# **Home Automation Server Documentation**

## **Overview**

The Home Automation Server is a simple server application written in C++ to control and monitor various devices in a home automation system. It utilizes sockets for communication and allows clients to send commands to control devices like lamps, fans, TVs, refrigerators, air coolers, and washing machines.

## **Code Structure**

The code is organized into a HomeAutomationServer class, which handles the server functionality, and a main function to instantiate and start the server.

### **Class: HomeAutomationServer**

#### Properties

* server\_socket: Socket descriptor for the server.
* server\_address: Structure containing server address information.
* devices: An unordered map to store the status (on/off) of various devices.

#### Methods

* Constructor (HomeAutomationServer()):
  + Creates a socket (server\_socket) and initializes server address information.
  + Binds the socket to a specific port.
  + Listens for incoming connections.
* Method (start()):
  + Accepts client connections in a loop.
  + Calls handle\_client for each connected client.
* Method (handle\_client(int client\_socket)):
  + Handles communication with a connected client.
  + Receives commands from the client and performs corresponding actions on devices.
  + Sends a response back to the client based on the command execution.

### **main Function**

* Instantiates an object of the HomeAutomationServer class.
* Calls the start method to begin listening for client connections.

## **Communication Protocol**

The server expects commands in the format device,action, where device represents the target device (e.g., "lamp") and action represents the desired action (e.g., "on" or "off"). The server responds with the status of the requested device or an error message.

## **Device Control**

The server supports the following commands:

* Get Status:
  + Command Format: device,get\_status
  + Response: "on" or "off" indicating the current status.
* Turn On/Off:
  + Command Format: device,on or device,off
  + Response: Confirmation message or an error message if the command is invalid.

## **Error Handling**

* The server provides error messages for various scenarios, such as invalid commands, invalid devices, errors in socket operations, and failed client connections.

## **Usage**

* Compile the code using a C++ compiler.
* Run the compiled executable to start the Home Automation Server.
* Connect to the server from client devices and send commands using the specified protocol.

## **Example Usage**

bash

Copy code

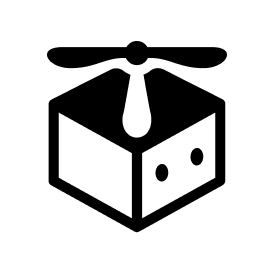
$ ./home\_automation\_server

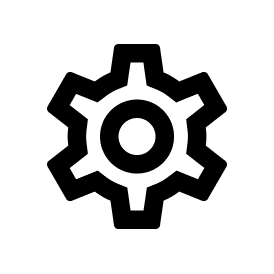
Home Automation Server is listening on port 3333

Connection established with a client!

Copy And Save

Share

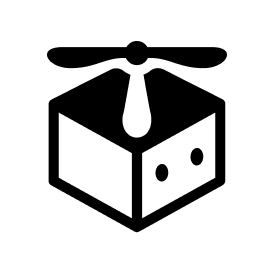
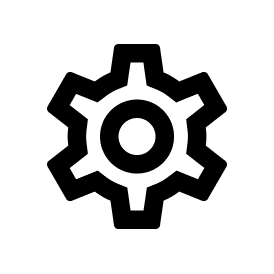
Ask Copilot



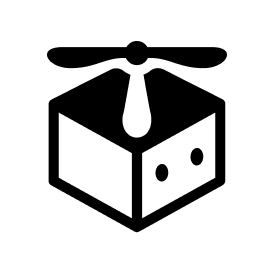
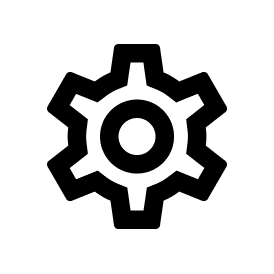
### **Client Interaction**

* Send Command:
* bash
* Copy code

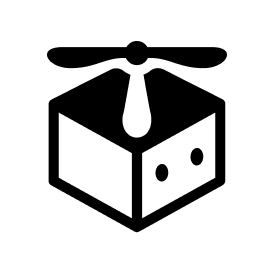
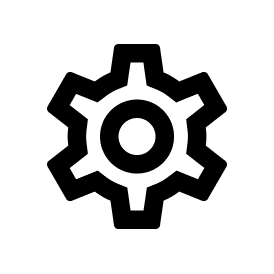
$ echo "lamp,on" | nc localhost 3333

* Copy And Save
* Share
* Ask Copilot
* 
* Server Response:
* csharp
* Copy code

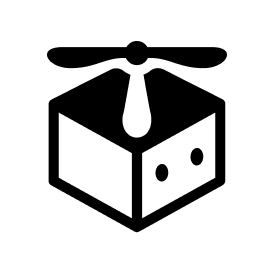
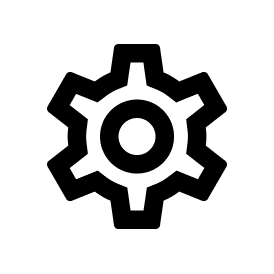
lamp is turned on

* Copy And Save
* Share
* Ask Copilot
* 
* Get Status:
* bash
* Copy code

$ echo "fan,get\_status" | nc localhost 3333

* Copy And Save
* Share
* Ask Copilot
* 
* Server Response:
* vbnet
* Copy code

off

* Copy And Save
* Share
* Ask Copilot
* 

## **Conclusion**

The Home Automation Server provides a foundation for controlling and monitoring devices in a home automation system. It can be extended to support additional devices and commands as needed. Ensure that clients adhere to the specified communication protocol for seamless interaction with the server.

# **Home Automation Client Documentation**

The Home Automation Client is a C++ program that allows users to interact with the Home Automation Server to control and monitor various devices. It establishes a connection with the server and provides a menu-driven interface for users to view device statuses, send commands to control devices, and exit the program.

## **Code Structure**

The code is organized into a HomeAutomationClient class, which handles client functionality, and a main function to instantiate and run the client.

### **Class: HomeAutomationClient**

#### Properties

* client\_socket: Socket descriptor for the client.
* server\_address: Structure containing the server address information.

#### Methods

* Constructor (HomeAutomationClient()):
  + Creates a socket (client\_socket) for communication with the server.
  + Initializes the server address.
  + Connects to the Home Automation Server.
* Method (send\_command(const std::string &command)):
  + Sends a command to the server using the established socket connection.
* Method (Close()):
  + Closes the client socket.
* Method (get\_status(const std::string &device) -> std::string):
  + Retrieves the status of a specific device from the server.
* Method (control\_device(const std::string &device, const std::string &action)):
  + Sends a control command to the server for a specific device.
* Method (menu()):
  + Provides a menu-driven interface for users to interact with the server.
  + Displays the status of available devices and allows users to send control commands.

### **main Function**

* Instantiates an object of the HomeAutomationClient class.
* Invokes the menu method to start the interactive menu.
* Closes the client connection when the user exits the program.

## **User Interaction**

The client menu provides the following options:

* View Device Status:
  + Displays the current status (on/off) of each available device.
* Control Devices:
  + Allows the user to turn devices on or off.
* Exit:
  + Closes the client program.

## **Usage**

* Compile the code using a C++ compiler.
* Run the compiled executable to start the Home Automation Client.
* Follow the on-screen menu prompts to interact with the Home Automation Server.

## **Example Interaction**

bash

Copy code

$ ./home\_automation\_client

Available Devices with current status:

1. Lamp: off

2. Fan: on

3. TV: off

4. Refrigerator: on

5. Air Cooler: off

6. Washing Machine: on

7. Exit

Enter your choice: 2

Enter action (on/off): off

Server response: fan is turned off

Available Devices with current status:

1. Lamp: off

2. Fan: off

3. TV: off

4. Refrigerator: on

5. Air Cooler: off

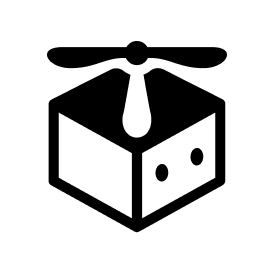
6. Washing Machine: on

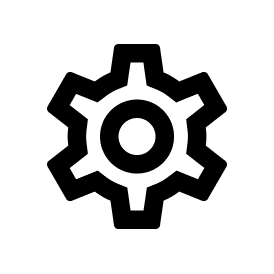
7. Exit

...

Copy And Save

Share

Ask Copilot



## **Conclusion**

The Home Automation Client provides a user-friendly interface to interact with the Home Automation Server. Users can easily view the status of devices and control them through a simple menu-driven system. Ensure that the Home Automation Server is running before starting the client for successful communication.