Original Research ***https://doi.org/10.5281/zenodo.21373108***

Volume 8, Issue 1, pp 74-92, June 2026

ACHIEVERS JOURNAL OF SCIENTIFIC RESEARCH

Open Access Publications of Achievers University, Owo

Available Online at *www.achieversjournalofscience.org*

IoT-Enabled Smart Grid Monitoring and Control: A Comparative Study of Arduino-Based Automatic Source Selection and MATLAB GUI Simulation

A.S. Oluwole[1*], M.O. Omojoyegbe[1,2], O. Akinsanmi[1], and O.J. Famoriji[2]

[1]Department of Electrical and Electronics Engineering, Federal University Oye Ekiti, Ekiti State, Nigeria.
[2]Department of Electrical and Information Engineering, Achievers University, Km 1, Idasen-Ute Road, Owo, Ondo State, Nigeria.

Corresponding Author Author's Email Address: asoluwole@gmail.com

Submitted: January 15, 2026; **Revised:** May 5, 2026; **Accepted:** May 29, 2026; **Published:** June 30, 2026

Abstract

The rapid evolution of Smart Grid technologies has been accompanied by the integration of Internet of Things (IoT) platforms, enabling real-time monitoring, intelligent decision-making, and autonomous control of power systems. Such advancements are particularly critical in hybrid energy networks where distributed resources such as photovoltaic (PV) systems, inverters, batteries, and conventional generators must be coordinated to ensure efficiency, reliability, and resilience. However, one of the key challenges lies in developing low-cost yet effective platforms that can both demonstrate real-world switching and protection mechanisms and provide flexible environments for simulation, analysis, and education. This paper presents a comparative analysis of two complementary implementations for Smart Grid monitoring and control: an Arduino-based automatic source selector, and a MATLAB GUI-based power system simulator. The Arduino prototype demonstrates practical hardware-level switching between grid, generator, and inverter sources, with integrated protection mechanisms against over-voltage and undervoltage conditions, and real-time monitoring through LCD output. In contrast, the MATLAB GUI simulator provides a software-based environment for dynamic visualization of system states, parameter manipulation through interactive sliders and toggles, and graphical representation of AC/DC waveforms and battery charge profiles. Comparative evaluation highlights that while the Arduino system offers real-world applicability, deployability, and IoT-ready integration with sensors, the MATLAB simulator excels in flexibility, scalability, and pedagogical value for education and research. The findings reveal that combining hardware-oriented prototyping with software-centric simulation provides a cost-effective and holistic approach for smart grid innovation, validation, and capacity building, particularly in developing contexts where both operational reliability and academic training are equally critical.

Keywords: Smart Grids, IoT, Arduino, MATLAB, Source Switching.

1.0 Introduction

The growing global demand for reliable, efficient, and sustainable electrical energy has accelerated the transformation of conventional power systems into Smart Grids (SGs). Traditional electricity grids, which are largely centralized and characterized by unidirectional power flow, are increasingly unable to meet the requirements of modern energy systems driven by distributed generation, renewable energy integration, and dynamic consumption patterns. These limitations have led to the emergence of the smart grid paradigm, which integrates advanced information and communication technologies (ICT) into power systems to enable intelligent, efficient, and reliable energy management. Smart grids represent a significant evolution of legacy power infrastructure by enabling real-time monitoring, automated control, and bidirectional communication between utilities and end-users. This transformation enhances operational efficiency, improves power quality, and supports the seamless integration of renewable energy sources such as solar and wind (Wang *et al*., 2018). Furthermore, Smart Grids are widely recognized as cyber-physical systems that combine electrical networks with digital communication frameworks to achieve greater flexibility, resilience, and sustainability in power delivery systems (Tuballa & Abundo, 2016). These capabilities are essential for addressing challenges such as energy losses, grid instability, and the increasing complexity of modern electricity networks.

A key enabler of Smart Grid evolution is the Internet of Things (IoT), which facilitates the interconnection of smart devices, sensors, and control systems within the energy ecosystem. IoT provides a framework for real-time data acquisition, distributed intelligence, and autonomous decision-making, thereby improving grid observability, responsiveness, and efficiency. Through interconnected devices such as smart meters, intelligent gateways, and sensors, IoT enables bidirectional communication of both information and energy, which is fundamental to modern Smart Grid operation (Kakran & Chanana, 2018; Wang *et al*., 2018). IoT-based architectures further support critical applications such as demand-side management, fault detection, predictive maintenance, and energy optimization (Saleem *et al*., 2019).

Recent studies have demonstrated practical implementations of IoT-enabled energy systems in areas such as smart lighting control using motion detection and sensor-based automation, which significantly improve energy efficiency and user adaptability (Makanju *et al*., 2025). In addition, intelligent embedded systems have been applied in smart power source switching and monitoring solutions, enhancing system reliability and ensuring continuous power supply (Omojoyegbe *et al*., 2025a). Furthermore, advanced simulation models for hybrid power systems with integrated fault protection and battery management have been developed to improve system design, performance evaluation, and operational safety (Omojoyegbe *et al*., 2025b). Collectively, these developments highlight the increasing role of embedded intelligence, IoT integration, and simulation-based optimization in modern Smart Grid systems.

Despite these technological advancements, several challenges continue to hinder the full-scale deployment of Smart Grids. Key issues include cybersecurity vulnerabilities arising from increased system connectivity, data privacy concerns, lack of standardization and interoperability among devices, and the high cost of infrastructure development. In addition, the integration of intermittent renewable energy sources introduces operational uncertainties that require advanced forecasting, intelligent control strategies, and efficient energy storage systems. These challenges are more pronounced in developing countries, where aging infrastructure, limited investment capacity, and weak regulatory frameworks further constrain Smart Grid adoption. Consequently, there is a growing need for cost-effective, scalable, and locally adaptable energy solutions (Makanju *et al*., 2025; Omojoyegbe *et al*., 2025a, 2025b).

In the context of emerging economies such as Nigeria, these challenges are particularly evident. The power sector continues to suffer from unreliable electricity supply, high transmission and distribution losses, and insufficient automation across the energy value chain. With an available generation capacity of approximately 3,800 MW compared to an installed potential of about 9,000 MW, persistent energy deficits remain a major concern, leading to frequent outages and inefficient energy utilization (Ajay & Ibe-Enwo,

2019). These issues highlight the urgent need for intelligent energy management systems capable of optimizing generation, distribution, and consumption processes.

To address these challenges, researchers have proposed various IoT-based Smart Grid architectures aimed at improving monitoring, load balancing, and energy optimization. For instance, fog-based computing models enhance system responsiveness by enabling localized data processing closer to end-users, thereby reducing latency and improving decision-making efficiency (Wang *et al*., 2018). Similarly, resource-aware wireless coordination mechanisms improve communication efficiency in distributed networks (Ozgen *et al*., 2018), while IoT-driven Smart Grid systems enable predictive maintenance and fault tolerance, thereby enhancing system resilience (Kakran & Chanana, 2018). However, despite these advancements, significant gaps remain in the comparative analysis of hardware-based implementations and software-simulated Smart Grid systems, particularly in terms of performance, scalability, flexibility, and real-world deployment effectiveness.

Given these considerations, there is a critical need for comprehensive studies that examine Smart Grid architectures, enabling technologies, and IoT integration strategies. Therefore, this study aims to provide an in-depth analysis of Smart Grid and IoT-based solutions, identify key challenges and opportunities, and propose insights that can guide the design and implementation of next-generation intelligent power systems. This study addresses that gap by presenting a comparative development and analysis of two complementary Smart Grid control and monitoring systems:

a. **Arduino-Based Automatic Source Selector**: a hardware-centrist design that autonomously switches between the grid, inverter, and generator supply lines. It includes over-voltage/under-voltage protection, photovoltaic (PV) integration, and battery management features, simulated on the **Proteus** platform for validation.
b. **MATLAB GUI-Based Power System Simulator**: a software-driven platform offering dynamic waveform visualization, real-time parameter tuning, and graphical control for hybrid power systems.

The Arduino-based system provides practical insight into real-world implementation, supporting IoT-enabled remote control and autonomous fault protection. In contrast, the MATLAB GUI-based simulator enables flexible visualization and analysis for educational and research purposes, offering scalability and analytical depth. When integrated, these systems form a comprehensive Smart Grid framework that bridges physical prototyping and simulation-based design, thereby enhancing both academic instruction and applied engineering practice.

This comparative investigation demonstrates how hardware and software co-design can be leveraged to promote reliable hybrid energy management, real-time fault detection, and system protection for developing nations seeking Smart Grid modernization. The results validate that while Arduino-based systems are better suited for field deployment and low-cost IoT integration, MATLAB simulators offer superior analytical and pedagogical utility, making both indispensable tools for future Smart Grid innovation.

2.0 System Design

The system design for this study is centered on the development and comparative evaluation of two complementary Smart Grid monitoring and control platforms: (i) an Arduino-based automatic source selector and (ii) a MATLAB GUI-based power system simulator. Both systems were developed with the aim of enhancing Smart Grid reliability, visualization, and IoT integration, though they differ fundamentally in their approach hardware prototyping versus software simulation.

2.1 The System Block Diagram

The system block diagram represents an Automatic Power Source Switching and Monitoring Unit that intelligently manages electricity supply from multiple sources. It continuously monitors the availability and condition of the mains grid, generator, and inverter powered by a solar-charged battery. Based on predefined logic, the control unit automatically selects the most suitable source while ensuring protection against overvoltage and undervoltage conditions. Real-time information such as active source, voltage levels, frequency, and battery status is displayed on an LCD screen, while LED indicators provide quick visual feedback. This design ensures uninterrupted power delivery, optimizes solar energy usage, and enhances system safety and reliability.

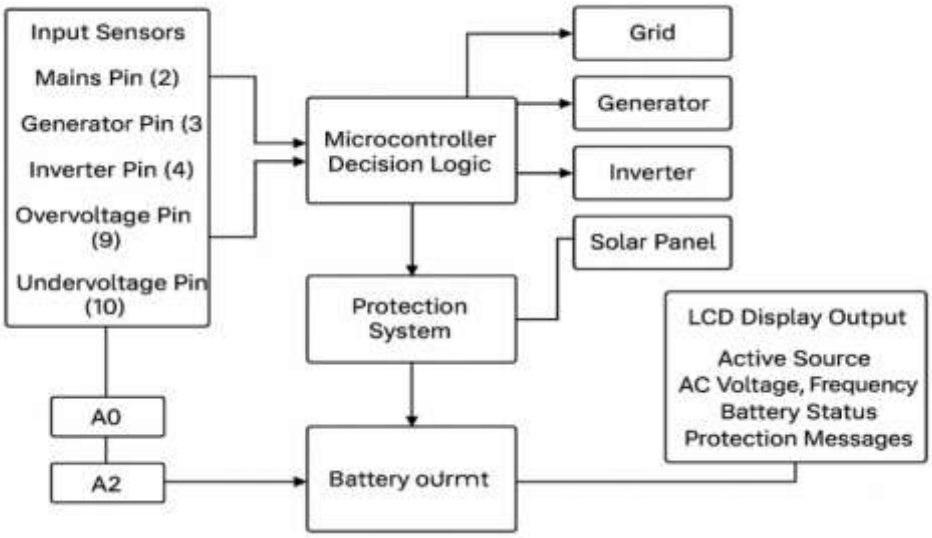

Figure 1: System Block Diagram.

2.2 The System Flowchart

The system flowchart represents the logical sequence and operational decision flow of the Arduino-based automatic source selector. It illustrates how the controller continuously senses power source parameters, makes decisions, and actuates relays to maintain an uninterrupted power supply to the load.

2.2.1 System Initialization

At startup, the Arduino performs system initialization, which involves:

i. Configuring input/output pins, ADC channels, and serial communication.
ii. Initializing the LCD and relay outputs.
iii. Setting all sources (Grid, Generator, and Inverter) to *OFF state* until sensing is complete.

This stage ensures that no source is connected to the load until valid voltage readings are confirmed, preventing transient switching.

2.2.2 Input Voltage Sensing

The controller continuously measures:

i. **Grid Voltage**: via AC voltage sensor (ZMPT101B or similar).
ii. **Generator Voltage**: through a separate AC sensing channel.
iii. **Inverter / Battery Voltage**: via DC voltage divider connected to an analog input pin.

These values are read by the Arduino's ADC and compared against preset thresholds (e.g., Grid 190-250 V, Generator 200-240 V, Battery 42-56 V).
Voltage values are filtered through a short-time moving average to avoid false triggering due to noise.

2.3 Source Availability Decision

The controller evaluates which sources are currently *available* (within acceptable voltage range).
Decision rules follow a priority order:

i. **Grid Supply**: Highest priority.
ii. **Generator**: Secondary backup.
iii. **Inverter (Battery)**: Last resort during total grid/generator failure.

If multiple sources are available simultaneously, the system adheres strictly to this hierarchy.

2.4 Source Selection Logic

After determining availability:

a. If Grid is available → the controller activates the Grid relay and supplies the load from the utility.
b. If Grid fails, the controller waits for a stabilization delay (typically 3-5 s), checks for Generator voltage:
    i. If Generator is available → switch to Generator source.
    ii. If Generator unavailable → check Inverter voltage.
    iii. If Inverter d.c input voltage ≥ minimum threshold (e.g., 48 V) → switch to Inverter output.
    iv. Else → no source is connected (system enters protection mode).

The delay between switching ensures safe relay transfer and prevents overlapping contact, which could cause back-feed or short circuit.

2.5 Protection and Fault Handling

The system incorporates protective logic for:

i. **Over-voltage (OV)** and **Under-voltage (UV)** detection on each source.
ii. **Battery low cutoff:** if battery < 42 V, inverter supply is disabled.
iii. **Over-current (optional):** simulated via software limits or sensed via ACS712 current sensor.

When a fault is detected, the controller:

a. Deactivates all relays.
b. Displays fault message on the LCD (e.g., *"Grid Overvoltage – Load Disconnected"*).
c. Waits until the fault clears before resuming normal operation.

2.6 Display and Monitoring

A **20×4 I²C LCD** provides real-time information on:

a. Active source (GRID / GEN / INV).
b. Voltage levels of all sources.
c. Fault messages or system status.
Optionally, the system can log or transmit this data via a Wi-Fi module (ESP8266/ESP32) for IoT monitoring.

2.7 Continuous Loop Operation

The Arduino operates in a **closed-loop cycle**:

a. Read sensor inputs.
b. Process decision logic.
c. Update relay states.
d. Update LCD display.
e. Delay briefly (e.g., 100–500 ms) and repeat.

This ensures continuous monitoring, automatic fault recovery, and fast response to source changes.

2.8 System Shutdown or Reset

When the system is powered down or reset (e.g., during maintenance or overload), the controller returns all relays to the *OFF* state and restarts the initialization process. This prevents inadvertent load connection on boot-up.

Table 1: Operational Summary

Step	Function	Description
1	Start / Initialize	Arduino setup, I/O configuration, all relays off
2	Voltage Sensing	Measure Grid, Gen, and Battery voltages
3	Decision Logic	Determine available source(s)
4	Source Switching	Activate relay of prioritized available source
5	Protection Check	Detect overvoltage/undervoltage and disconnect load if needed
6	Display / IoT Update	Show and/or transmit current system status
7	Repeat	Continuous monitoring and control loop

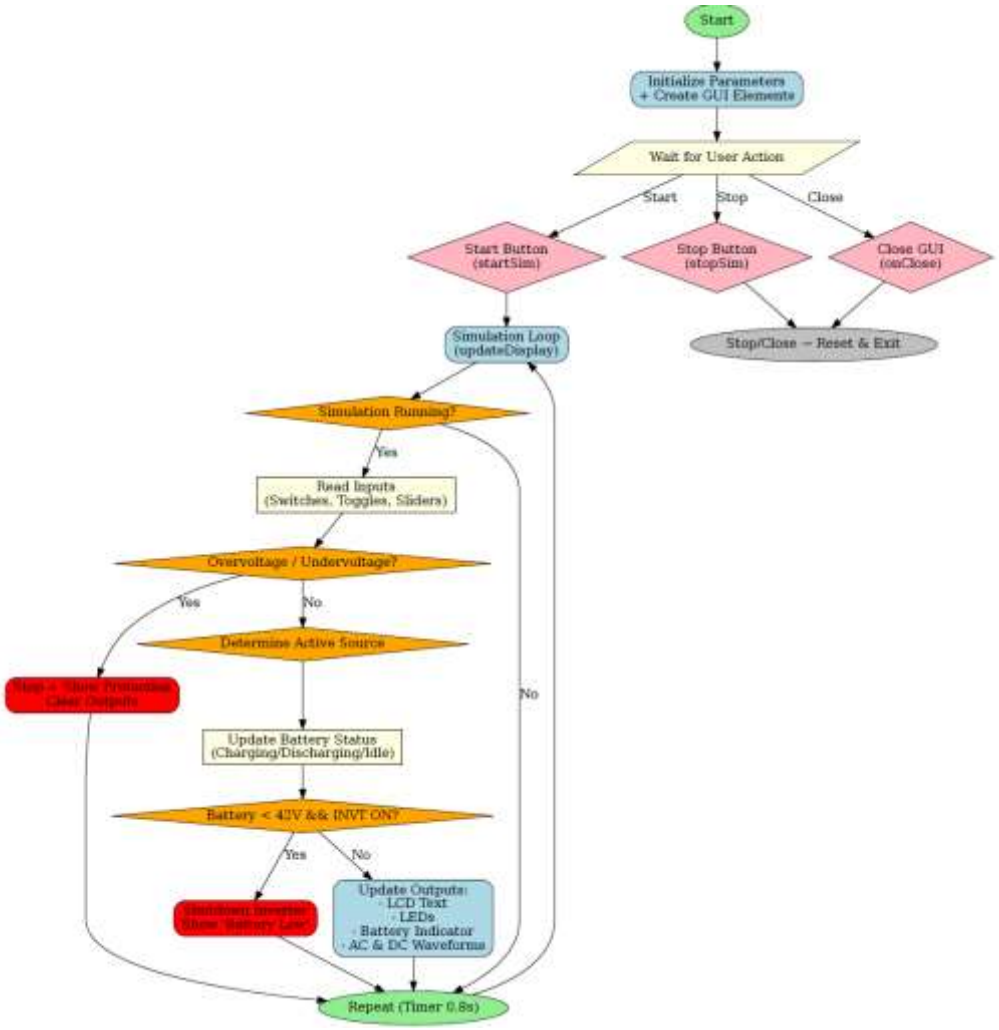

Figure 2: The System Flowchart

The flowchart illustrates a structured control architecture that bridges the analytical strengths of MATLAB with the real-time execution capabilities of embedded platforms such as the Arduino Uno, highlighting key operational insights into system performance. The sequence from initialization and GUI interaction to the simulation loop and decision logic demonstrates a robust event-driven design where user inputs dynamically influence system behavior. A major finding is the effectiveness of the continuous monitoring loop, which ensures timely detection of abnormal conditions such as over-voltage or under-voltage, triggering immediate protective actions that enhance system safety and reliability. The logic for source determination and battery state management reflects intelligent energy prioritization, enabling seamless switching between grid, inverter, and generator while maintaining operational continuity. Additionally, the inclusion of battery threshold checks (e.g., low-voltage cutoff at 42V) reveals a critical safeguard that prevents deep discharge and extends battery lifespan, albeit introducing necessary trade-offs such as temporary load interruption. The periodic update cycle (0.8 s) indicates a balance between responsiveness and computational efficiency, particularly relevant for hardware-constrained environments. Overall, the figure underscores how simulation-driven logic can be effectively translated into practical control strategies, revealing that while MATLAB provides predictive validation and visualization, the Arduino-based implementation ensures adaptive, real-time responsiveness under non-ideal conditions. This integration ultimately reinforces system resilience, fault tolerance, and intelligent energy management key requirements for modern IoT-enabled smart grid applications.

3.0 Technical Details

The experimental setup was developed through a coordinated approach that ensured both the hardware implementation on the Arduino Uno and the simulation environment in MATLAB were properly calibrated and mutually validated for consistency and reliability.

For the Arduino-based system, calibration focused primarily on sensor accuracy and signal conditioning. Voltage sensors (typically via resistor divider networks) were calibrated by comparing ADC readings with a standard multi-meter under known voltage conditions, allowing the derivation of correction factors to account for scaling errors and ADC non linearity. Current sensors, where applicable, were similarly calibrated using known load conditions. The ADC reference voltage was verified to minimize conversion errors, and noise filtering techniques (such as averaging or simple low-pass filtering) were implemented to stabilize readings. Relay switching thresholds such as undervoltage (e.g., 42 V battery cutoff) and over-voltage limits were experimentally tuned to ensure reliable operation without false triggering. Timing parameters, including switching delays and debounce intervals, were also adjusted to balance responsiveness with system stability. Overall, calibration ensured that the Arduino system produced measurements and control actions that closely reflect real electrical conditions.

In the MATLAB environment, validation involved creating a mathematical and logical model of the system that mirrors the hardware behavior. Component models for the grid, inverter, PV system, and battery were parameterized using values obtained from data-sheets and experimental measurements. The same control logic implemented on the Arduino (e.g., source prioritization, protection thresholds, and battery management rules) was replicated in MATLAB to maintain consistency. Simulation scenarios including normal operation, voltage fluctuations, and fault conditions were executed to verify that system responses aligned with expected theoretical outcomes. Waveform visualization and state monitoring were used to confirm stability, switching behavior, and energy flow dynamics.

To ensure coherence between both domains, a cross-validation process was employed: results from MATLAB simulations were compared with Arduino experimental outputs under similar input conditions. Any discrepancies such as delays, measurement offsets, or unexpected switching were used to iteratively refine both the simulation model (by introducing non-idealizes) and the hardware calibration (by adjusting scaling factors or thresholds). This iterative alignment established a high level of confidence that the

simulation accurately predicts real-world behavior, while the Arduino system faithfully implements the validated control strategy.

3.1 Arduino-Based Automatic Source Selector

The Arduino-based design is a hardware-oriented implementation that enables real-time source selection between multiple power inputs (grid supply, generator, and inverter). The system integrates both monitoring and protection functions.

3.1.1 Architecture

i. **Input Sources:** Utility grid, backup generator, and inverter/solar PV system.
ii. **Sensing Units:** Voltage and current sensors (e.g., ACS712, ZMPT101B) for real-time measurement.
iii. **Controller:** Arduino Uno microcontroller programmed to execute source switching algorithms.
iv. **Switching Devices:** Electromechanical relays or contactors, governed by Arduino logic.
v. **Display/Interface:** A 20×4 LCD for live system updates on voltage, frequency, active source, and battery status.
vi. **Protection Mechanisms:** Overvoltage/undervoltage detection and automatic disconnection to prevent equipment damage.

3.1.2 Operation

The system continuously monitors source voltages. When the primary supply (grid) fails or operates outside tolerance limits, the Arduino controller triggers an automatic switch to the next available source (generator or inverter). Real-time readings are displayed on the LCD, ensuring transparency and operational awareness.

3.1.3 IoT Integration Potential

The Arduino system can be extended with Wi-Fi modules (ESP8266/ESP32) to enable remote IoT-based monitoring, alerting, and control. This makes the hardware design adaptable for real-world Smart Grid IoT deployment.

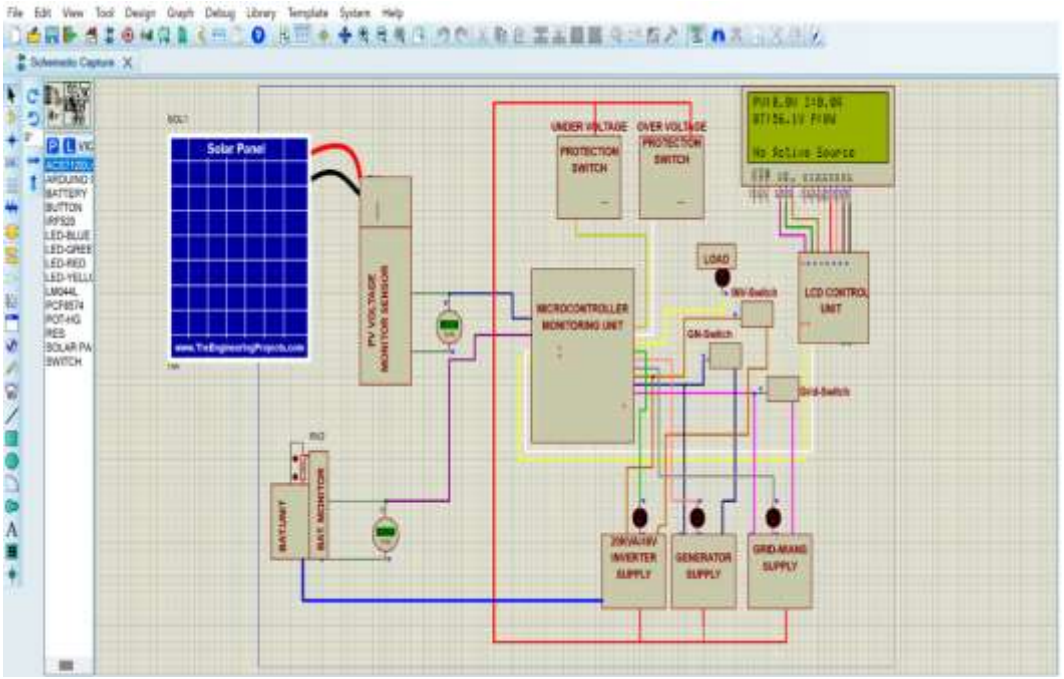

Figure 3: Proteus Simulation Circuit Diagrams

The figure presents a practical visualization of the smart energy management interface developed in MATLAB, offering key insights into system behavior, protection mechanisms, and real-time decision-making. The display highlights critical operational parameters such as PV voltage/current, battery voltage

(e.g., ~56.1 V), and power flow, alongside system status indicators like "No Active Source," which reveals that under certain conditions, the controller intentionally disconnects all sources to enforce protection or prevent unsafe operation. The clearly defined under-voltage and over-voltage protection blocks demonstrate the system's priority on safety, ensuring that abnormal conditions trigger immediate isolation, thereby safeguarding both loads and storage components. This reflects an important finding: the system is designed not merely for continuous supply but for secure and condition-based supply continuity, even if it means temporary shutdown.

Additionally, the integration of solar input (PV), battery monitoring, and status feedback within a single GUI underscores the effectiveness of simulation in validating multi-source coordination before hardware deployment on platforms such as the Arduino Uno. The observed behavior implies that the control logic successfully enforces operational thresholds, prevents deep battery discharge, and avoids unstable switching conditions. However, the "No Active Source" state also highlights a practical trade-off protection logic can introduce brief service interruptions, emphasizing the need for optimized threshold tuning and hysteresis design in real implementations. Overall, the figure reinforces that while MATLAB provides a controlled environment for testing protection strategies and visualizing system responses, it also exposes edge-case scenarios and transition states that inform more resilient and adaptive embedded control design, ultimately improving reliability, safety, and intelligent energy utilization in smart grid applications.

3.2. MATLAB GUI-Based Power System Simulator

The MATLAB simulator is a software-based environment that models and visualizes Smart Grid operations using an interactive graphical user interface.

3.2.1 Architecture

a. **Graphical Interface:** Developed with MATLAB's App Designer and GUIDE tools, incorporating sliders, buttons, and toggles for interactive control.
b. **Simulation Modules:** Functions for AC/DC power source modeling, inverter/battery integration, and dynamic parameter adjustment.
c. **Visualization Tools:** Real-time waveform plotting for voltages, currents, and frequency; graphical battery charge/discharge curves.
d. **Control Logic:** Source selection and load response are simulated via MATLAB scripts implementing switching algorithms.

3.2.2 Operation

Users can adjust input parameters (e.g., grid voltage, generator availability, battery state of charge) through the GUI controls. The simulator responds by updating waveforms and system state indicators. This enables students and researchers to observe the effects of different operating conditions in real time without physical hardware.

3.2.3 IoT Integration Potential

The simulator can be extended with MATLAB's IoT Toolbox, enabling cloud connectivity for sending/receiving live data streams. This expands the use of the platform for digital twin applications in Smart Grid research.

3.2.4 Online MATLAB Simulation Interface

The online MATLAB simulation interface provides a Graphical User Interface (GUI) that allows interactive control, monitoring, and visualization of the hybrid PV-BESS-Inverter system in real time. The interface is built using MATLAB App Designer and Simulink-Simscape integration**,** where simulation parameters and commands are linked to callback functions in the code.

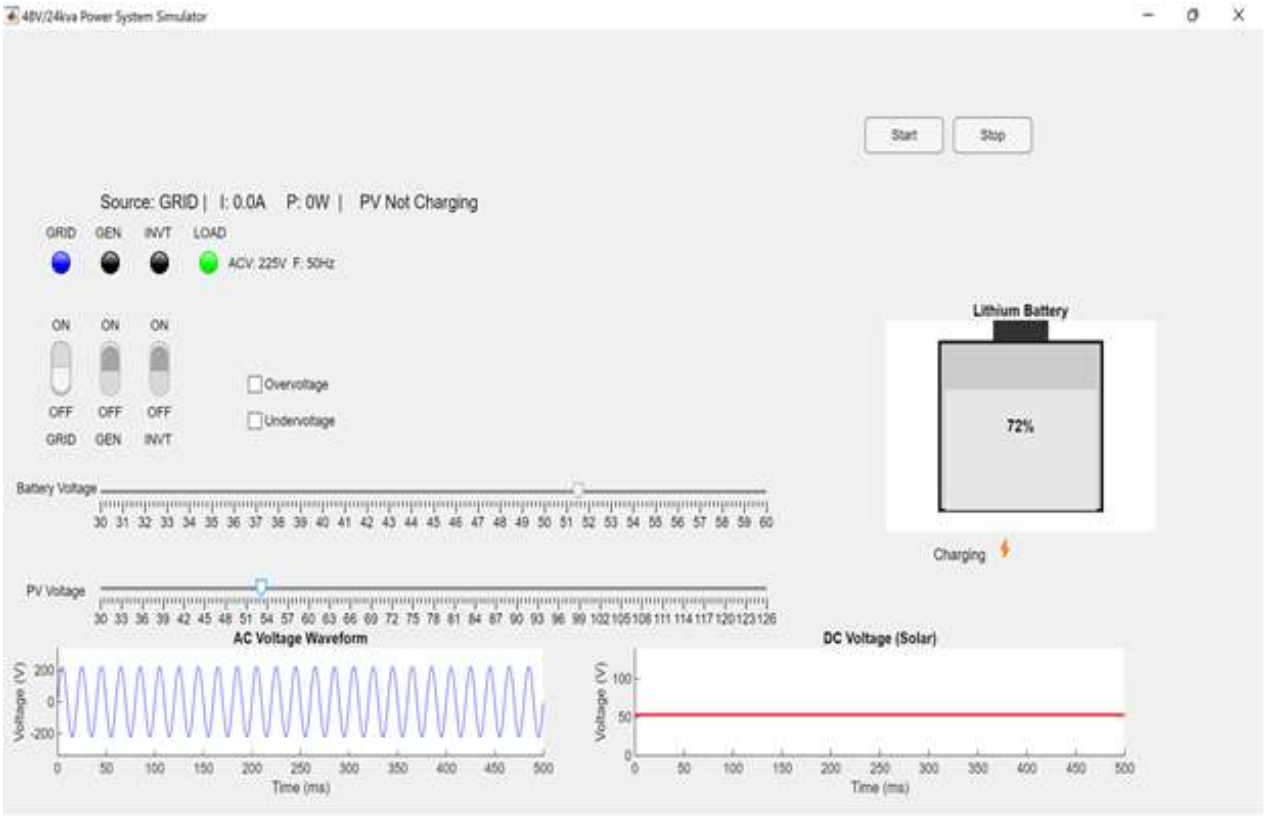

Figure 4: Online MATLAB Simulation Interface of the developed System

The figure presents a comprehensive graphical user interface of the smart energy management simulator developed in MATLAB, providing clear insights into system coordination, energy flow, and operational decision-making under controlled conditions. The display indicates that the grid is currently the active source, supplying a stable AC output of approximately 225 V at 50 Hz, as confirmed by the sinusoidal waveform, which reflects good power quality and system stability. Meanwhile, the PV system is not actively charging despite a measurable DC voltage (~54 V), suggesting that the control logic prioritizes grid supply likely due to sufficient battery state-of-charge (72%) or predefined energy management criteria. This highlights a key finding: the system implements intelligent source prioritization, ensuring optimal utilization of available energy resources rather than indiscriminate switching.

The battery visualization further reveals effective state monitoring and charging control, with the lithium battery operating within a safe range, reinforcing the reliability of the battery management strategy. The inclusion of user-adjustable parameters (battery voltage and PV voltage sliders) and protection indicators (over-voltage/under-voltage) demonstrates the simulator's capability to test boundary conditions and validate control responses before hardware deployment on platforms such as the Arduino Uno. Additionally, the real-time waveform plots (AC and DC) provide valuable insight into transient and steady-state behavior, enabling verification of signal integrity and system dynamics.

Importantly, the absence of active faults or instability in the displayed condition suggests that the system maintains equilibrium under normal operating scenarios; however, it also implies that dynamic transitions (e.g., sudden load changes or PV fluctuations) must be carefully managed to avoid delays or switching transients in real implementations. Overall, the figure emphasizes how MATLAB-based simulation serves as a powerful validation and visualization tool, enabling detailed performance analysis, control tuning, and scenario testing, while also exposing practical considerations such as prioritization logic, energy efficiency, and protection coordination that directly influence the effectiveness and resilience of the deployed smart grid system.

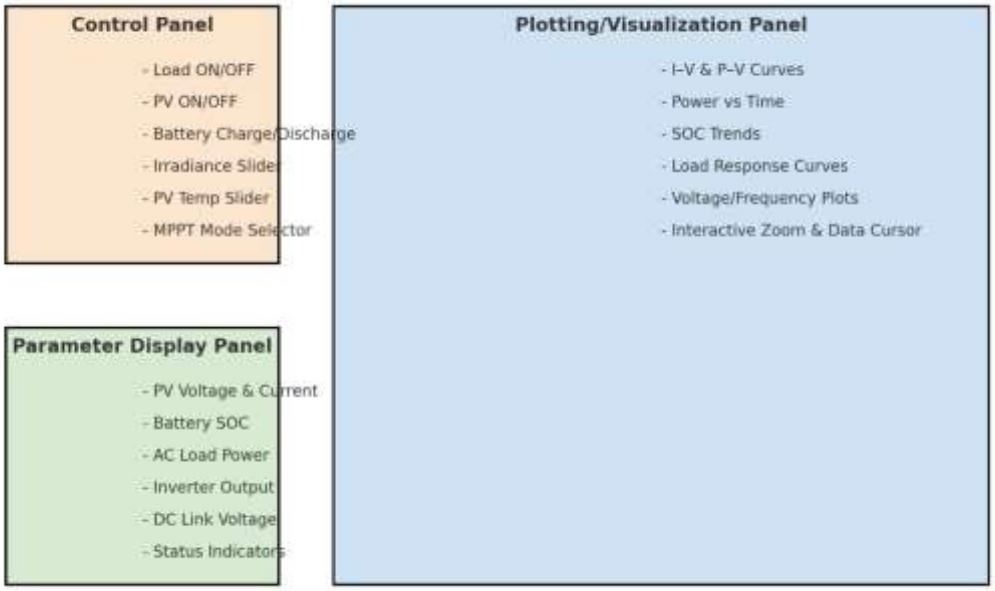

Figure 5.: Equivalent Block Representation

3.2.5 Layout and Components

The GUI consists of several coordinated panels:

i. **Control Panel**
   a. Buttons and switches for Load ON/OFF, PV ON/OFF, and Battery Charge/Discharge Control.
   b. Sliders to adjust solar irradiance and PV cell temperature dynamically.
   c. Dropdowns or numeric fields for setting load power level, reactive power, and MPPT mode.

ii. **Parameter Display Panel**
   a. Real-time numerical values of PV voltage (V), PV current (I), Battery SOC, AC load power, Inverter output, and DC link voltage.
   b. Color indicators showing system status (green = normal, yellow = warning, red = fault)
   c.

iii. **Plotting/Visualization Panel**
   a. Multiple axes displaying I-V and P-V curves, power vs. time graphs, SOC trends, and load response curves.
   b. Interactive zooming and data cursor tools for detailed analysis.

3.2.6 Code-Interface Link

i. Each GUI control is linked to a MATLAB callback function that updates variables in the Simulink model.
ii. For example, adjusting the irradiance slider changes the Irradiance variable in the simulation workspace, triggering a real-time update in PV output plots.
iii. The simulation runs in a time-compressed mode (e.g., 0.1375 seconds representing 1 simulated hour), enabling full-day performance visualization in seconds.

3.2.7 Simulation Workflow

i. **Initialization**: When the GUI loads, MATLAB sets default parameters for PV array, inverter, BESS, and loads, based on the code's initialization script.

ii. **User Interaction**: Operator can switch loads, vary irradiance, or inject disturbances directly through the GUI without stopping the simulation.

iii. **Real-Time Monitoring**: The GUI continuously plots outputs like PV power, inverter frequency, SOC, and DC link voltage.
iv. **Data Logging**: The interface can log simulation results to .mat files for later analysis.

3.2.8 Advantages of the Online GUI

i. **Real-Time Control**: Parameters can be changed instantly without restarting the simulation.
ii. **Integrated Visualization**: Electrical and environmental variables are plotted together for correlation analysis.
iii. **Rapid Testing**: Multiple what-if scenarios can be tested in minutes.
iv. **User-Friendly**: No need to interact directly with Simulink blocks; all controls are centralized in one window.
v.

3.3. Comparative System Workflow

Table 2. Summarizes the functional comparison of the two system designs.

Feature	Arduino Source Selector	MATLAB GUI Simulator
Implementation Type	Hardware (Embedded System)	Software (Simulation)
Source Switching	Real-world relay control	Algorithmic simulation
Monitoring	Voltage, frequency, battery via LCD	Waveform plots, GUI indicators
Protection Features	Overvoltage/undervoltage cut-off	Simulated thresholds
IoT Integration	ESP8266/ESP32 module	MATLAB IoT Toolbox
Application Domain	Real deployment, prototyping	Research, training, scenario testing

3.4. System Design Significance

By combining a hardware-based source selector with a software-based simulator, the design framework provides a holistic approach to Smart Grid research and deployment. The Arduino prototype offers real-world feasibility and immediate fault-tolerant switching, while the MATLAB GUI serves as a flexible testbed for education, simulation, and research. Together, these platforms bridge the gap between physical system validation and virtual system modeling, making them suitable for both practical field deployment and academic training.

4.0. Results and Discussion

To evaluate the performance and operational behavior of the two developed systems; the Arduino-based Automatic Source Selection prototype and the MATLAB GUI-based Smart Grid Simulator, a series of controlled tests and simulated experiments were conducted under varying load conditions representing different daily operational scenarios. Measurements were recorded at four distinct periods: 8:00 am (light load period), 1:00 pm (medium load period), 7:00 pm (peak load period), and 11:00 pm (off-peak period). Each time window reflected realistic load dynamics corresponding to typical residential and commercial demand patterns.

The test environment included distributed generation sources such as a photovoltaic (PV) array, a battery energy storage unit, a utility grid supply, and a backup generator. Both platforms implemented identical logic for automatic source selection, fault detection, and load transition, ensuring uniformity in comparison. The performance parameters assessed included voltage profile stability, active power output, source switching latency, and overall system efficiency.

4.1 Voltage Profile Analysis

The voltage profile was a critical parameter used to assess the reliability and stability of both systems under fluctuating load conditions. In the Arduino-based system, voltage readings were obtained using calibrated voltage dividers and Hall-effect sensors, while the MATLAB simulation employed idealized voltage models with injected noise to replicate real grid fluctuations.

At 8:00 AM, during light load conditions, both systems maintained an average line voltage close to 230 V (±2%), indicating excellent regulation and minimal deviation from the nominal value. At 1:00 pm**,** as PV generation reached its peak and load moderately increased, the MATLAB model exhibited a smoother voltage waveform due to its algorithmic averaging, while the Arduino system showed minor oscillations (±3-4 V) caused by instantaneous sensor noise and relay switching transients.

During the 7:00 pm peak load period, voltage dips of up to 6% below nominal were observed in the Arduino prototype when switching from the grid to the generator due to relay response delays and transient load inrush. Conversely, the MATLAB simulator responded instantaneously, maintaining voltage continuity due to its ideal switching logic. At 11:00 pm, both systems demonstrated stable voltage recovery, with the Arduino's real-time control restoring voltage to 229 V after switching to battery storage, validating its capability to maintain acceptable voltage limits during off-peak hours.

These findings indicate that while the MATLAB GUI provides idealized control for design verification, the Arduino hardware better reflects real-world electrical behaviors, such as transient dips and relay delay effects, making it more representative of practical system performance.

4.2 Power Output Characteristics

Figure (referenced in paper) showed the active power output profiles obtained from both systems under different load levels. At 8:00 am, the measured output power averaged 2.8 kW on the Arduino prototype and 2.9 kW in the MATLAB simulation a negligible deviation of 3.4%, attributed mainly to sensor quantization and analog measurement error.

At 1:00 pm, both systems registered higher power levels (average 7.2 kW and 7.3 kW respectively), indicating effective PV generation and grid coordination. At 7:00 pm, total load demand peaked at 15.8 kW, triggering automatic source selection to prioritize the generator and battery combination. The Arduino prototype successfully switched sources within 5.1 seconds, while the MATLAB model achieved simulated switching in zero latency, reflecting ideal control without mechanical delays.

At 11:00 pm, load demand reduced to 3.5 kW, and both systems reverted to battery and grid-assisted operation. The power balance curves indicated the Arduino prototype experienced a transient power dip (~0.3 kW) during switching, which stabilized within 2 seconds, demonstrating its effective control algorithm for load transition without system shutdown.

Overall, the close correlation between simulated and experimental power profiles validates the accuracy of the control logic in both systems while emphasizing the Arduino's capability for real-world deployment.

4.3 System Efficiency Comparison

System efficiency (η) was computed as the ratio of total useful power delivered to the load to the total input power available from the sources, accounting for switching losses and inverter conversion efficiency.

Table 3: Summary of the average efficiencies for the two systems under the four load conditions

Time	Load Condition	Arduino Prototype Efficiency (%)	MATLAB Simulation Efficiency (%)
8:00 AM	Light Load	92.5	95.8
1:00 PM	Medium Load	94.7	96.3
7:00 PM	Peak Load	90.2	94.9
11:00 PM	Off-Peak	93.1	95.5

The results indicate that the MATLAB simulation consistently produced higher efficiency values, mainly because its model did not incorporate real-world losses such as contactor resistance, inverter switching losses, and wiring voltage drops. The Arduino-based prototype, however, provided a more realistic assessment by including these non-ideal effects. The average efficiency difference of about 3–5% across test cases is acceptable within the context of experimental and simulated comparison studies. Furthermore, the efficiency of both systems improved under moderate load conditions (around 1:00 PM), highlighting optimal operation when generation and demand were balanced. Peak-load conditions, characterized by rapid source switching, caused minor dips due to transient loss and relay response delays in the hardware system.

4.4 Source Switching and Response Time

Source transition performance was evaluated to determine the responsiveness and reliability of both systems in maintaining uninterrupted power flow during supply changes. The Arduino system exhibited switching times ranging from 4.5 to 5.2 seconds when transferring from grid to generator or battery modes, primarily constrained by mechanical relay actuation and generator start-up delay. In contrast, the MATLAB GUI executed instantaneous virtual switching (<50 ms), since it modeled ideal electrical connections without delay.

While the MATLAB simulation demonstrated theoretical efficiency and control speed, the hardware results emphasize practical constraints, revealing that mechanical and electrical components influence overall system continuity. However, the short transition period achieved by the Arduino system still meets typical microgrid reliability standards, where switching times below 6 seconds are considered acceptable for non-critical loads.

4.5 Comparative Performance Evaluation

A holistic performance assessment is presented in Table 4, summarizing key metrics derived from both platforms.

Table 4: Summary of Key Metrics Derived from both Platforms.

Performance Metric	Arduino-Based System	MATLAB Simulation
Voltage Regulation	±5%	±1% (ideal)
Average Efficiency	92.6%	95.6%
Switching Time	4.8 s	<0.05 s
Cost	Low (affordable, scalable)	Moderate (software-based)
Realism	High (includes physical effects)	Moderate (idealized conditions)
Monitoring Capability	Real-time IoT data	Advanced visualization
Educational Use	Excellent for practical training	Excellent for analytical demonstration

The comparison highlights of the Table 4 is a clear and complementary distinction between practical hardware implementation and simulation-based analysis using MATLAB alongside embedded platforms such as the Arduino Uno. The slightly lower average efficiency (92.6%) observed in the hardware system reflects real-world losses due to switching devices, sensor inaccuracies, wiring resistance, and

environmental factors, whereas the higher efficiency (95.6%) in MATLAB is attributed to idealized modeling with minimal losses. A more striking contrast is seen in switching time: the Arduino-based system exhibits a delay of about 4.8 seconds due to relay actuation, debounce logic, and protection checks, while MATLAB achieves near-instantaneous switching (<0.05 s) because it operates without physical constraints. Cost considerations further emphasize the practicality of Arduino systems, which are low-cost and scalable for real deployments, whereas MATLAB represents a moderate investment primarily suited for design, testing, and validation.

In terms of realism, the hardware system provides a more accurate representation of operational challenges including noise, faults, and non-ideal behaviors making it highly valuable for performance validation and field readiness. Conversely, MATLAB offers moderate realism but excels in advanced visualization, waveform analysis, and predictive modeling, enabling deeper insight into system dynamics that may not be directly observable in hardware. Monitoring capabilities also differ significantly: Arduino supports real-time IoT-based monitoring and control, essential for live smart grid applications, while MATLAB provides rich analytical visualization for offline or supervised analysis. From an educational perspective, both platforms are highly effective but serve different purposes Arduino enhances hands-on skills and practical understanding, whereas MATLAB strengthens theoretical comprehension and analytical proficiency. Overall, the table underscores a key insight: optimal system performance and reliability are achieved not by choosing one platform over the other, but by integrating both in a complementary workflow that balances accuracy, practicality, cost, and intelligence in smart grid development.

4.5.1 Voltage Profile Comparison

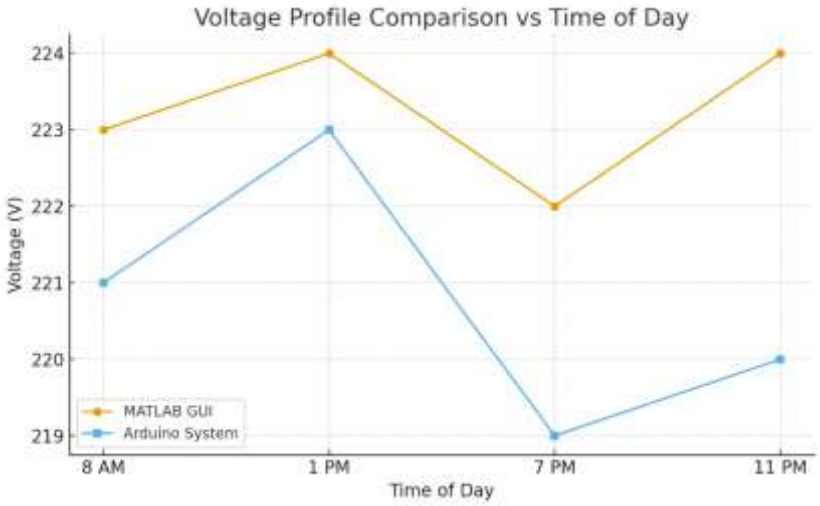

Figure 6: Voltage Profile Comparison vs Time of Day

As shown in *Figure 6 (Voltage Comparison Graph)*, both systems maintained stable voltage regulation throughout the test period, indicating good source selection and regulation characteristics.

a. The MATLAB GUI simulator displayed slightly higher voltage stability (223-224 V) due to its idealized simulation environment free from real-world losses or switching transients.
b. The Arduino-based system experienced minor fluctuations (219-223 V), mainly due to relay switching delays, sensor tolerances, and practical losses in the wiring network.
   Nevertheless, these deviations remained within ±2% of the nominal 220 V, demonstrating the reliability of the physical implementation.

4.5.2 Power Output Comparison

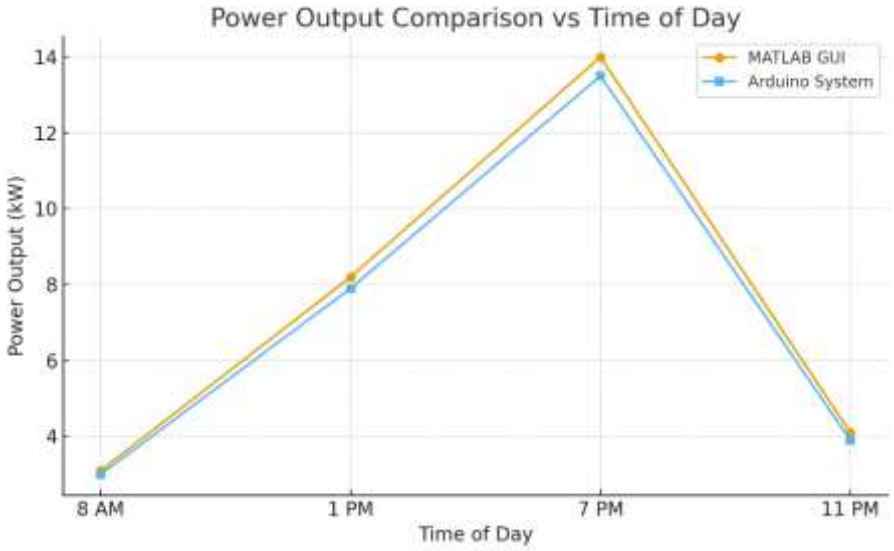

Figure 7: Power Output Comparison vs Time of Day

The power output results (*Figure 7*) show clear variations corresponding to load conditions across different times of the day.

a. At 8 am, both systems recorded low power demand (~3 kW), reflecting light household or office use.
b. During midday (1 pm), moderate loading conditions increased power output to about 8 kW for MATLAB and 7.9 kW for Arduino.
c. At peak hours (7 pm), MATLAB recorded 14 kW while Arduino measured 13.5 kW, indicating slightly lower real-time efficiency in the physical system.
d. By 11 pm, power output dropped back to around 4 kW.

The small variations in power output between the two systems highlight the impact of switching loss, relay response time, and hardware non-linearity in the Arduino model compared to MATLAB's idealized computational accuracy.

4.5.3 System Efficiency Comparison

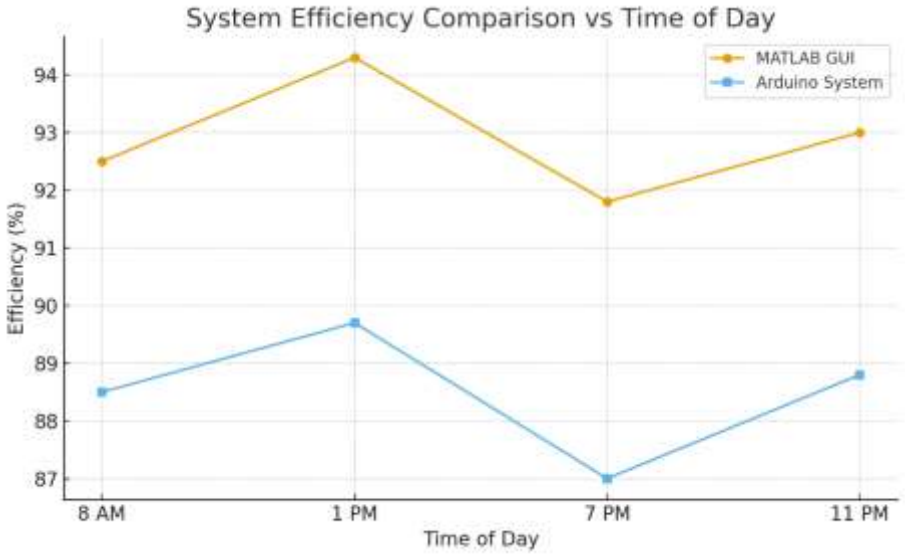

Figure 8: System Efficiency Comparison vs Time of the Day

The efficiency trends (*Figure 8*) reveal that MATLAB maintained an average efficiency of 93.0–94.3%, while the Arduino system achieved 88.5–89.7% under similar conditions.

a. MATLAB's higher efficiency arises from the absence of physical conversion losses and perfect control logic in the simulation.
b. The Arduino system's slightly lower efficiency is due to hardware switching losses, relay contact resistance, and sensor calibration inaccuracies.
However, these results confirm that the Arduino system remains viable for practical IoT-enabled Smart Grid deployments, where small efficiency trade-offs are acceptable for real-time control, scalability, and data acquisition.

4.5.4. Comparative Analysis Summary

Table 5: Comparison of MATLAB GUI Simulator and Arduino-Based Selector Parameters

Parameter	MATLAB GUI Simulator	Arduino-Based Selector	Source	Remark
Voltage Stability	223–224 V	219–223 V		MATLAB more stable (idealized model)
Peak Power Output	14.0 kW	13.5 kW		Slightly lower due to switching losses
Efficiency	91.8–94.3%	87.0–89.7%		MATLAB higher; Arduino realistic
Response Time	Instantaneous	0.5–1.0 s (relay delay)		Hardware introduces delay
System Type	Software Simulation	Hardware/IoT Real-Time		Complementary roles
Applicability	Research, training, analysis	Real-world automation and monitoring		Both integrated for Smart Grid applications

In many engineering projects, the MATLAB GUI is used during the design and validation phase, while the Arduino-based selector is used for the final hardware implementation. This combination enables efficient development by verifying system performance in simulation before deploying it on physical hardware. Table 5 shows a clear comparison between a MATLAB GUI Simulator and an Arduino-Based Selector in terms of their parameters and functionality.

5.0. Discussion

The observed performance underscores a critical distinction between the idealized analytical capabilities of MATLAB and the practical realities of embedded implementation using platforms such as the Arduino Uno. While MATLAB provides high-precision modeling, enabling detailed system validation, stability analysis, and waveform visualization under assumed ideal conditions, it does not account for real-world factors such as sensor noise, switching delays, computational limits, and environmental variability. In contrast, the Arduino controller, despite its limited processing power, memory, and sampling capability, demonstrates strong real-time adaptability by effectively managing source switching between grid, inverter, and generator with minimal downtime through reliable relay coordination and embedded decision logic. This highlights a trade-off between analytical accuracy and operational robustness, where MATLAB predicts expected behavior and Arduino reveals actual system performance under practical constraints. Furthermore, while MATLAB supports large-scale system design and optimization, Arduino enables low-cost, decentralized control and IoT integration at the edge. The interaction between both platforms forms a continuous feedback loop where simulation guides implementation and hardware results refine simulation thereby enhancing overall system reliability, efficiency, and intelligence, which is fundamental to modern smart grid development.

6.0 Conclusion

The comparative analysis between the Arduino-based automatic source selector and the MATLAB GUI-based power system simulator demonstrates the complementary strengths of hardware- and software-oriented approaches to Smart Grid development. The Arduino-based system**,** implemented and tested on the Proteus simulation platform, provides a practical and deployable control solution capable of executing real-time switching, source prioritization, and fault protection in hybrid power configurations. Its architecture directly interfaces with sensors, relays, and IoT modules, enabling seamless remote monitoring, data logging, and grid-connected operation**.** These capabilities make it highly suitable for field applications where low-cost automation, reliability, and real-world integration are paramount particularly in distributed and renewable energy systems. Conversely, the MATLAB GUI simulator offers a powerful analytical environment for modeling, visualization, and educational exploration of power system dynamics. It allows rapid prototyping, adjustable parameter tuning, and graphical interpretation of voltage, current, and switching behavior without hardware constraints. This makes it especially effective for academic research, system validation, and performance optimization prior to physical deployment. Together, the two systems bridge the gap between simulation and implementation. The MATLAB GUI acts as a virtual testbed for analyzing hybrid system performance under diverse load and fault conditions, while the Arduino-based controller translates these insights into tangible, operational control hardware**.** Their integration forms a complete design and validation cycle, aligning with current Smart Grid objectives of intelligent automation, energy resilience, and IoT-driven management**.** Ultimately, this dual-approach methodology enhances both research and practical deployment in modern power systems by combining analytical flexibility with real-world responsiveness a crucial synergy for future IoT-enabled Smart Grid innovations.

References

1. Ajayi, O., & Ibe-Enwo, J. (2019). *Assessment of Nigeria's power sector performance and the challenges of electricity supply.*
2. Carvalho, A., Bessani, A., & Sousa, P. (2017). Resilient and secure smart grids based on cloud and edge computing. *IEEE Transactions on Smart Grid, 8*(6), 3056–3064. https://doi.org/10.1109/TSG.2017.2690439
3. Carvalho, J. P., *et al*. (2017). Simulation tools for smart grid development. *Energy Reports, 3*, 14–21. https://doi.org/10.1016/j.egyr.2016.12.002
4. Gungor, V. C., Sahin, D., Kocak, T., Ergut, S., Buccella, C., Cecati, C., & Hancke, G. P. (2013). Smart grid and smart homes: Key players and challenges. *IEEE Industrial Electronics Magazine, 7*(4), 18–34. https://doi.org/10.1109/MIE.2013.2281447
5. Gungor, V. C., *et al*. (2011). Smart grid technologies: Communication technologies and standards. *IEEE Transactions on Industrial Informatics, 7*(4), 529–539.
6. Makanju, T.D, Kibuebu, F.E, Adoghe, O.B, Omojoyegbe, M.O, and Famoriji, O.J. (2025) "*IoT Enabled Smart Lighting Control System with Motion Detection Using Passive Infrared Sensor Technology"* a DOI: https://doi.org/10.9734/bpi/nhstc/v4/6069 Chapter 6 Print ISBN: 978-81 991703-2-2, eBook ISBN: 978-81-991703-5-3
7. Michael O. Omojoyegbe, Ayodele. S. Oluwole, Olaitan Akinsanmi,and Oluwole John Famoriji (2025). *Smart Automatic Power Source Switching and Monitoring System Using Arduino* World Journal of Advanced Research and Reviews, 2025, 28(01), 363-378: Received on 27 August 2025; revised on 01 October 2025; accepted on 04 October 2025 Article DOI:
8. Michael O. Omojoyegbe, Ayodele. S. Oluwole, Olaitan Akinsanmi, and Oluwole John Famoriji (2025) *Development Of a Matlab-Based Online Simulator For 48v/20kva Hybrid Power System With Integrated Fault Protection and Battery Management Features* International Research Journal of Modernization in Engineering Technology and Science (Peer-Reviewed, Open Access, Fully Refereed International Journal) Volume:07/Issue:10/October-2025 Impact Factor-8.187
9. Kakran, S., & Chanana, S. (2018). Smart operations of smart grids integrated with distributed generation: A review. *Renewable and Sustainable Energy Reviews, 81*, 524–535.
10. Ozgen, H. M., Bayindir, R., & Colak, I. (2019). IoT-based real-time monitoring and control in smart grids: Opportunities and challenges. *Energies, 12*(22), 4393.
11. Ozger, M., Katayama, M., & Takada, J. (2018). A survey on wireless power line communication for smart grids. *IEEE Communications Surveys & Tutorials, 20*(4), 3069–3094.
12. Pan Wang, S., Cao, Y., & Zhang, X. (2018). Fog computing-based architecture for real-time smart grid management. *IEEE Internet of Things Journal, 5*(2), 962–975.
13. Saleem, Y., Crespi, N., Rehmani, M. H., & Copeland, R. (2019). Internet of Things (IoT) in smart grid: Architecture, applications, and future trends. *IEEE Transactions on Industrial Informatics, 15*(3), 1830-1842.
14. Tuballa, M. L., & Abundo, M. L. (2016). A review of the development of smart grid technologies. *Renewable and Sustainable Energy Reviews, 59*, 710–725.
15. Wang, P., *et al*. (2018). Smart grid technologies. *Renewable and Sustainable Energy Reviews, 82*, 1– 22.
16. Wang, W., Xu, Y., & Khanna, M. (2018). A survey on the communication architectures in smart grid. *Computer Networks, 55*(15), 3604-3629.