ENERGY MANAGEMENT FOR SMART HOMES WITH LOAD DEMAND RESPONSE

A PROJECT REPORT

SUBMITTED BY:

ABDUL BASID K.P (MES20EE001)

ASLAM C.V (MES20EE006)

MUSAAB (MES20EE023)

NABEEL THARI (MES20EE024)

(NAAC ACCREDITED WITH 'A' GRADE)

DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING

[B.TECH. PROGRAMME ACCREDITED BY NBA]

MES COLLEGE OF ENGINEERING KUTTIPPURAM THIRKKANAPURAM

P.O, MALAPPURAM DT, KERALA, INDIA 679582

2023-2024

DECLARATION

We undersigned hereby declare that the project report "**Energy Management For Smart Homes With Load Demand Response",** submitted for partial fulfilment of the requirements for the award of degree of Bachelor of Technology of the APJ Abdul Kalam Technological University, Kerala is a Bonafide work done under supervision of Dr. Maya G declared that this Project report is my own work and has not been previously submitted for assessment. We have acknowledged all sources of information used in this report and have referenced them according to the citation guidelines. We understand that any breach of academic honesty will result in disciplinary action.

SIGNATURE

ABDUL BASID K.P

ASLAM C.V

MUSAAB

NABEEL THARI

DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING

MES COLLEGE OF ENGINEERING, KUTTIPPURAM

CERTIFICATE

This is to certify that the report entitled "**Energy Management For Smart Homes With Load Demand Response"** submitted by **ABDUL BASID K.P, ASLAM C.V, NABEEL THARI** and **MUSAAB**, to the APJ Abdul Kalam Technological University in partial requirements for the award of the Degree of Bachelor of Technology in Electrical and Electronics Engineering is a Bonafide record of the project work carried out under my guidance and supervision.

PROJECT GUIDE

DR. MAYA G

ASSOCIATE PROFESSOR

DEPT. OF ELECTRICAL & ELECTRONICS ENGINEERING

MES COLLEGE OF ENGINEERING

HEAD OF DEPARTMENT

DR. NAFEESA K

PROFESSOR & HEAD

DEPT. OF ELECTRICAL & ELECTRONICS ENGINEERING

MES COLLEGE OF ENGINEERING

ACKNOWLEDGMENT

We are grateful to ***Dr. Rahumathunza I***, Principal, MES College of Engineering, Kuttippuram, for providing the right atmosphere to conduct this Project. I would like to extend our sincere gratitude to ***Dr. Nafeesa K***, Head of the Department, EEE, MES College of Engineering, Kuttippuram.

We are deeply indebted to the Project coordinator ***Dr. Thasneem Fathima N K,*** Associate Professor, Department of Electrical and Electronics Engineering for her continued support.

It is with great pleasure that we express a deep gratitude to Project guide ***Dr. Maya G,*** Associate Professor, Department of Electrical and Electronics Engineering, for her guidance, supervision, encouragement and valuable advice in every phase.

We would like to thank all other faculty members and fellow students at MES College of Engineering, Kuttippuram for their warm friendship, support and help.

ABDUL BASID K.P

ASLAM C.V

MUSAAB

NABEEL THARI

ABSTRACT

The concept of Smart Home Energy Management System using Internet of things (IoT) and advanced technologies such as energy metre, load demand response, programmed operations and sensors has emerged as a potential solution to reduce energy consumption and support sustainable living. The system incorporates real-time monitoring and control of energy usage, allowing homeowners to optimise their energy consumption while minimising waste.

This project focuses on the development of a Smart Home Energy Management System that leverages advanced technologies to optimise energy consumption and promote sustainable living practices. The system comprises two key components: a simulation module dedicated to energy management with load demand response and energy optimization using Genetic Algorithms (GA), and a hardware implementation for real-time energy monitoring on the supply side. While these components operate independently without direct connection, they collectively aim to revolutionise residential energy management practices. The simulation segment explores energy optimization strategies through GA, enhancing the system's adaptability and efficiency in optimising energy consumption patterns. Concurrently, the hardware side provides real-time monitoring of energy usage on the supply side, enabling informed decision-making. By integrating advanced simulation techniques with hardware-based monitoring capabilities, the project seeks to enhance energy efficiency, reduce waste, and support a more sustainable future in residential energy consumption.

This Project presents a comprehensive overview of the Smart Home Energy Management System, its architecture, and the integration of advanced technologies that make it possible using the MATLAB simulation model and hardware model .

CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATION

Abbreviation	Definition
GA	Genetics Algorithm
IoT	Internet of Things
GSM	Global System for Mobile communication
AWS	Amazon Web Service
DR	Demand Response
DC	Direct Current
W	Watts
KW-h	Kilo-Watt per Hour
EMS	Energy Management System
EM	Energy Management
PV	Photovoltaic

CHAPTER 1
INTRODUCTION

1.1 BACKGROUND

Smart houses have developed as a viable technological application that can improve residents' energy efficiency, comfort, and convenience. With rising energy demand and a growing emphasis on sustainability, researchers and industry professionals have prioritised effective energy management in smart homes.

Load demand response is critical to managing energy use in smart homes because it allows for dynamic modifications to electricity usage based on external factors such as pricing, renewable energy availability, and grid conditions. By employing load demand response systems, homeowners can save money on energy, lower peak demand, and help to maintain grid stability and reliability.

Genetic algorithms (GAs) mimic natural selection and genetic variation, and can optimise energy consumption patterns in smart homes. Real-time monitoring of energy parameters like power and voltage informs homeowners about energy usage, aiding in decision-making for optimization. Hardware-based monitoring systems collect real-time energy data, while simulation tools test energy optimization algorithms virtually.

1.2 OBJECTIVES

This section outlines the objectives of the project. It clearly states what the project aims to achieve in this objective, to explore the effectiveness of genetic algorithms in optimising energy management for smart homes using simulation-based approaches, focusing on load demand response strategies & Hardware-based monitoring systems enabling the collection and analysis of real-time energy data, facilitating proactive energy management practices.

1.3 SCOPE

This project aims to optimise energy management in a smart home by implementing a genetic algorithm approach. It focuses on integrating solar power to reduce grid electricity dependence and costs. Load demand response will also be used to adjust electricity usage based on pricing and grid conditions. Real-time energy monitoring will analyse consumption patterns and optimise power source utilisation. Cost analysis and optimization will play a crucial role in lowering overall energy costs through more efficient scheduling of power sources. Hardware implementation will also be considered in the study.

1.4 SIGNIFICANCE OF THE STUDY

The significance of the study lies in its potential to advance energy management practices in smart homes through the application of genetic algorithms and real-time energy monitoring. The project's goal is to minimise energy expenditures, improve grid stability, and promote sustainable energy practices by optimising energy strategies using renewable sources such as solar power and load demand response systems. The findings may be useful for homes, energy providers, and legislators looking to increase energy efficiency, make informed decisions based on real-time data, and contribute to a more sustainable energy future. Also, the project's focus on cost-effective energy management in smart homes emphasises its importance in meeting the growing need for energy efficiency and sustainability in residential settings.

CHAPTER 2
LITERATURE REVIEW

[1] Nwulu, N. I. ,Xiaohua Xia (2016). "Optimal dispatch for a microgrid incorporating renewables and demand response. " Renewable Energy, 99, 1123-1134. http://dx.doi.org/10.1016/j.renene.2016.08.026

The paper discusses the optimization of the dispatch of a microgrid that includes renewable energy sources and demand response. The author explores the challenges and opportunities associated with integrating renewable energy sources, such as solar and wind, into a microgrid system. The paper presents a mathematical model and optimization algorithm to determine the optimal dispatch strategy that minimises the overall cost of operation while considering the variability of renewable energy generation and the potential for demand response. The results of the study provide valuable insights into the efficient management of microgrids with renewable energy sources and demand response capabilities.

[2] Elkholy, M.H.; Senjyu, T.; Lotfy, M.E.; Elgarhy, A.; Ali, N.S.; Gaafar, T.S. " Design and Implementation of a Real-Time Smart Home Management System Considering Energy Saving" Sustainability (2022), 14, 13840. http://doi.org/10.3390/su142113840

The paper focuses on the development and implementation of a real-time smart home management system that prioritises energy-saving strategies. The authors discuss the design and architecture of the system, as well as the integration of various components such as smart metres, energy storage systems, and renewable energy sources. The paper also presents the results of a case study or experiment to demonstrate the effectiveness of the system in optimising energy consumption and reducing electricity costs in a residential setting. The research contributes to the field of sustainable energy management in smart homes.

[3] Munoz, O.; Ruelas, A.; Rosales, P.; Acuña, A.; Suastegui, A.; Lara, F. " Design and Development of an IoT Smart Meter with Load Control for Home Energy Management Systems." Sensors 2022, 22, 7536. https://doi.org/10.3390/s22197536

The paper discusses the architecture and functionality of the IoT smart metre, which includes features such as real-time energy monitoring, load control, and communication capabilities. The smart metre allows homeowners to monitor and control their energy usage remotely through a mobile application or web interface. The authors also present the results of experiments and simulations conducted to evaluate the performance and effectiveness of the IoT smart metre. They demonstrate how the load control feature can be utilised to optimise energy consumption, reduce peak loads, and achieve energy savings. Overall, the research contributes to the development of smart home energy management systems by providing a practical and efficient solution for monitoring and controlling energy usage at the household level.

[4] Ramin Torkan, Adrian Ilinca, Milad Ghorbanzadeh, "A genetic algorithm optimization approach for smart energy management of microgrids", Renewable Energy, Volume 197, 2022, Pages 852-863, ISSN 0960-1481, https://doi.org/10.1016/j.renene.2022.07.055.

The paper discusses the use of genetic algorithms as an optimization approach for smart energy management in microgrids. Microgrids are localised energy systems that can operate independently or in conjunction with the main power grid. Efficient energy management within microgrids is essential to ensure optimal utilisation of renewable energy sources, storage systems, and demand-side resources. They apply genetic algorithms to optimise energy management strategies within microgrids. This involves determining the optimal scheduling of energy generation, storage, and consumption to minimise costs, reduce emissions, and enhance the overall performance of the microgrid. The research contributes to the field of smart energy management by demonstrating the effectiveness of genetic algorithms in addressing the challenges of optimising energy systems in microgrids. By leveraging genetic algorithms, microgrid operators can improve the efficiency, reliability, and sustainability of their energy management practices.

[5] Arabali, A., Ghofrani, M., Etezadi-Amoli, M., Fadali, M. S., & Baghzouz, Y. (2013). "Genetic-Algorithm-Based Optimization Approach for Energy Management". IEEE Transactions on Power Delivery, DOI: 10.1109/TPWRD.2012.2219598

The article presents a study on utilising genetic algorithms for optimising energy management systems. Genetic algorithms are computational methods inspired by natural selection and genetics that can efficiently solve complex optimization problems. In this research, the authors propose an approach that leverages genetic algorithms to optimise energy management processes. By applying this methodology, energy systems can be optimised to enhance efficiency, reduce costs, and improve overall performance. The study aims to address the challenges faced in energy management by providing a systematic and effective optimization solution. This research contributes to the field of energy management by introducing a novel optimization approach that can potentially lead to significant advancements in energy efficiency and sustainability.

[6] R. Govindarajan, S. Meikandasivam and D. Vijayakumar, 2019." Low Cost Arduino Based Smart Energy Monitoring System Using Internet of Things". Journal of Engineering and Applied Sciences, 14: 170-177. DOI: 10.36478/jeasci.2019.170.177

The paper focuses on the development of a smart energy monitoring system that utilises Arduino, an open-source electronics platform, and the Internet of Things (IoT) technology. The authors present a cost-effective solution for monitoring energy consumption and optimising energy usage in various applications. The study contributes to the field of smart energy management by introducing a scalable and accessible monitoring system that empowers users to track and control their energy usage effectively. The use of Arduino and IoT in energy monitoring showcases the potential for innovative solutions to address energy efficiency challenges and promote sustainable practices.

CHAPTER 3
METHODOLOGY

3.1 ENERGY MANAGEMENT WITH LOAD DEMAND RESPONSE & ENERGY OPTIMIZATION USING GENETIC ALGORITHM

The methodology used involves energy management for a smart home scenario using load demand response & energy optimisation using genetic algorithm(GA).

The energy management is performed based on peak and non-peak hour considerations, leveraging solar power generation and battery storage. The factors considered for energy management is as follows:

[1] Load Demand Response:

Load demand response(DR) is a key aspect of energy management optimization, where the electricity consumption of appliances and devices is adjusted based on external factors such as pricing signals or grid conditions. During peak hours, DR mechanisms can reduce the load by shifting or curtailing energy usage, thereby alleviating strain on the grid and avoiding high electricity costs. Non-peak hours may allow for more flexible energy consumption patterns to optimise energy efficiency and cost savings.

[2] Solar Power Generation:

Solar power generation plays a crucial role in smart home energy management, offering a sustainable and renewable source of electricity. Photovoltaic (PV) systems installed on rooftops can harness solar energy to power the home and even feed excess energy back to the grid. Integrating solar power generation into energy optimization strategies enables the smart home to leverage clean energy resources and reduce reliance on grid electricity, especially during sunny hours.

[3] Battery Storage:

Battery storage systems complement solar power generation by storing excess energy generated during peak solar hours for later use when sunlight is insufficient. Batteries can store energy during off-peak periods when electricity prices are low and discharge it during peak hours to offset grid electricity consumption. Efficient

management of battery storage enhances energy resilience, reduces peak demand, and optimises cost savings in smart home energy systems.

[4] Energy management Algorithm:

The algorithm used in smart home energy management considers factors such as load profiles, solar power generation, battery capacity, and demand response strategies. By formulating mathematical models and algorithms, the management process aims to dynamically allocate energy from solar, grid, and battery sources to meet energy demand, minimise costs, and maximise renewable energy utilisation. Peak and non-peak hour strategies are employed to tailor energy management decisions based on varying electricity requirements and pricing dynamics.

[5] Cost-Efficient Strategies:

Cost-efficient energy management strategies focus on energy management of solar power, grid electricity, and battery storage to minimise electricity costs while maintaining energy reliability. By calculating the costs associated with energy consumption from different sources and applying intelligent decision-making logic, smart homes can achieve optimal energy utilisation, reduce overall expenses, and contribute to a more sustainable energy ecosystem.

Energy optimization is a critical process that aims to efficiently manage energy resources to minimise costs, reduce waste, and promote sustainability. This involves the strategic allocation of energy from renewable (such as solar) and non-renewable (like grid electricity) sources to meet energy demand effectively. By utilising advanced optimization techniques such as mathematical models, heuristic algorithms like Genetic Algorithm(GA), and machine learning approaches, energy systems can be optimised to operate more efficiently. Energy optimization finds applications in smart homes, industrial systems, and transportation sectors, enabling cost savings, environmental impact reduction, and improved reliability of energy supply. Despite challenges like complexity and data management, future trends point towards the integration of smart technologies and decentralised energy systems to further enhance energy optimization practices and contribute to a sustainable energy future.

In this project, we use GA into energy management systems. The optimization process can dynamically allocate power from various sources such as solar, grid, and battery to minimise costs and meet energy demands effectively. The methodology outlined in the above code theory utilises GA to iteratively evolve solutions that optimise energy consumption in a smart home setting. Through the principles of natural selection and genetic

operations like selection, crossover, and mutation, GA can efficiently search and find near-optimal solutions in the vast solution space of energy management. This adaptive and robust nature of GA makes it well-suited for tackling the challenges of energy optimization, enabling households and organisations to make informed decisions for sustainable energy usage and cost savings. The factors consider in GA follows:

[1] Objective Function Definition

The first step in the methodology involves defining an objective function that quantifies the energy consumption pattern that needs to be minimised within the smart home system. This function serves as the benchmark for evaluating the effectiveness of different energy consumption strategies.

[2] Setting Genetic Algorithm Parameters

Prior to running the Genetic Algorithm, specific parameters such as the population size, number of generations, crossover probability, and mutation rate are established. These parameters dictate the behaviour and performance of the Genetic Algorithm during the optimization process.

[3] Initialization of Population

To initiate the optimization process, an initial population of potential solutions is generated randomly. Each solution in this population represents a distinct energy consumption strategy that the Genetic Algorithm will iteratively evaluate and improve upon.

[4] Genetic Algorithm Operations

The Genetic Algorithm executes a series of operations, including selection, crossover, and mutation, to evolve the population of solutions over multiple generations. These operations mimic the principles of natural selection and genetic variation to gradually enhance the quality of solutions with respect to the objective function.

[5] Fitness Evaluation

After each generation, the fitness of each solution in the population is assessed by evaluating its performance against the defined objective function. This step determines how well each solution aligns with the optimization goal of minimising energy consumption in the smart home system.

[6] Selection of Fittest Individuals

Through a selection process based on fitness values, the Genetic Algorithm identifies the fittest individuals in the population. These individuals, representing the

most promising solutions, are given a higher likelihood of being selected for reproduction in the next generation.

[7] Iterative Evolution

The Genetic Algorithm iterates through multiple generations, continuously refining the population of solutions to converge towards a near-optimal energy consumption strategy. This iterative process aims to progressively improve the overall performance and efficiency of the smart home system.

[8] Optimal Solution Convergence

Ultimately, the Genetic Algorithm aims to converge to an optimal or near-optimal solution that minimises energy costs and effectively meets the energy demand of the smart home system. This convergence signifies the successful optimization of energy consumption within the smart home environment.

3.2 REAL TIME ENERGY MONITORING SYSTEM WITH IoT INTEGRATION

Real-time energy monitoring with IoT integration involves leveraging Internet of Things (IoT) technologies to enhance the monitoring and management of energy consumption. IoT devices, such as smart sensors and connected metres, are deployed to collect real-time energy data from various sources within a system or building. These devices communicate with each other and with cloud-based platforms to transmit the collected data for analysis and visualisation. Through IoT integration, users can remotely access and monitor their energy consumption data in real-time using web-based dashboards or mobile applications. IoT-enabled energy monitoring systems offer features such as automated alerts for abnormal energy usage, predictive analytics for forecasting energy trends, and the ability to control energy-consuming devices remotely.

IoT integration allows for the seamless integration of energy monitoring systems with other smart devices and systems, enabling a holistic approach to energy management. By combining real-time energy data with data from other IoT devices, such as smart thermostats, lighting controls, or appliances, users can optimise energy usage, reduce waste, and achieve greater efficiency in their operations.

The real-time energy monitoring with IoT integration provides users with actionable insights, improved decision-making capabilities, and greater control over their energy

consumption, leading to cost savings, sustainability benefits, and enhanced operational efficiency.

The basic factors involves in real time energy monitoring system with IoT integration is follows:

[1] Hardware Selection

In the initial phase, the focus is on selecting the appropriate hardware components essential for the energy monitoring system. This includes identifying sensors, microcontrollers, communication modules, and power supplies. Factors such as accuracy, reliability, compatibility, and cost play a crucial role in the selection process to ensure the system's effectiveness and efficiency.

[2] Sensor Integration

The next step involves integrating energy monitoring sensors into the hardware setup. Sensors like current sensors, voltage sensors, and power metres are essential for capturing real-time energy consumption data accurately. Proper calibration and placement of sensors are crucial to ensure precise data collection for monitoring energy usage effectively.

[3] Microcontroller Setup

Following sensor integration, configuring a microcontroller, such as Arduino or Raspberry Pi, is necessary to interface with the sensors and collect energy consumption data. Developing firmware or software routines to read sensor data and process it for transmission is a critical part of setting up the microcontroller for efficient data handling.

[4] Data Transmission

Implementing communication modules for real-time data transmission from the hardware system to the cloud service is essential. Communication options like Wi-Fi, Ethernet, or GSM enable seamless data transfer. Ensuring secure and reliable data transfer protocols is crucial to protect sensitive energy consumption information during transmission.

[5] Cloud Service Integration

Establishing a connection between the hardware system and a cloud service platform is vital for seamless data transfer and storage. Integration with cloud service providers like AWS IoT, Microsoft Azure, or Google Cloud Platform enables data storage, analysis, and visualisation for monitoring energy consumption trends effectively.

[6] Data Storage and Processing

Setting up cloud-based databases or storage solutions to store incoming energy consumption data is key to data management. Implementing data processing algorithms for analysing and aggregating data helps in generating insights and reports on energy usage patterns, facilitating informed decision-making regarding energy consumption.

[7] Real-time Monitoring Dashboard

Developing a user-friendly dashboard interface that connects to the cloud service is crucial for real-time monitoring of energy consumption metrics. Interactive visualisations, alerts, and notifications on the dashboard keep users informed about their energy usage trends, allowing them to make informed decisions for efficient energy management.

[8] Scalability and Reliability

Ensuring the scalability of the hardware setup and cloud service architecture is essential to accommodate the growing needs of monitored devices and users. Implementing redundancy and failover mechanisms enhances system reliability and uptime, ensuring continuous monitoring and data access.

[9] Security Measures

Implementing robust security measures at both hardware and cloud service levels is critical to safeguard sensitive energy data. Encryption, authentication, and access control mechanisms help protect data integrity and privacy, mitigating potential risks of unauthorised access or data breaches.

[10] Testing and Validation

Conducting thorough testing and validation of the hardware setup and cloud service integration is crucial to ensure proper functionality and performance. Real-world field tests validate the system's effectiveness in monitoring energy consumption in real-time, providing insights for further optimization and enhancements.

3.3 IMPLEMENTATION APPROACH

In this project, we use both simulation & hardware set up to provide different energy saving measures. The factors that involves in implementation approach are:

[1] Hardware Implementation

Begin by setting up the hardware components required for sensing energy data, such as metres, sensors, and communication devices. Ensure the hardware is configured to collect real-time energy consumption data accurately. Verify that the hardware setup can transmit data to a designated storage or processing unit seamlessly.

[2] Simulation Development

Develop a separate simulation model using simulation tools to replicate the behaviour of the energy monitoring system. Define parameters and inputs for the simulation based on the characteristics of the hardware components and energy consumption patterns. Implement algorithms and logic within the simulation model to simulate data collection, processing, and analysis.

[3] Cloud Service Integration

Integrate the hardware setup with a cloud service platform for data storage and analysis. Configure the cloud service to receive real-time energy data from the hardware components. Develop a user interface or dashboard to monitor energy consumption data stored in the cloud service, ensuring easy access and visualisation for users.

[4] Simulation Deployment

Deploy the simulation model in a separate environment to generate simulated energy consumption data. Validate that the simulation accurately reflects the behaviour of the actual energy monitoring system. Utilise the simulated data for testing and validation of algorithms and analysis techniques, ensuring that the simulation model aligns with real-world scenarios.

By implementing the simulation and hardware components separately and integration at the cloud service level, you can still achieve effective real-time energy monitoring with the ability to analyse both real-world and simulated data for improved decision-making and energy management.

CHAPTER 4
SIMULATION & HARDWARE IMPLEMENTATION

4.1 ENERGY MANAGEMENT WITH LOAD DEMAND RESPONSE

This simulation implementation using the energy management with load demand response is implemented using factors below:

1. **Algorithm Implemented**

   The algorithm(refer Fig.4.1) implemented focuses on optimising EM with cost-efficient savings. It involves initialising data and parameters, determining energy consumption, executing peak and non-peak hour energy management logic, calculating costs, visualising results, comparing total costs, and providing insights. The algorithm then iterates through hourly energy consumption calculations, distinguishing between peak and non-peak hours to apply specific energy management strategies accordingly.

   During peak hours, the algorithm focuses on maximising energy efficiency by considering factors like solar availability and load prioritisation. On the other hand, during non-peak hours, the emphasis shifts to balancing energy consumption between solar power, grid electricity, and battery storage to minimise costs effectively.

   Cost calculations are performed at each step to track and analyse the financial implications of energy management decisions. The algorithm visualises the results and cost savings achieved through energy optimization, providing a clear overview of the benefits derived from the implemented strategies.

   By comparing total costs before and after optimization, the algorithm offers a comprehensive evaluation of the effectiveness of the energy management approach. It concludes by presenting actionable insights and recommendations based on the analysed data, empowering users to make informed decisions for sustainable and cost-efficient energy utilisation.

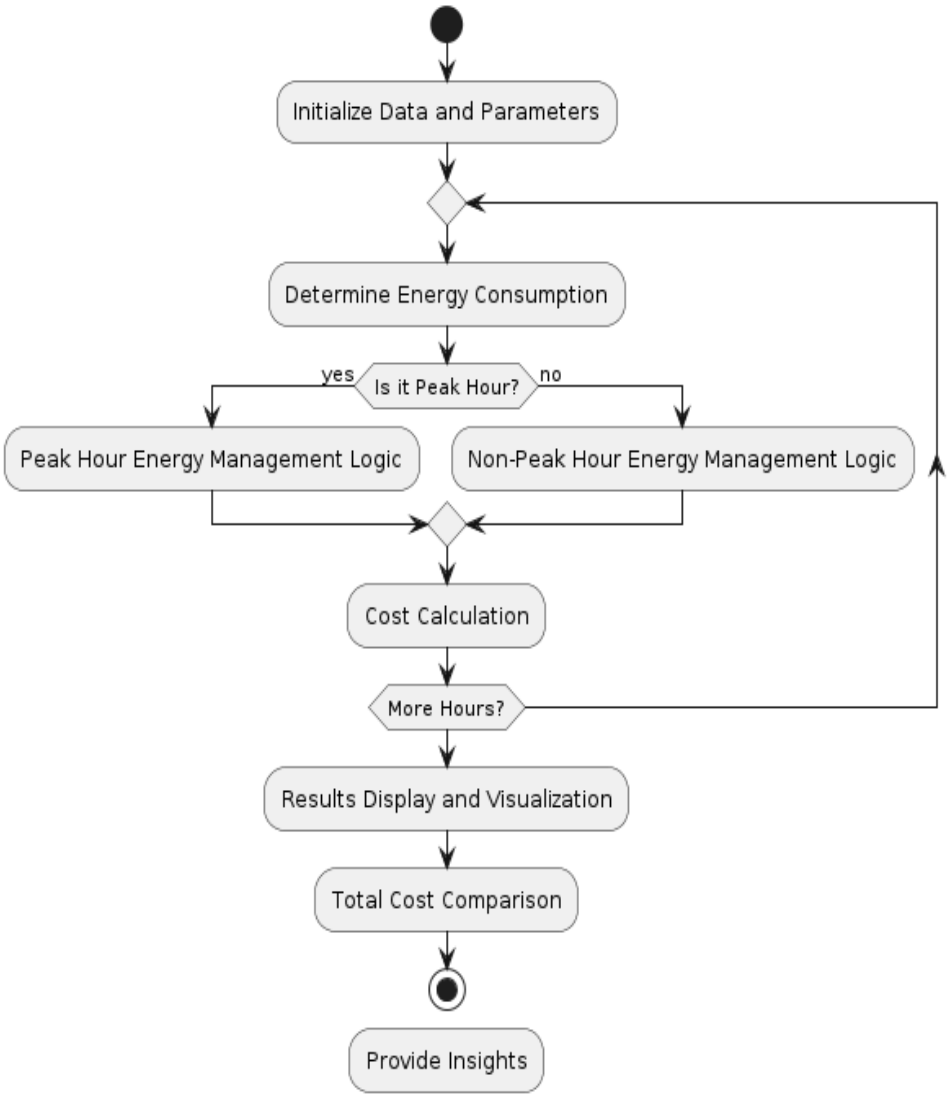

Fig. 4.1: Energy Management with Load Demand Response Algorithm Flowchart

2. Load Demand Response

Different load profiles (load1, load2, load3) are considered with DR factors for peak and non-peak hours.

a.For Peak Hour

$$\left[\text{Total Energy Consumed}_{\text{Peak}}(i) = \sum_{j=1}^{3} \text{Load}_j(i) \times \text{Demand Response Factor}_{\text{Peak}}\right]$$

b.For Non-Peak Hour

$$\left[\text{Total Energy Consumed}_{\text{Non-Peak}}(i) = \sum_{j=1}^{3} \text{Load}_j(i) \times \text{Demand Response Factor}_{\text{Non-Peak}}\right]$$

Here n=3 (load 1,load 2,load 3)

Load demand response factors adjust the original load to reduce electricity consumption during peak hours.

3. Solar Power Generation

Solar power generation profile (solar_power) is integrated into the energy management strategy. Solar power is used to offset energy consumption and reduce reliance on grid electricity.

4. Battery Storage

Battery capacity (battery_capacity) and charging/discharging efficiencies are defined. The battery is utilised to store excess solar energy for later use and cost savings.

5. Energy Management Logic

The energy management logic determines the energy allocation from solar, grid, and battery sources based on availability and consumption requirements during each hour (i)

a.Energy from Solar:

$$\left[\textbf{Total Energy from Solar(i)=min(Solar Power(i),Total Energy Consumed(i))}\right]$$

b.Energy from Grid:

$$\left[\textbf{TotalEnergyfromGrid(}i\textbf{)=max(0,TotalEnergyConsumed(}i\textbf{)}-\textbf{TotalEnergyfromSolar(}i\textbf{))}\right]$$

c.Energy from Battery:
The battery usage is calculated based on available storage and efficiencies.

Peak Hours Logic:

During peak hours (6-9 and 18-20), the total energy consumed is calculated using demand response factors. Energy management logic

determines energy allocation from solar, grid, and battery sources based on availability and consumption requirements. If solar power is sufficient, energy is sourced from solar; otherwise, battery storage and grid electricity are utilised efficiently.

Non-Peak Hours Logic

During non-peak hours, energy consumption is adjusted with demand response factors. Energy management logic allocates energy from solar and grid sources based on availability and consumption requirements. Energy deficit is calculated if total energy generated is less than total energy consumed.

6. Cost Calculations

Grid electricity prices (grid_price_peak, grid_price_non_peak), solar electricity price (solar_price), and battery charging cost (battery_cost) are considered.

Total costs for energy consumption from solar, grid, and battery are calculated for cost-efficient optimization.

a.Total Cost Solar:

$$[\text{Total Cost Solar}(i) = \text{Total Energy from Solar}(i) \times \text{Solar Price}]$$

b.Total Cost Grid:

$$[\text{Total Cost Grid}(i) = \text{Total Energy from Grid}(i) \times \text{Grid Price}]$$

c.Total Cost Battery:

[Total Cost Battery(i)=Battery Used(i)×Battery Cost]

d.Total Cost:

$$[\text{Total Cost}(i) = \text{Total Cost Solar}(i) + \text{Total Cost Grid}(i) + \text{Total Cost Battery}(i)]$$

7. **Energy Cost Management:**

Without Energy management:

Total energy consumed is calculated without energy management for peak hours using original load profiles.Total cost without energy management is determined based on grid electricity prices during peak hours.

With Energy management:

Total energy consumed is adjusted during peak hours with demand response factors for cost-efficient savings. Energy from solar, grid, and battery sources is allocated based on availability and cost considerations. Total costs for solar, grid, and battery sources are calculated for cost-efficient saving during peak hours.

a.Total Energy Consumed without Energy Management:

$$\left[\text{Total Energy Consumed}_{\text{NoEm}}(i) = \sum_{j=1}^{3} \text{Load}_j(i)\right]$$

b.Total Cost without Energy Management:

$$[\text{Total Cost}_{\text{NoEm}}(i) = \text{Total Energy Consumed}_{\text{NoEm}}(i) \times \text{Grid Price}_{\text{Peak}}]$$

4.2 ENERGY OPTIMIZATION USING GENETIC ALGORITHM

This simulation implementation using the energy optimization using GA is implemented using factors below:

1. **Algorithm Implemented:**

The energy management optimization algorithm(Refer Fig.4.2) employs a genetic algorithm to iteratively optimise power source allocations for efficient energy usage. Initially, the algorithm initialises parameters such as cost factors, power limits, and genetic algorithm settings. It then defines a fitness

function to evaluate the cost based on the distribution of power from solar, grid, and battery sources. The population is initialised with random solutions, and the genetic algorithm loop begins. Within each iteration, the algorithm evaluates the fitness of each solution, selects the best individuals, performs crossover and mutation operations to create new solutions, and updates the population. The algorithm tracks power source consumption throughout the optimization process and visualises the results. This iterative process continues until a termination condition is met, at which point the algorithm displays the optimised solution and the corresponding cost savings achieved. The algorithm's systematic approach enables the efficient management of energy resources to minimise costs and enhance sustainability.

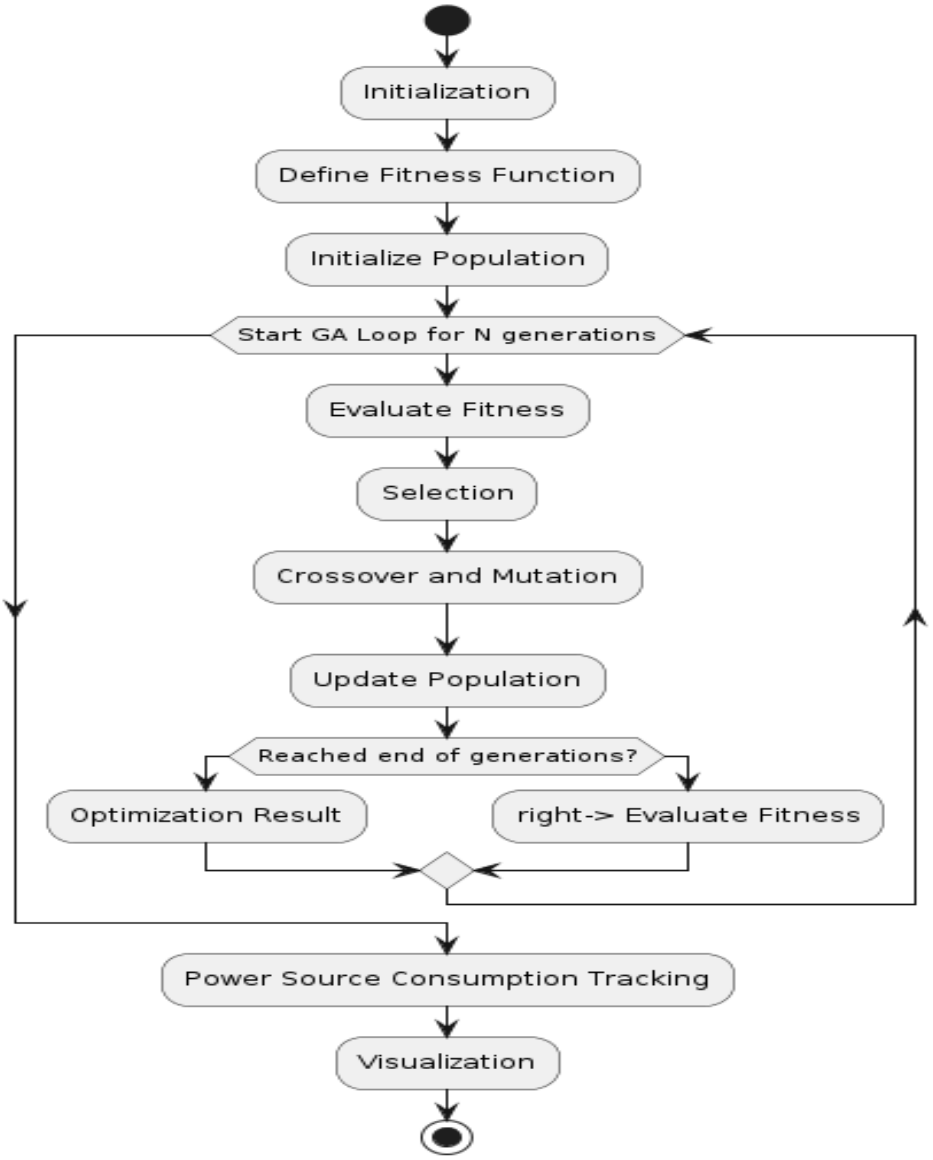

Fig. 4.2: Energy Optimization using GA Algorithm Flowchart

2. **Objective Function**

The objective is to minimise the total cost of energy consumption by optimising the allocation of power from solar, grid, and battery sources.

The objective function can be defined as:

$$[\text{Total Cost} = \text{Solar Usage} \times \text{Solar Price} + \text{Grid Usage} \times \text{Grid Price} + \text{Battery Usage} \times \text{Battery Cost}]$$

3. **Decision Variables**

The decision variables represent the power allocation from solar, grid, and battery sources:

$$[x = [\text{Solar Usage}, \text{Grid Usage}, \text{Battery Usage}]]$$

4. **Genetic Algorithm (GA)**

GA is used to find the optimal solution by evolving a population of potential solutions over multiple generations.

The GA involves the following steps:

a. Initialization

Initialise the population with random solutions represented by power values for solar, grid, and battery sources.

b.Fitness Function

Define a fitness function that evaluates the total cost based on the energy consumption from different sources.

$$[\text{Fitness Function}(x) = \text{Total Cost}]$$

c. Selection:

Select individuals from the population for the next generation based on their fitness values.

d. Crossover:

Perform crossover operations to create new solutions by combining characteristics of two parent solutions.

e.Mutation:

Introduce random changes to the solutions to maintain genetic diversity in the population.

f. Termination:

Repeat the selection, crossover, and mutation steps for a specified number of generations or until a termination condition is met.

5. **Optimization Results**

   After the GA optimization process, the best solution with the lowest total cost is identified.The optimised solution represents the power allocation from solar, grid, and battery sources that minimise energy costs.

6. **Energy Consumption Visualization**

   Load profiles, solar power generation, and power source consumption profiles during optimization can be visualised to analyse energy consumption patterns and strategies.

7. **Conclusion**

   The GA-based energy management optimization algorithm aims to find a cost-effective energy consumption strategy by dynamically allocating power from renewable (solar) and conventional (grid, battery) sources in a smart home environment.

- Other Mathematical equation used

1. Total Energy Consumed and Cost Calculation

The total energy consumed(E_{total}) is calculated as the sum of energy from solar, grid, and battery sources:

$$[E_{\text{total}} = E_{\text{solar}} + E_{\text{grid}} + E_{\text{battery}}]$$

The total cost (C_{total}) associated with energy consumption is determined by considering the costs of using solar power, grid electricity, and battery charging:

$$[C_{\text{total}} = \max(0, C_{\text{solar}} + C_{\text{grid}} + C_{\text{battery}})]$$

Where:

(E_{solar}), (E_{grid}), and (E_{battery}) represent the energy consumed from solar, grid, and battery sources, respectively.

(C_{solar}), (C_{grid}), and ($C_{battery}$) denote the costs associated with solar power, grid electricity, and battery charging, respectively.

2. Fitness Function Evaluation

The fitness function evaluates the total cost based on the decision variables representing the power allocation from different sources:

$$\text{Fitness} = \text{energy\_cost\_function}(x, \text{solar\_power}, \text{load1}, \text{load2}, \text{load3}, \text{grid\_price\_peak}, \text{grid\_price\_non\_peak}, \text{solar\_price}, \text{battery\_cost})$$

3. Optimization Objective

The main objective of the optimization process is to minimise the total cost (C_{total}) by finding the best combination of solar, grid, and battery power allocations (E_{solar}), (E_{grid}), and ($E_{battery}$).

4.3 SIMULATION RESULT

On doing the simulation of Energy management with load demand response , we obtained:

```
>> EMSWLD
Total Energy Consumed: 5.60 kWh
Total Energy Generated: 3.00 kWh
Total Energy Deficit: 2.60 kWh
Energy from Solar: 3.00 kWh
Energy from Grid: 2.60 kWh

Total Cost without Energy Management: Rs2475.00
Total Cost after Energy Management: Rs973.12
```

Fig 4.3(a) Output of Energy Management with Load demand response

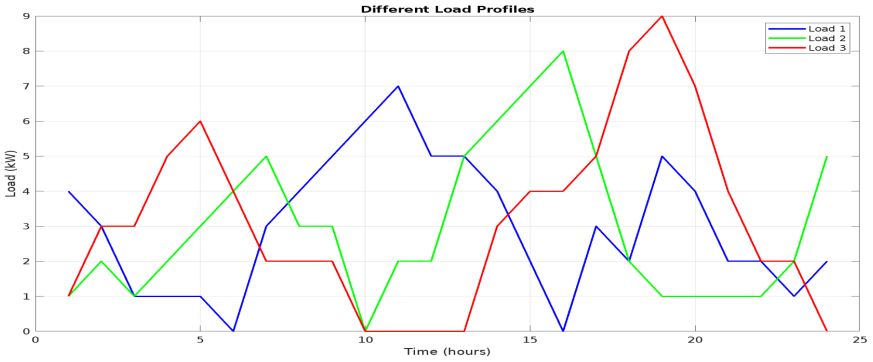

Fig 4.3(b) Load Profile of EM with Load DR

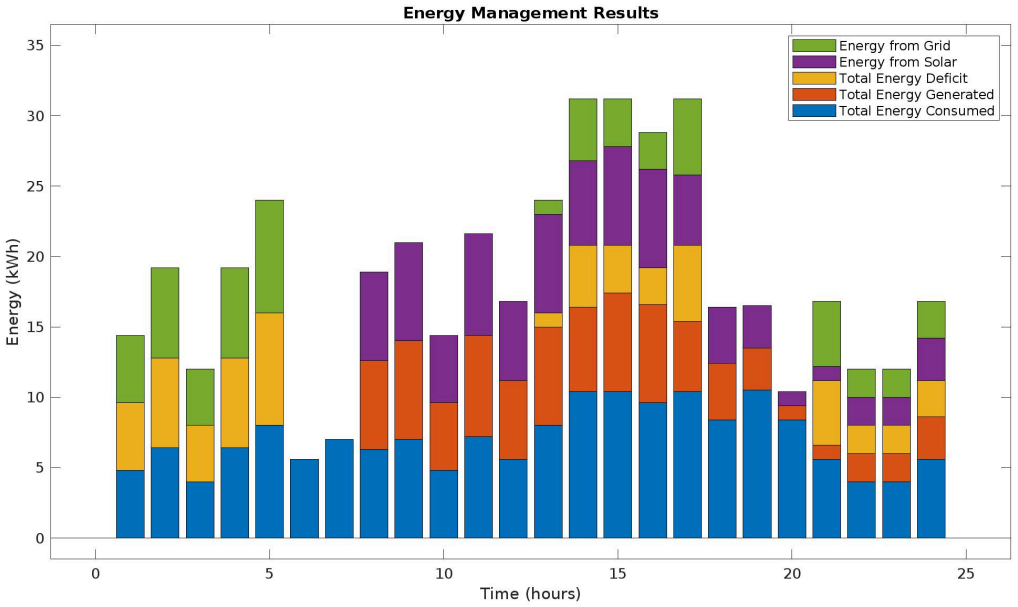

Fig 4.3(c) Energy detail of EM with Load DR

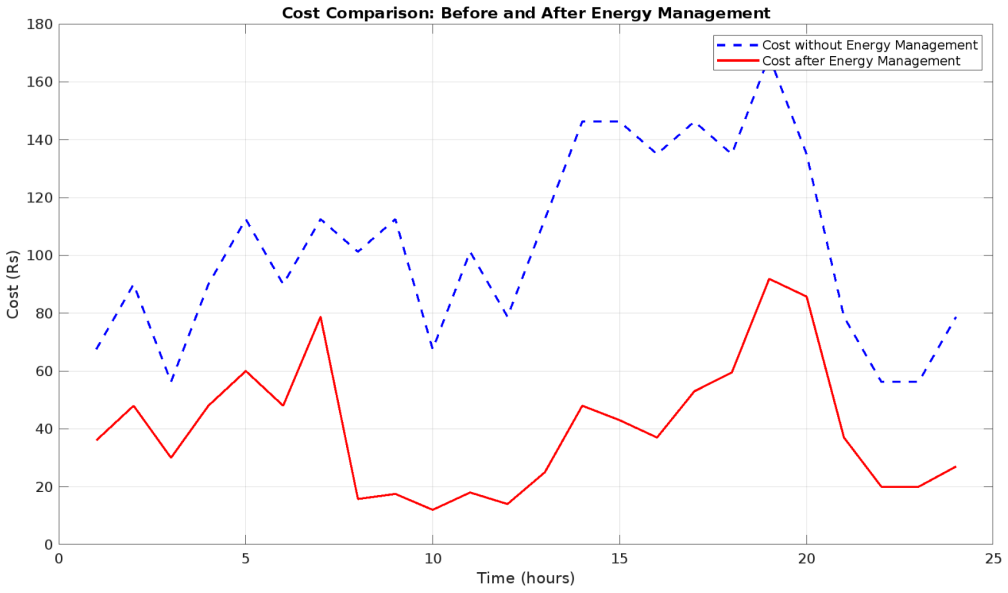

Fig 4.3(d) Cost Comparison of EM with Load DR

It can be concluded from these results above obtained by doing simulation in matlab that, the Energy consumed(This is the sum of energy consumed by all loads during the day), Energy generated(This represents the sum of energy generated by renewable sources and utilised by loads) , energy deficient(This is the difference between the total energy consumed

and the total energy generated), total energy from solar , total energy from grid , total cost with & without energy management (Refer Fig.4.3(a)) values are provided along with the graphical representation of output obtained of energy detail (Refer Fig.4.3(c)). And with the output graphical representation of load profile(Refer Fig.4.3(b)) such that the loads operate Power in x-axis & Time in 24 hours in y-axis providing the energy consumption of load in 24 hour time frame. Along with the graphical representation of output obtained of energy detail (Refer Fig.4.3(c)). The cost comparison (Refer Fig 4.3(d)) between without & after Energy management is also shown as the comparison detail.

In conclusion to the simulation of Energy management with load demand response, the cost has been reduced by 60.69% (i.e from Rs. 2475 to Rs. 973.12) by implementing the Energy management developed.

On doing the simulation of Energy optimization using GA, we obtained:

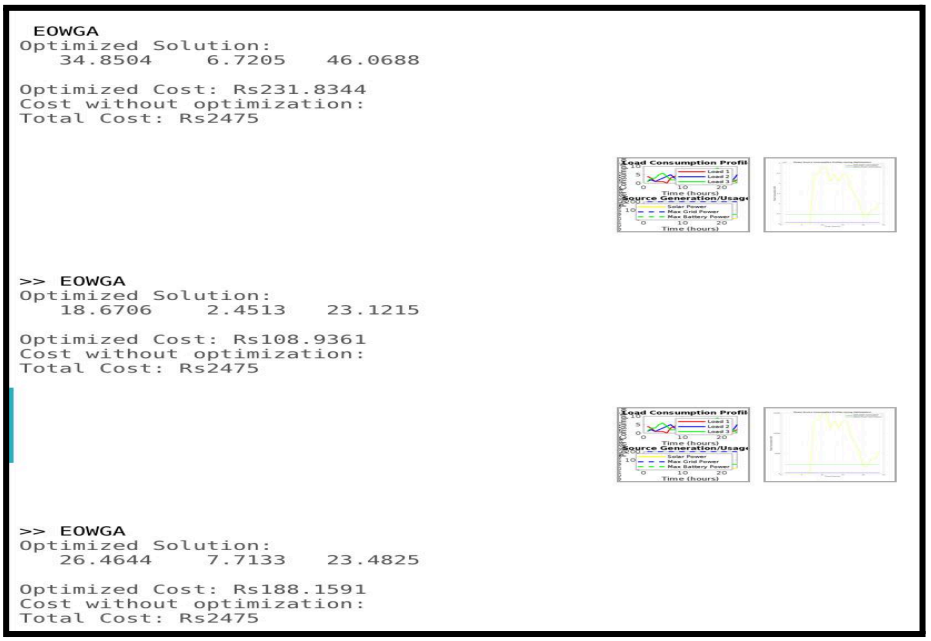

Fig 4.3(e) Output of Energy Optimization using GA

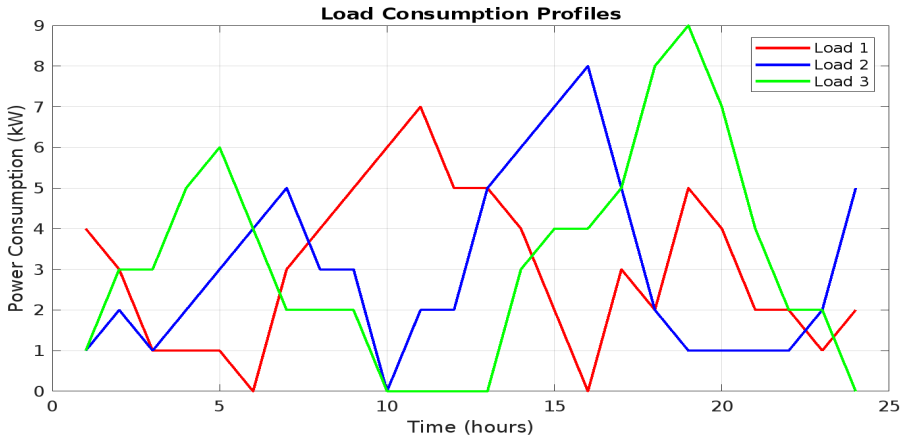

Fig 4.3(f) Load Profile of Energy Optimization using GA

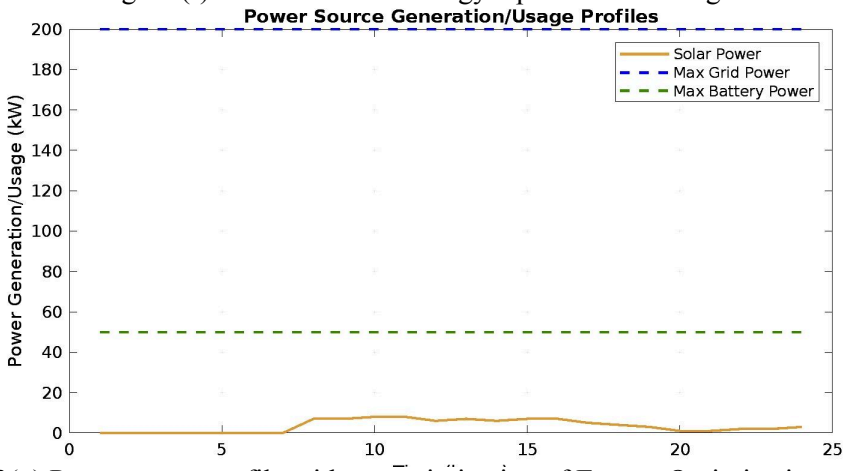

Fig 4.3(g) Power usage profile without optimization of Energy Optimization using GA

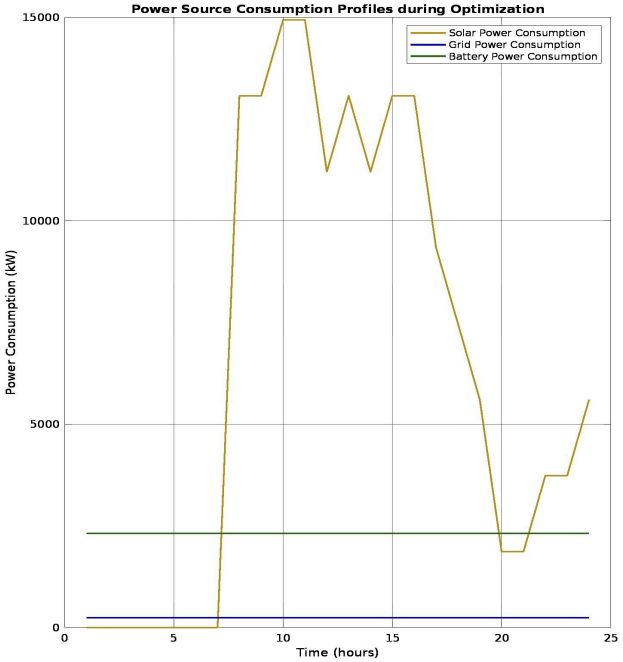

Fig 4.3(h) Power usage profile with optimization of Energy Optimization using GA

It can be concluded from these results above obtained by doing simulation in matlab that, The optimised solution(Refer Fig.4.3(e)) represents the optimal power allocations from solar, grid, and battery sources that result in the minimum energy cost. And with the output graphical representation of load profile(Refer Fig.4.3(f)) such that the loads operate Power in x-axis & Time in 24 hours in y-axis providing the energy consumption of load in 24 hour time frame. Along with the graphical representation of output obtained from the Power usage profile without optimization (Refer Fig.4.3(g)). The power usage profile with optimization (Refer Fig 4.3(h)) is also obtained.

The results demonstrate that the minimum cost achieved through energy optimization using GA has been significantly reduced by 95.6% (from Rs. 2475 to Rs. 108.93) compared to the previous energy management implementation. By implementing GA optimization, the cost reduction is much more substantial, indicating its efficiency in minimising costs.

Comparing both simulation results, it is evident that energy optimization using GA offers greater cost reduction benefits. Specifically, in the same load profile scenario, energy management with load demand response resulted in a 60.69% reduction from the non-optimized case, while GA optimization achieved a remarkable 95.6% reduction. This significant difference underscores the superior efficiency and cost minimization capabilities of energy optimization using GA.

4.4 REAL TIME ENERGY MONITORING SYSTEM WITH IoT INTEGRATION IMPLEMENTATION

The implementation of the real-time monitoring system for tracking power, voltage, and current on the supply side (solar, battery) and monitoring the loads consisting of a charger, bulb, and fan involves integrating specific hardware components and designing a robust system architecture.

The hardware setup includes sensors for measuring power, voltage, and current, strategically placed to capture accurate data from the supply side sources and the load components. These sensors are interfaced with microcontrollers or processing units responsible for data acquisition, processing, and communication. The microcontrollers collect sensor data in real-time and transmit it to the monitoring interface for display and analysis.

Power supply design is crucial to ensure continuous operation of the monitoring system, with appropriate power sources and management solutions in place. Safety considerations are paramount, with compliance to industry standards and regulations being a key focus when dealing with electrical measurements.

The monitoring interface is designed to provide real-time visualisation of the collected data, enabling users to track the performance of the supply side sources and monitor the power consumption of the loads. The interface includes features for data visualisation, alerts for abnormal readings, and historical data tracking for analysis and decision-making.

Testing and validation procedures are conducted to calibrate sensors, verify data accuracy, and validate the functionality of the real-time monitoring system. Maintenance guidelines are established to ensure the system operates efficiently, with provisions for troubleshooting, calibration, and software updates as needed.

The implementation of this hardware system involves a comprehensive approach that integrates sensors, microcontrollers, power management solutions, communication protocols, and a user-friendly monitoring interface. By following the outlined hardware design and specifications, the real-time monitoring system can effectively track and analyse power, voltage, and current data, providing valuable insights for efficient energy management..

4.4.1. DESIGN AND SPECIFICATION

LOAD	SPECIFICATION	QUANTITY	TOTAL WATTAGE
BULB	9 W	1	9 W
FAN (EXHAUST)	5w	1	5 W
CHARGER	20 w	1	20 W
CONTROLLER	2 W(max)	1	2 W
TOTAL WATTAGE			36 W

Fig 4.4.1. Load Specification

The hardware design for monitoring the power consumption of a 9W bulb, 5W fan, and 20W charger (Refer Fig 4.4.1) involves a systematic approach to ensure accurate measurement and efficient monitoring of energy usage. By selecting appropriate sensors for current, voltage, and power, the system can precisely capture the energy consumption of each load in real-time. These sensors play a crucial role in providing data that is essential for effective energy management.

Integrating these sensors with a microcontroller allows for data processing and analysis, enabling the system to calculate power consumption, monitor voltage levels, and track current readings. The microcontroller acts as the central processing unit that controls the flow of data and interfaces with a display unit to present the gathered information in a user-friendly manner. This real-time display of power-related data empowers users to make informed decisions regarding energy usage and optimization.

Ensuring a stable power supply is fundamental to the system's reliability and continuous operation. By selecting a power source that can adequately support the monitoring hardware and the loads being monitored, the system can function optimally without interruptions. Additionally, incorporating safety features such as fuses, isolation components, and proper grounding safeguards both the system components and users from potential electrical risks.

Calibrating the sensors is a critical step in guaranteeing the accuracy of the measurements obtained. Calibration procedures fine-tune the sensors to provide precise data, which is essential for making informed decisions based on the energy usage patterns of the monitored loads. Thorough testing of the system under various load conditions validates its performance and ensures its reliability in different scenarios. By housing the hardware

components in an enclosure, protection from environmental factors is provided, enhancing the longevity and durability of the system. Moreover, organising the setup and considering mounting options facilitate ease of installation and maintenance, contributing to the system's overall efficiency and user-friendliness.

In selecting a 40-watt solar panel and a 150-watt battery for the system, standard values were chosen to ensure sufficient power generation and storage capacity to support the monitoring system effectively.

4.4.2. BLOCK DIAGRAM & CIRCUIT SET UP

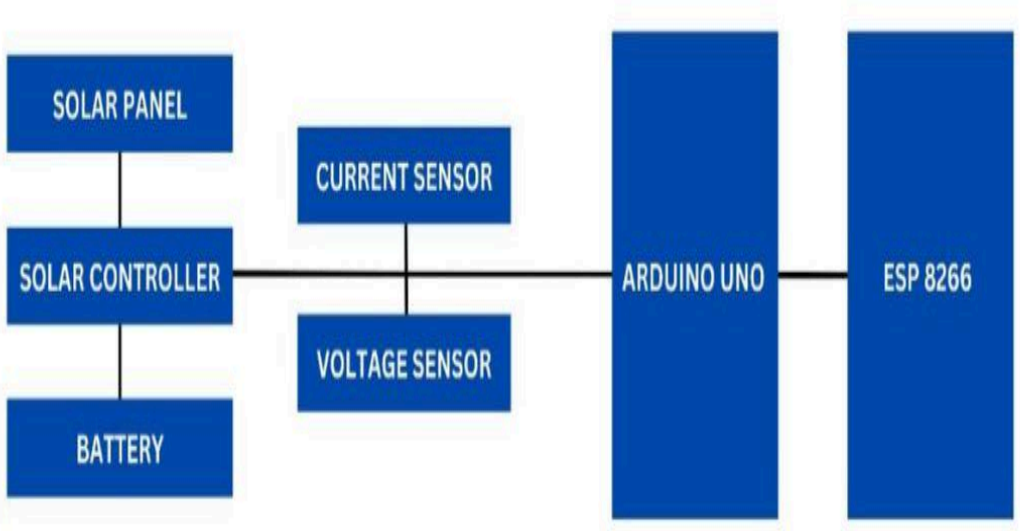

Fig 4.4.2.(a) Block diagram of energy real-time monitoring system.

The following description outlines the block diagram(Refer Fig 4.4.2(a)) stages:

1. Solar Panel, Solar Controller, and Battery Stage:

The first stage of the block diagram comprises the solar panel, solar controller, and battery components. The solar panel captures sunlight and converts it into electrical energy. This energy is regulated and stored in the battery by the solar

controller. The battery serves as the power source for the system, providing energy for continuous operation, especially during low light conditions.

2. Current and Voltage Sensor Stage:

The next stage involves the current and voltage sensors, which are connected to the loads (9W bulb, 5W fan, and 20W charger) to measure the energy consumption accurately. The current sensor measures the current drawn by each load, while the voltage sensor monitors the voltage supplied to the loads. These sensors play a crucial role in providing real-time data on power consumption.

3. Arduino Microcontroller Stage:

Subsequently, the Arduino microcontroller stage is introduced in the block diagram. The Arduino board processes the data collected by the current and voltage sensors, calculates the power consumption of each load, and interfaces with other components of the system. It acts as the central processing unit that controls the flow of information and data processing.

4. ESP8266 Module Stage:

The final stage of the block diagram includes the ESP8266 module, which provides wireless communication capabilities to the system. The ESP8266 module enables data transmission to external devices or cloud services for remote monitoring and analysis. It facilitates the connectivity of the monitoring system to the internet, allowing users to access real-time power consumption data remotely.

By integrating these stages in the block diagram, the hardware system for monitoring power consumption effectively captures energy usage information, processes data accurately, and enables remote monitoring and control through wireless communication. This structured approach ensures a comprehensive and efficient design for monitoring the specified loads with the designated components.

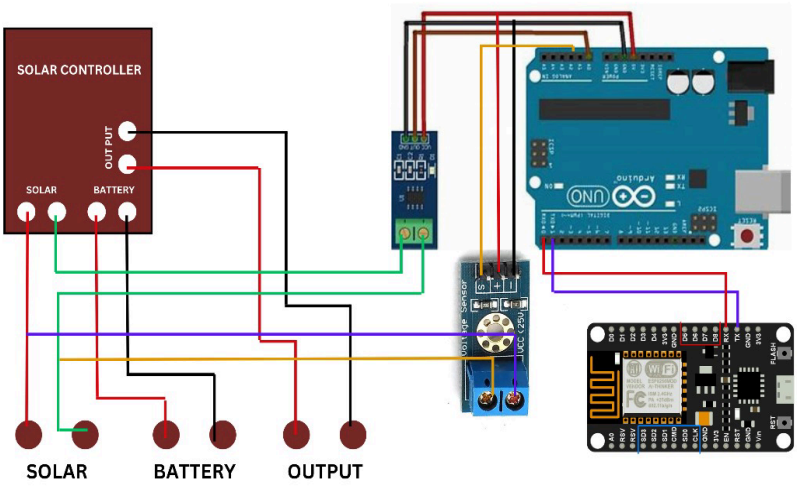

Fig 4.4.2.(b) Circuit set up of an energy real-time monitoring system.

The circuit setup (Refer 4.4.2 (b)) for monitoring the power consumption of a 9W bulb, 5W fan, and 20W charger involves the integration of various components to accurately measure and monitor energy usage. Here is a description of the circuit setup:

1. Solar Panel, Solar Controller, and Battery Setup:

The solar panel is connected to the solar controller to regulate the incoming solar energy and charge the battery. The solar controller ensures that the battery is charged efficiently and protects it from overcharging or deep discharging. The battery serves as the power source for the entire system, providing continuous energy supply.

2. Current and Voltage Sensors Integration:

Current sensors are placed in series with each load (9W bulb, 5W fan, and 20W charger) to measure the current flowing through them. Voltage sensors are connected in parallel to monitor the voltage levels supplied to the loads. These sensors provide crucial data for calculating power consumption accurately.

3. Arduino Microcontroller Connection:

The current and voltage sensor outputs are connected to the analog input pins of the Arduino microcontroller. The Arduino reads the sensor data, calculates the

power consumption of each load using Ohm's Law **(P = V x I)**, and processes the information for display or transmission.

4. ESP8266 Module Integration:

The Arduino communicates with the ESP8266 module via serial communication. The ESP8266 module facilitates wireless connectivity and data transmission to external devices or cloud services. It allows for remote monitoring and control of the power consumption data collected by the system.

By following this circuit setup description, the hardware system can effectively monitor the power of the supply side , provide accurate data for analysis, and enable remote monitoring capabilities for efficient energy management.

4.4.3. COMPONENTS DETAIL

1. Solar Panel

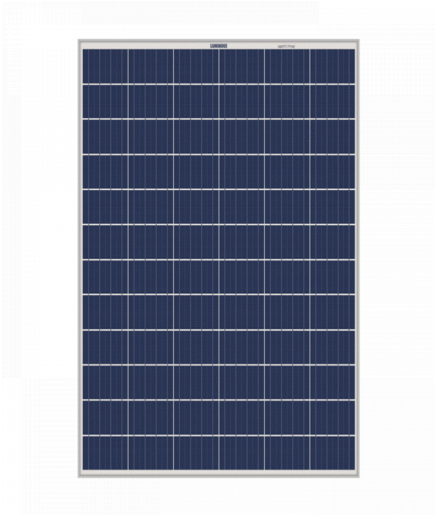

Fig 4.4.3.(a) Solar PV panel

In this setup, a 40W solar panel is used. The 40W solar panel is a compact and efficient photovoltaic module designed to convert sunlight into electrical energy. With a durable frame and high-quality solar cells, it can generate up to 40 watts of power under standard conditions. Suitable for off-grid applications, the panel offers reliable performance and easy installation options, making it ideal for powering small-scale

systems like remote monitoring setups, RVs, boats, and electronic devices. Its versatility, efficiency, and sustainable energy production make it a valuable component for environmentally friendly power generation.

2. Solar charge controller

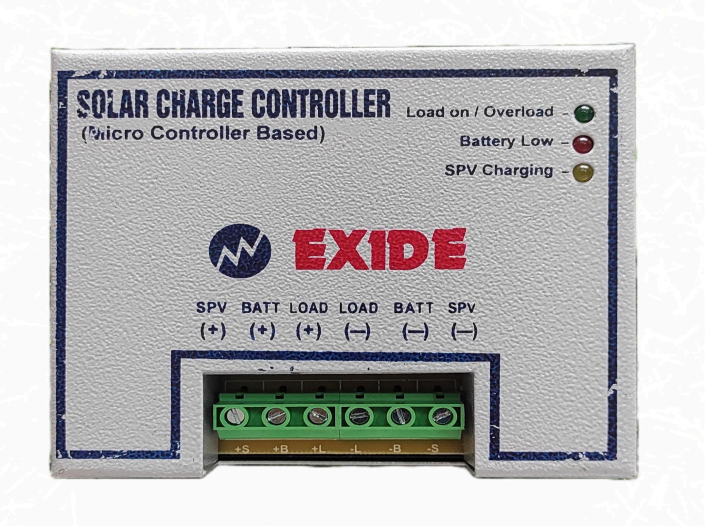

Fig 4.4.3.(b) Exide Solar charge controller

The Exide solar charge controller is a crucial component in solar power systems, designed to regulate the flow of electricity from the solar panels to the battery bank. This device ensures efficient charging and prevents overcharging or deep discharging of the batteries, thereby extending their lifespan. The Exide solar charge controller typically features advanced technology such as pulse width modulation (PWM) or maximum power point tracking (MPPT) to optimise energy conversion and maximise power output. With built-in protection mechanisms against short circuits, reverse polarity, and overloading, the Exide solar charge controller offers reliable and safe operation. Its user-friendly interface and robust construction make it suitable for a wide range of solar applications, providing efficient energy management and enhancing the overall performance of solar power systems.

3. Battery

Fig 4.4.3.(c) Lead Acid battery

In this circuit setup, we used a 150W lead acid battery. The 150W battery serves as a vital energy storage component in various applications, providing a reliable power source for off-grid systems, backup power solutions, and portable devices. This battery typically features a capacity to store up to 150 watt-hours of energy, allowing for sustained power supply during periods of low sunlight or grid outages. With a compact and durable design, the 150W battery is often equipped with advanced lithium-ion technology, offering high energy density, long cycle life, and lightweight portability. It is commonly used in conjunction with solar panels, inverters, and charge controllers to create efficient and sustainable energy systems. The 150W battery's versatility, performance, and capacity make it a valuable asset for powering a wide range of electronic devices, appliances, and equipment, both indoors and outdoors.

4. Inverter

Fig 4.4.3.(d) Battery Inverter.

The 150W inverter is a compact and versatile device designed to convert DC power from a battery or solar panel into AC power for running various electronic devices and appliances. With a power output of 150 watts, this inverter is suitable for powering small devices such as laptops, smartphones, lights, and other low-power electronics. Equipped with multiple safety features like overload protection, short circuit protection, and overheat protection, the 150W inverter ensures safe and reliable operation. Its lightweight and portable design make it ideal for use in off-grid applications, camping trips, emergency situations, and other situations where access to grid power is limited. The 150W inverter provides a convenient and efficient solution for converting DC to AC power, enabling users to stay connected and powered up wherever they go.

5. Current sensor

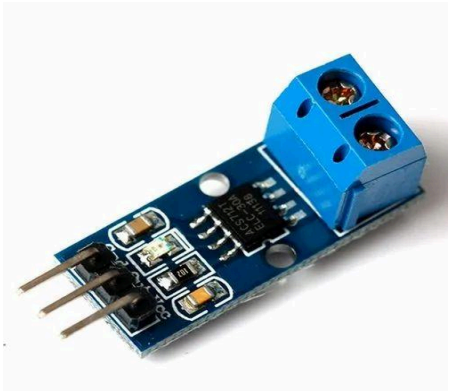

Fig 4.4.3.(e) Current sensor.

A current sensor is a device used to measure the flow of electric current in a circuit. It detects and quantifies the amount of current passing through a conductor without the need for direct electrical connection. Current sensors are essential components in various applications, including power monitoring, energy management, motor control, and safety systems. They provide valuable data on electrical currents, enabling precise monitoring, control, and protection of electrical systems. By accurately measuring current levels, these sensors help optimise energy usage, prevent overloads, and ensure the efficient operation of electrical equipment. With different types available, such as Hall-effect sensors, shunt resistors, and current transformers, current sensors offer versatility and reliability in monitoring and managing electrical currents in a wide range of industrial, commercial, and residential settings.

6. Voltage sensor

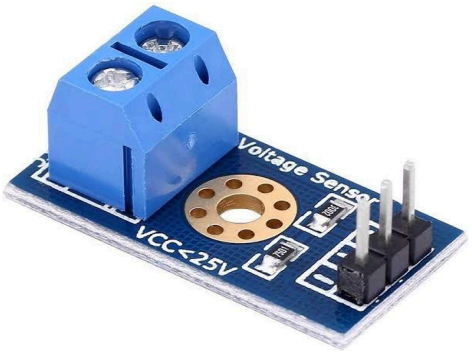

Fig 4.4.3.(f) Voltage sensor.

A voltage sensor is a device that is used to measure the electrical potential difference between two points in an electrical circuit. It detects and quantifies the voltage level present in a system, providing valuable information for monitoring, control, and protection purposes. Voltage sensors are essential components in a variety of applications, including power distribution systems, renewable energy systems, battery management systems, and electronic devices. By accurately measuring voltage levels, these sensors help ensure the safe and efficient operation of electrical equipment, prevent overvoltage conditions, and optimise energy usage. Voltage sensors come in various types, such as resistive voltage dividers, capacitive

voltage sensors, and electromagnetic field sensors, each offering specific advantages depending on the application requirements. Overall, voltage sensors play a critical role in electrical systems by providing real-time voltage measurements and enabling effective management of electrical power.

7. Microcontroller

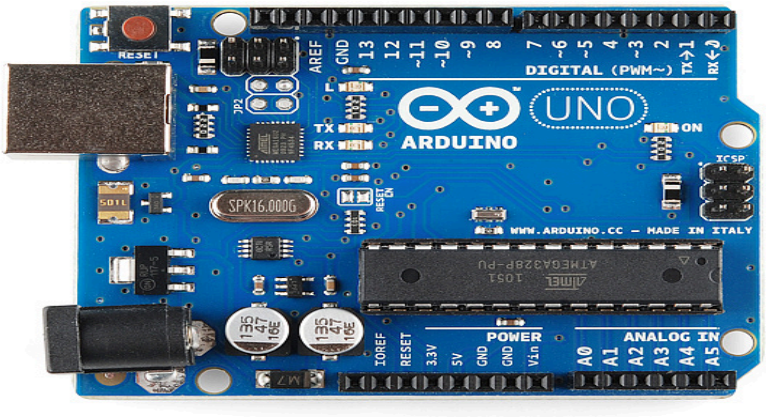

Fig 4.4.3.(g) Arduino Uno R3 ATmega328 Microcontroller.

In this setup, we used Arduino Uno R3 ATmega328. The Arduino ATmega328 is a microcontroller board based on the ATmega328P chip, which is the heart of the popular Arduino Uno board. The ATmega328P is a high-performance, low-power 8-bit AVR microcontroller with advanced RISC architecture, featuring 32KB of flash memory, 2KB of SRAM, and 1KB of EEPROM. This microcontroller is widely used in various embedded systems and DIY projects due to its versatility, ease of use, and extensive community support. The Arduino ATmega328 board provides a convenient platform for prototyping and developing electronic projects, offering a range of digital and analog input/output pins, PWM outputs, communication interfaces like UART, SPI, and I2C, and compatibility with a wide range of sensors, actuators, and shields. Users can program the ATmega328P using the Arduino IDE, which simplifies the development process with its user-friendly interface and extensive library of pre-written code examples. Overall, the Arduino ATmega328 is a versatile and powerful microcontroller board that is well-suited for a wide range of applications, from robotics and automation to IoT and wearable technology projects.

8. Wi-Fi Module

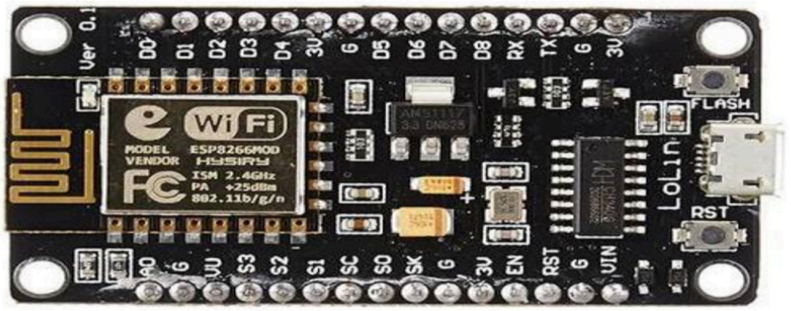

Fig 4.4.3.(h) ESP8266 Wi-Fi module.

In this setup, we used ESP8266 as a Wi-Fi module. The ESP8266 is a popular and versatile Wi-Fi module known for its low cost, small size, and powerful capabilities. It features a 32-bit microcontroller with integrated Wi-Fi connectivity, making it an ideal choice for IoT (Internet of Things) projects and wireless communication applications. The ESP8266 module supports both TCP/IP and UDP protocols, allowing for seamless integration with existing networks and cloud services. With a range of GPIO pins, SPI, I2C, and UART interfaces, the ESP8266 offers flexibility for connecting sensors, actuators, and other peripherals. It can be programmed using the Arduino IDE or other development platforms, enabling users to create custom firmware and applications tailored to their specific needs. The ESP8266 has gained popularity for its ability to provide wireless connectivity to a wide range of devices, from smart home gadgets and weather stations to industrial automation systems and remote monitoring solutions. Overall, the ESP8266 module has revolutionised the world of IoT by offering an affordable and feature-rich solution for adding Wi-Fi connectivity to a diverse array of projects.

9. Cloud integration

Fig 4.4.3.(i) Ubidots user interface.

In this setup, we used Ubidots as cloud software. Ubidots is a cloud integration software platform that enables users to easily connect, visualise, and analyse data from IoT devices and sensors in real-time. With Ubidots, users can securely transmit data from their devices to the cloud, where it can be visualised through customizable dashboards, graphs, and widgets. The platform offers a user-friendly interface and drag-and-drop tools for creating interactive visualisations, alerts, and notifications based on the data collected. Ubidots supports various communication protocols, such as HTTP, MQTT, and TCP, making it compatible with a wide range of IoT devices and hardware. Additionally, Ubidots provides robust data analytics features, including trend analysis, data export, and integration with third-party services like Zapier and IFTTT. With its scalability, reliability, and ease of use, Ubidots is a popular choice for businesses and developers looking to build IoT applications, monitor assets remotely, and leverage data-driven insights for decision-making.

4.4.4 WORKING DESCRIPTION

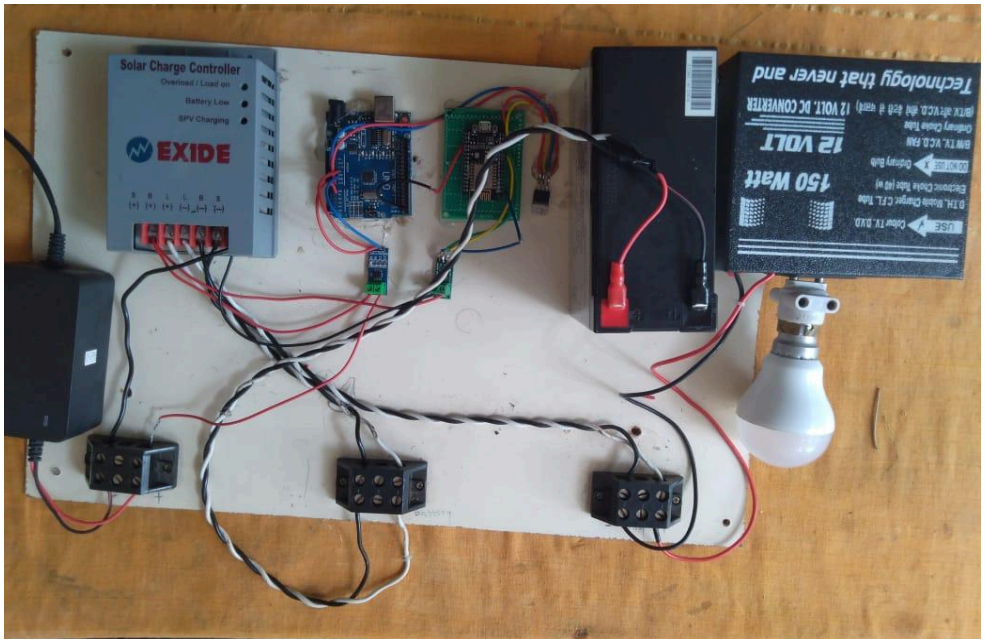

Fig 4.4.4. The Working Circuit set up

In a setup designed for real-time monitoring of solar and battery power generation, the integration of sensors, a microcontroller, and the Ubidots cloud service enables users to efficiently manage and monitor their renewable energy system. The system operates by collecting data from sensors that measure amperes and voltage output from solar panels and a battery bank. These sensors are connected to a microcontroller, such as an Arduino or ESP8266, which acts as the central processing unit for the data acquisition process.

The sensors continuously gather data at 5-second intervals, capturing real-time information on the performance of the solar panels and battery bank. The microcontroller processes this data and sends it to the Ubidots cloud service using either Wi-Fi or Ethernet connectivity. Once the data reaches the Ubidots platform, it is stored and displayed on a customizable dashboard that provides users with a comprehensive overview of their renewable energy system. The dashboard on Ubidots is updated every 5 seconds to reflect the most recent data received from the sensors. This real-time updating feature ensures that users have access to the latest information regarding amperes, voltage and power generated. By monitoring these key metrics in real time, users can track the efficiency of their solar and battery system, identify any fluctuations or anomalies, and make informed decisions to optimise energy usage.

4.5 HARDWARE RESULT

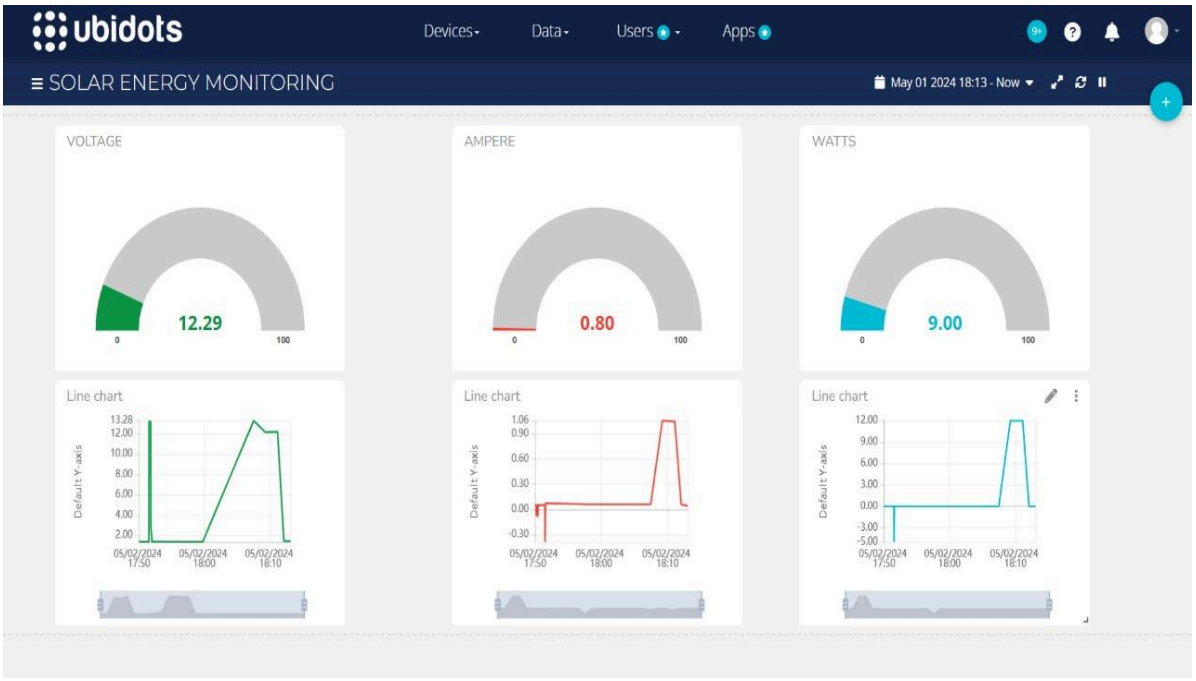

Fig 4.5 Ubidot dashboard result.

The hardware setup for energy monitoring of supply voltage, amperage, and power from solar panels and a battery system, utilising sensors, a microcontroller, and a Wi-Fi module for data transmission to the Ubidots cloud service, delivers real-time monitoring capabilities and insightful data analysis. The sensors capture precise measurements of supply voltage and amperage from the solar panels and battery, providing accurate data on power generation and consumption. This data is processed by the microcontroller and seamlessly transmitted to the Ubidots cloud service via the Wi-Fi module every 5 seconds. Users can conveniently access the Ubidots dashboard remotely to monitor energy production and consumption trends, enabling informed decision-making for optimising system efficiency. The real-time data monitoring, remote accessibility, and detailed data analysis empower users to identify patterns, troubleshoot issues, and make informed adjustments to maximise the performance of their renewable energy system. This integrated hardware solution offers a comprehensive and user-friendly approach to energy monitoring, facilitating efficient management and optimization of solar and battery systems for sustainable energy usage.

With the result provided above (refer to Fig 4.5), the sample data for voltage, amperage, and wattage generation on May 1, 2024, is displayed.

CHAPTER 5
CONCLUSION & FUTURE SCOPE

5.1 CONCLUSION

The project successfully achieved its objectives outlined in this section. It effectively explored the effectiveness of genetic algorithms in optimising energy management for smart homes using simulation-based approaches, focusing on load demand response strategies. Additionally, the implementation of hardware-based monitoring systems enabled the collection and analysis of real-time energy data, facilitating proactive energy management practices.

Also identified the several challenges, such as implementation of Load Demand response is a much more complex process when approaching hardware models.

Based on the findings, it has been concluded that implementing energy management systems with load demand response in smart homes is a promising solution for efficient energy consumption. It aligns with the growing trend of smart technology adoption and sustainability goals.

The comparison between the simulation results of Energy optimization using GA and Energy management with load demand response considering the cost factor was also able to be concluded in this project

The hardware setup described provides a robust and efficient solution for real-time energy monitoring of solar panels and battery systems. By utilising sensors, a microcontroller, and a Wi-Fi module to transmit data to the Ubidots cloud service, users can access detailed insights remotely and make informed decisions to optimise system efficiency.

5.2 FUTURE SCOPE

Looking ahead, the project's future direction involves further refining the integration between simulation and hardware components to create a more robust and intelligent energy management system. This includes exploring advanced algorithms, leveraging machine learning for predictive maintenance, and implementing smart grid technologies for enhanced grid integration and demand response.

REFERENCES

[1] Nwulu, N. I. ,Xiaohua Xia (2016). "Optimal dispatch for a microgrid incorporating renewables and demand response. " Renewable Energy, 99, 1123-1134. http://dx.doi.org/10.1016/j.renene.2016.08.026

[2] Elkholy, M.H.; Senjyu, T.; Lotfy, M.E.; Elgarhy, A.; Ali, N.S.; Gaafar, T.S. " Design and Implementation of a Real-Time Smart Home Management System Considering Energy Saving" Sustainability (2022), 14, 13840. http://doi.org/10.3390/su142113840

[3] Munoz, O.; Ruelas, A.; Rosales, P.; Acuña, A.; Suastegui, A.; Lara, F. " Design and Development of an IoT Smart Meter with Load Control for Home Energy Management Systems." Sensors 2022, 22, 7536. https://doi.org/10.3390/s22197536

[4] Ramin Torkan, Adrian Ilinca, Milad Ghorbanzadeh, "A genetic algorithm optimization approach for smart energy management of microgrids", Renewable Energy, Volume 197, 2022, Pages 852-863, ISSN 0960-1481, https://doi.org/10.1016/j.renene.2022.07.055.

[5] Arabali, A., Ghofrani, M., Etezadi-Amoli, M., Fadali, M. S., & Baghzouz, Y. (2013). "Genetic-Algorithm-Based Optimization Approach for Energy Management". IEEE Transactions on Power Delivery, DOI: 10.1109/TPWRD.2012.2219598

[6] R. Govindarajan, S. Meikandasivam and D. Vijayakumar, 2019." Low Cost Arduino Based Smart Energy Monitoring System Using Internet of Things". Journal of Engineering and Applied Sciences, 14: 170-177. DOI: 10.36478/jeasci.2019.170.177

[7] Megha Sharma, Namita Mittal, Anukram Mishra, Arun Gupta, "Survey of Electricity Demand Forecasting And Demand Side Management Techniques in Different Sectors to Identify Scope for Improvement"Smart Grids and Sustainable Energy (2023) 8:9 https://doi.org/10.1007/s40866-023-00168-z

[8] X. Liu, P. Wang and P. C. Loh, "A Hybrid AC/DC Microgrid and Its Coordination Control," in IEEE Transactions on Smart Grid, vol. 2, no. 2, pp. 278-286, June 2011, doi: 10.1109/TSG.2011.2116162.

[9] Jonathan LeSage (2023).Microgrid Energy Management System Development Using Optimization based Methods . Accessed: Nov 21,2023
Available:https://www.mathworks.com/matlabcentral/fileexchange/73139-microgrid-energy-management-system-ems-using-optimization

[10] Yu Zhang, Yan-Wu Wang, Xiao-Kang Liu, Wu Yang, Shu-Ming Liang, "Distributed Predefined-Time Control for Hybrid AC/DC Microgrid", IEEE Transactions on Industrial Electronics, vol.70, no.8, pp.8324-8333, 2023.

[11] C. L. Nge, O. -M. Midtgård and L. Norum, "PV with battery in smart grid paradigm: Price-based energy management system," 2012 38th IEEE Photovoltaic Specialists Conference, Austin, TX, USA, 2012, pp. 000575-000579, doi: 10.1109/PVSC.2012.6317679.

APPENDIX

PROGRAM 1: ENERGY MANAGEMENT WITH LOAD DEMAND RESPONSE

Filename: EMSWLD.m

```
% Program

% Define data arrays

time = 1:24; % Time in hours

% Define updated data arrays for more realistic home usage

load1 = [4, 3, 1, 1, 1, 0, 3, 4, 5, 6, 7, 5, 5, 4, 2, 0, 3, 2, 5, 4, 2, 2, 1, 2]; % Load profile for load 1

load2 = [1, 2, 1, 2, 3, 4, 5, 3, 3, 0, 2, 2, 5, 6, 7, 8, 5, 2, 1, 1, 1, 1, 2, 5]; % Load profile for load 2

load3 = [1, 3, 3, 5, 6, 4, 2, 2, 2, 0, 0, 0, 0, 3, 4, 4, 5, 8, 9, 7, 4, 2, 2, 0]; % Load profile for load 3

solar_power = [0, 0, 0, 0, 0, 0, 0, 7, 7, 8, 8, 6, 7, 6, 7, 7, 5, 4, 3, 1, 1, 2, 2, 3]; % Solar power
generation profile

battery_capacity = 20; % Battery capacity in kWh

battery_charge_efficiency = 0.9; % Charging efficiency of the battery

battery_discharge_efficiency = 0.8; % Discharging efficiency of the battery

% Define load demand response parameters

demand_response_factor_peak = 0.7; % Load demand response factor during peak hours (70% of
original load)

demand_response_factor_non_peak = 0.8; % Load demand response factor during non-peak hours
(80% of original load)

% Initialize variables

total_energy_consumed = zeros(1, 24);

total_energy_generated = zeros(1, 24);

total_energy_deficit = zeros(1, 24);

total_energy_from_solar = zeros(1, 24);

total_energy_from_grid = zeros(1, 24);

total_energy_from_battery = zeros(1, 24);

battery_storage = 0;

% Energy management optimization

for i = 1:length(time)

    if ismember(i, [6, 7, 8, 9, 18, 19, 20])
```

```
        % Peak hours
        total_energy_consumed(i) = load1(i) * demand_response_factor_peak + load2(i) * demand_response_factor_peak + load3(i) * demand_response_factor_peak;
        % Peak hour energy management logic
        if solar_power(i) >= total_energy_consumed(i)
            total_energy_from_solar(i) = total_energy_consumed(i);
        elseif solar_power(i) + battery_storage >= total_energy_consumed(i)
            total_energy_from_solar(i) = solar_power(i);
            battery_storage = max(0, battery_storage + solar_power(i) - total_energy_consumed(i));
        else
            total_energy_from_solar(i) = solar_power(i) + min(total_energy_consumed(i) - solar_power(i), battery_storage / battery_discharge_efficiency);
            battery_storage = max(0, battery_storage - (total_energy_consumed(i) - solar_power(i)) * battery_charge_efficiency);
        end
        total_energy_generated(i) = total_energy_from_solar(i);
    else
        % Non-peak hours
total_energy_consumed(i) = load1(i) * demand_response_factor_non_peak + load2(i) * demand_response_factor_non_peak + load3(i) * demand_response_factor_non_peak;
% Non-peak hour energy management logic
                if solar_power(i) >= total_energy_consumed(i)
                        total_energy_from_solar(i) = total_energy_consumed(i);
                else
                        total_energy_from_solar(i) = solar_power(i);
                        total_energy_from_grid(i) = max(0, total_energy_consumed(i) - solar_power(i));
                end
                        total_energy_generated(i) = total_energy_from_solar(i);

                if total_energy_generated(i) < total_energy_consumed(i)
                        total_energy_deficit(i) = total_energy_consumed(i) - total_energy_generated(i);
                end
```

```
    end
end
        % Display results for each hour
    fprintf('Total Energy Consumed: %.2f kWh\n', total_energy_consumed(i));
    fprintf('Total Energy Generated: %.2f kWh\n', total_energy_generated(i));
    fprintf('Total Energy Deficit: %.2f kWh\n', total_energy_deficit(i));
    fprintf('Energy from Solar: %.2f kWh\n', total_energy_from_solar(i));
    fprintf('Energy from Grid: %.2f kWh\n\n', total_energy_from_grid(i));
% Plot the energy profiles as curves
figure;
plot(time, total_energy_consumed, 'b', 'LineWidth', 1.5);
hold on;
plot(time, total_energy_generated, 'g', 'LineWidth', 1.5);
plot(time, total_energy_deficit, 'r', 'LineWidth', 1.5);
plot(time, total_energy_from_solar, 'm', 'LineWidth', 1.5);
plot(time, total_energy_from_grid, 'c', 'LineWidth', 1.5);
xlabel('Time (hours)');
ylabel('Energy (kWh)');
legend('Total Energy Consumed', 'Total Energy Generated', 'Total Energy Deficit', 'Energy from Solar',
'Energy from Grid');
title('Energy Management Results');
grid on;
% Create a bar chart to visualize the results
figure;
bar(time, [total_energy_consumed; total_energy_generated; total_energy_deficit;
total_energy_from_solar; total_energy_from_grid]', 'stacked');
xlabel('Time (hours)');
ylabel('Energy (kWh)');
legend('Total Energy Consumed', 'Total Energy Generated', 'Total Energy Deficit', 'Energy from Solar',
'Energy from Grid');
title('Energy Management Results');


% Define cost parameters
```

```
grid_price_peak = 11.25; % Grid electricity price during peak hours (Rs/kWh)
grid_price_non_peak = 7.50; % Grid electricity price during non-peak hours (Rs/kWh)
solar_price = 2.50; % Solar electricity price (Rs/kWh)
battery_cost = 1.50; % Battery charging cost (Rs/kWh)

% Initialize cost variables
total_cost_grid = zeros(1, 24);
total_cost_solar = zeros(1, 24);
total_cost_battery = zeros(1, 24);

% Initialize variables
total_energy_consumed = zeros(1, 24);
total_energy_generated = zeros(1, 24);
total_energy_deficit = zeros(1, 24);
total_energy_from_solar = zeros(1, 24);
total_energy_from_grid = zeros(1, 24);
total_energy_from_battery = zeros(1, 24);
battery_storage = 0;

% Energy management optimization without cost-efficient savings
total_cost_no_optimization = zeros(1, 24);
for i = 1:length(time)
    total_energy_consumed(i) = load1(i) + load2(i) + load3(i);
    total_cost_no_optimization(i) = total_energy_consumed(i) * grid_price_peak;
end
% Energy management optimization with cost-efficient savings
total_cost_optimization = zeros(1, 24);
for i = 1:length(time)
    if ismember(i, [7, 8, 9, 18, 19, 20])
        % Peak hours
        total_energy_consumed(i) = load1(i) * demand_response_factor_peak + load2(i) *
demand_response_factor_peak + load3(i) * demand_response_factor_peak;
```

```
        % Peak hour energy management logic for cost savings
        if solar_power(i) >= total_energy_consumed(i)
            total_energy_from_solar(i) = total_energy_consumed(i);
        else
            total_energy_from_solar(i) = min(solar_power(i), total_energy_consumed(i));
            total_energy_from_grid(i) = total_energy_consumed(i) - total_energy_from_solar(i);
        end
        % Battery usage for cost-efficient savings during peak hours
        battery_used = min(total_energy_consumed(i) - total_energy_from_solar(i), battery_storage);
        total_energy_from_battery(i) = battery_used * battery_discharge_efficiency;
        battery_storage = max(0, battery_storage - battery_used);
        % Calculate costs for optimization
        total_cost_solar = total_energy_from_solar(i) * solar_price;
        total_cost_grid = total_energy_from_grid(i) * grid_price_peak;
        total_cost_battery = battery_used * battery_cost;
        total_cost_optimization(i) = total_cost_solar + total_cost_grid + total_cost_battery;
    else
        % Non-peak hours
        total_energy_consumed(i) = load1(i) * demand_response_factor_non_peak + load2(i) *
demand_response_factor_non_peak + load3(i) * demand_response_factor_non_peak;
        % Non-peak hour energy management logic for cost savings
        if solar_power(i) >= total_energy_consumed(i)
            total_energy_from_solar(i) = total_energy_consumed(i);
        else
            total_energy_from_solar(i) = solar_power(i);
            total_energy_from_grid(i) = total_energy_consumed(i) - solar_power(i);
        end
        % Calculate costs for optimization
        total_cost_solar = total_energy_from_solar(i) * solar_price;
        total_cost_grid = total_energy_from_grid(i) * grid_price_non_peak;
        total_cost_optimization(i) = total_cost_solar + total_cost_grid;
    end
end
```

```
% Calculate total costs for before and after optimization
total_costs_no_optimization = sum(total_cost_no_optimization);
total_costs_optimization = sum(total_cost_optimization);
% Display total costs for before and after optimization
fprintf('Total Cost without Energy Management: Rs%.2f\n', total_costs_no_optimization);
fprintf('Total Cost after Energy Management: Rs%.2f\n', total_costs_optimization);
% Plot the cost comparison
figure;
plot(time, total_cost_no_optimization, 'b--', 'LineWidth', 1.5);
hold on;
plot(time, total_cost_optimization, 'r', 'LineWidth', 1.5);
xlabel('Time (hours)');
ylabel('Cost (Rs)');
legend('Cost without Energy Management', 'Cost after Energy Management');
title('Cost Comparison: Before and After Energy Management');
grid on;
% Plot the different load profiles
figure;
bar(time, [load1; load2; load3]', 'stacked');
xlabel('Time (hours)'); ylabel('Load (kW)');
legend('Load 1', 'Load 2', 'Load 3');
title('Different Load Profiles');
 figure;
 plot(time, load1, 'b', 'LineWidth', 1.5);
 hold on; plot(time, load2, 'g', 'LineWidth', 1.5);
 plot(time, load3, 'r', 'LineWidth', 1.5); xlabel('Time (hours)');
 ylabel('Load (kW)'); legend('Load 1', 'Load 2', 'Load 3');
 title('Different Load Profiles');
 grid on;
```

PROGRAM 2: ENERGY OPTIMIZATION USING GA

Filename: EOWGA.m

```
%Program

% Define cost parameters

grid_price_peak = 11.25; % Grid electricity price during peak hours (Rs/kWh)

grid_price_non_peak = 7.50; % Grid electricity price during non-peak hours (Rs/kWh)

solar_price = 2.50; % Solar electricity price (Rs/kWh)

battery_cost = 1.50; % Battery charging cost (Rs/kWh)

% Define the maximum power values for solar, grid, and battery

max_solar_power = 100; % Maximum solar power available (kW)

max_grid_power = 200; % Maximum grid power available (kW)

max_battery_power = 50; % Maximum battery power available (kW)

% Define the demand response factors for peak and non-peak hours

demand_response_factor_peak = 0.8;

demand_response_factor_non_peak = 0.6;

% Initialize variables

time = 1:24;

load1 = [4, 3, 1, 1, 1, 0, 3, 4, 5, 6, 7, 5, 5, 4, 2, 0, 3, 2, 5, 4, 2, 2, 1, 2]; % Load profile for load 1

load2 = [1, 2, 1, 2, 3, 4, 5, 3, 3, 0, 2, 2, 5, 6, 7, 8, 5, 2, 1, 1, 1, 1, 2, 5]; % Load profile for load 2

load3 = [1, 3, 3, 5, 6, 4, 2, 2, 2, 0, 0, 0, 0, 3, 4, 4, 5, 8, 9, 7, 4, 2, 2, 0]; % Load profile for load 3

solar_power = [0, 0, 0, 0, 0, 0, 0, 7, 7, 8, 8, 6, 7, 6, 7, 7, 5, 4, 3, 1, 1, 2, 2, 3]; % Solar power
generation profile

battery_discharge_efficiency = 0.9; % Efficiency of battery discharge

% Define the GA parameters

population_size = 50;

generations = 100;

mutation_rate = 0.02;

% Define the fitness function

fitness_function = @(x) energy_cost_function(x, solar_power, load1, load2, load3, grid_price_peak,
grid_price_non_peak, solar_price, battery_cost);

% Initialize the population with random solutions

population = rand(population_size, 3) .* [max_solar_power, max_grid_power, max_battery_power];
```

```
% Main GA loop
for gen = 1:generations
    % Evaluate fitness of each individual in the population
    fitness = zeros(population_size, 1);
    for i = 1:population_size
        fitness(i) = fitness_function(population(i, :));
    end
    % Selection
    [~, idx] = sort(fitness);
    population = population(idx, :);
    % Crossover
    new_population = zeros(population_size, 3);
    for i = 1:2:population_size
        parent1 = population(mod(i, population_size) + 1, :);
        parent2 = population(mod(i + 1, population_size) + 1, :);
        crossover_point = randi([1, 2]);
        new_population(i, :) = [parent1(1:crossover_point), parent2(crossover_point + 1:end)];
        new_population(i + 1, :) = [parent2(1:crossover_point), parent1(crossover_point + 1:end)];
    end
    % Mutation
    for i = 1:population_size
        if rand < mutation_rate
            new_population(i, :) = new_population(i, :) + randn(1, 3) .* [max_solar_power,
max_grid_power, max_battery_power] .* 0.1;
            new_population(i, :) = max(new_population(i, :), 0); % Ensure values do not go negative
        end
    end
    population = new_population;
end
% Find the best solution
best_solution = population(1, :);
best_cost = fitness_function(best_solution);
```

```
disp('Optimized Solution:');

disp(best_solution);

disp(['Optimized Cost: Rs', num2str(best_cost)]);

% Define the energy_cost_function

function total_cost = energy_cost_function(x, solar_power, load1, load2, load3, grid_price_peak, grid_price_non_peak, solar_price, battery_cost)

    % Extract decision variables

    solar_usage = x(1);

    grid_usage = x(2);

    battery_usage = x(3);

    % Calculate total energy consumed and cost

    total_energy_consumed = solar_usage + grid_usage + battery_usage;

    total_cost = max(0, solar_usage * solar_price + grid_usage * grid_price_peak + battery_usage * battery_cost); % Ensure cost is non-negative

    % Add constraints if needed

    % Return the total cost

end

% Define cost parameters

grid_price_peak = 11.25; % Grid electricity price during peak hours (Rs/kWh)

grid_price_non_peak = 7.50; % Grid electricity price during non-peak hours (Rs/kWh)

solar_price = 2.50; % Solar electricity price (Rs/kWh)

battery_cost = 1.50; % Battery charging cost (Rs/kWh)

% Define the maximum power values for solar, grid, and battery

max_solar_power = 100; % Maximum solar power available (kW)

max_grid_power = 200; % Maximum grid power available (kW)

max_battery_power = 50; % Maximum battery power available (kW)

% Define the demand response factors for peak and non-peak hours

demand_response_factor_peak = 0.8;

demand_response_factor_non_peak = 0.6;

% Initialize variables

time = 1:24;

load1 = [4, 3, 1, 1, 1, 0, 3, 4, 5, 6, 7, 5, 5, 4, 2, 0, 3, 2, 5, 4, 2, 2, 1, 2]; % Load profile for load 1

load2 = [1, 2, 1, 2, 3, 4, 5, 3, 3, 0, 2, 2, 5, 6, 7, 8, 5, 2, 1, 1, 1, 1, 2, 5]; % Load profile for load 2
```

```
load3 = [1, 3, 3, 5, 6, 4, 2, 2, 2, 0, 0, 0, 0, 3, 4, 4, 5, 8, 9, 7, 4, 2, 2, 0]; % Load profile for load 3

solar_power = [0, 0, 0, 0, 0, 0, 0, 7, 7, 8, 8, 6, 7, 6, 7, 7, 5, 4, 3, 1, 1, 2, 2, 3]; % Solar power generation profile

battery_discharge_efficiency = 0.9; % Efficiency of battery discharge

% Calculate cost without optimization

total_cost_no_opt = sum(load1 * grid_price_peak + load2 * grid_price_peak + load3 * grid_price_peak); % Assuming all loads are powered by the grid

% Initialize battery SOC

initial_battery_SOC = 20; % Initial Battery State of Charge (kWh)

battery_SOC = initial_battery_SOC;

% Plot load and power sources

figure;

subplot(2,1,1);

plot(time, load1, 'r', 'LineWidth', 1.5);

hold on;

plot(time, load2, 'b', 'LineWidth', 1.5);

plot(time, load3, 'g', 'LineWidth', 1.5);

xlabel('Time (hours)');

ylabel('Power Consumption (kW)');

title('Load Consumption Profiles');

legend('Load 1', 'Load 2', 'Load 3');

grid on;

hold off;

subplot(2,1,2);

plot(time, solar_power, 'y', 'LineWidth', 1.5);

hold on;

plot(time, ones(size(time))*max_grid_power, 'b--', 'LineWidth', 1.5);

plot(time, ones(size(time))*max_battery_power, 'g--', 'LineWidth', 1.5);

xlabel('Time (hours)');

ylabel('Power Generation/Usage (kW)');

title('Power Source Generation/Usage Profiles');

legend('Solar Power', 'Max Grid Power', 'Max Battery Power');

grid on;
```

```
hold off;
disp('Cost without optimization:');
disp(['Total Cost: Rs', num2str(total_cost_no_opt)]);

matlab
% Define cost parameters, variables, and functions as before...
% Initialize arrays to store power source consumption during optimization
solar_consumption = zeros(1, 24);
grid_consumption = zeros(1, 24);
battery_consumption = zeros(1, 24);
% Main GA loop
for gen = 1:generations
    % Update power source consumption arrays
    best_solution = population(1, :);
    solar_consumption = solar_consumption + best_solution(1) * solar_power;
    grid_consumption = grid_consumption + best_solution(2) * ones(1, 24);
    battery_consumption = battery_consumption + best_solution(3) * ones(1, 24);
end
% Plot power source consumption during optimization
figure;
plot(time, solar_consumption, 'y', 'LineWidth', 1.5);
hold on;
plot(time, grid_consumption, 'b', 'LineWidth', 1.5);
plot(time, battery_consumption, 'g', 'LineWidth', 1.5);
xlabel('Time (hours)');
ylabel('Power Consumption (kW)');
title('Power Source Consumption Profiles during Optimization');
legend('Solar Power Consumption', 'Grid Power Consumption', 'Battery Power Consumption');
grid on;
hold off;
```

PROGRAM 3: ENERGY REAL-TIME MONITORING SYSTEM (ARDUINO IDE)

Filename: ERTMS.ino

```
#include <ESP8266WiFi.h>
#include "Ubidots.h"

const char* ssid = "smartsolar";
const char* password = "123456789";
const char* ubidotsToken = "BBUS-aisgiRnNwSTPG8AclDb2aeSuzhWwVe";
const char* watts_id = "watts";
const char* voltage_id = "voltage";
const char* current_id = "current";

const int analogPin = A0;
const float Vref_actual = 3.38;
Ubidots ubidots(ubidotsToken);

unsigned long previousMillis = 0;
const long interval = 3000;  // Interval to send data to Ubidots (milliseconds)

String currentString = "";  // String to store received serial data
float voltage = 0.0; // Global variable to store voltage data
float current = 0.0; // Global variable to store current data
int watts = 0; // Global variable to store watts data

void setup() {
  Serial.begin(115200);
  delay(1000); // Allow time for serial monitor to initialize
  Serial.println("Connecting to WiFi...");
  ubidots.wifiConnect(ssid, password);
}
```

```
void loop() {
  // Read serial data
  while (Serial.available()) {
    char c = Serial.read();
    if (c == '\n') {
      // End of line, process data
      current = currentString.toFloat(); // Convert string to float
      int sensorValue = analogRead(analogPin);
      voltage = sensorValue * Vref_actual / 214.005;

      // Calculate watts
      watts = int(voltage * current); // Convert to integer

      Serial.print("Voltage: ");
      Serial.print(voltage, 2);
      Serial.println(" V");

      Serial.print("Current: ");
      Serial.print(current, 2);
      Serial.println(" A");

      Serial.print("Watts: ");
      Serial.print(watts);
      Serial.println(" W");

      // Reset string for next reading
      currentString = "";
    } else {
      // Append character to string
      currentString += c;
    }
```

```
  }

  // Send data to Ubidots at specified interval
  unsigned long currentMillis = millis();
  if (currentMillis - previousMillis >= interval) {
    previousMillis = currentMillis;

    // Send data to Ubidots
    ubidots.add(watts_id, watts);
    ubidots.add(voltage_id, voltage);
    ubidots.add(current_id, current);
    ubidots.send();
  }
}
```