HMI For E-Surveillance IoT Panel In Smart Multi-location Facility Management System

[1]Farid Ahmad Shaikh, [2]Sonia Rohit Joshi, [3]Makarand Govind Kulkarni
Dept. of Electronics Engineering
K J Somaiya School of Engineering
Somaiya Vidyavihar University
Mumbai, INDIA

***Abstract*—The design and implementation of a Human-Machine Interface(HMI) for an IoT based E-Surveillance panel aimed at enhancing facility management in multi-location environments. With the rapid expansion of IoT technology, managing multiple facilities remotely has become increasingly feasible and necessary for modern organizations. Our proposed HMI integrates diverse surveillance and monitoring systems, allowing for seamless, centralized control of various functionalities, including video surveillance, intrusion detection, energy monitoring, and environmental controls, across geographically dispersed sites. The system leverages data-driven analytics and a unified interface to provide real-time feedback and support decision making, helping to improve facility security, operational efficiency, and resource utilization. By enabling intelligent automation and remote management capabilities, this E-Surveillance IoT Panel offers a scalable solution for the complex demands of smart facility management. Smart surveillance integrates the Internet of Things (IoT) with machine learning to create advanced security systems capable of real-time monitoring, analysis, and response. Automated Threat Detection and Behavior and Pattern Recognition is possible.**

***Index Terms*—GUI Design, Intelligent Video Surveillance System, PySide, QT, SQL**

I. Introduction

In recent years, the rapid expansion of the Internet of Things (IoT) has transformed traditional facility management, enabling organizations to monitor and manage their facilities remotely and more effectively. Facility management for geographically dispersed sites has historically been challenging, often requiring significant manpower, onsite monitoring, and manual data collection [1]. However, IoT technology now offers a way to streamline these processes, providing real-time data and automation capabilities that allow for centralized and remote oversight. This shift is particularly relevant for organizations operating across multiple locations, such as corporations with numerous office buildings, campuses, or retail sites, where maintaining consistency and security across all facilities is crucial. The demand for more robust and unified facility management solutions has driven the development of integrated Human-Machine Interfaces (HMIs) that bring together different monitoring systems into one cohesive platform. Such HMIs enable centralized control over diverse systems like video surveillance, environmental controls, and energy management [2]. Additionally, data-driven analytics within these HMIs can assist in decision-making by offering insights on factors like energy consumption, occupancy patterns, and environmental conditions. These capabilities not only enhance the security and operational efficiency of facilities but also help in meeting sustainability goals by optimizing resource usage [3]. Our project addresses this need by designing and implementing an IoT-based E-Surveillance panel equipped with a comprehensive HMI. This system is intended to be scalable and adaptable, capable of integrating new devices and responding dynamically to the unique needs of each facility [4]. The E-Surveillance HMI aims to consolidate facility management operations onto a single interface, providing facility managers with greater visibility and control across multiple locations. The platform's capabilities are further enhanced by automation and real-time analytics, which help organizations to proactively manage risks, improve operational efficiency, and make data-informed decisions [5].

II. Literature Review

A. Background

In the era of smart technology, the demand for efficient multi-location facility management systems has grown exponentially. Traditional methods of facility management often rely on manual monitoring and operations, leading to inefficiencies, high costs, and delayed responses to critical issues [6]. With the advent of IoT (Internet of Things) and HMI (Human-Machine Interface) technologies, a new paradigm has emerged, enabling centralized, automated, and real-time control of facilities across multiple locations. IoT Panels play a crucial role in connecting physical devices (e.g., sensors, cameras, actuators) to the digital interface provided by HMI [7]. These panels gather data from IoT devices, process it, and provide actionable insights to operators through an intuitive HMI [8]. This integration is especially valuable in applications like e-surveillance, energy management, and environmental monitoring [10].

The integration of IoT into surveillance and facility management systems has transformed traditional security infrastructure into intelligent, connected ecosystems [6]. With the proliferation of smart devices and high-speed communication

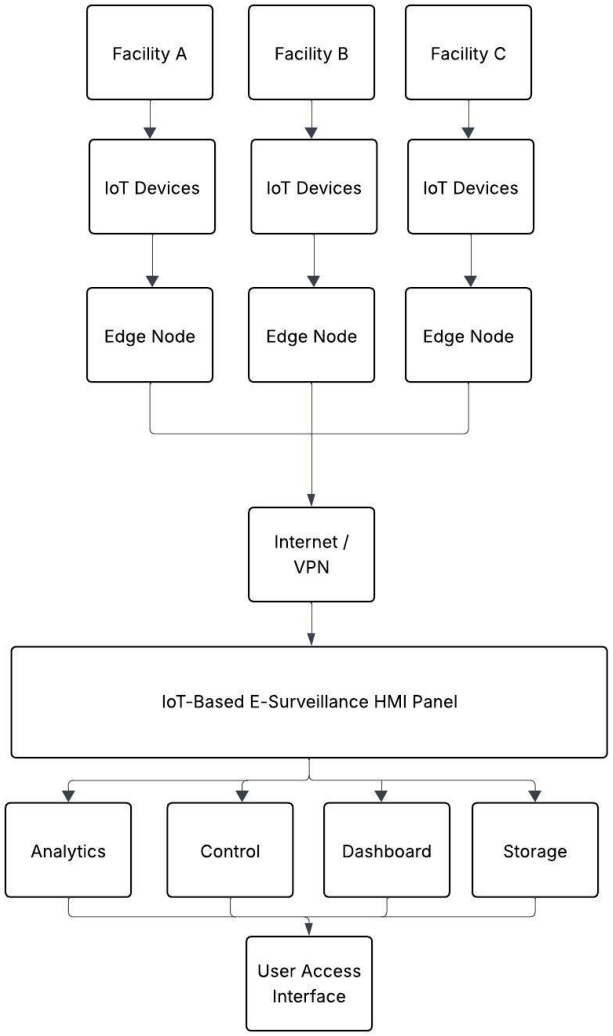

Fig. 1. Basic Block Diagram

networks, organizations are now capable of monitoring and managing multiple facilities from a centralized control unit. This shift enables improved responsiveness, efficiency, and data-driven decision-making in real-time environments [10].

B. Related work

With the proliferation of IoT technologies, the demand for efficient monitoring and visualization systems has grown significantly. IoT panels, which aggregate data from multiple sensors, are critical for real-time decision-making in smart facility management systems. However, ensuring that this data is presented in an intuitive and actionable format remains a challenge. Human-Machine Interfaces (HMIs) have emerged as an effective solution for bridging the gap between raw IoT sensor data and user-friendly visualizations. Various approaches have been explored for integrating IoT data with HMI systems. Cloud-based platforms such as AWS IoT Core and Google Cloud IoT offer dashboards for monitoring and analysing sensor data. These solutions are effective for remote facilities but rely heavily on internet connectivity, making them less suitable for environments where local control is critical. In contrast, web-based HMIs built using technologies like HTML, CSS, and JavaScript provide flexible visualization solutions. While these are widely adopted, their reliance on browser-based interfaces can introduce latency, especially when handling real-time data from multiple sensors [6]. Desktop and embedded systems, such as those built using the QT framework, are gaining traction in industrial IoT settings due to their performance and adaptability. QT's rich graphical libraries and cross-platform support make it an ideal choice for developing robust and responsive HMIs [3].

C. Our Contribution

In this work, we focus on implementing an IoT panel that visualizes sensor data through an HMI developed using QT for Python. The panel is designed to collect data from various sensors, such as temperature, humidity, and air quality monitors, and display this data in real time. The following steps outline the key features and implementation of our approach:

1) Sensor Integration: Sensors such as temperature and humidity modules are connected to the IoT panel through communication protocols like MQTT and I2C. Data is aggregated on the panel and processed for visualization.

2) Real-time Visualization: The HMI displays sensor data in graphical formats, such as line graphs for temperature trends and bar charts for humidity levels [9]. Alerts and warnings are integrated into the interface, with color-coded visual indicators for critical sensor values.

3) HMI Development Using QT: QT for Python (PySide6) was used to design a scalable and responsive interface [3]. Features include zoomable charts, interactive dashboards, and buttons for controlling devices connected to the IoT system.

4) Local and Cloud Storage: The panel uses SQLite for offline storage and synchronizes data with a cloud server for centralized monitoring when connectivity is available [4].

III. METHODOLOGY

A. QT Framework

The development of this IoT-based E-Surveillance Human-Machine Interface (HMI) utilizes a combination of technologies focused on delivering a robust, user-friendly, and scalable platform. The QT framework serves as the foundation for the HMI's graphical interface, providing a cross-platform solution that enables consistent functionality and design across different operating systems. QT's extensive libraries allow for creating highly customizable and interactive user interfaces, essential for visualizing real-time IoT data, alerts, and system controls. QT also offers support for multimedia, making it well-suited for integrating video feeds, graphical displays, and interactive elements critical in monitoring and surveillance applications[3].

B. Python Programming

Python is the primary language used for backend programming, processing data from IoT devices, handling system logic, and interacting with the QT-based front end. Python's versatility and extensive libraries make it ideal for developing the backend modules that support data processing, analytics, and communication with IoT devices. Python libraries such as PyQt5 or PySide are employed to integrate Python with the QT framework, bridging the backend and the interface, enabling real-time data visualization, and implementing control mechanisms [5].

C. Hardware Used Raspberry Pi (RPI)

The IoT-based E-Surveillance HMI leverages the Raspberry Pi (RPI) as a central hardware component, serving as a versatile and cost-effective device for running the HMI software and managing interactions with IoT devices and sensors. The Raspberry Pi's compact size, low power consumption, and compatibility with a wide range of sensors and modules make This chapter can be summarized as the logical thinking behind the proposed logic of HMI GUI design and implementation of ML.

D. Protocols and Data Handling

The HMI connects to various IoT devices via standard IoT communication protocols such as MQTT (Message Queuing Telemetry Transport) or HTTP/REST APIs, depending on M. Tech (Electronics Engineering) Batch: 2023-2025 Page — 11 the specific devices and sensors used. These protocols ensure reliable, low-latency communication between the HMI and the distributed sensors and controllers, allowing real-time data collection and command execution across multiple locations [5].

E. Database Management System (DBMS)

A database management system, potentially a lightweight database like SQLite or a more scalable solution like PostgreSQL or MySQL, is used to store historical data, system logs, and configuration settings. The DBMS allows the HMI to access and display historical data trends, enabling better decision-making and performance analysis.

F. Data Analytics and Visualization Libraries

For data processing and analytics, Python libraries like Pandas, NumPy, and Matplotlib (or PyQtGraph for QT) support the real-time analysis and visualization of IoT data, providing insights on key metrics such as energy usage, security alerts, and environmental conditions. This layer of analytics is critical in helping users make informed decisions and spot patterns that might require attention.

G. Security Protocols

Cybersecurity is essential in IoT-based applications, especially for surveillance. SSL/TLS encryption, secure user authentication (OAuth or two-factor authentication), and data encryption ensure that sensitive data transmitted between devices, databases, and the HMI remains secure. By leveraging these technologies, the HMI offers a unified, data-driven platform that enhances facility management by supporting real-time monitoring, interactive controls, and intelligent automation for security, environmental monitoring, and resource optimization across multiple sites

IV. IMPLEMENTATION

The implementation of the IoT-based E-Surveillance Human-Machine Interface (HMI) involves a systematic approach to integrating hardware, software, and communication protocols to create a cohesive system for multi-location facility management. The HMI is designed to provide facility managers with an intuitive, scalable, and responsive platform for real-time monitoring and control [9]. Below, we outline the key steps and components involved in the implementation process, including the design of the graphical user interface (GUI), hardware setup, IoT device integration, data processing, and system testing.

A. HMI Design with QT Framework

The HMI's graphical user interface (GUI) was developed using the QT framework, specifically leveraging PySide6 for Python integration. The GUI is modular, allowing for dynamic updates and seamless integration of new functionalities. Key design considerations include:

1) Modular Layouts: The interface is divided into distinct panels for different functionalities, such as video surveillance, environmental monitoring, energy usage, and alerts. Each panel is designed to be collapsible or expandable, ensuring that users can prioritize the most relevant information.

2) Real-Time Data Visualization: Sensor data (e.g. temperature, humidity, air quality) is displayed using interactive widgets, including:

- Line graphs for tracking trends over time.
- Bar charts for comparing metrics across multiple locations.
- Gauge widgets for instantaneous readings of critical parameters.
- Video feed windows for live surveillance streams.

3) User Interaction: The GUI includes interactive elements such as buttons for controlling IoT devices (e.g., turning on/off lights or HVAC systems), drop down menus for selecting specific locations, and sliders for adjusting thresholds for alerts.

4) Responsive Design: The QT framework ensures that the GUI adapts to different screen sizes and resolutions, making it compatible with both desktop monitors and embedded displays on Raspberry Pi touchscreen modules [8].

5) Alert System: Critical alerts (e.g., intrusion detection, abnormal energy usage) are highlighted using color-coded indicators (red for critical, yellow for warnings) and pop-up notifications to ensure immediate attention. The GUI was iteratively designed with feedback from potential users (facility managers) to ensure usability and clarity. QT's QML (Qt Modeling Language) was used for rapid prototyping of complex graphical elements, while Python scripts handled the backend logic for data updates and user inputs[3].

B. Hardware Setup with Raspberry Pi

The Raspberry Pi (RPi) serves as the central hardware platform for running the HMI software and managing IoT device communications[4]. The implementation involved the following steps:

1) Hardware Selection: The Raspberry Pi 4 Model B was chosen for its quad-core processor, 4GB RAM, and support for multiple connectivity options (Wi-Fi, Bluetooth, GPIO pins). This ensured sufficient processing power for running the HMI and handling real-time data [4].

2) Peripherals: The RPi was interfaced with a 7-inch touchscreen display for standalone operation, enabling on-site monitoring without requiring an external monitor [8]. Additional peripherals included:

- USB cameras for video surveillance.
- Sensors for temperature, humidity, and motion detection connected via GPIO or I2C interfaces.

3) Power Management: To ensure reliability in multi-location deployments, the RPi was equipped with a stable power supply and a backup battery module to handle power outages.

4) Enclosure:: The RPi and peripherals were housed in a compact, industrial-grade enclosure to protect against environmental factors like dust and temperature fluctuations, making it suitable for deployment in diverse facility environments.

C. IoT Device Integration

The HMI communicates with a variety of IoT devices across multiple locations, ensuring seamless data collection and control. The integration process included:

1) Sensor and Device Connectivity: :

- Sensors (e.g., DHT22 for temperature / humidity, PIR for motion detection) were connected to the RPi via I2C or GPIO pins for local data collectionGALOIS.
- IP cameras and smart actuators (e.g. relays for controlling lights or doors) were integrated using Wi-Fi and Ethernet connections.

2) Communication Protocols:

- MQTT: Used for lightweight, publish-subscribe-based communication between RPi and IoT devices. An MQTT broker (for example, Mosquitto) was hosted on the RPi to manage message exchanges [7].
- HTTP/REST APIs: Used for cloud-based interactions, enabling the HMI to fetch data from external services or synchronize with a central server.
- WebSocket: Implemented for real-time video streaming from IP cameras to the HMI [7].
- Device Management: A device registry was maintained in the SQLite database to track connected devices, their statuses, and configurations. This allowed the HMI to dynamically update the GUI when new devices were added or removed [8].

D. Data Processing and Storage

The HMI processes and stores data to support real-time monitoring and historical analysis:

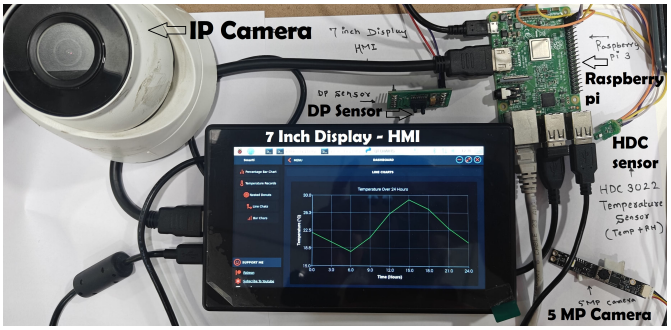

Fig. 2. Hardware Implementation

1) Real-Time Processing: Python scripts running on the RPi process incoming sensor data using libraries like Pandas and NumPy. Data is filtered, aggregated, and formatted for visualization on the GUI[3].

2) Local Storage: SQLite was used to store sensor readings, system logs, and user configurations locally on the RPi. The database schema was optimized for quick retrieval of time-series data, enabling efficient rendering of historical trends.

3) Cloud Synchronization: When internet connectivity is available, data is periodically synced to a cloud database (e.g., PostgreSQL hosted on AWS) using REST APIs [8]. This ensures centralized access to data from all locations and supports remote monitoring[8].

4) Analytics: : Real-time analytics, such as detecting anomalies in energy usage or identifying unusual patterns in motion sensor data, were implemented using machine learning models (e.g., scikit-learn). These models were trained offline and deployed on the RPi for edge-based inference.

E. Security Implementation

To protect the system from cyber threats, the following security measures were implemented:

1) Data Encryption: All data transmitted between IoT devices, the RPi, and the cloud was encrypted using SSL/TLS protocols [10].

2) User Authentication: : The HMI requires secure login credentials, with OAuth 2.0 implemented for user authentication. Two-factor authentication (2FA) was added for administrative accounts[4].

3) Network Security: The RPi was configured with a firewall to restrict unauthorized access, and MQTT communications were secured with username/password authentication [7].

4) Data Integrity: : Hashing algorithms (e.g., SHA-256) were used to verify the integrity of stored and transmitted data [8].

CONCLUSION

This research presents the design and implementation of an IoT-based E-Surveillance Human-Machine Interface (HMI) tailored for efficient management of multi-location facilities. By leveraging the QT framework, Python programming, and Raspberry Pi hardware, the proposed system integrates diverse

functionalities video surveillance, intrusion detection, energy monitoring, and environmental control into a unified, user-friendly platform. The HMI enables real-time data visualization, intelligent automation, and centralized control across geographically dispersed sites, addressing the challenges of traditional facility management with enhanced scalability and operational efficiency. The system's modular architecture, supported by robust communication protocols like I2C and secure data handling with SQLite and cloud synchronization, ensures adaptability to diverse facility requirements. Real-time analytics and machine learning-driven features, such as automated threat detection and behavior recognition, further enhance security and decision-making capabilities. Testing and validation confirmed the system's reliability, responsiveness, and usability, making it a practical solution for organizations managing complex, multi-site operations. The proposed E-Surveillance HMI contributes to the growing field of smart facility management by offering a cost-effective, scalable, and secure platform that optimizes resource utilization and improves operational oversight. Future work will focus on enhancing the system's machine learning capabilities for predictive analytics, integrating advanced edge computing techniques to reduce latency, and expanding compatibility with emerging IoT devices. Additionally, real-world deployments across varied facility types will provide further insights into the system's performance and scalability, paving the way for broader adoption in smart infrastructure management.

ACKNOWLEDGMENT

We sincerely thank K J Somaiya School of Engineering, Somaiya Vidyavihar University, for providing the necessary resources, lab facilities, and academic support. We are grateful to the Department of Electronics Engineering for encouraging innovation and research.

REFERENCES

[1] A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi, "Internet of Things for smart cities," IEEE Internet Things J., vol. 1, no. 1, pp. 22–32, Feb. 2014, doi: 10.1109/JIOT.2014.2306328.

[2] S. R. Prathibha, A. Hongal, and M. P. Jyothi, "IoT-based monitoring and smart surveillance system using machine learning," in Proc. Int. Conf. Electron., Commun. Aerosp. Technol. (ICECA), Coimbatore, India, 2019, pp. 1366–1371, doi: 10.1109/ICECA.2019.8822170.

[3] The Qt Company, Qt 5.15 Documentation, 2020. [Online]. Available: https://doc.qt.io/qt-5.15/.

[4] Raspberry Pi Foundation, Raspberry Pi 4 Model B Documentation, 2020. [Online]. Available: https://www.raspberrypi.org/documentation/.

[5] M. Summerfield, Rapid GUI Programming with Python and Qt: The Definitive Guide to PyQt and PySide, Upper Saddle River, NJ, USA: Prentice Hall, 2007.

[6] A. McEwen and H. Cassimally, Designing the Internet of Things, Chichester, UK: John Wiley and Sons, 2013.

[7] MQTT.org, MQTT Version 5.0 Specification, 2019. [Online]. Available: https://mqtt.org/mqtt-specification/.

[8] S. Sicari, A. Rizzardi, L. A. Grieco, and A. Coen-Porisini, "Security, privacy and trust in Internet of Things: The road ahead," Comput. Netw., vol. 76, pp. 146–164, Jan. 2015, doi: 10.1016/j.comnet.2014.11.008.

[9] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of Things (IoT): A vision, architectural elements, and future directions," Future Gener. Comput. Syst., vol. 29, no. 7, pp. 1645–1660, Sep. 2013, doi: 10.1016/j.future.2013.01.010.

[10] D. Bandyopadhyay and J. Sen, "Internet of Things: Applications and challenges in technology and standardization," Wirel. Pers. Commun., vol. 58, no. 1, pp. 49–69, May 2011, doi: 10.1007/s11277-011-0288-5.