COVER
- From Past to Present: Harnessing Matter to Integrate Legacy Devices into the Next-Gen Smart Home Ecosystem

SONG WOOJUNG
Dept. Information System
Hanyang University
Seoul, Republic of Korea
opusdeisong@gmail.com

KWON KITAE
Dept. Information System
Hanyang University
Seoul, Republic of Korea
jrinonamu@gmail.com

KIM JINA
Dept. Information System
Hanyang University
Seoul, Republic of Korea
drjina02@gmail.com

YEO DAKYUM
Dept. Information System
Hanyang University
Seoul, Republic of Korea
yeodakyum@gmail.com

***Abstract*—In recent times, the global smart home market has seen remarkable expansion, projected to cross $164 billion by 2028. Driven by the desire for convenience, security, and energy savings, we're witnessing sophisticated device integrations in our homes. However, this growth isn't without its challenges. The array of smart home devices from different brands has fragmented the user experience, leading to a maze of applications for device control. Many third-party apps, despite their intentions, often don't mesh seamlessly with central control devices like smartphones, a challenge evident in platforms such as Apple Home and Google Home App . Additionally, a significant number of devices still lack IoT integration, which challenges expanding smart home market. [1]**

Addressing these challenges, we introduce the LG Cover, an innovative solution designed to bridge gaps in the smart home ecosystem. We suggest specialized integration for all LG devices, resurrecting legacy devices through the Matter protocol, and ensuring consistent user experience across various brands. This system supports the integration of both Matter and ThinQ devices, and also supports older devices based on IR technology, revitalizing its utility within modern smart home setups.

To facilitate seamless integration, the LG Cover adopt a structured setup process involving the registration of the hub to the ThinQ app and the Matter API device. ThinQ supporting or IR supporting devices are connected to LG Cover with proprietary protocols, and LG Cover mediates the connected devices to Matter enabled controllers. Users are enabled to communicate with non-Matter-supporting devices through LG Cover using Matter API. With LG Cover's comprehensive approach, users can experience the convenience of a unified smart home ecosystem, irrespective of the device's age or brand.

Role Assignments

Roles	Name	Description
User/Customer	SONG WOOJUNG	User/Customer plays the role of a various ages who needs support with an old LG devices that doesn't support matter api. Despite the advancements in tech and the ubiquity of smart homes, not every device owned is compatible with the latest standards. This poses a challenge in achieving a cohesive experience, especially when most contemporary hubs primarily cater to the latest models and technologies. Therefore, the LG Cover plays an instrumental role in bridging this gap. While it is equipped to support the Matter API, its design philosophy is to be inclusive. The goal is to serve users like Alex, ensuring that their older, reliable LG devices don't become obsolete. This includes even those devices that don't support LG ThinQ.
Product Designer	KWON KITAE	A Product Designer is responsible for understanding user needs and business objectives to conceptualize and design products that provide effective solutions; they engage in user research to gather insights, sketch initial design ideas, create wireframes and interactive prototypes, collaborate with cross-functional teams like product managers, engineers, and marketers to refine designs, ensure consistency in brand aesthetics and user experience across products, gather and act on feedback from user testing sessions. Also works with documentation that helps other roles communicate with each other more effectively.

Software developer	KIM JINA	A software developer is responsible for collaborating with development manager, product owners, and product designer to understand software requirements, converting these into functional specifications, designing the software architecture based on these requirements, writing clean and maintainable code, ensuring software is scalable and robust, testing the software for bugs and inconsistencies, deploying software to production environments, maintaining and updating software as necessary, and documenting development processes and software functionalities to ensure continuity and ease of future modifications. Also works with connection between the matter API and ThinQ devices to maintain connectivity.
Development manager	YEO DAKYUM	A Development Manager oversees the software development process, coordinating with cross-functional teams to ensure projects are completed on time and within budget, establishing and implementing development methodologies, setting objectives and deliverables for the development team, mentoring and providing guidance to developers, ensuring code quality and adherence to standards. Also works with the actual hardware chip that will be used in Cover.

I. Introduction

A. Motivation

- Growth of Smart Home Systems
  The smart home market has experienced remarkable growth over the last few years. According to Statista, the global smart home market is projected to surpass $141 billion by 2023, up from $76.6 billion in 2018. [2] This exponential growth is driven by consumers' increasing desire for convenience, security, and energy efficiency. As homes integrate more devices, the need for a cohesive control system becomes critical.

- The Challenge of Diverse Devices from Different Companies
  As smart home devices proliferate, a significant challenge arises due to the sheer variety of brands and products available. Different companies offer their distinct range of smart devices, each with its dedicated application for control. This leads to an overflow of applications on users' smartphones or tablets, complicating what should be a simplified process. [3][4] The fragmentation of control interfaces creates inefficiency and reduces the user-friendly nature of the smart home system. There's a clear need for a centralized platform, like Matter, that can consolidate control, ensuring that users can manage their entire smart home ecosystem without juggling multiple applications.
  Additionally, there is a potential risk of user's personal information being exposed when remotely controlling products of diverse companies through IoT. Currently, when a user issues a command to turn on the lights from a mobile phone, it often goes through a cloud-based connection between the phone and the hub, and then the operation is executed via an API. [5] With Matter, all operations can be performed locally without passing through external servers, ensuring that personal information is not exposed.

- Bridging the IoT Gap with LG Cover
  While many modern devices are equipped with IoT capabilities, a substantial portion remains without this integration. The disparity between IoT-enabled devices and those lacking this feature can result in gaps within the smart home ecosystem. Matter, a unifying protocol for smart home devices, can introduce IoT capabilities to previously incompatible devices. LG Cover, supporting Matter, acts as a bridge, allowing even non-IoT devices to join the integrated smart home network. This inclusivity ensures that no device is left out, offering users a comprehensive smart home experience.

B. Problem Statement

- Gap in Dedicated Support for LG Devices
  The landscape of smart home devices is increasingly complex and fragmented, with a myriad of brands and protocols available in the market. Within this expansive ecosystem,

while there are many generic hubs available, there is a conspicuous absence of a dedicated hub designed specifically for LG's vast array of devices. Owners of LG products, both modern and legacy, find themselves in need of a hub that is acutely attuned to the nuances of LG's electronic architecture. Generic solutions often fall short in leveraging the full potential of LG devices, leaving users with a less-than-optimal experience.

- Neglect of Legacy IR Devices and Old ThinQ Products
  The tech industry's rapid advancement has led to a situation where many older, yet perfectly functional devices, especially those based on infrared (IR) technology and earlier versions of LG's ThinQ products, are left out of the modern smart home narrative. Most current hubs are squarely focused on the latest communication protocols, sidelining these older devices. This presents a challenge for many households that still rely heavily on these devices and are looking for ways to integrate them into a unified smart home ecosystem.

- Diverse Brand Environment Leading to Fragmented User Experiences
  The average household's smart device portfolio often comprises products from a variety of brands. Current IoT hubs, while claiming broad compatibility, tend to offer a disjointed user experience due to inconsistencies in how different brands' devices are integrated and managed. Users are thus left longing for a singular platform that can offer a seamless and harmonious experience across all their devices.

C. *Solution*

- Tailored Integration for LG's Spectrum of Devices
  The LG Cover stands out with its commitment to providing a seamless experience specifically for LG device owners. Whether it's the latest smart TV or a legacy LG appliance, the LG Cover ensures perfect harmony and optimized performance, something that can only be achieved with a deep understanding of LG's technological DNA.

- Inclusivity for Legacy Devices
  One of LG Cover's hallmark features is its pioneering capability to revitalize older devices using the Matter protocol. Whether it's an IR-based remote-controlled device or an older iteration of the ThinQ product line, the LG Cover, through its integration with Matter, ensures that these devices are not only recognized but also seamlessly incorporated into the modern smart home ecosystem. By leveraging the universal nature of Matter, LG Cover effectively bridges the gap between past and present, preserving the utility and value of legacy devices in today's interconnected world.

- Consistent Experience Across a Multitude of Brands
  Beyond its specialization in integrating LG products, the LG Cover showcases its prowess in providing a consistent and intuitive interface, even when dealing with devices from multiple manufacturers. This focus on providing a unified user experience ensures that managing a diverse smart home setup is no longer a daunting task, but a delightful experience utilizing Matter, it would enable us to incorporate our device to many readily available, existing home management application, such as Apple HomeKit or Google Home App, which is much well suited for corresponding mobile OS.

D. *Related Software*

- Google Nest Hub 2 Gen.
  Nest Hub is the device to use to manage your smart home since it is compatible with so many different things.
  We can watch a range of movies, music, and TV shows using Nest Hub, and we can use voice commands or a single tap to operate connected smart devices. A sleep sensor function can also promote restful sleep.
  Many Nest Hub functionalities are accessible through Quick Gestures. Quick Gestures uses

Motion Sense instead of a camera to identify when our hand moves.

- Samsung SmartThings Station
  Samsung SmartThings Station combines a smart home hub and wireless phone charger while supporting up to three separate programmed automation routines.
  It works with a broad selection of Matter and other Samsung smart home products, and its accompanying app makes it simple to set up routines you can activate from the Station with a tap.
  It is incapable of playing music or supporting voice control because it lacks a speaker and microphone. Instead, this hub serves as a wireless charging station for devices that are Qi compliant.
  Both the Android and iOS platforms currently support Samsung SmartThings Station.

- SwitchBot Hub 2
  SwitchBot Hub 2 supports old infrared appliances, we can link additional SwitchBot goods through Bluetooth and Wi-Fi to create a smart home environment.
  We can integrate Hub 2 into our smart home ecosystem to deploy Matter over Wi-Fi in the future for current Bluetooth-only products like SwitchBot Curtain, SwitchBot Lock, and SwitchBot Blind Tilt, eliminating the need to purchase additional hardware to make them Matter compatible.

- Ikea Dirigera Hub
  DIRIGERA hub is the central control hub for a smart home in the IKEA Home smart app.
  It enables us to connect and control a variety of smart gadgets, including speakers, blinds, and lighting. These devices allow for remote control, the creation of personalized settings for various situations and emotions, and the scheduling of automated chores.
  We can operate your smart home using the app, remotes, voice commands, or motion thanks to the hub's flawless integration into our house's design. It is made to be user-friendly for everyone, including visitors, of all ages.
  IKEA offers a growing selection of smart items to expand our setup over time and updates the app often to improve the smart home experience.

- Bosch Smart Home Controller 2nd Gen.
  Bosch Smart Home Controller II is a crucial part of the Bosch Smart Home ecosystem, serving as both hardware and software hub. It connects to various smart devices like thermostats and cameras and can integrate with third-party products. Users control it through a user-friendly app, web interface, or voice commands via popular voice assistants. It communicates using Wi-Fi, Zigbee, and Z-Wave.
  Security and privacy are a priority with strong encryption. Regular updates enhance performance and security. It's energy-efficient and scales easily, making it the central unit for smart homes, providing control, security, and efficiency.
  Smart Home Controller II ensures the security of our data, even when accessed remotely while we're on the go. We have the flexibility to either control our system exclusively within our home network or remotely to monitor its status.

- Homebridge
  Homebridge is a lightweight NodeJS server that emulates the iOS HomeKit API. It allows users to integrate various smart home devices that do not natively support Apple's HomeKit into the HomeKit ecosystem. Homebridge facilitates a broad array of plugins from various developers, which enables the connection and control of non-HomeKit devices via your iOS devices.
  Being open-source software, Homebridge allows for transparency, community input, and modification. Users can delve into the code, understand how it works, and even contribute to its development. This openness fosters innovation and enables continuous improvement of the platform.
  Its open-source nature, extensive plugin support, and active community make it a unique and invaluable tool for smart home

enthusiasts, especially those invested in the Apple ecosystem.

II. Requirements

A. *Installing Hub*

1. Register LG Cover to ThinQ
   To use non-Matter-supportive devices on home assistants as if they were Matter devices, LG Cover is to be registered to ThinQ app. The information regarding the device control function will be given to LG Cover to generate Matter-supportive QR code.
   Registration procedure is as follows:

   a. In the ThinQ app, press '+' to add new devices. A camera screen that reads QR codes appears.

   b. Above the square frame for QR codes, click 'matter' button to add the LG Cover hub.

   c. Read the QR code on LG Cover's screen to register the hub to ThinQ app.

   d. LG Cover is set to ThinQ app.

2. Register LG Cover to Matter API device
   LG Cover needs to be connected to user's Matter API device to provide information from ThinQ app to home assistants such as Google Home, Apple Home, Alexa, and more. Matter API devices will recognize LG Cover as a Matter device and send or receive signals upon Matter protocols.
   Registration procedure is as follows:

   a. In the home assistant application of the user's device, execute menu for adding new devices.

   b. Among the various ways and brands of devices, select 'Matter' or 'Add Matter device'. A camera screen that reads QR codes appears.

   c. Read the QR code on LG Cover's screen and click "Add to Home Assistant".

   d. LG Cover is set to the user's Matter API devices.

B. *Register device to LG Cover*

Devices registered to ThinQ app can be operated from user's home assistant just like Matter devices via LG Cover. ThinQ and LG Cover is connected, and LG Cover, the bridge hub, mediates ThinQ and Matter API home assistants. Therefore, as users register their devices to Register devices to LG Cover using one of the following three methods according to the supported module for the device.

1. Matter device
   All matter devices use setup code, which is provided on the device, in the packaging, or in an app. Register the Matter device to ThinQ to control device though LG Cover. Matter devices can be registered to the Matter API device through ThinQ or directly to home assistants.
   Registration procedure is as follows:

   a. In the ThinQ app, press '+' to add new devices. A camera screen that reads QR codes appears.

   b. Above the square frame for QR codes, click 'matter' button to add device.

   c. Read the QR code on the device to register the device to ThinQ app.

   d. Click "Connect through NFC" or "Connect through Pin" to connect the device by tapping on the NFC tag or manually entering the setup digits.

   e. Device is registered to ThinQ.

   Registration procedure to home assistant is as follows:

a. Read the QR code on the device.

b. Page will be redirected to registration page for home assistant app.

c. Click “Add to Home Assistant”.

d. Device is registered to Matter API home assistant.

2. ThinQ device
LG appliances that support ThinQ can be registered easily on the app. Devices connected to ThinQ are automatically added to Matter devices via LG Cover acting as a bridge. Register the device to ThinQ to manage the device from home assistants through LG Cover. Registration procedure is as follows:

a. In the ThinQ app, press ‘+’ to add new devices. A camera screen that reads QR codes appears.

b. Read the QR code of the device or find the device from the list of products provided.

c. Device is registered to ThinQ app.

3. Legacy Infrared(IR) device
Infrared supporting devices could or could not be LG’s products. If the device is not in LG’s database, IR signals should be recorded with IR receiver. If the device is LG’s product, IR signals would be listed in LG’s database. In this context, LG Cover must possess the capability to access the database of infrared signals associated with a particular device.
Devices that supports only infrared remote control can be registered to ThinQ and be managed by LG Cover.
Registration procedure is as follows:

a. In the ThinQ app, click ”Add device by IR”.

b. ThinQ requests for IR signals.

c. User sends signals to LG Cover through remote controllers or any kind of IR controllers.

d. The name of the device will be set by smart matching or manually by user.

e. The device is registered to ThinQ.

C. *Integration*

LG cover is able to access ThinQ API via RESTful API. As new devices that do not support Matter protocol are registered to ThinQ app, LG Cover pulls users’s devices from ThinQ. The information can hold the name of the device and list of signals for device operation to be managed in Cover.
LG Cover receives information about the newly registered device. LG Cover is already linked to Matter API devices, or home assistants. Therefore, LG Cover mediates the newly registered device to home assistants and lets Matter API devices to recognize the device as a Matter-supporting device and communicate with the device using matter signals.
The user’s actions to trigger these operations and the corresponding signal flow are as follows:

1. Matter device
Devices that support Matter can be controlled directly from home assistants using Matter API.

2. ThinQ device
Communication from user to device can be processed as following:

a. User sends a Matter signal through home assistant.

b. LG Cover receives the Matter signal.

c. LG Cover delivers the command to ThinQ API by network.

d. ThinQ app receives the command from LG Cover and executes the command.

Communication from device to user can be processed as following:

a. Device sends a signal to ThinQ through network.

b. ThinQ receives the signal from device and LG Cover acquires the data from ThinQ API.

c. LG Cover translates the signal and delivers the data to home assistant by Matter signal.

d. Matter API device, home assistant, receives the data and conveys information.

3. IR device
Communication from user to device can be processed as following:

a. User sends a Matter signal through home assistant.

b. LG Cover receives signal via Matter.

c. LG Cover indicates the Matter signal and transmits the corresponding IR signal for IR device to receive.

D. *Control device using LG Cover*

1. Matter device
Matter devices provide simple and complicated control options available by Cover and home assistants.
Cluster is a group of functions, such as On/Off Cluster or Level Control Cluster. In general, user should be able to use a variety of clusters supported by home assistant, along with manufacturer-specified functions. Among the clusters supported by home assistant, some of the most common ones include On/Off, Open/Close, Color Control, Level Control, Temperature Measurement, and more.
Node encapsulates the full product and controls all endpoints of the particular device. Each endpoint includes clusters, and each cluster holds attributes and events.
Consider a device that belongs to On/Off Light Switch Device Type, and holds two endpoints. Endpoint 1 includes the Server On/Off Cluster, and Endpoint 2 includes the Client On/Off Cluster.
The flow of an On/Off control is as follows:

a. User clicks the On/Off button from home assistant.

b. The Client Cluster controls the Server Cluster.

c. Server Cluster triggers the command to turn on or turn off the switch.

   i. If the OnOff attribute of the Node is “On”, the switch would be turned off.

   ii. If the OnOff attribute of the Node is “Off”, the switch would be turned on.

2. ThinQ device
Complicated control functions of appliances can also be added to Matter API device in forms of shortcuts. Cover enables user selected commands be registered as shortcuts in the device, and user is able to execute the command by just one click.
Simple Control:

a. Turn On / Turn Off
User can turn on and turn off the matter device from home assistants by clicking “On/Off” button from the home assistant app or from user defined shortcuts.

Complicated Control:

a. Refrigerator Temperature
User can check the temperature of each section through home assistants.
User can set the temperature to personal preference through home assistants.

b. Washing Machine
Washing mode, water temperature, number of rinse, and other control functions can be set with home assistants.

3. IR device

a. LG product
LG holds information of all LG products. Therefore, all commands of IR devices are available even if the device does not support ThinQ. When LG IR device is registered to ThinQ through LG Cover, ThinQ can manage several commands of the device with LG Cover for Matter controls.
Simple Control:

i. Turn On / Turn Off
User can turn on and turn off the matter device from home assistants by clicking "On/Off" button from the home assistant app or from user defined shortcuts.

Complicated Control:

i. Air conditioner

A) Wind direction
Direction of the be wind, from top to bottom, left to right, can set through home assistants.

B) Wind temperature
Increase and decrease in wind temperature can be set through home assistants.

C) Operation Mode
Operation mode such as Air-conditioning, dehumidification, and heating can be set through home assistants.

ii. TV

A) Remote Controller
Sound, channels, and other control functions can be set with home assistants.

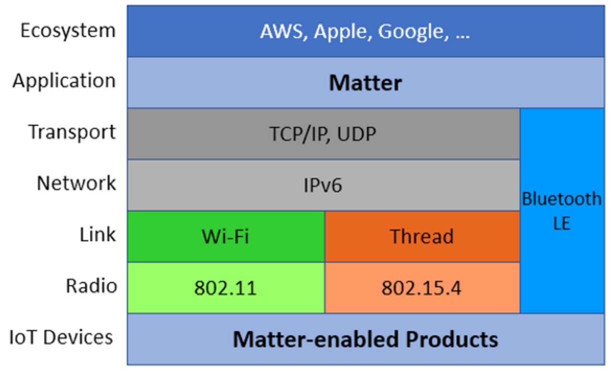
b. Non-LG product
Information of IR devices of other manufacturers is not available by LG. Therefore, when the device is registered to ThinQ through LG Cover, information to be sent is very limited. Only simple control functions can be registered to Cover.

i. Turn On / Turn Off
User can turn on and turn off the matter device from home assistants by clicking "On/Off" button from the home assistant app or from user defined shortcuts.

III. Development Environment

A. *Choice of hardware development environment*

- Consideratoins for Matter [6]



- Requirements for Matter
As you can see from Matter's formal name CHIP (Connected home IP) it runs on existing IP. However, there is bit of difference in PHY layer that it could use Thread running on IEEE 802.15.4 radio instead of WiFi on IEEE 802.11 radio.
For Matter enabled devices and especially a Matter hub that connects with various devices, most likely it would require 802.11, 802.15.4 and BLE connectivity. It would be very cost prohibitive and detrimental to implement various radios one by one with generic off the shelf 8 bit microcontroller such as AVR or PIC, and workload required for matter is questionable for it to run on old, almost deprecated 8bit MCUs of past. While we could use 32 bit microcontroller such as ARM cortex lineup to solve the processor performance

issue, it would still leave the issue of having to implement such radios. [7]
It would be much better off with dedicated SoC with everything needed implemented, ready to go. While designing customized chip is completely outside of this scope, thankfully there already exists a MCU supporting Wifi, Thread, and BLE in one single convenient package; ESP32 series. [8]

- ESP32
  ESP32 is a series of low-cost, low-power system on a chip microcontrollers with integrated Wi-Fi and dual-mode Bluetooth. The ESP32 series employs either a Tensilica Xtensa LX6 microprocessor in both dual-core and single-core variations, Xtensa LX7 dual-core microprocessor or a single-core RISC-V microprocessor and includes built-in antenna switches, RF balun, power amplifier, low-noise receive amplifier, filters, and power-management modules. ESP32 is created and developed by Espressif Systems, a Chinese company based in Shanghai, and is manufactured by TSMC using their 40 nm process. It is a successor to the ESP8266 microcontroller.
  Since the release of the original ESP32, a number of variants have been introduced and announced. They form the ESP32 family of microcontrollers. These chips have different CPUs and capabilities, but all share the same SDK and are largely code-compatible.
  A minimal table to compare the Espressif's MCU families.

- Comparison of ESP32, ESP32-S3, and ESP32-C6

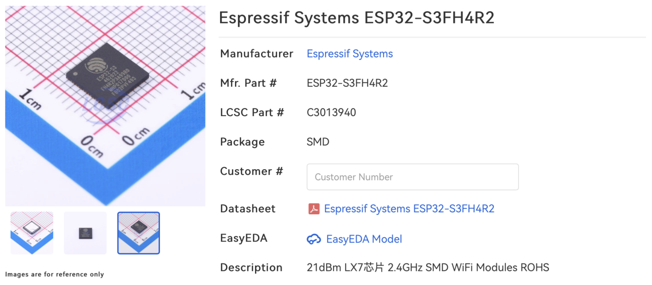
Model	**ESP32**	**ESP32-S3**	**ESP32-C6**
Announcement Date	2016, September	2020, December	2021, April
Main processor	Tensilica Xtensa 32-bit LX6 (up to 240MHz) (optionally dual core)	Tensilica Xtensa 32-bit LX7 dual core (up to 240MHz)	RISC-V 32-bit (up to 160MHz)
SRAM	520KB	512KB	400KB
ROM	448KB	384KB	384KB
JTAG	O	?	O
Cache	64KB	?	?
WiFi	Wi-Fi 4	Wi-Fi 4	Wi-Fi 6
Bluetooth	BLE 4.2 (upgrade to 5.0, with limitations)	BLE 5.0	BLE 5.0
Ethernet	O	?	?
RTC memory	16KB	16KB	?
PMU	O	?	?
ULP coprocessor	O	?	?
Cryptographic Accelerator	SHA, RSA, AES, RNG	SHA, RSA, AES, RNG, HMAC, Digital Signature	SHA, RSA, AES, RNG, HMAC, Digital Signature
Secure boot	O	O	O
Flash encryption	O	O	XTS-AES-128
SPI	4	?	?
I2C	2	?	?
I2S	2	?	?
UART	3	?	?
SDIO Host	1	2	0
SDIO Slave	1	0	0
GPIO	34	44	22
LED PWM	16	?	?
MCPWM	6	2	0
Pulse counter	8	?	X
GDMA*	0	?	?
USB	X	?	?
TWAI**	1	?	?
ADC	2x 12-bit SAR, up to 18 channels	?	?
DAC	2x 8-bit	?	X
RMT	8x transmission/reception	?	?
Timer	4x 64-bit	?	?
Temperature Sensor	O	?	?
Hall Sensor	O	?	?
Touch Sensor	10	?	?

While ESP32-C6 seems like a best choice since it has 802.15.4 capability, It is currently logistically difficult to get a development board. So we had to opt for ESP32-S3, while lacking 802.15.4 support, as current state of 802.15.4 Thread have long way to go before going mainstream we concluded it was not vital.

ESP32-S3 is a dual-core XTensa LX7 MCU, capable of running at 240 MHz. Apart from its 512 KB of internal SRAM, it also comes with integrated 2.4 GHz, 802.11 b/g/n Wi-Fi and Bluetooth 5 (LE) connectivity that provides long-range support. It has 45 programmable GPIOs and supports a rich set of peripherals. ESP32-S3 supports larger, high-speed octal SPI flash, and PSRAM with configurable data and instruction cache. [9]
As we are not just building a virtual docker image, there is some considerations that have to be made.

- ESP32-S3 module
  ESP32-S3 comes in different form factors, but there is mainly two. One being just a bare chip in QFN56 package, and another being with basic components laid out in PCB with antenna and RF shield.

1. ESP32-S3

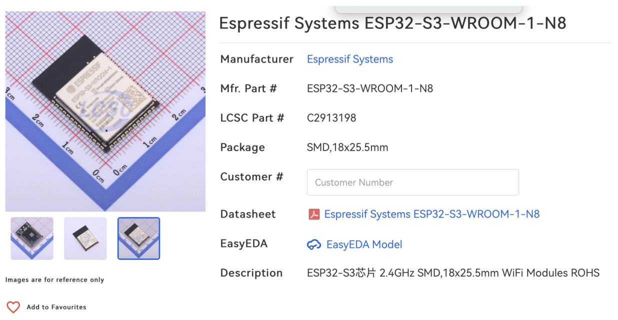


Espressif Systems ESP32-S3FH4R2

Manufacturer	Espressif Systems
Mfr. Part #	ESP32-S3FH4R2
LCSC Part #	C3013940
Package	SMD
Customer #	Customer Number
Datasheet	Espressif Systems ESP32-S3FH4R2
EasyEDA	EasyEDA Model
Description	21dBm LX7芯片 2.4GHz SMD WiFi Modules ROHS

   - very difficult to design RF PCB
   - could lead to sub-optimal design
   - not FCC certified

2. ESP32-S3-WROOM-1



Espressif Systems ESP32-S3-WROOM-1-N8

Manufacturer	Espressif Systems
Mfr. Part #	ESP32-S3-WROOM-1-N8
LCSC Part #	C2913198
Package	SMD,18x25.5mm
Customer #	Customer Number
Datasheet	Espressif Systems ESP32-S3-WROOM-1-N8
EasyEDA	EasyEDA Model
Description	ESP32-S3芯片 2.4GHz SMD,18x25.5mm WiFi Modules ROHS

   - Easier to implement
   - have ready made pcb / rf shield
   - FCC certified
   - easier to meet local RF regulation laws

Ordering Code	Flash	PSRAM
ESP32-S3-WROOM-1-N4	4 MB (Quad SPI)	-
ESP32-S3-WROOM-1-N8	8 MB (Quad SPI)	-
ESP32-S3-WROOM-1-N16	16 MB (Quad SPI)	-
ESP32-S3-WROOM-1-H4	4 MB (Quad SPI)	-
ESP32-S3-WROOM-1-N4R2	4 MB (Quad SPI)	2 MB (Quad SPI)
ESP32-S3-WROOM-1-N8R2	8 MB (Quad SPI)	2 MB (Quad SPI)
ESP32-S3-WROOM-1-N16R2	16 MB (Quad SPI)	2 MB (Quad SPI)
ESP32-S3-WROOM-1-N4R8	4 MB (Quad SPI)	8 MB (Octal SPI)
ESP32-S3-WROOM-1-N8R8	8 MB (Quad SPI)	8 MB (Octal SPI)
ESP32-S3-WROOM-1-N16R8	16 MB (Quad SPI)	8 MB (Octal SPI)

- Wi-Fi + Bluetooth 5 (LE)
  ESP32-S3 supports a 2.4 GHz Wi-Fi (802.11 b/g/n) with 40 MHz of bandwidth support. The Bluetooth Low Energy subsystem supports long range through Coded PHY and advertisement extension. It also supports higher transmission speed and data throughput, with 2 Mbps PHY. Both Wi-Fi and Bluetooth LE have superior RF performance that is maintained even at high temperatures.

- Mature Software Support
  ESP32-S3 is supported through Espressif's popular ESP-IDF platform that already powers millions of devices on the market. ESP-IDF comes with rigorous testing, regular updates and an unparalleled support policy. Based on ESP-IDF's mature software architecture, developers can easily build applications anew or migrate their own applications to the ESP32-S3 platform, and continue working with the trusted ESP-IDF tools and APIs.

- Security
  ESP32-S3 provides all the necessary security requirements for building securely connected devices, without requiring any external components. It supports AES-XTS-based flash encryption, RSA-based secure boot, digital signature and HMAC. ESP32-S3 also has a

"World Controller" peripheral that provides two fully-isolated execution environments, which enables the implementation of a trusted-execution environment or a privilege-separation scheme. [10]

B. *Choice of software development platform*

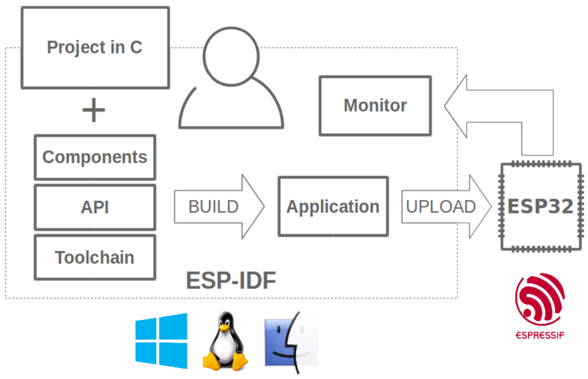
1. Development Platform

   a. Linux
   Linux is based on the Unix operating system which supports a wide range of programming languages, tools, and frameworks. The operating system is used for server environments, desktop applications, and embedded systems development, among others. Linux's command-line utilities are particularly well-suited for working with development boards and IoT devices. Linux is apt for IoT development due to its performance and the extensive community support available for troubleshooting and enhancements.

   b. macOS Monterey
   macOS Monterey is Apple Inc.'s desktop operating system for Macintosh computers. It is compatible with many development tools, languages, and frameworks essential for different kinds of software development, including web development, mobile app development, and hardware programming.

2. Language and Framework

   a. Espressif ESP-IDF [11]
   ESP-IDF is Espressif's official IoT Development Framework for the ESP32, ESP32-S, ESP32-C and ESP32-H series of SoCs. It provides a self-sufficient SDK for any generic application development on these platforms. ESP-IDF currently powers millions of devices in the field, and empowers building a variety of network-connected products, ranging from simple light bulbs to big appliances and industrial devices. ESP-IDF runs on the FreeRTOS operating system, with modifications to better support the SoC's features and capabilities. It uses C as the programming language and provides a suite of development tools for flashing the SoC, monitoring output, and debugging. Various components, component level configuration, and command line support of ESP-IDF enables developers to build their projects at ease. [13]



   b. C/C++
   C is a procedural high-level programming language that is designed to provide efficient access to memory and language constructs that map efficiently to machine instructions. C++ supports both procedural and object-oriented programming paradigms, and provides features such as overloading, templates, and exceptions, which are not available in C. The core components of ESP-IDF and the APIs provided for interacting with the ESP32 hardware are written in C. This allows for efficient interaction with the hardware and is suitable for the resource-constrained environments typically associated with embedded systems. The framework provides support for C++ exceptions and other features, allowing developers to leverage the object-oriented aspects of C++ in their ESP32 projects.

   c. xtensa-esp-elf
   xtensa-esp-elf is a toolchain for 32-bit Xtensa microcontrollers, based on GCC. It provides all the necessary tools to transform high-level code into a binary form that the

microcontroller can execute, handling the specifics of the Xtensa architecture, including its custom instructions and configurations. xtensa-esp-elf includes a collection of binaries such as an assembler, linker, and archiver that prepare the code for execution on the microcontroller. The compiler, extensa-esp-elf-gcc, is the customized version of GCC provided by Espressif, tailored specifically for the Xtensa architecture of ESP32 and ESP32-S series. It allows developers to compile C and C++ code into efficient machine language that can run directly on the low-level hardware. This version of GCC has been optimized to harness the unique features and capabilities of the ESP32's processor, ensuring that developers can maximize performance for a variety of applications, from simple IoT devices to complex networking systems, while leveraging the rich set of features and the robust development framework provided by ESP-IDF. Since the Xtensa cores are different from the architecture of a typical development computer (x86_64 or arm), the xtensa-esp-elf-gcc performs cross-compilation, generating executable code for a different architecture than that of the build host. The xtensa-esp-elf toolchain is versatile and supports development across multiple operating systems, including Linux, macOS, and Windows, thus enabling a wide range of developers to create applications for Espressif's hardware platforms regardless of their preferred development OS. [14]

d. CMake
   CMake is a cross-platform, open-source build system generator widely used for managing the build processes of software in a compiler-independent manner. It enables developers to define build configurations using simple, platform- and compiler-agnostic configuration files, known as makefile. The primary function of CMake is to control the software compilation process using simple platform and compiler-independent configuration files to generate native makefiles or project files that can be used in the compiler environment of your choice. CMake supports out-of-source builds, allowing for multiple build types to be generated from a single source tree and facilitating cleaner build directories and easier maintenance. It is equipped to handle large projects spanning multiple directories and can also manage complex dependency chains between components. With its ability to target various platforms and its extensive scripting capabilities, CMake has become a go-to tool for developers looking to streamline their build processes and improve portability and scalability of their software projects. [15]

e. Ninja
   Ninja is a small build system with a focus on speed. It takes as input the build files generated by tools like CMake and executes the build as efficiently as possible. Unlike CMake, which is a build system generator, Ninja is the actual build system that does the compiling, linking, and other build steps. Ninja aims to outperform other build systems by using simple build files and aggressively caching intermediate build results to minimize redundant work. This design philosophy makes Ninja particularly well-suited for large codebases and for continuous integration systems where build times can significantly affect the development workflow. Although Ninja is not designed to be written by humans directly, its simplicity and high performance have made it an attractive backend for build system generators that prioritize speed and efficiency. [16]

f. React Native
   React Native is an open-source framework developed by Facebook for creating native-style mobile apps using React and JavaScript. It enables developers to build applications for multiple platforms, including iOS and Android, with a single codebase, leveraging a rich ecosystem of libraries and tools. Its component-based structure allows for the reuse of code across different platforms, while maintaining the ability to integrate with

native functionality and third-party libraries. The framework's live and hot reloading capabilities significantly enhance developer productivity, and its large community provides strong support, with contributions extending its reach to platforms like Windows and macOS. Given these benefits, a significant number of developers opt for React Native to create mobile applications that deliver a native user experience with the agility of web app development. [17]

g. Node.js
Node.js is a JavaScript runtime environment that enables server-side execution of JavaScript. It allows for efficient and fast backend development. Its non-blocking, event-driven architecture makes it ideal for handling numerous concurrent processes, which is essential for mobile applications that require real-time data processing and responsiveness. Node.js is particularly beneficial for smartphone app backends due to its ability to manage high volumes of I/O operations, essential for handling multiple user requests simultaneously. This makes it great for real-time apps like chat apps or social media platforms. Additionally, its vast library ecosystem, through NPM, provides numerous tools and modules that simplify adding features and functionalities to mobile backends. This, combined with its unified JavaScript development (for both frontend and backend), streamlines the development process, making it a popular choice for mobile app backend development. [18]

C. *Software in use*

1. LG ThinQ, LG SmartHome Application
The LG ThinQ app is a smart application designed to connect, manage, and communicate with LG smart appliances, providing users with a seamless and intuitive way to ensure their devices operate smoothly. It allows for remote monitoring and control of LG appliances, offering features such as laundry cycle tracking, refrigerator management, and the ability to send cooking instructions to ovens. Integration with voice assistants like Google Assistant and Alexa further enhances the user experience, enabling voice commands for convenience. The app's Smart Diagnosis and proactive alerts system aid in maintaining appliance health, alerting users to potential issues and facilitating troubleshooting. Additionally, ThinQ UP feature keeps appliances up-to-date with the latest software, allowing for personalization and enhanced functionality tailored to users' lifestyles and preferences, underscoring LG's commitment to smart home innovation and user-centric design. [19]

2. VSCode (Visual Studio Code)
Visual Studio Code (VSCode) is a free, open-source code editor developed by Microsoft, renowned for its performance, extensibility, and support for a wide range of programming languages and tools. It combines simplicity with powerful developer tooling, like built-in Git commands, a robust debugging suite, and IntelliSense for smart code completion. Its lightweight design ensures smooth operation, while extensions available through its marketplace allow developers to customize and enhance their coding environment to suit a multitude of workflows. This flexibility, coupled with features like live share for collaborative coding and a built-in terminal, has made VSCode a popular choice among developers across disciplines, from web development to data science and beyond. [20]

3. GitHub
GitHub is a web-based platform that provides hosting for software development version control using Git, facilitating collaborative coding from contributors worldwide. It acts as a repository hosting service, allowing developers to share their code, track changes, and manage versions with robust version control and collaboration features. GitHub extends Git's functionality with its own features, such as issue tracking, feature requests, task management, continuous integration, and wikis for project documentation. It has become an

essential tool for both open-source and private software projects, creating a community that fosters learning, sharing, and collaboration on a global scale. GitHub's impact is substantial, providing a nexus for software development and a repository for a vast array of projects across different fields and technologies.

4. Figma
   Figma is a cloud-based design tool that specializes in user interface and user experience design with real-time collaboration. It stands out for its intuitive interface and vector-based graphics, which are pivotal in modern web and mobile application design. As a browser-based tool, it allows designers to work on projects from anywhere, share their work instantly, and receive feedback dynamically. Teams can simultaneously edit projects, significantly streamlining the design process. Figma supports the entire design journey—from wireframing to prototyping to delivering final assets—all within the same environment. Its component system and design libraries enable consistency across designs, while plugins and integrations enhance its functionality. This all-encompassing solution has made Figma a go-to tool for designers, product teams, and organizations looking to create and iterate on designs efficiently.

5. LaTeX
   LaTeX is a high-quality typesetting system; it includes features designed for the production of technical and scientific documentation. It is the de facto standard for the communication and publication of scientific documents in many fields, including mathematics, computer science, engineering, physics, chemistry, and economics. LaTeX is based on the TeX typesetting system, and it excels at complex documents that contain a lot of mathematics, references, citations, and cross-referencing. Rather than focusing on the appearance of the document directly, it encourages authors to concentrate on its structure by defining commands that describe the logical structure of a document. These commands are compiled to produce a beautifully formatted document in PDF or other formats, with high typographical quality, suitable for academic and professional publication. LaTeX is widely used in academia for the submission of scientific papers and is available as free software.

6. Notion
   Notion is an all-in-one workspace application designed to integrate notes, tasks, wikis, and databases into a single, user-friendly interface, promoting productivity and organization for both personal and professional use. It offers a highly flexible environment where users can create custom pages and databases with rich content types, such as text, images, checklists, and links, and connect these elements in a modular way. Notion's powerful organizational tools, including nested pages, tagging, and sorting, allow for detailed knowledge management and project tracking. Its collaborative features enable real-time teamwork and sharing, making it popular among remote teams, students, and individuals seeking a centralized platform to manage their work and studies. With its sleek design and versatile functionality, Notion stands out as a multifaceted tool that adapts to various workflows and simplifies the integration of diverse information and task management.

D. *Cost*

Name	Price	Quantity
ESP32-S3	3 dollars	1
Laptop	1000 dollars	1
Apple TV 4k	150 dollars	1

IV. Specification

A. *Technical Specifications*

- **Connectivity**: Wi-Fi, NFC, Infrared (IR), and Matter protocol support.

- **Compatibility**: Works with LG and non-LG devices that support ThinQ, Matter, or IR.

- **Security**: Must use secure connections and encrypted communications.
- **APIs**: RESTful API for ThinQ for backend integration; Matter API for device communication.
- **User Interface**: ThinQ app for setup and control; home assistant interfaces for ongoing interaction.
- **Power**: Must support standard household power requirements and include power-saving modes.

1. Hardware Specifications

   a. Microcontroller

   - **Chip**: ESP32-S3
   - **Core**: Dual-core XTensa LX7
   - **Frequency**: 160 MHz
   - **Logic Level**: 3.3V
   - **Operating Voltage Range**: 3.0 to 3.6 V, with an over voltage protection up to 3.9 V for a duration of 10 ms
   - **Low-Power Modes**: Supports Active, Modem Sleep, Light Sleep, Deep Sleep, and Hibernation power modes with wake-up via RTC, GPIO, or other peripherals

   b. Memory

   - **Flash**: 8MB
   - **SRAM**: 512KB
   - **PSRAM**: N/A

   c. Power Management

   - **Supply Voltage**: 5v input voltage / with 3.3v logic level

Parameter	Description	Min	Typ	Max	Unit
VDDA, VDD3P3	Recommended input voltage	3.0	3.3	3.6	V
VDD3P3_RTC	Recommended input voltage	3.0	3.3	3.6	V
VDD_SPI (as input)	—	1.8	3.3	3.6	V
VDD3P3_CPU	Recommended input voltage	3.0	3.3	3.6	V
I_{VDD}	Cumulative input current	0.5	—	—	A

   - **Current Consumption**: less than 1A
   - **Main Power Input**: USB Type-C with a surge protection of up to 20 V
   - **Voltage Regulation**: 5V to 3.3V LDO or Buck Converter, capable of providing a minimum output current of 1 A, with ripple under 50 mV peak-to-peak
   - **Regulator for 5v → 3.3v / AMS1117-3.3**:

     i. **Current**: Up to 800mA output current

     ii. **Dropout Voltage**: 1.1V at 800mA load

     iii. **Package**: Available in SOT-223 and other packages

   - **Power Sequencing**: Engineered for proper power-up and power-down sequences for MCU and peripherals
   - **Power Monitoring**: On-chip ADC to monitor supply voltages and battery levels with interrupt-driven low-voltage warnings

   d. I/O Connectivity

   - **GPIO Pins**: Must allocate pins for IR receive / send

     i. The ESP32-S3 chip features 45 physical GPIO pins (GPIO0 GPIO21 and GPIO26 GPIO48). Each pin can be used as a general-purpose I/O, or be connected to an internal peripheral signal. Through GPIO matrix, IO MUX, and RTC IO MUX, peripheral input signals can be from any GPIO pin, and peripheral output signals can be routed to any GPIO pin.

ii. **Pin usage** [12]:

GPIO	Analog Function	RTC GPIO	Comment
GPIO0		RTC_GPIO0	Strapping pin
GPIO1	ADC1_CH0	RTC_GPIO1	
GPIO2	ADC1_CH1	RTC_GPIO2	
GPIO3	ADC1_CH2	RTC_GPIO3	Strapping pin
GPIO4	ADC1_CH3	RTC_GPIO4	
GPIO5	ADC1_CH4	RTC_GPIO5	
GPIO6	ADC1_CH5	RTC_GPIO6	
GPIO7	ADC1_CH6	RTC_GPIO7	
GPIO8	ADC1_CH7	RTC_GPIO8	
GPIO9	ADC1_CH8	RTC_GPIO9	
GPIO10	ADC1_CH9	RTC_GPIO10	
GPIO11	ADC2_CH0	RTC_GPIO11	
GPIO12	ADC2_CH1	RTC_GPIO12	
GPIO13	ADC2_CH2	RTC_GPIO13	
GPIO14	ADC2_CH3	RTC_GPIO14	
GPIO15	ADC2_CH4	RTC_GPIO15	
GPIO16	ADC2_CH5	RTC_GPIO16	
GPIO17	ADC2_CH6	RTC_GPIO17	
GPIO18	ADC2_CH7	RTC_GPIO18	
GPIO19	ADC2_CH8	RTC_GPIO19	USB-JTAG
GPIO20	ADC2_CH9	RTC_GPIO20	USB-JTAG
GPIO21		RTC_GPIO21	
...	...	...	...
GPIO45			Strapping pin
GPIO46			Strapping pin
GPIO47			
GPIO48			

- **UART**: 3.3V TTL, preset baud rate at 115200 for debug / upload purposes
- **SPI / I2C**: Not used

e. Security

- **Hardware Security**: AES-128/256, SHA, RSA, ECC, RNG
- **Secure Boot**: Enabled
- **Flash Encryption**: Enabled

f. Watchdog Timer (WDT)

- **Type**: Built-in hardware WDT
- **Timeout**: TBD

g. Infrared (IR) Capabilities

- **IR Receiver**:
  i. **Type**: 38 kHz demodulating IR receiver
  ii. **Voltage**: 3.3V compatible
  iii. **Range**: Around 10M depending on the environment. *further testing is needed
  iv. **Compatibility**: Compatible with standard IR remote control protocols (e.g., NEC, RC5)
- **IR Transmitter**:
  i. **LED**: High-powered IR LED
  ii. **Wavelength**: Typ. 940nm
  iii. **Voltage**: 3.3V with current limiting resistor or driven via a high speed MOSFET
  iv. **Range**: Around 10M depending on the environment **further testing is needed*
  v. **Driver Circuit**: Software approach to modulation is CPU intensive; Hardware modulation circuit is recommended

h. IR Interface

- **IR Signal Encoding/Decoding**: Software-based, utilizing available libraries for encoding and decoding IR signals
- **IR Protocols Supported**: e.g., NEC, Sony, Philips
- **Carrier Frequency**: Programmable, to support various consumer electronics

i. Power Management for IR

- **Current Draw**: max. 20mA

j. GPIO Assignments for IR

- **IR Receiver Pin**: TBD
- **IR Transmitter Pin**: TBD

k. Connectivity

- **IR Capabilities**: As previously specified
- **Wi-Fi**:

  i. Complies with IEEE 802.11 b/g/n standards

  ii. **Output Power**: Programmable up to +20 dBm in 1 dBm steps for control of range and power consumption

  iii. **Receiver Sensitivity**: -98 dBm for 802.11 b, DSSS 1Mbps

  iv. External IPEX connector for optional high-gain antenna

- **Bluetooth**:

  i. Bluetooth 5.0 with support for LE, Classic, and High Speed modes

  ii. **Output Power**: Configurable from -20 dBm to +10 dBm

  iii. **Receiver Sensitivity**: -97 dBm for BLE 1 Mbps PHY

  iv. Bluetooth antenna diversity for improved range and reliability

2. Software Specifications

a. Operating System

- **Firmware**: ESP-IDF, FreeRTOS
- **OTA Updates**: Support for Over-The-Air firmware upgrades

b. Matter Protocol Stack

- **Matter Version**: v1.2
- **Supported Features**:

  i. **Device Onboarding**: Implementation of secure onboarding processes compatible with the Matter standard. This includes support for QR code scanning, Bluetooth LE-based introduction, and Wi-Fi configuration transfer.

  ii. **Device Type**: Supports multiple device types as defined by Matter specifications. For instance, the device can be configured as a smart light, a thermostat, a lock, or a sensor, depending on the actual hardware capabilities.

  iii. **Commands and Events**: Ability to handle command and event interactions defined in Matter for device control and automation, such as turning on/off, adjusting parameters, or responding to environment changes.

  iv. **Attribute Reporting**: Implementation of reporting attributes for each endpoint, in accordance with Matter's data model specifications. This allows the device to report its status, such as temperature readings or battery levels, to controllers and other devices.

  v. **Scheduling and Timers**: Support for scheduling actions and setting timers, allowing for automated device behavior such as turning a light on or off at specific times.

vi. **Group Messaging**: Enabling the device to participate in group communication, where a single command can control multiple devices, useful in scenarios like setting scenes or grouped actions.

vii. **Over-The-Air (OTA) Software Updates**: Secure mechanism for updating the device's firmware through the network, ensuring up-to-date functionality and security.

viii. **Interoperability**: Ensures that the device can work with other Matter-enabled devices and ecosystems, such as smart speakers, hubs, or smartphones, regardless of the manufacturer.

ix. **Security**: Compliance with Matter's security requirements, including secure session establishment, end-to-end encryption, and ongoing security maintenance.

c. Network Protocols

- **IPv4/IPv6**: Supported
- **mDNS**: Enabled for local discovery
- **MQTT/HTTP(s)**: For cloud communication (if applicable)

d. BLE Services

- **GATT**: Custom services and characteristics as per application requirements
- **Provisioning**: BLE-based network provisioning

e. Wi-Fi Features

- **Modes**: STA/AP/STA+AP
- **Provisioning**: SoftAP & BLE based methods

f. Security Features

- **End-to-End Encryption**: For all network communication
- **Authentication**: Device attestation and mutual authentication in compliance with Matter specifications

3. Compliance and Certifications

- **Wi-Fi Alliance**
- **Bluetooth SIG**
- **Matter Compliance**: Certified by Connectivity Standards Alliance (CSA)

4. Environmental Conditions

- **Operating Temperature Range**: Industrial grade, from -40°C to +85°C
- **PCB Design**: 4-layer design with impedance matched traces for high-speed signals and thermal management
- **Enclosure**: IP-rated enclosure design compatible for desired level of ingress protection
- **Mounting Options**: PCB designed for surface-mount technology (SMT) and through-hole mounting points for secure enclosure attachment

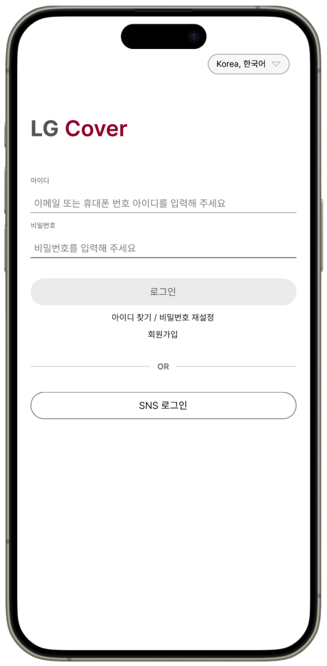
5. Physical Dimensions

- **Size**: TBD
- **Weight**: TBD

6. Development and Debugging
   - **Programming**: Via USB-to-UART Bridge
   - **Debugging**: JTAG/SWD Interface (if applicable)
7. Regulatory Information
   - **FCC ID**: N/A
   - **CE**: N/A
   - **RoHS Compliant**: Yes

B. *Software Specifications*

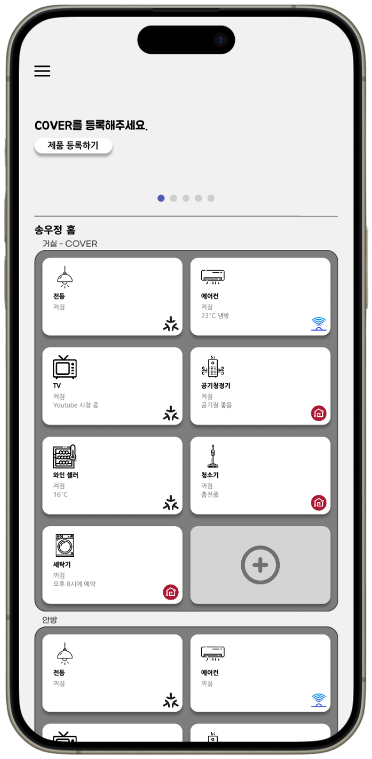
1. Login Page



Upon launching the application, users are greeted by the Login Page, which is the primary gateway for accessing the personalized management features within the app. The Login Page is essential as it ensures secure access for users to control and interact with their devices through the COVER application.
To begin, users are prompted to enter their username and password. For new users or those without login credentials, the page provides a series of features: 'Find ID / Reset Password,' 'Sign up,' and 'SNS login' options. The 'Sign up' process is a critical step in gaining access to the app's functionalities, leading users through a streamlined process to create their account.
A top right button on the Login Page allows users to customize their experience by selecting their country and language, ensuring that the app meets their regional preferences and language understanding.
When users complete the signup and hit the 'Register' button, the system initiates a Register Post API request to the server. Successful completion of this request redirects the user to the Login Page, where they can now log in with their new credentials.
The input fields during the signup process have specific constraints to ensure data integrity and user identification. For instance, the name field requires 2 to 5 characters and accepts input directly via the keyboard. The username field permits 5 to 20 characters, allowing only English letters and numbers, while the password field demands 8 to 20 characters, also limited to English letters and numbers, with a re-entry required for confirmation.

2. Main Page
   Upon logging in, the Main Page serves as the user's command center, presenting a comprehensive view of their registered devices within the application. The layout and functionality of this page are tailored to facilitate interaction with the user's smart home ecosystem.

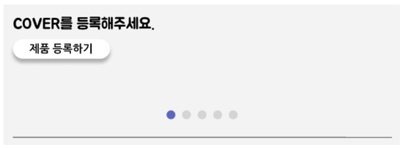
   a. Main Page Overview

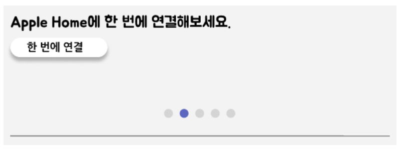


The Main Page features:

- A display of all the user's registered devices, organized for quick navigation and management.
- Options to manage device settings, monitor device status, add new devices, and link devices to home assistants like Apple Home.

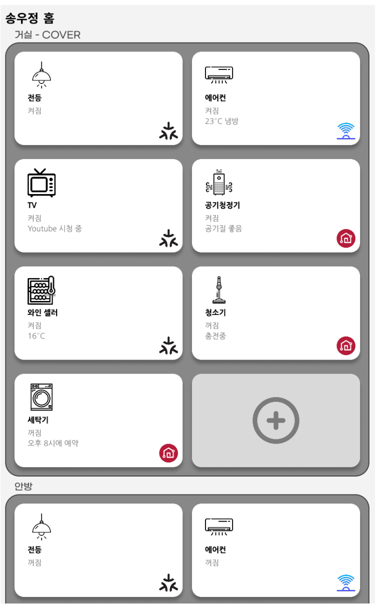
b. Top Navigation Bar (Top Nav)





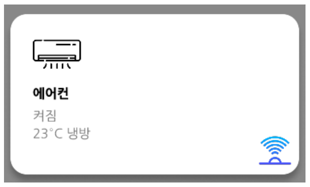
- The Top Nav, accessible by swiping horizontally, houses multiple features for device interaction.
- The 'Register Product' menu item leads users to the Device Registration Page, enabling the addition of new devices to the system.
- Additional menu items, revealed by swiping, include shortcuts for quickly connecting all COVER-registered devices to home assistants.

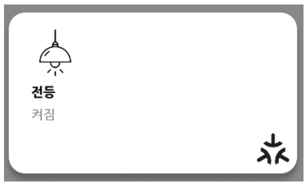
c. Devide Organization

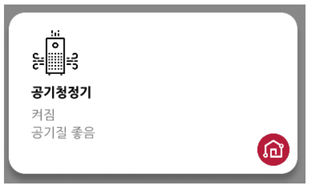


- Devices are displayed in a grid layout below the Top Nav, categorized by their physical location (e.g., 'living room') or the hub they are connected to.
- Each device is represented by a tile that includes an icon indicating the device type, the device name, and its current status (e.g., on/off).
- The '+' button at the end of each device category facilitates the registration of new devices to specific locations.

d. Device Tiles







Device tiles provide at-a-glance information:

- The upper left corner of each tile features a device-specific icon.
- The device name is located just below the icon.
- The current status of the device is displayed, with the available statuses varying by device type.
- The bottom right corner of each tile indicates the device's connection type (Matter, LG ThinQ, or IR), each with a corresponding icon.

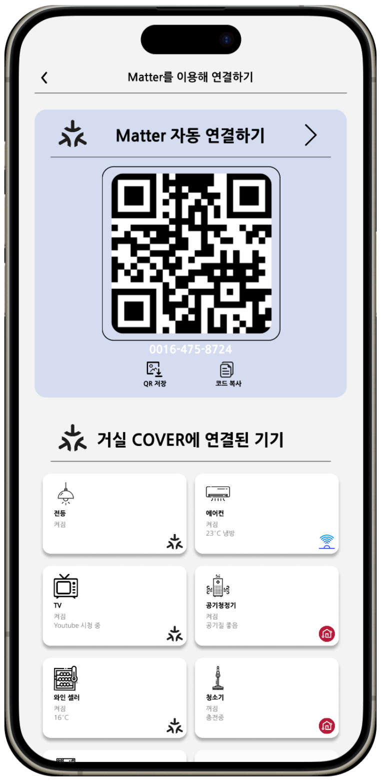
e. Interaction and Design

- The Main Page is designed to be intuitive and user-friendly, allowing for efficient management of the user's smart home devices.
- The interface is clean and organized, ensuring that users can easily find and operate their devices.

Device Type	Status
Light	On/Off
	Brightness
	Color temperature
	Reservation information
Air Conditioner	On/Off
	Temperature
	Mode
	Reservation information
TV	On/Off
	Currently playing content
Air Purifier	On/Off
	Air quality
	Reservation information
Wine Cellar	On/Off
	Temperature
Vacuum Cleaner	On/Off
	Charging
Washing Machine	On/Off
	Reservation information

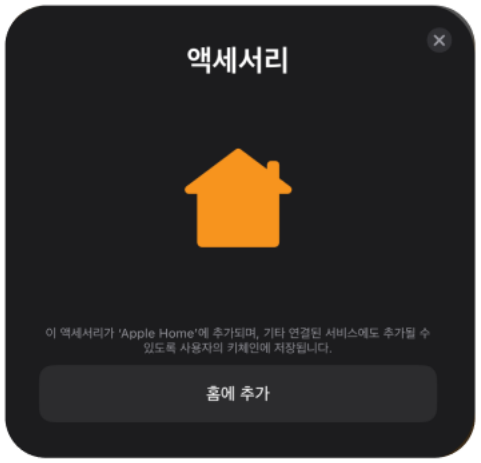
- The design is consistent with the application's overall aesthetic, providing a seamless user experience.

3. Bind with Matter Page



Upon accessing the Bind with Matter Page

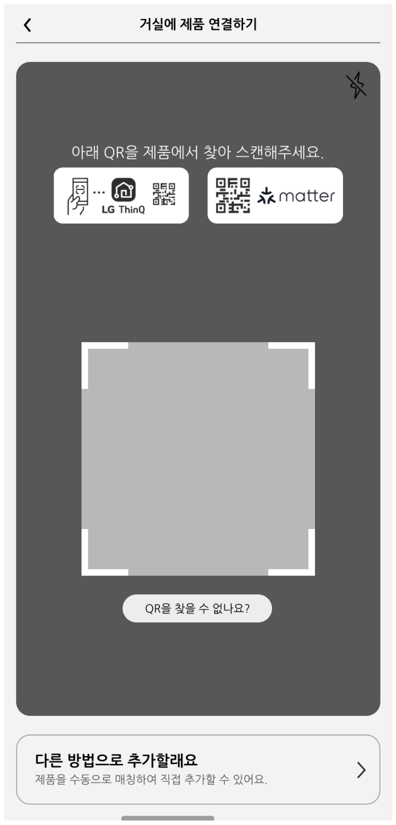
within the application, users are presented with a streamlined process to integrate their COVER-managed devices with Apple Home, harnessing the power of the Matter protocol. This page is designed to be a straightforward junction point between COVER and Apple Home, with a couple of pivotal actions available for the user to initiate the integration.
Users can secure their setup by using the 'QR 저장' button to save the QR code, a crucial step that anchors the device to their account. Alternatively, they can use the '코드 복사' button to copy the COVER setup code, allowing for a manual connection if needed. These options cater to different user preferences, ensuring versatility in the setup process.
Advancing in the setup, the 'Matter 자동연결하기' button is a significant feature on this page. With a simple press, users command the application to proceed with the automatic registration of COVER to Apple Home. This button effectively initiates a seamless transition to Apple Home's accessory addition page, where the previously registered devices on COVER become part of the Apple Home ecosystem, ready for centralized control.
However, users must be aware of certain limitations inherent to the system's design. Notably, when connecting COVER with Apple Home through Matter, there is a cap on the number of devices that can be managed. The system currently supports a maximum of 30 devices linked to COVER. This limit is set to ensure that each device maintains a robust and reliable connection within the smart home network. Attempting to add more than 30 devices could lead to potential performance issues, affecting both the new and existing devices' functionality.
This limitation emphasizes the importance of strategic planning in the user's smart home setup. For those users who wish to exceed the 30-device threshold, exploring additional configurations, such as incorporating another hub, may be necessary to distribute the network load and maintain system integrity.
In conclusion, the Bind with Matter Page serves as a gateway for users to expand and centralize their smart home control within Apple Home. While designed for ease of use and convenience, it operates within the defined device limitation to ensure optimal performance across the smart home network. Users should navigate this page with an understanding of these parameters to fully leverage the capabilities of their COVER and Apple Home integrated experience.



이 액세서리가 'Apple Home'에 추가되며, 기타 연결된 서비스에도 추가될 수 있도록 사용자의 키체인에 저장됩니다.

4. Device Registration Page
   The Device Registration Page acts as the gateway for incorporating new devices into the user's smart home setup, featuring a methodical and comprehensive approach for adding devices via various protocols like Matter, LG ThinQ, and Infrared (IR).

   a. Matter and LG ThinQ Device Registration



아래 QR을 제품에서 찾아 스캔해주세요.

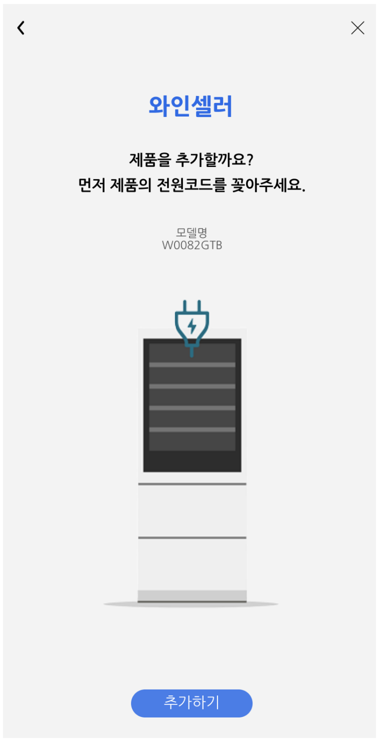
다른 방법으로 추가할래요

제품을 수동으로 매칭하여 직접 추가할 수 있어요.

Upon landing on the Device Registration Page, users are prompted to register their Matter or LG ThinQ devices by scanning a QR code. The page activates the camera on the user's mobile device, allowing for the QR code to be read seamlessly. The system is intuitive enough to distinguish between a Matter QR code and an LG ThinQ QR code:

- **For Matter Devices**: The system, upon recognizing a Matter device's QR code, proceeds to automatically add the device to the user's profile, integrating it into the smart home network without additional steps.

- **For LG ThinQ Devices**: If the QR code corresponds to an LG ThinQ device, the user is redirected to a specialized LG ThinQ Registration Page. This page not only displays the device's specifics, such as its type and model name, but also an image for visual confirmation. Here, the user can finalize the registration by pressing the '추가하기' button. This action triggers the system to store all relevant information, including control commands and status features, within LG's comprehensive database. Subsequently, COVER gains access to these details, allowing the user to manage the device via the COVER interface.



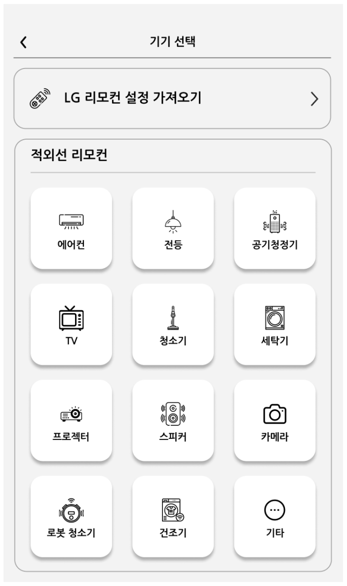
제품을 추가할까요?

먼저 제품의 전원코드를 꽂아주세요.

모델명

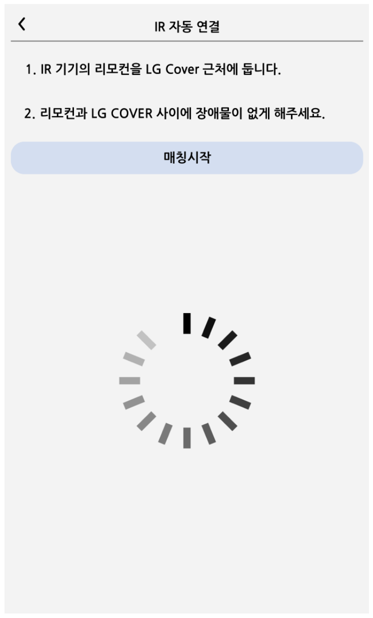
W0082GTB

b. IR Device Registration



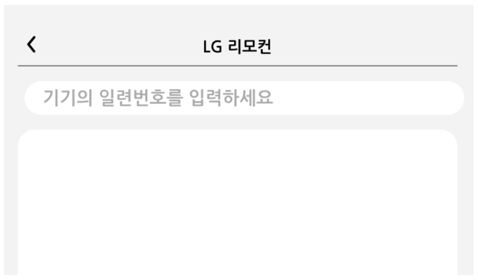
For devices that operate via IR, especially those that do not support QR code scanning, the registration process is facilitated through an alternative method:

- **For LG IR Devices**: Users can initiate the setup by clicking the ‘다른 방법으로 추가할래요’ (Add in another way) button, which redirects them to the IR Registration Page. If the IR device is manufactured by LG, users can fetch the remote control settings from LG’s database by pressing the ‘LG 리모컨 설정 가져오기’ (Retrieve LG remote control settings) button. The system then presents a list of devices matching the entered serial number from which the user can select the appropriate one.

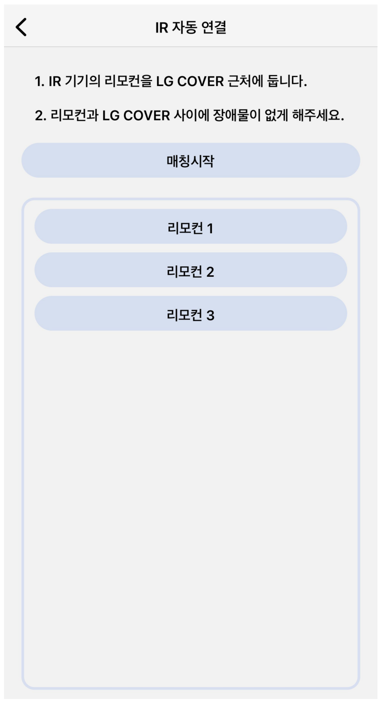


1. IR 기기의 리모컨을 LG Cover 근처에 둡니다.

2. 리모컨과 LG COVER 사이에 장애물이 없게 해주세요.



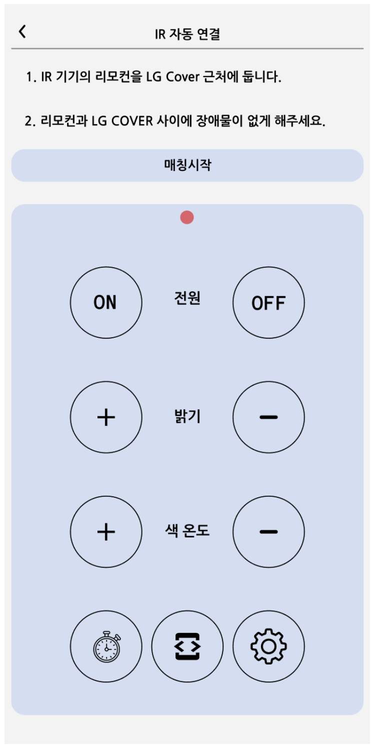
- **For Non-LG IR Devices**: When dealing with non-LG IR devices, the process requires the user to manually transmit the IR signal from the device’s remote control to COVER. This is done by selecting the type of device they wish to register and following the prompts to transmit each function’s corresponding signal. After the user selects the device type and transmits a signal to the COVER with the remote control, COVER retrieves the information of the remote control matching the received signal.



1. IR 기기의 리모컨을 LG COVER 근처에 둡니다.

2. 리모컨과 LG COVER 사이에 장애물이 없게 해주세요.

COVER will provide a list of remote controls that seem to be compatible with the device the user wants to register. Once the user selects the provided remote control, a remote control interface appears, displaying various control options supported by that remote.


IR 자동 연결

1. IR 기기의 리모컨을 LG Cover 근처에 둡니다.

2. 리모컨과 LG COVER 사이에 장애물이 없게 해주세요.

There can be various control options depending on the device, such as power, brightness, temperature, and mode. When a button on the remote control displayed on the cover is pressed, the corresponding infrared signal is emitted from the cover. If the signal is correct, the device to be registered will respond. If the currently displayed remote control, or its control signal, is deemed correct, the user presses the 'Registration Complete' button visible at the top of the remote. The cover stores both the type of the device and the information of the remote control, allowing the user to operate the device from the cover using the control signals of the selected remote control.

C. *User Documentation*

1. Compliance with Matter standards.

2. Secure QR code generation and scanning.

3. Robust error handling for failed device additions or communications.

4. Thorough testing with various brands and types of devices to ensure compatibility.

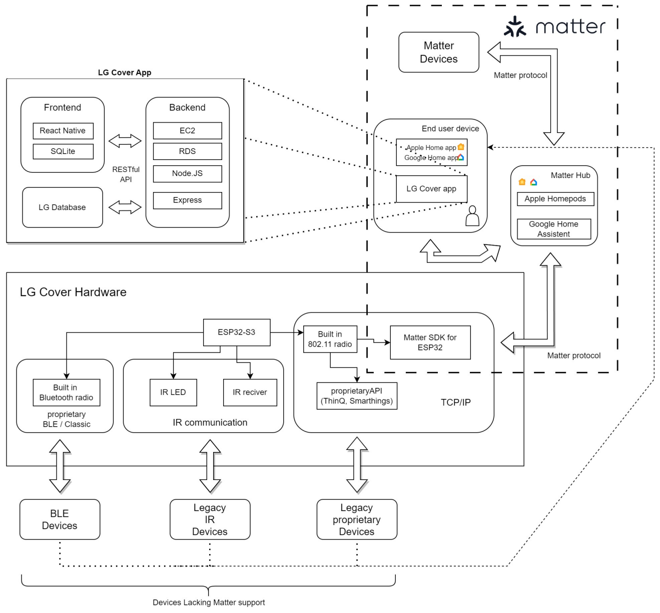
D. *Quality Assurance*

1. Detailed setup guide for LG Cover and device registration.

2. Troubleshooting tips for common issues during setup or use.

3. Contact information for support.

V. Architecture Design



A. *Overall Architecture*

The overall software architecture consists of three parts; frontend, backend and embedded systems. The first module is embedded systems; LG cover device. The main purpose of this device is to bridge "legacy devices" to matter environment. The device has Matter support, showing up as a matter device in the network. Implemented using ESP32 as hardware platform, the LG cover device enables seamless integration of legacy devices into the Matter environment by acting as a bridge. It ensures that these legacy devices can communicate and interact with other Matter devices on the network, providing a unified and interoperable ecosystem. Additionally, the embedded system of the LG cover device is responsible for managing and controlling the device's functionalities, allowing users to conveniently control and monitor their legacy devices through the frontend and backend interfaces.

The second module is the frontend application, designed with React Native, that interacts with the users. LG Cover application enables users to set up and configure devices with LG Cover. This app not only facilitates the management and control of LG ThinQ and LG IR devices but also extends its compatibility to include non-LG IR devices. By simply registering their devices with the LG Cover, users can operate even those devices that do not natively support Matter as if they were Matter-compatible, using their smartphones. Furthermore, this app integrates with existing Matter hubs like Google Home or Apple Homekit. This comprehensive system empowers users to seamlessly integrate all their connected devices, whether they are Matter-native or not, into the Matter environment, offering a unified and streamlined smart home experience.

The third module of the system is the backend, which has been developed using Node.js. This backend server plays a crucial role in managing app settings, custom user data, and an IR remote device database, as well as facilitating Over The Air (OTA) firmware updates for the LG cover device. In this system, the backend server is responsible for storing user information, including the control information of registered devices. This enables the frontend to communicate with the backend, allowing users to view and manage the control status of their devices through the user interface. Additionally, the backend server retrieves remote control signals for LG's depreciated legacy IR devices from LG's database. This feature enables users to operate these LG infrared devices through the LG Cover as if they were using Matter, obviating the need for physical remotes and thus significantly elevating the convenience and accessibility of the smart home system. To encapsulate, the backend server facilitates effortless maintenance and updates for the LG cover device. The backend server plays a pivotal role in ensuring the app's smooth and efficient functioning by adeptly managing all essential background tasks and overseeing data management processes.

B. *Module 1: Frontend*

1. Purpose
   React Native was used to build Cover mobile app, which is an open-source framework for building mobile applications. It enables developers to create apps that are natively rendered for both iOS and Android platforms using JavaScript and React, a popular library among web developers. It utilizes native components to ensure high performance and a consistent look and feel across different devices. Additionally, Expo, a framework and platform that sits on top of React Native, makes the process of developing and deploying React Native apps even simpler. Expo's notable features include live reloading, which allows developers to see changes in real-time, and the ability to push updates to apps quickly. The application stores the user's device information, and it will serve as a gateway for connecting various protocol devices with the home assistant of user's mobile device.

2. Functionality
   Users can utilize all the features of the Cover through the frontend of Cover application. Users can register and log in through the Cover to gain access to its features. On the main page visible after logging in, users can see their registered devices and navigate to a page to register new devices. ThinQ devices, Matter supportive devices, and depreciated infrared devices can be registered. Additionally, from the main page, users can turn devices on and off by touching the device, displayed as a block with supplementary information. When user presses the device block long enough, all controllable features of the pressed device is displayed, and users are able to operate unique features supported by each device. The functionality to register the Cover with Apple Home or Google Home can also be executed from the main page, thereby enabling the control of all devices connected to the Cover through a home assistant. Users are then

able to control all devices, from power to air temperature, from their home assistants.

3. Location of Source Code
   https://github.com/HYU-SE-COVER/Cover-FE

Directory	**File Name**	**Modules Used**
COVER-FE/COVER	App.js app.json babel.config.js package-lock.json package.json react-native.config.js	Frontend (React Native)
COVER-FE/COVER/app	_layout.js Home.js index.js	Frontend (React Native)
COVER-FE/COVER/app /auth	CreateAccount.js FindIDpwd.js Login.js SNSLogin.js	Frontend (React Native)
COVER-FE/COVER/app /register	Applehome.js IRremote.js RegisterDevice.js RegisterIR.js RegisterLGdb.js RegisterMatter.js RegisterThinQ.js	Frontend (React Native)
COVER-FE/COVER/app /components	ControlModal.js DeviceBlock.js GetURL.js MenuBtn.js NavigationSwiper.js	Frontend (React Native)
COVER-FE/COVER/assets /images	ir.png ir2.png logoImages.png matter.png matter2.png matterQR.png qrcode.png thing.png thingQR.png	
COVER-FE/COVER/assets /images/etc	copyCode.png option1.png option2.png option3.png saveQR.png	
COVER-FE/COVER/assets /images/devices	airconditioner.png airpurifier.png camera.png dryer.png else.png light.png plus.png projector.png remotecontroller.png robotcleaner.png speaker.png tv.png vaccumcleaner.png washingmachine.png winecellar.png winecellar2.png	

4. Class Component

   a. app folder
      This folder, along with the components folder, houses all the fundamental screens that constitute the application.

   b. index.js
      This file acts as the main entry point for the entire app. It displays the LG Cover logo and, upon completion of loading, redirects the user to a page where they can log in.

   c. Home.js
      This file is the core screen of the LG Cover application. All the features provided by the LG Cover application are accessible from the Home screen. It includes a menu button and a navigation slider, enabling users to register devices with the Cover and link the Cover to Apple Home. Additionally, all devices registered with the Cover are displayed, sorted by their location in the space. Users can view the current status of all registered devices at a glance and modify their states as needed.

   d. auth folder
      This folder contains all components related to user authentication.

   e. Login.js
      In this file, when a user enters their email and password to log in, and if their credentials are correct, they are granted access to the Home screen. This enables users to enjoy a personalized experience with the LG Cover.

   f. register folder
      This folder contains all components related

to adding new devices to the LG Cover. It includes functionalities for registering various types of devices with the LG Cover system.

g. RegisterDevice.js
This file is the screen for adding new devices to the LG Cover. It features a QR code reader capable of recognizing QR codes from ThinQ or Matter devices. After QR code recognition, the file redirects to the appropriate device addition page. For IR devices, a button at the bottom of the screen allows navigation to a separate page for addition.

h. RegisterIR.js
This file is a screen dedicated to registering infrared devices with the LG Cover. If the device is a LG's IR device, a button at the top of the screen leads to a different screen to find a matching device. For non-LG IR devices, the user can select appropriate options based on the type of device to store and register the infrared signals.

i. IRremote.js
This file is a screen used when the device to be registered with the LG Cover is a non-LG IR device. The Cover receives and interprets infrared signals sent from the device's remote control, suggesting a suitable remote control form to the user. The user selects the appropriate remote control and then checks if the control options supported by the remote are functioning properly. If the control signals are correctly applied to the device, the remote can be registered as is. Later, the device can be operated using the control signals of that remote control.

j. RegisterLGdb.js
This file id a screen utilized when the device to be registered is an LG IR device. Entering the device's serial number at the top of the screen prompts the LG database to provide remote control information, which is then stored for future use.

k. RegisterMatter.js
This file is a screen for registering Matter devices with the LG Cover. If the QR code read in the RegisterDevice screen belongs to a Matter device, it retrieves the device information for user confirmation before finalizing the registration.

l. RegisterThinQ.js
This file is a screen designed for registering LG ThinQ devices with the LG Cover. If the QR code scanned in the RegisterDevice screen is identified as a ThinQ device, it fetches the device information for verification before completing the registration process.

m. Applehome.js
This screen facilitates the registration of the LG Cover with Apple Home. It displays devices already registered with the Cover and, upon registration with Apple Home, allows control of all these devices from within the Apple Home environment.

n. components folder
This folder contains components that are used across various folders through imports, as well as components that require separate management. It serves as a central repository for shared and specialized components within the application.

o. NavigationSwiper.js
This file is a swiper component used on the top of the home screen, allowing users to navigate left and right. It facilitates navigation to pages where users can add devices to the Cover or register the Cover with Apple Home.

p. MenuBtn.js
This file is a menu button located at the top of the home screen. It provides users access to a variety of functions supported by the application.

q. GetURL.js
This file is utilized in multiple screens of the application to provide the base URL for requesting or sending information to the

backend.

r. DeviceBlock.js
This file is a block displayed on the home screen or the screen for registering the Cover with Apple Home. It displays information about the devices, including the device's name, power status, current state, and the type of protocol used.

s. ControlModal.js
This file is a remote control modal that allows control of various features of a device on the home screen. When a DeviceBlock is pressed and held on the home screen, this control modal appears, enabling control over settings like temperature and mode, in addition to power.

C. *Module 2: Backend*

1. Purpose
The primary purpose of the backend in this application is to store user registration information and manage the device information registered by each member. It stores the information sent from the frontend in the database, and extracts the requested information from the database to send it back in the required format. User login information is encrypted and stored to prevent the leakage of personal information. The backend stores registered devices, protocol information of each device, types of control options supported by each device, control commands or signals for each control option, the current status of the device, and device control scheduling information. The backend is built using Node.js and Express, and the database utilizes SQLite. Express is a streamlined and adaptable framework for Node.js, offering a comprehensive array of functionalities for both web and mobile apps. It comes equipped with an extensive range of HTTP tools and middleware options, facilitating the rapid and straightforward development of powerful APIs. SQLite is a library programmed in C, providing a compact, speedy, autonomous, highly reliable, and fully-equipped SQL database engine. SQLite is integrated into many mobile phone and is included in a wide variety of everyday applications used by many people.

2. Functionality
The backend enables all the functions used by the user in the frontend. When a user registers, their login information is stored in the backend's database, and when they attempt to log in, their registration information stored in the database is verified to grant access to the home page. As the home page loads, the frontend is provided with the user's stored device information from the database, allowing the frontend to display the user's registered device information. When a user operates a registered device, the varied device information is automatically saved back to the database. When a user registers a new device, details such as the device's protocol information, operable options, and control methods like control signals are stored. If the device registered by the user is a ThinQ device or an LG infrared device, it may also access LG's database. When a ThinQ device is registered, its QR code is scanned and all control options and APIs for that device are automatically retrieved from LG's device database. If the device being registered is not a ThinQ but an LG device that supports infrared, LG's database can be accessed to retrieve remote control signal information, as LG has signal information for all its devices. Therefore, if the product is from LG, the user does not need to register separate control options. However, if the device to be registered is not an LG product but supports infrared, the infrared control signals for that product must be separately stored in the database.

3. Location of Source Code
https://github.com/HYU-SE-COVER/Cover-BE

4. Class Component

a. routes/auth
This route is the hub of all user

Directory	File Name	Modules Used
COVER-BE/COVER	app.js package-lock.json package.json	express body-parser
COVER-BE/COVER/data	database.js	express fs path uuid
COVER-BE/COVER/routes	cover.js auth.js	express bcryptjs

authentication-related interactions, encompassing processes like membership registration and login. When a user signs up, their credentials are securely added to the database, with a particular focus on encrypting passwords to prevent any breach of personal information. During the login process, the system first verifies the existence of the entered email in the database. If the email is found, the route then compares the password associated with the account against the one entered by the user. Upon a successful match, the user is granted access, ensuring a secure and reliable authentication process. This route is pivotal in maintaining the integrity and security of user data, acting as a gatekeeper to authenticate user identities and permissions. By meticulously handling sensitive information and validating user credentials, it ensures a protected environment for users to interact with the application.

b. routes/cover
This endpoint is the cornerstone of device registration and management for the user's entire suite of devices. It handles all aspects of device interaction within the application, meticulously processing and responding to requests for device information. This ensures that the application can display device data accurately and timely. Moreover, when a user modifies device settings, such as power status or other options through the application, these changes are sent as requests to the endpoint. The endpoint then diligently updates these alterations in the database, maintaining a synchronized state between the application and the backend. This dynamic interchange facilitates real-time updates and robust device management, making it an essential component of the smart home ecosystem. This route exemplifies a seamless integration of front-end requests with backend processing, ensuring a smooth and efficient user experience.

D. *Embedded Systems*

1. Purpose
The core purpose of our project is to develop an embedded software solution tailored specifically for the ESP32-S3 MCU. The ESP32-S3 was chosen due to its sophisticated integration of multiple communication protocols, addressing the unique requirements of Matter (Connected Home over IP, CHIP) technology. Matter, as its name suggests, operates over IP networks. It can utilizes traditional Wi-Fi on IEEE 802.11 radio. This necessitates a approach in software development to fully exploit the MCU's capabilities.
In the context of Matter-enabled devices, and more so for a Matter hub that connects a multitude of devices, the demand for compute power with 802.11, BLE (Bluetooth Low Energy) connectivity is paramount. The challenge lies in integrating these various radios efficiently. Using generic, off-the-shelf 8-bit microcontrollers like AVR or PIC is not only cost-prohibitive but also technologically limiting due to their outdated architecture and inability to handle the workload required by Matter. While shifting to a 32-bit microcontroller like the ARM Cortex lineup could address the processor performance issue, it still leaves the challenge of integrating the necessary radios.
Our solution capitalizes on the ESP32 series, an SoC (System on Chip) that inherently supports Wi-Fi and BLE in a single, convenient package. This approach not only simplifies the hardware design but also provides a robust platform for our software to interface with these varied communication protocols effectively.

2. Functionality
Our embedded software is a tailored solution designed to exploit the full potential of the LG cover device, enhanced by the ESP32-S3 MCU's capabilities. The software's primary function is to manage the device's Matter capabilities, ensuring seamless operation over IP networks with support for BLE and Wi-Fi (IEEE 802.11). This is crucial for enabling the device to act as a versatile hub in Matter-enabled ecosystems.
Additionally, the software is responsible for handling IR communications, ensuring that the device can interact with a variety of legacy systems and devices. The integration of REST API connections is another critical function, enabling the device to communicate with web-based services and applications. BLE support is also a key feature, allowing for low-energy, short-range communication with a plethora of BLE-enabled devices.
Developed using C++, our software leverages the ESP-IDF SDK for optimized performance on the ESP32 series. The inclusion of FreeRTOS provides the necessary real-time operating capabilities, essential for handling concurrent tasks and managing hardware resources efficiently. The PlatformIO wrapper adds an extra layer of functionality, streamlining the development process and enhancing the software's compatibility with various development environments. Finally, the Matter SDK provided by Espreiff is integrated to ensure full compliance and interoperability with the Matter protocol, solidifying our device's position in the interconnected world of smart home devices.

3. Development Environment Setup
The software is almost exclusively written in C++ with platformIO and an esp32 Arduino core. esp32-arduino-matter-builder, along with Matter SDK and ESP-Matter, was used to build the required environment for development.
Rather than developing directly on ESP-IDF, esp-32 Arduino core was used as a wrapping layer to enable support for readily available libraries in the Arduino ecosystem. This approach allowed for faster development and easier integration of existing Arduino libraries into the project. Additionally, using platformIO provided a convenient and efficient way to manage dependencies and build the software for the ESP32 platform. One of these libraries is IRremoteESP8266, which offers high-level components for IR communications.
Some codes related to Matter required C++17 support, requireing extra setup.
Following are external resources used in the project.

Used Resources	**Git repo**
connectedhomeip	https://github.com/project-chip/connectedhomeip
ESP-Matter	https://github.com/espressif/esp-matter
ESP Arduino Matter	https://github.com/Yacubane/esp32-arduino-matter
IRremoteESP8266	https://github.com/crankyoldgit/IRremoteESP8266
Arduino Core for ESP32	https://github.com/espressif/arduino-esp32

4. Source Code
https://github.com/HYU-SE-COVER/esp32_main

Directory	**File Name**
exp32_main/src/	main.cpp
exp32_main/.pio/ESP32 Arduino Matter/src/	Matter.h
exp32_main/	platformio.ini

5. Class Component

   a. main.cpp
   This is the main script that runs on MCU. It includes the setup and loop functions, as well as any additional functions or variables needed for the project. The main.cpp file is responsible for initializing the necessary components and handling the overall logic of the project. Additionally, it may include code for interacting with external IR peripherals connected to the MCU. This is entire code runs everything. It uses various libaries mentioned above.

VI. Use Cases

A. *Use Case 1: Turn On the Application*

When the user taps the Cover app icon, the app opens displaying the phrase "LG COVER", and the app is launched.

B. *Use Case 2: Sign In*

After registering, the user logs in. The login screen features the LG Cover logo along with Made for matter, ThinQ, and IR logos. Below, there are fields to enter the user's ID and password for login. The ID can be the user's email or phone number. Options for ID recovery, password reset, and SNS login are also available. The login proceeds when the user enters the correct ID and password and presses the login button.

C. *Use Case 3: Register LG Cover*

The user registers their purchased LG Cover in the Cover app. This is done through the "Register Product" button on the main screen.

D. *Use Case 4: Register Device to LG Cover*

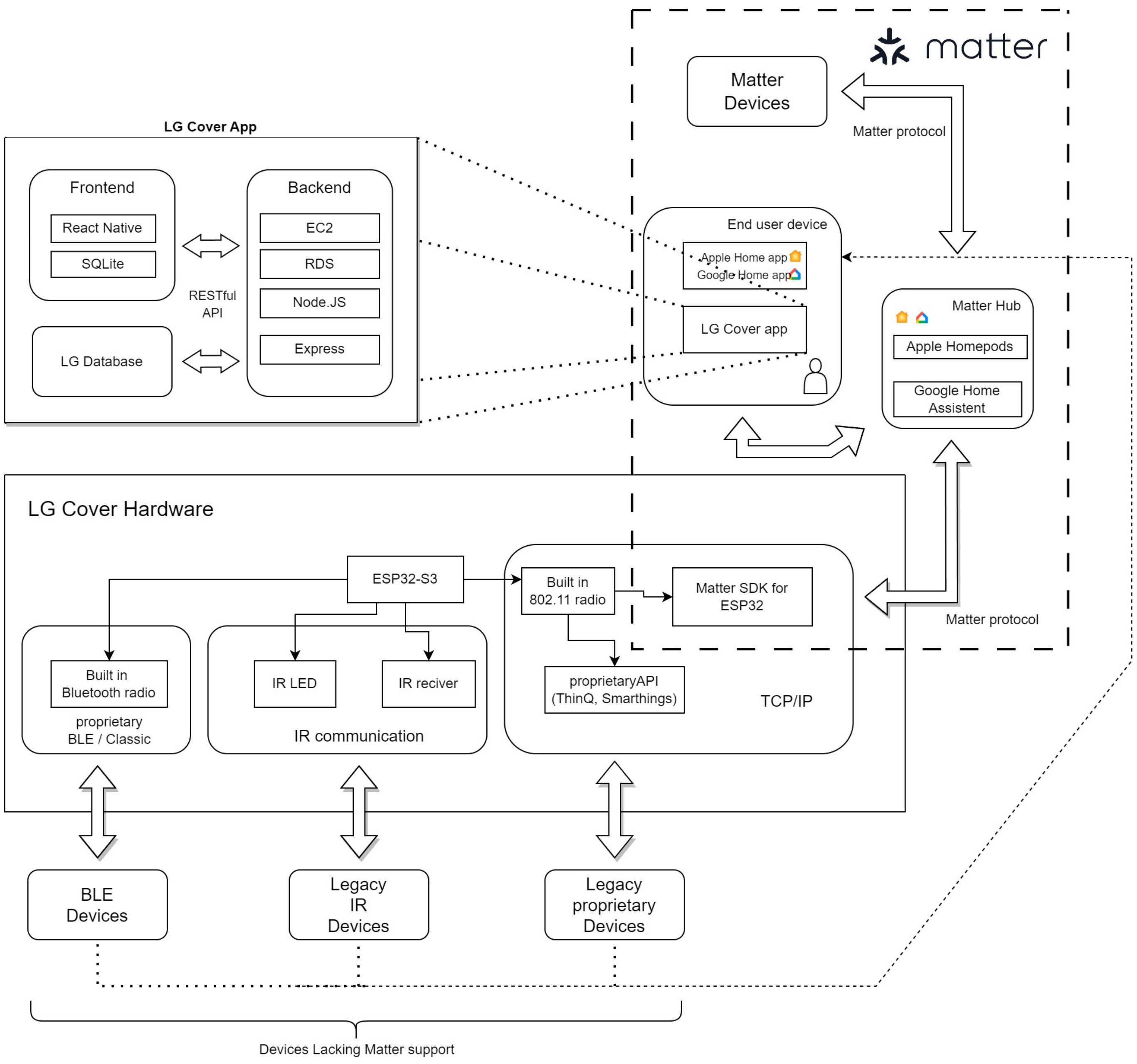
To add a device to Cover, the user presses the + button on the main screen, which leads to a screen for scanning QR codes. Here, the camera can recognize the ThinQ connection QR code or Matter connection QR code of the device to be added. Once the QR code is recognized, the device is added to the main screen. If the device does not have a QR code, the user presses the "Can't find QR?" button. If the additional device does not have a QR code, it is based on IR, thus the user is directed to the IR remote control settings screen.

E. *Use Case 5: Control Device*

If the added device supports Matter or ThinQ, the user clicks its icon on the main screen. Through the control panel that appears next, the device can be operated. If the device does not support Matter or ThinQ, the following steps are taken:
If the device to be controlled is an LG product, the user presses the "Import LG Remote Control Settings" button. Pressing this button leads to a serial number input page. Entering the device's serial number allows the user to control it with the corresponding remote control. If the product is not from LG, the user selects the type of device (e.g., air conditioner, lights, air purifier) from various buttons. Pressing the button leads to the 'IR Auto Connection' screen. The "Start Matching" button detects nearby IR device remotes for pairing, and successful pairing allows the use of the remote control in the Cover app.

F. *Use Case 6: Integration*

The user can register multiple devices in LG Cover to Apple Home for collective connection. From the main screen, pressing the "Connect All at Once" button leads to the 'Connect via Matter' screen. This screen displays a list of devices connected via QR codes. Scanning the QR code with a phone allows all connected devices to be registered on Apple Home.


matter
Matter Devices
Matter protocol
LG Cover App
Frontend
React Native
SQLite
Backend
EC2
RDS
Node.JS
Express
RESTful API
LG Database
End user device
Apple Home app
Google Home app
LG Cover app
Matter Hub
Apple Homepods
Google Home Assistent
LG Cover Hardware
ESP32-S3
Built in 802.11 radio
Matter SDK for ESP32
proprietaryAPI (ThinQ, Smarthings)
TCP/IP
Matter protocol
Built in Bluetooth radio
proprietary BLE / Classic
IR LED
IR reciver
IR communication
BLE Devices
Legacy IR Devices
Legacy proprietary Devices
Devices Lacking Matter support

REFERENCES

[1] Smart Home Market Revenue Trends and growth drivers - 2023. MarketsandMarkets. https://www.marketsandmarkets.com/Market-Reports/smart-homes-and-assisted-living-advanced-technologie-and-global-market-121.html?gclid=EAIaIQobChMImsWklrrwgQMV62sPAh2Blg4VEAAY ASAAEgLVQfD_BwE

[2] 2019 Digital America State of the U.S. Consumer Electronics Industry. https://cdn.cta.tech/cta/media/media/resources/i3/pdfs/digital-america-2019.pdf

[3] Ding, J., Nemati, M., Ranaweera, C., & Choi, J. (2020). IoT Connectivity Technologies and Applications: A Survey. IEEE Access, 8, 67646-67673.

[4] Shah, S.K., & Mahmood, W. (2020). Smart Home Automation Using IOT and its Low Cost Implementation. International Journal of Engineering and Manufacturing, 10, 28-36.

[5] Iqbal, W., Abbas, H., Daneshmand, M., Rauf, B., & Bangash, Y.A. (2020). An In-Depth Analysis of IoT Security Requirements, Challenges, and Their Countermeasures via Software-Defined Security. IEEE Internet of Things Journal, 7, 10250-10276.

[6] Moorhead, P. (2022, January 11). CES 2022: Matter And Thread Win The IoT Connectivity Wars. Forbes. https://www.forbes.com/sites/moorinsights/2022/01/11/ces-2022-matter-and-thread-win-the-iot-connectivity-wars/?sh=8777e0a19b1e

[7] https://csa-iot.org/wp-content/uploads/2022/11/22-27349-001_Matter-1.0-Core-Specification.pdf

[8] Connectivity Standards Alliance. (2023). Matter Application Cluster Specification (Version 1.1) [PDF]. Accepted for release by the Connectivity Standards Alliance Board of Directors on May 17, 2023. https://www.csa-iot.org

[9] Espressif Systems. (n.d.). SoCs. Retrieved November 10, 2023, from https://www.espressif.com/en/products/socs

[10] Espressif Systems. (2023). ESP32-S3 Series Datasheet: 2.4 GHz Wi-Fi + Bluetooth® LE SoC. Retrieved November 10, 2023, from https://www.espressif.com/sites/default/files/documentation/esp32-s3_datasheet_en.pdf

[11] Programming Guide - ESP32. Espressif's SDK for Matter latest documentation. (n.d.). https://docs.espressif.com/projects/esp-matter/en/latest/esp32/

[12] Espressif Systems, "ESP32-S3 Technical Reference Manual", https://www.espressif.com/sites/default/files/documentation/esp32_techn ical_reference_manual_en.pdf

[13] Espressif Systems. (2023). esp-idf: Espressif IoT Development Framework [Software]. GitHub. Retrieved November 10, 2023, from https://github.com/espressif/esp-idf

[14] Espressif Systems. (n.d.). About - ESP-IDF Programming Guide. Retrieved November 9, 2023, from https://docs.espressif.com/projects/esp-idf/en/latest/esp32s3/about.html

[15] Kitware. (2023). The standard build system - CMake. Retrieved November 9, 2023, from https://cmake.org/features/

[16] Ninja. (n.d.). [Home page]. Ninja-build.org. Retrieved November 9, 2023, from https://ninja-build.org/

[17] React Native. (2023). [Home page]. Reactnative.dev. Retrieved November 9, 2023, from https://reactnative.dev/

[18] Node.js. (n.d.). [Home page]. nodejs.org. Retrieved December 2, 2023, from https://nodejs.org/en/about

[19] LG. (2023). LG ThinQ. Retrieved November 9, 2023, from https://www.lg.com/us/lg-thinq

[20] Microsoft. (2023). [Home page]. Visual Studio Code. Retrieved November 9, 2023, from https://code.visualstudio.com/