International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

Role Based Authentication System for HMI

Dr. Rajashekar J.S[1], Mohammed Arfaan Shariff [2], Gokul Raj M[3]

[1] *EIE,Dayananda Sagar College of Engineering* Bengaluru,India
rajashekhar-eie@dayanandasagar.edu
[2] *EIE,Dayananda Sagar College of Engineering* Bengaluru,India
Arfaanshaiff@gmail.com
[3] *EIE,Dayananda Sagar College of Engineering* Bengaluru,India
Arfaanshaiff@gmail.com

ABSTRACT—

This paper presents a secure and scalable role-based authentication system for Human-Machine Interfaces (HMIs) in industrial environments using RFID technology and ESP8266 microcontrollers. It addresses the challenge of unauthorized access and improves operational security by assigning hierarchical roles such as Technician, Engineer, and Manager. A cloud-connected backend developed using Flask and Firebase enables real-time validation and data handling, while the Streamlit dashboard provides intuitive visualization. The system is tested for low-latency performance and robust security, demonstrating its viability as an Industry 4.0-compliant access control solution.

Keywords— Role-Based Access Control (RBAC), Industrial Automation, RFID, ESP8266, Flask, Streamlit, IoT Security.

Introduction

This paper presents a novel approach to role-based authentication in industrial Human-Machine Interfaces (HMIs) using ESP8266 microcontrollers and RFID technology. The proposed system addresses the critical challenge of maintaining secure, hierarchical access control in industrial environments while ensuring operational efficiency. By implementing a dynamic role-based authentication mechanism, the system provides granular access control for different user roles such as Line Managers, Technicians, and General Managers. The architecture integrates RFID-based user identification with a cloud-based backend system utilizing Firebase for real-time database operations and Google Sheets for flexible role management. This integration enables seamless updates to role privileges and access control configurations without system downtime. The solution demonstrates significant advantages in terms of cost-effectiveness, scalability, and security compared to traditional authentication methods. Implementation results show successful role differentiation and access control with minimal latency, making it suitable for time-sensitive industrial applications. The system's modular design allows for easy adaptation across various industrial settings and role hierarchies, contributing to enhanced operational security and workflow efficiency. This research provides valuable insights into developing practical, IoT-based security solutions for industrial HMI systems while maintaining robust access control standards.

BACKGROUND

In industrial environments, Human-Machine Interfaces (HMIs) serve as critical tools for controlling and monitoring processes. However, as these systems are increasingly connected through IoT technologies, they face significant challenges related to unauthorized access and operational inefficiencies. In many cases, HMIs are either openly accessible to all personnel or protected by basic password mechanisms, which are often shared or mismanaged. This lack of secure, role-specific access can lead to accidental or malicious misuse, compromising the safety and productivity of the entire system.

Several real-world incidents highlight these concerns. For example, unauthorized adjustments to HMI parameters have led to equipment malfunctions, production losses, and safety hazards. In one instance, an unauthorized operator's actions resulted in the overheating of an industrial machine, causing significant downtime and repair costs. Another common issue involves workers unintentionally tampering with controls beyond their expertise, leading to errors that disrupt workflow.

These challenges underscore the urgent need for a secure and efficient system that not only restricts access but also tailors HMI functions to specific roles. A robust role-based authentication system using advanced technologies like RFID and IoT-enabled devices can address these problems effectively, enhancing security, accountability, and operational transparency in industrial processes.

Moreover, as industrial processes grow more complex, ensuring accountability has become increasingly critical. Without a proper access control mechanism, it becomes challenging to track and audit user interactions with the HMI. In many incidents, the inability to identify who performed specific actions has delayed investigations and corrective measures. These issues emphasize the importance of implementing a system that not only restricts unauthorized access but also maintains detailed logs of user activities, providing clarity and accountability in the event of operational discrepancies.

METHODS

The role-based authentication system integrates RFID technology, ESP8266 microcontrollers, and web frameworks to establish secure HMI access control. The hardware layer utilizes MFRC522 RFID readers for user identification, with ESP8266 NodeMCU managing communication between RFID hardware and server through HTTP protocols. The system architecture incorporates a Flask backend for user authentication and role validation, while a Streamlit frontend provides an intuitive dashboard for role-specific access and monitoring. This integrated approach ensures seamless operation, with RFID data flowing from hardware through authentication to visualization, while supporting future integration with manufacturing systems and analytics capabilities.

Architecture of the Model:

The development of the model for the role-based authentication system for HMI combines RFID technology, ESP8266, Flask framework, and Streamlit, ensuring secure and role-specific access to industrial systems. At the hardware level, the system employs RFID readers (MFRC522) to scan RFID tags, which hold user identification data. The ESP8266 NodeMCU acts as the central microcontroller, facilitating communication between hardware and the server by transmitting data over HTTP protocol. Through this, the scanned RFID tag data is sent to the backend for verification.

The Flask-based backend server handles data processing and role validation. Using a predefined database of registered users and their access levels, the backend authorizes user interactions based on roles such as manager, technician, or operator. For seamless visualization and enhanced user interaction, a Streamlit interface is implemented. Streamlit acts as a frontend layer that displays role-specific access permissions, user logs, and authorized HMI functionalities through an intuitive web-based dashboard. This simplifies real-time monitoring and improves usability.

By integrating RFID hardware with ESP8266 and leveraging HTTP communication protocols, Flask for backend processing, and Streamlit for user-friendly visualization, the system offers an efficient, secure, and scalable solution to manage access control in industrial environments. This architecture enhances security, minimizes unauthorized access, and streamlines rolebased workflows.

Beyond access control, the system significantly improves operational efficiency and reliability in industrial setups. Detailed user logs generated through the Flask backend enable administrators to monitor user interactions and identify potential inefficiencies or security breaches in real-time. The role-specific segregation ensures that sensitive operations remain restricted to authorized personnel, reducing the risk of accidental errors or deliberate misuse.

Furthermore, the system's modular design facilitates future expansion, such as adding advanced analytics or integrating with external systems like manufacturing execution systems (MES) or enterprise resource planning (ERP) tools. Streamlit's interactive visualizations enable administrators to make data-driven decisions by analyzing usage patterns and identifying trends. This blend of modern technologies creates a versatile, secure, and user-centric platform to meet evolving industrial requirements. The figure 1. Shows the Architecture of the model. We will see the working of the model and workflow about the Role based Authentication System for HMI by reading the architure Diagram

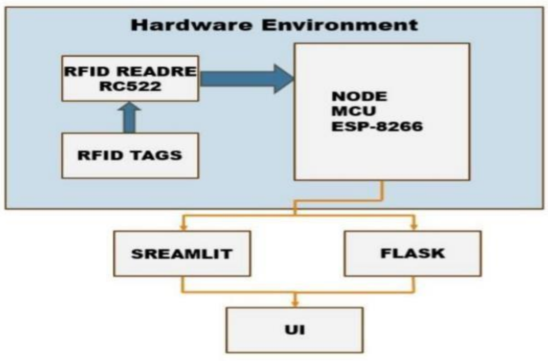


Fig 1-Architeture of Role Based Authentication System

The diagram depicts the architecture for a role-based authentication system utilizing RFID technology, ESP8266 microcontroller, and a software ecosystem that includes Flask for backend processing and Streamlit for frontend visualization. In the hardware environment, RFID readers (RC522) scan user-specific RFID tags containing identification information. The NodeMCU ESP8266 acts as a central processing unit, transmitting the RFID data to a backend server using HTTP protocol over Wi-Fi. This forms the first layer of secure data exchange.

The backend is powered by the Flask framework, which handles data validation, role-based access verification, and communication with a predefined user database. It maps each user's RFID tag to assigned roles such as manager, technician, or operator, granting or restricting access based on specific permissions. This server-side processing ensures only authorized personnel can access critical functionalities within the HMI system, minimizing security risks.

To provide an intuitive user interface, the system uses Streamlit as the front-end layer. Streamlit serves as a real-time dashboard to display role-based permissions, access logs, and other activity data in a user-friendly format. It offers industrial operators and managers a clear view of the system's status while enabling efficient monitoring and control. This layered integration of hardware with software creates a scalable, reliable, and secure role-based access control solution tailored for industrial environments.

Block Diagram of the Model

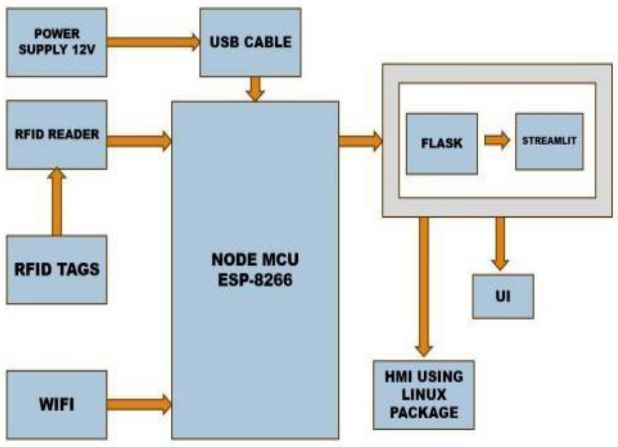
The role-based authentication system for securing a Human-Machine Interface (HMI) application integrates advanced technologies such as RFID, microcontroller-driven back-end processing, and user-friendly front-end frameworks. The core component of this system is the Node MCU ESP-8266, a powerful microcontroller that acts as the central processing unit (CPU), coordinating data transfer and communication between hardware and server. This microcontroller is equipped with wireless capabilities through a WiFi module, ensuring that the system remains connected to a network and is remotely accessible. The ESP-8266 operates using a Flask framework, which is critical for processing authentication requests and managing communication between the front-end interface and the back-end database.

At the heart of the authentication process is RFID technology. RFID tags are issued to users who wish to interact with the industrial HMI system. Each RFID tag contains a unique identifier, which is scanned by an RFID reader, typically an MFRC522 module. The reader sends the data to the ESP-8266, which identifies the user based on the RFID data and matches it against stored credentials and roles, such as operator, technician, or manager. These roles are predefined in a back-end database, often implemented in Flask with secure APIs to ensure privacy and rolespecific access control. Only authorized users whose RFID data aligns with the stored roles are granted access, ensuring that different levels of control are maintained according to the user's security clearance. In this way, the system prevents unauthorized access, significantly enhancing the overall security of the HMI platform. As a result, this solution provides a more reliable and scalable approach to access management compared to traditional password-based systems.

At the heart of the authentication process is RFID technology. RFID tags are issued to users who wish to interact with the industrial HMI system. Each RFID tag contains a unique identifier, which is scanned by an RFID reader, typically an MFRC522 module. The reader sends the data to the ESP-8266, which identifies the user based on the RFID data and matches it against stored credentials and roles, such as operator, technician, or manager. These roles are predefined in a back-end database, often implemented in Flask with secure APIs to ensure privacy and rolespecific access control. Only authorized users whose RFID data aligns with the stored roles are granted access, ensuring that different levels of control are maintained according to the user's security clearance. In this way, the system prevents unauthorized access, significantly enhancing the overall security of the HMI platform. As a result, this solution provides a more reliable and scalable approach to access management compared to traditional password-based systems.

The entire system interface is built with Streamlit, a Python-based framework that allows the creation of interactive and dynamic web applications. Streamlit’s simplicity and responsiveness make it an excellent choice for developing a user-friendly front-end interface where administrators can configure user access, monitor system health, and review activity logs. Streamlit provides an easy-to-navigate dashboard, which is essential for overseeing dayto-day operations. This dashboard can display user logs in real-time, update role-based permissions on the fly, and enable efficient management of security-related actions like modifying roles or revoking access from unauthorized users. This web interface integrates seamlessly with the Flask back-end, enabling smooth, real-time communication between the user interface and the server while maintaining secure data exchanges.

Fig 2-Block Diagram of Role Based Authentication System



The WiFi module on the ESP-8266 enables the system to function over a network, using HTTP protocols to communicate with the database and interface layer. This module allows all operations to occur remotely via the network, and all the server-side components interact with clients via secure web connections. By implementing HTTP for communication, the system is versatile enough to be deployed within larger industrial settings, supporting multiple users across various locations. These communication protocols enable devices on the network to stay synchronized, ensuring system updates, real-time status reporting, and troubleshooting through wireless communication.

This combination of hardware components (Node MCU ESP-8266, RFID reader, tags), software technologies (Flask, Streamlit), and secure communication protocols (HTTP over WiFi) enables the system to meet the security demands of modern industrial environments. The design ensures that the application can manage access dynamically while ensuring only users with proper credentials can access or modify critical operations within the system. The userfriendliness of the interface, coupled with the scalability and reliability of Flask and Streamlit for backend and front-end management, makes this model adaptable for future expansion, new roles, or system updates. Moreover, the system reduces the risks associated with unauthorized access, a critical factor in the safety and security of industrial machines and operations. By securing industrial HMI systems, this architecture not only minimizes the potential for human error and operational hazards but also helps in maintaining precise control and monitoring.

Additionally, this system can be integrated with other security measures like biometric authentication, motion detection, and AI-based monitoring for added security in high-risk applications. The scalability of the RFID-based model allows the system to grow with industrial infrastructure, supporting an ever-increasing number of users while maintaining high levels of security. Furthermore, because the system operates over a standard Wi-Fi network and uses HTTP protocols, it can be easily expanded to accommodate additional security features like cloud integration or integration with other enterprise systems, further enhancing its robustness. The figure 2 illustrates the block diagram of the model.

METHODOLOGY

The role-based authentication system integrates hardware, software, and communication protocols to provide secure and efficient access control tailored to specific user roles. Its functionality spans the interaction of various components, from the initial user authentication to rolespecific application access.

Working of Role Based Authentication System

Initial Authentication: The process begins with RFID (Radio Frequency Identification) technology, where users are issued RFID cards containing unique identifiers (UIDs). When a card is brought near the MFRC522 RFID reader, it scans the UID, ensuring the user’s identity can be recognized. The ESP8266 NodeMCU microcontroller is the central processor responsible for reading the RFID data. Once the UID is retrieved, it is transmitted via SPI (Serial Peripheral Interface) communication between the RFID reader and the ESP8266. The captured UID serves as the first level of identification in the authentication process.

Data Transmission Using HTTP Protocol: Upon reading the UID, the ESP8266 uses its built-in WiFi capabilities to establish a connection with the backend server hosted via Flask. The UID data is formatted and sent to the server using an HTTP POST request. The HTTP protocol ensures smooth and reliable communication between the hardware (ESP8266) and the server, making this wireless setup versatile for remote operations.

Backend Authentication and Role Assignment: The backend server, built using Flask, plays a pivotal role in validating the UID. Once the UID reaches the server, it is cross-verified against a pre-configured database containing registered users and their assigned roles. The database defines roles such as Technician, Control Engineer, or Manager, each with specific access privileges.

If the UID matches a record in the database, Flask identifies the corresponding role and prepares to respond accordingly. The system logs this interaction for security and accountability. If the UID is not recognized, the server denies access, ensuring only authorized users proceed. Figure 5.1.1 shows the code for the Backend and UID mapping using RFID setup.

Frontend: Streamlit Dashboard: Following backend processing, a response is sent back to the ESP8266. For authenticated users, Flask provides role-based data that triggers specific visualizations on the Streamlit dashboard. The dashboard is a user-friendly web application offering unique interfaces for each role:

- Technicians: Access maintenance logs, operational statistics, and system warnings.
- Control Engineers: View system configurations, adjust operational settings, and monitor critical processes.
- Managers: Analyze performance metrics, review reports, and manage access control.

The dashboard dynamically updates to reflect the user’s role, ensuring a tailored experience for each category. This segregation of views enhances usability and protects sensitive information from unauthorized access. Figure 3 &4 shows the code for the frontend and server setup and Hardware model.

Real-Time Interaction and Monitoring: The Arduino IDE plays a key role during development and debugging, particularly through the Serial Monitor, which tracks real-time data transmission between components. The data pipeline—spanning the RFID reader, ESP8266 microcontroller, and Flask backend—is constantly monitored to ensure the system's responsiveness. This real-time interaction helps resolve any operational issues immediately, making the system robust in live environments.

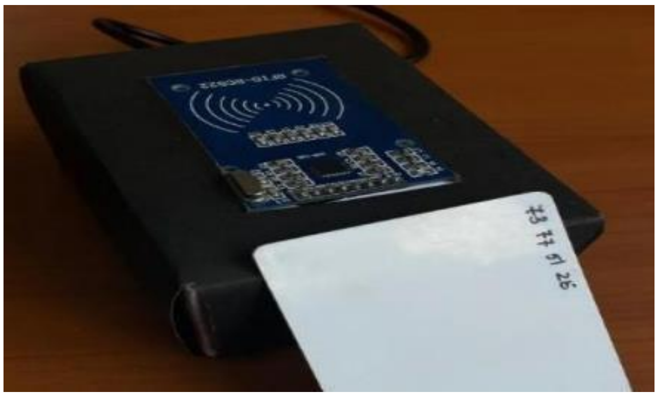
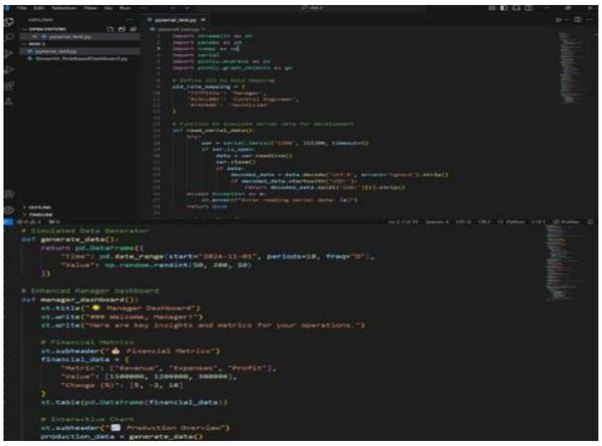


Fig 3-Hardware Model



```
import streamlit as st
import pandas as pd
import numpy as np
import serial
import plotly.express as px
import plotly.graph_objects as go

# Define UID to Role Mapping
uid_role_mapping = {
    "[illegible]": "Manager",
    "[illegible]": "Control Engineer",
    "[illegible]": "Technician"
}

# Function to simulate serial data for development
def read_serial_data():
    try:
        ser = serial.Serial('COM[illegible]', 115200, timeout=1)
        if ser.is_open:
            data = ser.readline()
            ser.close()
            if data:
                decoded_data = data.decode('utf-8', errors='ignore').strip()
                if decoded_data.startswith('UID: '):
                    return decoded_data.split('UID: ')[1].strip()
    except Exception as e:
        st.error(f"Error reading serial data: {e}")
    return None
```

```
# Simulated Data Generator
def generate_data():
    return pd.DataFrame({
        "Time": pd.date_range(start="2024-11-01", periods=10, freq="D"),
        "Value": np.random.randint(50, 200, 10)
    })

# Enhanced Manager Dashboard
def manager_dashboard():
    st.title("[illegible] Manager Dashboard")
    st.write("### Welcome, Manager!")
    st.write("Here are key insights and metrics for your operations.")

    # Financial Metrics
    st.subheader("💰 Financial Metrics")
    financial_data = {
        "Metric": ["Revenue", "Expenses", "Profit"],
        "Value": [1500000, 1200000, 300000],
        "Change (%)": [5, -2, 10]
    }
    st.table(pd.DataFrame(financial_data))

    # Interactive Chart
    st.subheader("📈 Production Overview")
    production_data = generate_data()
```

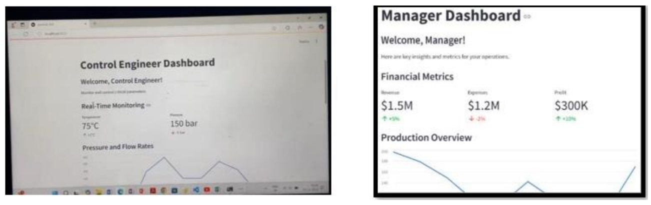
Fig 4-Frontend & Server Setup

RESULTS & DISCUSSION

The results of this role-based authentication system highlight its efficiency and effectiveness in achieving secure and role-specific access to a Human-Machine Interface (HMI) system. The system successfully integrated the key components, including RFID technology, ESP8266 microcontrollers, Flask as a backend framework, and Streamlit for front-end visualizations. Extensive testing confirmed the seamless interaction between hardware and software components, ensuring a reliable and secure operation in an industrial context.

In terms of hardware, the RFID module and ESP8266 microcontroller demonstrated consistent performance in capturing and transmitting unique identification data (UIDs) via the HTTP protocol. The system verified users accurately, based on the data fetched from the RFID tags. Each role-specific dashboard—technician, control engineer, and manager—responded efficiently by granting access to predefined features, adhering to the role hierarchy defined during the system setup. Real-time data transfer between the hardware and backend showed negligible latency, showcasing the capability of the ESP8266 in handling communication efficiently.

The Flask-based backend proved to be robust and versatile, managing role validation and maintaining a database of user credentials securely. By implementing security measures such as HTTP protocols and verification steps, the system prevented unauthorized access while enabling smooth transitions between role-specific permissions. The backend's ability to log activities ensured traceability and accountability for user interactions. Figure-5 shows the results of the fontend after tapping the hardware model the results on the screen with different role based features page on the HN



Control Engineer Dashboard

Welcome, Control Engineer!

Real-Time Monitoring

75°C

150 bar

Pressure and Flow Rates

Manager Dashboard

Welcome, Manager!

Here are key insights and metrics for your operations.

Financial Metrics

$1.5M

$1.2M

$300K

Production Overview

Fig 5- Role Based Dashboard

CONCLUSION

This role-based authentication system addresses the need for secure and efficient access management in industrial settings by leveraging RFID for identification, ESP8266 for communication, Flask for backend management, and Streamlit for user interaction. Tailored access privileges for roles like technicians, engineers, and managers ensure operational security, minimize unauthorized access, and streamline role management. Its scalable architecture, combining hardware and software, supports future enhancements such as advanced security protocols and automation. Developed collaboratively by Mohammed Arfaan Shariff and Gokul Raj, this project demonstrates innovation and teamwork, offering a robust, adaptable framework that transforms traditional systems into intelligent and secure solutions for modern industrial automation.

REFERENCES

[1] Sandhu, R. S., Coyne, E. J., Feinstein, H. L., and Youman, C. E."Role-Based Access Control Models." IEEE Computer, 1996. 29(2): p. 38-47

[2] Ferraiolo, D. F., Kuhn, D. R., and Chandramouli, R. "Role-Based Access Control." Artech House, 2003. ISBN: 978-1-58053-370-5.

[3] Kim, W. T., et al. "Secure access control using role-based authentication in cloud computing environments." Journal of Cloud Computing Advances, 2016. 5(4): p. 123-135.

[4] Zhang, X., Joshi, J. B. D., and Tritschel, B. "An access control framework for wireless sensor networks using role-based authentication." ACM Transactions on Sensor Networks, 2005. 2(2): p. 156-174.

[5] Kulkarni, S., and Tripathi, P. "Role-based authentication using RFID technology for secure systems." International Journal of Engineering Research and Technology, 2013. 2(10): p. 2171-2177

[6] Wu, X., et al. "Enhancing security in IoT-based applications with a role-based authentication scheme." Sensors, 2018. 18(10): p. 3432

[7] Thomas, R. K., and Sandhu, R. S. "Task-based authorization controls (TBAC): A family of models for active and enterprise-oriented authorization management." Proceedings of the IFIP WG11.3 Working Conference on Database Security, 1997. p. 166-181.

[8] Park, J. S., and Sandhu, R. S. "Smartcard-based implementation of role-based authentication for web services." Computer Communications Journal, 2000. 23(17): p. 1637- 1652.

[9] Gupta, P., et al. "Role-based access control in industrial automation using IoT frameworks." International Journal of Industrial Engineering Research, 2020. 9(4): p. 264275.

[10] Samarati, P., and de Vimercati, S. C. "Access control: Policies, models, and mechanisms." Foundations of Security Analysis and Design, 2001. 7(4): p. 137-196.