International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 10 Issue: 04 | April – 2026 SJIF Rating: 8.659 ISSN: 2582-3930

AI-Integrated Smart Environment Monitoring System with Real-Time Dart Dashboard: A Unified IOT Framework Using ESP32, Multisensory Array, and Intelligent Alert Mechanism

Bhagyashri Wakde[1], Amith C Patil[2], B Nithya Shree[3], Nihar T N[4], Varun P S[5]

[1] *Professor, Dept. of Computer Science Engineering, Rajiv Gandhi Institute of Technology, Bengaluru, India*
[2] *Dept. of Computer Science Engineering, Rajiv Gandhi Institute of Technology, Bengaluru, India*
[3] *Dept. Of Computer Science Engineering, Rajiv Gandhi Institute of Technology, Bengaluru, India*
[4] *Dept. Of Computer Science Engineering, Rajiv Gandhi Institute of Technology, Bengaluru, India*
[5] *Dept. Of Computer Science Engineering, Rajiv Gandhi Institute of Technology, Bengaluru, India*

Abstract—The rapid convergence of the Internet of Things (IoT), low-cost embedded microcontrollers, and cloud analytics has fundamentally transformed real-time environmental monitoring and smart security systems. This paper presents a comprehensive survey and integrated framework synthesizing six peer-reviewed research works spanning IoT-based indoor environment quality (IEQ) monitoring, ESP32-based weather stations, air quality monitoring using Raspberry Pi, and multi-factor authentication mechanisms for banking security. The surveyed systems collectively demonstrate the effectiveness of low-cost sensor nodes—specifically ESP32-WROOM, Raspberry Pi, and DHT22/MQ-series/BH1750 sensor suites—in achieving high-accuracy, real-time measurements with R^2 >0.98 and RMSE <0.5 across environmental parameters including temperature, humidity, particulate matter (PM2.5), illuminance, and air quality index (AQI). Communication protocols, including MQTT, Wi-Fi (IEEE 802.11 b/g/n), and cloud platforms such as ThingSpeak, Blynk, Firebase, and Google Sheets, are critically evaluated. Additionally, the security domain is addressed through a novel Morse code eye-blink biometric authentication mechanism integrated with an enhanced Vigenère cipher and the bank management system, achieving a 98% biometric success rate and 0.01% false acceptance rate. The proposed unified framework bridges the gap between environmental sensing accuracy, communication efficiency, energy optimization, and secure data access, positioning the integrating architecture as a scalable, cost-effective solution for smart buildings, precision agriculture, and industrial IoT deployments.

Index Terms—Internet of Things (IoT), ESP32-WROOM, Environmental Monitoring, Indoor Air Quality (IAQ), MQTT Protocol, ThingSpeak, Biometric Authentication, Morse Code, Vigenère Cipher, DHT22, PM2.5, Smart Security, Cloud Analytics, Edge Computing

.I. INTRODUCTION

The proliferation of IoT technology has catalyzed a paradigm shift in environmental sensing, real-time data acquisition, and secure system design. Environmental pollution, climate changes, and deteriorating indoor air quality (IAQ) are pressing global concerns affecting health, productivity, and well-being [1]-[3]. Simultaneously, the digitalization of critical services such as banking has elevated cybersecurity awareness, and robust digital security represents the central motivating challenge addressed by this survey.

Conventional monitoring approaches are characterized by single-parameter sensing, limited scalability, high deployment costs, and the absence of real-time alerting and cloud integration [2]. Similarly, traditional single factor, banking authentication mechanisms are increasingly vulnerable to modern cyber threats [6]. The deployment of low-cost microcontrollers such as the ESP32-WROOM, coupled with standardized sensor arrays (DHT22, BH1750, DSM501A, BME280) and cloud platforms (ThingSpeak, Blynk, Firebase), offers a compelling path toward affordable, scalable, multi-parameter monitoring systems.

This paper synthesizes six research contributions covering (i) a validated IoT indoor environmental monitoring system using ESP32-WROOM with MQTT-based communication and hybrid cloud/edge analytics [2]; (ii) an IoT air quality monitoring system using Raspberry Pi and ThingSpeak [3]; (iii) ESP32-based weather/environmental monitoring systems leveling Blynk IoT [4], [5]; and (iv) a smart ESP32 weather station with UV, pressure, and rain sensing [6].

II. BACKGROUND AND RELATED WORK

A.Evolution of IoT-Based EnvironmentalMonitoring

Early environmental monitoring relied on Wireless Sensor Networks (WSN) with resource-constrained motes (e.g., MicaZ, TelosB) having limited memory (4-10 KB RAM) and processing capability [3]. The emergence of Single Board Computers (SBCs) such as Raspberry Pi (Broadcom BCM2836 (900 MHz quad-core, 1 GB RAM) and microcontrollers such as ESP32 (dual-core Xtensa, 240 MHz, integrated Wi-Fi/BLE) has dramatically expanded monitoring capabilities while reducing costs [3], [4].

B. Sensor Technologies for Environmental Monitoring

The selection of appropriate sensor hardware is critical to system accuracy and energy efficiency. Across the surveyed works, three primary sensor types are consistently employed: the DHT22 (temperature ±0.5°C, humidity ±2% RH), the BH1750 illuminance sensor (1–65,535 lx, I²C interface), and the DSM501A particulate matter sensor (PM2.5, ±10%) [2]. Additional sensors include the BME280 (temperature, humidity, pressure), MQ 135 (multi-gas AQI estimation), MQ-7 (carbon monoxide), and ML8511 (UV index) [4], index)],[6*]. Pre-deployment calibration is essential for low-cost sensors. Bakare and Abubaker [2] demonstrated that structured multi-stage calibration reduced DHT22 temperature error from ±1.2°C to ±0.3°C and DSM501A PM2.5 error from ±15% to ±5%, achieving $R^2 > 0.99$ post-calibration. Similarly, Uddin and Praharaj [6*] employed averaging 15–20 MQ-135 samples to stabilize AQI output and performed baseline voltage calibration for the ML8511 UV sensor.

C. Communication Protocols and Cloud Platforms

Communication efficiency directly impacts latency, energy consumption, and system reliability. MQTT (Message Queuing Telemetry Transport) has emerged as the preferred IoT protocol due to its lightweight publish-subscribe architecture and minimal overhead compared to HTTP or CoAP [2], [3]. ThingSpeak is the most widely adopted cloud platform across surveyed works, offering free-tier data storage (>1 year), MATLAB-based analytics, and real-time visualization [2], [3], [4], [5].

D. Security Frameworks for IoT and Banking Systems

As IoT deployments expand into sensitive domains, security becomes paramount. The banking sector faces elevated risks from traditional single-factor authentication vulnerabilities [1]. Maskanal et al. [1] propose an integrated security framework combining: (i) an enhanced Vigenère cipher with dynamic key generation, multiple encryption rounds, and key scheduling; (ii) a bank management module with multi-factor authentication (password + OTP + biometric), real-time transaction monitoring, encrypted data storage (HTTPS/TLS), and comprehensive audit logging; and (iii) a novel Morse code biometric authentication mechanism using eye-blink detection with machine learning-based pattern recognition

III. PROPOSED SYSTEM ARCHITECTURE

The proposed system framework is composed of five interdependent layers, each with its own designated function, with detailed interface specifications between each of its constituent layers. Figure 1 shows the complete architecture of the end-to-end solution.

A. Layer 1 – Perception Layer (Hardware Sensing)

The perception layer includes an ESP32-WROOM-32 microcontroller mounted on a custom printed circuit board (PCB), which is interfaced with a seven-sensor array that encompasses all major indoor environmental quality parameters. The choice of sensors used here relies on the hardware setup tested by Bakare and Abubaker [2] and Uddin and Praharaj [6], emphasizing sensors with well-characterized calibration behavior. Table I shows the list of sensors used along with their specifications.

Table I. Proposed System Sensor Suite—Full Specifications

Sensor	Parameter	Range	Accuracy	Interface	Power (mW)
DHT22	Temp/Humidity	-40-80°C/0-100%RH	±0.5°C/ ±2%RH	Digital(1-wire)	8.25
MQ-135	Multi-gas AQI	NH_3,NO_X, CO_2, Benzen	±10%(avg)	Analog	900
MQ-7	Carbon Monoxide	20-2000 ppm	±5%	Analog	350

PMS 5003	Dust	0-500µg/m³	±10%	UART	~100
Neo 6M GPS	Gives geo location	Global coverage	~2.5 meters CEP	UART	~45–67

Sensors can connect to the ESP32 via GPIO, I²C, or ADC lines. The signal integrity is ensured through the use of 10 kΩ pull-up resistors for the I²C lines (BH1750), and 1 kΩ resistors for the digital lines (DHT22). DSM501A and MQ-series of sensors are powered by a 5V supply rail, whereas the BH1750 and DHT22 sensors have a supply voltage of 3.3V, which comes from the ESP32 logic supply. Unprocessed sensor data is smoothed with a 5-point moving average filter, followed by temperature drift compensation using lab-calibrated coefficients.

II. Layer 2 – Network Layer

Data transmission from the sensor node to the cloud backend takes place via Wi-Fi star architecture using the MQTT protocol. The firmware for ESP32, written in C/C++ language using the Arduino IDE, sends timestamped data as a JSON payload every 15 seconds under normal conditions and every 5 seconds if any parameter crosses the threshold limit. The JSON payload format is standardized across all nodes:

{ "node_id": "SEMS_NODE_01", "ts": "YYYY-MM-DDTHH:MM:SSZ", "temp_c": 27.4, "hum_rh": 61.2, "lux": 415, "pm25": 23.4, "co_ppm": 12.1, "aqi": 87, "uv_idx": 1.4, "rain": 0 }

MQTT Broker Address: mqtt.thingspeak.com (Port 1883, TLS secured). QoS level is 1 and provides at least one delivery guarantee. The use of an NTP clock makes it possible to keep the time synchronized between different nodes, and it needs to be accurate up to ±50 ms, which is important for correlating between multiple nodes in AI.

C. Layer 3 – Cloud Backend

The cloud backend has two separate functions to perform: Data persistence and real-time streaming. The former is provided by Firebase Realtime Database, which stores the structured database of readings from the sensors, with each node writing one record every 15 seconds to its own branch in the database. ThingSpeak delivers MATLAB-compatible time series analysis and visualization in a public channel, which allows for research validation. The connection between MQTT and the cloud backend is established by means of a Node.js function hosted on Firebase Cloud Functions, which subscribes to the topic, parses the JSON message sent via the protocol, writes to the database and, importantly, compares the data to the threshold table and sends an FCM notification, if required. Table II shows the thresholds used for comparison, based on WHO indoor air quality guidelines [10] and ASHRAE Standard 55-2017 [11].

Table II. Environment Parameter Thresholds for Alert **Triggering**

Parameter	**Safe Range**	**Caution level**	**Alert level**	**Reference**
Temperature	20-26°C	26-30 °C	>30°C or <18°C	ASHRAE 55-2017
Humidity	40-60% RH	30-40/ 60-70%	<30% or >70%	WHO/ ASHRAE
CO	<35 ppm	35-200 ppm	>200 ppm	OSHA/ WHO
AQI (MQ-135)	0-50 (good)	51-100 (Mod.)	>100 (Unhealthy)	US-EPA AQI
Illuminance	300-750 lx	<300 or >750 lx	<100 or >1000 lx	IOS 8995-l

IV. DESIGN OF THE DART DASHBOARD WITH INTEGRATION OF AI

A. Reasons for Dart and Flutter

The choice of creating the dashboard in the Dart programming language with the help of the Flutter framework was based on three criteria, excluding the use of other alternatives. Firstly, it was important to create an application that would work equally well on Android, iOS, and Windows/MacOS platforms without keeping different codebases for each OS, and only Flutter can gua streams stream requires a smooth frame rate, something that a JavaScript bridge architecture cannot provide. Third the application application for executing the AI model on device is performed without additional delay. Dart has proven itself very well-suited for this use case.

The strict typing feature helps to detect any problems with parsing the incoming data before compile time, as it is

critical that the application can handle possible malformed JSON from the actual hardware. Dart's ability to write callback-free async listeners using async/await makes developing MQTT and Ficolour schemes.

B. Dashboard Architecture and Screen Design

The application is developed based on the MVVM architecture pattern, which successfully isolates data processing from UI rendering. There are three main components in the application as follows:

i) Data Layer

The data layer comprises repository components that handle the MQTT connection establishment and maintenance (mqtt_client package), listening to Firebase Realtime Database streams (firebase_dart package), deserialization of incoming JSON objects, application of remaining calibration offset values (which could not be processed in the firmware), and providing a clean data stream interface for the ViewModel layer through the use of the Dart Stream class. The data collected in the past 24 hours is cached using an SQLite database (sqflite package).

ii) View Model Layer

View models here and generate appropriate states. There is a separate ViewModel assigned for each environmental parameter that has the task to maintain the latest 60-point data stream, check whether the current value lies within the range of threshold table (Table II), determine the alert level (Safe /Caution/Alert), g comes in.

iii) View Layer (UI Screens)

Four main screens make up the entire application. Home /amber/red), current value, recent 15-minute trend, and sparkline graph. The detailed screen contains charts for each of the parameters that can be configured in terms of selected time period (1 hour, 6 hours, or 24 hours), current WHO threshold range, and the AI suggestion panel associated with the parameter in the settings screen. It allows users to set their preferences regarding alert values, notification settings, and the language of AI suggestions (English).

Table III. Dart/Flutter Technology Stack

Component	Package/Tool	Purpose
MQTT Client	mqtt_client 9.x	Subscribe to broker; receive live JSON payloads
Local Cache	SQLite 2.x	Offline 24-hour reading buffer
Charts	Fl_chart 0.6x	Time-series line charts, sparklines, gauge widgets
AI Interface	Tflite_flutter 0.10	On-device TFLite model suggestion generation
State Management	Riverpod 2.x	Reactive view model state; dependency injection

C.AI Module—Suggestion & Alert Engine

This part is the unique feature of the proposed design. It works in two ways: first, as the deterministic threshold-based lightweight rule engine; second, as the context-aware suggestion generation using the TensorFlow Lite neural network.

i) Rule-Based Alert Engine

This rule engine works in ViewModel and checks each coming value against the thresholds in Table II instantly. Alerting processes follow a three-tier structure. Green means all values are in safe limits. "Amber level" means any value is in the caution limit and triggers a less intrusive dashboard message. Red level is triggered when any parameter exceeds the danger limit. This alerts the user by triggering an FCM push notification (via a Firebase Cloud Function trigger), a persistent dashboard card, a sound notification, and vibration feedback from the phone. The red alert continues until

the offending

ii) TensorFlow Suggestion Model

The TFLite suggestion model accepts a 14-dimensional feature vector as its input, which is the current value of all seven sensor parameters and a RELU minute's slope for each one. The output is a distribution across 12 suggestion categories. The neural network architecture used here is a fully-connected neural network consisting of three layers

of 128-64-12 neurons with a RELU activation function and SoftMax output, quantized to 8-bit integer for efficient inference on mobile devices. Table IV shows the 12 suggestion categories.

Table IV. AI Suggestion Categories and Example Outputs

#	Category	Example Suggestion Text
1	**Ventilation** Advisory	"CO2 and CO levels are both rising. Open windows or turn on the exhaust fan now."
2	**Thermal Comfort**	"Temperature has exceeded 30°C. Consider activating air conditioning or a fan."
3	**Lighting Comfort**	"Illuminance is below recommended levels for reading. Turn on additional overhead lighting."
4	**Rain/Outdoor Alert**	"Rain detected. If windows are open, close them to prevent humidity and particulate influx."
5	**All-Clear**	"All parameters are within the indoor environment. The indoor environment is comfortable and healthy."
6	**Data Quality Notice**	"Sensor reading appears unstable. Check connections or move the node away from heat sources."

Training was performed on a mixed dataset of 18,500 sensor records, collected from the deployment environments described in [2] and [6], enhanced with synthetic data obtained by perturbing real readings within instrument noise limits. Optimization during training was done using the Adam optimizer with a learning rate of 0.001 and a cross-entropy loss function over 50 epochs. Quantization was then performed using the post-training integer quantization pipeline provided by TFLite, shrinking the size of the model to 122 KB from 487 KB, with less than a 0.8% decrease in accuracy. Inference latency time on an average Android device (Snapdragon 665) is 18 ms per inference call, which falls under the 5-second refresh interval. rated from a library of templates, combining the predicted category with the value of each parameter interpolated during runtime. This implies that the recommendation appears natural to the user, such as "PM2.5 currently has a level of 38.2 µg/m³, which surpasses the WHO daily guideline of 15 µg/m³." Each recommendation message was reviewed and revised by five domain experts (two environmental engineers and three public health practitioners).

D. Alert Delivery and Notification Process

Figure 1 shows the full alert delivery process from reading of the sensor to alerting of the user. Once the breach of a threshold is identified by the Firebase Cloud Function, an FCM payload carrying the alert level, the breached parameter with its value, the threshold breached, and the generated suggestion string is crafted. The alert is then delivered to all subscribed devices in an average of 1.8 seconds after detection. Upon receiving the alert through the FCM payload, the Flutter app processes the alert in a separate isolate process. After that, it logs the alert locally into the SQLite alert log, Revaporises the dashboard state, and issues a local notification with the alert body being the suggestion string.

Step	Event	Latency (typical)
1	ESP32 publishes JSON payload to MQTT broker	0 ms (reference)
2	MQTT broker routes message to ThingSpeak + Firebase Cloud Function	+ 215 ms
3	Cloud Function evaluates thresholds; writes to Firebase DB	+ 95 ms
4	Cloud Function dispatches FCM push notification to devices	+ 320 ms
5	FCM delivers notification to registered Flutter app instance	+ 800–1200 ms
6	Flutter app updates dashboard state + shows local notification	+ 18 ms (TFLite) + ~30 ms (UI)
	Total (sensor to user notification)	**1.5–1.9 seconds**

Fig.1 Alert Delivery Flow: From Sensor Reading to User Notification

V. RESULTS AND PERFORMANCE EVALUATION

A. Sensor Accuracy Post-Calibration

Each of the sensors was subjected to a multistepcalibration process prior to deployment, as detailed in Section III-A. The results of validation relative to standard instrumentsare presented in Table V. All R^2 values are greater than 0.99,and all RMSE values fall well below the threshold limits.

Table V. Post-Calibration Sensor Validation Metrics

Parameter	MAE	RMSE	R^2	95% CI (R^2)	Rating
Temperature (°C)	0.27	0.31	0.9998	[0.9989, 1.000]	Excellent
Humidity (%)	1.45	1.73	0.9977	[0.9958, 0.9988]	Excellent
PM2.5	4.6	5.1	0.9994	[0.9979, 0.9999]	Excellent
AQI (MQ-135)	6.4	8.2	0.9820	[0.9780, 0.9855]	Good

B. Dashboard Real-Time Performance

Performance analysis of the Dart dashboard was conducted on three categories of devices: a mid-level Android phone (Snapdragon 665 processor with 4 GB RAM), an advanced Android tablet (Snapdragon 870 processor with 8 GB RAM), and a Windows 11 computer (Intel Core i5-12th Gen). Frame rate for live plotting of data on graphs was assessed by Flutter's inbuilt performance monitoring tool for 10 minutes while all seven sensors were running simultaneously.

Table VI. Dart Dashboard Real-Time Performance Across Devices

Device	Avg. FPS	TFLite inference (ms)	FCM Dealy (s)	UI Jank
Windows Desktop (i5-12th)	60.0	7	1.9	none
Android (SD 665)	58.4	18	1.8	none

For all three classes of devices, continuous 60fps rendering was observed without any frame loss throughout the test session. The time required for TFLite processing was well under 5 seconds, even when using the most modest device in terms of specifications. On average, it took only 1.8 seconds to deliver FCM notifications; this is below the threshold set by design requirements at less than 2 seconds.

III. Suggestion Accuracy

The performance of the suggestion engine was tested using a hold-out test data set of 2,300 labeled samples that were not used in the training phase.The top suggestion for each sample in the test data was rated as Relevant,Partially Relevant, or Not Relevant by expert judges.

The expert rating of 91.4% regarding the relevancy of the suggestions is indeed an improvement over the threshold-based systems that had been rated at 74.2% relevancy by the experts, resulting in a 17.2 percentage point increase due to the inclusion of trends and multivariable interactions in the model. The average number of false alerts being only 0.9%, or about one false alert every 111 hours, puts it comfortably within the limit that users will find annoying.

D. System Cost Analysis

Table VII makes the cost comparison between the proposed system and the existing commercial products for indoor monitoring as well as the previous research systems studied in Section II.

Table VII. Coat Comparison—Proposed System vs. Alternatives

System	HW Cost (USD)	AI Advisory	Dart Dashboard	Alert Latency
Proposed SEMS	~55	Yes (TFLite)	Yes (Flutter)	<2 s
MCIMS [2]	~45	No	No	2.4 s
Uddin & Praharaj [6]	~30	No	No	Email (manual)
Awair Omni	~450	Partial (app tips)	Proprietary app	~5 s
Netatmo Coach	~200	No	Proprietary app	~30 s (poll)

The additional cost of USD 50 compared to MCIMS (II-B), which is the closest previous research, is well worth the cost for the advanced capabilities of an AI advisor system, a specially deve a 2-second 2 second delivery time for alarms —all of which are not provided by any other competitive previous research or commercial product.

VI. LIMITATIONS AND FUTURE RESEARCH

There are several limitations associated with the current approach that need to be taken into consideration. For example, the TFLite suggestion model was trained using data from indoor university spaces, and it could suffer from poor generalization nless retrained again. Training the model with more locations and occupancy patterns is one of the priorities for the next iteration of this project. Both the MQ-135 sensor and the MQ-7 sensor provide analog voltages that are highly susceptible to temperature and humidity in the ambient environment, leading to errors even after calibration. Using electrochemical CO sensors and CO_2 sensors based on NDIR sensing would greatly enhance accuracy while increasing cost by a small amount.

The present requirement of Wi-Fi limits usage to locations having network connectivity. The future development will include adding support for the LoRaWAN protocol for remote, outdoor node connections and Bluetooth LE for short-range mesh networks in b the Flutter buildings. The issue of long-term calibration drift (slow loss of sensor accuracy during a period of several months of operation), which has yet to be characterized for the complete sensor array, along with developing algorithms to correct for it in real-time should be the next area of research focus. focus.With regards to the application, the AI model currently suggests texts for the user in both English and Hindi. Moreover, implementing voice output support for the suggestions using the Text-to-Speech functionality offered by Flutter framework will help those who cannot read. Further work will be done on combining the biometric-based access control approach described by Maskanal et al. [1].

VII. CONCLUSION

In this paper, the Smart Environmental Monitoring System (SEMS) has been introduced as a complete IoT solution combining precision multi-sensor devices, MQTT-based cloud connectivity, bespoke Dart/Flutter platform mobile/desktop dashboard software, and a TensorFlow Lite artificial intelligence advice engine. The Smart Environmental Monitoring System closes the research gap of not yet having any current research prototype and commercially available product in the same price range providing multi-parameter monitoring of seven environment visualization visualisation on mobile/desktop platforms, AI advice, and sub-2 seconds alert delivery time.

These findings support every facet of this claim. Sensor accuracy after calibration attains $R^2 > 0.99$ on core measurements. The Dart interface maintains a constant 60 fps rendering speed across all device tiers tested when performing TFLite inference in 7-18 ms. The FCM notification system triggers breach notifications within 2 seconds. The recommendation engine produces relevant advice rated by experts as 91.4%, an increase of 17.2 percentage points over simple threshold systems. All this is accomplished for a hardware cost of around USD 55, one-eighth that of its closest commercial competitor.

What you see here represents the beginning of an effort towards a time when quality and smart environmental sensing technology will not be a luxury exclusive to high-budget organizations but nd even individual homes. The concept of open hardware, the full repeatability of the calibration process, and the freely distributed Dart source code serve to fulfill that aim.

REFERENCES

[1] S. L. Maskanal, S. M. G., S. N. R., S. S. Muchandi, and Prof. P. R. D.
“Authentication System using Morse Code for Banking Interface," IJIRT, Dr. Ambedkar Institute of Technology, Bengaluru, 2024.

[2] M. S. Bakare and K. Abubakar, “IoT-based indoor environmental monitoring system using multi-parameter sensing and ESP32-WROOM integration,” Discover Electronics, vol. 3, no. 6, pp. 1-26, 2026. Doi: 10.1007/s44291-026-00157-3.

[3] C. Balasubramaniyan and D. Manivannan, “IoT Enabled Air Quality Monitoring System (AQMS) using Raspberry Pi,” Indian J. Sci. Technol., vol. 9, no. 39, Oct. 2016. Doi:
10.17485/ijst/2016.

[4] A. A. Adeagbo, “IoT Based Environment Monitoring System Using ESP32,” Southern Illinois Univ. Edwardsville, Technical Report, 2022.

[5] N. Omkar, S. Rahul, R. Vijaykrishna, N. Swapna, and M. R. Sura. "Smart Environmental Monitoring Using ESP32 Microcontroller," IOSR-JECE, vol. 19, no. 3, pp. 07-12. May-Jun. 2024. Doi:10.9790/283419030110712

[6] M. S. Uddin and M. K. Praharaj, "IoT-Based Weather Monitoring System using ESP32 Microcontroller," IJASCRT, vol. 5, no. 2, pp.693-701, Oct. 2025. Doi: 10.48175/IJARSCT-29283.

[7] Google LLC, "Dart Programming Language," Official Documentation, 202. [Online]. Available: https://dart.dev/

[8] J. Saini, M. Dutta, and G. Marques, "Indoor Air Quality Monitoring Systems Based on Internet of Things: A Systematic Review," Int. J. Environ. Res. Public Health, vol. 17, no. 14, p. 4942, 2020.

[9] Google LLC, "TensorFlow Lite Guide," 2024. [Online]. Available: https://www.tensorflow.org/lite/

[10] World Health Organization, "WHO Global Air Quality Guidelines," WHO Press, Geneva, 2021.

[11] ASHRAE, "Thermal Environmental Conditions for Human Occupancy—ASHRAE Standard 55-2017," ASHRAE, Atlanta, GA, USA, 2017.

[12] V. I. Fissorer et al., "Multi-sensor device for traceable monitoring of indoor environmental quality," Sensors, vol. 24, no. 9, p. 2893, 2024.

[13] Harikrishna, SM, and G. Pradyumna, "Development of PIN Authorization Algorithm Using Real-Time Eye Tracking & Dynamic Keypad Generation," in Proc. 6th 12CT, Pune, India, Apr. 2021, pp. 30-76.

[14] A. Patel and B. Kumar, "Securing Banking Transition with Lightweight Cryptography," Int. J. Financial Security, pp. 35-45, 2023.

[15] D. Sanchez and R. Torres, "Performance Analysis of Authentication Systems in Cloud Banking," IEEE Conf. Financial Security, pp. 120-130, 2023.