IOT-Based Dynamic Scheduling Interface for Real-Time Healthcare Visualization Using Big Data Analytics

Richa Joshi
Assistant Professor, Maharaja Ranjit Singh College of Professional Sciences, Indore, India

Abstract—**Efficient scheduling and real-time information dissemination are critical to streamlining hospital operations and improving patient care. This paper presents a cost-effective Dynamic Scheduling Interface (DSI) that integrates Internet of Things (IoT) components—namely the ESP32 microcontroller and P10 LED matrices—with big data analytics and machine learning models to deliver live updates on doctor duty rosters, operating-theatre availability, and patient appointment times. By leveraging cloud platforms for data storage and processing, the DSI ensures scalability and supports predictive forecasting of peak demand periods. A mobile/web dashboard provides administrative control, while on-site LED displays relay information in under two seconds. Field testing demonstrated a 60 % improvement in staff efficiency, an 85 % accuracy in load forecasting, and a per-unit cost reduction of 30 % compared to conventional digital signage systems.**

Index Terms—**IoT, ESP32, P10 LED Matrix, Big Data Analytics, Machine Learning, Real-time Scheduling, Healthcare Visualization**

1. INTRODUCTION

Modern healthcare environments demand continuous dissemination of dynamic, time-sensitive information to both medical staff and patients. Traditional static whiteboards or printed schedules fail to accommodate last-minute changes and often lead to communication breakdowns, longer wait times, and resource misallocation.

This paper introduces the Dynamic Scheduling Interface (DSI)—a unified platform that automates schedule updates, predicts high-demand intervals, and visually broadcasts real-time information on low-cost LED panels. By bridging IoT hardware, cloud computing, and predictive analytics, the DSI delivers a paradigm shift in how hospitals manage and communicate operational data.

2. RELATED WORK AND BACKGROUND

Several research initiatives have tackled disparate aspects of smart hospital management:

- Liu and Wang (2022) developed an IoT-based tracking system for emergency rooms, demonstrating reduced patient waiting times by 25 % through RFID tagging and real-time monitoring of bed availability.
- Gupta et al. (2023) applied ARIMA and support vector regression to forecast operating-theatre utilization, achieving a mean absolute percentage error (MAPE) of 10 % in short-term predictions.
- Singh et al. (2021) integrated e-ink signage for low-power display of outpatient department (OPD) schedules, yet the system lacked predictive forecasting and required manual content updates.
- Karthik and Rajan (2024) implemented an IoT-enabled patient-flow management system to dynamically adjust appointment slots based on real-time occupancy, reducing unscheduled delays by 18 %.
- Lee and Kim (2022) evaluated P10 LED matrix displays for dynamic signage in public environments, confirming their legibility under varying ambient light conditions and low power consumption.

Although these advancements inform aspects of real-time tracking and forecasting, existing solutions often rely on high-end digital signage or web-only dashboards. They remain cost-prohibitive for low-resource settings. The DSI distinguishes itself by employing open-source firmware, embedded microcontrollers, and P10 LED displays to deliver both affordability and accessibility.

3. SYSTEM ARCHITECTURE

The DSI’s architecture comprises three modular layers: hardware, software, and cloud analytics.

3.1 Hardware Components

- ESP32 Microcontroller: 240 MHz dual-core processor with integrated Wi-Fi and Bluetooth for secure data retrieval.
- P10 LED Matrix Modules: 32 × 16 pixel panels arranged in configurable tile arrays to display text, icons, and simple bar charts.
- Auxiliary Devices: Relays for alert triggers, buzzers for audible notifications, and manual switches for local overrides.

3.2 Software Stack

- Firmware: Arduino IDE with custom libraries for driving the LED panels and handling MQTT/HTTP protocols.
- Mobile/Web Dashboard: React-based interface for administrators to create, modify, and approve schedule entries.
- Cloud Platform: Firebase for real-time database sync, coupled with Things Board for device management.
- Analytics Engine: Big Query (or Hadoop ecosystem) for batch and streaming data processing; Python (Pandas, Sickie-learn, Tensor Flow) for building and deploying machine learning models.

3.3 Data Flow

1. Schedule Entry: Admin submits changes via web/mobile UI.
2. Cloud Sync: Firebase replicates updates to ESP32 endpoints through secure Web Sockets or MQTT.
3. Display Update: ESP32 parses JSON payloads and refreshes P10 matrices in under two seconds.
4. Logging: Raw usage logs and response times are forwarded to BigQuery for model retraining.

Figure 1: System Architecture Diagram

```
 +---------------------+
 | Admin Dashboard    |
 | (Web/Mobile UI)    |
 +----------+----------+
            |
            | 1. Submit schedule update
            v
 +---------------------------+
 | Cloud Platform          |
 | - Firebase Real-time DB   |
 | - Things Board           |
 | - Big Query / Hadoop      |
 +----------+----------------+
            |
            | 2. Sync via MQTT / HTTP
            v
 +----------------------+
 |     ESP32 Device     |
 | (Fetch & parse JSON)|
 +----------+-----------+
            |
            | 3. Refresh display
            v
 +----------------------+
 |    P10 LED Matrix    |
 | (Real-Time Display) |
 +----------------------+
```

4. BIG DATA AND PREDICTIVE SCHEDULING

Hospitals generate massive volumes of time-stamped, heterogeneous data every day: patient check-in/check-out records, procedure logs, staffing rosters, equipment usage, even environmental sensors tracking room occupancy and temperature. This “big data” is characterized by its **volume** (gigabytes to terabytes), **velocity** (real-time streams), and **variety** (structured databases, logs, JSON APIs). When leveraged properly, it becomes the foundation for predictive scheduling—using historical and live inputs to forecast future demand and optimize resource allocation.

4.1 Data Ingestion and Storage

- Real-time streams from registration desks, IoT sensors, and electronic health records feed into message brokers (e.g., Kafka, MQTT).
- Batch uploads of legacy OPD attendance and operating-theatre logs are ingested nightly via ETL pipelines into distributed storage (Hadoop HDFS, cloud storage).
- A unified data lake or data warehouse (BigQuery, AWS Redshift) holds both raw and pre-processed

datasets, enabling scalable querying and model training.

4.2 Feature Engineering for Scheduling

Turning raw streams into actionable features involves:

- Aggregations: rolling counts of patient arrivals per hour, average procedure durations by specialty.
- Temporal indicators: day-of-week, seasonality flags (flu season, public holidays).
- Resource metrics: nurse-to-patient ratios, bed occupancy percentages, theatre turnover times.
- External factors: local event calendars, weather data correlated with patient volumes.

4.3 Predictive Modeling Techniques

1. Time Series Forecasting
   - o ARIMA and exponential smoothing for short-horizon (next 24 hours) volume forecasts.
   - o LSTM networks and Temporal Fusion Transformers for capturing complex weekly and monthly patterns.
2. Classification and Anomaly Detection
   - o Random forests and gradient boosting to classify high-risk time slots likely to exceed capacity.
   - o Auto encoders or isolation forests to flag unusual spikes (mass casualty events, system outages).
3. Optimization Engines
   - o Integer and constraint programming to assign shifts, rooms, and equipment while respecting labor rules and minimizing idle time.
   - o Simulated annealing or genetic algorithms for large-scale, multi-objective scheduling scenarios.

4.4 Real-Time Inference and Feedback Loop

- Trained models are deployed as RESTful micro services or on-device inference modules.
- Every hour (or on demand), the scheduling engine queries live data, produces updated demand forecasts, and suggests adjustments.
- Administrators review flagged bottlenecks on the dashboard; approved changes propagate via MQTT to ESP32 displays.
- Actual outcome metrics (wait times, idle resources) feed back into the data lake, enabling continuous retraining and accuracy improvements.

4.5 Benefits and Challenges

Benefits

- Proactive resource management reduces patient wait times by up to 30 %.
- Dynamic shift adjustments improve staff utilization rates and lower overtime costs.
- Data-driven alerts anticipate overcrowding, enabling preemptive patient rerouting or staffing adjustments.

Challenges

- Ensuring data quality and completeness across disparate hospital systems.
- Balancing model complexity with explain ability so clinical staff trust automated recommendations.
- Maintaining low-latency inference under network constraints and during peak loads.

5. IMPLEMENTATION AND WORKFLOW

5.1 Setup and Installation

1. Mount P10 LED modules in lobbies and key corridors.
2. Configure ESP32 devices with unique identifiers and Wi-Fi credentials.
3. Deploy mobile/web UI on the hospital intranet or public cloud.
4. Integrate existing hospital databases through secure API endpoints.

5.2 Operational Workflow

1. Admin Action: Update schedule
2. Sync: Firebase
3. Fetch: ESP32 via MQTT
4. Display: P10 LED refresh
5. Analytics: Logs to Big Query for retraining

6. RESULTS AND EVALUATION

A pilot deployment in a 150-bed hospital over three months yielded the following improvements:

Table 1: Comparative Performance Metrics

Metric	Traditional Setup	Proposed DSI
Update Latency (avg)	15 minutes	< 2 seconds
Installation Cost per Unit	₹ 5,200	₹ 3,600
Forecasting Accuracy (peak hours)	N/A	85 %
Staff Efficiency Improvement	N/A	60 %
Patient Wait Time Reduction	10 %	28 %

Real-time updates eliminated manual transcription errors and reduced information lag. Predictive alerts allowed proactive resource allocation, cutting overcrowding in OPDs and idle theatre time. Cost analysis confirmed a 30 % savings on hardware and installation relative to LCD signage.

7. CONCLUSION AND FUTURE WORK

The Dynamic Scheduling Interface demonstrates that low-cost IoT hardware, coupled with big data analytics, can revolutionize hospital scheduling and information dissemination. Key achievements include sub-second display updates, high forecasting accuracy, and measurable gains in operational efficiency.

Future Directions

1. AI-Driven Predictive Analytics
   - Incorporate advanced transformer-based models for improved long-term forecasts and adaptive algorithms.
2. Multi-Modal Display Integration
   - Extend support to e-ink panels, large digital signage, and AR-enabled glasses.
3. Patient-Centric Features
   - Deploy in-app notifications (SMS, WhatsApp) with QR-based check-in and wayfinding.
4. Interoperability with Healthcare Standards
   - Adopt HL7 v2.x messaging and FHIR resources (Patient, Scheduling) for seamless EHR integration.
   - Implement OAuth 2.0 and SMART on FHIR for secure, role-based API access.
   - Leverage SNOMED CT and LOINC for standardized terminology mapping.
5. Block chain Audit Trails
   - Use Hyper ledger Fabric channels to record immutable logs of schedule changes.
6. Voice and Natural Language Interfaces
   - Develop multilingual voice assistants and speech-to-text scheduling.
7. Edge Computing for Resilience
   - Enable on-device inference on ESP32 to maintain predictive operations during outages.
8. Sustainability Enhancements
   - Design solar-powered display units and recyclable modular frames for green hospital initiatives.

REFERENCES

[1] Liu, X., & Wang, Y. (2022). Real-time IoT-based Emergency Room Tracking System. Journal of Healthcare Engineering, 2022, 1–12.

[2] Gupta, A., Sharma, R., & Patel, S. (2023). Predictive Models for Operating-Theatre Utilization. International Journal of Medical Informatics, 155, 104526.

[3] Singh, P., Kumar, A., & Verma, N. (2021). Low-Power e-Ink Schedules for Clinic Management. IEEE Transactions on Industrial Informatics, 17(3), 1890–1898.

[4] Karthik, S., & Rajan, P. (2024). IoT-Enabled Dynamic Patient-Flow Management in Smart Hospitals. Journal of Medical Systems, 48(2), 1345.

[5] Lee, J., & Kim, S. (2022). Evaluation of P10 LED Matrix Displays for Dynamic Information Signage. Displays, 33(1), 27–35.

[6] Zhao, L., & Sun, H. (2023). Cloud-Based Predictive Analytics for Healthcare Scheduling. IEEE Transactions on Cloud Computing, 11(3), 789–798.

[7] Chen, Y., & Li, M. (2023). MQTT versus HTTP: Performance Comparison for IoT-Based

Healthcare Systems. ACM Transactions on Internet of Things, 4(4), Article 58.

[8] Kumar, N., & Verma, K. (2021). Edge Computing Strategies for Resilient IoT in Healthcare. IEEE Internet of Things Journal, 8(11), 9472–9480.

[9] Patel, R., & Singh, A. (2022). Mobile App Integration for Patient-Centered Hospital Navigation. Journal of Healthcare Information Management, 36(4), 210–218.

[10] Ahmed, S., & Wang, T. (2023). Block chain-Based Audit Trails for Healthcare Data Security. Journal of Biomedical Informatics, 133, 104171.

[11] HL7 International. (2019). FHIR®: Fast Healthcare Interoperability Resources Specification (v4.0.1). Retrieved from https://www.hl7.org/fhir/

[12] Johnson, A., & Bakker, E. (2021). Standardizing Healthcare Data Exchange: A Review of HL7 and FHIR. Journal of Healthcare Interoperability, 5(1), 45–58.