ION EXCHANGE AND ADSORPTION
ISSN: 1001-5493

Vol. 23, Issue-02, 2023
pp.01-15

DESIGN AND ANALYSIS OF ELECTRICAL ENERGY USING IOT GATEWAY - PYTHON PROGRAMMING

Dr. Kamel Alikhan Siddiqui
(Assoc.Prof), Lords Institute of Engineering and Technology, IT Department, Hyderabad, India, kamel.ali.khan@lords.ac.in

Mrs. Hajira Sabuhi
(Astnt. Prof), Lords Institute of Engineering and Technology, IT Department, Hyderabad, India, hajirasabuhi@lords.ac.in

Mr. Mateen Ahmed
(Astnt. Prof), Lords Institute of Engineering and Technology, IT Department, Hyderabad, India, mateen.mohd@lords.ac.in

Mr.Zubair
(Astnt. Prof), Lords Institute of Engineering and Technology, IT Department, Hyderabad, India, juber@lords.ac.in

Abstract—
Smart homes offer a convenient and independent lifestyle in today's world. This research focuses on a wireless home automation system that enables fall detection, Bluetooth control switching, location tracking, and health monitoring. It aims to remotely control devices such as TVs, lights, and fans. The study highlights the need for monitoring systems to prevent excessive energy consumption and device malfunctions. To promote environmental sustainability and reduce carbon footprints, an optimization and monitoring detection system has been developed. The goal is to create a smart E-home by integrating automation and improving monitoring systems.
Keywords- IOT Gateway, Sennsors, BlueTooth, Arduino, Proteous, Keil and Python.

I. INTRODUCTION

As technology advances, the cost of electricity rises, prompting consumers to seek ways to manage and reduce their energy consumption. IoT offers a comprehensive solution by evaluating and optimizing energy usage across the entire household, not just at the device level. It allows for simple changes like turning off lights or adjusting device settings, as well as identifying issues caused by outdated equipment or broken appliances. In the business sector, IoT-enabled automated meter readers eliminate the need for physical visits and enable accurate billing. Overall, IoT streamlines energy management, cuts costs, and ensures system dependability by preventing overloads and monitoring consumption.

Smart E-Home

Smart E-homes utilize modern sensor technologies and intelligent systems to monitor energy consumption and provide self-learning capabilities. They consist of distributed sensors, motion detectors, and a wireless sensor network that collects real-time data on electricity usage[1,2,4].

By analyzing the data and learning from patterns, these systems can recognize regular and irregular activities and predict unexpected behavior. Smart E-homes offer convenience, energy savings, and enhanced security, and can be easily installed in existing homes without major modifications.

II. RUNTIME CONFIGURATIONS STEPS:

1. All systems are configured in DUAL BOOT mode i.e, Students can boot from Windows XP or Linux as per their lab requirement. This is very useful for user because they are familiar with different Operating Systems so that they can execute their programs in different programming environments.
2. 2. Each user has a separate login for database access
3. 3. Keil is a German based Software development company. It provides several development tools likeIDE (Integrated Development environment), Project Manager, Simulator, Debugger, C Cross Compiler, Cross Assembler, Locator/Linker[3]
4. 4. Keil Software provides user with software development tools for the 8051 family of microcontrollers. With these tools, user can generate embedded applications for the multitude of 8051 derivatives. Keil provides following tools for 8051 development C51 Optimizing C Cross Compiler, A51 Macro Assembler, 8051 Utilities (linker, object file converter, library manager), Source-Level Debugger/Simulator, µVision for Windows Integrated Development Environment.
5. 5. Systems are assigned numbers and same system is allotted for users for particular operation.
6.

Description about ES used:

Description about ES Concepts: Embedded systems are designed to do some specific task, rather than be a general-purpose computer for multiple tasks. Some also have real-time performance constraints that must be met, for reason such as safety and usability; others may have low or no performance requirements, allowing the system hardware to be simplified to reduce costs. Embedded systems are not always separate devices. Most often they are physically built-in to the devices they control.

The software written for embedded systems is often called firmware, and is stored in read-only memory or Flash memory chips rather than a disk drive. It often runs with limited computer hardware resources: small or no keyboard, screen, and little memory. Embedded systems range from no user interface at all — dedicated only to one task to full user interfaces similar to desktop operating systems in devices such as PDAs.

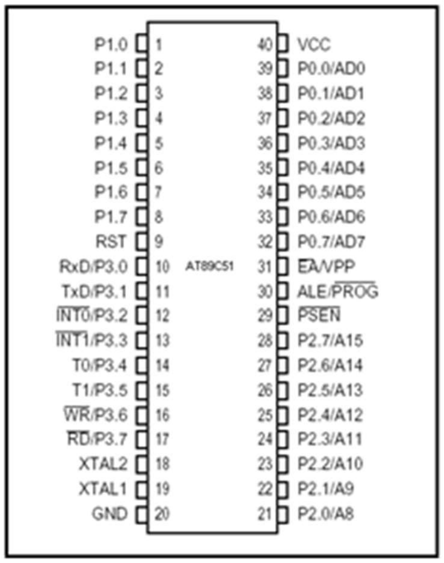
Simple embedded devices use buttons, LEDs, and small character- or digit-only displays, often with a simple menu system.

III. GENERAL DESCRIPTION 8051 AND AT89C51

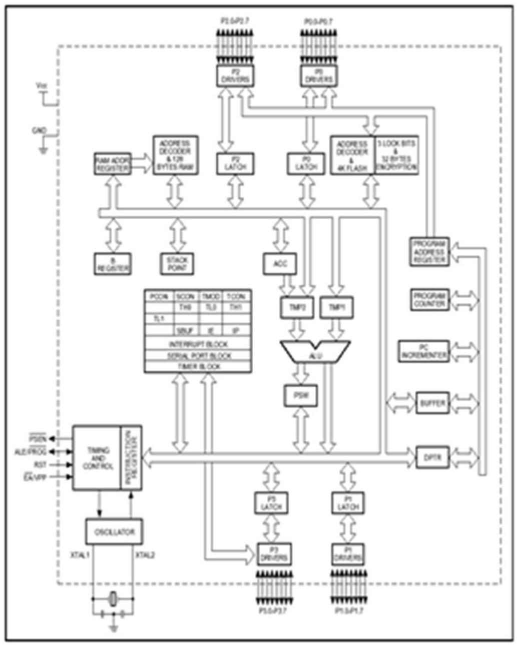
THE MICROCONTROLLER: A microcontroller is a general purpose device, but that is meant to read data, perform limited calculations on that data and control its environment based on those calculations. The prime use of a microcontroller is to control the operation of a machine using a fixed program that is stored in ROM and that does not change over the lifetime of the system [2].

The microcontroller design uses a much more limited set of single and double byte instructions that are used to move data and code from internal memory to the ALU. The microcontroller is

concerned with getting data from and to its own pins; the architecture and instruction set are optimized to handle data in bit and byte size.
The AT89C51 is a low-power, high-performance CMOS 8-bit microcontroller with 4k bytes of Flash Programmable and erasable read only memory (EROM). The device is manufactured using Atmel's high-density nonvolatile memory technology and is functionally compatible with the industry-standard 80C51 microcontroller instruction set and pin out. By combining versatile 8-bit CPU with Flash on a monolithic chip, the Atmel's AT89c51 is a powerful microcomputer, which provides a high flexible and cost- effective solution to many embedded control applications.
Pin configuration of AT89c51 Microcontroller [2]



AT89C51 Block Diagram [2]



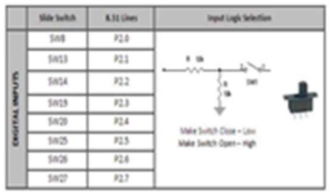
8051 PIN Description [2]:

IV. BUILD UP FOR NECESSARY PROGRAMS

1. PROGRAM TO DEMONSTRATE SWITCH & LED

A switch is an electrical component that can break an electrical circuit, interrupting the current or diverting it from one conductor to another. A switch may be directly manipulated by a human as a control signal to a system, or to control power flow in a circuit.

Pin Assignment with 8051



	Slide Switch	8051 Lines	Input Logic Selection
DIGITAL INPUTS	SW8	P2.0	Make Switch Close = Low Make Switch Open = High
	SW13	P2.1	
	SW14	P2.2	
	SW19	P2.3	
	SW20	P2.4	
	SW25	P2.5	
	SW26	P2.6	
	SW27	P2.7	

We now want to control the LED by using switches in 8051 Development board. It works by turning ON a LED & then turning it OFF when switch is going to LOW or HIGH.

The 8051 Development board has eight numbers of point LEDs, connected with I/O Port lines (P0.0 – P0.7) to make port pins high. Eight switches, connected with I/O port lines (P2.0 – P2.7) are used to control eight LEDs.

ALGORITHM:

1. Assign LED address for Microcontroller kit
2. Set the first switch press indicates led to be “ON
3. Delay for certain time.
4. Set switch press indicates led to be “OFF”
5. Delay for certain time

Code:

```
#include<reg51.h>
sbit led=P2^1;
sbit sw=P2^2;
main()
{
 P2=0x00;
 if(sw==0)
 led=0;
 else
 led=1;
 }
```

SCHEMATIC CIRCUIT DIAGRAM

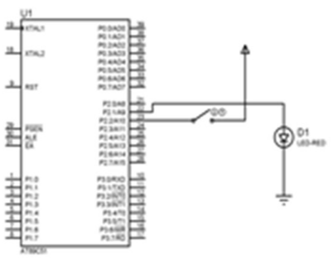
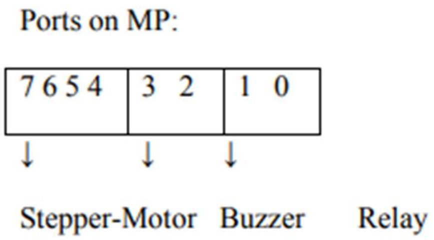


Fig: LED

2. PROGRAM TO DEMONSTRATE RELAY:

We demonstrate glowing of 2 relays one after the other . A relay is a device that responds to a small current or voltage change by activating switches or other devices in an electric circuit. Used for alarming system



Relay 1: Prev=Prev | (1<< 0) (for glowing) , Prev=Prev & ~(1<< 0) (for clearing)
Relay2 : Prev=Prev | (1<< 1) (for glowing) , Prev=Prev & ~(1<< 1) (for clearing)

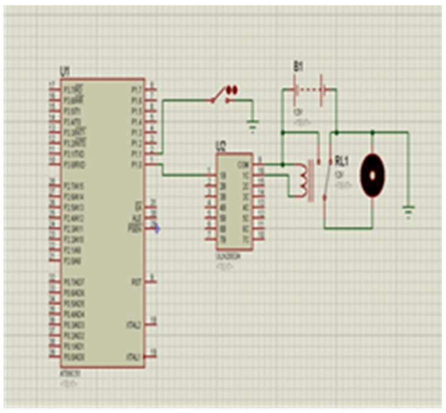
Algorithm

1. Assign LED address for Microcontroller kit
2. While true
3. Set the first relay to 1 which indicates that it is “ON”
4. Delay for certain time.
5. Set the relay to 0 which is “OFF”
6. Delay for certain time.
7. Set the realy2 to „1‟
8. Delay for certain time.
9. Set the relay2 to „0‟
10. Delay for certain time.

CODE:

```
#include<reg51.h>
sbit sw =P1^1;
sbit relay =P1^0;
void main()
{
sw=1;
relay=0;
while(1)
{
if(sw==0)
{
relay=1;
}
else
{
relay=0;
}
 }
}
```

SCHEMATIC CIRCUIT DIAGRAM:



3. Program to Glow the Alternate LEDs:

Light Emitting Diodes (LEDs) are simple and most commonly used electronic components to display the digital signal states. The LED emits light when current is passed through it. It could blow up if we pass more current, hence we put a current limiting resistor.

ALGORITHM:

step 1: assign led to port 2

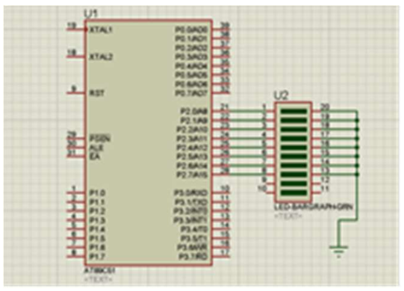
step 2: Initialize with LED address. For LED‟s to glow from Right to Left (R → L) one after the the other set the counter c=0 and increment the counter by one till the value reaches c=7.

step 3: . Delay for certain time

CODE:

```
#include<reg51.h>
#define port P2
main()
{
unsigned int i;
P2=0x00;
while(1)
{
port=0x55;
for(i=0;i<500;i++);
port=0xaa;
}
}
```

SCHEMATIC CIRCUIT DIAGRAM:



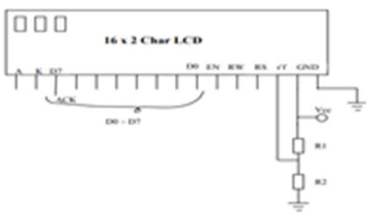
4. Program to Demonstrate LCDs using 8051:

The Liquid Crystal Display (LCD) is a low power device (microwatts). Now a days in most applications LCDs are using rather using of LED displays because of its specifications like low power consumption, ability to display numbers and special characters which are difficult to display with other displaying circuits and easy to program. An LCD requires an external or internal light source. Temperature range of LCD is 0ºC to 60ºC and lifetime is an area of concern, because LCDs can chemically degrade these are manufactured with liquid crystal material (normally organic for LCDs) that will flow like a liquid but whose molecular structure has some properties normally associated with solids.
RS (Command / Data): This bit is to specify weather received byte is command or data. So that LCD can recognize the operation to be performed based on the bit status.
RS = 0 => Command RS = 1 => Data RW (Read / Write): RW bit is to specify whether controller wants READ from LCD or WRITE to LCD.

The READ operation here is just ACK bit to know whether LCD is free or not. RW = 0 => Write RW = 1 => Read

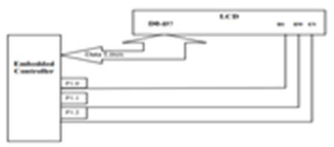
EN (Enable LCD): EN bit is to ENABLE or DISABLE the LCD. When ever controller wants to write some thing into LCD or READ acknowledgment from LCD it needs to enable the LCD.
EN = 0 => High Impedance EN = 1 => Low Impedance ACK (LCD Ready): ACK bit is to acknowledge the MCU that LCD is free so that it can send new command or data to be stored in its internal Ram locations ACK= 1 => Not ACK ACK = 0 => ACK

LCD diagram:



Block Diagram

Hardware connections:



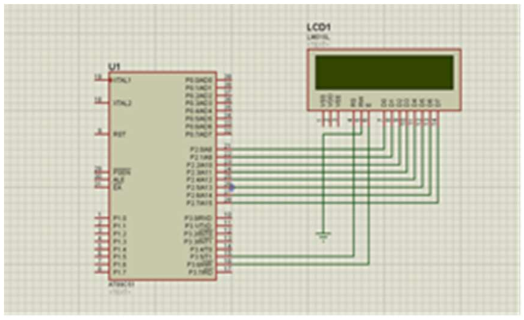
Algorithm
1. Initialize Set of Commands.
2. Initialize LCD with proper set of Commands.
3. Write each Command to LCD Command Write Address.
4. Clear LCD for any previous data.
5. Write Data to LCD Data Write Address.

CODE:

```
#include<reg51.h>
#define lcd P2
sbit rs=P3^5;
sbit rw=P3^6;
sbit en=P3^7;
lcd_init();
void lcd_cmd(unsigned char );
void lcd_data(unsigned char );
void lcd_display(unsigned char *ptr);
void lcd_delay(unsigned int t);
main()
{
lcd_init();
lcd_cmd(0x83);
lcd_display(" WELCOME TO  ");
lcd_cmd(0xc4);
lcd_display(" LORDS IT DEPT  ");
}
lcd_init()
{
lcd_cmd(0x01);
lcd_cmd(0x06);
lcd_cmd(0x38);
lcd_cmd(0x0c);
}
void lcd_cmd(unsigned char x)
{
lcd=x;
rs=0;
rw=0;
en=1;
lcd_delay(20);
en=0;
}
void lcd_data(unsigned char x)
{
lcd=x;
rs=1;
rw=0;
en=1;
lcd_delay(20);
en=0;
}
void lcd_display(unsigned char *ptr)
```

```
{
while(*ptr++)
lcd_data(*ptr);
}
void lcd_delay(unsigned int t)
{
unsigned int i,j;
for(i=0;i<t;i++)
for(j=0;j<1275;j++);
}
```

SCHEMATIC CIRCUIT DIAGRAM:



V. IOT GATEWAYS AND MICROCONTROLLER:

A gateway for IoT (Internet of Things) is a device or software that acts as a bridge between IoT devices and the internet. It connects IoT devices to a network and allows them to communicate with each other and with other devices or systems. A gateway can also provide security, data processing, and other functions for IoT devices. It can be a separate device or integrated into a larger system such as a router or a smart home hub. An IoT gateway can provide a variety of functionalities, including:

1. Communication: Enabling IoT devices to communicate with each other and with other devices or systems on the internet.
2. Security: Providing security features such as encryption and authentication to protect the data and devices from unauthorized access.
3. Data Processing: Collecting, filtering, and processing data from IoT devices before it is sent to the cloud or a local server for storage or analysis.
4. Networking: Managing the network connections for IoT devices and ensuring that they are properly configured and connected to the internet.
5. Device Management: Providing tools for monitoring, configuring, and updating IoT devices remotely.
6. Protocol Translation: Translating between different communication protocols used by IoT devices, allowing them to communicate with each other and with other systems.
7. Local Processing: Some gateways can perform certain computations locally, reducing the amount of data that needs to be sent to the cloud, which can be beneficial for devices that have limited connectivity or limited power sources.

8. Edge computing: Some gateway devices are equipped with significant computational power, storage, and memory which allows them to run complex tasks and store data locally instead of sending it to the cloud.
9. Cloud Integration: Integrating the gateway with cloud platforms such as AWS, Azure, and Google Cloud, enabling the devices to store and process data in the cloud.

IoT (Internet of Things) and microcontrollers are often used together in the design of IoT devices. A microcontroller is a small, integrated computer that contains a processor, memory, and input/output peripherals on a single chip. It is designed to control a specific function or system, such as a remote control, a thermostat, or a washing machine. IoT devices are connected to the internet and are able to communicate with other devices and systems, allowing them to share data and be controlled remotely. Microcontrollers are a key component of IoT devices as they allow the devices to perform the necessary functions and communicate with other systems. For example, a microcontroller in an IoT enabled smart thermostat can collect data from temperature and humidity sensors, process the data, and use it to control the heating and cooling system in the home. The microcontroller can also communicate with other devices such as a smartphone app or a smart home hub, allowing the user to control the thermostat remotely. In summary, microcontrollers are a crucial building block for IoT devices, as they provide the necessary processing and communication capabilities for the devices to function and interact with other systems.

VI. OVERVIEW OF THE AUTOMATION:

The project describe the basic amenities to deploy the Communication with the sensors for measuring and optimizing Automation in terms of parameters of Energy. The steps to start with is as follows, The use IoT to create a home automation system using Python, we will need to follow these general steps:

1. Set up your hardware: This will include selecting and connecting the various sensors and actuators (e.g. lights, temperature sensors, etc.) that you will use to control and monitor your home.
2. Install software and libraries: You will need to install the necessary software, such as a Python interpreter, and libraries such as PySerial, that will allow you to communicate with your hardware.
3. Write the code: Use Python to write the code that will control the system. This will include reading data from sensors, processing that data to make decisions about how to control the actuators, and sending commands to the actuators to make changes.
4. Connect to the internet: Connect your hardware to the internet by using an ethernet shield or wifi shield, this way you can access the system remotely and receive notifications.
5. Create a web interface: Create a web interface that allows you to monitor and control the system remotely. You can use Python libraries like Flask or Django to create the web interface.
6. Test the system: Test the system to ensure that everything is working properly, and make any necessary adjustments to the code.
7. Deploy the system: Once you've tested the system and are satisfied that it's working properly, deploy it in your home and enjoy the convenience of home automation!

It's worth mentioning that this is a high-level overview of the process, and there may be additional steps depending on the specific requirements of the E-home automation system.

VII. PROPOSED SMART IOT HOME APPLICATION

As seen in Figure 1, a system of sensors and detectors is utilized to collect data, regulate condition, and take action, much like conventional smart home systems. The monitoring software, which is operating on the Arduino board seen in Figure 1's center, receives the data collected by the sensors. Users of the system can decide whether to trigger an alert or alarm by comparing the values in the collected data with the values that have already been defined. In this study, a prototype smart E-house was created in order to conduct a number of tests. The system was created and tested before being installed.

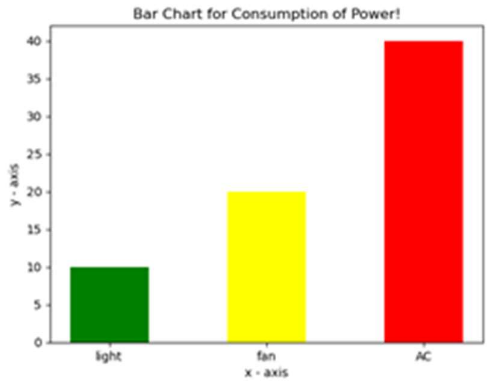
Python code and Visualization of Energy Consumption:

```
from tkinter import *
from tkinter import ttk
from tkinter import messagebox
# Call TK interface
import tkinter
#importing bargraph library matplotlib.pyplot to use functionalities of the bar/chart graphs in program/application.
import matplotlib.pyplot as plt

# Call message box
import tkinter.messagebox
root = Tk()
root.geometry("700x780")
# create a main frame
main_frame= Frame(root)
main_frame.pack(fill=BOTH, expand=1)
# Create a Canvas
my_canvas=Canvas(main_frame, bg="white")
my_canvas.pack(side=LEFT, fill=BOTH, expand=1)
# Add a Scrollbar to the Canvas
my_scrollbar=ttk.Scrollbar(main_frame,orient=VERTICAL, command=my_canvas.yview)
my_scrollbar.pack(side=RIGHT, fill=Y)

# configure the Canvas
my_canvas.configure(yscrollcommand=my_scrollbar.set)
my_canvas.bind('<Configure>',lambda e:my_canvas.configure(scrollregion = my_canvas.bbox("all")))

# Create Another Frame inside the Canvas
second_frame=Frame(my_canvas, bg="white")
third_frame=Frame(my_canvas, bg="white")
```

```
# Add that new frame to a Window in the Canvas
my_canvas.create_window((0,0), window=second_frame, anchor="nw")
my_canvas.create_window((400,60), window=third_frame, anchor="nw")
def bhm():
  d=bh.get()
def brm():
  ee=br.get()
def hrm():
  ee=hr.get()

#Button Check Status/Monitor
def clicked(value):
  #myLabel = Label(second_frame, text= value)
  #myLabel.pack()
  z =(value)
  #print (z)
  if z<=9:
    print ("You appliance are in Off State")
    messagebox.showwarning("You appliance are in Off State/Sleep mode")
  elif bh.get()==20 or br.get()==20 or hr.get()==20:
    messagebox.showinfo("Your Appliances: Attention Required Kindly turnoff the switches to
maintain optimum !")
  elif z>=10 and z<80 :
    print ("Attecntion required")
    messagebox.showwarning("Your Home is Ready,Take Rest")
  else:
    print( "Your Appliances: Attention Required Kindly turnoff the switches to maintain
optimum  !")
    messagebox.showwarning("Your Appliances: Attention Required Kindly turnoff the
switches to maintain optimum ")
# command button ended

#def popup():
 # messagebox.showinfo("Cats Health","Your Cat is healthy")

root.title ('Electricity Monitoring')
#bg="white"
Label(second_frame, text="E-Monitoring System ", width="40", bg="Brown", fg="White",
font=("ariel", 20, "bold","italic")).pack(pady=10, anchor=NW)

#Radiobutton(second_frame, text= 'Very Less', variable=ap, value=20,
command=apm,bg="white").pack(padx=80, anchor=W)
#Appliance Monitoring starts here
#Light-1
```

```
bh= IntVar()
Label(second_frame,text="1. Light State ?", font=("ariel", 14),bg="white").pack(padx=40,pady=10, anchor=W)
Radiobutton(second_frame, text= 'On', variable=bh, value=15,command=bhm,bg="white").pack(padx=80, anchor=W)
Radiobutton(second_frame, text= 'Off', variable=bh, value=0, command=bhm,bg="white").pack(padx=80, anchor=W)
Radiobutton(second_frame, text= 'Select if light is on state more than 12 hours', variable=bh, value=20, command=bhm,bg="white").pack(padx=80, anchor=W)
#Fan-2
br= IntVar()
Label(second_frame,text="2. Fan State ?", font=("ariel", 14),bg="white").pack(padx=40,pady=10, anchor=W)
Radiobutton(second_frame, text= 'On', variable=br, value=15,command=brm,bg="white").pack(padx=80, anchor=W)
Radiobutton(second_frame, text= 'Off', variable=br, value=0, command=brm,bg="white").pack(padx=80, anchor=W)
Radiobutton(second_frame, text= 'Select if Fan on state more than 12 hours', variable=br, value=20, command=brm,bg="white").pack(padx=80, anchor=W)
#AC-3
hr= IntVar()
Label(third_frame,text="3. AC State ?", font=("ariel", 14),bg="white").pack(padx=40, anchor=W)
Radiobutton(third_frame, text= 'On', variable=hr, value=15,command=hrm,bg="white").pack(padx=80, anchor=W)
Radiobutton(third_frame, text= 'Off', variable=hr, value=0, command=hrm,bg="white").pack(padx=80, anchor=W)
Radiobutton(third_frame, text= 'Select if Ac on state more than 12 hours', variable=hr, value=20, command=hrm,bg="white").pack(padx=80, anchor=W)

#myLabel = Label(second_frame, text=bt.get())
#myLabel.pack()
myButton = Button(second_frame, text="Check Status!",
            command=lambda:clicked((bh.get()+br.get()+ hr.get())),
            font=("helvittica", 16, "bold"),bg="Green", fg= "white")
myButton.pack(pady=10)

# x-coordinates of left sides of bars
left = [1, 2, 3]
# heights of bars
height = [10,20,40]
 # labels for bars
tick_label = ['light', 'fan', 'AC']
# plotting a bar chart
```

```
plt.bar(left, height, tick_label = tick_label,
        width = 0.5, color = ['green','yellow','red'])

# naming the x-axis
plt.xlabel('x - axis')
# naming the y-axis
plt.ylabel('y - axis')

# plot title
plt.title('Bar Chart for Consumption of Power!')

# function to show the plot
plt.show()
#Quit button process
qButton = Button(second_frame, text="Quit", width="10",command=root.destroy,
                 font=("helvittica", 16, "bold"),bg="Brown", fg= "white")
qButton.pack(anchor=E)
#Button(second_frame, text="Health", command=popup).pack()
root.mainloop
```



REFERENCES

1. Conference on Advanced Information Networking and Applications Workshops (AINAW'07).

2. Ayala & Gadre (2017): "The 8051 Microcontroller & Embedded Systems using Assembly and C", CENGAG.

3. Dr.Kamel Alikhan Siddiqui, The Ciência & Engenharia -Science & Engineering Journal ISSN: 0103-944XVolume 11 Issue 1, 2023pp: 1337-13451337 https://seer-ufu-br.online Progressive Analysis and Predictions of Leukemia (Cancer) Patients on a Machine Learning Model-The APPOLLO Hospitals Accredited.

4. Dr.Kamel Alikhan Siddiqui, SCOPUS INDEXED Specialusis Ugdymas (SU), Machine Learning Model Development based on Brazil's Covid-19 Data Set, ISSN No: 1392 5369, International journal, Sept/ 2022, Vol2 no:43(2022), 403 412: 10 pages, https://www.sumc.lt/index.php/se/article/view/1162

5. Moutacalli, M. T., Marmen, V., Bouzouane, A. and Bouchard, B., (2013). "Activity Pattern Mining using Temporal Relationships in a Smart Home", IEEE Symposium on Computational Intelligence in Healthcare and e-health (CICARE), 83-87.
6. "The 8051 Microcontroller and Embedded Systems", Mazidi, Mazidi, Pearson(2019).
7. Das, R., Tuna, G., (2015). "Machine-to-Machine Communications for Smart Homes", International Journal of Computer Networks and Applications (IJCNA), EverScience Publications, 2 (4), 196-202.
8. Activity Recognition System for Smart Home Services", In Proceedings of 11th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS), 74-80.
9. Son, Y. -S., Jo, J., Park, J. -H. and Pulkkinen, T., (2013). "Diabetic Patient Care using Home User Activity Recognition", ICTC 2013, 191- 196.
10. Alemdar, H. and Ersoy, C., (2010). "Wireless sensor networks for healthcare: A survey, Computer Networks", 54(10), 2688-2710. DOI: 10.1016/j.comnet.2010.05.003
11. Istepanian, R. S. H., Jovanov, E. and Zhang, Y. T., (2004). Guest Editorial Introduction to the Special Section on M-Health: Beyond Seamless Mobility and Global Wireless Health-Care Connectivity. IEEE Transactions on Information Technology in Biomedicine, 8(4), 405-414. DOI: 10.1109/TITB.2004.840019
12. Advances in Intelligent Computational Systems (RAICS), elderly people: A realistic approach", 2011 IEEE Recent 1 Suryadevara, N. K. and Mukhopadhyay, S. C., (2011). "Wireless sensors network based safe home to care -5