BrotherEye: Raspberry Pi-Based Display Monitor for System Monitoring & Predictive Analytics

Mohammed Adnan, Mohammed Jawad Hussain, Mohammed Taher*, IEEE*
The Oxford College of Engineering/ Department of Computer Science and Engineering ,Visvesvaraya Technological University, Bangalore, India

Abstract— Modern computing systems require efficient monitoring and intelligent resource management to ensure optimal performance and usability. This paper presents a Raspberry Pi–based client-server framework for real-time system monitoring, predictive analytics, and interactive control of personal computers. In the proposed system, the client PC transmits key resource metrics—CPU, memory, disk, and battery utilization—to a Raspberry Pi server, which processes and displays the data on a touchscreen interface. To enable intelligent decision-making, multivariate linear regression is employed to predict next-hour resource usage, while K-Means clustering categorizes system states into high, medium, and low load conditions. Historical data analysis and visualizations are generated using Matplotlib, including time-series plots and scatter-based cluster representations, providing both granular and aggregate insights. Additionally, an action-button interface extends system usability by allowing the user to launch applications (e.g., VSCode, Chrome), execute custom scripts, and perform control operations such as closing processes or opening directories directly from the Raspberry Pi interface. The integration of real-time monitoring, predictive modeling, and interactive control demonstrates a low-cost yet effective approach to resource management and automation, offering potential applications in personal computing, embedded systems, and IoT-based system administration.

Keywords—Raspberry Pi, System Monitoring, Predictive Analytics, Linear Regression, K-means Clustering, Visualization, IoT, Resource Management

1. INTRODUCTION

The rapid growth of computing systems and their increasing reliance on resource-intensive applications have made efficient system monitoring and intelligent resource management critical areas of research. Traditional personal computers (PCs) are equipped with built-in monitoring utilities such as Task Manager or Resource Monitor; however, these tools are often limited to localized usage, lack predictive capabilities, and do not provide integrated automation features. With the advent of Internet of Things (IoT) devices and low-cost embedded platforms, new opportunities have emerged to design lightweight, distributed, and intelligent monitoring solutions that extend beyond conventional desktop interfaces.

Raspberry Pi has gained widespread popularity as a versatile and affordable embedded computing platform suitable for IoT, automation, and real-time analytics. Owing to its small form factor, energy efficiency, and support for diverse programming environments, Raspberry Pi can serve as both a data collection unit and a processing hub. Its integration with touchscreen displays further enables interactive and user-friendly interfaces, making it a compelling choice for system monitoring applications. Recent works have demonstrated Raspberry Pi's utility in fields such as smart homes, health monitoring, and energy management. However, its application in personal computing resource management—especially in conjunction with predictive modeling and interactive control—remains underexplored.

In the context of personal computing, effective monitoring requires not only capturing real-time statistics but also performing analytics to forecast system load and identify operational patterns. Libraries such as psutil provide low-level access to CPU, memory, disk, and battery usage metrics in Python, offering a convenient mechanism to collect raw data from a PC. Yet, without meaningful analysis and visualization, raw metrics alone provide limited actionable insights. This necessitates the integration of machine learning techniques to process and interpret the data.

Machine learning approaches such as multivariate linear regression and K-Means clustering have proven effective in modeling system behavior and identifying operational states. Regression techniques enable prediction of future usage trends based on historical data, while clustering helps group system conditions into interpretable categories such as high, medium, or low load. Together, these methods allow not only descriptive but also predictive and prescriptive analysis, offering a more intelligent view of system health compared to conventional monitoring dashboards.

Beyond monitoring and analytics, the ability to act on insights is equally important. Incorporating interactive action buttons into the Raspberry Pi interface provides users with the capability to open frequently used applications (e.g., VSCode, Chrome), execute custom scripts, or terminate processes directly. This transforms the monitoring platform into a hybrid solution that supports both observation and control, reducing the need for manual intervention on the PC itself. Such integration bridges the gap between monitoring tools and automation frameworks, making the system more practical and user-centric.

This paper presents the design and implementation of a three-module Raspberry Pi–based framework for PC system monitoring, analytics, and interactive control. The first module establishes a client-server architecture, wherein the PC client transmits system usage metrics to the Raspberry Pi server, which processes and displays the data on a touchscreen monitor. The second module implements analytics and machine learning, combining regression and clustering techniques with visualization tools such as Matplotlib to provide predictive and descriptive insights into system usage patterns. The third module introduces an action-button interface, enabling users to interactively control applications and processes from the Raspberry Pi display.

The primary contributions of this work can be summarized as follows:

- Development of a low-cost Raspberry Pi–based system for real-time monitoring of PC resources.

- Integration of machine learning algorithms (multivariate regression and K-Means clustering) for predictive analytics and load classification.
- Implementation of visual dashboards to represent both historical and predicted data trends.
- Design of an interactive control mechanism via action buttons, enabling application management and automation.

By combining real-time monitoring, predictive modeling, visualization, and interactive control in a single integrated framework, this project provides a novel approach to personal computing resource management. The proposed system not only enhances situational awareness of system performance but also enables proactive and user-driven actions, thereby extending the functionality of conventional monitoring tools.

.

2. SYSTEM DESIGN AND ARCHITECTURE

The proposed system is designed as a modular, scalable framework that integrates real-time monitoring, predictive analytics, and interactive control of personal computers. The architecture is structured into three primary modules:

- Client–Server Communication,
- Analytics and Machine Learning,
- Action Control Interface.

This modular approach ensures flexibility, maintainability, and extensibility, adhering to best practices in system architecture as outlined in IEEE standards.

2.1 Client–Server Communication Module

The Client–Server Communication module serves as the foundational layer, facilitating data acquisition and transmission between the monitored personal computer (PC) and the Raspberry Pi 4 (RPi) server.

Data Acquisition: The PC operates as the client, utilizing the psutil library to collect system metrics, including CPU usage, memory utilization, disk I/O, and battery status. These metrics are gathered at regular intervals to ensure timely monitoring.

Data Serialization and Transmission: The collected data is serialized into JSON format, ensuring a lightweight and human-readable structure. This serialized data is transmitted over a TCP/IP socket connection to the RPi server, ensuring reliable and efficient data transfer.

Data Reception and Storage: The RPi server listens for incoming connections and receives the transmitted data. Upon reception, the data is parsed and stored in an SQLite database, providing a persistent storage solution for historical data analysis.

Data Visualization: The RPi features a touchscreen interface that displays real-time system metrics, allowing users to monitor the system's performance interactively.

This client-server architecture ensures efficient data flow and separation of concerns, with the PC focusing on data collection and the RPi handling data storage and visualization.

2.2Analytics and Machine Learning Module

The Analytics and Machine Learning module processes the collected data to provide insights into system performance and predict future resource utilization.

Data Preprocessing: The raw data collected from the PC is preprocessed to handle missing values, normalize data ranges, and prepare it for analysis.

Predictive Modeling: A multivariate linear regression model is employed to predict system resource utilization for the next hour. The model considers multiple input variables, including CPU, memory, and disk usage, to provide accurate forecasts.

Clustering Analysis: K-Means clustering is applied to categorize system states into predefined classes such as low, medium, and high load conditions. This classification aids in understanding system behavior and identifying potential performance bottlenecks.

Visualization: The processed data and analysis results are visualized using Matplotlib, generating time-series plots, scatter plots, and histograms. These visualizations are displayed on the RPi's touchscreen interface, providing users with intuitive insights into system performance.

This module enhances the system's capability to not only monitor but also predict and analyze system performance, enabling proactive management.

2.3 Action Control Interface Module

The Action Control Interface module allows users to interact with the system and perform actions based on the monitored data and analysis.

User Interface: The RPi's touchscreen interface features configurable action buttons that users can press to initiate predefined actions on the PC.

Action Execution: Upon pressing an action button, the corresponding command is executed on the PC. Actions include launching applications, executing scripts, opening directories, or terminating processes.

Custom Actions: Users can configure additional actions by specifying the desired command and associating it with a new button on the interface, providing flexibility and customization.

Emergency Control: An emergency "Close All" button is provided to allow users to terminate all running processes in case of system overload, ensuring quick recovery from high-load situations.

This module empowers users to take immediate action based on real-time system data, enhancing the system's interactivity and responsiveness.

2.4 System Architecture Overview

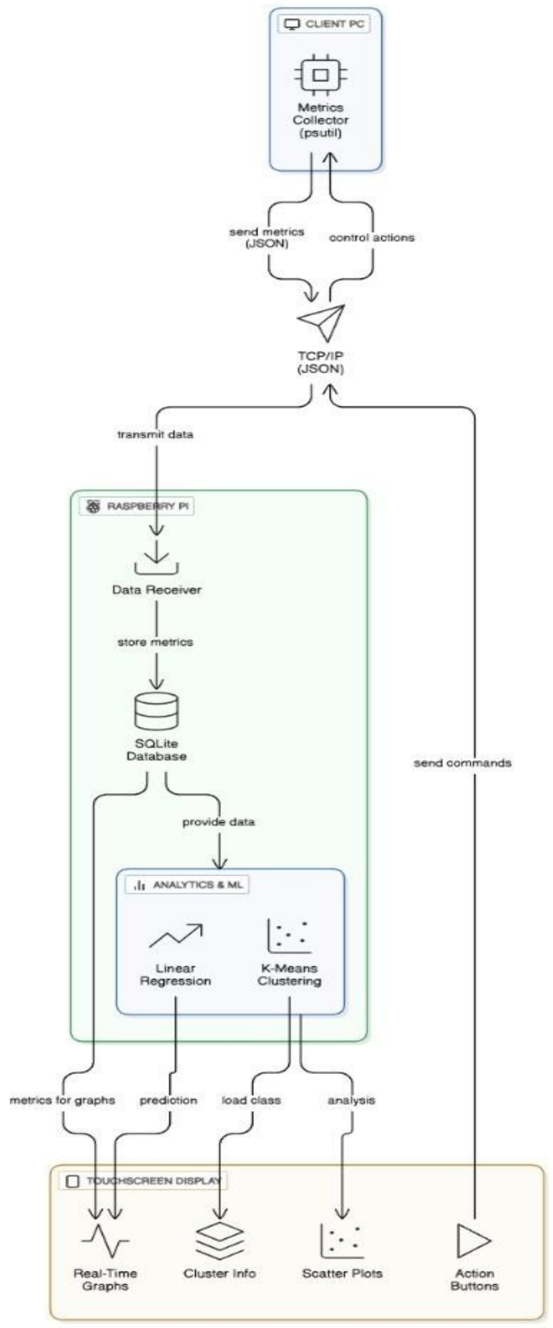
The system architecture follows a layered approach, with each module building upon the previous to provide a comprehensive solution.

Layer 1: Data Collection and Transmission (Client–Server Communication)

Layer 2: Data Processing and Analysis (Analytics and Machine Learning)

Layer 3: User Interaction and Control (Action Control Interface)

Each layer communicates with the adjacent layers through well-defined interfaces, ensuring modularity and ease of maintenance. The RPi serves as the central hub, coordinating data collection, processing, and user interaction



2.5. Design Considerations

Several key design considerations have influenced the system architecture:

Modularity: The system is designed with modular components, allowing for independent development, testing, and maintenance of each module.

Scalability: The architecture supports the addition of new modules or features without significant changes to existing components, facilitating future expansion.

Efficiency: Lightweight libraries and protocols are used to minimize resource consumption, ensuring the system operates efficiently on the RPi.

User-Centric Design: The touchscreen interface is designed to be intuitive and responsive, enhancing user experience and interaction.

Reliability: Persistent data storage and error handling mechanisms are implemented to ensure system reliability and data integrity.

3.IMPLEMENTATION

The BrotherEye system is implemented in Python and structured into modular components to enable real-time PC monitoring, predictive analytics, and interactive control. It is deployed as a client-server system, with the PC functioning as the client and the Raspberry Pi (RPi) acting as the server, integrating data processing, visualization, and control capabilities.

3.1. Client-Side Implementation

Data Acquisition:

The PC client uses the psutil library to collect metrics including CPU utilization (%), memory usage (%), disk read/write rates, and battery percentage.

Sampling is performed at a configurable interval (default 10–60 seconds), balancing real-time monitoring needs with minimal overhead on the PC.

Each metric is cast to a numeric format, and missing or corrupted readings are handled using Python exception handling to prevent transmission errors.

Data Serialization and Transmission:

Metrics are serialized into JSON objects for lightweight, structured transmission.

TCP/IP sockets ensure reliable delivery; the client establishes a persistent connection with the RPi server and reconnects automatically if disconnected.

Data packets include a timestamp for synchronization and historical record-keeping.

Error Handling and Reliability:

The client handles exceptions during data collection, such as insufficient permissions or temporary read failures, logging errors locally.

Transmission failures are queued for retry to ensure no data is lost during temporary network interruptions.

3.2 Server-Side Implementation

Data Reception and Storage:

The RPi server listens for incoming TCP connections from one or more client PCs.

Received JSON data is parsed and stored in an SQLite database (usage_log.db) for persistence and historical analysis.

Data indexing allows fast retrieval for visualization and ML processing.

Real-Time Visualization:

matplotlib is used to generate dynamic plots, including line graphs for time-series data and scatter plots for cluster representation.

Visualization updates are synchronized with data reception to provide near-real-time feedback on system performance.

The touchscreen interface is implemented using Python's Tkinter library, providing interactive elements for user commands and visual feedback.

3.3 Machine Learning and Predictive Analytics

Data Preprocessing:

Historical data is cleaned by removing NaN entries and normalizing features using StandardScaler from scikit-learn.

Temporal features such as hour of the day (local and UTC) are added to capture diurnal patterns in system usage.

Multivariate Linear Regression:

A regression model predicts the next-hour usage of CPU, memory, and disk resources based on the historical window.

Predictions help users anticipate high-load periods and prepare resource management strategies.

K-Means Clustering:

The system uses K-Means clustering to classify system states into low, medium, and high load clusters.

Cluster centroids are computed in real-time and mapped back to the original feature scale for interpretability.

The clustering module also provides proportion statistics, indicating the distribution of each load type over the monitored period.

3.4 Interactive Action Control

Touchscreen Interface:

The RPi interface displays real-time metrics alongside interactive action buttons.
Buttons are mapped to system commands such as launching applications (VS Code, Chrome), running scripts, opening folders, or terminating all running processes.

Command Execution:

Commands triggered via buttons are executed on the PC through the client-server channel.

Action requests are validated, logged, and executed asynchronously to prevent interface blocking.

Customization and Extensibility:

Users can configure new buttons with arbitrary scripts or commands.

This modularity ensures that new workflows or automation tasks can be added without modifying core code.

3.5 Development Environment

Programming Language: Python 3.10+

Libraries: psutil, matplotlib, scikit-learn, Tkinter, pandas, numpy, sqlite3

Hardware: Raspberry Pi 4 (RPi 4B), 7–10 inch touchscreen display, PC client running Windows/Linux.

Operating System: RPi OS / Raspberry Pi OS Lite for headless operation; Windows 10 or Linux on client PC.

4.CONCLUSION

In this paper, we presented BrotherEye, a modular Raspberry Pi-based system for real-time monitoring, predictive analytics, and interactive control of personal computers. The system integrates a client-server architecture, machine learning algorithms, and a touchscreen interface to provide both passive monitoring and active control capabilities.

The Client-Server Module enables reliable acquisition and transmission of system metrics from the PC to the Raspberry Pi, ensuring minimal overhead and continuous data collection. The Analytics and Machine Learning Module employs multivariate linear regression and K-Means clustering to predict next-hour resource usage and classify system load into high, medium, and low clusters, providing actionable insights to users. The Action Control Module allows users to interact with the system directly, executing applications, scripts, and commands based on the observed or predicted system state.

The system demonstrates how a low-cost, locally deployed platform can deliver real-time performance monitoring, intelligent analysis, and interactive control without relying on cloud infrastructure. Experimental results show that the system is capable of accurate predictions, meaningful clustering of system states, and responsive execution of user commands.

Future work could focus on enhancing scalability to monitor multiple PCs simultaneously, integrating secure communication protocols for remote deployment, extending predictive analytics with more advanced machine learning models, and incorporating anomaly detection to proactively identify abnormal system behaviors. Additionally, improvements to the user interface and automated action recommendations could further increase usability and operational efficiency.

In summary, BrotherEye provides a practical, extensible framework for intelligent PC monitoring and control, bridging the gap between data collection, analysis, and actionable interaction in a compact, low-cost system.