IIITDM
Kanceepuram

(Object Oriented Programming)

Smart Home Automation System

Nithin Ch
Roll no: CS23B1102

Sanshrey G
Roll no: CS23B2014

Varshith B
Roll no: CS23B2015

Raviteja B
Roll no: CS23B2011

Susan B
Roll no: CS23B2026

November 11, 2024

Smart Home Automation System

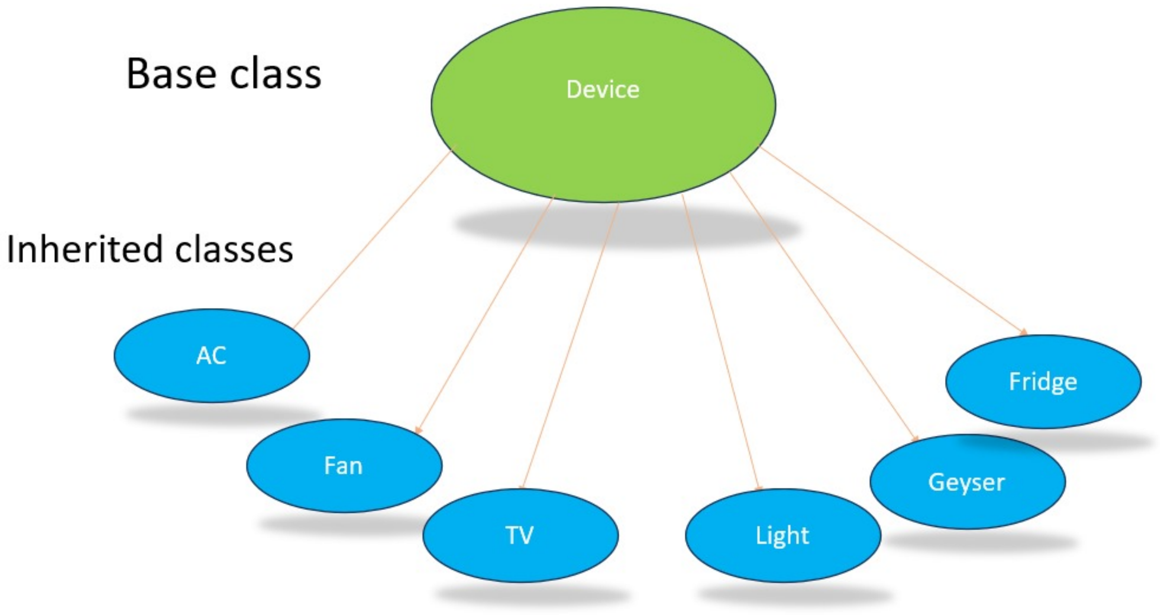
1. Introduction

This project, **Smart Home Device Control System**, provides an interactive console application that enables users to control and manage several household devices through a unified interface. By simulating various controls for each device, the project demonstrates object-oriented programming principles and real-world application of C++ classes.

The main objective is to implement a device control system where each device type (AC, Fan, Fridge, Light, Geyser, TV) has unique properties and methods, allowing users to adjust settings like temperature, speed, brightness, etc., and set timers to automate device behavior.

2. System overview

This centralizes control of all household appliances. Each device (e.g., fan, AC, light) is represented as an object, allowing users to monitor and manage these devices through a unified mobile interface. The system is designed to enable easy customization and scalability for various appliance types.



3. Class Structure and Design

The project adopts a hierarchical design with a base class **Device** and derived classes for each specific device type.

- **Device Class:** Serves as the foundation for all devices, containing common attributes such as status (on/off) and basic methods to turn the device on or off, check status, and perform other universal functions.
- **Derived Classes:** Each device type (AC, Fan, etc.) is represented by a derived class, with specialized methods to handle settings specific to each device. For example, the AC class includes methods for temperature control, swing setting, and fan speed adjustment.

This class hierarchy leverages polymorphism, enabling efficient management of multiple device types and enhancing code reusability.

4. Data encapsulation

Data encapsulation is implemented in the system to ensure that data is protected and only accessible through designated methods. This helps in maintaining data integrity and security by hiding internal implementation details from the user.which allows to monitor the device setting only by the class members. these system achieves encapsulation through following points

- **Device:** status is encapsulated as protected data member in device class which can be accessed in all other derived classes to operate their functionality of on and off.
- **TV:** channel volume, brightness are encapsulated to access the TV's present status
- **AC:** temperature, swing, fans are encapsulated to access the AC's present status
- **Fan:** speed are encapsulated to access the Fan's present status
- **Fridge:** cooling level, inner light status are encapsulated to access the Fridge's present status
- **Geyser:** temperature are encapsulated to access the geyser's present status
- **Light:** brightness, color shade are encapsulated to access the Light's present status

5. Operator overloading

The `<<` operator (via `operator<<`) overloaded `COUT` for easy display of device information. This operator allows to print details like the status, power level, or settings of a device directly to the console in a readable format. By overloading `<<`, streamline how device data is presented, making it straightforward to monitor devices without needing additional function calls. This approach enhances code readability and provides quick, clear insights into each device's current state.

6. Login and Account Setup for Smart Home System

Login System:

- Users are prompted to enter their unique ID.
- The system checks if the ID exists in the stored credentials.
- If the ID is found, the user is granted access to the system.
- If the ID is not found, the user is directed to create a new account.

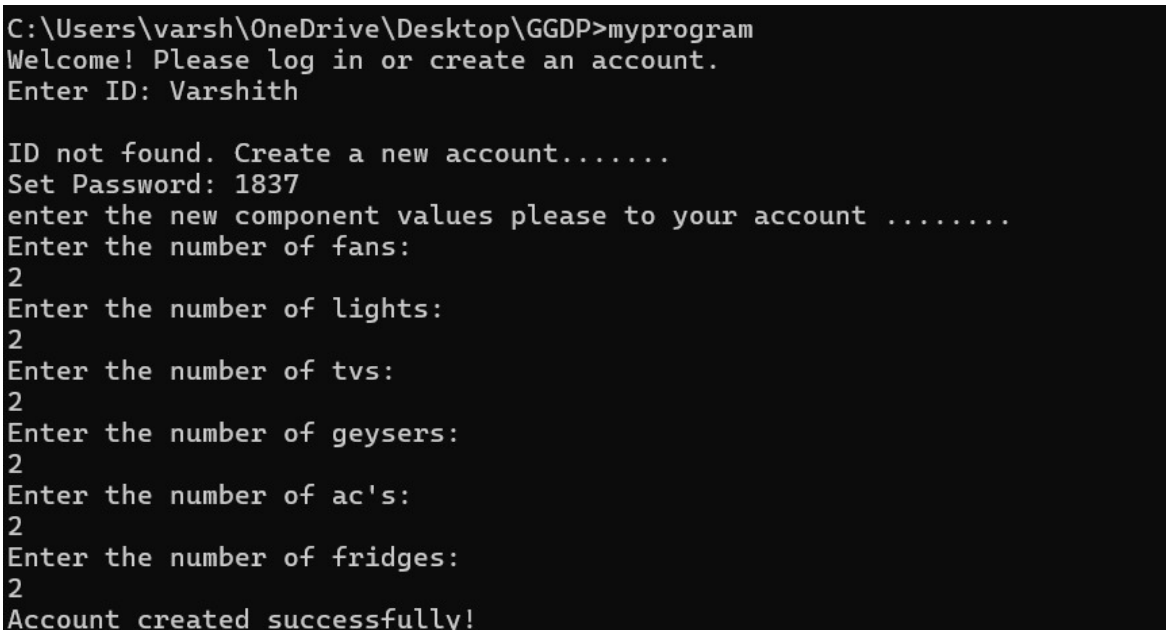
Account Creation:

- New users can set up their account by providing a unique ID and password.
- The entered ID and password are stored in a CSV file for future authentication.

Component Setup for Smart Home:

Upon account creation, users are asked to specify the number of smart home components they own, such as:

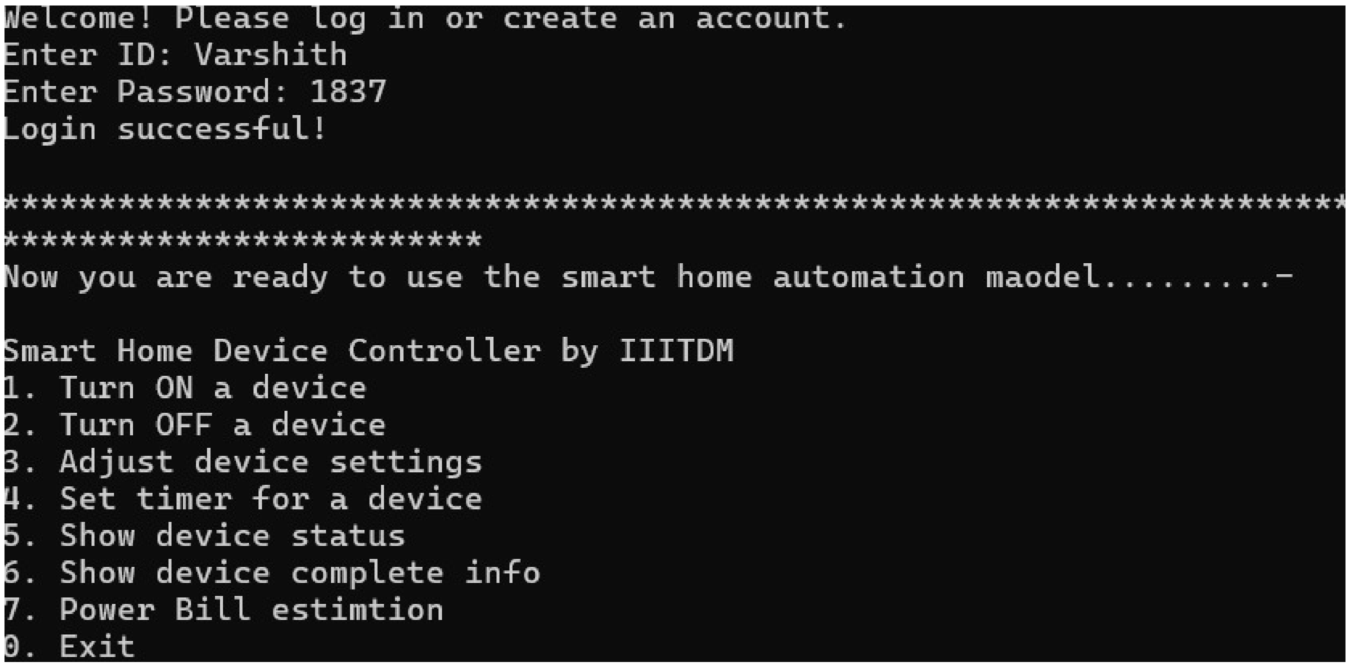
- Fans
- Lights
- TVs
- Geysers
- AC's
- Fridges



```
C:\Users\varsh\OneDrive\Desktop\GGDP>myprogram
Welcome! Please log in or create an account.
Enter ID: Varshith

ID not found. Create a new account.......
Set Password: 1837
enter the new component values please to your account ........
Enter the number of fans:
2
Enter the number of lights:
2
Enter the number of tvs:
2
Enter the number of geysers:
2
Enter the number of ac's:
2
Enter the number of fridges:
2
Account created successfully!
```

7. User-Specific File Storage for Component State

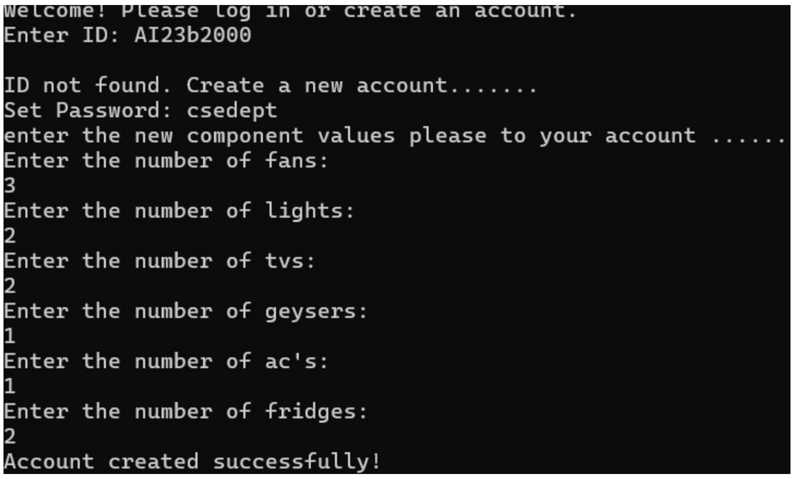
- Each user has a dedicated file that stores their smart home component data, ensuring their configuration and preferences are securely saved and uniquely identified. This allows the system to provide a personalized experience for every user, maintaining consistency between sessions.
- When a user logs in, the system accesses their file to load the previous state of all components, including their ON/OFF conditions. This eliminates the need for manual reconfiguration and ensures the user can seamlessly continue where they left off.components from their file.



```
Welcome! Please log in or create an account.
Enter ID: Varshith
Enter Password: 1837
Login successful!

**********************************************************************
*************************
Now you are ready to use the smart home automation maodel.........-

Smart Home Device Controller by IIITDM
1. Turn ON a device
2. Turn OFF a device
3. Adjust device settings
4. Set timer for a device
5. Show device status
6. Show device complete info
7. Power Bill estimtion
0. Exit
```

If account already exists, then previous state is loaded

- For new users, the system automatically creates a new file when they set up their account. This file stores the number of devices and their initial configurations, laying the groundwork for managing their smart home environment.
- The system updates each user's file in real-time whenever there are changes to their component states. This ensures that the most recent state of their smart home setup is always saved, providing a reliable and dynamic user experience.

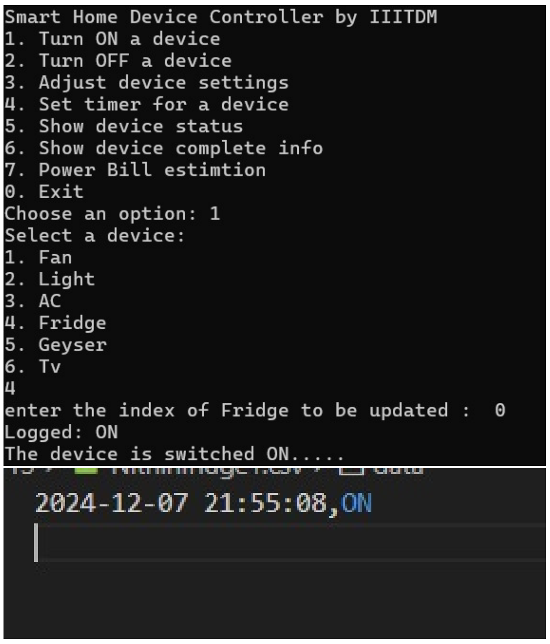


```
Welcome! Please log in or create an account.
Enter ID: AI23b2000

ID not found. Create a new account.......
Set Password: csedept
enter the new component values please to your account ......
Enter the number of fans:
3
Enter the number of lights:
2
Enter the number of tvs:
2
Enter the number of geysers:
1
Enter the number of ac's:
1
Enter the number of fridges:
2
Account created successfully!
```

For new user, data need to be extracted

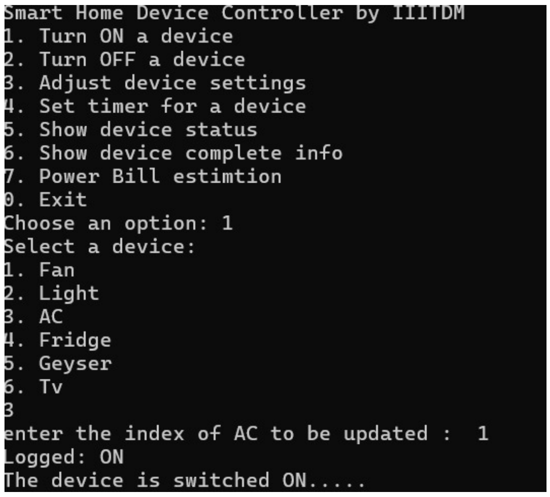
8. Device Control and Power Consumption Tracking

- **Login and Device Activation:** When a user logs in, they are presented with an option to turn specific smart home devices ON. This involves selecting the devices they wish to activate, such as fans, lights, TVs, geysers, AC's, and fridges



```
Smart Home Device Controller by IIITDM
1. Turn ON a device
2. Turn OFF a device
3. Adjust device settings
4. Set timer for a device
5. Show device status
6. Show device complete info
7. Power Bill estimtion
0. Exit
Choose an option: 1
Select a device:
1. Fan
2. Light
3. AC
4. Fridge
5. Geyser
6. Tv
4
enter the index of Fridge to be updated :  0
Logged: ON
The device is switched ON.....
```

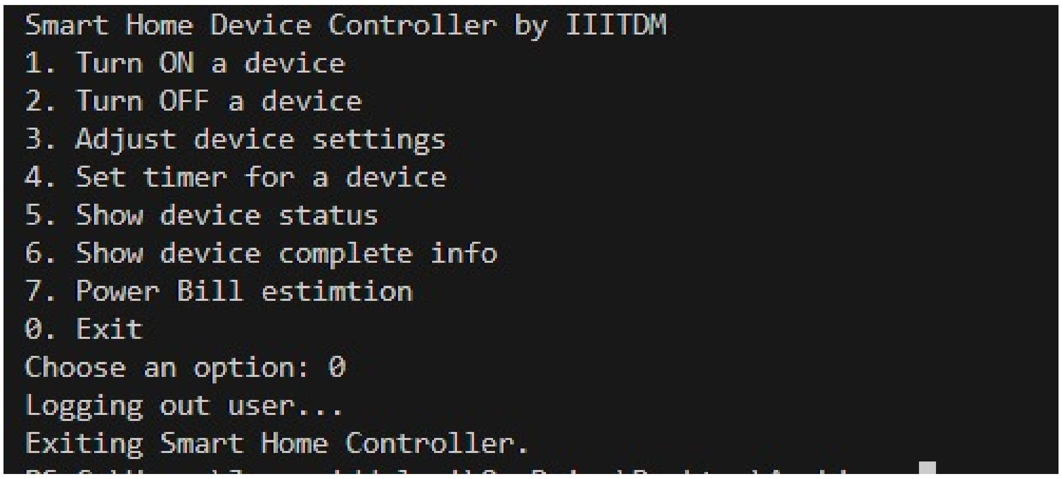
```
2024-12-07 21:55:08,ON
```



```
Smart Home Device Controller by IIITDM
1. Turn ON a device
2. Turn OFF a device
3. Adjust device settings
4. Set timer for a device
5. Show device status
6. Show device complete info
7. Power Bill estimtion
0. Exit
Choose an option: 1
Select a device:
1. Fan
2. Light
3. AC
4. Fridge
5. Geyser
6. Tv
3
enter the index of AC to be updated :  1
Logged: ON
The device is switched ON.....
```

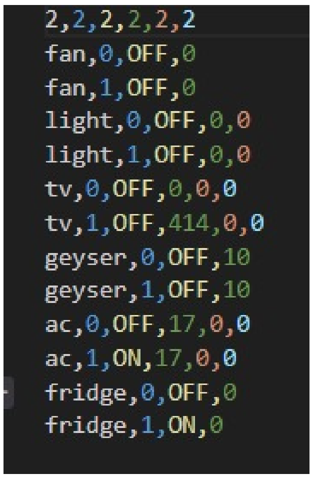
The device's on and off timings are recorded in a CSV file, enabling users to monitor power usage, calculate electricity bills, enforce parental controls, and track the computer's operational status for better management.

- **Logout Functionality:** After configuring the devices, the user can log out. During logout, the system ensures that the state of all devices (ON/OFF) is saved securely in the user's file. This state acts as a reference for the next session and ensures continuity in operations.



```
Smart Home Device Controller by IIITDM
1. Turn ON a device
2. Turn OFF a device
3. Adjust device settings
4. Set timer for a device
5. Show device status
6. Show device complete info
7. Power Bill estimtion
0. Exit
Choose an option: 0
Logging out user...
Exiting Smart Home Controller.
```

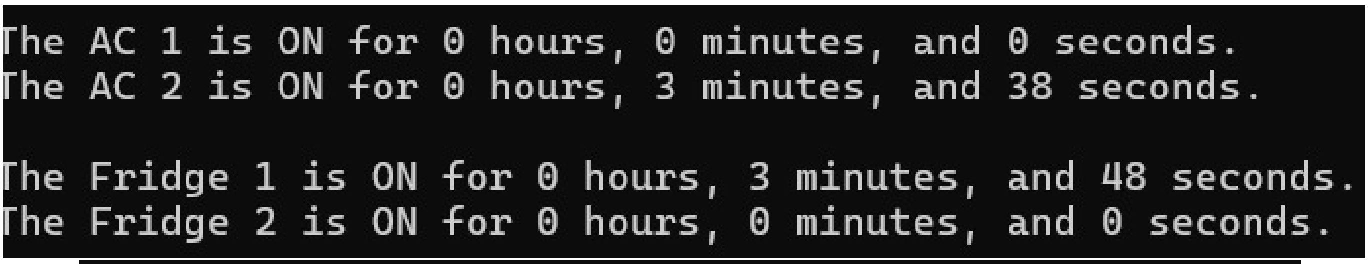
- **Re-login and Device Deactivation:** After configuring the devices, the user can log out. During logout, the system ensures that the state of all devices (ON/OFF) is saved securely in the user's file. This state acts as a reference for the next session and ensures continuity in operations.



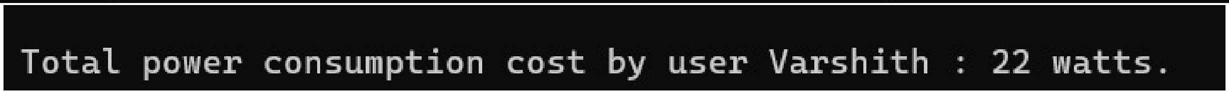
```
2,2,2,2,2,2
fan,0,OFF,0
fan,1,OFF,0
light,0,OFF,0,0
light,1,OFF,0,0
tv,0,OFF,0,0,0
tv,1,OFF,414,0,0
geyser,0,OFF,10
geyser,1,OFF,10
ac,0,OFF,17,0,0
ac,1,ON,17,0,0
fridge,0,OFF,0
fridge,1,ON,0
```

Each time a user logs out or exits the smart home controller, their current data is saved. When they log back in, the stored data is reloaded, ensuring a seamless and consistent experience.

- **Power Consumption Calculation:** Upon logging in again, the user can view the previously active devices and choose to turn them OFF. The system records the OFF time for each device, along with the duration for which it was active. This allows the system to calculate the energy usage for each device accurately.



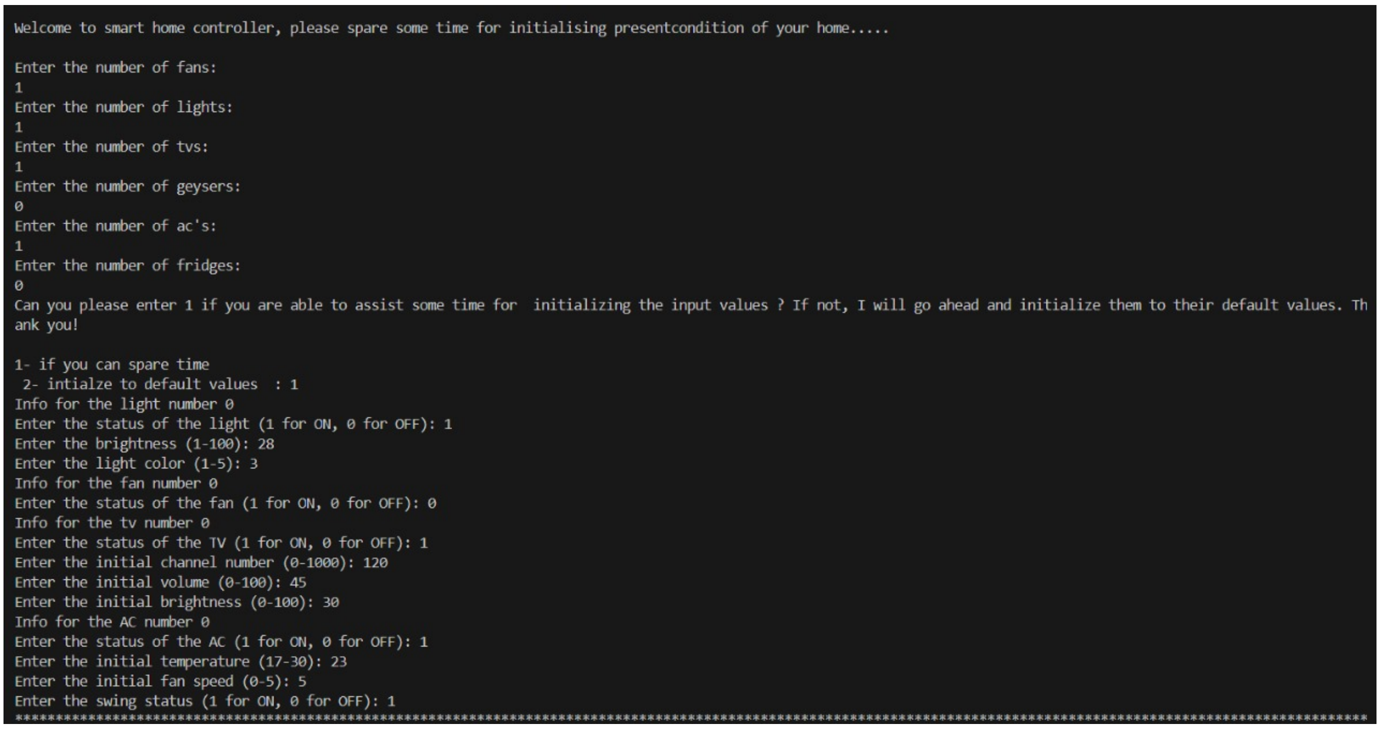
```
The AC 1 is ON for 0 hours, 0 minutes, and 0 seconds.
The AC 2 is ON for 0 hours, 3 minutes, and 38 seconds.

The Fridge 1 is ON for 0 hours, 3 minutes, and 48 seconds.
The Fridge 2 is ON for 0 hours, 0 minutes, and 0 seconds.
```



```
Total power consumption cost by user Varshith : 22 watts.
```

9. configuring initial home setup

the user specifies the initial configuration of their smart home, including the number and type of devices in each room. By entering these details, the system tailors the automation experience to match the user's home setup.

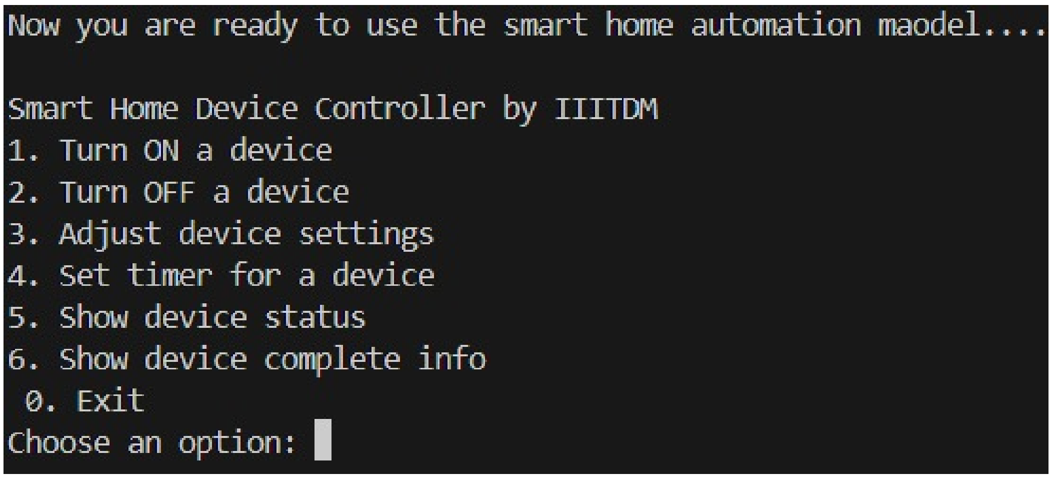


```
Welcome to smart home controller, please spare some time for initialising presentcondition of your home.....

Enter the number of fans:
1
Enter the number of lights:
1
Enter the number of tvs:
1
Enter the number of geysers:
0
Enter the number of ac's:
1
Enter the number of fridges:
0
Can you please enter 1 if you are able to assist some time for  initializing the input values ? If not, I will go ahead and initialize them to their default values. Th
ank you!

1- if you can spare time
 2- intialze to default values  : 1
Info for the light number 0
Enter the status of the light (1 for ON, 0 for OFF): 1
Enter the brightness (1-100): 28
Enter the light color (1-5): 3
Info for the fan number 0
Enter the status of the fan (1 for ON, 0 for OFF): 0
Info for the tv number 0
Enter the status of the TV (1 for ON, 0 for OFF): 1
Enter the initial channel number (0-1000): 120
Enter the initial volume (0-100): 45
Enter the initial brightness (0-100): 30
Info for the AC number 0
Enter the status of the AC (1 for ON, 0 for OFF): 1
Enter the initial temperature (17-30): 23
Enter the initial fan speed (0-5): 5
Enter the swing status (1 for ON, 0 for OFF): 1
******************************************************************************************************************************************************************
```

10. User control interface

The controller aggregates the status and control options for each device, enabling streamlined management across the entire home. Key features include:

- **Real-Time Monitoring:** View the current status of each device, including power state, speed, brightness, etc.
- **Unified Control Access:** Adjust device settings such as turning devices on/off, changing speeds, or setting timers.



```
Now you are ready to use the smart home automation maodel....

Smart Home Device Controller by IIITDM
1. Turn ON a device
2. Turn OFF a device
3. Adjust device settings
4. Set timer for a device
5. Show device status
6. Show device complete info
 0. Exit
Choose an option: 
```

11. Object oriented Design : Class breakdown

Device Class

The `Device` class serves as the base class for all other devices. It contains common functionalities that are inherited by each specific device class (AC, Fan, Fridge, etc.).

- **Constructor**: `Device(bool stat = false)`
  Initializes the device status to either false (off) or the provided status value.
- **Destructor:** `~Device()`
  A simple destructor to clean up when a Device object is destroyed.
- `bool getstatus()`
  Returns the current status of the device (true for on, false for off).
- `void ON()`
  Turns the device on by setting the status (`stat`) to true.
- `void OFF()`
  Turns the device off by setting the status (`stat`) to false.
- `void signal()`
  This method simulates sending a signal to the device (useful for remote control systems).
- `void check_power_supply()`
  Verifies if the device is connected to power. This could involve a simple check for the `stat` variable; if it's false, the device is not functional due to a lack of power.

[12pt]article amsmath xcolor graphicx

AC Class

The AC class inherits from `Device` and adds methods specific to controlling an air conditioning unit.

- **Constructor:** `AC(bool stat, bool swin, bool temp, int fanspeed)`
  Initializes the AC's status, swing mode, temperature, and fan speed.

- **Default Constructor:** `AC()`
  Initializes the AC to default settings (e.g., off, normal fan speed, standard temperature).

- **Destructor:** `~AC()`
  Clean-up operations when the AC object is destroyed.

- `void increase_temp()` / `void decrease_temp()`
  Increases or decreases the temperature within a specified range (e.g., 16–30°C).

- `void increase_fanspeed()` / `void decrease_fanspeed()`
  Adjusts the fan speed, allowing for faster or slower cooling.

- `void in_swing()` / `void off_swing()`
  Activates or deactivates the swing functionality, directing airflow throughout the room.

- `void set_timer()`
  Sets a timer to turn the AC off automatically after a specific period.

- `void set_temp()`
  Allows direct setting of the temperature to a desired value.

- `void set_swing()`
  Toggles the swing mode directly based on user input.

- `void set_fanspeed()`
  Directly sets the fan speed to a user-defined value.

- `void set_device_status()`
  Sets the device's on/off status. Useful for quickly managing power state based on conditions.

- `friend ostream& operator<<(ostream& out, AC& ac)`
  Overloads the << operator to allow easy printing of the AC's current settings, such as temperature and fan speed.

Fan Class

The Fan class represents a fan with attributes and methods for speed control and on/off status.

- **Constructor:** `Fan(bool stat, int sp)` - Initializes the fan's status and speed.

- **Default Constructor:** `Fan()` - Initializes the fan with default settings (e.g., off, medium speed).

- **Destructor:** `~Fan()` - Cleans up when the Fan object is no longer needed.

- `void set_speed(int sp)` - Sets the speed to the provided level within a valid range (e.g., 1-5).
- **Operator Overloads:** `operator++ / operator--` - Incrementally increases or decreases the fan speed, with boundaries to prevent going below or above valid speed levels.
- `void set_timer(int time)` - Sets a timer to turn the fan off after a specified duration.
- `void set_device_status(bool stat)` - Updates the fan's on/off status.
- `friend ostream& operator<<(ostream& out, Fan& fn)` - Overloads the `<<` operator to display the fan's speed and status.

Fridge Class

The Fridge class provides control over a fridge's cooling level and on/off status.

- **Constructor:** `Fridge(bool stat, int level)` - Initializes the fridge's status and cooling level.
- **Default Constructor:** `Fridge()` - Sets the fridge to default cooling and off status.
- **Destructor:** `~Fridge()` - Handles clean-up when the Fridge object is destroyed.
- `void increase_cooling()` / `void decrease_cooling()` - Adjusts the cooling level within specified boundaries to prevent freezing or undercooling.
- `void set_timer()` - Sets a timer for the fridge to turn off after a certain period, useful for energy saving.
- `void set_device_status(bool stat)` - Turns the fridge on or off based on the input status.
- `void set_cooling_level(int level)` - Directly sets the cooling level to a specified value.
- `friend ostream& operator<<(ostream& out, Fridge& fr)` - Overloads the `<<` operator to print the fridge's cooling level and status.

Light Class

The Light class enables control over brightness, color, and status of a light.

- **Constructor:** `Light(bool stat, int brt, int color)` - Sets initial values for brightness, color, and status.
- **Destructor:** `~Light()` - Destructor for clean-up operations.
- `void operator++()` / `void operator--()` - Adjusts brightness, allowing for a user-friendly way to increase or decrease brightness using `++` or `--`.

- `void change_color()` - Cycles through preset colors, allowing the user to change the light's color.
- `void set_timer(int time)` - Sets a timer to turn the light off after a set duration.
- `void set_device_status(bool stat)` - Turns the light on or off based on input.
- `void set_brightness(int brt)` - Sets brightness to a user-defined level within the range.
- `void set_color(int color)` - Directly sets the color to a specified value.
- `friend ostream& operator<<(ostream& out, Light& lt)` - Allows the current brightness and color of the light to be displayed easily.

Geyser Class

The Geyser class provides temperature control and on/off functionality for a geyser.

- **Constructor:** `Geyser(bool stat, int temp)` - Initializes the status and temperature of the geyser.
- **Destructor:** `~Geyser()` - Manages resource release upon object destruction.
- `void increase_temp()` / `void decrease_temp()` - Adjusts the temperature for heating water, within the operational range of the geyser.
- `void set_timer(int time)` - Sets a timer to automatically turn off the geyser after a set time.
- `void set_device_status(bool stat)` - Turns the geyser on or off based on input.
- `void set_temperature(int temp)` - Directly sets the temperature to a specified value.
- `friend ostream& operator<<(ostream& out, Geyser& gy)` - Overloads the `<<` operator to print the geyser's temperature and status.

TV Class

The TV class allows channel selection, volume, and brightness control, along with on/off status.

- **Constructor:** `Tv(bool stat, int chan, int vol, int bri)` - Initializes status, channel, volume, and brightness.
- **Destructor:** `~Tv()` - Manages clean-up upon object destruction.
- `void next()` / `void prev()` - Navigates to the next or previous channel in a predefined list.
- `void to_chan(int channel)` - Jumps directly to a specified channel.

- `void inc_vol()` / `void dec_vol()` - Adjusts the volume incrementally, maintaining it within limits.
- `void inc_bri()` / `void dec_bri()` - Adjusts brightness, increasing or decreasing it within a valid range.
- `void set_timer(int time)` - Sets a timer for the TV to turn off automatically after the specified time.
- `void set_channel(int channel)` - Directly sets the current channel.
- `void set_volume(int volume)` - Directly sets the volume level.
- `void set_brightness(int brightness)` - Directly sets the brightness level.
- `void set_device_status(bool stat)` - Turns the TV on or off based on input.
- `friend ostream& operator<<(ostream& out, Tv& tv)` - Overloads the << operator to display the TV's current settings.

Program Flow

1. **Main Loop:** `main()` starts by calling `displayMenu()` and receiving user input.
2. **Device Control:** Based on input, `controlDevice()` is called to execute the desired operation.
3. **Error Handling:** Each method checks for invalid inputs (e.g., setting a negative temperature) and reports an error message.

Limitations

- **Limited UI:** Console-based, may not be user-friendly for non-technical users.
- **Timer Limit:** Timer settings have a set range, which limits some device automation options.

Error Handling

- **Input Validation:** Ensures correct data types and values.
- **Device Status Check:** Methods check device status before performing actions (e.g., cannot adjust settings if a device is off).

Future Enhancements

1. **Graphical User Interface (GUI):** Improve usability by replacing the console interface with a GUI.
2. **Advanced Device Simulation:** Expand functionalities to simulate real-time device responses.
3. **Remote Access:** Enable remote access to control devices via the internet.

User Manual

Getting Started

1. **Launching the Program:** Run the program, which will display a main menu with available devices.
2. **Device Selection:** Choose a device by entering the corresponding number in the menu.

Controlling Devices

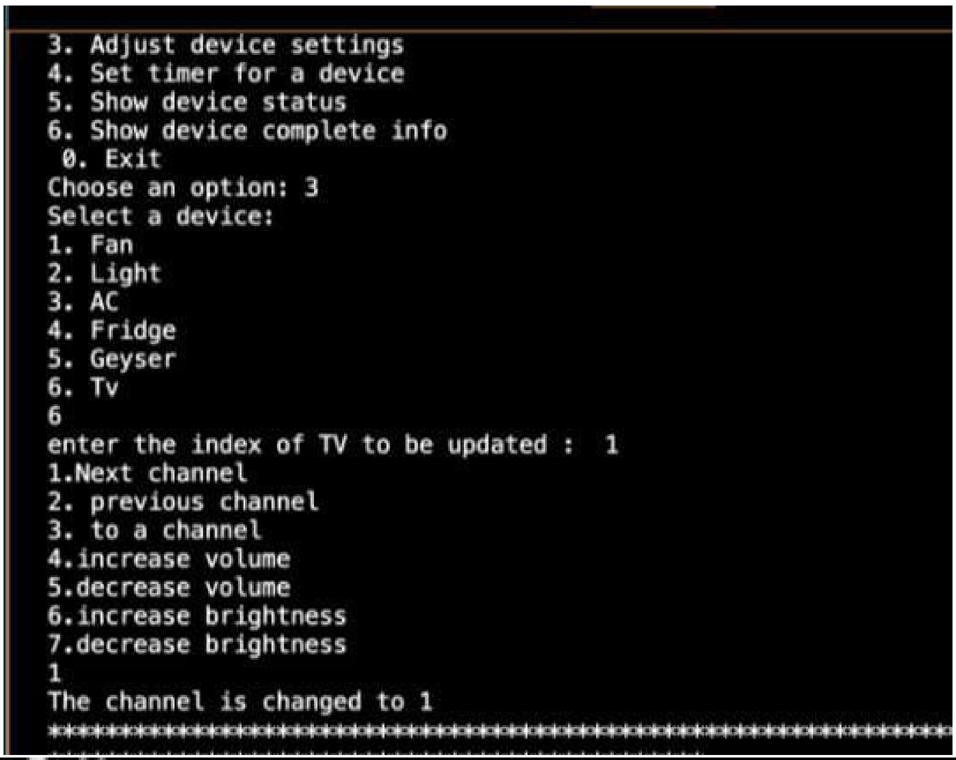
Each device type can be controlled by selecting an option in the menu. Common operations include:

- **Turning On/Off:** Toggle the power status.
- **Adjusting Settings:** Modify device-specific settings, such as temperature for AC, speed for Fan, brightness for Light, etc.
- **Setting Timers:** Automate the device with a timer, after which it will automatically turn off.

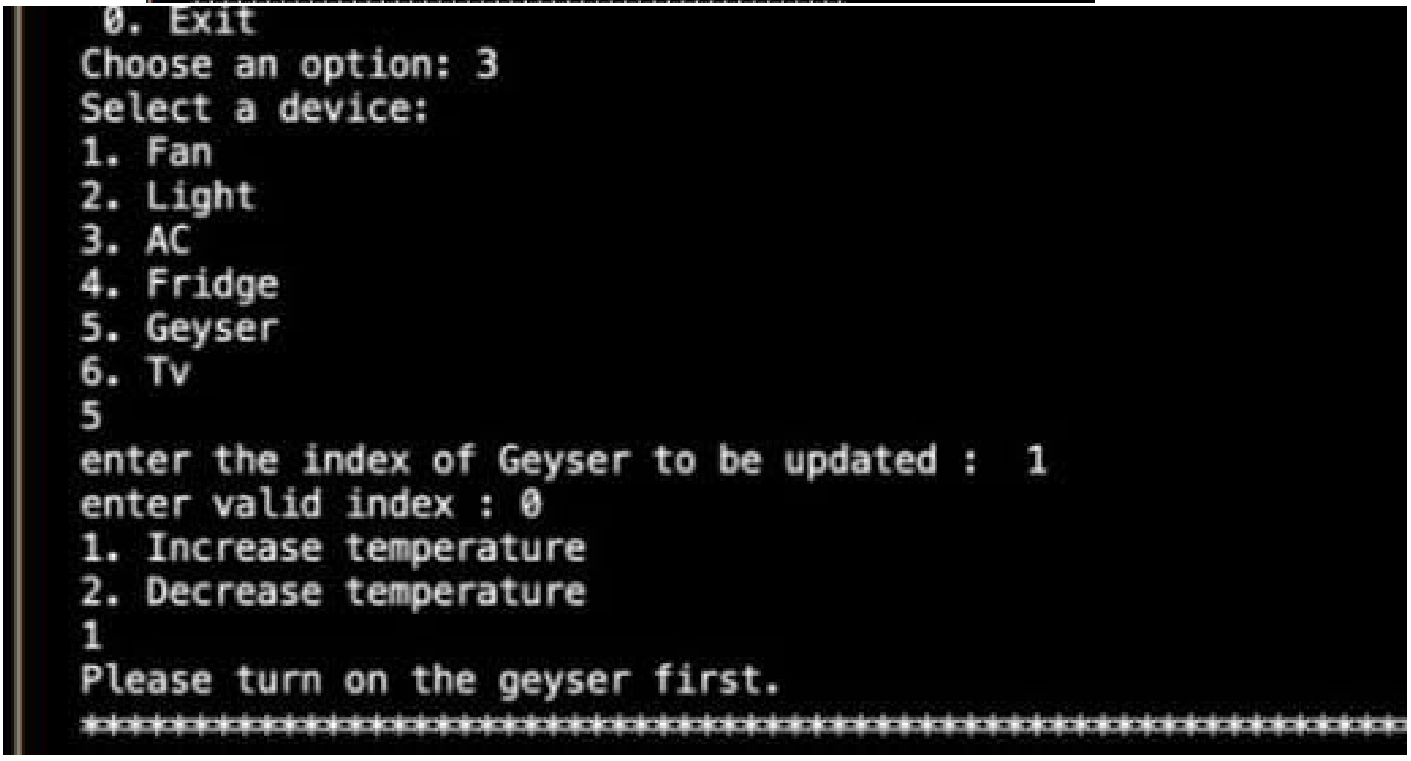
Common Use Cases

1. **Setting a Timer on the AC:** Select the AC option, set the desired temperature, fan speed, and swing, then configure a timer.
2. **Adjusting TV Volume and Channel:** Select TV, use `next()` or `prev()` to navigate channels, and `inc_vol()` or `dec_vol()` to change volume.
3. **Adjusting Lights, Color & Brightness:** select light -> use increase / decrease brightness () to control brightness & change color() to control color gradient.
4. **Adjusting Fans, speed:** select fan -> use increase / decrease speed () to control fan's speed.
5. **Adjusting AC's, swing, temperature & fan speed:** select AC -> use increase / decrease to control temperature (), change swing status() to control swing.

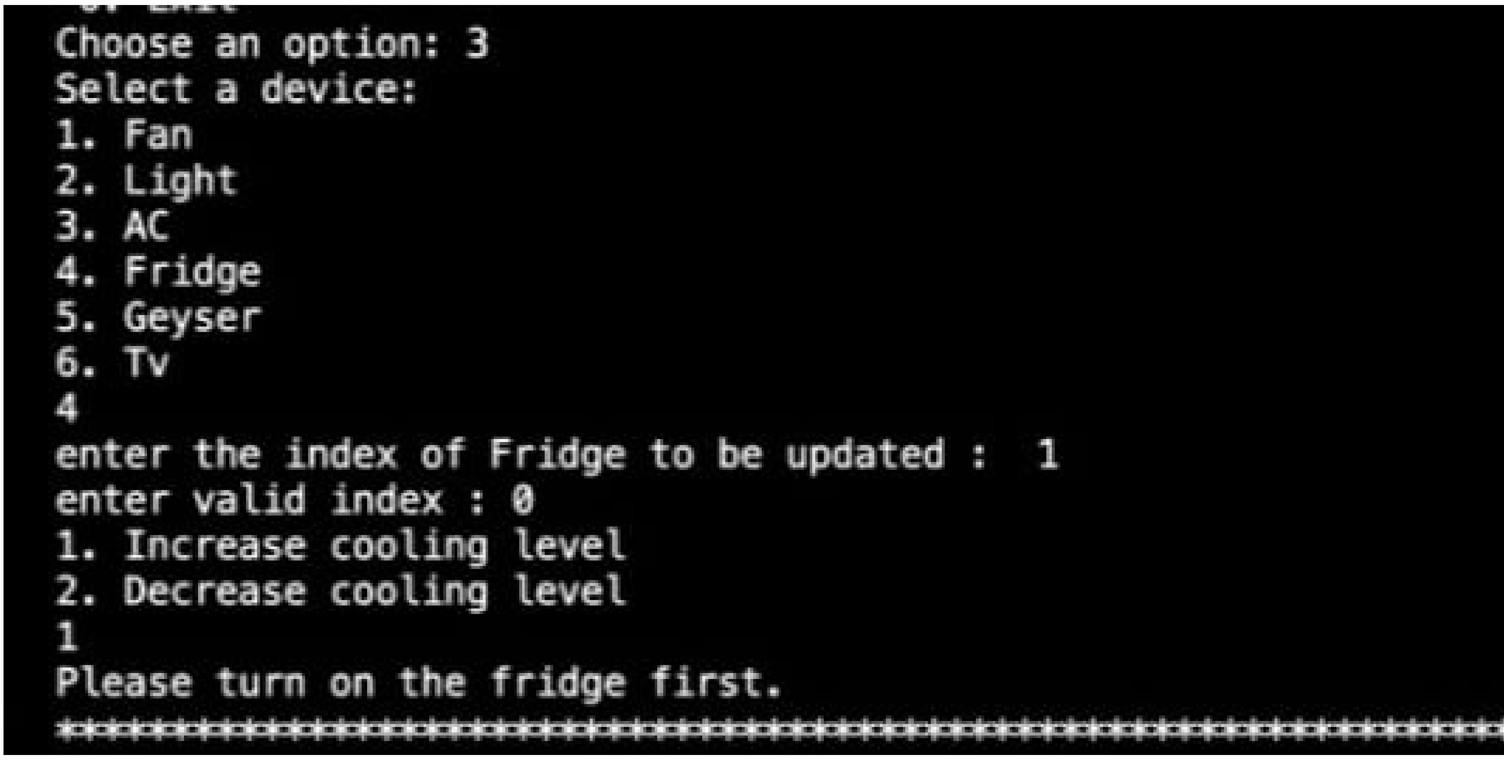
Reference picture's



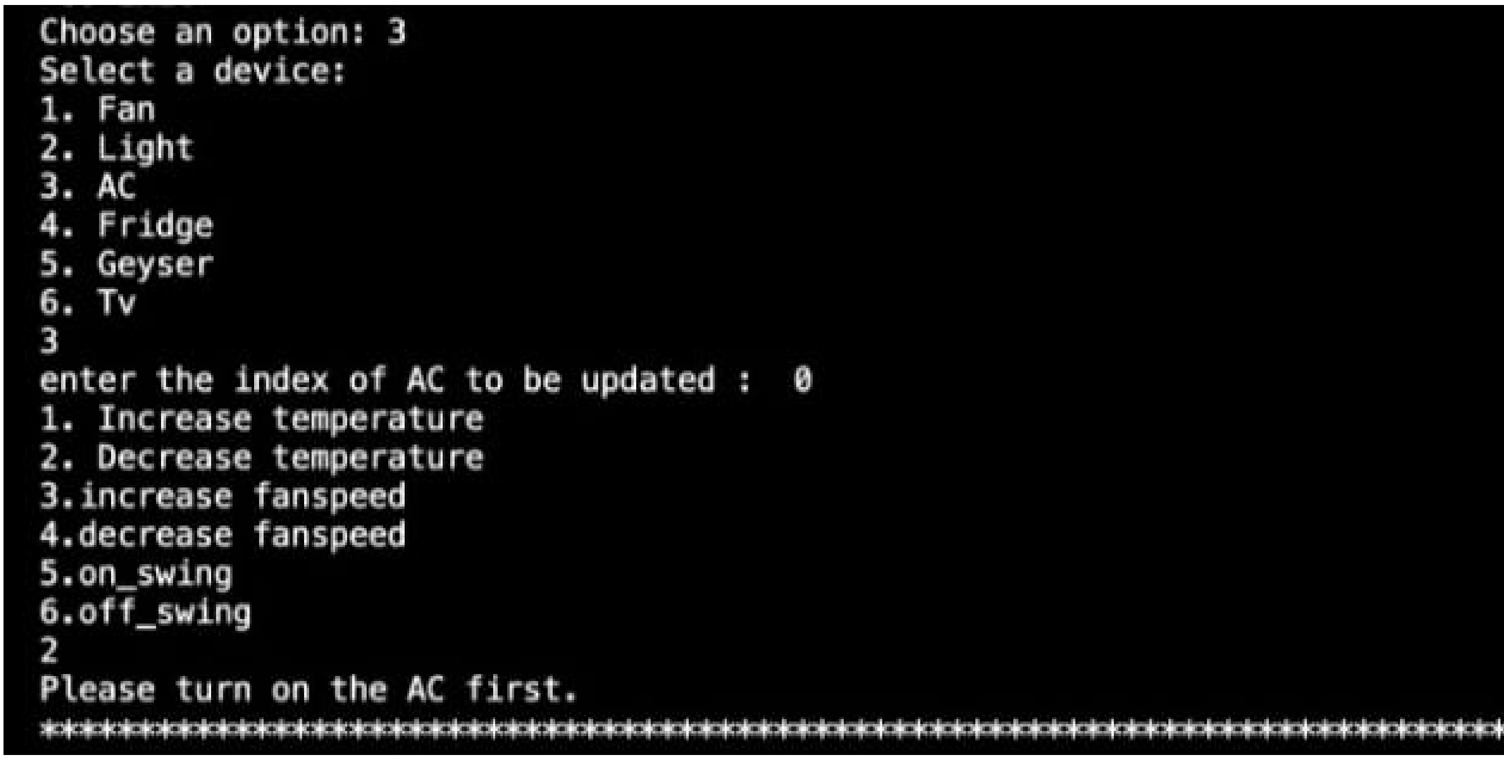
```
3. Adjust device settings
4. Set timer for a device
5. Show device status
6. Show device complete info
 0. Exit
Choose an option: 3
Select a device:
1. Fan
2. Light
3. AC
4. Fridge
5. Geyser
6. Tv
6
enter the index of TV to be updated :  1
1.Next channel
2. previous channel
3. to a channel
4.increase volume
5.decrease volume
6.increase brightness
7.decrease brightness
1
The channel is changed to 1
*****************************************************************
```



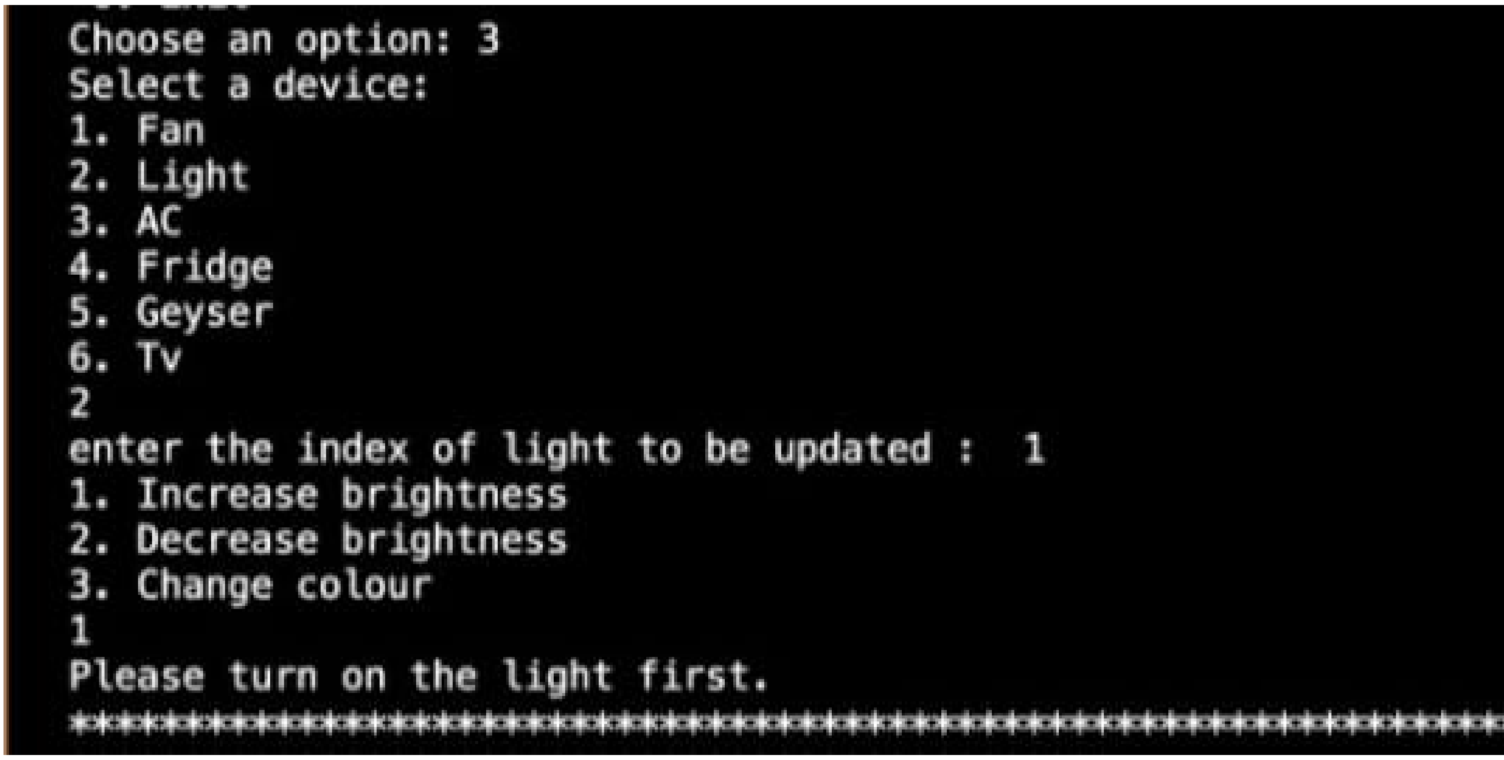
```
 0. Exit
Choose an option: 3
Select a device:
1. Fan
2. Light
3. AC
4. Fridge
5. Geyser
6. Tv
5
enter the index of Geyser to be updated :  1
enter valid index : 0
1. Increase temperature
2. Decrease temperature
1
Please turn on the geyser first.
****************************************************
```



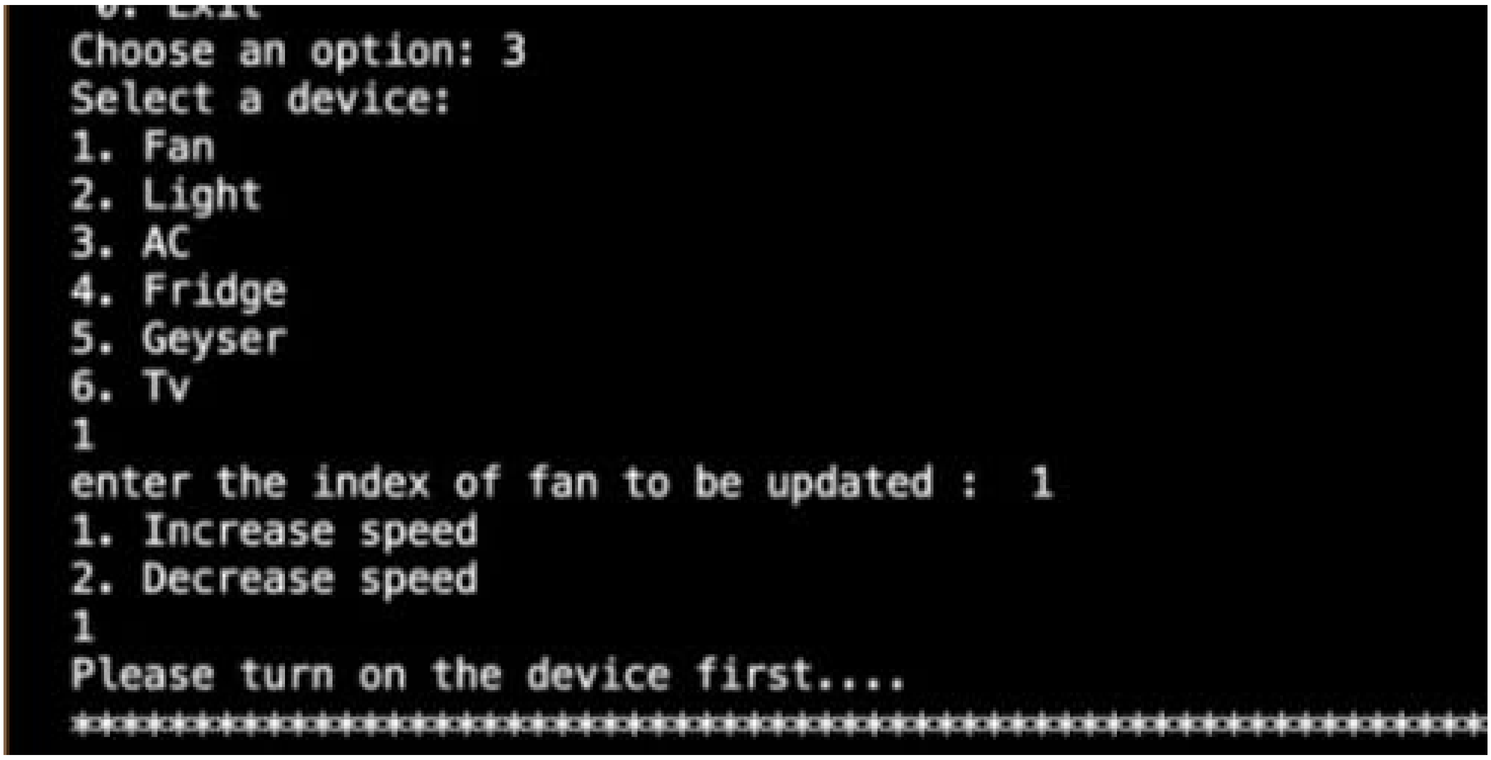
```
Choose an option: 3
Select a device:
1. Fan
2. Light
3. AC
4. Fridge
5. Geyser
6. Tv
4
enter the index of Fridge to be updated :  1
enter valid index : 0
1. Increase cooling level
2. Decrease cooling level
1
Please turn on the fridge first.
*******************************************************************
```



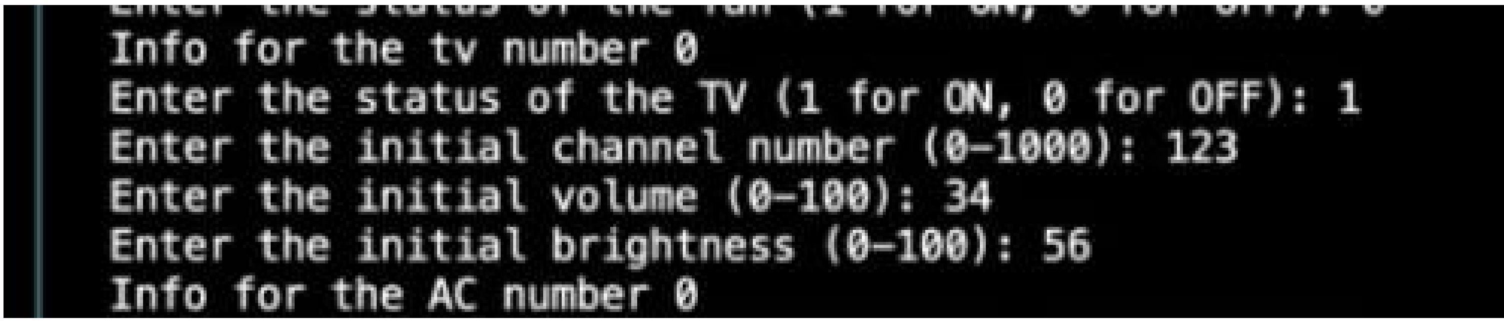
```
Choose an option: 3
Select a device:
1. Fan
2. Light
3. AC
4. Fridge
5. Geyser
6. Tv
3
enter the index of AC to be updated :  0
1. Increase temperature
2. Decrease temperature
3.increase fanspeed
4.decrease fanspeed
5.on_swing
6.off_swing
2
Please turn on the AC first.
*******************************************************************************
```



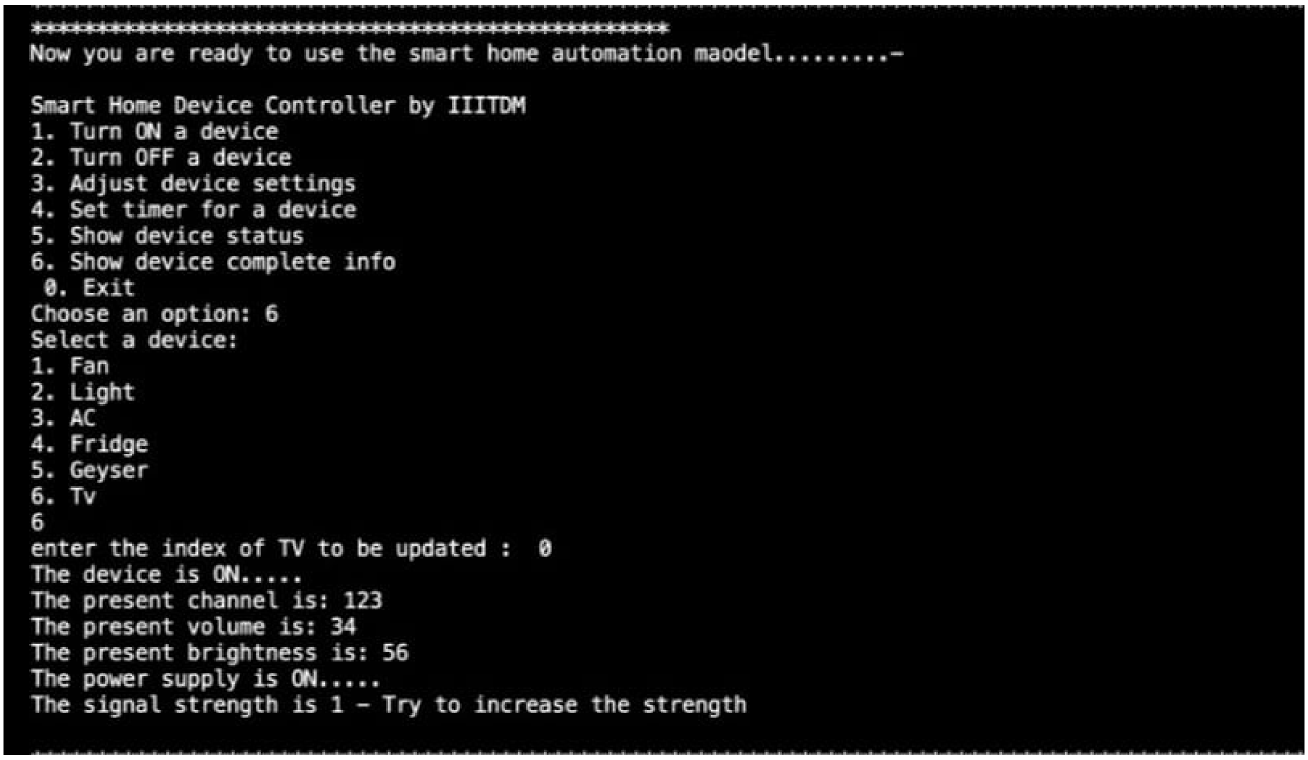
```
Choose an option: 3
Select a device:
1. Fan
2. Light
3. AC
4. Fridge
5. Geyser
6. Tv
2
enter the index of light to be updated :  1
1. Increase brightness
2. Decrease brightness
3. Change colour
1
Please turn on the light first.
****************************************************************
```



```
Choose an option: 3
Select a device:
1. Fan
2. Light
3. AC
4. Fridge
5. Geyser
6. Tv
1
enter the index of fan to be updated :  1
1. Increase speed
2. Decrease speed
1
Please turn on the device first....
****************************************************************
```



```
Info for the tv number 0
Enter the status of the TV (1 for ON, 0 for OFF): 1
Enter the initial channel number (0-1000): 123
Enter the initial volume (0-100): 34
Enter the initial brightness (0-100): 56
Info for the AC number 0
```



```
************************************************
Now you are ready to use the smart home automation maodel.........-

Smart Home Device Controller by IIITDM
1. Turn ON a device
2. Turn OFF a device
3. Adjust device settings
4. Set timer for a device
5. Show device status
6. Show device complete info
 0. Exit
Choose an option: 6
Select a device:
1. Fan
2. Light
3. AC
4. Fridge
5. Geyser
6. Tv
6
enter the index of TV to be updated :  0
The device is ON.....
The present channel is: 123
The present volume is: 34
The present brightness is: 56
The power supply is ON.....
The signal strength is 1 - Try to increase the strength
```

Your's team 13

Thank You