

**Software Design Specifications**

**SiLabs Door Lock**

|  |  |
| --- | --- |
| **Version**  **Status**  **Date** | 1.1  Draft  01-Sept-2020 |

**Confidentiality Notice**

Copyright (c) 2020 eInfochips. - All rights reserved

This document is authored by eInfochips and is eInfochips intellectual property, including the copyrights in all countries in the world. This document is provided under a license to use only with all other rights, including ownership rights, being retained by eInfochips. This file may not be distributed, copied, or reproduced in any manner, electronic or otherwise, without the express written consent of eInfochips.

Contents

[**1.** **Document Details** 4](#_Toc46347688)

[**1.1** **Revision & Approval History** 4](#_Toc46347689)

[**1.2** **Definition, Acronyms and Abbreviations** 4](#_Toc46347690)

[**1.3** **References** 6](#_Toc46347691)

[**2.** **Introduction** 7](#_Toc46347692)

[**2.1** **Purpose of the Document** 7](#_Toc46347693)

[**2.2** **Intended Audience** 7](#_Toc46347694)

[**2.3** **Scope of Design** 7](#_Toc46347695)

[**3.** **System Overview** 8](#_Toc46347696)

[**3.1** **Top Level System Diagram** 8](#_Toc46347697)

[**3.2** **Description** 9](#_Toc46347698)

[**4.** **SYSTEM ARCHITECTURE** 10](#_Toc46347699)

[**4.1** **Software Architecture** 10](#_Toc46347700)

[**4.1.1** **Firmware & Software** 10](#_Toc46347701)

[**4.1.1.1** **User enrolment** 11](#_Toc46347702)

[**4.1.1.2** **1 step authentication** 13](#_Toc46347703)

[**4.1.1.3** **2-step authentication** 13](#_Toc46347704)

[**4.1.1.4** **Switching between 1-step and 2-step authentication** 13](#_Toc46347705)

[**4.1.1.5** **Mode of authentication** 13](#_Toc46347706)

[**4.1.1.6** **OTA update** 14](#_Toc46347707)

[**4.1.1.7** **Power Management** 15](#_Toc46347708)

[**4.1.2** **Mobile Application** 17](#_Toc46347709)

[**4.2** **Hardware Architecture** 17](#_Toc46347710)

[**4.3** **Architecture Diagram** 18](#_Toc46347711)

[**4.4** **Communication Architecture** 19](#_Toc46347712)

[**4.5** **Resource Consumption Details** 19](#_Toc46347713)

[**4.5.1 Resource consumptions for mobile application development environment** 19](#_Toc46347714)

[**5.** **Integration Details** 20](#_Toc46347715)

[**5.1** **Integration Sequence** 20](#_Toc46347716)

[**5.2** **Integration Procedure** 20](#_Toc46347717)

[**6.** **Tools for Architecture and Design** 21](#_Toc46347718)

[**6.1** **Software Tools** 21](#_Toc46347719)

[**6.1.1** **List of Software Items** 21](#_Toc46347720)

[**6.1.2** **List of SOUP Items** 21](#_Toc46347721)

[**6.2** **Hardware Tools** 21](#_Toc46347722)

[**7.** **Detailed Design** 22](#_Toc46347723)

[**7.1** **Software Detailed Design** 22](#_Toc46347724)

[**7.1.1** **Firmware & Software** 22](#_Toc46347725)

[**7.1.1.1** **Face Recognition module** 22](#_Toc46347726)

[**7.1.1.2** **Fingerprint Sensor** 26](#_Toc46347727)

[**7.1.1.3** **Keypad Module** 29](#_Toc46347728)

[**7.1.2** **Mobile Application** 30](#_Toc46347729)

[**7.1.2.1** **Bluetooth connection with EFR32 Boards** 30](#_Toc46347730)

[**7.1.2.2** **Obtaining the initial device parameters** 31](#_Toc46347731)

[**7.1.2.3** **Communication with EFR32** 33](#_Toc46347732)

[**7.1.2.4** **Authentication for the lock/unlock request** 34](#_Toc46347733)

[**7.1.2.5** **Notifications on mobile app** 35](#_Toc46347734)

[**7.1.2.6** **OTA Update Provisioning** 36](#_Toc46347735)

[**8.** **Data Design** 38](#_Toc46347736)

[**8.1** **UART Communication low level api for the face and finger authentication module:** 38](#_Toc46347737)

[**8.2** **I2C low level api for the keypad authentication module** 38](#_Toc46347738)

[**9.** **External Interface Design** 40](#_Toc46347739)

[**10.** **Human Machine / User Interface** 41](#_Toc46347740)

[**10.1** **ZigBee Verification:** 41](#_Toc46347741)

[**10.2** **Interface Design Rules** 41](#_Toc46347742)

[**10.3** **Inputs** 41](#_Toc46347743)

[**10.3.1** **Mobile App inputs** 41](#_Toc46347744)

[**10.4** **Outputs** 41](#_Toc46347745)

[**10.4.1** **Mobile App inputs** 41](#_Toc46347746)

[**10.5** **Navigational Hierarchy** 41](#_Toc46347747)

[**10.5.1** **Mobile App Screen** 43](#_Toc46347748)

[**11.** **Constraints** 46](#_Toc46347749)

[**12.** **Risk Analysis Output** 47](#_Toc46347750)

[**13.** **Appendix** 48](#_Toc46347751)

# **Document Details**

## **Revision & Approval History**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Version** | **Author** | | **Reviewer** | | **Approver** | |
| **Name** | **Date**  **(DD-MMM-YYYY)** | **Name** | **Date**  **(DD-MMM-YYYY)** | **Name** | **Date**  **(DD-MMM-YYYY)** |
| Draft 0.1 | Manish Thakur,  Hardik Kumar Patel,  Prassanna Sakore, Vipul Kole | 02-Jul-20 | Shachi Mahajan,  Archana Benur | *-* | *-* | *-* |
| Draft 0.2 | Manish Thakur,  Hardik Kumar Patel,  Prassanna Sakore, Vipul Kole | 07-Jul-20 | Shachi Mahajan,  Archana Benur,  Ben Klingelhefer,  Jan Schnepp |  |  |  |
| Draft 0.3 | Manish Thakur,  Hardik Kumar Patel,  Prassanna Sakore, Vipul Kole | 14-Jul-20 | Shachi Mahajan |  |  |  |
| Draft 0.4 | Manish Thakur,  Hardik Kumar Patel,  Prassanna Sakore, Vipul Kole | 22-Jul-20 | Ben Klingelhefer,  Jan Schnepp |  | Ben Klingelhefer | 23-Jul-20 |
| Baseline 1.0 | Manish Thakur,  Hardik Kumar Patel,  Prassanna Sakore, Vipul Kole | 23/07/20 |  |  |  |  |
| Draft 1.1 | Prassanna Sakore | 01/09/20 | Manish Thakur | 01/09/20 |  |  |

|  |  |
| --- | --- |
| **Version** | **Description Of Changes** |
|
| Draft 0.1 | Created initial draft version |
| Draft 0.2 | Incorporated review comments by Archana and Shachi |
| Draft 0.3 | Updated ZigBee as backend networking protocol |
| Draft 0.4 | Incorporated review comments by Ben |
| Baseline 1.0 | Approved received from customer |
| Draft 1.1 | Updated software architecture diagram and enrolment and authentication figure |

## **Definition, Acronyms and Abbreviations**

|  |  |
| --- | --- |
| Definition/Acronym/Abbreviation | Description |
| SRS | Software requirement specification |
| UUID | Universalunique identifier |
|  |  |
|  |  |

## **References**

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Document** | **Version** | **Remarks** |
| 1 | door\_lock\_system\_SRS\_draft\_02 | 0.2 | SRS draft |
| 2 | R307\_fingerprint\_module\_user\_manual.pdf | NA | Fingerprint user manual |
| 3 | TTP229 user manual | NA | Keypad user manual |
| 5 | <https://omronfs.omron.com/en_US/ecb/products/pdf/en-b5t.pdf> | NA | Omron B5T specification |

# **Introduction**

## **Purpose of the Document**

The main purpose of this document is to

* Describe detail design of how door lock application will function according to defined requirement.
* Defines the Software and Firmware architecture, modules, sub-modules, and interface between EFR32MG21 and input/output device.
* Describes communication between EFR32MG21 and android mobile app over Bluetooth.

## **Intended Audience**

Intended audience of this document is developers, project team members, testers, and all stakeholders.

## **Scope of Design**

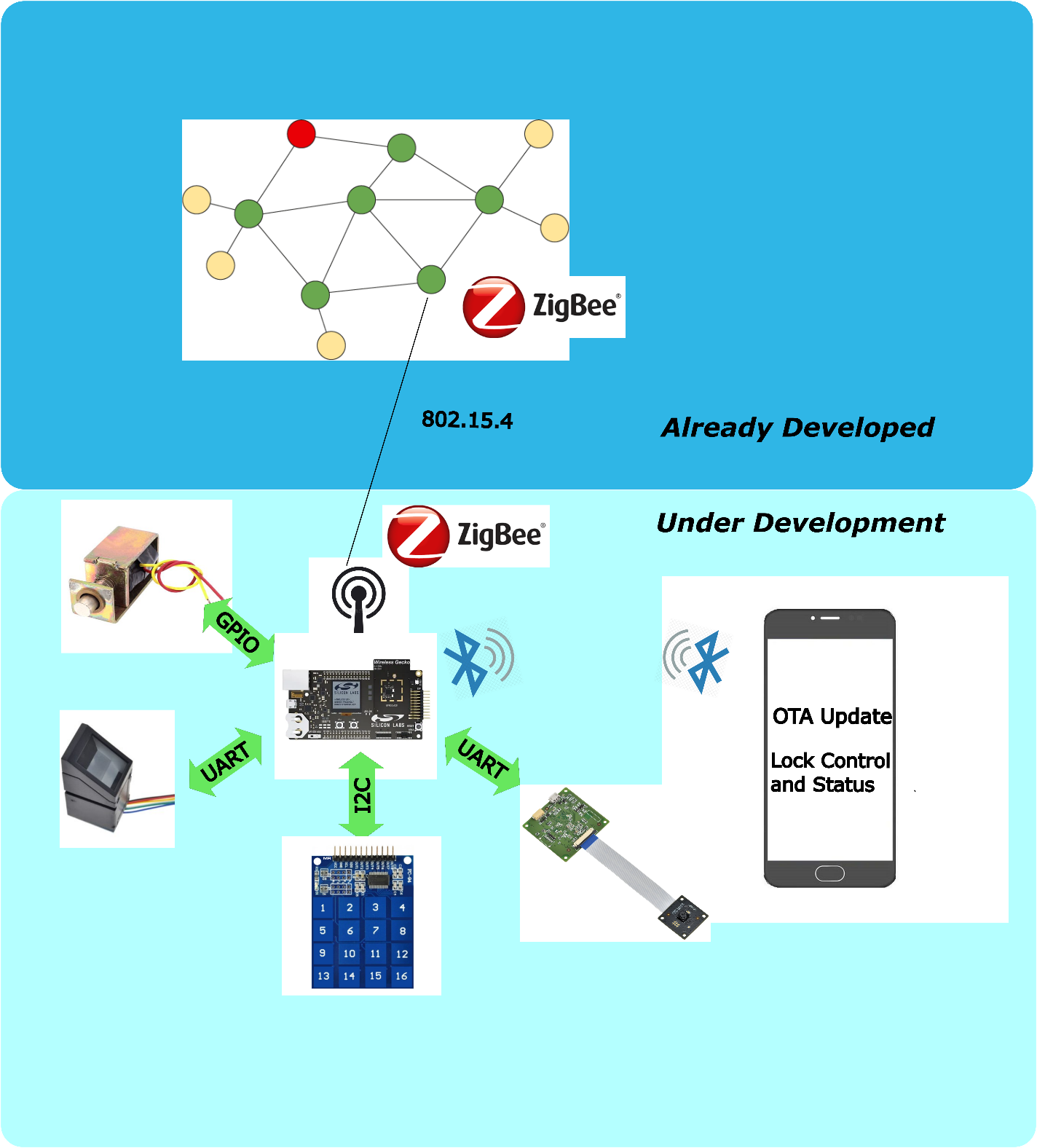
This design document covers detail design of door lock application workings, assembling the different kind of input devices such as Fingerprint sensor, keypad input device and face recognition device to interface with EFR32MG21 dev kit. This includes Firmware design and development of standalone android application that fulfills requirements covered in requirement specifications. Below are the two SDK’s will be used for the development:

• EmberZNet SDK

• Bluetooth SDK

# **System Overview**

## **Top Level System Diagram**

**

**Fig 01: Top Level System Diagram**

## **Description**

Door lock application based on EFR32MG21 application layer framework is used to achieve the smart locking of home, building and commercial space. EFR32MG21 supports dynamic multiprotocol which allows ZigBee and BLE to work concurrently. Where ZigBee will be used as backend networking and BLE will be used for mobile connectivity. This reference platform provides locking and unlocking of door using authentication. It has 1 step and 2-step authentication to open/close the door.

1-step authentication is using:

* Fingerprint sensor authentication
* Keypad input
* Face recognition

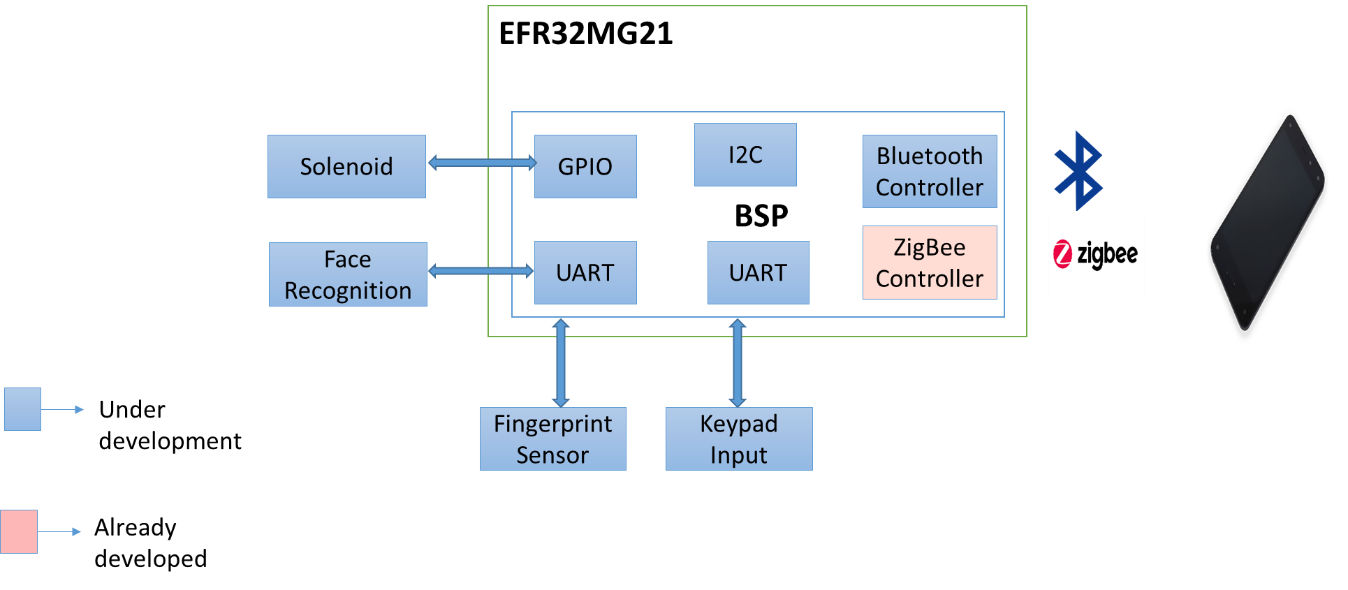
2-step authentication is using:

* Keypad input and Face recognition
* Keypad input and Fingerprint match

Mobile application is used in the application to control or monitor the status of the door lock. Also, OTA update functionality is supported to download the firmware updates to the EFR board.

# **SYSTEM ARCHITECTURE**

## **Software Architecture**



**Fig 02: Software Architecture Diagram**

### **Firmware & Software**

Firmware and software include the driver of I2C, UART based protocol which will enable to communicate with input devices (fingerprint sensors, keypad input, face recognition). The above-mentioned drivers will be integrated with the SiLabs provided BSP. Below is the requirement which we are going to cover in this design document:

|  |  |
| --- | --- |
| **Respective Requirement ID** | **Requirement** |
| RQ01 | Implementation of EFR32 lock example application |
| RQ02 | User enrollment |
| RQ03 | Face recognition enrollment |
| RQ04 | Fingerprint enrollment |
| RQ05 | Pass-key enrollment |
| RQ06 | 1-step authentication |
| RQ07 | 2-step authentication |
| RQ08 | Switching between 1-step and 2-step authentication |
| RQ09 | Mode of authentication |
| RQ10 | Fingerprint authentication use case |
| RQ11 | Keypad authentication use case |
| RQ12 | Face recognition use case |
| RQ13 | OTA update |
| RQ14 | Power consumption |
| RQ15 | Verifying ZigBee control side |

|  |  |
| --- | --- |
| **Software Components** | **Description** |
| **Programming Language** | C and C++ |
| **Development Tool** | Simplicity Studio IDE (version 4) |
| **SDK/BSP/OS** | GEEKO/Silabs SDK Bluetooth, Simplicity studio component |

#### **User enrolment**

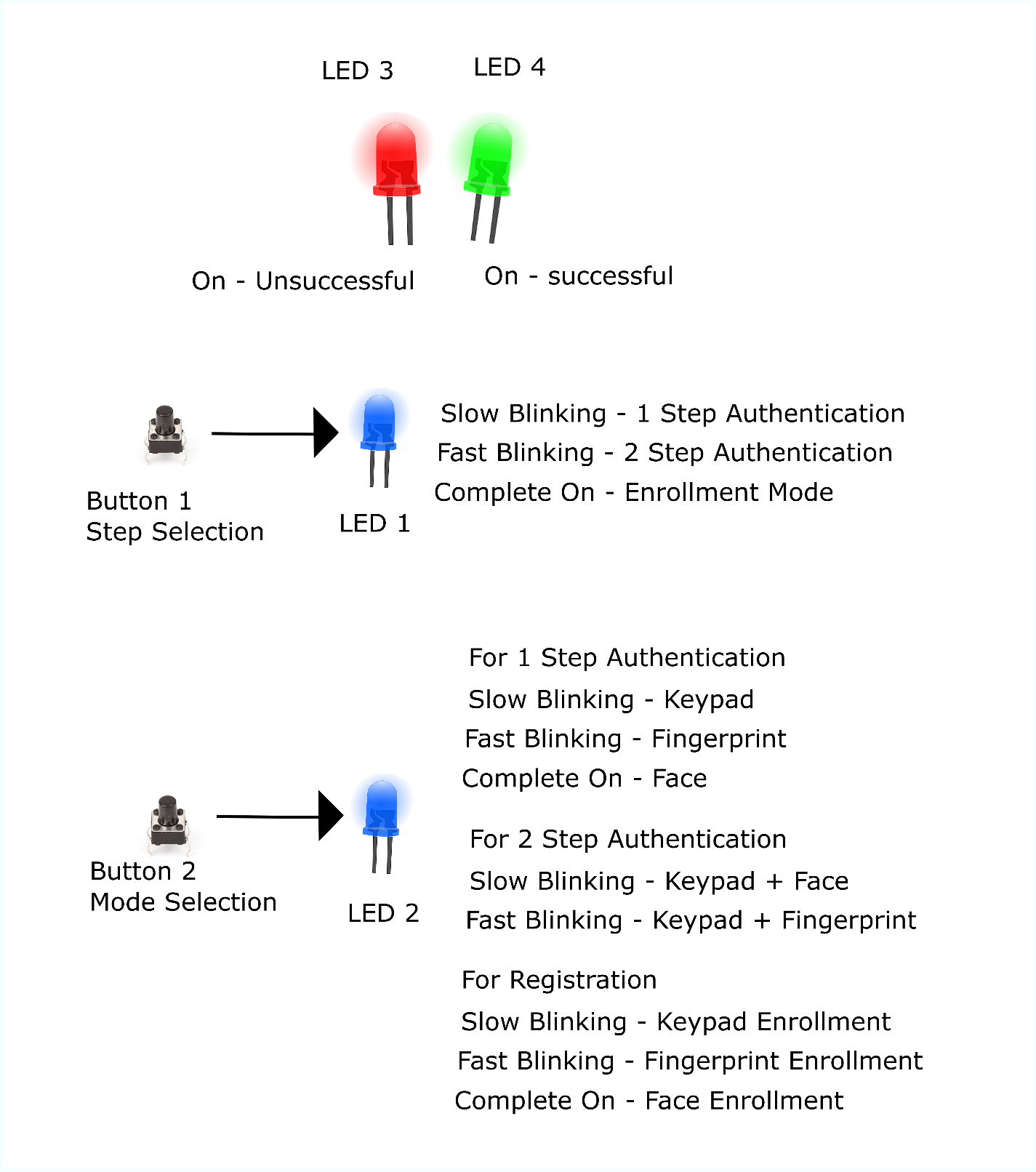
Door Lock application will have user enrolment process for all the input devices such as face recognition, fingerprint sensor and keypad input device. Details of user enrolment for each of the input devices are covered under section 7.1.1.

Door Lock application will allow user to select “Mode of enrolment”. User can long press button 1 + button 2 to enter enrolment mode. Solid LED1 ON will reflect that user entered the enrolment mode. Also, user need to check the LED’s status to know whether device is in wake up or sleep mode. Also refer Section 4.1.1.8 for power consumption related functionality.

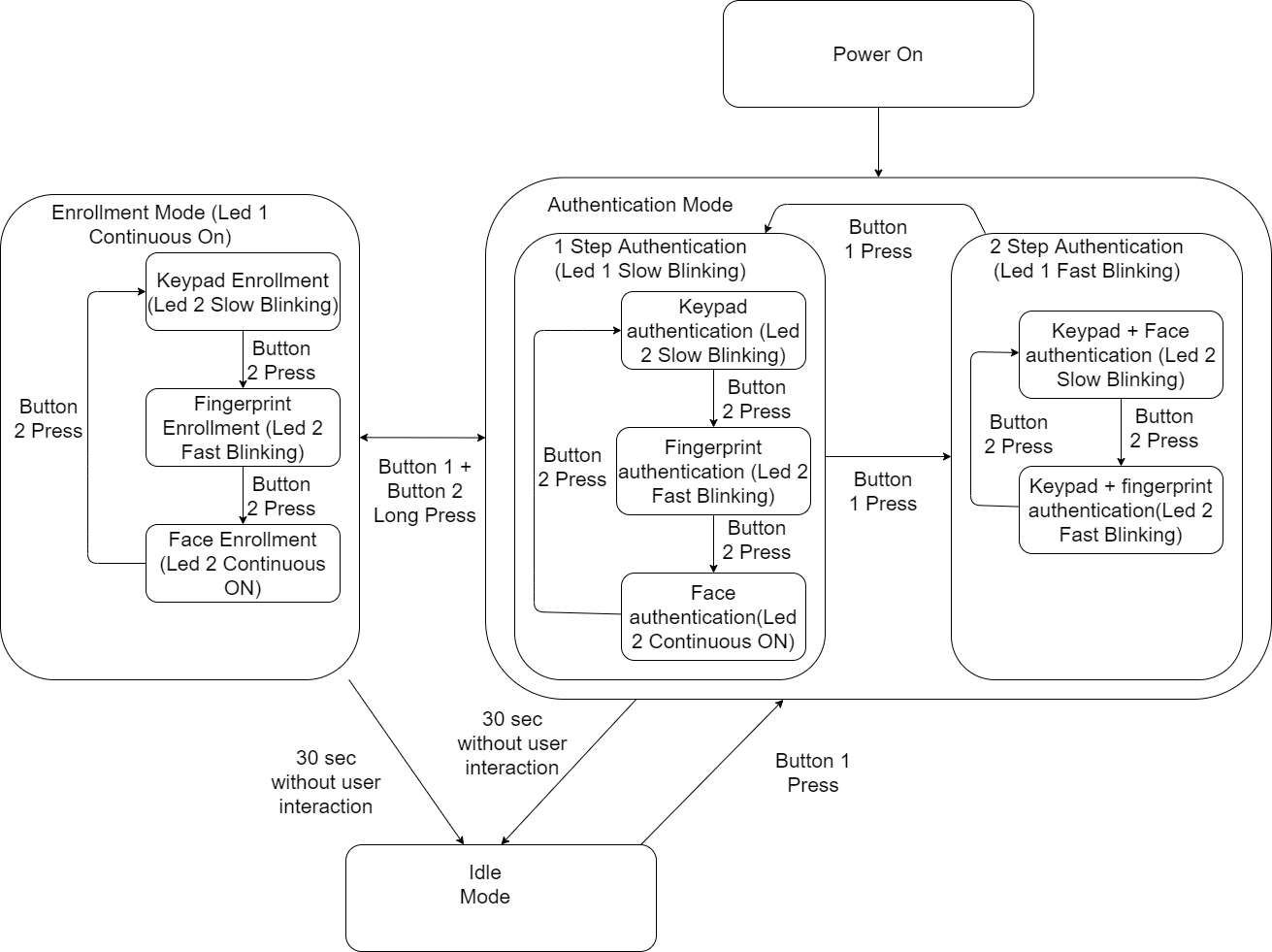
Once user is into enrolment mode, he can observe the module selected for enrolling on LED 2. Please refer below table for LED2 pattern:

|  |  |
| --- | --- |
| **Mode of enrolment** | **LED-2 pattern** |
| Keypad enrolment | Slow blinking |
| Fingerprint enrolment | Fast blinking |
| Face enrolment | Complete on |

**Note**: LED-1 will complete on when it is in enrolment mode.



**Fig 03: User enrolment and mode selection**



**Fig 04: Enrolment and authentication step**

#### **1 step authentication**

User can enter 1-step authentication after exiting the enrolment process. User can do that by pressing long pressing button 1 + button 2. 1-step authentication mode is default when we enter authentication mode. Enrolment process details are given section 7.1.1.

LED-1 slow blinking will confirm the 1-step authentication. Please refer Fig 03 and Fig 04 for the details.

#### **2-step authentication**

User can enter 2 step authentication mode by pressing button 1 when in authentication mode. LED-1 fast blinking will confirm the 2-step authentication. Please refer Fig 03 and fig 04 for the details.

#### **Switching between 1-step and 2-step authentication**

User can switch between 1-step authentication and 2-step authentication by pressing button1. LED 1 shows the step of authentication selected as given in the below table.

|  |  |
| --- | --- |
| **Step of authentication** | **LED-1 pattern** |
| 1-step authentication | Slow blinking |
| 2-step authentication | Fast blinking |

**Note**: Switching will only work when user is in authentication mode not in the enrolment mode.

#### **Mode of authentication**

User can select mode of authentication depending on 1-step or 2-step authentication. User need to press button-2 to select the mode of authentication. By default, it will enter keypad authentication then user can press button-2 again to move to next authentication mode.

If user is in 1-step authentication, then LED-2 blinking confirms the mode of authentication. Below are the LED-2 blinking patterns:

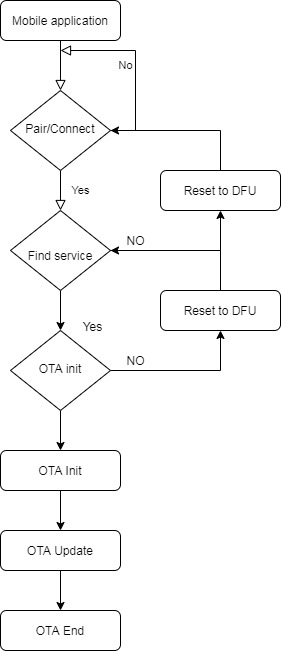
|  |  |
| --- | --- |
| **Mode of authentication** | **LED-2 pattern** |
| Keypad authentication | Slow blinking |
| Fingerprint authentication | Fast blinking |
| Face authentication | Solid ON |

If user is in 2-step authentication, the LED-2 blinking confirms the mode of authentication. Below are the LED-2 blinking patterns:

|  |  |
| --- | --- |
| **Mode of authentication** | **LED-2 pattern** |
| Keypad + Fingerprint authentication | Slow blinking |
| Keypad + Face authentication | Fast blinking |

#### **OTA update**

1. Mobile app for OTA update will connect to target device.
2. App requests target device to reboot into DFU mode.
3. After reboot, app connects again.
4. After reboot, during the 2nd connection, target device will be running AppLoader (not the user application).
5. New firmware image (application.gbl) is uploaded to the target.
6. AppLoader copies the new application on top of the existing application.
7. When upload is finished and connection closed, AppLoader reboots back to normal mode
8. Update complete.



**Fig 5: OTA update**

**Note**: - The above step might change based on the bootloader and the application selection which is TBD

#### **Power Management**

The Power management depending on which device is used and based on the configuration. Below are the steps we will take care to reduce the power consumption however it could vary as it is dependent on the performance.

1. Device will use the standby modes for low power consumption in transmit, receive.
2. Decrease the advertisement packet interval for reducing the power consumption.
3. Peripheral device will be in the sleep mode/low power mode. It will wake up on the interrupt.
4. For the scanning interval should be low as much possible to reduce the power.

**Sleep mode**: When the system is not used for 30 secs i.e. user has not interacted with the system, then the EFR device will put all the modules to standby mode and then go into sleep mode. The device will come out of sleep by pressing the button 1. This will drastically decrease the power consumption.

The steps for going into sleep will be:

1. Check if 30 secs have elapsed since last interaction
2. If 30 secs have elapsed, then send commands to all the devices to enter sleep/standby mode
3. Save all the necessary data to flash if required
4. Change the button 1 detection method from polling to interrupt based, to wake up the device by interrupt.
5. Put the EFR board to sleep

The steps for waking up will be:

1. Interrupt will be generated by button 1 when pressed, this will wake up the device from sleep
2. Data will be read from the flash memory
3. All the modules will be put into normal operating mode
4. Interrupt will be disabled on button 1 and polling will be done after waking up
5. Normal operation will resume

**Note:** - Power will be calculated based on power requirement on actual EVM & Kits during different modes & operations.

### **Mobile Application**

The mobile app will follow MVC (Model View Controller) architecture pattern. There will be 3 layers: the model, the view, the controller.

* **Model**: Here the data resides. Networking, communication, persistence storage, parsers code will reside here.
* **View**: This layer is the actual UI/UX of app. The classes, view which resides here are usually reusable since there isn’t any domain specific logic in them. For example, a Textbox is a view that presents text on the screen, and it is easily reusable.
* **Controller**: It mediates between the view and the model, typically via the delegation pattern. In the ideal scenario, the controller entity will not know the concrete view it is dealing with. Instead, it will communicate with an abstraction via a protocol. A classic example is the way controller communicates to other hardware via Bluetooth or Wi-Fi.

The mobile application will have following modules:

|  |  |
| --- | --- |
| Module Name | Respective Requirement ID |
| Discovery of EFR32 devices | RQ16 |
| Obtaining Initial Parameters | RQ17 |
| Communication with EFR32 | RQ18 |
| Authentication for Lock/Unlock Request | RQ19 |
| Notifications on mobile app | RQ20 |
| OTA update provisioning | RQ21 |

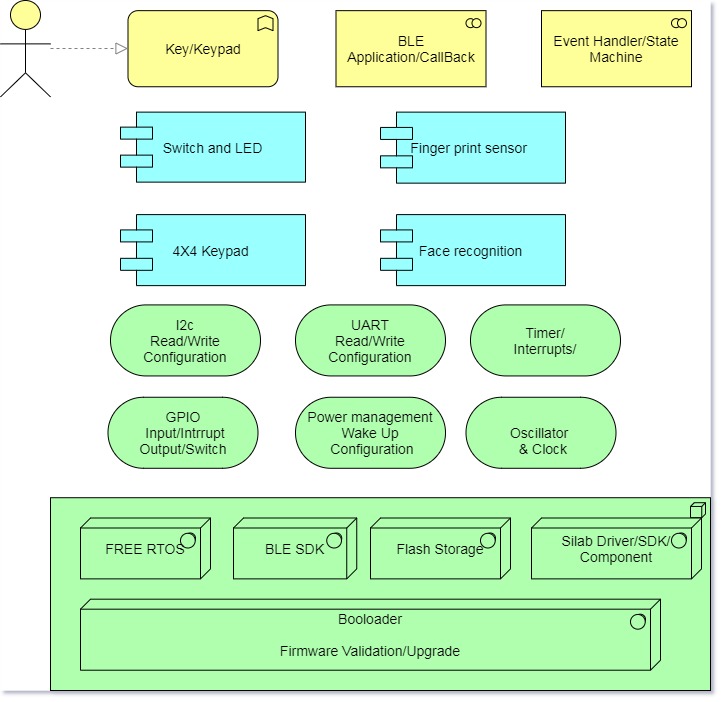
**Table 2: Mobile application modules**

|  |  |
| --- | --- |
| **(H/W, S/W) Components** | **Description** |
| **Programming Language** | Java |
| **Development Tool** | Android Studio IDE (version 3.5+) |
| **Android SDK** | API level 28+ (Oreo) |
| **Android Mobile for Testing** | Any (having android OS version 8 or greater) |

## **Hardware Architecture**

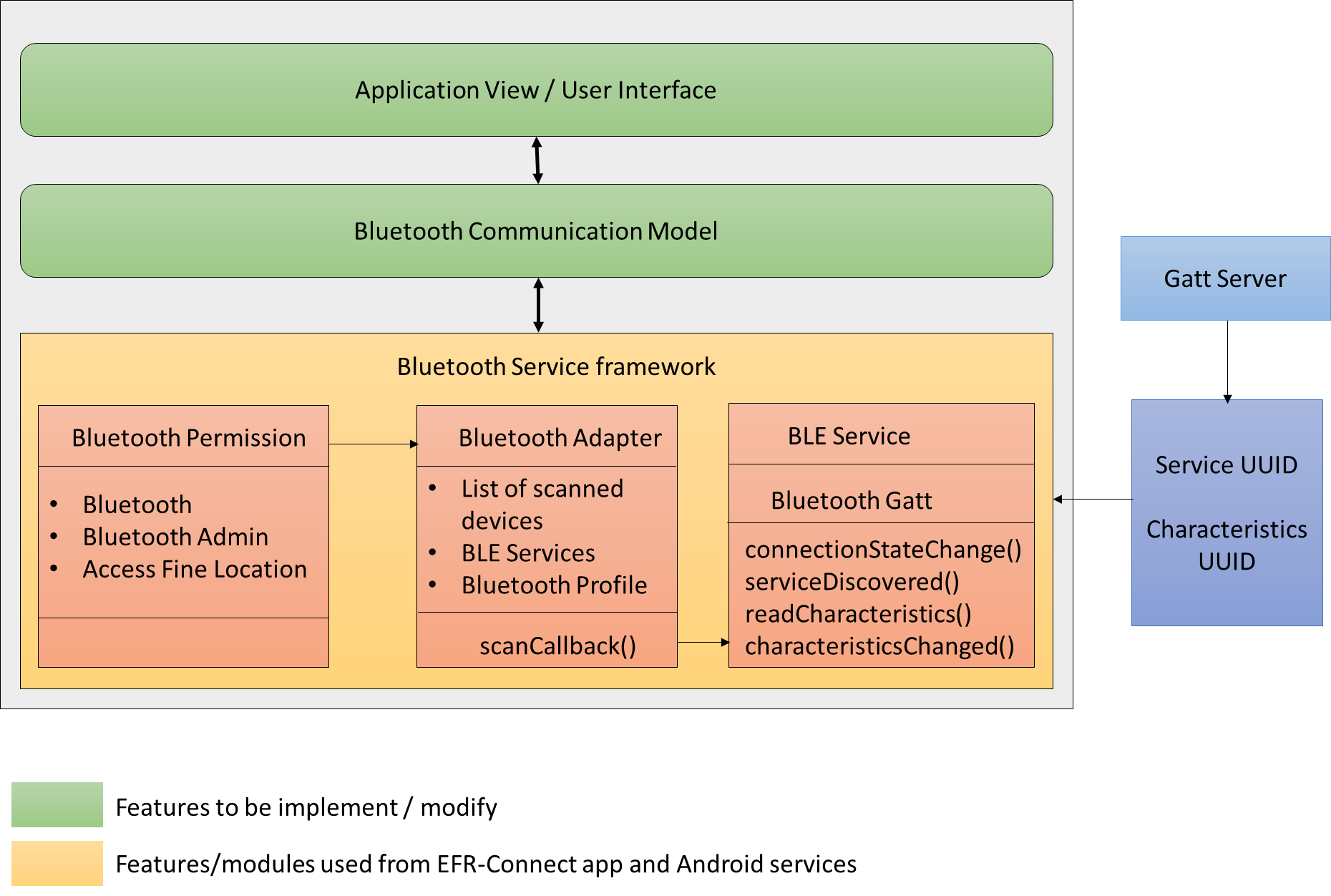
NA

## **Architecture Diagram**



**Fig 06: Architecture diagram**

## **Communication Architecture**



**Fig 07: Bluetooth communication architecture**

## **Resource Consumption Details**

## **4.5.1 Resource consumptions for mobile application development environment**

* It would typically use 50% of CPU utilization.
* 8GB RAM is used while development
* Minimum Core i3 or above configuration required
* Windows 10 / Ubuntu 14+ can be used.

# **Integration Details**

Keypad, Face recognition and fingerprint driver will be integrated with the SiLabs provided BSP. After integration BSP package will be built for generating the image for EFR32MG21 board.

Android app will be an independent module.

## **Integration Sequence**

NA

## **Integration Procedure**

NA

# **Tools for Architecture and Design**

## **Software Tools**

### **List of Software Items**

| **Name** | **Required Version(s)** |
| --- | --- |
| Windows OS | 10 |
| Ubuntu | 20 |
| Android Studio | 3.5 |
| Simplicity Studio | TBD |
| Java | 1.8+ |
| Git/SVN | Latest |
| SCA (TBD) | TBD |

### **List of SOUP Items**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Item Name** | **Purpose** | **Manufacturer/ Vendor Name** | **Hardware/ Software requirements** | **List of Functions** |
|  |  |  |  |  |
|  |  |  |  |  |

## **Hardware Tools**

| **Name** | **Specific Details** |
| --- | --- |
| Android mobile device | RAM 2GB(Minimum),  Internal Memory 8GB (Minimum),  Screen size: 5+ inch  Bluetooth Version 2+ |
| Development PC for Mobile App | Core i3 or above  Minimum 8GB RAM  Wide screen display (1920 \*1080)  500GB Hard Disk |
| BLE/ZigBee gateway | EFR32MG21 Dev kit |
| ZigBee switch | EFR32MG21 Dev kit |

# **Detailed Design**

## **Software Detailed Design**

ZigBee commissioning process is yet to be finalized. So the details of the commissioning process, device id, lock/unlock operation would be added later on.

### **Firmware & Software**

In the below section all the modules of the EFR32MG21 has been detailed out:

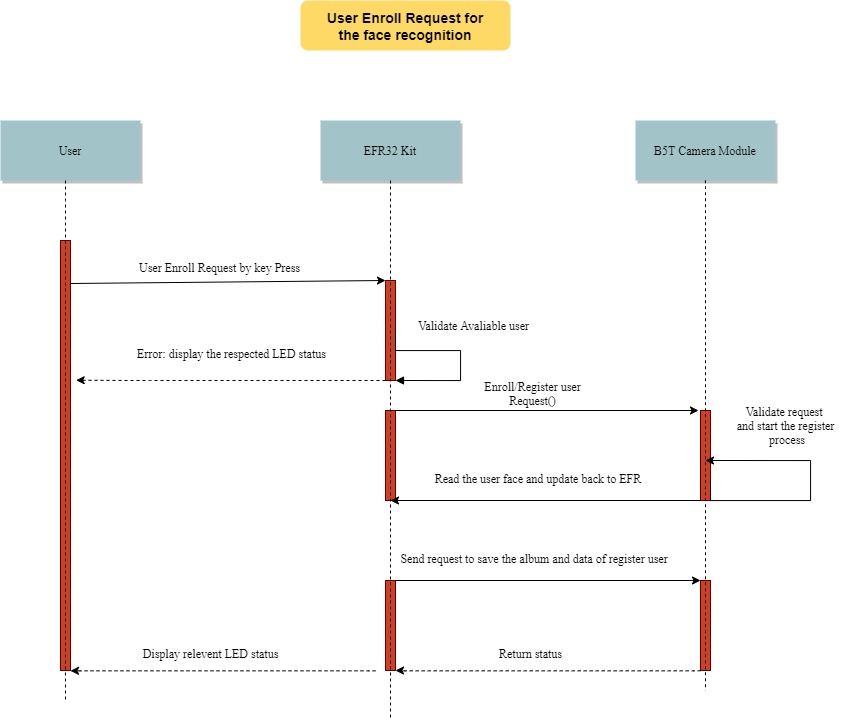
#### **Face Recognition module**

Face recognition can recognize in real time face of a person. It will help in the door lock system to provide personalized service to the residents identified with Face Recognition.

For Face Recognition used Omron B5T face Recognition module. Human Vision Components (HVC), the image sensor for Human Recognition, is a combination of Omron’s own image sensing technology

Face Recognition Authentication includes two parts: face enrolment and face matching. Face Recognition will compare the detected faces with the face recognition data registered beforehand in the album to recognize them among the registered users.

**7.1.1.1.1 Enrolment Process Flow of face recognition**

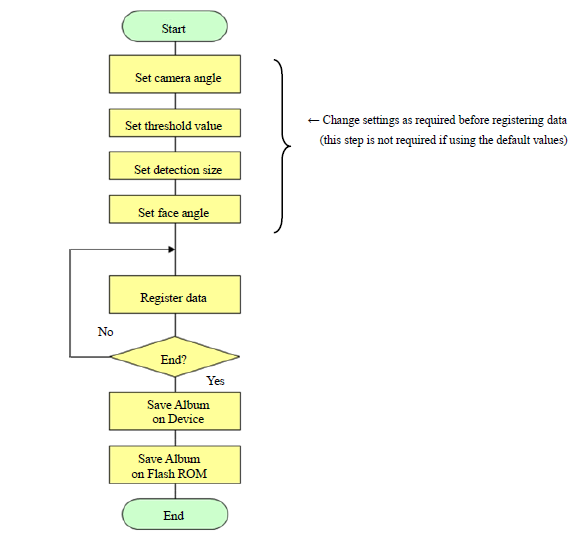


**Fig 08: Enrolment of face recognition**

As shown above in the diagram, below is the procedure for user enrolment for face recognition. User has to press the key to register/enrol for the face authentication, after pressing the key user has to stand in front of the camera and B5T module will capture the face and after successfully register the face appropriate LED status will be displayed.

1. User will place the face in front of the camera
2. Once the process of enrolment is over green led will glow for 2 sec
3. Every time the face recognition is to be registered a new user will be created

Below is the standard process flow of B5T module for user enrolment process for Face Recognition (when every command resulted in a normal end) executed on the Host side.

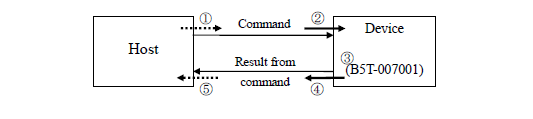


**Fig 09: Procedure of enrolment**

Take a picture of the user to be registered with the Device and get their recognition data. The recognition data will be stored in the Album as Album data. The Album data will be registered in the Album by specifying the user ID (0 to 99) and data ID (0 to 9). The album data will store in the flash ROM of the camera module.

This module will be controlled by the EFR32 on the UART, UART Buadrate will be 9600 (Default value), should be configurable from software up to Max: - 921600. On the uart receives the command controlling the module from the host and sends back the detection result info.

**Command Flow**



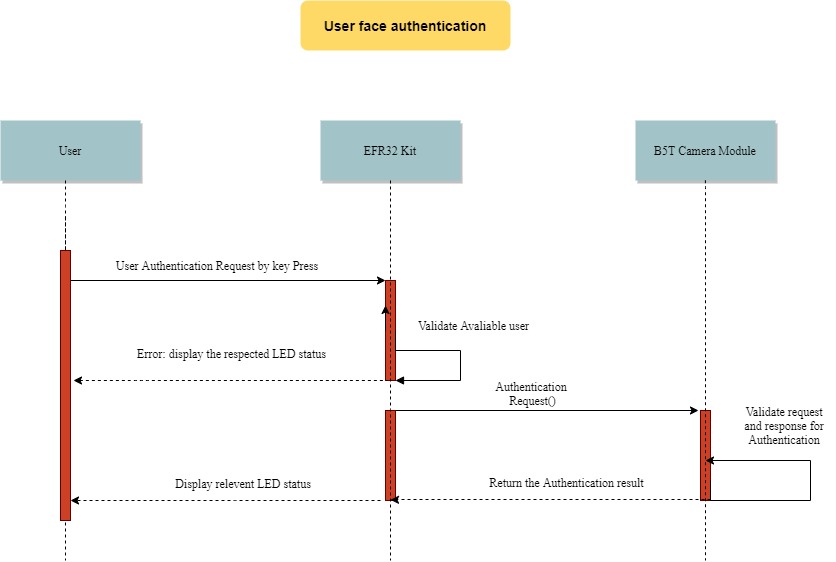
**Fig 11: Command Flow**

The common command flow between the Host side and the Device is as described below.

1. The command is sent from the Host side to the Device.
2. The Device receives the command.
3. The Device executes the command.
4. The Device sends the result of the executed command back to the Host side.
5. The Host side received the result of the executed command.

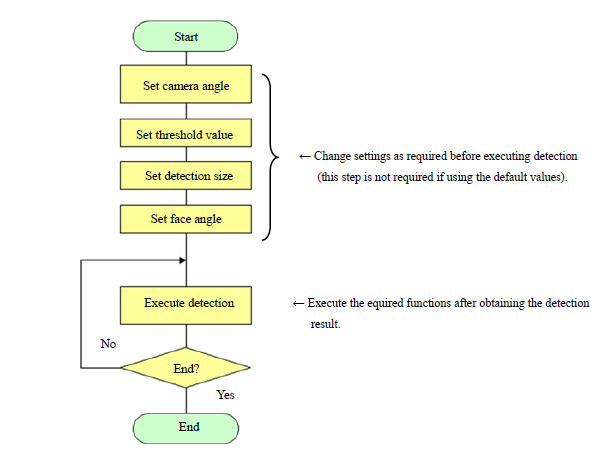
**7.1.1.1.2 Face recognition use case:**

**Authentication/Detection Process Flow**



**Fig 12: Authentication process flow**

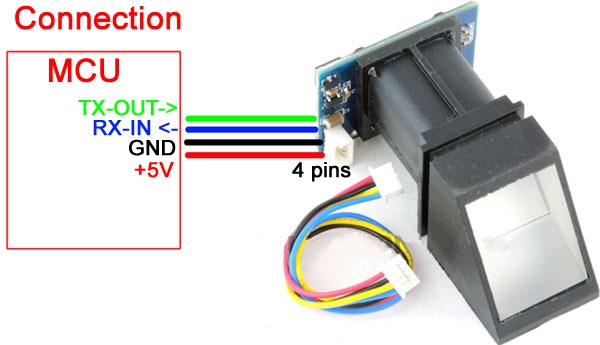
Above diagram is for user face recognition to detect the valid user ID and below flowchart is for the B5T module and it is the standard process flow for the detection process (when every command resulted in a normal end) executed on the Host side.



**Fig 10: Face authentication**

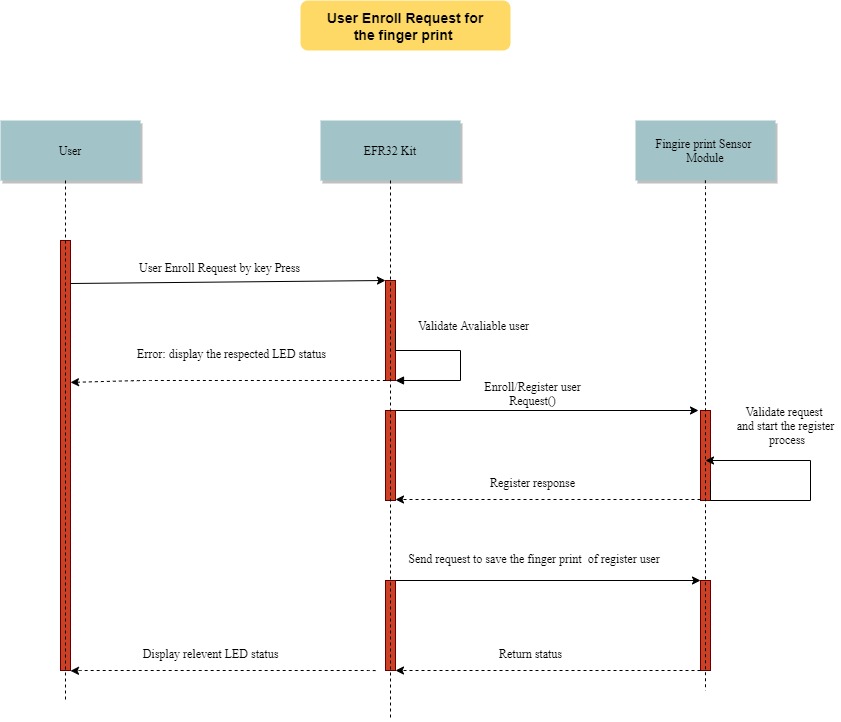
#### **Fingerprint Sensor**

Fingerprint processing includes two parts: fingerprint enrolment and fingerprint matching (the matching can be 1:1 or 1: N). When enrolling, user needs to enter the finger two times. The system will process the two-time finger images, generate a template of the finger based on processing results and store the template. When matching, user enters the finger through optical sensor and system will generate a template of the finger and compare it with templates of the finger library. For 1:1 matching, system will compare the live finger with specific template designated in the Module; for 1: N matching, or searching, system will search the whole finger library for the matching finger. In both circumstances, system will return the matching result, success or failure.



**Fig 13: Fingerprint sensor**

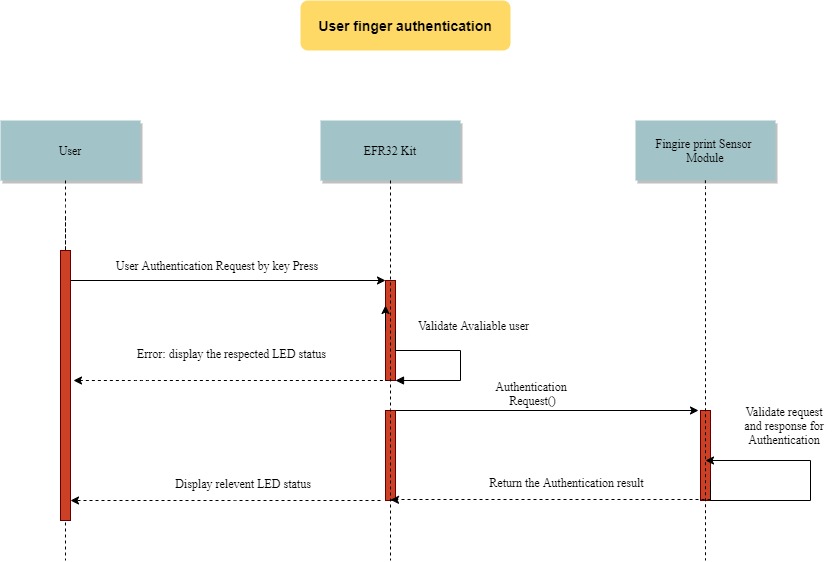
##### **Fingerprint enrolment:**



**Fig 14: Fingerprint enrolment**

1. User need to enter enrolment mode and select fingerprint sensor by following the section 4.1.1.2
2. User will place the finger on the fingerprint scanner
3. If finger image is acceptable the green led will glow for 2 sec. If unacceptable, red led will glow.
4. Once the finger is accepted user will have to place the finger again to take the second image. Until user gets green 2 times the enrolment will not be completed

##### **Authentication/Detection Process Flow**



**Fig 15: Fingerprint authentication**

When matching, user enters the finger through optical sensor and system will generate a template of the finger and compare it with templates of the finger library. For 1:1 matching, system will compare the live finger with specific template designated in the Module; for 1: N matching, or searching, system will search the whole finger library for the matching finger. In both circumstances, system will return the matching result, success, or failure.

##### **Fingerprint use case:**

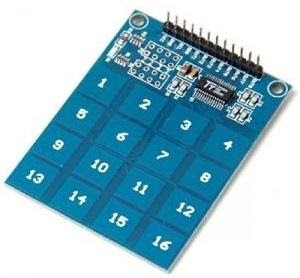
The fingerprint sensor support UART interface to communicate with the host device. Fingerprint sensor uses command response interface for communication. For authentication, using this module user must have enrolled his finger prior to authentication.

Once enrolment is done, user can exit the enrolment mode by pressing button 1 and button 2 simultaneously. After exiting the enrolment user will be in 1-step authentication mode by default. This is indicated by slow blinking of LED 1. User can then select mode of authentication as fingerprint by pressing the button 2 until LED 2 is fast blinking. Fast blinking LED 2 means fingerprint is selected.

After selecting the fingerprint sensor, user will have to place his finger on the sensor to unlock the door. Door will unlock only if there is a match for fingerprint already enrolled. Green Led will indicate that the fingerprint is accepted. Red Led will indicate that the module could not find any matching fingerprints.

#### **Keypad Module**

The TTP229 TonTouchTM IC is capacitive sensing design specifically for touch pad controls. The device built in regulator for touch sensor. Stable sensing method can cover diversity conditions. Human interfaces control panel links through non-conductive dielectric material. The main application is focused at replacing of the mechanical switch or button. The ASSP can independently handle the 8 touch pads or up to 16 touch pads.



**Fig 16: Keypad input**

##### **Keypad enrolment:**

1. User need to enter enrolment mode and select keypad input by following the section 4.1.1.2
2. User will enter the default/current password
3. If the password is correct, then green LED will glow for 2 sec
4. If password is wrong red LED will glow and user will not be able to proceed unless correct password is entered
5. If the default/current password is correct green LED will glow and user will have to enter the password to change/save.
6. User will have to enter 4-digit pin. If correct number of digits are entered green LED will glow again else red LED will glow, and user will have to enter password again

##### **Keypad Authentication:**

1. User need to enter authentication mode by pressing the button 1 and button 2 simultaneously.
2. User will then select the authentication mode as keypad by pressing the button 2 until the module is selected. Keypad selection is shown by slow blinking of LED 2.
3. User will enter the default/current password to login
4. The password will be matched with the enrolled password, if the password is correct the door will be unlocked, and green led will be ON for 2 sec. If the password is wrong the red led will be ON for 2 sec indicating that the password entered is wrong and user should retry.

##### **Keypad use case:**

The Keypad support I2C interface to communicate with the host device. Keypad module uses command response interface for communication. The TTP229 module only supports read commands. For authentication using this module user must have enrolled a password prior to authentication. There will be a master password if not enrolled.

Once enrolment is done, user can exit the enrolment mode by pressing button 1 and button 2 simultaneously. After exiting the enrolment user will be in 1-step authentication mode by default. This is indicated by slow blinking of LED 1. User can then select mode of authentication as keypad by pressing the button 2 until LED 2 is slow blinking. Slow blinking LED 2 means keypad is selected.

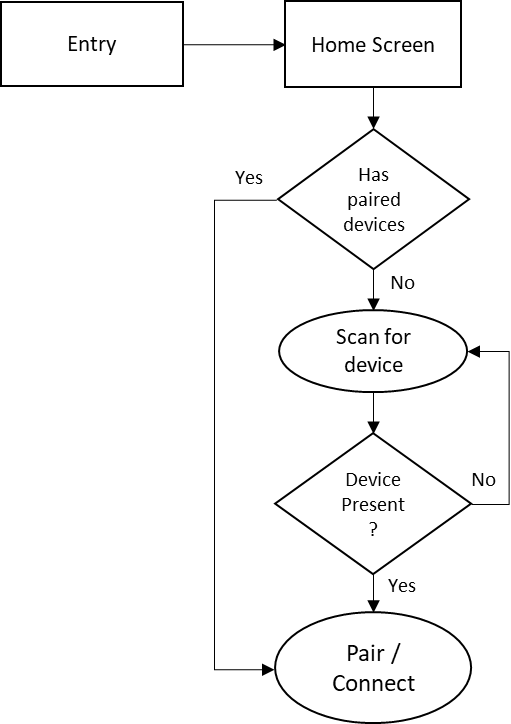
After selecting the Keypad module, user can enter the password to unlock the door. Door will unlock only if there is a match for password. Green Led will indicate that the password is accepted. Red Led will indicate that the wrong password was entered.

### **Mobile Application**

The Mobile application will be an independent module which will connect the EFR32MG21 device using Bluetooth and communicate with the hardware for lock/unlock events triggered by the end user.

#### **Bluetooth connection with EFR32 Boards**

This will provide functionality to discover EFR32 devices. The user will launch the app and will scan for the devices. As a result, app will show the filtered list of nearby EFR32 devices by UUID.



**Fig 17: Flow diagram for EFR32 discovery**

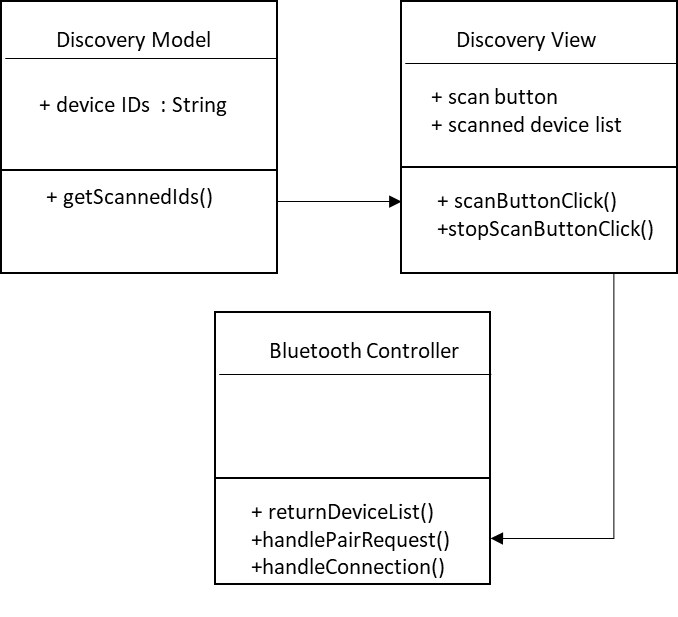


Fig 18: Class diagram for EFR32 discovery

#### **Obtaining the initial device parameters**

After pairing successfully with EFR32 device, the mobile will receive initial device parameters such as door status and will update the state in mobile app.

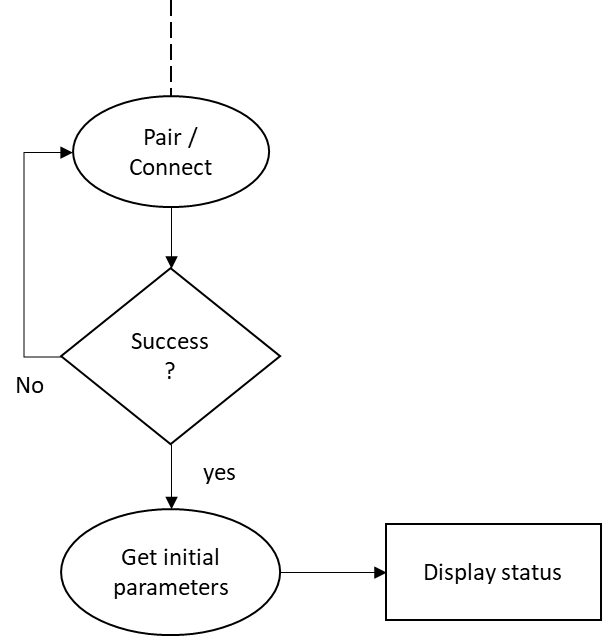


Fig 19: Flow diagram for obtaining initial device parameter

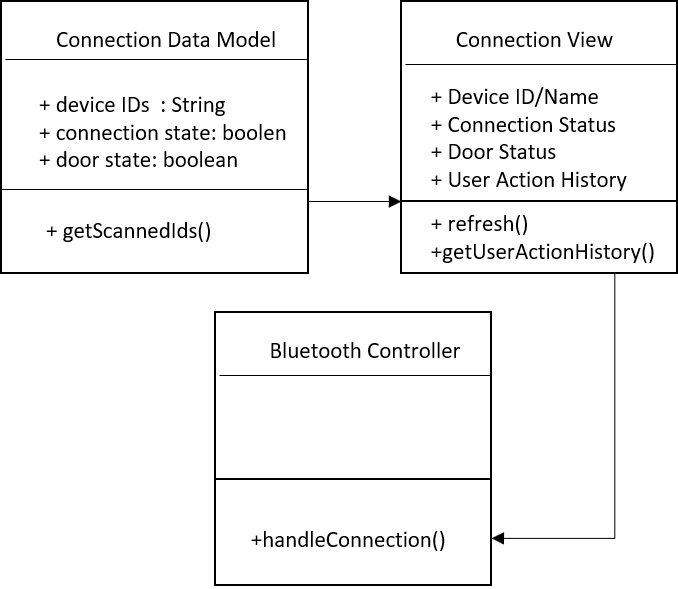
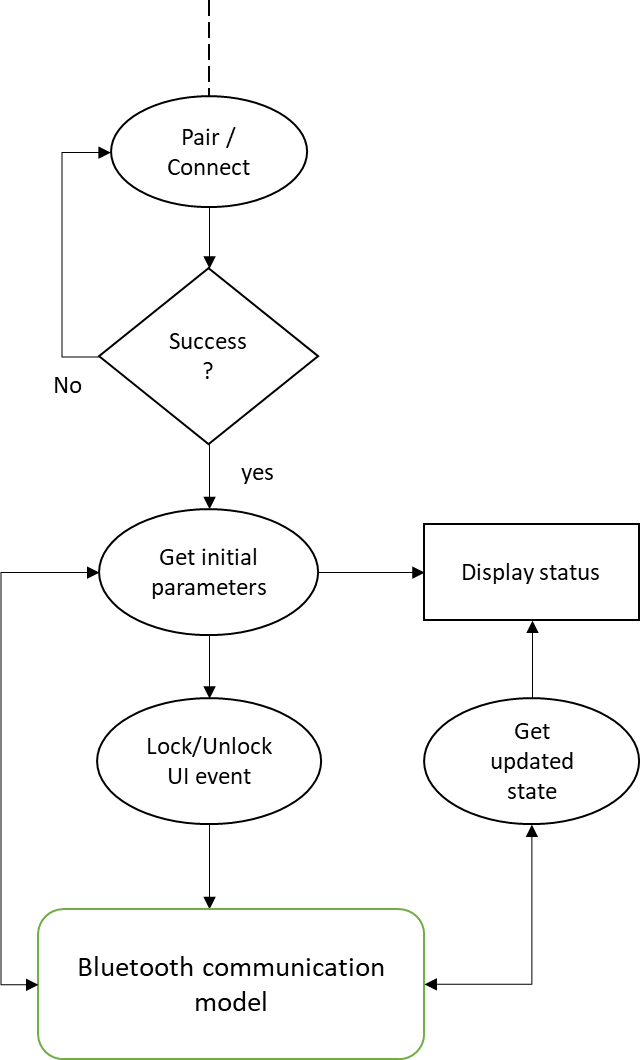


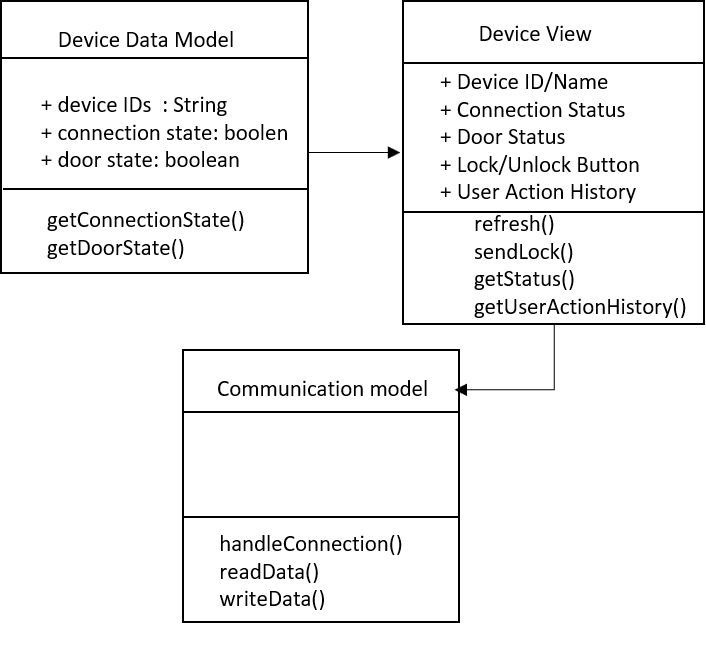
Fig 20: Class diagram for obtaining initial device parameters

#### **Communication with EFR32**

Once mobile app is paired and connected with EFR32, it will be ready for data exchange. User will able to send the door lock/unlock commands from the mobile app by clicking the lock/unlock button from mobile app screen. Also app will be able to show the user action history that was happened when mobile is not connected with the board.



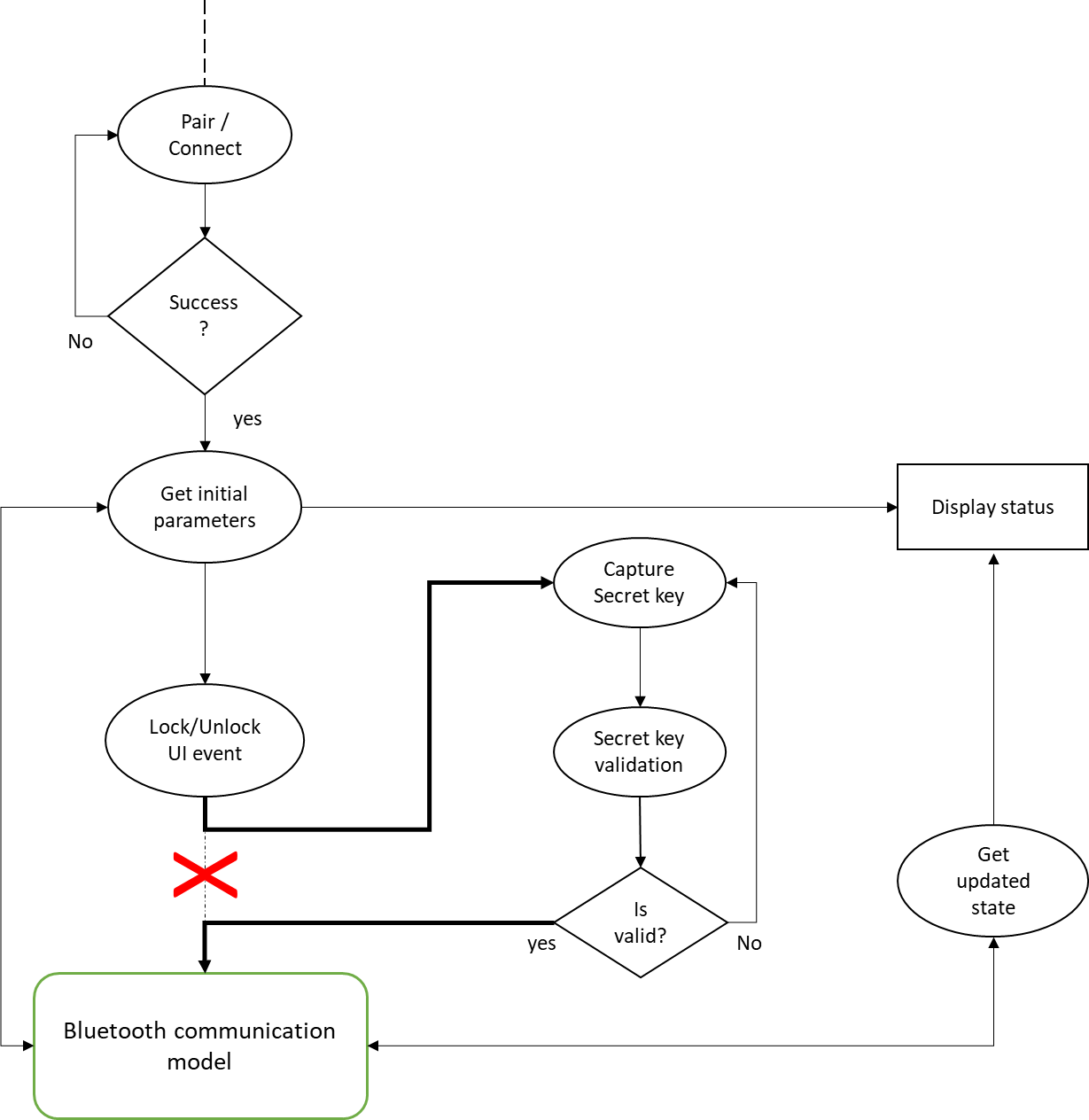
**Fig 21: Flow diagram for Communication with EFR32**



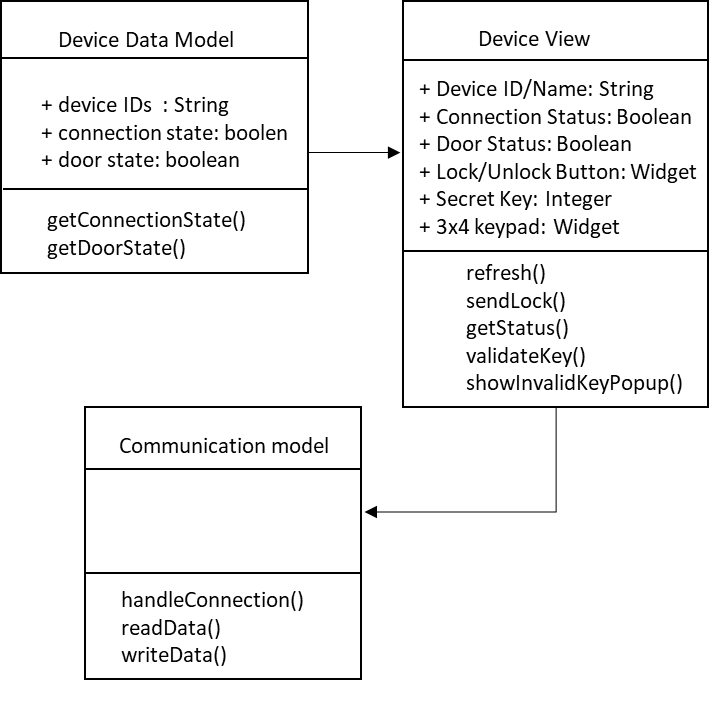
**Fig 22: Class diagram for communication with EFR32**

#### **Authentication for the lock/unlock request**

In this stage app is going to authenticate the request using numeric secret key. As of now the secret key is stored as static (later user will be able to configure the key). When user clicks on lock/unlock button, the app will ask user to enter secret key. If the secret key is valid then the lock/unlock operation will perform otherwise app will not send any lock/unlock request to EFR32.



**Fig 23: Flow diagram for Authentication of lock/unlock request**

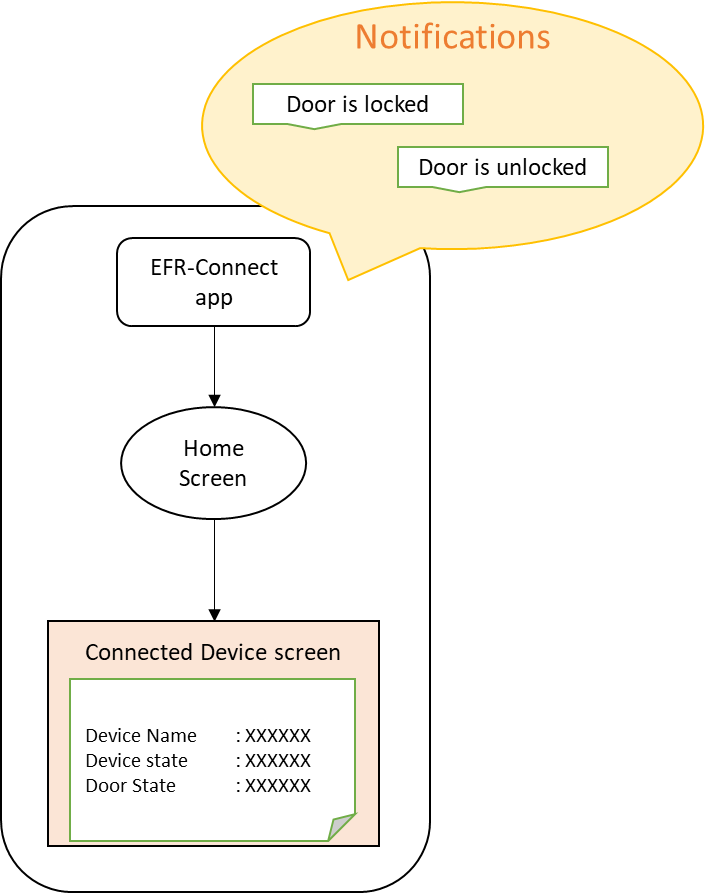


**Fig 24: Class diagram for authentication of lock/unlock request**

#### **Notifications on mobile app**

When user locks/unlocks the door using mobile app or other interface like fingerprint or face then app will receive the event of door status if it is connected to EFR32 and will notify user for the door state using sound or on-screen notification.

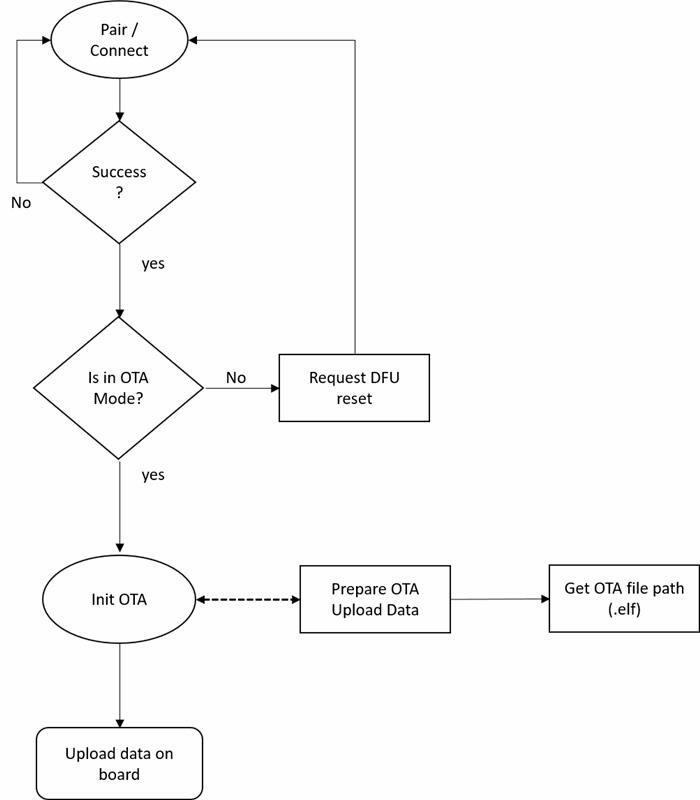
If mobile is connected and a user authenticated using fingerprint or face recognition then user will be notified on the mobile app with the user id. If mobile app is not connected while authentication then user will have an option to synchronize with the device later on using an option called “Sync” in the mobile app.



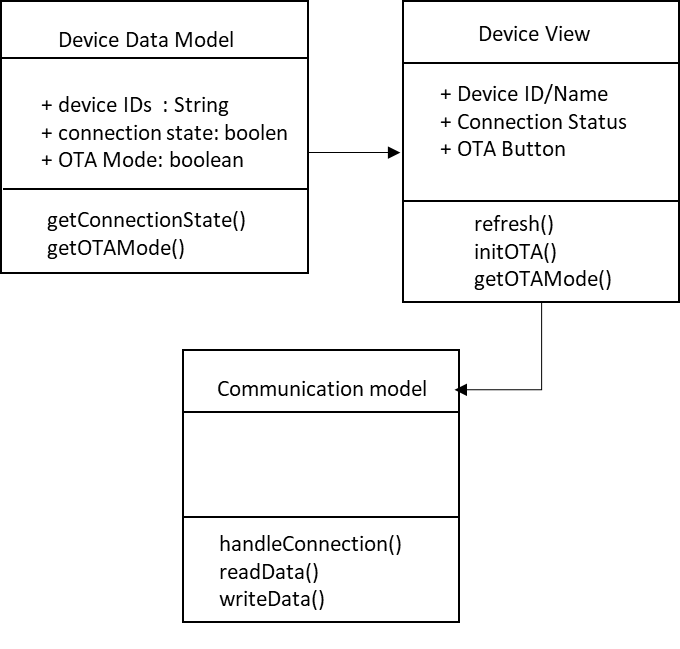
**Fig 25: Notifications on mobile app**

#### **OTA Update Provisioning**

It helps to configure/provision the OTA update of EFR32 firmware. If device is connected, it will have an option for OTA update. User need to click on mobile app OTA update option to process update request on EFR32 board.



**Fig 26: Flow diagram from OTA provisioning support from mobile app**



**Fig 27: Class diagram for OTA provisioning from mobile app**

# **Data Design**

Enum for the generic error code for the application logic

typedef enum generic\_error\_code

{

NOT\_FOUND = -3,

INVALID\_DATA = -2,

FAILURE = -1,

SUCCESS = 0

} generic\_error\_code;

## **UART Communication low level api for the face and finger authentication module:**

**UART init sequence API,**

Ecode\_t UARTDRV\_InitUart(UARTDRV\_Handle\_t handle, const UARTDRV\_InitUart\_t \* initData );

Parameters:

[in] handle Pointer to a UART driver handle.

[in] data Receive data buffer.

[in] count Number of bytes received.

Returns

ECODE\_EMDRV\_UARTDRV\_OK on success.

**UART transmit the data API,**

Ecode\_t UARTDRV\_Transmit( UARTDRV\_Handle\_t handle, uint8\_t \* data, UARTDRV\_Count\_t count, UARTDRV\_Callback\_t callback );

Start a non-blocking transmit.

Parameters

[in] handle Pointer to a UART driver handle.

[in] data Transmit data buffer.

[in] count Number of bytes to transmit.

[in] callback Transfer completion callback.

Returns

ECODE\_EMDRV\_UARTDRV\_OK on success.

**UART receive** **the data API,**

Ecode\_t UARTDRV\_Receive(UARTDRV\_Handle\_t handle, uint8\_t \* data, UARTDRV\_Count\_t count, UARTDRV\_Callback\_t callback );

Parameters

[in] handle Pointer to a UART driver handle.

[in] data Receive data buffer.

[in] count Number of bytes received.

[in] callback Transfer completion callback.

Returns

ECODE\_EMDRV\_UARTDRV\_OK on success.

## **I2C low level api for the keypad authentication module**

**I2C init sequence API,**

void I2C\_Init(I2C\_TypeDef \* i2c, const I2C\_Init\_TypeDef \* init );

**Parameters**

|  |  |  |
| --- | --- | --- |
| [in] | **i2c** | Pointer to I2C peripheral register block. |
| [in] | **init** | Pointer to I2C initialization structure. |

**I2C transfer the data API,**

I2C\_TransferReturn\_TypeDef I2C\_Transfer(I2C\_TypeDef \* i2c);

**I2C keypad read the data API,**

int KEYPAD\_Read(I2C\_TypeDef \* i2c, uint8\_t addr, unsigned int offset, uint8\_t \*data, unsigned int len );

Read data from keypad.

Parameters

[in] i2c Pointer to I2C peripheral register block.

[in] addr I2C address for keypad, in 8 bit format, where LSB is reserved for R/W bit.

[in] offset Offset in keypad to start reading from.

[out] data Location to place read data, must be at least len long.

[in] len Number of bytes to read.

Returns

Returns number of bytes read. Negative value is returned is some sort of error occurred during read.

**Note**: The API’s and enum may change during the implementation. The given one is for reference.

# **External Interface Design**

NA

# **Human Machine / User Interface**

## **ZigBee Verification:**

Mobile app for controlling the ZigBee operation is still under discussion. Once it is finalized app screens would be added here in this section.

## **Interface Design Rules**

Android UI interface rules will be followed for android application

## **Inputs**

### **Mobile App inputs**

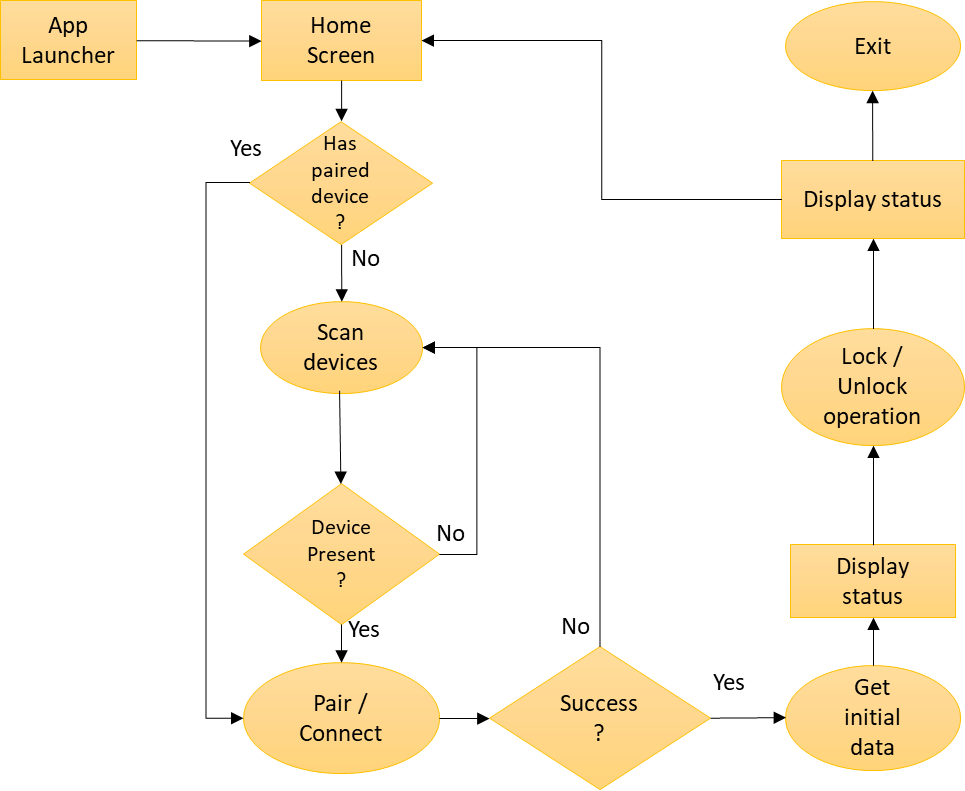
Some inputs for information to screen are data entry points like numeric secret keys.

## **Outputs**

### **Mobile App inputs**

Some inputs for information to screen are data entry points like numeric secret keys.

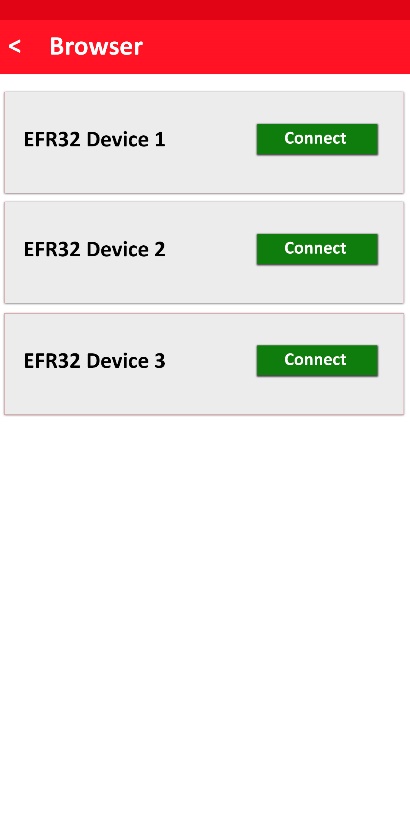
## **Navigational Hierarchy**



**Fig 28: Navigation hierarchy for mobile application**

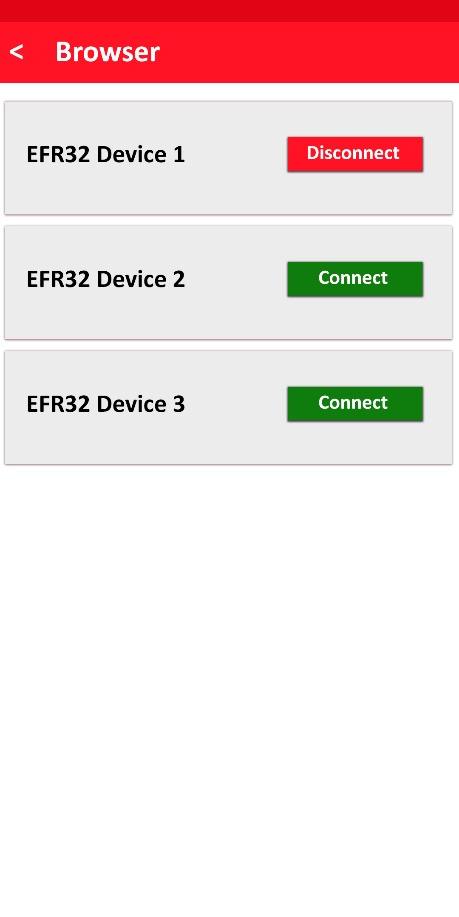
### **Mobile App Screen**

* + - 1. **Device Discovery**



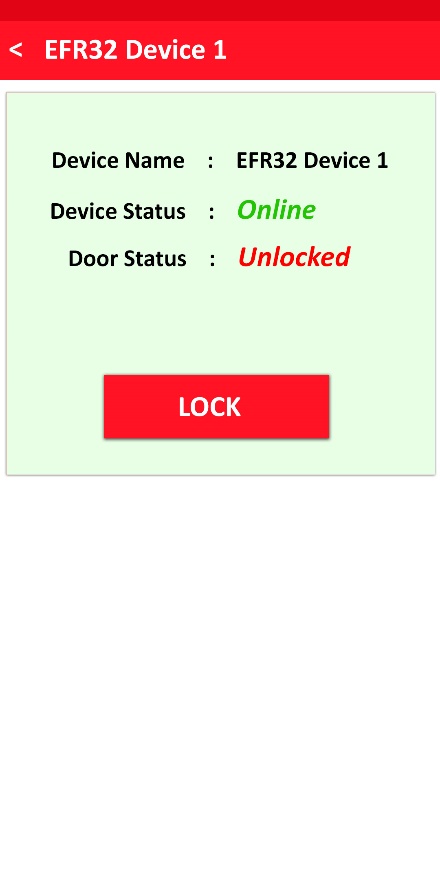
**Fig 29: Device discovery**

* + - 1. **Connected state after paired**



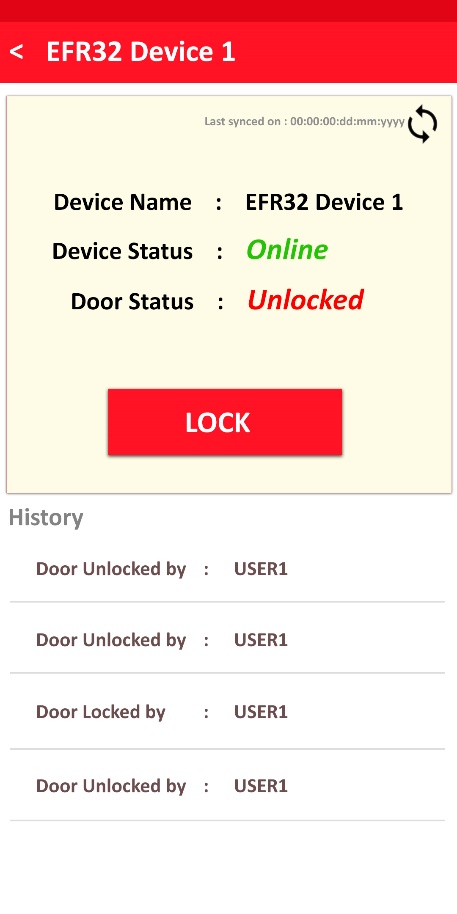
**Fig 30: Connected state after paired**

* + - 1. **Device parameter screen**



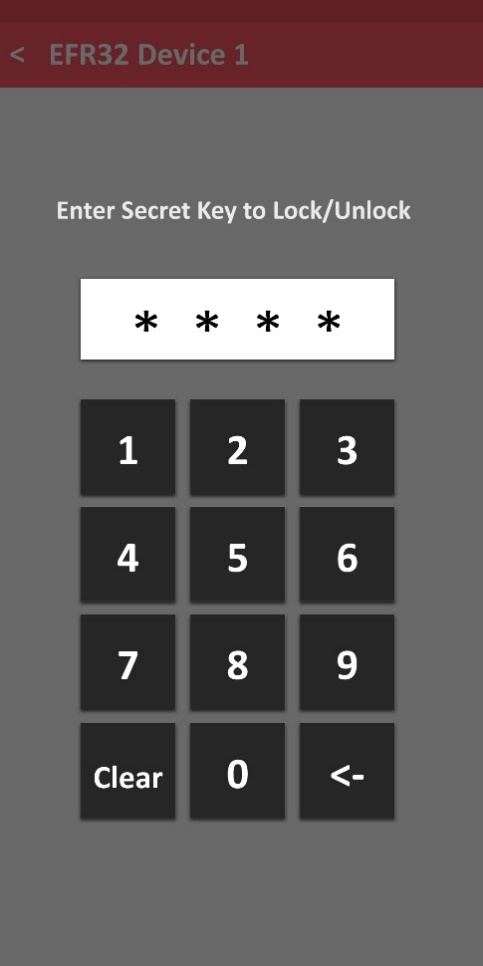
**Fig 31:** **Device parameter screen**

* + - 1. **Device parameter with history**



**Fig 32: Device parameter with history of user action**

* + - 1. **Secret key authentication**



**Fig 33: Secret key authentication**

**Note:** Mobile App screens in section 10 to be considered as a sample view. It can be different/modified while implementation.

# **Constraints**

1. At a time one authentication step will work.
2. Fingerprint sensor algorithm is going to use of 3rd party so accuracy and limitation will be based on it.
3. Image detection algorithm is going to use of 3rd party so accuracy and limitation will be based on it.
4. Face recognition module have maximum of 100 users and 10 maximum of photos per user to store data in their own flash memory.
5. Fingerprint sensor have maximum of 1000 storage for 10 users.
6. Some required data (secret key) will be temporary stored in App database and OTA related files are stored in internal memory of mobile.
7. Mobile app user needs to be in the range of the EFR32MG21 device to appropriately use the door lock/unlock.
8. For demo purpose for all authentication method will support maximum for 1 user.

# **Risk Analysis Output**

NA

# **Appendix**