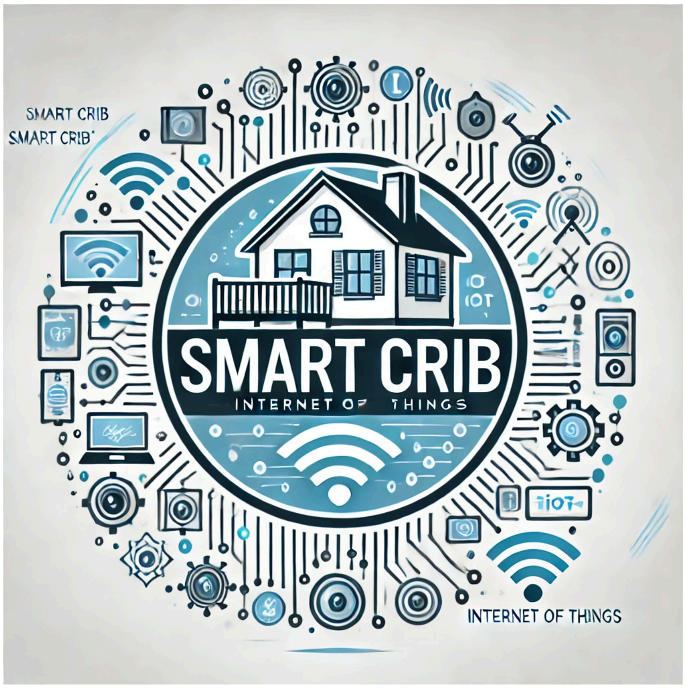
Department of Computer Science & Engineering The University of Texas at Arlington

Architectural Design Specification CSE 4316: Senior Design I Summer 2024



Internet of things

Smart Crib

Don Dang
Luis Del Rio Carrillo
Zait Martinez
Pranav Umakant Pujar

Revision History

Revision	Date	Author(s)	Description
0.1	08/04/2024	DD, PP, LR	document creation
0.2	08/04/2024	DD, PP	Table of Contents
0.3	08/04/2024	DD	List of Figures
0.4	08/04/2024	DD	List of Tables
1.0	08/04/2024	DD	Introduction
2.0	08/05/2024	DD	System Overview Title
2.1	08/05/2024	DD	Smart Device Layer Description
2.2	08/05/2024	DD	Cloud Server Layer Description
2.3	08/05/2024	PP	App Layer Description
3.0	08/05/2024	PP	Subsystems Definitions & Data Flow
4.1	08/05/2024	LR	Change Title - Smart Device Layer Subsystems
4.1.1	08/05/2024	DD	Assumptions - Hardware
4.1.2	08/05/2024	DD	Responsibilities - Hardware
4.1.3	08/05/2024	DD	Subsystem Interfaces - Hardware
4.2.1	08/05/2024	LR	Assumptions - Software
4.2.2	08/05/2024	LR	Responsibilities - Software
4.2.3	08/05/2024	DD	Subsystem Interfaces - Software
4.3.1	08/05/2024	DD	Assumptions - API Communication
4.3.2	08/05/2024	DD	Responsibilities - Software
4.3.3	08/05/2024	DD	Subsystem Interfaces - DD
5.1.1	08/05/2024	PP	Cloud/Business Logic Layer
6.1.1	08/05/2024	PP	App Layer Subsystems

Table Of Contents

List of Figures

List of Tables

1 Introduction

We plan to create an IoT system from the ground up. This system will be made up of three main components all of which will be able to communicate remotely. This system will come together to create a home automation system that is capable, expandable, and easy to use.

2 System Overview

We have divided our system into three main components based on independence and functionality of individual components. These three components include the application, central server, and the smart devices. Each component can be developed individually then connected for testing and verification later on in the development process. Once basic requirements are set for each component, individual development will begin. After a basic form of the component is completed, api development must begin making sure each component meets each other's api requirements. The goal is for both the application and devices to request data from a centralized server, allowing remote control of smart devices

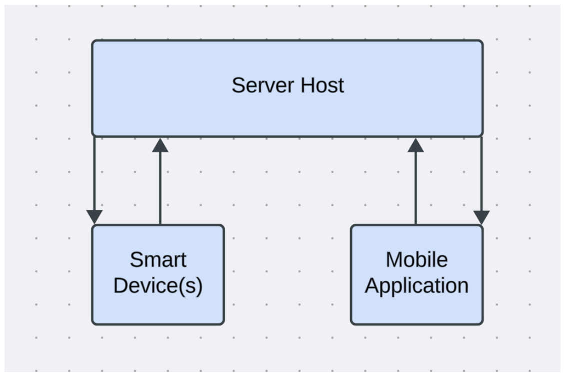


Figure 1: A simple architectural layer diagram

2.1 Smart Device Layer Description

All smart devices will be composed of at least a ESP32 board capable of communicating through wifi connectivity. This microcontroller will then connect and control a different set of hardware components such as sensors, lights, motors etc. This will compose the physical Smart Device Layer, this layer will be controlled remotely or set automatically depending on its corresponding softwared. Smart devices will allow users to add specific features to their home allowing for easy integration and centralized control of all smart devices within their home. The smart device will be connected to the internet via wifi and request data from a central server. Here it will gather data and update itself and the server accordingly, properly setting its hardware components to reflect users commands.

2.2 Cloud/Business Logic Layer Description

Each layer should be described separately in detail. Descriptions should include the features, functions, critical interfaces and interactions of the layer. The description should clearly define the services that the layer provides. Also include any conventions that your team will use in describing the structure: naming conventions for layers, subsystems, modules, and data flows; interface specifications; how layers and subsystems are defined; etc.

All the smart devices have different states (e.g., color of smart light, brightness of smart light, thermostat temperature, etc) which need to be controllable by the mobile app. These state changes are initiated by a user through the app, which are communicated to the server. The server tracks the changes and when an API request is made by a hardware device to check for state change requests, the response body sent from the Azure server will contain the state change request initiated by the client. The rest is handled by the Hardware Layer.

2.3 Layer Z Description

Each layer should be described separately in detail. Descriptions should include the features, functions, critical interfaces and interactions of the layer. The description should clearly define the services that the layer provides. Also include any conventions that your team will use in describing the structure: naming conventions for layers, subsystems, modules, and data flows; interface specifications; how layers and subsystems are defined; etc.

The App layer contains various UI features to allow for users to make changes in the state of the smart hardware devices present. These functions are enabled through a user-friendly React Native Mobile App interface. This app can send HTTP POST requests to the Azure server to initiate state changes in hardware as requested by the user.

3 Subsystem Definitions & Data Flow

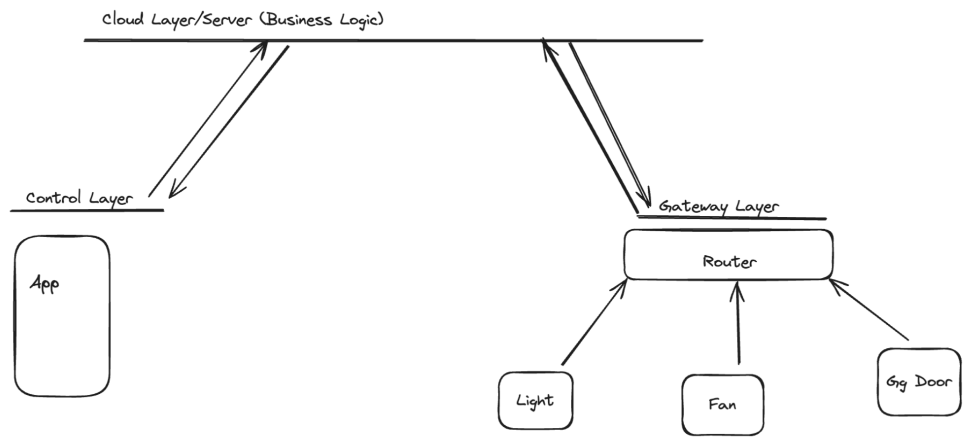


Figure 2: A simple data flow diagram

4 Smart Device Layer Subsystems

In this section, the layer is described in some detail in terms of its specific subsystems. Describe each of the layers and its subsystems in a separate chapter/major subsection of this document. The content of each subsystem description should be similar. Include in this section any special considerations and/or trade-offs considered for the approach you have chosen.

For this example I will be using our Smart Light Device which is the first device we will integrate into our system

The Smart Light Device will have its hardware subsystem; these components give the device functionality, it will have a software subsystem to collect and control its hardware components accordingly, and lastly the api/communication subsystem which will be required to communicate necessary data back and forth between other layers.

4.1 Subsystem 1 Hardware

For the hardware portion of the device a ESP32 microcontroller will be physically connected to different components, for our Smart Light Device these components include LED module, Motion Sensor Module, and a Light Sensor Module. The ESP32 has the capability to power, receive and send signals to all of these components. These components capture raw data and send it back to the microcontroller which decides what to do with this data and can also give new instructions to components. The microcontroller will use this data to give other layers feedback about itself and its environment, the LED's will reflect these commands accordingly.

4.1.1 Assumptions

The assumptions made are that all our devices will use the same main microcontroller (ESP32) but the components connected will vary from device to device, All hardware components will be physically connected to GPIO pins for input and output interfaces

4.1.2 Responsibilities

The responsibilities of this subsystem is to reflect commands and automatic settings, hardware will be what users get to see and interact with when it comes to home automation (aside from application). For each user command there must be a hardware component to reflect the command accordingly.

4.1.3 Subsystem Interfaces

Interface between ESP32 and LED Module

Interface between ESP32 and Motion Sensor

Interface between ESP32 and Light Sensor

Table 1: LED interfaces

ID	Description	Inputs	Outputs
SD1	ESP32 to LED MODULE	LED command	Color change Brightness Change On/Off
SD2	Motion Sensor to ESP32	Sense Motion	Corresponding LED command
SD3	Motion Sensor to ESP32	Sense Light	Brightness change Command

4.2 Subsystem 2 Software

Software will be embedded into every smart device to control the hardware modules it has been given access to. Our Smart Light Device will include software that communicates with the LED module, and sensor modules. Corresponding libraries will be utilized to ensure communication compatibility between components is fully functional. Data collection will also be customized and collected accordingly through software such as when the last time we sensed motion, the amount of light in the room, the current state of the LED, the color, and brightness. This will all later be used to send feedback to other layers in the system. Software will also be in charge of adjusting and updating hardware components according to changes in its environment or changes requested by other layers.

4.2.1 Assumptions

Software will differ from device to device since data collection will be different and so will outputs and inputs. All software in Smart Devices will be written in the same language and format.

4.2.2 Responsibilities

Software must collect useful data available and react in correlation with data.

4.2.3 Subsystem Interfaces

Interface between input data from hardware and output signals

Interface between server Data and current hardware status data

Table 2: Server interface

ID	Description	Inputs	Outputs
US1	Server data update	Request Update from Server	Process data update hardware status
US2	Current Device Status Update	Request Update from Server	Server Update current hardware status

4.3 Subsystem 3 Api / Communication

The ESP32 board will be in charge of communicating with our server constantly in order to keep the server up to date with current data, as well as keeping itself updated and aligned with user requests. Wifi connection is crucial as it is the means through which we will be communicating. Http requests will be made to the server a couple times a minute ensuring we are always updated, however responsiveness is not a high priority therefore we see no issue with having a small delay in reaction times when it comes to updating device status when users request or command a change/adjustment. The Smart Device will only communicate with the server who will be available at all times. Every time they communicate they will both transfer data, the device will let the server know if any adjustments or commands have been made by a user and the device will let it know its current state allowing the device to compare this data and adjust accordingly, this system will allow Smart Devices to be controlled remotely from anywhere with internet access.

4.3.1 Assumptions

All layers will communicate using http request in a server client style model. They will have internet access at all times and available for one another at all times.

4.3.2 Responsibilities

This subsystem is responsible for communicating relevant data with other layers as well as gathering relevant data from other layers in order to allow embedded software to control hardware components accordingly.

4.3.3 Subsystem Interfaces

Interface between Server and Device

Table 2: Subsystem interfaces

ID	Description	Inputs	Outputs
SR1	Device makes Server Request	Update Request	Update Server and Update Device

5 Cloud/ Business Logic Layer Subsystems

In this section, the layer is described in some detail in terms of its specific subsystems. Describe each of the layers and its subsystems in a separate chapter/major subsection of this document. The content of each subsystem description should be similar. Include in this section any special considerations and/or trade-offs considered for the approach you have chosen.

This layer is responsible for being an always-on point of contact for the client (mobile application) as well as the individual hardware systems within the hardware layer (layer 1). It should facilitate one-way command transmission from the client to the respective hardware device.

5.1 Subsystem 1 - Azure App Service

The first and only subsystem is "Azure App Service". This is a PaaS service that will allow us to permanently host a server on the cloud. It will also allow the various hardware devices as well as the mobile pap client to interact with it through API/HTTP requests.

5.1.1 Assumptions

We assume the following:

1. The costs of this Azure cloud service will be reasonably within the $200 sign-up credit offered for the new account.
2. The API connection between the backend of the mobile app and the server hosted in the cloud will be reasonably fast in order to show functionality in real time.
3. The ESP32 chips that are installed on the smart devices will be able to connect to the server via API/HTTP requests.

5.1.2 Responsibilities

Handle API requests from the backend of the client app as well as the individual hardware devices. Execute business logic in order to ensure state changes in the individual devices happen as per the client's request.

5.1.3 Subsystem Interfaces

Table 3: API interfaces

ID	Description	Inputs	Outputs
C1	Client (Mobile App)	Payload from API Request	API Response
C2	Individual Hardware Subsystems	Payload from API Request	API Response

6 App Layer Subsystems

The SmartCrib React Native app layer consists of several key subsystems that collectively provide a seamless user experience for home automation. The User Interface (UI) subsystem is responsible for rendering screens and handling user interactions, while the Backend Communication subsystem manages data exchange with the server. The Device Management subsystem enables integration, control, and monitoring of connected devices. Together, these subsystems ensure secure, efficient, and user-friendly control of smart home devices.

6.1 Subsystem 1: User Interface

The User Interface (UI) subsystem is responsible for all aspects of user interaction with the SmartCrib app. This includes rendering screens, handling user inputs, and displaying data from connected devices. The UI provides an intuitive and seamless experience, allowing users to easily control and monitor their smart home devices. It serves as the front-end layer that communicates with the backend services to fetch and display data, as well as to send user commands to the devices.

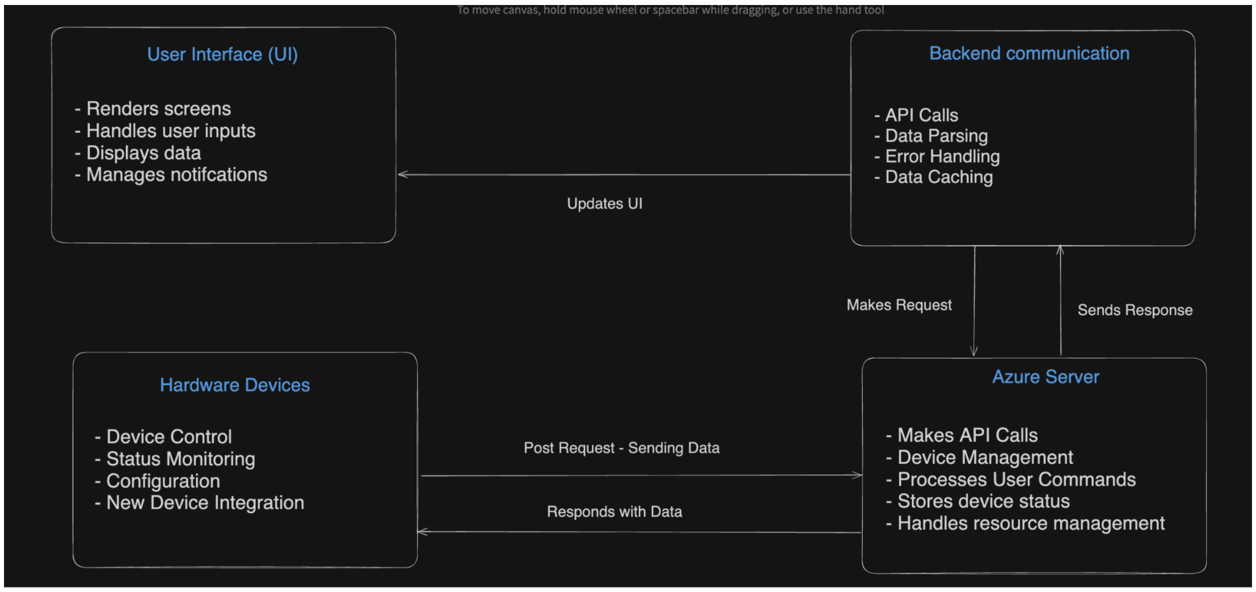


Figure 3: App Layer Diagram

6.1.1 Assumptions

- The UI subsystem assumes that users have a basic understanding of mobile apps and can navigate through typical app interfaces.
- The UI is designed to interact with the device's native features, such as touch input and notifications.
- It is assumed that the devices running the app will have internet connectivity to communicate with the backend systems.
- Interfaces and interactions with the backend server are stable and will not undergo frequent changes.

6.1.2 Responsibilities

- **Rendering Screens**: The UI subsystem is responsible for rendering the various screens of the app, such as the home screen, device control screen, and settings.
- **User Interaction**: It handles all user interactions, including button presses, gestures, and navigation between different screens.
- **Displaying Data**: Fetches and displays data from the backend, such as the status of connected devices.
- **User Notifications**: Manages notifications to alert users about important events or updates from the SmartCrib system.

6.1.3 Subsystem Interfaces - User Interface

Table 4: User Interface

ID	**Description**	**Inputs**	**Outputs**
UI-1	Main Navigation Interface	User gestures, commands	Screen transitions
UI-2	Device Status Display Interface	Device status data	Visual status indicators
UI-3	User Notification Interface	Backend notifications	App notifications
UI-4	Settings Configuration Interface	User Input	Updated settings data

6.2 Subsystem 2: Backend Communication

The Backend Communication subsystem manages all interactions between the SmartCrib app and the backend server. It handles API requests to fetch and send data, ensuring that the app remains synchronized with the backend. This subsystem is crucial for maintaining the flow of information between the app and the devices, enabling users to control their smart home devices and receive real-time updates.

6.2.1 Assumptions

- Assumes stable and reliable internet connectivity to communicate with backend servers.
- The backend APIs follow RESTful principles and provide consistent and predictable responses.
- Authentication and authorization mechanisms are securely implemented on the backend.

6.2.2 Responsibilities

- **API Calls**: Makes HTTP requests to the backend server to fetch and send data.
- **Data Parsing**: Parses JSON responses from the server and converts them into usable data structures within the app.
- **Error Handling**: Manages errors related to network issues, server downtime, or invalid responses.
- **Data Caching**: Implements caching mechanisms to store frequently accessed data and reduce the number of network requests.

6.2.3 Backend Communication Interface

ID	**Description**	**Inputs**	**Outputs**
BC-1	Device Control API Interface	User commands	Server responses
BC-2	Device Status Fetch Interface	API requests	Device status data
BC-3	User Authentication Interface	Login credentials	Authentication tokens
BC-4	Notification Fetch Interface	API requests	Notification data

6.3 Subsystem 3: Device Management

The Device Management subsystem is responsible for the integration and control of smart home devices within the SmartCrib ecosystem. This subsystem allows users to add new devices, configure existing ones, and manage their operations. It ensures that devices are properly connected and can be controlled through the app, providing a seamless user experience.

6.3.1 Assumptions

- Assumes that the devices being controlled (e.g., lights, garage doors) are compatible with the SmartCrib system and follow the expected communication protocols.
- Assumes stable communication between the app and the devices, either directly or via the backend server.

6.3.2 Responsibilities

- **Device Control:** Sends commands to devices to change their state (e.g., turn lights on/off).
- **Device Status Monitoring**: Continuously monitors and fetches the current status of connected devices.
- **Device Configuration:** Allows users to configure device settings such as names, icons, and operational parameters.
- **Integration with New Devices:** Provides mechanisms to integrate and manage new devices added to the SmartCrib system.

6.3.3 DEVICE MANAGEMENT INTERFACE

ID	**Description**	**Input**	**Output**
DM-1	Device Command Interface	User commands	Device actions
DM-2	Device Status Interface	Device status data	Displayed status
DM-3	Device Configuration Interface	Configuration data	Updated device settings
DM-4	New Device Integration Interface	Device information	Integrated device data

References

ip_admin. “Unpacking IoT Architecture: Layers and Components Explained.” *Device Authority*, 16 Sept. 2023, deviceauthority.com/unpacking-iot-architecture-layers-and-components-explained/#:~:text=IoT%20architecture%20is%20an%20organised.

“LibGuides: Internet of Things (IoT): Architecture of Internet of Things.” *Libguides.com*, 2020, bowiestate.libguides.com/internet_of_things_iot/architecture_of_Internet_of_things. Accessed 6 Aug. 2024.

Simmons, Adam. “Internet of Things (IoT) Architecture: Layers Explained.” *Dgtl Infra*, 14 Nov. 2022, dgtlinfra.com/internet-of-things-iot-architecture/.