**PO1\_DGELV\_Digital**

**GDD Document**

**Version 1.2**

**Proposed**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Document Change History** | | | | |
| **Version** | **Author** | **Date** | **Change** | **Status** |
| 1.0 | - Salma Amr | 29/2/2020 | * Initial creation | Draft |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Document Change History** | | | | |
| **Version** | **Author** | **Date** | **Change** | **Status** |
| 1.0 | - Salma Amr | 29/2/2020 | * Initial creation | Draft |
| 1.0 | - Salma Amr | 29/2/2020 | * Initial creation * Adding project Description * Adding static architecture | Proposed |
| 1.1 | - Mira Mousa | 01/3/2020 | * Adding Software Context * Adding Input and Output Signals | Proposed |
| 1.2 | -Mohamed Nafea | 02/3/2020 | * Adding API components | Proposed |
| 1.3 | -Mohamed Nafea  -Salma Amr | 06/3/2020 | * Adding Application layer APIs | Proposed |

|  |  |  |
| --- | --- | --- |
| **Reference Documents** | | |
| **Document Name** | **Version** | **Status** |
| SRS | 1.6 | Proposed |

**Table of Contents**

[**1.** **Project Description** 7](#_Toc34059066)

[**1.1** **Software Context Diagram** 7](#_Toc34059067)

[**2.** **Input and Output Signals** 8](#_Toc34059068)

[**2.1** **Login Verification** 8](#_Toc34059069)

[**2.2** **Alarm Component** 9](#_Toc34059070)

[**2.3** **Display** 10](#_Toc34059071)

[**2.4** **Input Reading** 12](#_Toc34059072)

[**2.5** **Elevator Motion Control** 13](#_Toc34059073)

[**3.** **Static Architecture** 14](#_Toc34059074)

[**3.1** **Layered Architecture** 14](#_Toc34059075)

[**4.** **Software Components** 16](#_Toc34059076)

[4.1 MCAL APIs 16](#_Toc34059077)

[4.2.1 GPIO 16](#_Toc34059078)

[4.2.1.1 GPIO Initiate 16](#_Toc34059079)

[1. STD\_ERROR GPIO\_voidInit (void) 16](#_Toc34059080)

[4.2.1.2 GPIO Set Pin Mode 16](#_Toc34059081)

[2. - Req\_PO1\_DGELV\_GDD\_002\_V01 16](#_Toc34059082)

[3. STD\_ERROR GPIO\_voidSetPinMode (u8 Copy\_u8Port, u8 Copy\_u8Pin, u8 Copy\_u8Mode) 16](#_Toc34059083)

[4.2.2.1 GPIO Set Pin Value 16](#_Toc34059084)

[4. STD\_ERROR GPIO\_voidSetPinValue (u8 Copy\_u8Port, u8 Copy\_u8Pin, u8 Copy\_u8Value) 16](#_Toc34059085)

[4.2.2.2 GPIO Get Pin Value 17](#_Toc34059086)

[5. STD\_ERROR GPIO\_u8GetPinValue (u8 Copy\_u8Port, u8 Copy\_u8Pin, u8 \* Copy\_u8Value) 17](#_Toc34059087)

[4.2.1 Timer 17](#_Toc34059088)

[4.2.2.3 Timer Initiate 17](#_Toc34059089)

[- Req\_PO1\_DGELV\_GDD\_005\_V01 17](#_Toc34059090)

[6. STD\_ERROR TIMER\_ STD\_ERRORInit (void) 17](#_Toc34059091)

[4.2.2.4 Timer Initiate 17](#_Toc34059092)

[- Req\_PO1\_DGELV\_GDD\_005\_V01 17](#_Toc34059093)

[7. STD\_ERROR TIMER\_ STD\_ERRORInit (void) 17](#_Toc34059094)

[4.2.2.5 Timer Set Callback 18](#_Toc34059095)

[- Req\_PO1\_DGELV\_GDD\_006\_V01 18](#_Toc34059096)

[8. STD\_ERROR TIMER\_ STD\_ERRORSetCallback (void(\* Copy\_PtrCallback)(void)) 18](#_Toc34059097)

[4.2.2.6 PWM Start 18](#_Toc34059098)

[- Req\_PO1\_DGELV\_GDD\_007\_V01 18](#_Toc34059099)

[9. STD\_ERROR TIMER\_ STD\_ERRORPWMStart (void) 18](#_Toc34059100)

[4.2.2.7 PWM Generate Signal 18](#_Toc34059101)

[- Req\_PO1\_DGELV\_GDD\_008\_V01 18](#_Toc34059102)

[10. STD\_ERROR TIMER\_ STD\_ERRORPWMGenerateSignal (u16 Copy\_OnPeriod, u16 Copy\_TotalPeriod) 18](#_Toc34059103)

[4.2.2.8 Timer Set Callback 19](#_Toc34059104)

[- Req\_PO1\_DGELV\_GDD\_006\_V01 19](#_Toc34059105)

[11. STD\_ERROR TIMER\_ STD\_ERRORSetCallback (void(\* Copy\_PtrCallback)(void)) 19](#_Toc34059106)

[4.2.2.9 PWM Start 19](#_Toc34059107)

[- Req\_PO1\_DGELV\_GDD\_007\_V01 19](#_Toc34059108)

[12. STD\_ERROR TIMER\_ STD\_ERRORPWMStart (void) 19](#_Toc34059109)

[4.2.2.10 PWM Generate Signal 19](#_Toc34059110)

[- Req\_PO1\_DGELV\_GDD\_008\_V01 19](#_Toc34059111)

[13. STD\_ERROR TIMER\_ STD\_ERRORPWMGenerateSignal (u16 Copy\_OnPeriod, u16 Copy\_TotalPeriod) 19](#_Toc34059112)

[4.2 ECUAL APIs 20](#_Toc34059113)

[4.2.1 Switch 20](#_Toc34059114)

[4.2.1.1 Switch Initiate 20](#_Toc34059115)

[- Req\_PO1\_DGELV\_GDD\_016\_V01 20](#_Toc34059116)

[14. STD\_ERROR SWITCH\_STD\_ERRORInit (void) 20](#_Toc34059117)

[4.2.1.2 Get Switch State 20](#_Toc34059118)

[- Req\_PO1\_DGELV\_GDD\_017\_V01 20](#_Toc34059119)

[15. STD\_ERROR SWITCH\_STD\_ERRORGetSwitchState (u8 \* Copy\_u8PtrSwitchState) 20](#_Toc34059120)

[4.2.2 Buzzer 21](#_Toc34059121)

[4.2.3 LCD 21](#_Toc34059122)

[4.2.4 Keypad 21](#_Toc34059123)

[4.2.5 Motor 21](#_Toc34059124)

[4.3 Application APIs 21](#_Toc34059125)

[4.3.1 Login Verification 21](#_Toc34059126)

[4.3.2 Alarm Component 21](#_Toc34059127)

[4.3.3 Display 21](#_Toc34059128)

[4.3.4 Input Reading 21](#_Toc34059129)

[4.3.5 Elevator Motion Control 21](#_Toc34059130)

# **Project Description**

The major feature of the digital elevator as listed below:

* Check if the user is signed in/up or not.
* Check the limit number of users (maximum number of users is 10).
* Display the status using a user-friendly interface.
* Require the password and user ID.
* Two buttons to control the system:

1. Button for up movement.
2. Button for down movement.
3. Button for on/off and reset the system.

## **Software Context Diagram**

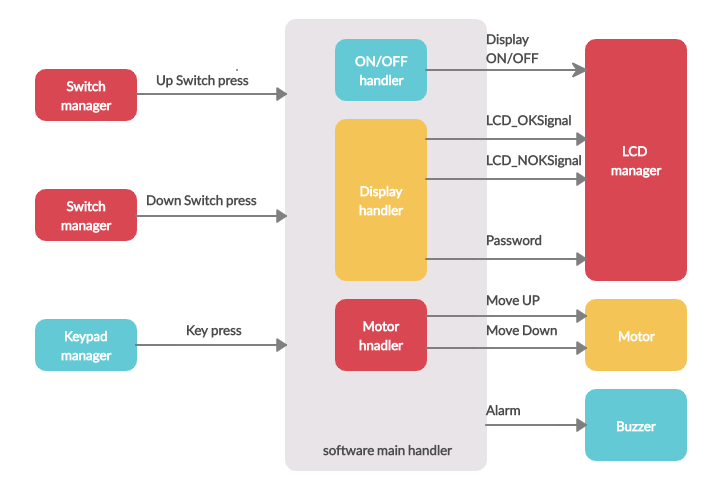


Figure 1| Software context diagram

# **Input and Output Signals**

The input and output signals in the project are listed below in figures, with explained information about each signal.

## **Login Verification**

**Input signals:** UserID\_Signal, UserPass\_Signal.

**Output signals:** LoggedInAccepted\_Signal, LoggedInRejected\_Signal.

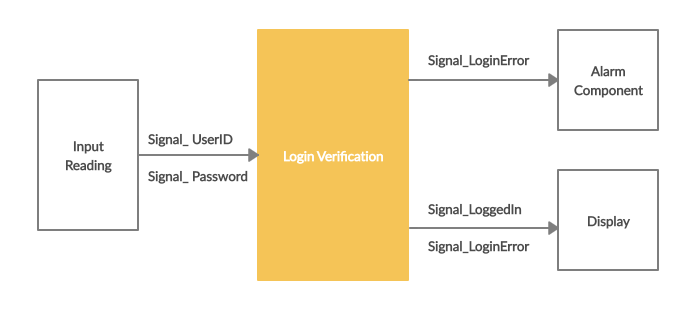


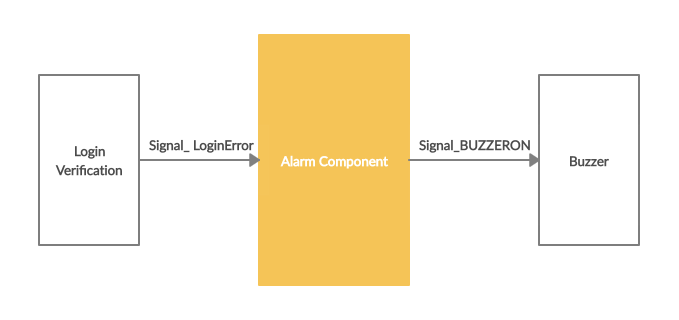
Figure 2| Login Verification Control Signals

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| UserID\_Signal | |  | UserPass\_Signal | |
| **Range** | [a-z,A-Z] |  | **Range** | [0000,9999] |
| **Unit** | NA |  | **Unit** | NA |
|  |  |  |  |  |
| LoggedInAccepted\_Signal | |  | LoggedInRejected\_Signal | |
| **Range** | [1] |  | **Range** | [0] |
| **Unit** | NA |  | **Unit** | NA |

## **Alarm Component**

**Input signals:** LoggedInRejected\_Signal.

**Output signals:** BUZZERON\_Signal.



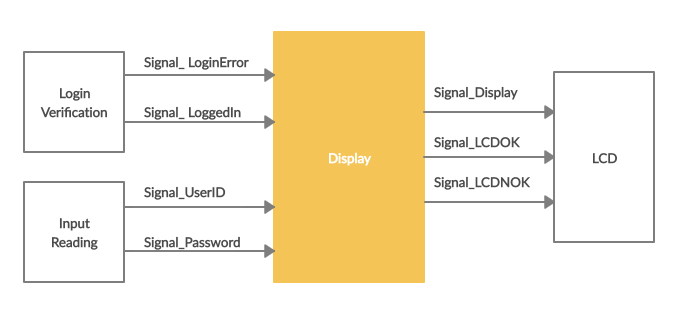
*Figure ‎2-2Alarm Component Control Signals*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| BUZZERON\_Signal | |  | LoggedInRejected\_Signal | |
| **Range** | [0,1] |  | **Range** | [0] |
| **Unit** | NA |  | **Unit** | NA |

## **Display**

**Input signals:** Signal\_ UserID, Signal\_ Password, Signal\_LoggedIn, Signal\_LoginError.

**Output signals:** Signal\_Display, Signal\_LCDOK, Signal\_LCDNOK.



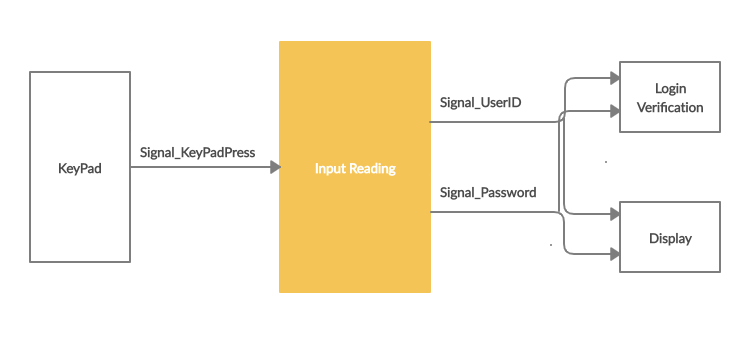
*Figure ‎2-3Display Control Signals*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Signal\_ LoggedIn | |  | LoggedInRejected\_Signal | |
| **Range** | [1] |  | **Range** | [0] |
| **Unit** | NA |  | **Unit** | NA |
|  | |
| Signal\_ UserID | |  | Signal\_ Password | |
| **Range** | [A,C] |  | **Range** | [0000,9999] |
| **Unit** | NA |  | **Unit** | NA |
|  | |
| Signal\_ Display | |  | Signal\_ LCDNOK | |
| **Range** | [0,1] |  | **Range** | [0] |
| **Unit** | NA |  | **Unit** | NA |
|  | |
| Signal\_ LCDOK | |  |  | |
| **Range** | [1] |  |  |  |
| **Unit** | NA |  |  |  |

## **Input Reading**

**Input signals:** Signal\_KeypadPressed.

**Output signals:** Signal\_ UserID, Signal\_ Password.



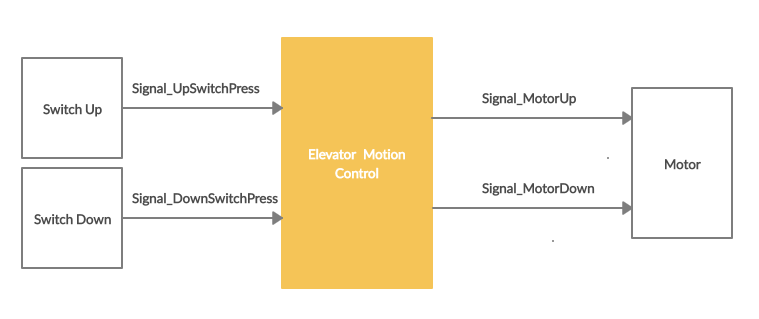
*Figure ‎2-4Input ReadingControl Signals*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Signal\_ UserID | |  | Signal\_ Password | |
| **Range** | [A,C] |  | **Range** | [0000,9999] |
| **Unit** | NA |  | **Unit** | NA |
|  |  |  |  |  |
| Signal\_ KeyPadPress | |  |  | |
| **Range** | [0,9 and A,C] |  |  |  |
| **Unit** | NA |  |  |  |

## **Elevator Motion Control**

**Input signals:** Signal\_UpSwitchPressed, Signal\_DownSwitchPressed.

**Output signals:** Signal\_ MotorUp, Signal\_MotorDown.

*Figure ‎2-5Elevator Motion Control Signals*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Signal\_UpSwitchPress | |  | Signal\_ DownSwitchPress | |
| **Range** | [0,1] |  | **Range** | [0,1] |
| **Unit** | NA |  | **Unit** | NA |
|  | |
| Signal\_ MotorDown | |  | Signal\_ MotorUp | |
| **Range** | [00 or 01] |  | **Range** | [00 or 10] |
| **Unit** | NA |  | **Unit** | NA |

# **Static Architecture**

## **Layered Architecture**

The layered architecture represents the architecture of the project as separate horizontal layers, and shows the dependency of each module in any layer on other modules as shown in Figure ‎3.

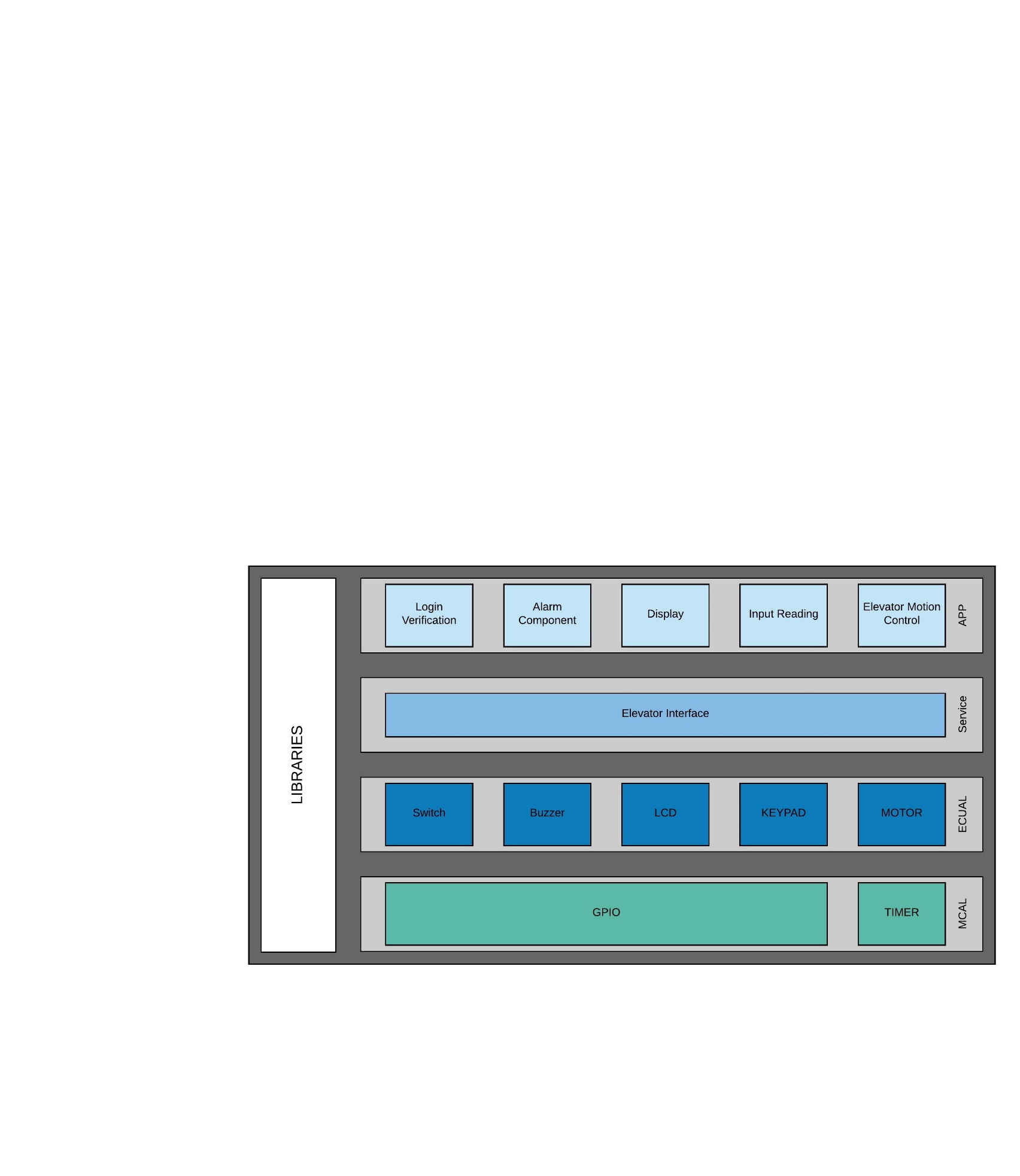


Figure 3| Project Layered Architecture

# **Software Components**

## MCAL APIs

### 4.1.1 GPIO

#### GPIO Initialize

|  |  |
| --- | --- |
| Requirement ID | Req\_PO1\_DGELV\_GDD\_001\_V01.3 |
| API | STD\_ERROR GPIO\_voidInit (void) |
| Description | This API shall initialize GPIO pins with initial values or configured values if exists |
| Arguments | * Void |
| Return | * STD\_ERROR that holds the error state of the API implementation, it shall be 0 for OK if everything is done correctly, and 1 for NOK if anything goes wrong. |
| Covers | * NA |

#### GPIO Set Pin Mode

|  |  |
| --- | --- |
| Requirement ID | Req\_PO1\_DGELV\_GDD\_002\_V01.3 |
| API | STD\_ERROR GPIO\_voidSetPinMode (u8 Copy\_u8Port, u8 Copy\_u8Pin, u8 Copy\_u8Mode) |
| Description | This API shall set pin mode as input or output |
| Arguments | * Copy\_u8Port: specifies pin’s port, which shall be one of this options: A, B, C, D * Copy\_u8Pin: specifies pin number, which shall be number in this range: [0-7] * Copy\_u8Mode: specifies pin mode, which is 0 for input and 1 for output |
| Return | * STD\_ERROR that holds the error state of the API implementation, it shall be 0 for OK if everything is done correctly, and 1 for NOK if anything goes wrong. |
| Covers | * NA |

#### GPIO Set Pin Value

|  |  |
| --- | --- |
| Requirement ID | Req\_PO1\_DGELV\_GDD\_003\_V01.3 |
| API | STD\_ERROR GPIO\_voidSetPinValue (u8 Copy\_u8Port, u8 Copy\_u8Pin, u8 Copy\_u8Value) |
| Description | This API shall set pin value |
| Arguments | Copy\_u8Port: specifies pin’s port, which shall be one of this options: A, B, C, DCopy\_u8Pin: specifies pin number, which shall be number in this range: [0-7]Copy\_u8Mode: specifies value set to the pin, which shall be 0 or 1 |
| Return | STD\_ERROR that holds the error state of the API implementation, it shall be 0 for OK if everything is done correctly, and 1 for NOK if anything goes wrong. |
| Covers | NA |

#### GPIO Get Pin Value

|  |  |
| --- | --- |
| Requirement ID | Req\_PO1\_DGELV\_GDD\_04\_V01.3 |
| API | STD\_ERROR GPIO\_u8GetPinValue (u8 Copy\_u8Port, u8 Copy\_u8Pin, u8 \* Copy\_u8Value) |
| Description | This API shall get pin value |
| Arguments | Copy\_u8Port: specifies pin’s port, which shall be one of this options: A, B, C, DCopy\_u8Pin: specifies pin number, which shall be number in this range: [0-7]Copy\_u8Value: pointer that holds the value of the pin, it shall be 0 or 1 |
| Return | STD\_ERROR that holds the error state of the API implementation, it shall be 0 for OK if everything is done correctly, and 1 for NOK if anything goes wrong. |

### Timer

#### Timer Initialize

|  |  |
| --- | --- |
| Requirement ID | Req\_PO1\_DGELV\_GDD\_005\_V01.3 |
| API | STD\_ERROR TIMER\_ STD\_ERRORInit (void) |
| Description | This API shall initiate timer |
| Arguments | * void |
| Return | * STD\_ERROR that holds the error state of the API implementation, it shall be 0 for OK if everything is done correctly, and 1 for NOK if anything goes wrong. |
| Covers | * NA |

#### Timer Initiate

|  |  |
| --- | --- |
| Requirement ID | Req\_PO1\_DGELV\_GDD\_006\_V01.3 |
| API | STD\_ERROR TIMER\_ voidInit (void) |
| Description | This API shall initiate timer |
| Arguments | * void |
| Return | * STD\_ERROR that holds the error state of the API implementation, it shall be 0 for OK if everything is done correctly, and 1 for NOK if anything goes wrong. |
| Covers | * NA |

#### Timer Set Callback

|  |  |
| --- | --- |
| Requirement ID | Req\_PO1\_DGELV\_GDD\_007\_V01.3 |
| API | STD\_ERROR TIMER\_ voidSetCallback (void(\* Copy\_PtrCallback)(void)) |
| Description | This API shall set call back function of timer interrupt |
| Arguments | * Copy\_PtrCallback: pointer that holds the address of callback function |
| Return | * STD\_ERROR that holds the error state of the API implementation, it shall be 0 for OK if everything is done correctly, and 1 for NOK if anything goes wrong. |
| Covers | * NA |

#### PWM Start

|  |  |
| --- | --- |
| Requirement ID | Req\_PO1\_DGELV\_GDD\_008\_V01.3 |
| API | STD\_ERROR TIMER\_voidPWMStart (void) |
| Description | This API shall start PWM by setting callback function related to PWM |
| Arguments | * void |
| Return | * STD\_ERROR that holds the error state of the API implementation, it shall be 0 for OK if everything is done correctly, and 1 for NOK if anything goes wrong. |
| Covers | * NA |

#### PWM Generate Signal

|  |  |
| --- | --- |
| Requirement ID | Req\_PO1\_DGELV\_GDD\_009\_V01.3 |
| API | STD\_ERROR TIMER\_ voidPWMGenerateSignal (u16 Copy\_OnPeriod, u16 Copy\_TotalPeriod) |
| Description | This API shall generate PWM signal by setting its On period, total period and peak value |
| Arguments | * Copy\_OnPeriod: specifies the duration of the On part of the period * Copy\_TotalPeriod: specifies the duration of the total period |
| Return | * STD\_ERROR that holds the error state of the API implementation, it shall be 0 for OK if everything is done correctly, and 1 for NOK if anything goes wrong. |
| Covers | * NA |

#### Timer Set Callback

|  |  |
| --- | --- |
| Requirement ID | Req\_PO1\_DGELV\_GDD\_010\_V01.3 |
| API | STD\_ERROR TIMER\_ voidSetCallback (void(\* Copy\_PtrCallback)(void)) |
| Description | This API shall set call back function of timer interrupt |
| Arguments | * Copy\_PtrCallback: pointer that holds the address of callback function |
| Return | * STD\_ERROR that holds the error state of the API implementation, it shall be 0 for OK if everything is done correctly, and 1 for NOK if anything goes wrong. |
| Covers | * NA |

#### PWM Start

|  |  |
| --- | --- |
| Requirement ID | Req\_PO1\_DGELV\_GDD\_011\_V01.3 |
| API | STD\_ERROR TIMER\_ voidPWMStart (void) |
| Description | This API shall start PWM by setting callback function related to PWM |
| Arguments | * void |
| Return | * STD\_ERROR that holds the error state of the API implementation, it shall be 0 for OK if everything is done correctly, and 1 for NOK if anything goes wrong. |
| Covers | * NA |

#### PWM Generate Signal

|  |  |
| --- | --- |
| Requirement ID | Req\_PO1\_DGELV\_GDD\_012\_V01.3 |
| API | STD\_ERROR TIMER\_voidPWMGenerateSignal (u16 Copy\_OnPeriod, u16 Copy\_TotalPeriod) |
| Description | This API shall generate PWM signal by setting its On period, total period and peak value |
| Arguments | * Copy\_OnPeriod: specifies the duration of the On part of the period * Copy\_TotalPeriod: specifies the duration of the total period |
| Return | * STD\_ERROR that holds the error state of the API implementation, it shall be 0 for OK if everything is done correctly, and 1 for NOK if anything goes wrong. |
| Covers | * NA |

## 4.2 ECUAL APIs

### Switch

#### 4.2.1.1 Switch Initiatilize

|  |  |
| --- | --- |
| Requirement ID | Req\_PO1\_DGELV\_GDD\_013\_V01.3 |
| API | STD\_ERROR SWITCH\_voidInit (void) |
| Description | This API shall initiate switch on its configured pins |
| Arguments | * void |
| Return | * STD\_ERROR that holds the error state of the API implementation, it shall be 0 for OK if everything is done correctly, and 1 for NOK if anything goes wrong. |
| Covers | * NA |

#### Get Switch State

|  |  |
| --- | --- |
| Requirement ID | Req\_PO1\_DGELV\_GDD\_014\_V01.3 |
| API | STD\_ERROR SWITCH\_voidGetSwitchState (u8 \* Copy\_u8PtrSwitchState) |
| Description | This API shall get the switch state |
| Arguments | * Copy\_u8PtrSwitchState: pointer that holds the switch state, it shall be 1 if switch is pressed and 0 if switch is released. |
| Return | * STD\_ERROR that holds the error state of the API implementation, it shall be 0 for OK if everything is done correctly, and 1 for NOK if anything goes wrong. |
| Covers | * NA |

### Buzzer

#### Buzzer Initiate

|  |  |
| --- | --- |
| Requirement ID | Req\_PO1\_DGELV\_GDD\_015\_V01.3 |
| API | STD\_ERROR BUZZER\_voidInit (void) |
| Description | This API shall initiate buzzer on its configured pins |
| Arguments | * void |
| Return | * STD\_ERROR that holds the error state of the API implementation, it shall be 0 for OK if everything is done correctly, and 1 for NOK if anything goes wrong. |
| Covers | * NA |

#### Run Buzzer

|  |  |
| --- | --- |
| Requirement ID | Req\_PO1\_DGELV\_GDD\_016\_V01.3 |
| API | STD\_ERROR BUZZER\_voidRunBuzzer (u16 Copy\_u16Frequency, u8 Copy\_u8Duty) |
| Description | This API shall run buzzer for specific frequency in specific duty cycle |
| Arguments | * Copy\_u16Frequency: specifies the buzzer frequency, it shall be 250 or 500 Hz * Copy\_u8Duty: specifies the percentage of duty cycle |
| Return | * STD\_ERROR that holds the error state of the API implementation, it shall be 0 for OK if everything is done correctly, and 1 for NOK if anything goes wrong. |
| Covers | * NA |

### LCD

#### LCD Initialize

|  |  |
| --- | --- |
| Requirement ID | Req\_PO1\_DGELV\_GDD\_017\_V01.3 |
| API | STD\_ERROR LCD\_voidInit (void) |
| Description | This API shall initiate LCD configurations |
| Arguments | * void |
| Return | * STD\_ERROR that holds the error state of the API implementation, it shall be 0 for OK if everything is done correctly, and 1 for NOK if anything goes wrong. |
| Covers | * NA |

#### LCD Write Data

|  |  |
| --- | --- |
| Requirement ID | Req\_PO1\_DGELV\_GDD\_018\_V01.3 |
| API | STD\_ERROR LCD\_voidWriteString (u8 \* Copy\_u8PtrString) |
| Description | This API shall display string on LCD |
| Arguments | * Copy\_u8PtrString: pointer that holds the string value |
| Return | * STD\_ERROR that holds the error state of the API implementation, it shall be 0 for OK if everything is done correctly, and 1 for NOK if anything goes wrong. |
| Covers | * NA |

#### LCD Write Command

|  |  |
| --- | --- |
| Requirement ID | Req\_PO1\_DGELV\_GDD\_019\_V01.3 |
| API | STD\_ERROR LCD\_voidWriteCmd (u8 Copy\_u8Cmd) |
| Description | This API shall send command to LCD |
| Arguments | * Copy\_ u8Cmd: specifies command to be sent to LCD |
| Return | * STD\_ERROR that holds the error state of the API implementation, it shall be 0 for OK if everything is done correctly, and 1 for NOK if anything goes wrong. |
| Covers | * NA |

#### LCD Go to Location

|  |  |
| --- | --- |
| Requirement ID | Req\_PO1\_DGELV\_GDD\_020\_V01.3 |
| API | STD\_ERROR LCD\_voidGoToLocation (u8 Copy\_u8Line, u8 Copy\_u8Position) |
| Description | This API shall set LCD cursor at specific line and position |
| Arguments | * Copy\_ u8Line: specifies line of the cursor, it shall be 1 or 2 * Copy\_ u8Position: specifies position of the cursor in the line, it shall be in the range [0-15] |
| Return | * STD\_ERROR that holds the error state of the API implementation, it shall be 0 for OK if everything is done correctly, and 1 for NOK if anything goes wrong. |
| Covers | * NA |

#### LCD Clear

|  |  |
| --- | --- |
| Requirement ID | Req\_PO1\_DGELV\_GDD\_021\_V01.3 |
| API | STD\_ERROR LCD\_voidClear (void) |
| Description | This API shall clear the display of LCD |
| Arguments | * void |
| Return | * STD\_ERROR that holds the error state of the API implementation, it shall be 0 for OK if everything is done correctly, and 1 for NOK if anything goes wrong. |
| Covers | * NA |

### Keypad

#### Keypad Initialize

|  |  |
| --- | --- |
| Requirement ID | Req\_PO1\_DGELV\_GDD\_021\_V01.3 |
| API | STD\_ERROR KEYPAD\_voidInit (void) |
| Description | This API shall initiate keypad on its configured pins |
| Arguments | * void |
| Return | * STD\_ERROR that holds the error state of the API implementation, it shall be 0 for OK if everything is done correctly, and 1 for NOK if anything goes wrong. |
| Covers | * NA |

#### Keypad Get Pressed Key

|  |  |
| --- | --- |
| Requirement ID | Req\_PO1\_DGELV\_GDD\_022\_V01.3 |
| API | STD\_ERROR KEYPAD\_voidGetKeyPressed (u8 \* Copy\_u8PtrKey) |
| Description | This API shall get the keypad pressed key |
| Arguments | * Copy\_u8PtrKey: pointer that holds the value of the pressed key |
| Return | * STD\_ERROR that holds the error state of the API implementation, it shall be 0 for OK if everything is done correctly, and 1 for NOK if anything goes wrong. |
| Covers | * NA |

### Motor

4.1.6.1 Motor Initialize

#### 4.1.6.1 Motor Initialize

|  |  |
| --- | --- |
| Requirement ID | Req\_PO1\_DGELV\_GDD\_023\_V01.3 |
| API | STD\_ERROR MOTOR\_voidMotorInit (void) |
| Description | This API shall initialize the motor |
| Arguments | * void |
| Return | * STD\_ERROR that holds the error state of the API implementation, it shall be 0 for OK if everything is done correctly, and 1 for NOK if anything goes wrong. |
| Covers | * NA |

#### 4.1.6.2 Motor Move ClockWise

|  |  |
| --- | --- |
| Requirement ID | Req\_PO1\_DGELV\_GDD\_024\_V01.3 |
| API | STD\_ERROR MOTOR\_voidMoveClkWise(void) |
| Description | This API shall move the motor in clockwise direction. |
| Arguments | * void |
| Return | * STD\_ERROR that holds the error state of the API implementation, it shall be 0 for OK if everything is done correctly, and 1 for NOK if anything goes wrong. |
| Covers | * NA |

#### 4.1.6.3 Motor Move Anti-ClockWise

|  |  |
| --- | --- |
| Requirement ID | Req\_PO1\_DGELV\_GDD\_025\_V01.3 |
| API | STD\_ERROR MOTOR\_voidMoveAntiClkWise (void) |
| Description | This API shall move the motor in anti-clockwise direction. |
| Arguments | * void |
| Return | * STD\_ERROR that holds the error state of the API implementation, it shall be 0 for OK if everything is done correctly, and 1 for NOK if anything goes wrong. |
| Covers | * NA |

#### 4.1.6.4 Motor Stop

|  |  |
| --- | --- |
| Requirement ID | Req\_PO1\_DGELV\_GDD\_026\_V01.3 |
| API | STD\_ERROR MOTOR\_voidStop (void) |
| Description | This API shall stop the motor |
| Arguments | * void |
| Return | * STD\_ERROR that holds the error state of the API implementation, it shall be 0 for OK if everything is done correctly, and 1 for NOK if anything goes wrong. |
| Covers | * NA |

## 4.3 Application APIs

### 4.3.1 Login Verification

#### 4.3.1.1 User Input

|  |  |
| --- | --- |
| Requirement ID | Req\_PO1\_DGELV\_GDD\_027\_V01.3 |
| API | STD\_ERROR voidUserInput (u8\* Name, u8\* Password) |
| Description | This API shall take user name and password |
| Arguments | * u*8\* Name*: pointer that holds the name of the user, it shall be letter in the range [A, B, C, D]. * *u8\* Password :* pointer that holds the password of the user, it shall be number in the range [0-9]. |
| Return | * STD\_ERROR that holds the error state of the API implementation, it shall be 0 for OK if everything is done correctly, and 1 for NOK if anything goes wrong. |
| Covers |  |

### 4.3.2 Alarm Component

#### 4.3.2.1 Alarm Signal

|  |  |
| --- | --- |
| Requirement ID | Req\_PO1\_DGELV\_GDD\_028\_V01.3 |
| API | STD\_ERROR Alarm\_voidSignal (u8 Copy\_u8State) |
| Description | This API shall run buzzer by specific tone |
| Arguments | * Copy\_u8State: specifies which tone to be run on buzzer, this shall be Tone. That has specific configurations for duty and frequency. |
| Return | * *STD\_ERROR* that holds the error state of the API implementation, it shall be 0 for OK if everything is done correctly, and 1 for NOK if anything goes wrong. |
| Covers |  |

### Display

|  |  |
| --- | --- |
| Requirement ID | * *Req\_PO1\_DGELV\_GDD\_029\_V01.3* |
| API | * + - 1. *STD\_ERROR* DISPLAY\_voidDisplayMsg ( u8 \* Copy\_u8PtrMsg) |
| Description | This API shall display message on LCD |
| Arguments | * Copy\_u8 PtrMsg: pointer that holds the string message to be displayed, it shall display login try status message (succeed or failed) |
| Return | * *STD\_ERROR* that holds the error state of the API implementation, it shall be 0 for OK if everything is done correctly, and 1 for NOK if anything goes wrong. |
| Covers |  |

### Input Reading

4.3.4.1 Elevator Level

|  |  |
| --- | --- |
| Requirement ID | * Req\_PO1\_DGC\_GDD\_030\_V01 |
| API | * *STD\_ERROR* Elevator\_voidLevel(u8 level) |
| Description | This API shall take input from user through keypad buttons inside the elevator to move the elevator to the specified level. |
| Arguments | u8 level : this argument holds the level’s value that the elevator should move to it. |
| Return | * *STD\_ERROR* that holds the error state of the API implementation, it shall be 0 for OK if everything is done correctly, and 1 for NOK if anything goes wrong. |
| Covers |  |

### Elevator Motion Control

4.3.5.1 Elevator direction

|  |  |
| --- | --- |
| Requirement ID | * Req\_PO1\_DGC\_GDD\_031\_V01 |
| API | * *STD\_ERROR* MOVE\_voidDirection(u8 direction) |
| Description | This API shall take input from user through keypad buttons outside the elevator to call the elevator to move up or down. |
| Arguments | u8 direction : this argument holds the value of the direction of the elevator up or down |
| Return | * *STD\_ERROR* that holds the error state of the API implementation, it shall be 0 for OK if everything is done correctly, and 1 for NOK if anything goes wrong. |
| Covers |  |