IoT based Smart Home Energy Management System

Thesis to be submitted

in partial fulfillment of the requirements for the award of the

Master of Technology

in

Power Electronics and Drives

(under the Dual-Degree Programme)

by

Gaurav Barmola

17EE02001

Under the supervision of

Dr. Chandrasekhar Perumalla

SCHOOL OF ELECTRICAL SCIENCES

INDIAN INSTITUTE OF TECHNOLOGY BHUBANESWAR

8 July, 2022

APPROVAL OF THE VIVA-VOCE BOARD

8 July, 2022

Certified that the report entitled **'IoT based Smart Home Energy Management System'** submitted by **Gaurav Barmola (17EE02001)** to the Indian Institute of Technology Bhubaneswar in partial fulfilment of the requirements for the award of the Master of Technology in Power Electronics and Drives under the Dual-Degree Programme has been accepted by the examiners during the viva-voce examination held today.

(Supervisor) **(External Examiner)**

(Internal Examiner 1) **(Internal Examiner 2)**

SCHOOL OF ELECTRICAL SCIENCES

INDIAN INSTITUTE OF TECHNOLOGY BHUBANESWAR

CERTIFICATE

This is to certify that the report entitled "**IoT based Smart Home Energy Management System**" submitted by **Gaurav Barmola** (Roll No. 17EE02001) to Indian Institute of Technology Bhubaneswar is a record of bonafide research work under my supervision and the report is submitted for end-semester evaluation of the Dual Degree thesis work.

Date: 8 July, 2022
Place: Bhubaneswar

Dr. Chandrasekhar Perumalla
School of Electrical Sciences
Indian Institute of Technology
Bhubaneswar

DECLARATION

I certify that

(a) The work contained in the thesis is original and has been done by myself under the general supervision of my supervisor.

(b) The work has not been submitted to any other Institute for any degree or diploma.

(c) I have followed the guidelines provided by the Institute in writing the report.

(d) I have conformed to the norms and guidelines given in the Ethical Code of Conduct of the Institute.

(e) Whenever I have used materials (data, theoretical analysis, figures, and text) from other sources, I have given due credit to them by citing them in the text of the thesis and giving their details in the references.

(f) Whenever I have quoted written materials from other sources, I have put them under quotation marks and given due credits to the sources by citing them and giving required details in the references.

Date: 8 July, 2022 (Gaurav Barmola)

Place: Bhubaneswar (17EE02001)

Abstract

Name of the student: **Gaurav Barmola** Roll No: **17EE02001**

Degree for which submitted: **Master of Technology**

Department: **School of Electrical Sciences**

Thesis title: **IoT based Smart Home Energy Management System**

Thesis supervisor: **Dr. Chandrasekhar Perumalla**

Month and year of thesis submission: **8 July, 2022**

With the increasing adoptions of smart home devices and development of smart grids, Smart Home Energy Management Systems , or SHEMS, have a massive role to reduce energy consumption and maximise the use of renewable energy. In this work, a practical SHEMS model is established, containing renewable energy generation, energy storage capablities, all types of electrical home appliances along with electric vehicle. Then, a modified genetic algorithm (GA) based load scheduling is proposed with the goal to maximise the utilisaton of renewable energy and minimise the electricity cost. Simulations are performed to demonstrate that the proposed scheme has a significant effect on reduction of costs and energy saving. Results confirm that the proposed method has high robustness and avoids disadvantage of traditional heuristic based optimisations. Finally, prototypes of the proposed smart plugs and EMS are implemented in hardware using IoT devices. The setup allows remote monitoring and control to the user through internet dashboard, mobile application or physical smart switches; and perform real time energy optimisation.

Contents

List of Figures

Chapter 1

Introduction

In recent times the residential energy consumtion is increasing rapidly due to improved quality of living, and rapid urbanisation and industrialization. At the same time, there is a massive global push to reduce the energy dependency on conventional fuels. An increasing number of residential buildings have renewable energy production capablities, mostly through solar. Integrating these distributed energy production with grid while optimising the energy saving is a very important problem. Their are many technology dependent solutions to ensure most efficient utilisation of energy in domestic settings. Smart home adoption have seen massie uptick in the last decade owing to reduction in cost of IoT smart devices, increasing investment in the field from governments and corporations alike and increasing connectivity.

In the above context of prioritizing energy savings, achievement intelligent home appliances if necessary. Demand side management (DSM) along with optimising the cost from users perspective, should also helps in shifting the loads away from the peak without affecting the consumption and hence the user's comfort, to ensure stability of the power system. Smart energy management have attracted a lot of attention as it promises to solve all these problems. Home Energy Management

System (HEMS) make use of actuators and sensors to monitor the energy usage and make smart globally optimal load scheduling decisions while taking into account all the parameters and tradeoff between user comfort and cost savings.

The HEMS uses optimal scheduling, along with control methods to achieve maximum power savings along with comfort for the user. It is, therefore, very important to seek practical and efficient energy management algorithms to overcome the problem of optimal appliance scheduling in smart homes. It is also necessary to come up with the hardware architecture of the HEMS. A reliable and wireless communication must be used for establishing home are network. The home network should be connected to the internet to allow remote monitoring and big data analysis by the utilities. In this work we try to address all these problems and propose a optimal energy scheduling algorithm and implement an reliable and robust IoT system.

Chapter 2

Literature Survey

There are various works in the field of SHEMS that have proposed different methods for optimal energy sheduling in smart home models. A considerable number of works focuses on the deployed algorithms for effective demand-side energy management.

The authors of [1] implemented a HEMS that is developed based on heuristic-based optimization algorithms. The work employs Binary Particle Swarm Optimization (BPSO) and Wind Driven Optimization (WDO) . Although here the authors did not categorize or prioritize the residential appliances and also did not specify the duration of operation for the different appliances.

the work [2] uses a genetic algorithm to reduce costs in a time-of-use (TOU)-based HEMS process. To divert load away from peak hours, DR is also taken into consideration in the work. However, the model does not account for battery storage or renewable energy. Additionally, classical GA are sensitive to initial conditions and is simple to trap in a local optimal.

[3] proposes a novel method for optimal energy consumption utilising a GA and the Bat Algorithm (BA), a type of particle swarm optimization, and refers to the

created algorithm as the bat genetic algorithm (BGA). Time of Use (TOU) pricing is used to implement it. Peak to average ratio and electricity costs are minimised, but renewable energy or battery storage are not taken into account.

[4] uses reinforcement learning to suggest a HEMS for residential settings. It employs a reinforcement learning (RL) technique called Q-learning, which is based on a reward mechanism, to develop an optimal schedule for domestic appliances that minimizes electricity costs while preserving user comfort and preferences.

In [5], authors developed a HEMS model that takes the user's preferences into account and keeps power usage within the predetermined limit to ensure the user's comfort. The model did not, however, propose an operating strategy taking into account variation in price of electricity in real-time to minimize the cost, rather, it only employed a signal for Demand response to influence the user decisions.

[6] uses a scheduling method that combines Artificial Neural Network (ANN) with GA to find the ideal pairings that give the best weekly schedules. Work aims to maximise the utilisation of renewable sources while reducing reliance on grid energy. The authors do not take battery storage into consideration.

In [7], evaluation of the effectiveness of EMS by consideration of energy consumption data, , room and ambient temperature along with user's usage profile before and after the using the EMS system.

In the literature, design of several energy management frameworks is based on several communication technologies, including Wi-Fi, ZigBee, and powerline carriers. [8] presented a home EMS with real-time data monitoring and power line communication-based intelligent control features. With the help of the statistical Reduce model, which analyses the energy-saving options by taking a wide range of factors into account, the proposed system offers remote monitoring functionality for greater energy conservation.

In [9] authors create a supervised machine learning classification challenge out of our scheduling task, activating the load on one of two desired time slots. Discuss implementation concerns further utilising a microcontroller system in conjunction with serverless computing in the cloud and dynamic data storage.

An IoT-based system is designed and them deployed in [10], which improves the interactivity of the energy management systems in the homes. The system which is proposed expands on approaches already existing by incorporating large amount of data such as residential data, electricity prices, ambient temperature and humidity data, and user preferences.

The paper [11] devlops an IoT-based HEMS with a solar generation system connected to a home area networks, making HEMS construction more accessible, versatile, and scalable. It developed and implemented the proposed system, and carried out an experiment to validate the proposed system's performance.

A controlled partial load shedding procedure, intended to take the place of a complete power outage in a given area in a smart energy management system is developed in [12]. In addition to developing an IoT environment for data analytics and storage, ZigBee communication for home area networks is also constructed.

2.1 Scope and Objective

The development of an algorithm that can generate optimal scheduling of home appliances is a very important problem. In conclusion, the following gaps exist and should be properly addressed:

- Despite the fact that the models mentioned above can reduce electricity cost and increase user satisfaction, the use of renewable energy, battery sustainability, and variable electricity costs are still not properly considered in the previous works, which can lead to energy waste and additional costs.

- The above heuristic based algorithms are prone to be trapped in local optima. Additionally, they tend to be sensitive to the initial conditions and initial solutions.

Additionally, the hardware architecture for the HEMS is proposed and implemented. The communication is done over the internet using wifi, as it provides the best combination of scalability, reliability, and versatility.

Chapter 3

SHEMS Design and Modelling

In this chapter, we are first going to take a brief look at the design and working of the proposed SHEMS. Afterward the entire smart home is mathematically modeled with all appliance, systems and constraint. These models are necessary order to develop and test effective optimisation algorithm.

3.1 Overview of the HEMS

The smart home is a technological platform that includes both hardware and software. Hardware serves primarily as a communications foundation. A gateway to other WAN or smart grid networks is provided by the home-area network (HAN), which connects digital devices into a common network via wireless or wired technology.

Energy management system (HEMS) makes all of the decisions in a smart home. In practise, the terms 'smart house' and 'home energy management system' are used interchangeably. To avoid this misunderstanding, it is critical to thoroughly define both concepts. Essentially, smart homes deal with infrastructure, whereas HEMS

deals with decision support. Not only does a smart home imply more infrastructure, base platform, and hardware concepts, but also a home energy management system that works as a decision support system on smart house infrastructure. Thus, by changing energy consumption behavior, the home user can make better decisions about reducing energy consumption and managing energy resources.

The proposed HEMS is depicted in the diagram below. Each appliance in the house is interfaced with a smart plug, which connects to the power socket. These individual smart plugs will connect to the HEMS's central node, which will handle energy monitoring. When a user turns on an appliance, the smart plug sends a request to the central node. Each appliance is connected to a smart plug, and the central node communicates with each plug wirelessly through the internet using the wifi network to obtain fully automatic control of home appliances.

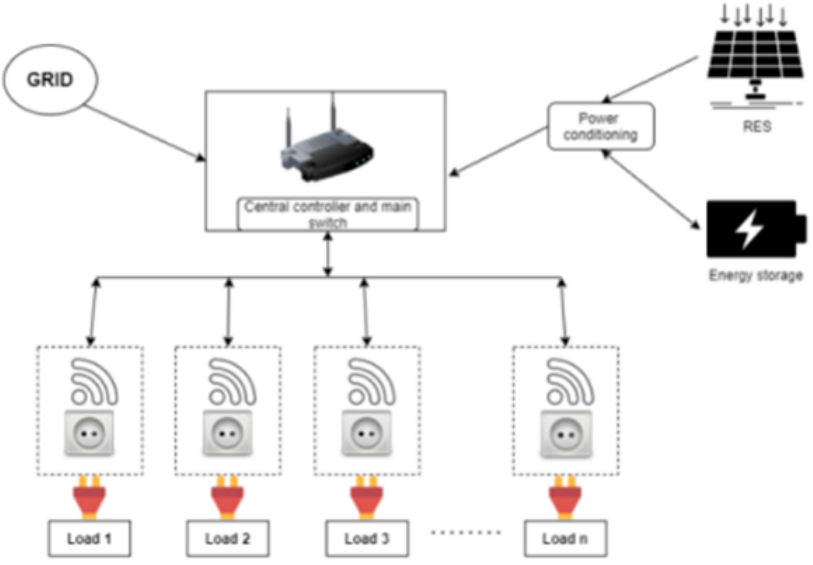

FIGURE 3.1: Structure of proposed HEMS

The load scheduling algorithm will then be executed by the central node. The algorithm can take into account the appliance's priority, total energy consumption,

and RES power generation, users preference etc. The appliance can be turned on or scheduled for a later time based on the decision of such an algorithm. When a central node turns on an appliance, it can run until it is turned off, or it can be pre-empted by the central node depending on the situation.

3.2 System Model

The model includes household appliances, generation of solar power , electric vehicle (EV), and battery storage system. The excess energy can be sold to the grid when combined with the Real Time Pricing and demand response information provided by the grid.

3.2.1 Model of appliances

The load can be classified into rigid and flexible loads depending on whether their run time can be scheduled. For example lighting, television, and computers are rigid loads as they directly impact users comfort. When the rigid load is requested to be turened on, it should be provided right away. If the rigid load does not turn on immediately, the user's comfort will decline. The flexible load is time elastic and responsive to RTP and PV generation, and it has little impact on user comfort.

All the appliances in the home are modelled with their properties and constraints. The entire scheduling space is defines as T. This scheduling space is divided into time slots, where each time slot is referred to as t. The lenght of a time slot used in the work is 30 minutes, so t=1, 2, 3, ... , 47, 48. The set of all appliances is reffered to as A, and individual appliance is reffered to as a, where a $\in A$.

Following are symbols used in the modelling :

$$T = scheduling\ Space$$

$$t = Specific\ Time\ Slot$$

$$A = Set\ of\ all\ Appliances$$

$$a = specific\ Appliance$$

$$P_a = Power\ of\ device\ a$$

$$P_a(t) = Power\ of\ Device\ a\ in\ time\ slot\ t$$

$$d_a = Duration\ of\ time\ set\ by\ the\ user\ for\ running\ the\ device$$

$$[\ \alpha_a\ , \beta_a\ ] = The\ range\ in\ which\ appliance\ is\ allowed\ to\ operate$$

$$S_a = Binary\ variable\ representing\ the\ state\ of\ appliance;$$

$$S_a(t) = 1 \Rightarrow Appliance\ is\ ON\ at\ time\ t$$

$$S_a(t) = 0 \Rightarrow Appliance\ is\ OFF\ at\ time\ t$$

Every appliance must satisfy the following basic time constraints:

$$\sum_{\alpha_a}^{\beta_a} S_a(t) = d_a$$

$$1 \quad \leq \quad \alpha_a \quad \leq \quad T - d_a + 1$$

$$d_a \quad \leq \quad \beta_a \quad \leq \quad T$$

$$S_a(t) = 0 \quad \forall \quad t \in T - [\alpha_a, \beta_a]$$

$$E_a = P_a \cdot d_a$$

The constraints listed above apply to all flexible loads, and in the equation above, Pa and Ea respectively refer to the power rating of an appliance and the total energy consumed in a unit slot.

3.2.2 Load models

Appliances are classified into rigid or non-schedulable and schedulable or flexible appliances. Schedulable appliances are further classified into interruptible and non-interruptible appliances.

The schedulable appliances are further classified into interruptible appliances and non-interruptible appliances. Example of interruptible loads are washing machines, over or television.

can be divided into the interruptible load (such as EVs) and non-interruptible load (such as washing machines and dishwashers).

Following are the models for different types of loads:

- **Interruptible loads**:

$$S_a(t) = \{0, 1\} \quad , \quad t \in [\alpha_a, \beta_a]$$

$$\sum_{t=\alpha_a}^{\beta_a} S_a(t) \cdot P_a = E_a$$

- **Non-interruptible loads:** These device cannot be interrupted once they are started untill the task is completed. Ex. washing machine, oven. These appliance have new set of constraints.

$$\sum_{t=\tau+1}^{\tau+d_a} S_a(t) \quad \geq \quad d_a \cdot [S_a(\tau+1) - S_a(\tau)]$$
$$where \qquad \tau \in [\ \alpha_a - 1\ ,\ \beta_a - d_a]$$

- **Air conditioning system:**

$$T_{in}(t+1) \quad = \quad T_{in} \cdot e^{-\left(\frac{\Delta h}{\xi}\right)} + R_{eq} \cdot P_a(t) \cdot \left[K_{air} \cdot \left(1 - e^{-\left(\frac{\Delta h}{\xi}\right)}\right)\right]$$

Where,

$$T_{in} = Indoor\ temprature$$
$$T_{out} = Outdoor\ temprature$$
$$P_a(t) = Power\ of\ AC\ system$$
$$R_{eq} = Eqivalent\ thermal\ resistence\ of\ home$$
$$K_{air} = Power\ conversion\ coefficient$$
$$\xi = R_{eq} \cdot C \cdot M_{air}$$
$$C = atmospheric\ heat\ capacity\ at\ 1\ atm\ pressure$$
$$= \ 0.525 kWh/Celsius$$

The temperature inside must be always within the allowable range of temperature to ensure users comfort.

$$T_{in}^{min} \quad \leq \quad T_{in}(t) \quad \leq \quad T_{in}^{max}$$

3.2.3 Battery Energy Storage Model

The battery storage system can dynamically switch between charging or discharging behavior based on system energy, allowing for greater optimization and flexibility. The amount of charge in the battery system or the state of charge is represented by the variable C_{bat}. C_{batt} is a floating-point variable with value between 0 and 1, and represent the remaining energy as a ratio of rated capacity of the battery system.

Auxiliary binary variable S_{bat}^{ch} *and* S_{bat}^{dis} is used to represent the state of charging or discharging of battery. $S_{bat}^{ch} = 1$ means the battery is charging, and $S_{bat}^{dis} = 1$ means the battery is discharging. S_{bat}^{ch} *and* S_{bat}^{dis} cannot be both 1 simuntaneously.

$$C\left(t+1\right) \;=\; C\left(t\right) + \frac{\left(P_{batt}^{ch}\left(t\right) - P_{batt}^{dis}\left(t\right)\right) \cdot t}{E_{batt}}$$

Where,

$$P_{batt}^{ch} = Charging\ rated\ power\ of\ battery\ storage$$
$$P_{batt}^{dis} = Disharging\ rated\ power\ of\ battery\ storage$$
$$E_{batt} = Rated\ energy\ capacity\ of\ the\ battery\ system$$

To extend the battery's lifetime, it is important to limit the State of charge to a specific range.

$$C^{min} \;\leq\; C\left(t\right) \;\leq\; C^{max}$$

C^{min} and C^{max} are the lower and the upper limits on state of charge for the battery storage. The charging and discharging limits of the battery storage system after considering the efficiency are following:

$$0 \leq \frac{P^{ch}_{batt}}{\eta_{ch}} \leq P^{ch}_{batt} \cdot S^{ch}_{batt}(t)$$

$$0 \leq P^{dis}_{batt} \cdot \eta_{dis} \leq P^{dis}_{batt} \cdot S^{dis}_{batt}(t)$$

Where,

$$\eta_{ch} = Charging\ efficiency\ of\ the\ battery\ storage$$

$$\eta_{dis} = Discharging\ efficiency\ of\ the\ battery\ storage$$

The output power from the battery storage is given below. Here a positive value means the battery is charging and a negative value represents discharging.

$$P_{batt}(t) = \frac{P^{ch}_{batt}}{\eta_{ch}} \cdot S^{ch}_{batt}(t) - P^{dis}_{batt} \cdot \eta_{dis} \cdot \left(1 - S^{dis}_{batt}(t)\right)$$

3.2.4 Model of EV

EV model is identical to the battery storage but EV can only be charged and cannot be discharged like the battery storage. This is done to ensure the longest battery life and range of the EV. The process are as follows:

$$C_{ev}(t+1) \;=\; C_{ev}(t) \;+\; \frac{\left(P_{ev}^{ch}(t) - P_{ev}^{dis}(t)\right) \cdot t}{E_{ev}}$$

$$C_{ev}^{min} \;\leq\; C_{ev}(t) \;\leq\; C_{ev}^{max}$$

$$0 \;\leq\; \frac{P_{ev}^{ch}}{\eta_{ch}} \;\leq\; P_{ev}^{ch} \;\cdot\; S_{ev}(t)$$

all the symbols have the same meaning as in battery storage. EV model have one additional constraint where it can only be charged during a specific time range, when the EV is in home.

3.2.5 Power Consumption and Cost

The total amount of power drawn from the grid in a single time slot is given as:

$$P_{total}(t) = \sum_{a=1}^{m+n} \quad S_a(t) \cdot P_a(t) \;+\; P_{rigid}(t) \;+\; P_{batt}(t) \;-\; P_{PV}(t)$$

Total cost of the electricity throughout the day is therefore given as follows :

$$Cost \;=\; \sum_{h=1}^{T} [\; P_{total}(t) \;\cdot\; RTP(t)]$$

Constraint on total duration:

$$\sum_{t=1}^{T} \sum_{a=1}^{m+n} \quad S_a(t) \;=\; \sum_{a=1}^{m+n} d_a$$

Chapter 4

Algorithm Design

In this chapted an energy optimisation scheme is presented which is a modified Genetic algorithm. who belong to the category of evolutionary algorithms. Genetic algorithms are based on natural selection and genetics. that are a subset of evolutionary algorithms.Natural selection and genetics are the foundations of genetic algorithms. The most fit individuals in the population are chosen first by natural selection. They have offspring who inherit the attributes of their parents and are passed down to the next generation. If parents are more fit, their children will be fitter than their parents and have a better chance of survival. This process is repeated until a generation of the fittest individuals is discovered. GA can be used to solve both linear as well as non-linear optimisation; and problems with both continuous and discrete constraint and objective function.

But GA are especially prone to get stuck in local minima and are quite sensitive to the initial conditions. There are several modifications over traditional Genetic algorithms that are designed in the following sections to overcome these shortcomings.

4.1 Coding of Chromosome

A chromosome (also known as a genotype) in genetic algorithms is a set of parameters that define a proposed solution to the problem that the genetic algorithm is attempting to solve. The blueprint of the chromosome of a valid solution is needed, as it represents a valid individual.

Here, the chromosome in the GA has encoded using binary sting, where each bit representing the working state of an appliance at any given time slot. A 0 means that the device turned OFF in that time slot, and 1 represent the ON state. Using this definition the chromosome of a single appliance will be a 1-dimentional string.

$$X_a = \begin{bmatrix} X_a^1 & X_a^2 & X_a^3 & \cdots & X_a^H \end{bmatrix}$$

$$i.e.\ X_a = \begin{bmatrix} X_a^h & , & h \in \{0, H\} \end{bmatrix}$$

For non-intereptable appliance all the 1's in window of operation will be clubbed together. Essentially all the scheduling information in a window of operation on interreptable device ca be boiled down to a single value, the start time of the appliance. After that there will be da number of consecutive 1s followed by 0s. the time of starting is Xa(t) , and the time of ending is Xa(t + da). All conditions of the non-interruptible device can be expressed by only the time of starting with a series of '1's as the genetic code. The total number of ones will be equal to the total working duration of the appliance The set range of operation will be enforced by restricting the start and the end range of the series of '1's

The chromosome of valid shedule is achieved vy compositing the chromosome of all the appliances to generate a global chromosome. The chromosome of the solution is a 2-dimentional entity and is defined below.

$$X \;=\; \{ \quad X_a \quad , \quad a \in [1, m+n] \quad \}$$

$$X \;=\; \begin{bmatrix} X_1^1 & X_1^2 & X_1^3 & \cdots & X_1^H \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ X_m^1 & X_m^2 & X_m^3 & \cdots & X_m^H \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ X_{m+n}^1 & X_{m+n}^2 & X_{m+n}^3 & \cdots & X_{m+n}^H \end{bmatrix}$$

4.2 Generation of Initial population:

The initial population is created randomly at when the algorithm starts to find for optimal schedule. It is mandatory to ensure that all the population in initial population are valid and follow all the constraints. The initialization operator handles the generation of a valid initial population, it can do so using the duration of usage da, and the time range of operation. The intuition behind this is shown below. The initialization operator keeps randomly generating the solutions for an appliance till a valid solution is achieved.

$$X_a \;=\; 00\cdots\underbrace{01101011}00\cdots00$$

$$[\alpha_a, \beta_a]$$

The initial population is ensured to be random as well as uniformly distributed. This will ensure the following

- Sensitivity to the initial solution is reduced
- The convergence performance can be improved

4.3 Fitness function

A fitness function is a type of objective function used to summarise, how close a specified design solution is to achieving the set goals. It is the basis on which the the quality of a individual is judged and the evolutionary process is moved forward.

As the algorithm is seeking to minimise the energy costs, following is the objective function which is used to reduce the effectiveness of the solution into one single variable, that is the total cost.

$$Obj\,(h) = \sum_{t=1}^{T} \left[\left\{ \sum_{a=1}^{m+n} \quad S_a\,(t) \cdot P_a\,(t) \;+\; P_{rigid}\,(t) \;+\; P_{batt}\,(t) \;-\; P_{PV}\,(t) \right\} \cdot RTP\,(t) \right]$$

GA uses the fitness level to determine how much an individual in the given population have the probability to pass on its genotype to the future generation, and its fitness have to be nonnegative value. It is very important to change the objective function so that fitness function is taken as follows:

$$Fitness \;=\; \frac{1}{Obj\,(t) + c}$$

4.4 Selection Process

The roulette wheel selection is used to select the best individuals probabilistically. The individuals which are selected have the opportunity to pass down their genotype to the next generation. The selection procedure involves stochastically selecting from one generation to form the foundation of the next generation.

In roulette wheel selection is a probabilistic selection process. Deterministic methods include selecting the most fit individuals in a population. They're not considered because they produce populations that reach a local maximum and then stop evolving. This mimics nature in that fitter individuals have a better chance of survival and will go on to pass their genotype for the next generation. An individual's likelihood of being chosen is closely correlated with their level of fitness. The most fit individuals have very high chance of survival but even the weakest candidate has a slim chance of being chosen.

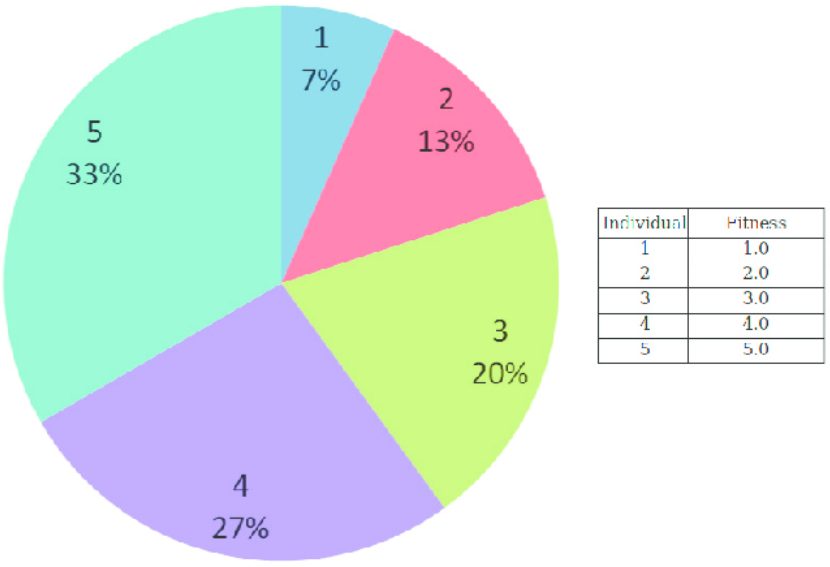

FIGURE 4.1: Roullete Wheel scheduling

4.5 Crossover process

Crossover is the process of selecting two individual and crossing them to produce a new offspring. A genetic operator called crossover is used to change a chromosome's programming from one generation to the next. The crossover probability Pc determines the likelihood that a individual will participate in the crossover procedure.

We are using a modified single point crossover here, which does a row wise single point crossover of the solution. This is same as each appliance selectively doing a single point crossover with its chromosome. In a single-point crossover, the point at which parents exchange information to have children is determined by a randomly selected crossover point. This random point should be within the allowed operable time range.

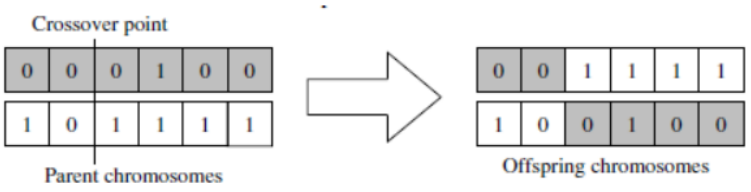

FIGURE 4.2: Single point crossover in a 1-d binary string

There is no use in crossing for the non-interruptible device can be adequately represented by just one bit (start time). As a result, it won't participate, and one of the parent's rows will be chosen at random to be handed on.

4.6 Repair Operator

Crossover operation have chance to make a invalid offspring. To tackle this a repair operator is used to detect and repair ant invalid offspring that may be created. Following are the functions of the Repair Operator:

- After the crossover, the chromosomes must be examined for validity, those that are invalid must be corrected.
- Such invalid chromosomes will be fixed using the repair operator to make them functional.
- The proper genotype will then be created randomly until a valid genotype is reached. The repair operator will first detect how much the invalid genotype deviates from the constraints, of the related appliance.
- After that, until a valid genotype is obtained, the operator will repair genotype will be created at random.

4.7 Mutation process

Mutation is a divergence operation. It us useful in breaking one or more population members out of a local minimuma could lead to the discovery of a superior minima space. Because mutation keeps genetic diversity from being lost in succeeding generations, you can avoid an early convergence on a local maximum or minimum. A mutation probability that is too high results in sluggish or nonexistent convergence. Depending on the genetic variety in the population, it is possible to set up an adaptive process for changing the crossover and mutation probability. We employ both low and high mutation probability to get the best of both worlds.

Following is a brief summary of the working of mutation:

- A individuals likelihood of participating in the mutatation process is defined by the crossover probabilities Pm1 and Pm2.
- Individuals who are involved in the mutation if they have a randomly generated number that is smaller than the mutation probability Pm1.
- The mutation rate is changed to Pm2 when the solution begins to approach the optimal solution, which occurs when For a number of generations, the fitness of the ideal solution has not altered appreciably.
- By decreasing the mutation probability we ensure that the solution does not get stuck is local minima.

4.8 Overview of the modified GA

The GA has the potential to balance the global and local due to the crossover process and mutation cooperating and competing with one another. A mutation procedure is used to replace genotypes those who have been lost during crossover. The population's diversity is preserved while the mutation represents GA's ability to conduct local searches. The GA will prevent starting too soon if the rate of mutation crossover are chosen appropriately.

Specifically because of the following three points, premature convergence can be avoided effectively:

1. The initialization operator can produce a randomly distributed, directional population.

2. The repair operator can abandon local solutions as the population approaches local optimal or inferior solutions in order to increase population diversity.

3. Dynamic rate of mutation : The mutation rate is higher in the early stages of GA to enhance search performance across the board. In order to avoid the destruction of the better solution, the probability of mutation should be reduced in the later part of the algorithm. When the solution starts to approach the optima, mutation probability will be changed.

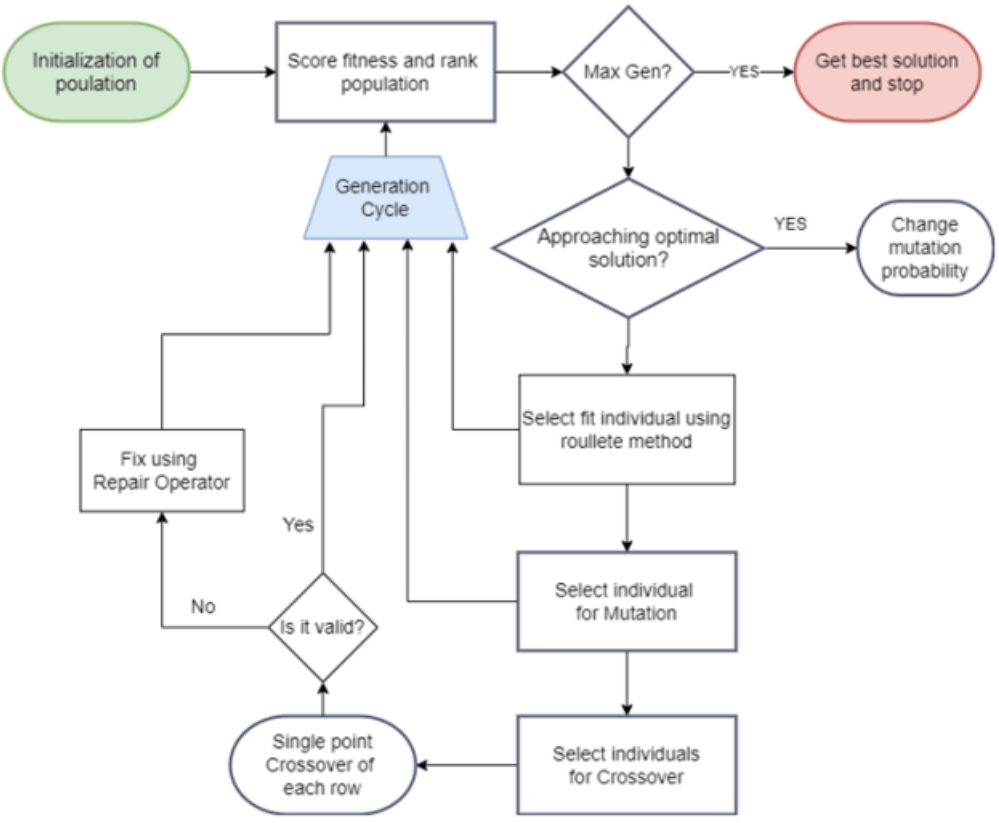

FIGURE 4.3: Flow chart of the Modified Genetic Algorithm

Chapter 5

Experimental setup

The proposed SHEMS architecture is implemented in hardware. In this chapter we will take a look at the implementation, designs, the devices used, and the software needed.

We employed the following hardware and software to design an end-to-end functional prototype of the suggested IoT system :

- NodeMCU ESP8266
- 4 Channel electric Relay
- Blynk IoT platform and Android app
- Arduino IDE
- Proteus Software

5.1 NodeMCU ESP8266

The ESP-12E module, which has an ESP8266 chip with a Tensilica Xtensa 32-bit LX106 RISC microprocessor operating at a configurable clock frequency of 80 to 160 MHz and supporting RTOS, is equipped with the development board.

Additionally, there is enough RAM (128 KB) and Flash memory (4 MB) for software and data storage to process the big strings that make up web pages, JSON/XML data, and everything else used with IoT devices.

The ESP8266 incorporates an 802.11b/g/n HT40 Wi-Fi transceiver, enabling it to establish a network of its own and accept direct connections from other devices in addition to connecting to WiFi networks and interacting with the Internet. This increases the adaptability of the ESP8266 NodeMCU.

The board has an LDO voltage regulator to maintain a constant voltage of 3.3V because the ESP8266's operational voltage range is 3V to 3.6V. When ESP8266 draws up to 80mA during RF broadcasts, its capacity of 600mA should be more than sufficient. Additionally, the regulator's output is divided up and marked with the number 3V3 on one of the board's sides. Power can be provided to external components using this pin.

The on-board MicroB USB connector provides power to the ESP8266 NodeMCU. Alternately, the ESP8266 and its peripherals can be powered directly via the VIN pin if you have a 5V voltage source that is controlled. It have a total of 17 GPIO pins, the ESP8266 NodeMCU is separated out to pin headers on both sides of the development board. These pins can be used for a variety of auxiliary tasks.

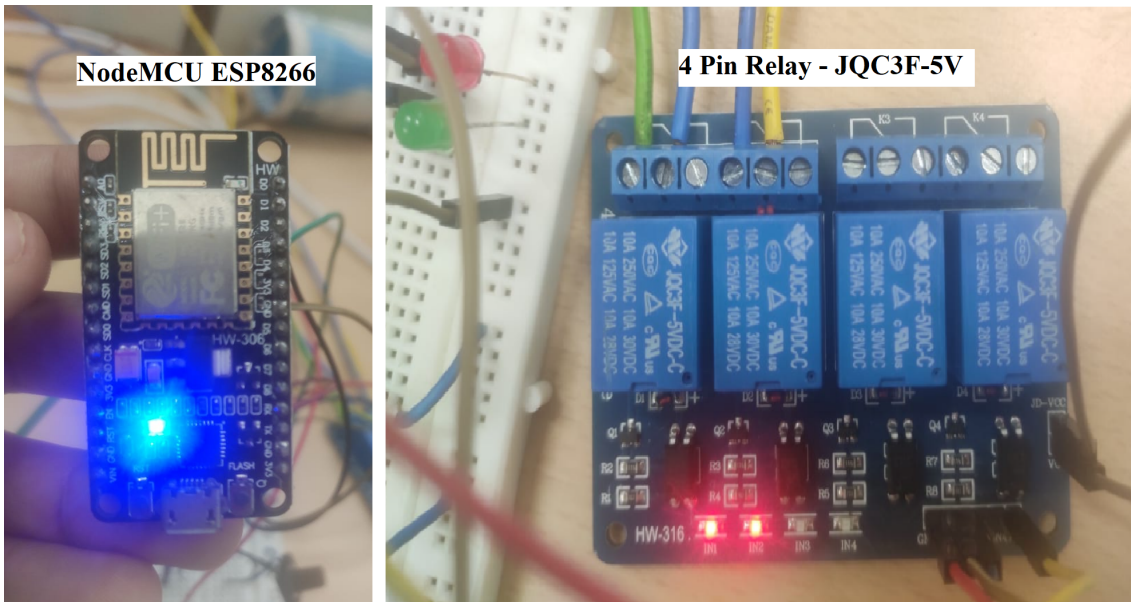

FIGURE 5.1: Node MCU and 4 channel relay with their connections

5.2 4 Channel electric Relay

An electromagnetic coil creates a magnetic field that an electromechanical relay uses to operate when a control signal is given to it. As it has moving contacts in output circuit that are controlled by an electrical signal, it is referred to as electromechanical. A small control signal is to be supplied to input terminals in the input section. The output section is made up of a movable armature and mechanical contacts that are both movable and stationary, and the motion of the armature makes or stops the electrical circuit. The control section has an electromagnetic coil that is activated when a control input signal is applied to the input terminals.

A 120–240V switch coupled to an electromagnet is located inside the relay. The electromagnet charges up and moves the switch contacts open or closed when the relay receives a HIGH signal at the signal pin.

Following are the different modes and pins ofan electric relay:

- Normally closed 120-240V terminal
- Normally open 120-240V terminal
- Common terminal
- Pin Connects to the ground pin on the Arduino
- 5V Vcc pin Connects the Arduino's 5V pin
- Signal pin Carries the trigger signal from the Arduino that activates the relay

The relay has normally open (NO) and usually closed (NC) electrical connections within (NC). Choosing one will depend on whether you want the 5V signal to activate or deactivate the switch. In both arrangements, the relay's common (C) terminal is where the 120-240V supply current enters. Use the NO terminal to operate the normally open connections. Use the NC terminal to operate the normally closed contacts.

NORMALLY OPEN: When the relay gets a HIGH signal in the typically open mode, the 120-240V switch closes, allowing electricity to flow from the C terminal to the NO terminal. The relay is turned off and the current is stopped by a LOW signal. Therefore, utilise the normally open terminal if you want the HIGH signal to switch on the relay:

NORMALLY CLOSED: A HIGH signal opens the switch in the normally closed form, interrupting the 120–240V current. When the switch is closed by a LOW signal, current can move from the C terminal to the NC terminal. Therefore, utilise the normally closed terminal if you want the HIGH signal to cut off the 120-240V current.

5.3 Smart Plug with Wifi

Smart Plug with Wifi A smart plug with wifi capablities and ability to communicate via internet protocol is implemented. The smart home will contain many such plugs connecting to all home appliances. The key component here is the NodeMCU. NodeMCU is a microcontroller with wifi module. It is inexpensive and compact making it ideal for the purpose of a amrt plug.

The device turn the appliances ON/OFF is the electric relay. It receives the signal form the nodeMCU and turn the appliances on or off as per the instructions given by the microcontroller. NodeMCU is connected to a cloud server, which acts as the central node of the HEMS. This provides a great deal of flexiblity to the user as the EMS is not constraint to the home and user can remotely monitor as well as control the home.

Physical buttons are also connected in the system, as the users are most accustomed to them. When an appliance is turened on usig a button, the microcontroller detects it and send a signal to the cloud server. The central can then either schedule the appliance for a later time, or alternatively turn the device on and update its status on all the dashboards and interfaces.

The programming of the ndoeMCU is done using ArduinoIDE. NodeMCU is programmed in C, and a number of libraries are needed for this work. Key among those is the blynk library which provides method for connection and interfacing with the cloud server. The board must also be installed in the ArduinoIDE for ot to successfully burn the firmware into the microcontoller.

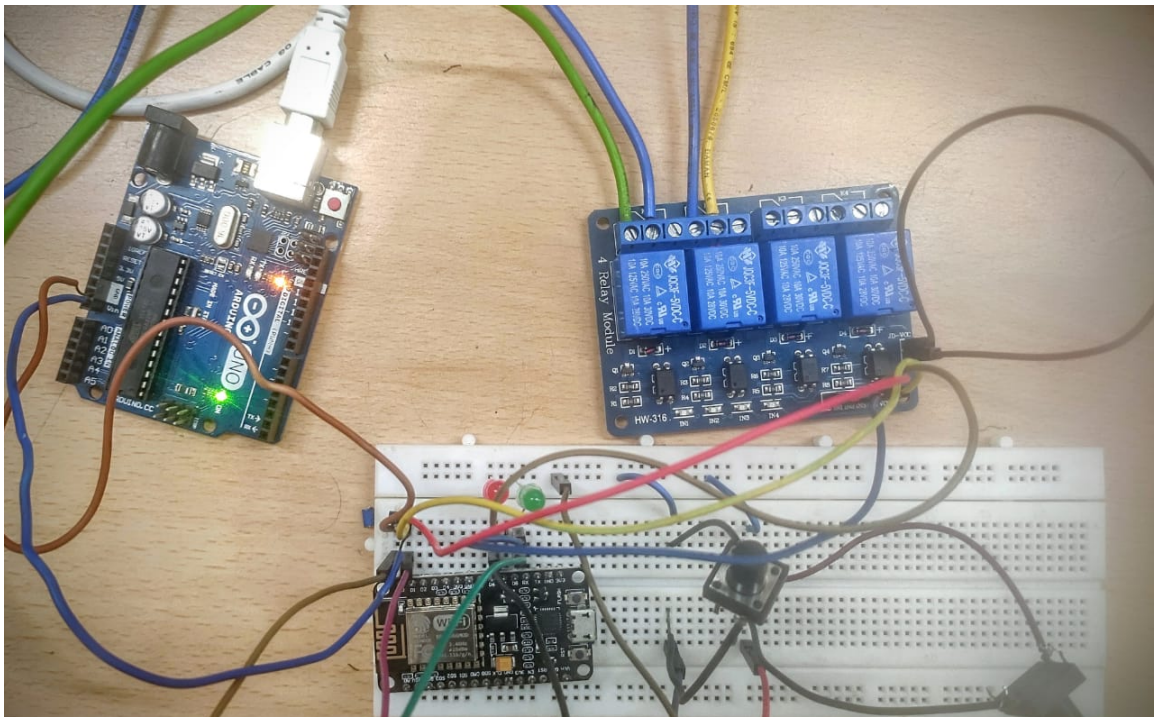

FIGURE 5.2: Experimental setup for smart plug

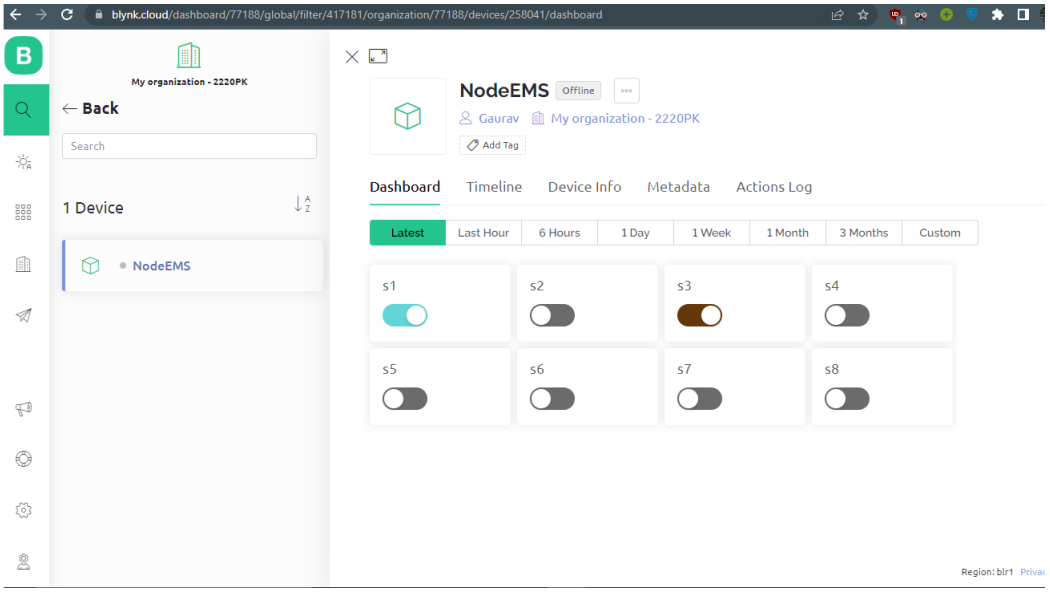

FIGURE 5.3: Screenshot of Blynk Dashboard

5.4 User interface using Blynk

Here we are using a popular IoT platform called Blynk to create the user interface and the cloud server. The smart plug modules are connected to the internet using the wifi module of nodeMCU. NodeMCU can connect to any open or closed wifi network using the ssid and password of that network. Once connected, nodeMCU can communicate via internet using internet protocols to send and receives data and instructions.

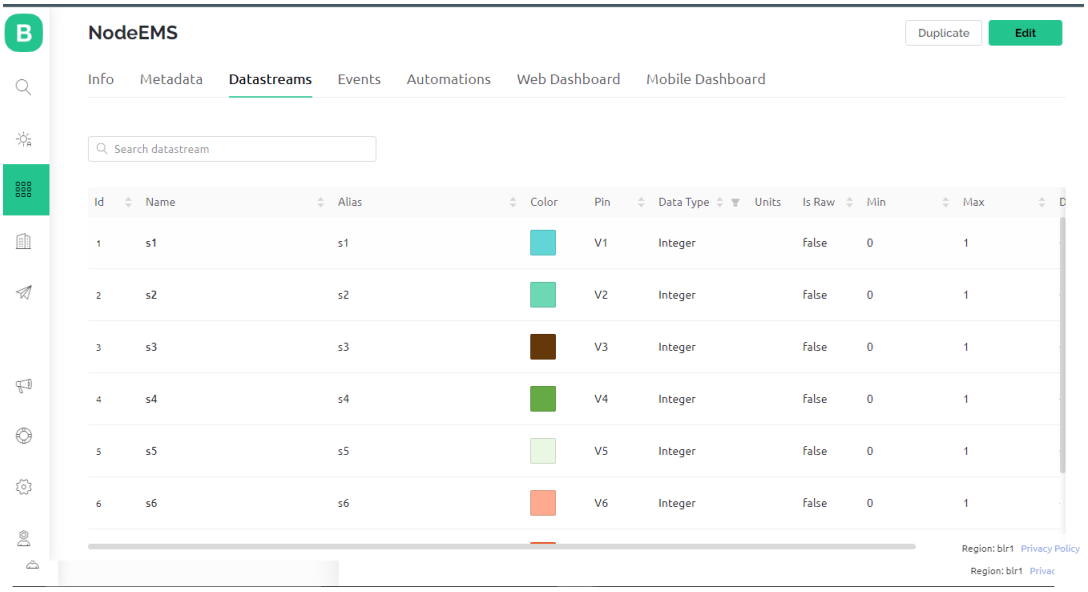

FIGURE 5.4: Blynk datastreams

Blynk is an IoT platform which allows users to create dashboard and set up cloud servers for their IoT network. Blynk make is very accessible to create user interfaces and dashboards, to effectively control and monitor IoT systems. Blynk can connect with devices via USB, bluetooth ar over internet. Although setting up connection over internet goves best reliablitu and versatility.

When a IoT system is created in Blynk platform, it provides a unique name as well as a unique authentication token for that project. This token must be added while

coding the microcontroller and then should be burned into the firmware. Using the unique authentication token, the IoT devices are able to login to the blynk system, hance being able to send and receive data and instructions. Blynk libraries are needed to be installed in order to perform all the above tasks. These libraries can be easily found in the library manager of the Arduino IDE.

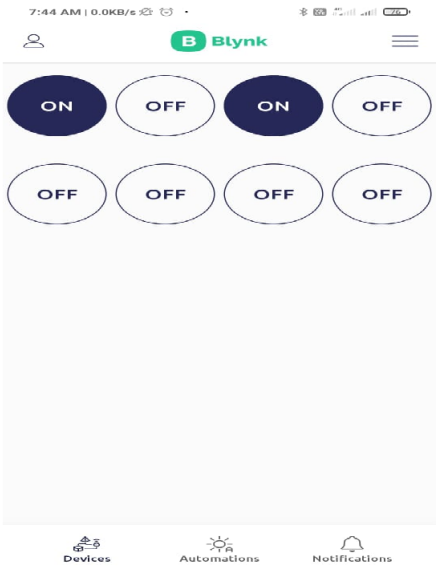

FIGURE 5.5: Blynk GUI on smartphone

Using the dashboard feature in blink User interfaces is created. The smart plugs can be controlled using these the dashboard. The mobile app of blynk is also downloaded and a graphical user interface to monitor and control the smart plugs is setup. This provides a lot of flexibility to the user as it allies remote monitoring and control of the home appliances that too from a variety of devices. The switches can also be turned on or off manually using the physical buttons, in this case the smart plug detects this and send signal to the cloud server, which can update the device status or even schedule the appliance for a later time.

5.5 Online Load Sheduling

The system can implement a optimal schedule from the HEMS, but it is also capable of handling the loads online without forecasting. Here a very simple online load scheduling algorithm is implemented for the sake of simplicity.

The algorithms use only real-time pricing into consideration as there is no integrated PV generation in our hardware implementation. The algorithm successfully trips low priority load when the price is higher and schedule them for a later time.

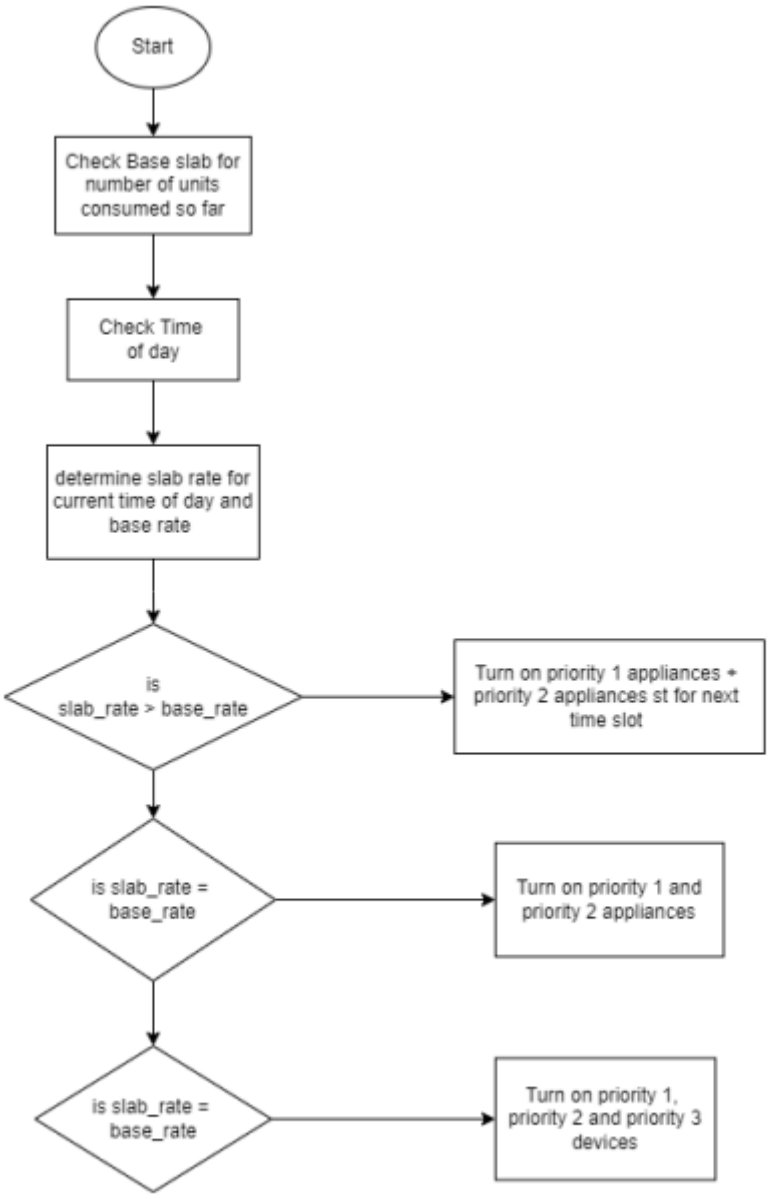

FIGURE 5.6: Online load sheduling using RTP

Chapter 6

Results and Discussion

The whole smart home and the appliances are mathematically modeled using Python, and then the proposed modified GA is implemented and the whole system is simulated. Finally, a end-to-end working prototype of the proposed HEMS is implemented in hardware using IoT devices and services. This chapter discusses the key results of the work.

6.1 Results of Simulation

24 hour simulation is performed where the modified GA is trained using the user usage data to search the optimal schedule. The real-time pricing of electricity is used, figure 6.2 refers to the pricing model used in the analysis.

Appliance	[α , β]	Duration (h)	Rated Power (kW)
LED Bulb	6:00 - 9:00	-	0.4
	18:00 - 00:00	-	
Television	8:00 - 10:00	-	0.1
	19:00 - 23:00	-	
Air conditioner	00:00 - 8:00	-	0.75
	16:00 - 00:00	-	
Humidifier	00:00 - 9:00	4	0.15
	14:00 - 20:00	4	
Water heater	4:00 - 8:30	3	0.74
	16:00 - 20:00	2	
	21:00 - 00:00	1.5	
Water Pump	00:00 - 8:00	3	2
	7:00 - 18:00	4	
	16:00 - 00:00	4	
Dish washer	8:00 - 12:00	1.5	0.73
	20:00 - 23:00	1.5	
Waching machine	6:00 - 7:30	1	0.8
	15:00: 17:00	1	
EV	00:00 - 6:00	3.5	2.5

FIGURE 6.1: Parameters of simulation of appliances

Above is one example of the parameters of the appliance in a smart home. Here the expected duration of usage, the allowable time range for operation and the power rating of each device are presented, this data is used to get the optimal schedule.

The following variable pricing model is used to proce electricity form the grid:

Time of Day (Hours)	*Additional Tariff (paise)*
6:00-10:00	+100
10:00-18:00	Flat Tariff
18:00-22:00	+100
22:00-6:00	-100

FIGURE 6.2: Real Time Pricing used

To see the effect of different models of smart home, and observe how various modules affect the electricity cost, 5 different smart home models are considered. In each new smart home model, a separate module is added starting with a traditional household without PV or battery storage. Thereafter battery, PV and ability to sell to the grid are added individually. The results of this analysis is shown in table 6.3 where the Y or whether the system is capable of selling power to grid, and in case of PV and battery storage, whether the system contains the corresponding module. Cost of all the different models are compared and the improvement is noted. Improvement from one model to other shows how much a given module effects the overall cost savings which gives us valuable insights into the system.

It can be clearly noted from the table that the total cost of a non-smart home that lacks all modules is Rs.134.5, whereas the cost of the model with all the modules present is Rs. 43.1, this is a massive improvement and justify the adoption of the smart home. Renewable energy capability plays the most role in cost reductions as can be noted from the massive 48.4% improvement over the non-smart home in case 1. The rest two models lack renewable energy, but provide a good amount of saving with comparable performance and effectiveness. So, it is observed that the developed model have huge potential for cost savings and energy utilization. The rest of the results are focused in the complete smart home as in case 5. This provides most amount of options and optimizations.

Case	PV	Battery Storage	Sell to grid	Electricity cost (Rs)	Improvement (%)
1	N	N	N	134.5	-
2	Y	N	N	69.33	48.4
3	Y	Y	N	63.71	8.1
4	Y	N	Y	56.69	11.1
5	Y	Y	Y	43.1	-

FIGURE 6.3: Cost of different smart home models

Following are the key system parameters for the following simulation:

$$Battery\ capacity = 3kVh$$
$$Battery\ charging\ rate = 1kW$$
$$Charging\ efficiency = 0.8$$
$$Discharging\ rate = 1kW$$
$$Discharge\ efficiency = 0.8$$
$$Maximum\ allowable\ charge = 0.9$$
$$Minimum\ allowable\ charge = 0.1$$
$$Max\ PV\ Generation = 3kV$$
$$Real\ Time\ Pricing = Fig.6.2$$
$$Initial\ state\ of\ battery = 0.25 + 0.75 * rand(0, 1)$$

The figure 6.4 shows the amount of energy drawn from the grid by the HEMS. When compared to stochastic optimisation their is significant reduction is power drawn. Moreover the HEMS also shifts the loads away from the peak which will ensure better stability of the utility.

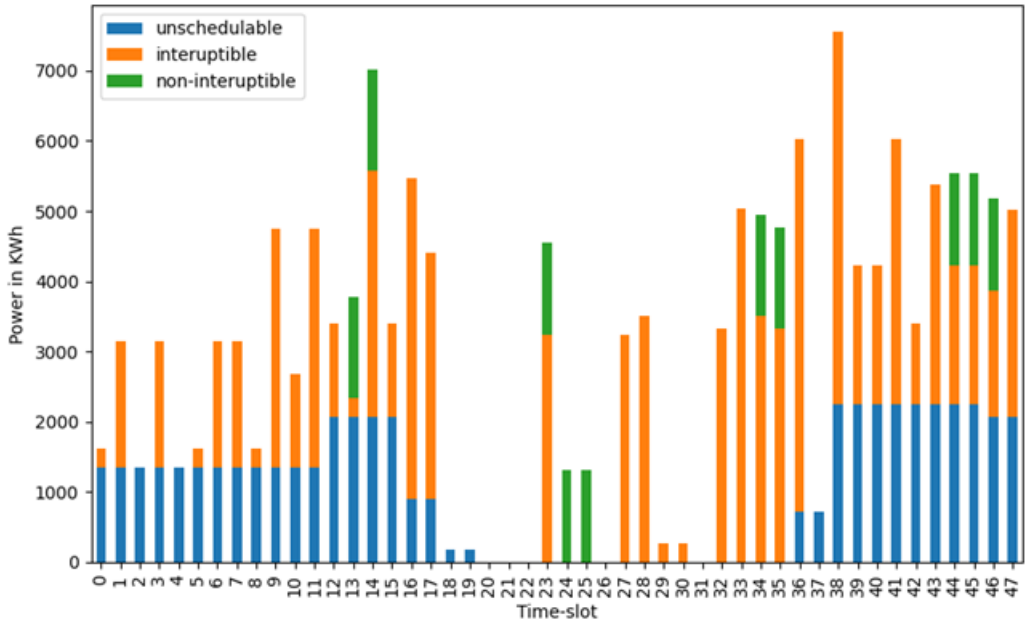

FIGURE 6.4: Results of the algorithm optimisation throughout the day

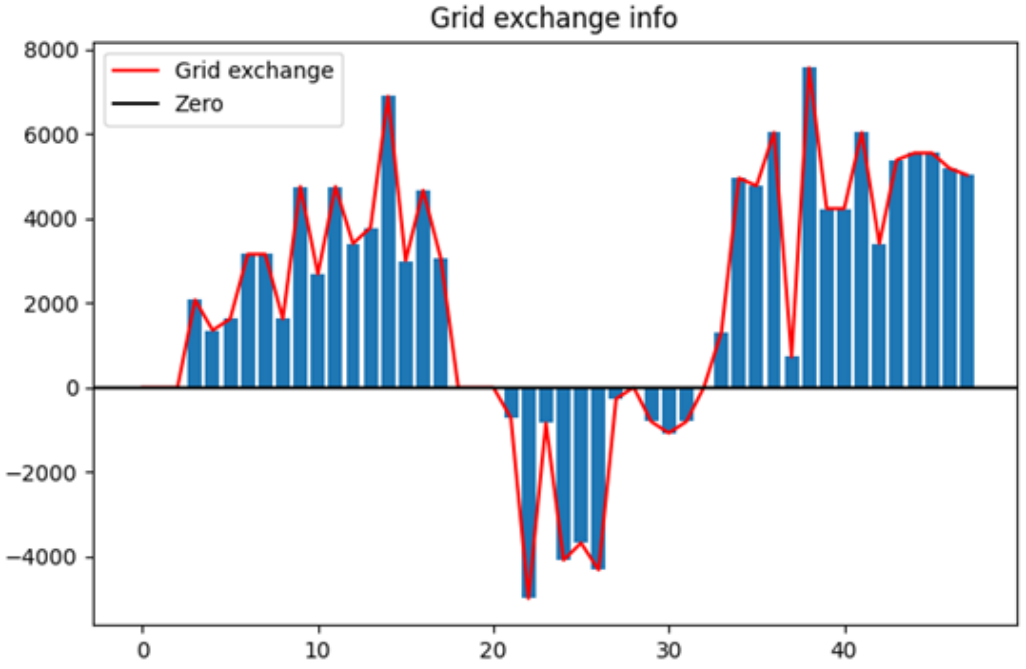

FIGURE 6.5: Exchange of power between the grid and the HEMS throughout the day

The fig 6.5 shows the transfer of power between the grid and the HEMS. The HEMS have the ability to sell excess electricity back to the grid. In this simulation, the price to sell power back to the grid is set as 50% of the RTP of electricity. It is seen that when the price is low, the HEMS draws power form the grid to run appliances, but during high price period as the PV generation also rises the HEMS sells the power back to the grid at a good price. When the PV generation drops during the high price period home is supplied using the battery to save cost. It can be concluded that the modified GA comes up with smart strategies to minimize the cost to user. This smart decison making is also seen in Fig. 6.6

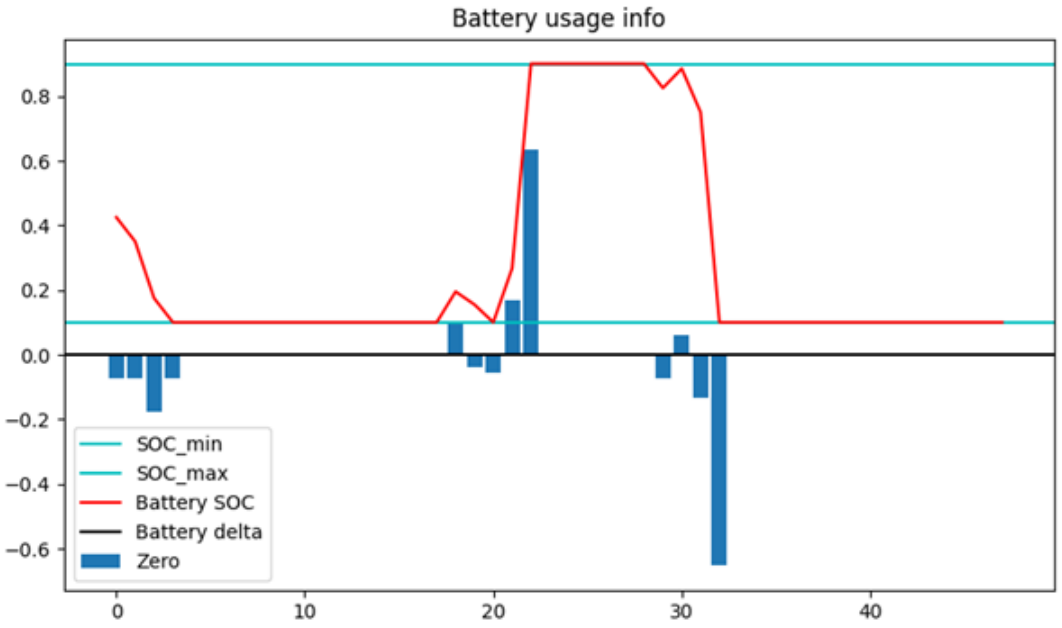

FIGURE 6.6: State of charge of battery

6.2 Convergence and StdDev

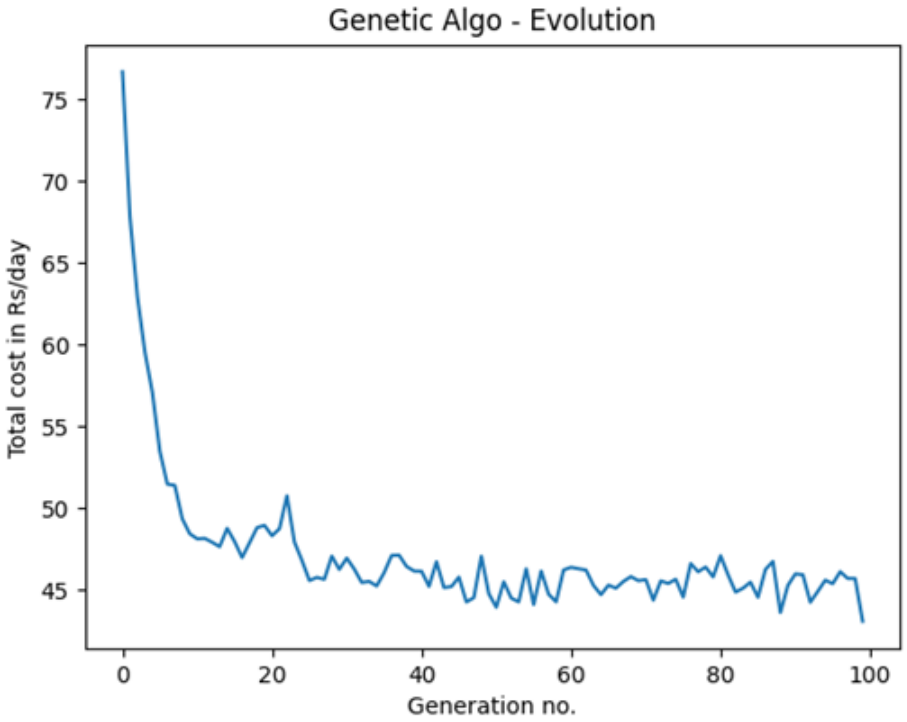

FIGURE 6.7: Convergence of optimal solution with the GA evolution

The above figure shows the convergence performance of the algorithm. Before the first 20 generations of the evolutionary process, the discrepancy between the optimum value and the mean valueis relatively considerable, indicating that there are many useless schemas in the population. After 50 generations, the variance gets lower, indicating that all population options are excellent. In other words, all the unfit genotypes are eliminated and most of the surviving population have a high degree of fitness. Only the fittest is retained after the initial generations.

It can be seen that throughout the course of around 25 iterations, the price of electricity has decreased steadily and afterward it has reached a point of saturation.

This finding suggests that iterations are enough for GA with current parameters to achieve an optimal solution.

The standard deviation of each generation is also measured to evaluate the performance of the algorithm. High StdDev usually means that the solution is scattered and therefore there is a high likelihood of the solution being stuch in local minima.

StdDev of generation 100 = 6.56

This is a relatively small value (less than 15% of the solution) which indicates that the algorithm effectively handles these issues.

Following are the major parameter of the algorithm in the simulation:

$$Number\ of\ individual\ per\ generation\ =\ 100$$

$$Number\ of\ generations\ =\ 100$$

$$Chromosome\ precision\ =\ [48 * 10]$$

$$Selection\ rate\ =\ 0.1$$

$$Mutation\ probability\ 1\ =\ 0.05$$

$$Mutation\ probability\ 2\ =\ 0.025$$

$$Crossover\ Probability\ =\ 0.5$$

The analysis of convergence algebra shows that Modified GA undergoes fewer iterations than the others similar Heuristic-based algorithms. Modified GA has a significant benefit in discovering the best patterns from among chromosomes fast, which speeds up the search for the best solution.

6.3 Analysis of Robustness

The Optimizing algorithm is based on using forecast data like users' preferences and behavior, solar generation, and variable pricing. The performance of scheduling may differ as a result of the discrepancy between the expected value and true value. To analyze this, we swap out the prediction data and add stocastic and random noise to make it closer to real data, and check the optimization's capacity to resist disturbance. This is called robust optimization. We want to do is to investigate the relationship between solar utilization and cost with respect to uncertainty in the prediction of PV. The power cost's sensitivity along with the robustness in relation to various optimizations are shown in the following figure.

Parameter	Stocastic Sheduling	Robust Optimisation	Modified GA Optimisation
PV utilisation (%)	65.28	95.32	100
Power to grid (kWh)	7.45	12.8	13.2
Cost (Rs)	74.6	45.63	43.1

FIGURE 6.8: Robustness analysis

In all three situations, the overall energy usage is essentially the same. In other words, the scheduling method little affects the overall amount of energy used. But in all scenarios, there are significant variances in the cost of electricity, energy sold to the grid, and utilization of PV. Modified GA has almost the same utilization as robust optimization.

6.4 Analysis of sensitivity to initial conditions

To observe how the algorithm reacts under various initial conditions and system parameters and specifications, sensitivity analysis is required. The cost outcome is seen for various PV generation and battery parameters capacity specifications. As expected better capacity leads to more energy savings as it provides the HEMS with more flexibility to schedule appliances.

Case	Peak PV generation (kW)	Battery Capacity (kWh)	Cost (Rs)
1	3	3	46.64
2	3	4	44.51
3	3	5	43.1
4	5	3	37.59
5	5	4	35.53
6	5	5	33.46

FIGURE 6.9: Sensitivity analysis of PV generation and battery capacity

Expanding the battery's capacity can help you save money on electricity. Therefore, installing large-capacity energy storage solutions as much as feasible is more cost-effective for users. Even though the early stage investment costs are high, the long-term benefit is clear.

Case	Price to Sell (Rs)	Energy sold to grid (kWh)	Electricity Cost (Rs)
1	30% of RTP	20.5	71.36
2	50% of RTP	13.2	43.1
3	Average of RTP	15.1	51.22
4	2.25	18	57.41
5	3.25	15.75	52.84

FIGURE 6.10: Sensitivity to selling Price

It is seen that the price at which electricity is sold to the grid is important. The RTP is connected in examples 1 and 2. These two modes have the benefit of allowing the user to sell power to the utility for a high price. More power is required in the system at this time because the load peak always coincides with the high RTP. This will shift load away from the peak. They and the grid will both profit if users sell electricity during this time.

Chapter 7

Conclusion

In this thesis project, a SHEMS model is proposed to address a number of the problems in smart homes today. A modified genetic algorithm is designed and developed to solve the problem of optimal energy scheduling. Many improvements over traditional GA like dynamically changing mutation rates, uniform and random initialization, reparing to invalid solution are implemented. These overcome the drawbacks of getting trapped in local optima and the sensitivity to initial conditions and solutions. The smart home is modelled and simulated and very good cost savings are observed with the approach. In addition the proposed method is much more robust and less sensitive to errors.

Finally, an end to end IoT based hardware prototype is of the system is created and connected over the internet using wifi. The smart switch is designed and implemented which can communicate with cloud server to monitor and control the smart home.

Bibliography

[1] N. U. Rehman and H. Rahim, ""heuristic algorithm based energy management system in smart grid."."

[2] A. Al-Duais and M. Osman, "Optimal real-time pricing-based scheduling in home energy management system using genetic algorithms."

[3] U. Latif and N. Javaid, "Cost optimization in home energy management system using genetic algorithm, bat algorithm and hybrid bat genetic algorithm."

[4] F. Alfaverh and M. D. Y. Sun, "Demand response strategy based on reinforcement learning and fuzzy reasoning for home energy management."

[5] S. Ranjan and M. S. Thomas, "An efficient home energy management algorithm for demand response analysis in indian scenario."

[6] B. Yuce and M. M. Y. Rezgui, "Annaga smart appliance scheduling for optimised ˆ energy management in the domestic sector."

[7] O. S. T. Ueno, F. Sano, "Effectiveness of an energy-consumption information system on energy savings in residential houses based on monitored data,."

[8] A. Radhakrishnan and M. P. Selvan, "Load scheduling for smart energy management in residential buildings with renewable sources." in power systems conference (npsc),."

[9] R. Kaur and C. Schaye, "Machine learning and price-based load scheduling for an optimal iot control in the smart and frugal home."

[10] V. M. andHaris Doukas, "An advanced iot-based system for intelligent energy management in buildings."

[11] J. B. Jongbae Kim, "An iot-based home energy management system over dynamic home area networks."

[12] "Design and development of advanced smart energy management system integrated with iot framework in smart grid environment."

Code for smart plug

```
#define BLYNK_TEMPLATE_ID "TMPLSwqA0-jU"
#define BLYNK_DEVICE_NAME "NodeEMS"
#define BLYNK_AUTH_TOKEN "CFu6-QgK8HEna2ZNbap05NJP7MEWUZJe"


char ssid[] = "barmola";
char pass[] = "ggg000ggg";

bool fetch_blynk_state = true;

//#define BLYNK_PRINT Serial
#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>
#include <AceButton.h>
using namespace ace_button;


#define RelayPin1 5  //D1
#define RelayPin2 4  //D2
#define RelayPin3 14 //D5
#define RelayPin4 12 //D6

#define SwitchPin1 10  //SD3
#define SwitchPin2 D3   //D3
#define SwitchPin3 13  //D7
#define SwitchPin4 3   //RX

#define wifiLed   16   //D0
```

```
#define VPIN_BUTTON_1    V1
#define VPIN_BUTTON_2    V2
#define VPIN_BUTTON_3    V3
#define VPIN_BUTTON_4    V4

// Relay State
bool toggleState_1 = LOW; //Define integer to remember the toggle state for relay 1
bool toggleState_2 = LOW; //Define integer to remember the toggle state for relay 2
bool toggleState_3 = LOW; //Define integer to remember the toggle state for relay 3
bool toggleState_4 = LOW; //Define integer to remember the toggle state for relay 4

int wifiFlag = 0;

char auth[] = BLYNK_AUTH_TOKEN;

ButtonConfig config1;
AceButton button1(&config1);
ButtonConfig config2;
AceButton button2(&config2);
ButtonConfig config3;
AceButton button3(&config3);
ButtonConfig config4;
AceButton button4(&config4);

void handleEvent1(AceButton*, uint8_t, uint8_t);
void handleEvent2(AceButton*, uint8_t, uint8_t);
void handleEvent3(AceButton*, uint8_t, uint8_t);
void handleEvent4(AceButton*, uint8_t, uint8_t);

BlynkTimer timer;

// When App button is pushed - switch the state

BLYNK_WRITE(VPIN_BUTTON_1) {
  toggleState_1 = param.asInt();
  digitalWrite(RelayPin1, !toggleState_1);
}

BLYNK_WRITE(VPIN_BUTTON_2) {
  toggleState_2 = param.asInt();
  digitalWrite(RelayPin2, !toggleState_2);
}
```

```
BLYNK_WRITE(VPIN_BUTTON_3) {
  toggleState_3 = param.asInt();
  digitalWrite(RelayPin3, !toggleState_3);
}

BLYNK_WRITE(VPIN_BUTTON_4) {
  toggleState_4 = param.asInt();
  digitalWrite(RelayPin4, !toggleState_4);
}

void checkBlynkStatus() { // called every 2 seconds by SimpleTimer

  bool isconnected = Blynk.connected();
  if (isconnected == false) {
    wifiFlag = 1;
    Serial.println("Blynk Not Connected");
    digitalWrite(wifiLed, HIGH);
  }
  if (isconnected == true) {
    wifiFlag = 0;
    if (!fetch_blynk_state){
    Blynk.virtualWrite(VPIN_BUTTON_1, toggleState_1);
    Blynk.virtualWrite(VPIN_BUTTON_2, toggleState_2);
    Blynk.virtualWrite(VPIN_BUTTON_3, toggleState_3);
    Blynk.virtualWrite(VPIN_BUTTON_4, toggleState_4);
    }
    digitalWrite(wifiLed, LOW);
    Serial.println("Blynk Connected");
  }
}

BLYNK_CONNECTED() {
  // Request the latest state from the server
  if (fetch_blynk_state){
    Blynk.syncVirtual(VPIN_BUTTON_1);
    Blynk.syncVirtual(VPIN_BUTTON_2);
    Blynk.syncVirtual(VPIN_BUTTON_3);
    Blynk.syncVirtual(VPIN_BUTTON_4);
  }
}
```

```
void setup()
{
  Serial.begin(9600);

  pinMode(RelayPin1, OUTPUT);
  pinMode(RelayPin2, OUTPUT);
  pinMode(RelayPin3, OUTPUT);
  pinMode(RelayPin4, OUTPUT);

  pinMode(wifiLed, OUTPUT);

  pinMode(SwitchPin1, INPUT_PULLUP);
  pinMode(SwitchPin2, INPUT_PULLUP);
  pinMode(SwitchPin3, INPUT_PULLUP);
  pinMode(SwitchPin4, INPUT_PULLUP);

  //During Starting all Relays should TURN OFF
  digitalWrite(RelayPin1, !toggleState_1);
  digitalWrite(RelayPin2, !toggleState_2);
  digitalWrite(RelayPin3, !toggleState_3);
  digitalWrite(RelayPin4, !toggleState_4);

  digitalWrite(wifiLed, HIGH);

  config1.setEventHandler(button1Handler);
  config2.setEventHandler(button2Handler);
  config3.setEventHandler(button3Handler);
  config4.setEventHandler(button4Handler);

  button1.init(SwitchPin1);
  button2.init(SwitchPin2);
  button3.init(SwitchPin3);
  button4.init(SwitchPin4);

  //Blynk.begin(auth, ssid, pass);
  WiFi.begin(ssid, pass);
  timer.setInterval(2000L, checkBlynkStatus); // check if Blynk server is connected every
  Blynk.config(auth);
  delay(1000);

  if (!fetch_blynk_state){
    Blynk.virtualWrite(VPIN_BUTTON_1, toggleState_1);
```

```
    Blynk.virtualWrite(VPIN_BUTTON_2, toggleState_2);
    Blynk.virtualWrite(VPIN_BUTTON_3, toggleState_3);
    Blynk.virtualWrite(VPIN_BUTTON_4, toggleState_4);
  }
}

void loop()
{
  Blynk.run();
  timer.run(); // Initiates SimpleTimer

  button1.check();
  button2.check();
  button3.check();
  button4.check();
}

void button1Handler(AceButton* button, uint8_t eventType, uint8_t buttonState) {
  Serial.println("EVENT1");
  switch (eventType) {
    case AceButton::kEventReleased:
      Serial.println("kEventReleased");
      digitalWrite(RelayPin1, toggleState_1);
      toggleState_1 = !toggleState_1;
      Blynk.virtualWrite(VPIN_BUTTON_1, toggleState_1);
      break;
  }
}
void button2Handler(AceButton* button, uint8_t eventType, uint8_t buttonState) {
  Serial.println("EVENT2");
  switch (eventType) {
    case AceButton::kEventReleased:
      Serial.println("kEventReleased");
      digitalWrite(RelayPin2, toggleState_2);
      toggleState_2 = !toggleState_2;
      Blynk.virtualWrite(VPIN_BUTTON_2, toggleState_2);
      break;
  }
}
void button3Handler(AceButton* button, uint8_t eventType, uint8_t buttonState) {
  Serial.println("EVENT3");
```

Python Code for simulation

.1 main.py

```
from unschedulable_app import Unschedulable
from interuptible_app import Interuptible
from non_interuptible_app import NonInteruptible

from photovoltaic import PV_cell
from battery import Battery
from household import Household

num_slots = 48

# Configure the Photo-voltaic cell
pv_area = 30.0       # in m2
pv_intensity = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2.5, 7.5, 15.0, 25.0, 35.0, 50.0, 65.0,
                92.5, 80.0, 65.0, 50.0, 35.0, 25.0, 15.0, 7.5, 2.5, 0, 0, 0, 0, 0, 0, 0, 0, 0,
pv_power = [pv_area*x/1000.0 for x in pv_intensity]     # in KW
pv_cell = PV_cell(pv_power)

# Configuring the Battery
total_capacity_KWh = 5             # In units
total_capacity_KJ = total_capacity_KWh*60*60        # In KJ
charge_efficiency = 2.0             # In KW
discharge_efficiency = 2.0          # In KW
SOC = 0.5
battery = Battery(total_capacity_KJ, charge_efficiency,
                  discharge_efficiency, SOC)

# Configuring the Appliances
```

```
app_types = [Unschedulable, Interuptible, NonInteruptible]

names = ["LED Bulb", "Television", "Air conditioner", "Humidifier",
         "Water heater", "Water Pump", "Dish washer", "Waching machine"]
ids = [0, 0, 0, 1, 1, 1, 2, 2]
slots = [[[12, 18], [36, 48]], [[16, 20], [38, 46]], [[0, 16], [38, 48]], [[0, 18, 8], [28, 40, 8]
    42, 48, 3]], [[0, 16, 6], [14, 36, 8], [32, 48, 8], [36, 48, 8]], [[16, 24, 3], [40, 46, 3]],
powers = [[0.4, 0.4], [0.1, 0.1], [0.75, 0.75], [0.15, 0.15], [
    0.74, 0.70, 0.64], [1.0, 1.8, 1.0, 1.1], [0.73, 0.73], [0.8, 0.8]]

allAppliances = []
for i, x in enumerate(zip(names, ids, slots, powers)):
    name, id, slot, power = x
    appliance = app_types[id](name, slot, power)
    allAppliances.append(appliance)

# Configuring Real-time pricing environment
rtp = [225, 225, 225, 225, 225, 225, 225, 225, 225, 225, 225, 225, 425, 425, 425, 425, 425, 425, 4
       325, 325, 325, 325, 325, 325, 325, 325, 325, 325, 325, 425, 425, 425, 425, 425, 425, 425, 4
rtp_KJ = [x/(100*60*60) for x in rtp]      # in Rs per KJ

household = Household(allAppliances, battery, pv_cell, rtp_KJ)
household.make_mapping()

schedule = household.random_scheduler()

# total_cost = household.simulate(schedule, visualize=True)
total_cost = household.simulate(schedule)

print("Total cost in Rs for Random scheduling: ", total_cost)

schedule = household.genetic_scheduler()
total_cost = household.simulate(schedule)
# for Plots
# schedule = household.genetic_scheduler(visualize=True)
# total_cost = household.simulate(schedule, visualize=True)

print("Total cost in Rs for Genetic scheduling: ", total_cost)
```

.2 interreptible-app.py

```
"""
Non-interuptible appliances have to be scheduled in the given time slot in range [a,b] for c times
"""

class Interuptible:
    def __init__(self,name,slots,power):
        self.name = name
        self.slots = slots
        self.power = power

    def __str__(self):
        s = f"{self.name} : This an interuptible load\n"
        for i in range(len(self.slots)):
            s += f"It must operate for {self.slots[i][2]} slots in [{self.slots[i][0]}-{self.slots
        return s
```

.3 non-interreptible-app.py

```
"""
Non-interuptible appliances have to be scheduled in the given time slot in range [a,b] for c times
"""

class NonInteruptible:
    def __init__(self,name,slots,power):
        self.name = name
        self.slots = slots
        self.power = power

    def __str__(self):
        s  = f"{self.name} : This an non-interuptible load\n"
        for i in range(len(self.slots)):
            s += f"It must operate for {self.slots[i][2]} slots in [{self.slots[i][0]}-{self.slots
        return s
```

.4 unschedulable.py

```
"""
Unschedulable appliances have to be scheduled in the mentioned slots else the customer satisfactio
"""

class Unschedulable:
    def __init__(self,name,slots,power):
        self.name = name
        self.slots = slots
        self.power = power

    def __str__(self):
        s = f"{self.name} : This an unschedulable load\n"
        for i in range(len(self.slots)):
            s += f"It must operate throughout [{self.slots[i][0]}-{self.slots[i][1]}] with power {
        return s
```

.5 battery.py

```
class Battery:
    def __init__(self,total_capacity,charge_efficiency,discharge_efficiency,SOC):
        self.total_capacity = total_capacity
        self.charge_efficiency = charge_efficiency
        self.discharge_efficiency = discharge_efficiency

        self.SOC = SOC
        self.SOC_min = 0.1
        self.SOC_max = 0.9

    def update(self,delta_energy):
        # positive delta indicates surplus
        if(delta_energy>0.0):
            Bcc = (self.SOC_max - self.SOC)*self.total_capacity
            Bce = self.charge_efficiency*30*60
            if(Bcc<Bce):
                if(delta_energy>Bcc):
                    self.SOC = self.SOC_max
                    return delta_energy-Bcc
```

```
                else:
                    self.SOC += delta_energy/self.total_capacity
                    return 0.0
            else:
                if(delta_energy>Bce):
                    self.SOC = self.SOC_max
                    return delta_energy-Bce
                else:
                    self.SOC += delta_energy/self.total_capacity
                    return 0.0
        else:
            delta_energy *= -1
            Bdc = (self.SOC - self.SOC_min)*self.total_capacity
            Bde = self.discharge_efficiency*30*60
            if(Bdc<Bde):
                if(delta_energy>Bdc):
                    self.SOC = self.SOC_min
                    return -(delta_energy-Bdc)
                else:
                    self.SOC -= delta_energy/self.total_capacity
                    return 0.0
            else:
                if(delta_energy>Bde):
                    self.SOC = self.SOC_min
                    return -(delta_energy-Bde)
                else:
                    self.SOC -= delta_energy/self.total_capacity
                    return 0.0
```

.6 household.py

```
from matplotlib import style
from unschedulable_app import Unschedulable
from interuptible_app import Interuptible
from non_interuptible_app import NonInteruptible

import random
import numpy as np
import pandas as pd
```

```
import matplotlib.pyplot as plt


class Household:
    def __init__(self, appliances, battery, pv_cell, rtp):
        self.num_slots = 48
        self.app_types = [Unschedulable, Interuptible, NonInteruptible]
        self.appliances = appliances
        self.battery = battery
        self.SOC_init = battery.SOC
        self.pv_cell = pv_cell
        self.rtp = rtp
        self.mapping = "NA"

    def make_mapping(self):
        self.mapping = []
        for i, appliance in enumerate(self.appliances):
            if(type(appliance) == self.app_types[0]):
                for j in range(len(appliance.slots)):
                    self.mapping.append((0, (i, j)))
            elif(type(appliance) == self.app_types[1]):
                for j in range(len(appliance.slots)):
                    self.mapping.append((1, (i, j)))
            else:
                for j in range(len(appliance.slots)):
                    self.mapping.append((2, (i, j)))

    def simulate(self, schedule, visualize=False):
        total_cost = 0.0

        unschedulable_load = [0.0 for _ in range(self.num_slots)]
        interuptible_load = [0.0 for _ in range(self.num_slots)]
        non_interuptible_load = [0.0 for _ in range(self.num_slots)]

        grid_deltas = []

        prev_SOC = self.battery.SOC
        battery_delta = []
        battery_SOC = []
        for j in range(self.num_slots):
            # Compute total energy consumption from various appliances
            consumption_energy = 0.0
```

```
            for i in range(len(self.mapping)):
                type, pos = self.mapping[i]
                i_, j_ = pos
                if(type == 2):
                    _, _, duration = self.appliances[i_].slots[j_]
                    scheduled = False
                    for k in range(j-duration, j):
                        if(schedule[i][k]):
                            scheduled = True
                            break
                    if(scheduled):
                        non_interuptible_load[j] += self.appliances[i_].power[j_]*30.0*60.0
                        consumption_energy += self.appliances[i_].power[j_]*30.0*60.0
                else:
                    if(schedule[i][j] > 0.0):
                        if(type == 0):
                            unschedulable_load[j] += self.appliances[i_].power[j_]*30.0*60.0
                            consumption_energy += self.appliances[i_].power[j_]*30.0*60.0
                        else:
                            interuptible_load[j] += self.appliances[i_].power[j_]*30.0*60.0
                            consumption_energy += self.appliances[i_].power[j_]*30.0*60.0

            # Renewable energy from PV cell
            harvested_energy = self.pv_cell.get_renewable_energy(j)

            # Compute energy delta & make battery decision
            delta_energy = harvested_energy-consumption_energy
            grid_delta = self.battery.update(delta_energy)
            battery_delta.append(self.battery.SOC-prev_SOC)
            battery_SOC.append(self.battery.SOC)
            prev_SOC = self.battery.SOC

            # Buy or sell from grid based on grid delta
            if(grid_delta < 0):
                total_cost -= self.rtp[j]*grid_delta
            else:
                # total_cost -= (self.rtp[j]/2)*grid_delta
                temp = self.rtp[j]*0.5
                total_cost -= (temp)*grid_delta
            grid_deltas.append(-grid_delta)

        if(visualize):
```

```
            # Appliances load
            appliance_load = pd.DataFrame(
                {"unschedulable": unschedulable_load, "interuptible": interuptible_load, "non-inte
            appliance_load.plot(kind='bar', stacked=True)
            plt.xlabel('Time-slot')
            plt.ylabel('Power in KWh')
            plt.show()

            # Grid energy exchange
            plt.title("Grid exchange info")
            plt.bar(list(range(self.num_slots)), grid_deltas)
            plt.plot(list(range(self.num_slots)), grid_deltas, 'r')
            plt.axhline(0, 0, self.num_slots, color='black')
            plt.legend(["Grid exchange", "Zero"])
            plt.show()

            # Battery SOC
            plt.title("Battery usage info")
            plt.axhline(self.battery.SOC_min, 0, self.num_slots, color='c')
            plt.axhline(self.battery.SOC_max, 0, self.num_slots, color='c')
            plt.plot(list(range(self.num_slots)), battery_SOC, 'r')
            plt.bar(list(range(self.num_slots)), battery_delta)
            plt.axhline(0, 0, self.num_slots, color='black')
            plt.legend(["SOC_min", "SOC_max", "Battery SOC",
                        "Battery delta", "Zero"])
            plt.show()

        # Initialize battery for next simulation
        self.battery.SOC = self.SOC_init

        return total_cost

    def random_scheduler(self):
        schedule = [[0 for i in range(self.num_slots)]
                    for j in range(len(self.mapping))]
        for x in range(len(self.mapping)):
            type, pos = self.mapping[x]
            i, j = pos
            if(type == 0):
                app = self.appliances[i]
                start, end = app.slots[j]
                for k in range(start, end):
```

```
                schedule[x][k] = 1
            elif(type == 1):
                app = self.appliances[i]
                start, end, duration = app.slots[j]
                arr = list(range(start, end))
                random.shuffle(arr)
                for k in range(duration):
                    schedule[x][arr[k]] = 1
            else:
                app = self.appliances[i]
                start, end, duration = app.slots[j]
                k = random.randint(start, end-duration+1)
                schedule[x][k] = 1

        return schedule

    def genetic_scheduler(self, num_generations=100, num_siblings=100, selection_probability=0.1,
        next_gen = int(num_siblings*selection_probability)
        curr_generation = [self.random_scheduler() for _ in range(next_gen)]
        prev_generation = "NA"
        generation_costs = []
        for i_ in range(num_generations):
            prev_generation = curr_generation
            curr_generation = []

            # Mutate variants in prev generation
            curr_mutations = []
            for j_ in range(num_siblings):
                i = 0
                j = 0
                while(i == j):
                    i = random.randint(0, len(prev_generation)-1)
                    j = random.randint(0, len(prev_generation)-1)
                mutant = self.mutate(prev_generation[i], prev_generation[j])
                curr_mutations.append(mutant)
            curr_mutations += prev_generation

            # Evaluate mutations
            costs = []
            for mutant in curr_mutations:
                costs.append(self.simulate(mutant))
            generation_costs.append(sum(costs)/len(costs))
```

```
            # Natural selection
            idxs = list(np.argsort(np.array(costs))[:next_gen])
            for i in idxs:
                curr_generation.append(curr_mutations[i])

            if(visualize):
                print(
                    f"Generation-{i_+1}: Best mutation cost -> {self.simulate(curr_generation[0])}

        if(visualize):
            plt.title("Genetic Algo - Evolution")
            plt.xlabel("Generation no.")
            plt.ylabel("Total cost in Rs/day")
            plt.plot(list(range(len(generation_costs))), generation_costs)
            # plt.savefig("./Results/Genetic-learning")
            plt.show()
            plt.close()

        return curr_generation[0]

    def mutate(self, mutation1, mutation2):
        mutation = []
        for x in range(len(self.mapping)):
            type, pos = self.mapping[x]
            i, j = pos
            if(type == 0):
                mutation.append(mutation1[x])
            elif(type == 1):
                row = []
                row1 = mutation1[x]
                row2 = mutation2[x]
                idxs = []
                for i_ in range(self.num_slots):
                    if(row1[i_]+row2[i_] == 1):
                        idxs.append(i_)
                random.shuffle(idxs)
                idxs = idxs[:len(idxs)//2]
                for i_, (x1, x2) in enumerate(zip(row1, row2)):
                    if(x1+x2 > 0):
                        if(x1 > 0 and x2 > 0):
                            row.append(1)
```

```
                    else:
                        if i_ in idxs:
                            row.append(1)
                        else:
                            row.append(0)
                else:
                    row.append(0)

            p = random.random()
            if(p < 0.5):
                # Mutate
                app = self.appliances[i]
                start, end, duration = app.slots[j]
                numSwaps = 1
                for _ in range(numSwaps):
                    i_ = random.randint(start, end-1)
                    j_ = random.randint(start, end-1)
                    temp = row[i_]
                    row[i_] = row[j_]
                    row[j_] = temp
            mutation.append(row)
        else:
            sample = random.random()
            if(sample < 0.5):
                mutation.append(mutation1[x])
            else:
                mutation.append(mutation2[x])
    return mutation
```