counts, round IDs and model versioning; comparing centralised, local-only and federated training; adding network-traffic features; adding authentication and integrity protection to UART messages; improving dashboard logs; adding more client nodes and measuring bandwidth, memory, training time and convergence; and testing under noisy sensor conditions.

References

[1] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. 20th Int. Conf. Artificial Intelligence and Statistics (AISTATS)*, 2017.

[2] P. Kairouz *et al.*, "Advances and open problems in federated learning," *Found. Trends Mach. Learn.*, vol. 14, no. 1–2, pp. 1–210, 2021.

[3] Y. Mirsky, T. Doitshman, Y. Elovici, and A. Shabtai, "Kitsune: An ensemble of autoencoders for online network intrusion detection," in *Proc. Network and Distributed System Security Symp. (NDSS)*, 2018.

[4] T. D. Nguyen, S. Marchal, M. Miettinen, H. Fereidooni, N. Asokan, and A.-R. Sadeghi, "D¨IoT: A federated self-learning anomaly detection system for IoT," in *Proc. IEEE 39th Int. Conf. Distributed Computing Systems (ICDCS)*, 2019, pp. 756–767.

[5] C. Kolias, G. Kambourakis, A. Stavrou, and J. Voas, "DDoS in the IoT: Mirai and other botnets," *Computer*, vol. 50, no. 7, pp. 80–84, 2017.

[6] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet Things J.*, vol. 3, no. 5, pp. 637–646, 2016.

[7] D. W. Hosmer, S. Lemeshow, and R. X. Sturdivant, *Applied Logistic Regression*, 3rd ed. Hoboken, NJ, USA: Wiley, 2013.

[8] Espressif Systems, "ESP32 series datasheet and Arduino-ESP32 documentation."

[9] NXP Semiconductors, "MFRC522 standard performance MIFARE and NTAG frontend," data sheet.

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