

# Home Safety and Security System

## 1.0.0

Generated by Doxygen 1.9.6



|  |          |
|--|----------|
| <b>1 Home Security And Safety System</b>       | <b>1</b> |
| <b>2 Module Index</b>                          | <b>3</b> |
| 2.1 Modules . . . . .                          | 3        |
| <b>3 Data Structure Index</b>                  | <b>5</b> |
| 3.1 Data Structures . . . . .                  | 5        |
| <b>4 File Index</b>                            | <b>7</b> |
| 4.1 File List . . . . .                        | 7        |
| <b>5 Module Documentation</b>                  | <b>9</b> |
| 5.1 ADC . . . . .                              | 9        |
| 5.1.1 Detailed Description . . . . .           | 9        |
| 5.2 ADC_Private_Constants . . . . .            | 9        |
| 5.2.1 Detailed Description . . . . .           | 9        |
| 5.3 ADC_Private_Variables . . . . .            | 10       |
| 5.3.1 Detailed Description . . . . .           | 10       |
| 5.4 ADC_Private_Functions . . . . .            | 10       |
| 5.4.1 Detailed Description . . . . .           | 10       |
| 5.4.2 Function Documentation . . . . .         | 10       |
| 5.4.2.1 configReadADC() . . . . .              | 10       |
| 5.5 ADC_Public_Functions . . . . .             | 11       |
| 5.5.1 Detailed Description . . . . .           | 11       |
| 5.5.2 Function Documentation . . . . .         | 11       |
| 5.5.2.1 checkFireDetected() . . . . .          | 11       |
| 5.5.2.2 checkWaterDetected() . . . . .         | 11       |
| 5.5.2.3 initializeAdcSensors() . . . . .       | 12       |
| 5.6 Common_Methods . . . . .                   | 12       |
| 5.6.1 Detailed Description . . . . .           | 12       |
| 5.7 Common_Methods_Public_Functions . . . . .  | 12       |
| 5.7.1 Detailed Description . . . . .           | 13       |
| 5.7.2 Function Documentation . . . . .         | 13       |
| 5.7.2.1 enableClocks() . . . . .               | 13       |
| 5.7.2.2 isInRange() . . . . .                  | 13       |
| 5.7.2.3 wait() . . . . .                       | 14       |
| 5.8 MAIN . . . . .                             | 14       |
| 5.8.1 Detailed Description . . . . .           | 14       |
| 5.9 MAIN_Enum . . . . .                        | 14       |
| 5.9.1 Detailed Description . . . . .           | 14       |
| 5.9.2 Enumeration Type Documentation . . . . . | 15       |
| 5.9.2.1 CurrPage . . . . .                     | 15       |
| 5.10 MAIN_Private_Constants . . . . .          | 15       |
| 5.10.1 Detailed Description . . . . .          | 15       |

|   |    |
|---|----|
| 5.11 MAIN_Private_Variables . . . . .             | 15 |
| 5.11.1 Detailed Description . . . . .             | 16 |
| 5.12 MAIN_Private_FunctionPrototypes . . . . .    | 16 |
| 5.12.1 Detailed Description . . . . .             | 16 |
| 5.13 MAIN_Private_Functions . . . . .             | 16 |
| 5.13.1 Detailed Description . . . . .             | 17 |
| 5.13.2 Function Documentation . . . . .           | 17 |
| 5.13.2.1 Error_Handler() . . . . .                | 17 |
| 5.13.2.2 main() . . . . .                         | 17 |
| 5.13.2.3 SystemClock_Config() . . . . .           | 17 |
| 5.14 MEMBRANE_KEY_PAD . . . . .                   | 19 |
| 5.14.1 Detailed Description . . . . .             | 19 |
| 5.15 MEMBRANE_KEY_PAD_Private_Constants . . . . . | 19 |
| 5.15.1 Detailed Description . . . . .             | 20 |
| 5.16 MEMBRANE_KEY_PAD_Private_Variables . . . . . | 20 |
| 5.16.1 Detailed Description . . . . .             | 20 |
| 5.17 MEMBRANE_KEY_PAD_Structs . . . . .           | 20 |
| 5.17.1 Detailed Description . . . . .             | 20 |
| 5.18 MEMBRANE_KEY_PAD_Private_Functions . . . . . | 20 |
| 5.18.1 Detailed Description . . . . .             | 21 |
| 5.18.2 Function Documentation . . . . .           | 21 |
| 5.18.2.1 getMembraneNum() . . . . .               | 21 |
| 5.18.2.2 initializeMembrane() . . . . .           | 21 |
| 5.18.2.3 readMembrane() . . . . .                 | 22 |
| 5.18.2.4 resetMembranePins() . . . . .            | 22 |
| 5.19 MEMBRANE_KEY_PAD_Public_Functions . . . . .  | 22 |
| 5.19.1 Detailed Description . . . . .             | 23 |
| 5.19.2 Function Documentation . . . . .           | 23 |
| 5.19.2.1 checkMemPin() . . . . .                  | 23 |
| 5.19.2.2 initializeMembranePins() . . . . .       | 23 |
| 5.20 PIN_PAD_PAGE . . . . .                       | 24 |
| 5.20.1 Detailed Description . . . . .             | 24 |
| 5.21 PIN_PAD_PAGE_Private_Constants . . . . .     | 24 |
| 5.21.1 Detailed Description . . . . .             | 24 |
| 5.21.2 Variable Documentation . . . . .           | 24 |
| 5.21.2.1 numPadCoordinates . . . . .              | 24 |
| 5.22 PIN_PAD_PAGE_Private_Variables . . . . .     | 25 |
| 5.22.1 Detailed Description . . . . .             | 25 |
| 5.23 PIN_PAD_PAGE_Private_Functions . . . . .     | 25 |
| 5.23.1 Detailed Description . . . . .             | 25 |
| 5.23.2 Function Documentation . . . . .           | 25 |
| 5.23.2.1 checkCoords() . . . . .                  | 25 |

|   |    |
|---|----|
| 5.23.2.2 checkPassword()                | 26 |
| 5.23.2.3 clearDigits()                  | 26 |
| 5.23.2.4 clearText()                    | 27 |
| 5.23.2.5 deleteDigit()                  | 27 |
| 5.23.2.6 dispCorrectPassword()          | 27 |
| 5.23.2.7 dispDigit()                    | 28 |
| 5.23.2.8 dispDigitSpaces()              | 28 |
| 5.23.2.9 dispWrongPassword()            | 28 |
| 5.23.2.10 drawDigitBox()                | 29 |
| 5.23.2.11 drawNumPad()                  | 29 |
| 5.24 PIN_PAD_PAGE_Public_Functions      | 29 |
| 5.24.1 Detailed Description             | 30 |
| 5.24.2 Function Documentation           | 30 |
| 5.24.2.1 checkPin()                     | 30 |
| 5.24.2.2 displayLockScreen()            | 30 |
| 5.24.2.3 initializeDisplay()            | 31 |
| 5.25 SERVO_MOTOR                        | 31 |
| 5.25.1 Detailed Description             | 31 |
| 5.26 SERVO_MOTOR_Private_Functions      | 31 |
| 5.26.1 Detailed Description             | 31 |
| 5.26.2 Function Documentation           | 32 |
| 5.26.2.1 PB4_Init()                     | 32 |
| 5.27 SERVO_MOTOR_Public_Functions       | 32 |
| 5.27.1 Detailed Description             | 32 |
| 5.27.2 Function Documentation           | 32 |
| 5.27.2.1 initServoMotor()               | 32 |
| 5.27.2.2 lockDoor()                     | 33 |
| 5.27.2.3 unlockDoor()                   | 33 |
| 5.28 STATUS_PAGE                        | 34 |
| 5.28.1 Detailed Description             | 34 |
| 5.29 STATUS_PAGE_Private_Constants      | 34 |
| 5.29.1 Detailed Description             | 34 |
| 5.30 STATUS_PAGE_Private_Variables      | 34 |
| 5.30.1 Detailed Description             | 34 |
| 5.31 STATUS_PAGE_Enum                   | 35 |
| 5.31.1 Detailed Description             | 35 |
| 5.31.2 Enumeration Type Documentation   | 35 |
| 5.31.2.1 StatusPageButtons              | 35 |
| 5.32 STATUS_PAGE_Private_Functions      | 35 |
| 5.32.1 Detailed Description             | 35 |
| 5.32.2 Function Documentation           | 36 |
| 5.32.2.1 statusPage_checkButtonCoords() | 36 |

|   |           |
|---|-----------|
| 5.33 STATUS_PAGE_Public_Functions . . . . .           | 36        |
| 5.33.1 Detailed Description . . . . .                 | 36        |
| 5.33.2 Function Documentation . . . . .               | 36        |
| 5.33.2.1 checkStatusPageTouch() . . . . .             | 36        |
| 5.33.2.2 displayStatusPage() . . . . .                | 37        |
| 5.33.2.3 updateStatusPage() . . . . .                 | 37        |
| 5.34 CORE . . . . .                                   | 38        |
| 5.34.1 Detailed Description . . . . .                 | 38        |
| 5.35 PERIPHERALS . . . . .                            | 38        |
| 5.35.1 Detailed Description . . . . .                 | 38        |
| 5.36 DISPLAY . . . . .                                | 38        |
| 5.36.1 Detailed Description . . . . .                 | 38        |
| <b>6 Data Structure Documentation</b>                 | <b>39</b> |
| 6.1 pin Struct Reference . . . . .                    | 39        |
| 6.1.1 Field Documentation . . . . .                   | 39        |
| 6.1.1.1 gpio . . . . .                                | 39        |
| 6.1.1.2 GPIOx . . . . .                               | 39        |
| <b>7 File Documentation</b>                           | <b>41</b> |
| 7.1 Core/Inc/adcSensors.h File Reference . . . . .    | 41        |
| 7.1.1 Detailed Description . . . . .                  | 41        |
| 7.2 adcSensors.h . . . . .                            | 42        |
| 7.3 Core/Inc/commonMethods.h File Reference . . . . . | 42        |
| 7.3.1 Detailed Description . . . . .                  | 42        |
| 7.4 commonMethods.h . . . . .                         | 42        |
| 7.5 Core/Inc/main.h File Reference . . . . .          | 43        |
| 7.5.1 Detailed Description . . . . .                  | 43        |
| 7.6 main.h . . . . .                                  | 43        |
| 7.7 Core/Inc/membrane.h File Reference . . . . .      | 44        |
| 7.7.1 Detailed Description . . . . .                  | 44        |
| 7.8 membrane.h . . . . .                              | 44        |
| 7.9 Core/Inc/pinpad.h File Reference . . . . .        | 45        |
| 7.9.1 Detailed Description . . . . .                  | 45        |
| 7.10 pinpad.h . . . . .                               | 45        |
| 7.11 Core/Inc/servoMotor.h File Reference . . . . .   | 45        |
| 7.11.1 Detailed Description . . . . .                 | 46        |
| 7.12 servoMotor.h . . . . .                           | 46        |
| 7.13 Core/Inc/statusPage.h File Reference . . . . .   | 46        |
| 7.13.1 Detailed Description . . . . .                 | 47        |
| 7.14 statusPage.h . . . . .                           | 47        |
| 7.15 Core/Src/adcSensors.c File Reference . . . . .   | 47        |
| 7.15.1 Detailed Description . . . . .                 | 48        |

---

|  |           |
|--|-----------|
| 7.16 Core/Src/commonMethods.c File Reference . . . . . | 48        |
| 7.16.1 Detailed Description . . . . .                  | 49        |
| 7.17 Core/Src/main.c File Reference . . . . .          | 49        |
| 7.17.1 Detailed Description . . . . .                  | 50        |
| 7.18 Core/Src/membrane.c File Reference . . . . .      | 50        |
| 7.18.1 Detailed Description . . . . .                  | 51        |
| 7.19 Core/Src/pinpad.c File Reference . . . . .        | 52        |
| 7.19.1 Detailed Description . . . . .                  | 53        |
| 7.20 Core/Src/servoMotor.c File Reference . . . . .    | 53        |
| 7.20.1 Detailed Description . . . . .                  | 53        |
| 7.21 Core/Src/statusPage.c File Reference . . . . .    | 54        |
| 7.21.1 Detailed Description . . . . .                  | 54        |
| <b>Index</b>   | <b>55</b> |





## Chapter 1

# Home Security And Safety System

### Author

Ali Nanji ( [A.Nanji@uea.ac.uk](mailto:A.Nanji@uea.ac.uk) )

Sahil Modak ( [S.Modak@uea.ac.uk](mailto:S.Modak@uea.ac.uk) )

Here you should tell us about how your application works. If it's a game then how to play, any special rules you have, etc. Also, explain any non-trivial design decisions you make, like how your ball gets its position updated, how you designed your buffer for `readchar()`, etc. You should also comment on the stability of your code. Any big bugs should be listed here. Basically, anything that you think we need to know in general about your project should go here.

Our project is a system that intends to protect a house by providing safety and security insights. The "Safety" aspect of the system alerts the user of dangers such as an active fire or water leak, both of which would damage a house.



## Chapter 2

# Module Index

### 2.1 Modules

Here is a list of all modules:

|  |    |
|--|----|
| CORE . . . . .                               | 38 |
| Common_Methods . . . . .                     | 12 |
| Common_Methods_Public_Functions . . . . .    | 12 |
| MAIN . . . . .                               | 14 |
| MAIN_Enum . . . . .                          | 14 |
| MAIN_Private_Constants . . . . .             | 15 |
| MAIN_Private_Variables . . . . .             | 15 |
| MAIN_Private_FunctionPrototypes . . . . .    | 16 |
| MAIN_Private_Functions . . . . .             | 16 |
| PERIPHERALS . . . . .                        | 38 |
| ADC . . . . .                                | 9  |
| ADC_Private_Constants . . . . .              | 9  |
| ADC_Private_Variables . . . . .              | 10 |
| ADC_Private_Functions . . . . .              | 10 |
| ADC_Public_Functions . . . . .               | 11 |
| MEMBRANE_KEY_PAD . . . . .                   | 19 |
| MEMBRANE_KEY_PAD_Private_Constants . . . . . | 19 |
| MEMBRANE_KEY_PAD_Private_Variables . . . . . | 20 |
| MEMBRANE_KEY_PAD_Structs . . . . .           | 20 |
| MEMBRANE_KEY_PAD_Private_Functions . . . . . | 20 |
| MEMBRANE_KEY_PAD_Public_Functions . . . . .  | 22 |
| SERVO_MOTOR . . . . .                        | 31 |
| SERVO_MOTOR_Private_Functions . . . . .      | 31 |
| SERVO_MOTOR_Public_Functions . . . . .       | 32 |
| DISPLAY . . . . .                            | 38 |
| PIN_PAD_PAGE . . . . .                       | 24 |
| PIN_PAD_PAGE_Private_Constants . . . . .     | 24 |
| PIN_PAD_PAGE_Private_Variables . . . . .     | 25 |
| PIN_PAD_PAGE_Private_Functions . . . . .     | 25 |
| PIN_PAD_PAGE_Public_Functions . . . . .      | 29 |
| STATUS_PAGE . . . . .                        | 34 |
| STATUS_PAGE_Private_Constants . . . . .      | 34 |
| STATUS_PAGE_Private_Variables . . . . .      | 34 |
| STATUS_PAGE_Enum . . . . .                   | 35 |
| STATUS_PAGE_Private_Functions . . . . .      | 35 |
| STATUS_PAGE_Public_Functions . . . . .       | 36 |



## Chapter 3

# Data Structure Index

### 3.1 Data Structures

Here are the data structures with brief descriptions:

|                               |                    |
|-------------------------------|--------------------|
| <a href="#">pin</a> . . . . . | <a href="#">39</a> |
|-------------------------------|--------------------|



## Chapter 4

# File Index

### 4.1 File List

Here is a list of all documented files with brief descriptions:

|  |    |
|--|----|
| Core/Inc/ <a href="#">adcSensors.h</a>   |    |
| : Header file for <a href="#">adcSensors.c</a> . . . . .   | 41 |
| Core/Inc/ <a href="#">commonMethods.h</a>  |    |
| : Header file for <a href="#">commonMethods.c</a> . . . . .  | 42 |
| Core/Inc/ <a href="#">main.h</a>   |    |
| : Header for <a href="#">main.c</a> file. This file contains the common defines of the application . . . . . | 43 |
| Core/Inc/ <a href="#">membrane.h</a>   |    |
| : Header file for <a href="#">membrane.c</a> . . . . .   | 44 |
| Core/Inc/ <a href="#">pinpad.h</a>   |    |
| : Header file for <a href="#">pinpad.c</a> . . . . .   | 45 |
| Core/Inc/ <a href="#">servoMotor.h</a>   |    |
| : Header file for <a href="#">servoMotor.c</a> . . . . .   | 45 |
| Core/Inc/ <a href="#">statusPage.h</a>   |    |
| : Header file for <a href="#">statusPage.c</a> . . . . .   | 46 |
| Core/Src/ <a href="#">adcSensors.c</a>   |    |
| : Initializes and handles peripherals that require ADC (Analog to digital conversion) . . . . .              | 47 |
| Core/Src/ <a href="#">commonMethods.c</a>  |    |
| : Defines common methods used by multiple files . . . . .  | 48 |
| Core/Src/ <a href="#">main.c</a>   |    |
| : Main program body . . . . .  | 49 |
| Core/Src/ <a href="#">membrane.c</a>   |    |
| : Initializes and handles membrane keypad . . . . .  | 50 |
| Core/Src/ <a href="#">pinpad.c</a>   |    |
| : Handles Pin Pad page of the UI . . . . .   | 52 |
| Core/Src/ <a href="#">servoMotor.c</a>   |    |
| : Initializes and handles servo motor . . . . .  | 53 |
| Core/Src/ <a href="#">statusPage.c</a>   |    |
| : Handles Status page of the UI . . . . .  | 54 |





## Chapter 5

# Module Documentation

### 5.1 ADC

#### Modules

- [ADC\\_Private\\_Constants](#)
- [ADC\\_Private\\_Variables](#)
- [ADC\\_Private\\_Functions](#)
- [ADC\\_Public\\_Functions](#)

#### Macros

- `#define wait_delay HAL_Delay`

#### 5.1.1 Detailed Description

Handles peripherals that require ADC (Analog to digital conversion)

### 5.2 ADC\_Private\_Constants

#### Variables

- `const float analogReadRange = 1023.0`
- `const float maxVoltage = 5.0`
- `const float waterVoltageThreshold = 3.0`
- `const float fireVoltageThreshold = 4.5`

#### 5.2.1 Detailed Description

Constants defined in [adcSensors.c](#)

## 5.3 ADC\_Private\_Variables

### Variables

- GPIO\_InitTypeDef **GPIO\_InitStruct\_WaterSensor**
- GPIO\_InitTypeDef **GPIO\_InitStruct\_FireSensor**

### 5.3.1 Detailed Description

Variables defined in [adcSensors.c](#)

## 5.4 ADC\_Private\_Functions

### Functions

- float [configReadADC](#) (ADC\_HandleTypeDef \*hadc, uint32\_t channelNum)  
*Read ADC value on provided channel Read adc value on specified channel and return converted voltage reading.*

### 5.4.1 Detailed Description

Private functions defined in [adcSensors.c](#)

### 5.4.2 Function Documentation

#### 5.4.2.1 configReadADC()

```
float configReadADC (  
    ADC_HandleTypeDef * hadc,  
    uint32_t channelNum )
```

Read ADC value on provided channel Read adc value on specified channel and return converted voltage reading.

#### Parameters

|                   |  |
|-------------------|--|
| <i>hadc</i>       | Pointer to the ADC Handle Structure                |
| <i>channelNum</i> | Channel number at which the ADC value will be read |

#### Return values

|                  |                 |
|------------------|-----------------|
| <i>Converted</i> | voltage reading |
|------------------|-----------------|

## 5.5 ADC\_Public\_Functions

### Functions

- void [initializeAdcSensors](#) (ADC\_HandleTypeDef \*hadc)  
*Wrapper method to call initialization methods for ADC peripherals Initialize GPIO pins and ADC.*
- bool [checkWaterDetected](#) (ADC\_HandleTypeDef \*hadc)  
*Checks if any water has been detected.*
- bool [checkFireDetected](#) (ADC\_HandleTypeDef \*hadc)  
*Checks if any flames have been detected.*

### 5.5.1 Detailed Description

Public Functions defined in [adcSensors.c](#)

### 5.5.2 Function Documentation

#### 5.5.2.1 checkFireDetected()

```
bool checkFireDetected (  
    ADC_HandleTypeDef * hadc )
```

Checks if any flames have been detected.

##### Parameters

|             |                                     |
|-------------|-------------------------------------|
| <i>hadc</i> | Pointer to the ADC Handle Structure |
|-------------|-------------------------------------|

##### Return values

|             |   |
|-------------|---|
| <i>True</i> | if flames have been detected, false otherwise |
|-------------|---|

#### 5.5.2.2 checkWaterDetected()

```
bool checkWaterDetected (  
    ADC_HandleTypeDef * hadc )
```

Checks if any water has been detected.

##### Parameters

|             |                                     |
|-------------|-------------------------------------|
| <i>hadc</i> | Pointer to the ADC Handle Structure |
|-------------|-------------------------------------|

## Return values

|             |   |
|-------------|---|
| <i>True</i> | if water has been detected, false otherwise |
|-------------|---|

**5.5.2.3 initializeAdcSensors()**

```
void initializeAdcSensors (
    ADC_HandleTypeDef * hadc )
```

Wrapper method to call initialization methods for ADC peripherals Initialize GPIO pins and ADC.

## Parameters

|             |                                     |
|-------------|-------------------------------------|
| <i>hadc</i> | Pointer to the ADC Handle Structure |
|-------------|-------------------------------------|

## Return values

|             |  |
|-------------|--|
| <i>None</i> |  |
|-------------|--|

**5.6 Common\_Methods****Modules**

- [Common\\_Methods\\_Public\\_Functions](#)

**5.6.1 Detailed Description**

Common Methods used in various files

**5.7 Common\_Methods\_Public\_Functions****Functions**

- void [wait](#) (int delay)  
*Adds a delay to the program.*
- bool [isInRange](#) (int min, int max, int num)  
*Checks if num is between min and max.*
- void [enableClocks](#) (void)  
*Enables GPIO Clocks required for various peripherals.*

### 5.7.1 Detailed Description

Public Functions defined in [commonMethods.c](#)

### 5.7.2 Function Documentation

#### 5.7.2.1 enableClocks()

```
void enableClocks (  
    void )
```

Enables GPIO Clocks required for various peripherals.

##### Parameters

|             |  |
|-------------|--|
| <i>None</i> |  |
|-------------|--|

##### Return values

|             |  |
|-------------|--|
| <i>None</i> |  |
|-------------|--|

#### 5.7.2.2 isInRange()

```
bool isInRange (  
    int min,  
    int max,  
    int num )
```

Checks if num is between min and max.

##### Parameters

|            |  |
|------------|--|
| <i>min</i> | Lower bound of range                                     |
| <i>max</i> | Upper bound of range                                     |
| <i>num</i> | Number to check check if its between the bounds provided |

##### Return values

|             |                                |
|-------------|--------------------------------|
| <i>True</i> | if num is in range, else false |
|-------------|--------------------------------|

### 5.7.2.3 wait()

```
void wait (
    int delay )
```

Adds a delay to the program.

#### Parameters

|              |                                       |
|--------------|---------------------------------------|
| <i>delay</i> | Amount of delay to add to the program |
|--------------|---------------------------------------|

#### Return values

|             |  |
|-------------|--|
| <i>None</i> |  |
|-------------|--|

## 5.8 MAIN

main file

### Modules

- [MAIN\\_Enum](#)
- [MAIN\\_Private\\_Constants](#)
- [MAIN\\_Private\\_Variables](#)
- [MAIN\\_Private\\_FunctionPrototypes](#)
- [MAIN\\_Private\\_Functions](#)

### 5.8.1 Detailed Description

main file

Contains application entry point and main loop

## 5.9 MAIN\_Enum

### Enumerations

- enum [CurrPage](#) { [PINPAD](#) , [STATUS](#) }  
*Enum defined to keep track of current page on UI.*

### 5.9.1 Detailed Description

Enum defined in [main.c](#)

## 5.9.2 Enumeration Type Documentation

### 5.9.2.1 CurrPage

enum [CurrPage](#)

Enum defined to keep track of current page on UI.

Enumerator

|        |   |
|--------|---|
| PINPAD | Indicates that the current UI page is the PINPAD page         |
| STATUS | Indicates that the UI is currently displaying the STATUS page |

## 5.10 MAIN\_Private\_Constants

### Variables

- const uint8\_t **BT\_TX\_MUTE** = 0
- const uint8\_t **BT\_TX\_WATER\_DETECTED** = 25
- const uint8\_t **BT\_TX\_WATER\_NOT\_DETECTED** = 26
- const uint8\_t **BT\_TX\_FIRE\_DETECTED** = 50
- const uint8\_t **BT\_TX\_FIRE\_NOT\_DETECTED** = 51
- const uint8\_t **BT\_TX\_LOCKED** = 75
- const uint8\_t **BT\_TX\_UNLOCKED** = 76
- const uint8\_t **BT\_TX\_BELL\_PRESSED** = 100
- const uint8\_t **BT\_TX\_BELL\_NOT\_PRESSED** = 101
- const uint8\_t **BT\_TX\_DISCARD** = 125
- const uint8\_t **BT\_RX\_MUTE** = 50
- const uint8\_t **BT\_RX\_DISCARD** = 100
- const int **SCREEN\_PIN\_LEN** = 4
- const int **MEMBRANE\_PIN\_LEN** = 4
- const int **DISCARDED\_WARNING\_WAIT\_PERIOD** = 5000

### 5.10.1 Detailed Description

Constants defined in [main.c](#)

## 5.11 MAIN\_Private\_Variables

### Variables

- uint8\_t **RX\_BUFFER**
- CRC\_HandleTypeDef **hcrc**

- DMA2D\_HandleTypeDef **hdma2d**
- UART\_HandleTypeDef **huart7**
- ADC\_HandleTypeDef **hadc**
- GPIO\_InitTypeDef **gpio\_buzzer**
- GPIO\_InitTypeDef **gpio\_touchSensor**
- TIM\_HandleTypeDef **htim3**
- TS\_StateTypeDef **ts**
- char **xTouchStr** [10]
- bool **waterDetected** = false
- bool **fireDetected** = false
- bool **doorLocked** = true
- bool **bellPressed** = false
- bool **muted** = false
- bool **btPreviousWaterTr** = false
- bool **btPreviousFireTr** = false
- bool **btPreviousLockTr** = true
- bool **btPreviousBellTr** = false
- int **lastBtTx** = 0
- int **lastBtTx\_spam** = 0
- bool **showWaterWarning** = false
- bool **showFireWarning** = false
- bool **showDoorLockWarning** = false
- bool **showDoorBellWarning** = false
- int **doorLock\_discardedWarningWait** = 0
- int **doorBell\_discardedWarningWait** = 0
- int **water\_discardedWarningWait** = 0
- int **fire\_discardedWarningWait** = 0

### 5.11.1 Detailed Description

Private Variables defined in [main.c](#)

## 5.12 MAIN\_Private\_FunctionPrototypes

### 5.12.1 Detailed Description

Function Prototypes for functions in [main.c](#)

## 5.13 MAIN\_Private\_Functions

### Functions

- int [main](#) (void)  
*The application entry point.*
- void **HAL\_UART\_RxCpltCallback** (UART\_HandleTypeDef \*huart)
- void [Error\\_Handler](#) (void)  
*This function is executed in case of error occurrence.*
- void [SystemClock\\_Config](#) (void)  
*System Clock Configuration.*



### 5.13.1 Detailed Description

Private Functions defined in [main.c](#)

### 5.13.2 Function Documentation

#### 5.13.2.1 Error\_Handler()

```
void Error_Handler (  
    void )
```

This function is executed in case of error occurrence.

##### Parameters

|      |  |
|------|--|
| None |  |
|------|--|

##### Return values

|      |  |
|------|--|
| None |  |
|------|--|

#### 5.13.2.2 main()

```
int main (  
    void )
```

The application entry point.

##### Parameters

|      |  |
|------|--|
| None |  |
|------|--|

##### Return values

|     |  |
|-----|--|
| int |  |
|-----|--|

#### 5.13.2.3 SystemClock\_Config()

```
void SystemClock_Config (  
    void )
```

System Clock Configuration.

## Return values

|      |  |
|------|--|
| None |  |
|------|--|

System Clock Configuration The system Clock is configured as follow : System Clock source = PLL (HSE) SYSCLK(Hz) = 200000000 HCLK(Hz) = 200000000 AHB Prescaler = 1 APB1 Prescaler = 4 APB2 Prescaler = 2 HSE Frequency(Hz) = 25000000 PLL\_M = 12 PLL\_N = 192 PLL\_P = 2 PLL\_Q = 9 PLLSAI\_P = 8 VDD(V) = 3.3 Main regulator output voltage = Scale1 mode Flash Latency(WS) = 6

## Parameters

|      |  |
|------|--|
| None |  |
|------|--|

## Return values

|      |  |
|------|--|
| None |  |
|------|--|

Configure the main internal regulator output voltage

Initializes the RCC Oscillators according to the specified parameters in the RCC\_OscInitTypeDef structure.

Activate the Over-Drive mode

Initializes the CPU, AHB and APB buses clocks

## 5.14 MEMBRANE\_KEY\_PAD

### Modules

- [MEMBRANE\\_KEY\\_PAD\\_Private\\_Constants](#)
- [MEMBRANE\\_KEY\\_PAD\\_Private\\_Variables](#)
- [MEMBRANE\\_KEY\\_PAD\\_Structs](#)
- [MEMBRANE\\_KEY\\_PAD\\_Private\\_Functions](#)
- [MEMBRANE\\_KEY\\_PAD\\_Public\\_Functions](#)

### 5.14.1 Detailed Description

Handles Membrane Key Pad

## 5.15 MEMBRANE\_KEY\_PAD\_Private\_Constants

### Variables

- const int **currMemPin** = 1234

### 5.15.1 Detailed Description

Constants defined in [membrane.c](#)

## 5.16 MEMBRANE\_KEY\_PAD\_Private\_Variables

### Variables

- GPIO\_InitTypeDef **gpio8**
- GPIO\_InitTypeDef **gpio9**
- GPIO\_InitTypeDef **gpio10**
- GPIO\_InitTypeDef **gpio11**
- GPIO\_InitTypeDef **gpio12**
- GPIO\_InitTypeDef **gpio13**
- GPIO\_InitTypeDef **gpio14**
- GPIO\_InitTypeDef **gpio15**

### 5.16.1 Detailed Description

Variables defined in [membrane.c](#)

## 5.17 MEMBRANE\_KEY\_PAD\_Structs

### Data Structures

- struct [pin](#)

### Variables

- struct [pin](#) pin

### 5.17.1 Detailed Description

Structs defined in [membrane.c](#)

## 5.18 MEMBRANE\_KEY\_PAD\_Private\_Functions

### Functions

- void [resetMembranePins](#) (int start, int stop, GPIO\_PinState value)  
*Resets Membrane pins to give value.*
- void [initializeMembrane](#) (GPIO\_InitTypeDef \*gpio, GPIO\_TypeDef \*GPIOx, uint16\_t GPIO\_Pin, uint16\_t pos)  
*Initializes membrane keypad pins, to input or output.*
- int [getMembraneNum](#) (int col, int row)  
*calculates number pressed depending on membrane keypad input*
- int [readMembrane](#) ()  
*Checks which button on the membrane keypad is pressed.*

### 5.18.1 Detailed Description

Private Functions defined in [membrane.c](#)

### 5.18.2 Function Documentation

#### 5.18.2.1 getMembraneNum()

```
int getMembraneNum (
    int col,
    int row )
```

calculates number pressed depending on membrane keypad input

##### Parameters

|            |                                 |
|------------|---------------------------------|
| <i>row</i> | Row number of button pressed    |
| <i>col</i> | Column number of button pressed |

##### Return values

|               |                                |
|---------------|--------------------------------|
| <i>number</i> | pressed on the membrane keypad |
|---------------|--------------------------------|

#### 5.18.2.2 initializeMembrane()

```
void initializeMembrane (
    GPIO_InitTypeDef * gpio,
    GPIO_TypeDef * GPIOx,
    uint16_t GPIO_Pin,
    uint16_t pos )
```

Initializes membrane keypad pins, to input or output.

##### Parameters

|                 |  |
|-----------------|--|
| <i>gpio</i>     | Pointer to the pin structure                               |
| <i>GPIOx</i>    | Pointer to the gpio type                                   |
| <i>GPIO_Pin</i> | Pin number @Param pos Position of pin in the allPins array |

##### Return values

|             |  |
|-------------|--|
| <i>None</i> |  |
|-------------|--|

### 5.18.2.3 readMembrane()

```
int readMembrane ( )
```

Checks which button on the membrane keypad is pressed.

#### Parameters

|      |  |
|------|--|
| None |  |
|------|--|

#### Return values

|       |  |
|-------|--|
| Value | of the button pressed on the membrane keypad |
|-------|--|

### 5.18.2.4 resetMembranePins()

```
void resetMembranePins (
    int start,
    int stop,
    GPIO_PinState value )
```

Resets Membrane pins to give value.

#### Parameters

|              |                              |
|--------------|------------------------------|
| <i>start</i> | Start index of list to reset |
| <i>stop</i>  | Last index of list to reset  |
| <i>value</i> | New pin state for pins       |

#### Return values

|      |  |
|------|--|
| None |  |
|------|--|

## 5.19 MEMBRANE\_KEY\_PAD\_Public\_Functions

### Functions

- void [initializeMembranePins](#) (void)  
*Initializes all pins that will be used by the membrane keypad The membrane keypad uses pin D8 - D15 on the board. It is configured as follows: Mode = Input/Output Pull = PULL DOWN Speed = SPEED HIGH.*
- bool [checkMemPin](#) (int \*currentMemDigitCount, int \*currentMemPin, int membranePinLen)  
*Checks if the PIN entered on the keypad is correct.*

### 5.19.1 Detailed Description

Public Functions defined in [membrane.c](#)

### 5.19.2 Function Documentation

#### 5.19.2.1 checkMemPin()

```
bool checkMemPin (
    int * currentMemDigitCount,
    int * currentMemPin,
    int membranePinLen )
```

Checks if the PIN entered on the keypad is correct.

##### Parameters

|                             |  |
|-----------------------------|--|
| <i>currentMemDigitCount</i> | Pointer to the count of digits currently entered |
| <i>currentMemPin</i>        | Current value of Pin entered                     |
| <i>membranePinLen</i>       | Length of PIN to be verified                     |

##### Return values

|             |                                    |
|-------------|------------------------------------|
| <i>True</i> | if PIN is correct, False otherwise |
|-------------|------------------------------------|

#### 5.19.2.2 initializeMembranePins()

```
void initializeMembranePins (
    void )
```

Initializes all pins that will be used by the membrane keypad The membrane keypad uses pin D8 - D15 on the board. It is configured as follows: Mode = Input/Output Pull = PULL DOWN Speed = SPEED HIGH.

##### Parameters

|             |  |
|-------------|--|
| <i>None</i> |  |
|-------------|--|

##### Return values

|             |  |
|-------------|--|
| <i>None</i> |  |
|-------------|--|

## 5.20 PIN\_PAD\_PAGE

### Modules

- [PIN\\_PAD\\_PAGE\\_Private\\_Constants](#)
- [PIN\\_PAD\\_PAGE\\_Private\\_Variables](#)
- [PIN\\_PAD\\_PAGE\\_Private\\_Functions](#)
- [PIN\\_PAD\\_PAGE\\_Public\\_Functions](#)

### Macros

- `#define wait_delay HAL_Delay`

### 5.20.1 Detailed Description

Handle Pin Pad page on the UI

## 5.21 PIN\_PAD\_PAGE\_Private\_Constants

### Variables

- const int **dispDigitsLimit** = 4
- const int **numPadBoxWidth** = 157
- const int **numPadBoxHeight** = 38
- const int **correctPassword** [4] = { 0, 0, 0, 0 }
- const int [numPadCoordinates](#) [10][3]

### 5.21.1 Detailed Description

Constants defined in [pinpad.c](#)

### 5.21.2 Variable Documentation

#### 5.21.2.1 numPadCoordinates

```
const int numPadCoordinates[10][3]
```

#### Initial value:

```
= { { 160, 231, 0 },
    { 1, 111, 1 }, { 160, 111, 2 }, { 319, 111, 3 },
    { 1, 151, 4 }, { 160, 151, 5 }, { 319, 151, 6 },
    { 1, 191, 7 }, { 160, 191, 8 }, { 319, 191, 9 }
}
```



## 5.22 PIN\_PAD\_PAGE\_Private\_Variables

### Variables

- TS\_StateTypeDef **tsc\_state**

### 5.22.1 Detailed Description

Variables defined in [pinpad.c](#)

## 5.23 PIN\_PAD\_PAGE\_Private\_Functions

### Functions

- void [drawDigitBox](#) (int x, int y, int digit)  
*Draw a rectangle for a key on the pin pad.*
- int [checkCoords](#) (int x, int y)  
*Check the coordinate of the user's touch to see what key has been pressed.*
- void [drawNumPad](#) (void)  
*Draw the entire num pad.*
- void [dispDigitSpaces](#) (void)  
*Draw dashes to display current pin.*
- void [dispDigit](#) (int num, int currentDispDigitCount)  
*Display the digit the user touched on.*
- void [deleteDigit](#) (int currentDispDigitCount)  
*Delete one digit from current pin.*
- void [clearDigits](#) (void)  
*Delete all digits from current pin.*
- void [clearText](#) (void)  
*Clear any banner.*
- void [dispWrongPassword](#) (void)  
*Display "Wrong Password" banner.*
- void [dispCorrectPassword](#) (void)  
*Display "Correct Password" banner.*
- bool [checkPassword](#) (int currentDispDigits[], int \*currentDispDigitCount)  
*Check if the currently input pin is correct.*

### 5.23.1 Detailed Description

Private functions defined in [pinpad.c](#)

### 5.23.2 Function Documentation

#### 5.23.2.1 checkCoords()

```
int checkCoords (  
    int x,  
    int y )
```

Check the coordinate of the user's touch to see what key has been pressed.

**Parameters**

|          |                         |
|----------|-------------------------|
| <i>x</i> | X position of the touch |
| <i>y</i> | Y position of the touch |

**Return values**

|            |         |
|------------|---------|
| <i>Key</i> | touched |
|------------|---------|

**5.23.2.2 checkPassword()**

```
bool checkPassword (
    int currentDispDigits[],
    int * currentDispDigitCount )
```

Check if the currently input pin is correct.

**Parameters**

|                              |                                     |
|------------------------------|-------------------------------------|
| <i>currentDispDigits</i>     | Array of currently input pin digits |
| <i>currentDispDigitCount</i> | Pointer to current pin length count |

**Return values**

|             |   |
|-------------|---|
| <i>True</i> | if password is correct, false otherwise |
|-------------|---|

**5.23.2.3 clearDigits()**

```
void clearDigits (
    void )
```

Delete all digits from current pin.

**Parameters**

|             |  |
|-------------|--|
| <i>None</i> |  |
|-------------|--|

**Return values**

|             |  |
|-------------|--|
| <i>None</i> |  |
|-------------|--|

#### 5.23.2.4 clearText()

```
void clearText (
    void )
```

Clear any banner.

##### Parameters

|      |  |
|------|--|
| None |  |
|------|--|

##### Return values

|      |  |
|------|--|
| None |  |
|------|--|

#### 5.23.2.5 deleteDigit()

```
void deleteDigit (
    int currentDispDigitCount )
```

Delete one digit from current pin.

##### Parameters

|                              |                                |
|------------------------------|--------------------------------|
| <i>currentDispDigitCount</i> | Position to delete number from |
|------------------------------|--------------------------------|

##### Return values

|      |  |
|------|--|
| None |  |
|------|--|

#### 5.23.2.6 dispCorrectPassword()

```
void dispCorrectPassword (
    void )
```

Display "Correct Password" banner.

##### Parameters

|      |  |
|------|--|
| None |  |
|------|--|

##### Return values

|      |  |
|------|--|
| None |  |
|------|--|

### 5.23.2.7 dispDigit()

```
void dispDigit (
    int num,
    int currentDispDigitCount )
```

Display the digit the user touched on.

#### Parameters

|                              |                               |
|------------------------------|-------------------------------|
| <i>num</i>                   | Number to display             |
| <i>currentDispDigitCount</i> | Position to display number at |

#### Return values

|             |  |
|-------------|--|
| <i>None</i> |  |
|-------------|--|

### 5.23.2.8 dispDigitSpaces()

```
void dispDigitSpaces (
    void )
```

Draw dashes to display current pin.

#### Parameters

|             |  |
|-------------|--|
| <i>None</i> |  |
|-------------|--|

#### Return values

|             |  |
|-------------|--|
| <i>None</i> |  |
|-------------|--|

### 5.23.2.9 dispWrongPassword()

```
void dispWrongPassword (
    void )
```

Display "Wrong Password" banner.

#### Parameters

|             |  |
|-------------|--|
| <i>None</i> |  |
|-------------|--|

## Return values

|             |  |
|-------------|--|
| <i>None</i> |  |
|-------------|--|

**5.23.2.10 drawDigitBox()**

```
void drawDigitBox (
    int x,
    int y,
    int digit )
```

Draw a rectangle for a key on the pin pad.

## Parameters

|              |                                   |
|--------------|-----------------------------------|
| <i>x</i>     | Top left x position               |
| <i>y</i>     | Top Left y position               |
| <i>digit</i> | Number to display inside rectagle |

## Return values

|             |  |
|-------------|--|
| <i>None</i> |  |
|-------------|--|

**5.23.2.11 drawNumPad()**

```
void drawNumPad (
    void )
```

Draw the entire num pad.

## Parameters

|             |  |
|-------------|--|
| <i>None</i> |  |
|-------------|--|

## Return values

|             |  |
|-------------|--|
| <i>None</i> |  |
|-------------|--|

**5.24 PIN\_PAD\_PAGE\_Public\_Functions****Functions**

- void [displayLockScreen](#) (void)

*Draw the entire lock screen.*

- void `initializeDisplay` (void)

*Initialize everything needed for LCD and touch.*

- bool `checkPin` (int \*currentDispDigitCount, int \*currentDispDigits)

*Wrapper method called from main loop to check current pin.*

### 5.24.1 Detailed Description

Public functions defined in `pinpad.c`

### 5.24.2 Function Documentation

#### 5.24.2.1 `checkPin()`

```
bool checkPin (
    int * currentDispDigitCount,
    int * currentDispDigits )
```

Wrapper method called from main loop to check current pin.

This method contains all the main logic for the Pin Pad page. It checks if the user has currently tapped any key and does corresponding action. The method also remains in a loop until the user stops touching the screen to prevent unintentional touches.

#### Parameters

|                                    |                                     |
|------------------------------------|-------------------------------------|
| <code>currentDispDigits</code>     | Array of currently input pin digits |
| <code>currentDispDigitCount</code> | Pointer to current pin length count |

#### Return values

|                   |  |
|-------------------|--|
| <code>None</code> |  |
|-------------------|--|

#### 5.24.2.2 `displayLockScreen()`

```
void displayLockScreen (
    void )
```

Draw the entire lock screen.

#### Parameters

|                   |  |
|-------------------|--|
| <code>None</code> |  |
|-------------------|--|

## Return values

|      |  |
|------|--|
| None |  |
|------|--|

**5.24.2.3 initializeDisplay()**

```
void initializeDisplay (
    void )
```

Initialize everything needed for LCD and touch.

## Parameters

|      |  |
|------|--|
| None |  |
|------|--|

## Return values

|      |  |
|------|--|
| None |  |
|------|--|

**5.25 SERVO\_MOTOR****Modules**

- [SERVO\\_MOTOR\\_Private\\_Functions](#)
- [SERVO\\_MOTOR\\_Public\\_Functions](#)

**5.25.1 Detailed Description**

Handles Servo Motor

**5.26 SERVO\_MOTOR\_Private\_Functions****Functions**

- void [PB4\\_Init](#) (void)

*Servo Motor Pin PB4 Initialization Function The Servo Motor uses pin D3 on the board. It is configured as follows: Mode = Alternate Function Push Pull Pull = No Pull Speed = SPEED HIGH Pin = PIN 4 Alternate = TIM3 Alternate Function.*

**5.26.1 Detailed Description**

Private Functions defined in [servoMotor.c](#)

## 5.26.2 Function Documentation

### 5.26.2.1 PB4\_Init()

```
void PB4_Init (
    void )
```

Servo Motor Pin PB4 Initialization Function The Servo Motor uses pin D3 on the board. It is configured as follows: Mode = Alternate Function Push Pull Pull = No Pull Speed = SPEED HIGH Pin = PIN 4 Alternate = TIM3 Alternate Function.

#### Parameters

|      |  |
|------|--|
| None |  |
|------|--|

#### Return values

|      |  |
|------|--|
| None |  |
|------|--|

## 5.27 SERVO\_MOTOR\_Public\_Functions

### Functions

- void [initServoMotor](#) (TIM\_HandleTypeDef \*htim3)  
*Wrapper method for initializations of the servo motor.*
- void [lockDoor](#) (TIM\_HandleTypeDef \*htim3)  
*Moves servo motor arm to 90 Degree position to lock the door.*
- void [unlockDoor](#) (TIM\_HandleTypeDef \*htim3)  
*Moves servo motor arm to 180 Degree position to unlock the door.*

### 5.27.1 Detailed Description

Public Functions defined in [servoMotor.c](#)

## 5.27.2 Function Documentation

### 5.27.2.1 initServoMotor()

```
void initServoMotor (
    TIM_HandleTypeDef * htim3 )
```

Wrapper method for initializations of the servo motor.



## Parameters

|              |                                  |
|--------------|----------------------------------|
| <i>htim3</i> | Pointer to TIM3 Handle Structure |
|--------------|----------------------------------|

## Return values

|             |  |
|-------------|--|
| <i>None</i> |  |
|-------------|--|

**5.27.2.2 lockDoor()**

```
void lockDoor (
    TIM_HandleTypeDef * htim3 )
```

Moves servo motor arm to 90 Degree position to lock the door.

## Parameters

|              |                                  |
|--------------|----------------------------------|
| <i>htim3</i> | Pointer to TIM3 Handle Structure |
|--------------|----------------------------------|

## Return values

|             |  |
|-------------|--|
| <i>None</i> |  |
|-------------|--|

**5.27.2.3 unlockDoor()**

```
void unlockDoor (
    TIM_HandleTypeDef * htim3 )
```

Moves servo motor arm to 180 Degree position to unlock the door.

## Parameters

|              |                                  |
|--------------|----------------------------------|
| <i>htim3</i> | Pointer to TIM3 Handle Structure |
|--------------|----------------------------------|

## Return values

|             |  |
|-------------|--|
| <i>None</i> |  |
|-------------|--|

## 5.28 STATUS\_PAGE

### Modules

- [STATUS\\_PAGE\\_Private\\_Constants](#)
- [STATUS\\_PAGE\\_Private\\_Variables](#)
- [STATUS\\_PAGE\\_Enum](#)
- [STATUS\\_PAGE\\_Private\\_Functions](#)
- [STATUS\\_PAGE\\_Public\\_Functions](#)

### Macros

- `#define wait_delay HAL_Delay`

#### 5.28.1 Detailed Description

Handle Status page

## 5.29 STATUS\_PAGE\_Private\_Constants

### Variables

- `const int statusPage_lineYCoord [4] = {70, 105, 140, 175}`
- `const int statusPage_statusLabelXCoord = 10`
- `const int statusPage_statusXCoord = 305`
- `const int statusPage_buttonsXCoord [3] = {20, 190, 310}`
- `const int statusPage_buttonsYCoord = 220`
- `const int statusPage_buttonsHeight = 40`
- `const int statusPage_buttonsWidth = 150`
- `const int statusPage_lockButtonHeight = 40`
- `const int statusPage_lockButtonWidth = 105`

#### 5.29.1 Detailed Description

Constants defined in [statusPage.c](#)

## 5.30 STATUS\_PAGE\_Private\_Variables

### Variables

- `TS_StateTypeDef tsc_state_statusPage`

#### 5.30.1 Detailed Description

Variables defined in [statusPage.c](#)

## 5.31 STATUS\_PAGE\_Enum

### Enumerations

- enum [StatusPageButtons](#) {  
    [LOCK](#) , [MUTE](#) , [DISCARD](#) , [DOOR\\_LOCK](#) ,  
    [NONE](#) }

*Enum defined to communicate what button has been pressed on the Status page.*

#### 5.31.1 Detailed Description

Enums defined in [statusPage.c](#)

#### 5.31.2 Enumeration Type Documentation

##### 5.31.2.1 StatusPageButtons

enum [StatusPageButtons](#)

Enum defined to communicate what button has been pressed on the Status page.

##### Enumerator

|           |  |
|-----------|--|
| LOCK      | Lock display screen  |
| MUTE      | Mute Warnings (Display warnings but mute sound)                              |
| DISCARD   | Discard Warnings (Hide all warnings. Door Lock warning can not be discarded) |
| DOOR_LOCK | Lock Front Door  |
| NONE      | No Button Pressed  |

## 5.32 STATUS\_PAGE\_Private\_Functions

### Functions

- enum [StatusPageButtons](#) [statusPage\\_checkButtonCoords](#) (int x, int y)

*Check what button has been clicked on by the user.*

#### 5.32.1 Detailed Description

Private Functions defined in [statusPage.c](#)

## 5.32.2 Function Documentation

### 5.32.2.1 statusPage\_checkButtonCoords()

```
enum StatusPageButtons statusPage_checkButtonCoords (
    int x,
    int y )
```

Check what button has been clicked on by the user.

#### Parameters

|          |                         |
|----------|-------------------------|
| <i>x</i> | X position of the touch |
| <i>y</i> | Y position of the touch |

#### Return values

|             |                         |
|-------------|-------------------------|
| <i>Enum</i> | value of button touched |
|-------------|-------------------------|

## 5.33 STATUS\_PAGE\_Public\_Functions

### Functions

- void [displayStatusPage](#) (bool muted)  
*Draw status page and all status labels.*
- void [updateStatusPage](#) (bool waterLeak, bool fire, bool doorLocked, bool bellPressed, bool muted)  
*Update status page.*
- enum [StatusPageButtons](#) [checkStatusPageTouch](#) (void)  
*Check if any buttons have been pressed on the status page.*

### 5.33.1 Detailed Description

Public Functions defined in [statusPage.c](#)

## 5.33.2 Function Documentation

### 5.33.2.1 checkStatusPageTouch()

```
enum StatusPageButtons checkStatusPageTouch (
    void )
```

Check if any buttons have been pressed on the status page.

## Parameters

|             |  |
|-------------|--|
| <i>None</i> |  |
|-------------|--|

## Return values

|             |                             |
|-------------|-----------------------------|
| <i>Enum</i> | value of the button pressed |
|-------------|-----------------------------|

**5.33.2.2 displayStatusPage()**

```
void displayStatusPage (  
    bool muted )
```

Draw status page and all status labels.

## Parameters

|              |  |
|--------------|--|
| <i>muted</i> | Boolean indicating if device is currently muted or not |
|--------------|--|

## Return values

|             |  |
|-------------|--|
| <i>None</i> |  |
|-------------|--|

**5.33.2.3 updateStatusPage()**

```
void updateStatusPage (  
    bool waterLeak,  
    bool fire,  
    bool doorLocked,  
    bool bellPressed,  
    bool muted )
```

Update status page.

Update warnings for each label and the mute button (from mute to unmute or vice versa)

## Parameters

|                    |  |
|--------------------|--|
| <i>waterLeak</i>   | Boolean indicating if the water leak warning needs to be displayed |
| <i>fire</i>        | Boolean indicating if the fire warning needs to be displayed       |
| <i>doorLocked</i>  | Boolean indicating if the door is locked or not                    |
| <i>bellPressed</i> | Boolean indicating if the bell is pressed or not                   |
| <i>muted</i>       | Boolean indicating if the user has muted the device or not         |

Return values

|             |  |
|-------------|--|
| <i>None</i> |  |
|-------------|--|

## 5.34 CORE

### Modules

- [Common\\_Methods](#)
- [MAIN](#)
  - main file*
- [PERIPHERALS](#)
- [DISPLAY](#)

#### 5.34.1 Detailed Description

All modules implemented by us excluding all libraries

## 5.35 PERIPHERALS

### Modules

- [ADC](#)
- [MEMBRANE\\_KEY\\_PAD](#)
- [SERVO\\_MOTOR](#)

#### 5.35.1 Detailed Description

All modules that initialize and handle the various peripherals used in this project

## 5.36 DISPLAY

### Modules

- [PIN\\_PAD\\_PAGE](#)
- [STATUS\\_PAGE](#)

#### 5.36.1 Detailed Description

Contains modules that handle the various UI pages

## Chapter 6

# Data Structure Documentation

### 6.1 pin Struct Reference

#### Data Fields

- GPIO\_InitTypeDef \* [gpio](#)
- GPIO\_TypeDef \* [GPIOx](#)

#### 6.1.1 Field Documentation

##### 6.1.1.1 gpio

GPIO\_InitTypeDef\* gpio

GPIO Init structure definition

##### 6.1.1.2 GPIOx

GPIO\_TypeDef\* GPIOx

GPIO Type

The documentation for this struct was generated from the following file:

- Core/Src/[membrane.c](#)





# Chapter 7

## File Documentation

### 7.1 Core/Inc/adcSensors.h File Reference

: Header file for [adcSensors.c](#)

```
#include <stdbool.h>
#include "stm32f7xx_hal.h"
```

#### Functions

- void [initializeAdcSensors](#) (ADC\_HandleTypeDef \*hadc)  
*Wrapper method to call initialization methods for ADC peripherals Initialize GPIO pins and ADC.*
- bool [checkWaterDetected](#) (ADC\_HandleTypeDef \*hadc\_waterSensor)  
*Checks if any water has been detected.*
- bool [checkFireDetected](#) (ADC\_HandleTypeDef \*hadc\_fireSensor)  
*Checks if any flames have been detected.*

#### 7.1.1 Detailed Description

: Header file for [adcSensors.c](#)

##### Author

- : Sahil Modak
- : Ali Nanji

This file contains the function prototypes for all public functions in `adcSensor.c`

## 7.2 adcSensors.h

[Go to the documentation of this file.](#)

```
00001
00012 #include <stdbool.h>
00013 #include "stm32f7xx_hal.h"
00014
00015 // Call initialization methods for adc peripherals
00016 void initializeAdcSensors(ADC_HandleTypeDef *hadc);
00017
00018 // Check if any water has been detected
00019 bool checkWaterDetected(ADC_HandleTypeDef *hadc_waterSensor);
00020
00021 // Check if any fire has been detected
00022 bool checkFireDetected(ADC_HandleTypeDef *hadc_fireSensor);
```

## 7.3 Core/Inc/commonMethods.h File Reference

: Header file for [commonMethods.c](#).

```
#include <stdbool.h>
```

### Functions

- void [wait](#) (int delay)  
*Adds a delay to the program.*
- bool [isInRange](#) (int min, int max, int num)  
*Checks if num is between min and max.*
- void [enableClocks](#) (void)  
*Enables GPIO Clocks required for various peripherals.*

### 7.3.1 Detailed Description

: Header file for [commonMethods.c](#).

#### Author

: Sahil Modak  
: Ali Nanji

This file contains the function prototypes for all public functions in [commonMethods.c](#)

## 7.4 commonMethods.h

[Go to the documentation of this file.](#)

```
00001
00012 #include <stdbool.h>
00013
00014 // Adds a delay to the program
00015 void wait(int delay);
00016
00017 // Checks if num is between min and max
00018 bool isInRange(int min, int max, int num);
00019
00020 // Enable gpio clocks needed for peripherals
00021 void enableClocks(void);
```

## 7.5 Core/Inc/main.h File Reference

: Header for [main.c](#) file. This file contains the common defines of the application.

```
#include "stm32f7xx_hal.h"
#include "stm32746g_discovery.h"
#include "stm32746g_discovery_sdram.h"
#include "stm32746g_discovery_ts.h"
#include "stm32746g_discovery_lcd.h"
```

### Functions

- void [Error\\_Handler](#) (void)

*This function is executed in case of error occurrence.*

### 7.5.1 Detailed Description

: Header for [main.c](#) file. This file contains the common defines of the application.

#### Attention

Copyright (c) 2023 STMicroelectronics. All rights reserved.

This software is licensed under terms that can be found in the LICENSE file in the root directory of this software component. If no LICENSE file comes with this software, it is provided AS-IS.

## 7.6 main.h

[Go to the documentation of this file.](#)

```
00001 /* USER CODE BEGIN Header */
00019 /* USER CODE END Header */
00020
00021 /* Define to prevent recursive inclusion -----*/
00022 #ifndef __MAIN_H
00023 #define __MAIN_H
00024
00025 #ifdef __cplusplus
00026 extern "C" {
00027 #endif
00028
00029 /* Includes -----*/
00030 #include "stm32f7xx_hal.h"
00031
00032 #include "stm32746g_discovery.h"
00033 #include "stm32746g_discovery_sdram.h"
00034 #include "stm32746g_discovery_ts.h"
00035 #include "stm32746g_discovery_lcd.h"
00036
00037 /* Private includes -----*/
00038 /* USER CODE BEGIN Includes */
00039
00040 /* USER CODE END Includes */
00041
00042 /* Exported types -----*/
00043 /* USER CODE BEGIN ET */
00044
00045 /* USER CODE END ET */
```

```

00046
00047 /* Exported constants -----*/
00048 /* USER CODE BEGIN EC */
00049
00050 /* USER CODE END EC */
00051
00052 /* Exported macro -----*/
00053 /* USER CODE BEGIN EM */
00054
00055 /* USER CODE END EM */
00056
00057 /* Exported functions prototypes -----*/
00058 void Error\_Handler(void);
00059
00060 /* USER CODE BEGIN EFP */
00061
00062 /* USER CODE END EFP */
00063
00064 /* Private defines -----*/
00065 /* USER CODE BEGIN Private defines */
00066
00067 /* USER CODE END Private defines */
00068
00069 #ifdef __cplusplus
00070 }
00071 #endif
00072
00073 #endif /* __MAIN_H */

```

## 7.7 Core/Inc/membrane.h File Reference

: Header file for [membrane.c](#).

```
#include <stdbool.h>
```

### Functions

- void [initializeMembranePins](#) (void)  
*Initializes all pins that will be used by the membrane keypad The membrane keypad uses pin D8 - D15 on the board. It is configured as follows: Mode = Input/Output Pull = PULL DOWN Speed = SPEED HIGH.*
- bool [checkMemPin](#) (int \*currentMemDigitCount, int \*currentMemPin, int membranePinLen)  
*Checks if the PIN entered on the keypad is correct.*

### 7.7.1 Detailed Description

: Header file for [membrane.c](#).

Author

: Sahil Modak

This file contains the function prototypes for all public functions in [membrane.c](#)

## 7.8 membrane.h

[Go to the documentation of this file.](#)

```

00001
00011 #include <stdbool.h>
00012
00013 void initializeMembranePins (void);
00014 bool checkMemPin(int *currentMemDigitCount, int *currentMemPin, int membranePinLen);

```

## 7.9 Core/Inc/pinpad.h File Reference

: Header file for [pinpad.c](#).

```
#include <stdbool.h>
```

### Functions

- void [initializeDisplay](#) (void)  
*Initialize everything needed for LCD and touch.*
- void [displayLockScreen](#) (void)  
*Draw the entire lock screen.*
- bool [checkPin](#) (int \*currentDispDigitCount, int \*currentDispDigits)  
*Wrapper method called from main loop to check current pin.*

### 7.9.1 Detailed Description

: Header file for [pinpad.c](#).

Author

: Ali Nanji

This file contains the function prototypes for all public functions in [pinpad.c](#)

## 7.10 pinpad.h

[Go to the documentation of this file.](#)

```
00001
00011 #include <stdbool.h>
00012
00013 // Call all initialization methods for touch and display
00014 void initializeDisplay(void);
00015
00016 // Display lockscreen with pinpad
00017 void displayLockScreen(void);
00018
00019 // Check if the user has touched the screen, and if they have, is the currently inputted value the
    correct pin
00020 // Returns true of the pin is correct and false otherwise
00021 bool checkPin(int *currentDispDigitCount, int *currentDispDigits);
```

## 7.11 Core/Inc/servoMotor.h File Reference

: Header file for [servoMotor.c](#).

```
#include "stm32f7xx_hal.h"
```

## Functions

- void `initServoMotor` (TIM\_HandleTypeDef \*htim3)  
*Wrapper method for initializations of the servo motor.*
- void `lockDoor` (TIM\_HandleTypeDef \*htim3)  
*Moves servo motor arm to 90 Degree position to lock the door.*
- void `unlockDoor` (TIM\_HandleTypeDef \*htim3)  
*Moves servo motor arm to 180 Degree position to unlock the door.*

### 7.11.1 Detailed Description

: Header file for `servoMotor.c`.

Author

: Sahil Modak

This file contains the function prototypes for all public functions in `servoMotor.c`

## 7.12 servoMotor.h

[Go to the documentation of this file.](#)

```
00001
00011 #include "stm32f7xx_hal.h"
00012
00013 // Initializations for servo motor
00014 void initServoMotor(TIM_HandleTypeDef *htim3);
00015
00016 // Set servo motor angle to lock the door
00017 void lockDoor(TIM_HandleTypeDef *htim3);
00018
00019 // Set servo motor angle to unlock the door
00020 void unlockDoor(TIM_HandleTypeDef *htim3);
```

## 7.13 Core/Inc/statusPage.h File Reference

: Header file for `statusPage.c`.

```
#include <stdbool.h>
```

## Enumerations

- enum `StatusPageButtons` {  
    `LOCK` , `MUTE` , `DISCARD` , `DOOR_LOCK` ,  
    `NONE` }  
*Enum defined to communicate what button has been pressed on the Status page.*

## Functions

- void [displayStatusPage](#) (bool muted)  
*Draw status page and all status labels.*
- void [updateStatusPage](#) (bool waterLeak, bool fire, bool doorLocked, bool bellPressed, bool muted)  
*Update status page.*
- enum [StatusPageButtons](#) [checkStatusPageTouch](#) (void)  
*Check if any buttons have been pressed on the status page.*

### 7.13.1 Detailed Description

: Header file for [statusPage.c](#).

Author

: Ali Nanji

This file contains the function prototypes for all public functions in [statusPage.c](#), and also defines the StatusPageButtons Enum.

## 7.14 statusPage.h

[Go to the documentation of this file.](#)

```
00001
00012 #include <stdbool.h>
00013
00021 enum StatusPageButtons {
00022     LOCK,
00023     MUTE,
00024     DISCARD,
00025     DOOR_LOCK,
00026     NONE
00027 };
00032 // Display basic status page without any warnings
00033 void displayStatusPage(bool muted);
00034
00035 // Update status page with warnings
00036 void updateStatusPage(bool waterLeak, bool fire, bool doorLocked, bool bellPressed, bool muted);
00037
00038 // Check if any button on the status page has been clicked on and return enum defined above
    accordingly
00039 enum StatusPageButtons checkStatusPageTouch(void);
```

## 7.15 Core/Src/adcSensors.c File Reference

: Initializes and handles peripherals that require ADC (Analog to digital conversion)

```
#include "adcSensors.h"
#include "stm32f7xx_hal.h"
#include <stdio.h>
```

## Macros

- #define **wait\_delay** HAL\_Delay

## Functions

- float `configReadADC` (ADC\_HandleTypeDef \*hadc, uint32\_t channelNum)  
*Read ADC value on provided channel Read adc value on specified channel and return converted voltage reading.*
- void `initializeAdcSensors` (ADC\_HandleTypeDef \*hadc)  
*Wrapper method to call initialization methods for ADC peripherals Initialize GPIO pins and ADC.*
- bool `checkWaterDetected` (ADC\_HandleTypeDef \*hadc)  
*Checks if any water has been detected.*
- bool `checkFireDetected` (ADC\_HandleTypeDef \*hadc)  
*Checks if any flames have been detected.*

## Variables

- const float `analogReadRange` = 1023.0
- const float `maxVoltage` = 5.0
- const float `waterVoltageThreshold` = 3.0
- const float `fireVoltageThreshold` = 4.5
- GPIO\_InitTypeDef `GPIO_InitStruct_WaterSensor`
- GPIO\_InitTypeDef `GPIO_InitStruct_FireSensor`

### 7.15.1 Detailed Description

: Initializes and handles peripherals that require ADC (Analog to digital conversion)

Author

: Sahil Modak  
: Ali Nanji

This file contains all the methods required to initialize and handle the peripherals that require ADC, which are the water sensor and the flame sensor. The public functions' prototypes have been defined in adcSensor.h

## 7.16 Core/Src/commonMethods.c File Reference

: Defines common methods used by multiple files

```
#include "commonMethods.h"
#include "stm32f7xx_hal.h"
```

## Functions

- void `wait` (int delay)  
*Adds a delay to the program.*
- bool `isInRange` (int min, int max, int num)  
*Checks if num is between min and max.*
- void `enableClocks` (void)  
*Enables GPIO Clocks required for various peripherals.*



### 7.16.1 Detailed Description

: Defines common methods used by multiple files

#### Author

: Sahil Modak

: Ali Nanji

## 7.17 Core/Src/main.c File Reference

: Main program body

```
#include "main.h"
#include "pinpad.h"
#include "statusPage.h"
#include "commonMethods.h"
#include "membrane.h"
#include "adcSensors.h"
#include "servoMotor.h"
```

### Enumerations

- enum [CurrPage](#) { [PINPAD](#) , [STATUS](#) }  
*Enum defined to keep track of current page on UI.*

### Functions

- void [SystemClock\\_Config](#) (void)  
*System Clock Configuration.*
- int [main](#) (void)  
*The application entry point.*
- void [HAL\\_UART\\_RxCpltCallback](#) (UART\_HandleTypeDef \*huart)
- void [Error\\_Handler](#) (void)  
*This function is executed in case of error occurrence.*

### Variables

- const uint8\_t [BT\\_TX\\_MUTE](#) = 0
- const uint8\_t [BT\\_TX\\_WATER\\_DETECTED](#) = 25
- const uint8\_t [BT\\_TX\\_WATER\\_NOT\\_DETECTED](#) = 26
- const uint8\_t [BT\\_TX\\_FIRE\\_DETECTED](#) = 50
- const uint8\_t [BT\\_TX\\_FIRE\\_NOT\\_DETECTED](#) = 51
- const uint8\_t [BT\\_TX\\_LOCKED](#) = 75
- const uint8\_t [BT\\_TX\\_UNLOCKED](#) = 76
- const uint8\_t [BT\\_TX\\_BELL\\_PRESSED](#) = 100
- const uint8\_t [BT\\_TX\\_BELL\\_NOT\\_PRESSED](#) = 101
- const uint8\_t [BT\\_TX\\_DISCARD](#) = 125

- const uint8\_t **BT\_RX\_MUTE** = 50
- const uint8\_t **BT\_RX\_DISCARD** = 100
- const int **SCREEN\_PIN\_LEN** = 4
- const int **MEMBRANE\_PIN\_LEN** = 4
- const int **DISCARDED\_WARNING\_WAIT\_PERIOD** = 5000
- uint8\_t **RX\_BUFFER**
- CRC\_HandleTypeDef **hcrc**
- DMA2D\_HandleTypeDef **hdma2d**
- UART\_HandleTypeDef **huart7**
- ADC\_HandleTypeDef **hadc**
- GPIO\_InitTypeDef **gpio\_buzzer**
- GPIO\_InitTypeDef **gpio\_touchSensor**
- TIM\_HandleTypeDef **htim3**
- TS\_StateTypeDef **ts**
- char **xTouchStr** [10]
- bool **waterDetected** = false
- bool **fireDetected** = false
- bool **doorLocked** = true
- bool **bellPressed** = false
- bool **muted** = false
- bool **btPreviousWaterTr** = false
- bool **btPreviousFireTr** = false
- bool **btPreviousLockTr** = true
- bool **btPreviousBellTr** = false
- int **lastBtTx** = 0
- int **lastBtTx\_spam** = 0
- bool **showWaterWarning** = false
- bool **showFireWarning** = false
- bool **showDoorLockWarning** = false
- bool **showDoorBellWarning** = false
- int **doorLock\_discardedWarningWait** = 0
- int **doorBell\_discardedWarningWait** = 0
- int **water\_discardedWarningWait** = 0
- int **fire\_discardedWarningWait** = 0

### 7.17.1 Detailed Description

: Main program body

Author

: Sahil Modak & Ali Nanji

Contains application entry point and main infinite loop

## 7.18 Core/Src/membrane.c File Reference

: Initializes and handles membrane keypad

```
#include "stm32f7xx_hal.h"
#include "stm32f7xx_hal_gpio.h"
#include "commonMethods.h"
#include "membrane.h"
```

## Data Structures

- struct [pin](#)

## Functions

- void [resetMembranePins](#) (int start, int stop, GPIO\_PinState value)  
*Resets Membrane pins to give value.*
- void [initializeMembrane](#) (GPIO\_InitTypeDef \*gpio, GPIO\_TypeDef \*GPIOx, uint16\_t GPIO\_Pin, uint16\_t pos)  
*Initializes membrane keypad pins, to input or output.*
- int [getMembraneNum](#) (int col, int row)  
*calculates number pressed depending on membrane keypad input*
- int [readMembrane](#) ()  
*Checks which button on the membrane keypad is pressed.*
- void [initializeMembranePins](#) (void)  
*Initializes all pins that will be used by the membrane keypad The membrane keypad uses pin D8 - D15 on the board. It is configured as follows: Mode = Input/Output Pull = PULL DOWN Speed = SPEED HIGH.*
- bool [checkMemPin](#) (int \*currentMemDigitCount, int \*currentMemPin, int membranePinLen)  
*Checks if the PIN entered on the keypad is correct.*

## Variables

- const int **currMemPin** = 1234
- GPIO\_InitTypeDef **gpio8**
- GPIO\_InitTypeDef **gpio9**
- GPIO\_InitTypeDef **gpio10**
- GPIO\_InitTypeDef **gpio11**
- GPIO\_InitTypeDef **gpio12**
- GPIO\_InitTypeDef **gpio13**
- GPIO\_InitTypeDef **gpio14**
- GPIO\_InitTypeDef **gpio15**
- struct [pin](#) **pin**

### 7.18.1 Detailed Description

: Initializes and handles membrane keypad

Author

: Sahil Modak

This file contains all the methods required to initialize and handle the membrane keypad. The public functions' prototypes have been defined in [membrane.h](#)

## 7.19 Core/Src/pinpad.c File Reference

: Handles Pin Pad page of the UI

```
#include "pinpad.h"
#include <stdio.h>
#include "stm32f7xx_hal.h"
#include "stm32746g_discovery_lcd.h"
#include "stm32746g_discovery_ts.h"
#include "commonMethods.h"
```

### Macros

- #define **wait\_delay** HAL\_Delay

### Functions

- void **drawDigitBox** (int x, int y, int digit)  
*Draw a rectangle for a key on the pin pad.*
- int **checkCoords** (int x, int y)  
*Check the coordinate of the user's touch to see what key has been pressed.*
- void **drawNumPad** (void)  
*Draw the entire num pad.*
- void **dispDigitSpaces** (void)  
*Draw dashes to display current pin.*
- void **dispDigit** (int num, int currentDispDigitCount)  
*Display the digit the user touched on.*
- void **deleteDigit** (int currentDispDigitCount)  
*Delete one digit from current pin.*
- void **clearDigits** (void)  
*Delete all digits from current pin.*
- void **clearText** (void)  
*Clear any banner.*
- void **dispWrongPassword** (void)  
*Display "Wrong Password" banner.*
- void **dispCorrectPassword** (void)  
*Display "Correct Password" banner.*
- bool **checkPassword** (int currentDispDigits[], int \*currentDispDigitCount)  
*Check if the currently input pin is correct.*
- void **displayLockScreen** (void)  
*Draw the entire lock screen.*
- void **initializeDisplay** (void)  
*Initialize everything needed for LCD and touch.*
- bool **checkPin** (int \*currentDispDigitCount, int \*currentDispDigits)  
*Wrapper method called from main loop to check current pin.*

## Variables

- const int **dispDigitsLimit** = 4
- const int **numPadBoxWidth** = 157
- const int **numPadBoxHeight** = 38
- const int **correctPassword** [4] = { 0, 0, 0, 0}
- const int **numPadCoordinates** [10][3]
- TS\_StateTypeDef **tsc\_state**

### 7.19.1 Detailed Description

: Handles Pin Pad page of the UI

Author

: Ali Nanji

This file contains all the methods required to manage the Pin Pad page of the UI. The pin pad page displays a pin pad where if the user enters the correct pin, they are directed to the status page. The public functions' prototypes have been defined in [pinpad.h](#)

## 7.20 Core/Src/servoMotor.c File Reference

: Initializes and handles servo motor

```
#include "servoMotor.h"
#include "stm32f7xx_hal.h"
```

## Functions

- void [PB4\\_Init](#) (void)  
*Servo Motor Pin PB4 Initialization Function The Servo Motor uses pin D3 on the board. It is configured as follows: Mode = Alternate Function Push Pull Pull = No Pull Speed = SPEED HIGH Pin = PIN 4 Alternate = TIM3 Alternate Function.*
- void [initServoMotor](#) (TIM\_HandleTypeDef \*htim3)  
*Wrapper method for initializations of the servo motor.*
- void [lockDoor](#) (TIM\_HandleTypeDef \*htim3)  
*Moves servo motor arm to 90 Degree position to lock the door.*
- void [unlockDoor](#) (TIM\_HandleTypeDef \*htim3)  
*Moves servo motor arm to 180 Degree position to unlock the door.*

### 7.20.1 Detailed Description

: Initializes and handles servo motor

Author

: Sahil Modak

This file contains all the methods required to lock and unlock the front door using the servo motor. The public functions' prototypes have been defined in [servoMotor.h](#)

## 7.21 Core/Src/statusPage.c File Reference

: Handles Status page of the UI

```
#include "statusPage.h"
#include "commonMethods.h"
#include <stdio.h>
#include "stm32f7xx_hal.h"
#include "stm32746g_discovery_lcd.h"
#include "stm32746g_discovery_ts.h"
```

### Macros

- #define **wait\_delay** HAL\_Delay

### Functions

- enum [StatusPageButtons](#) [statusPage\\_checkButtonCoords](#) (int x, int y)  
*Check what button has been clicked on by the user.*
- void [displayStatusPage](#) (bool muted)  
*Draw status page and all status labels.*
- void [updateStatusPage](#) (bool waterLeak, bool fire, bool doorLocked, bool bellPressed, bool muted)  
*Update status page.*
- enum [StatusPageButtons](#) [checkStatusPageTouch](#) (void)  
*Check if any buttons have been pressed on the status page.*

### Variables

- const int **statusPage\_lineYCoord** [4] = {70, 105, 140, 175}
- const int **statusPage\_statusLabelXCoord** = 10
- const int **statusPage\_statusXCoord** = 305
- const int **statusPage\_buttonsXCoord** [3] = {20, 190, 310}
- const int **statusPage\_buttonsYCoord** = 220
- const int **statusPage\_buttonsHeight** = 40
- const int **statusPage\_buttonsWidth** = 150
- const int **statusPage\_lockButtonHeight** = 40
- const int **statusPage\_lockButtonWidth** = 105
- TS\_StateTypeDef **tsc\_state\_statusPage**

#### 7.21.1 Detailed Description

: Handles Status page of the UI

Author

: Ali Nanji

This file contains all the methods required to manage the status page of the UI. The status page displays all the warnings related to each of the peripherals. The public functions' prototypes have been defined in [statusPage.h](#)

# Index

- ADC, [9](#)
- ADC\_Private\_Constants, [9](#)
- ADC\_Private\_Functions, [10](#)
  - configReadADC, [10](#)
- ADC\_Private\_Variables, [10](#)
- ADC\_Public\_Functions, [11](#)
  - checkFireDetected, [11](#)
  - checkWaterDetected, [11](#)
  - initializeAdcSensors, [12](#)
- checkCoords
  - PIN\_PAD\_PAGE\_Private\_Functions, [25](#)
- checkFireDetected
  - ADC\_Public\_Functions, [11](#)
- checkMemPin
  - MEMBRANE\_KEY\_PAD\_Public\_Functions, [23](#)
- checkPassword
  - PIN\_PAD\_PAGE\_Private\_Functions, [26](#)
- checkPin
  - PIN\_PAD\_PAGE\_Public\_Functions, [30](#)
- checkStatusPageTouch
  - STATUS\_PAGE\_Public\_Functions, [36](#)
- checkWaterDetected
  - ADC\_Public\_Functions, [11](#)
- clearDigits
  - PIN\_PAD\_PAGE\_Private\_Functions, [26](#)
- clearText
  - PIN\_PAD\_PAGE\_Private\_Functions, [26](#)
- Common\_Methods, [12](#)
- Common\_Methods\_Public\_Functions, [12](#)
  - enableClocks, [13](#)
  - isInRange, [13](#)
  - wait, [13](#)
- configReadADC
  - ADC\_Private\_Functions, [10](#)
- CORE, [38](#)
- Core/Inc/adcSensors.h, [41](#), [42](#)
- Core/Inc/commonMethods.h, [42](#)
- Core/Inc/main.h, [43](#)
- Core/Inc/membrane.h, [44](#)
- Core/Inc/pinpad.h, [45](#)
- Core/Inc/servoMotor.h, [45](#), [46](#)
- Core/Inc/statusPage.h, [46](#), [47](#)
- Core/Src/adcSensors.c, [47](#)
- Core/Src/commonMethods.c, [48](#)
- Core/Src/main.c, [49](#)
- Core/Src/membrane.c, [50](#)
- Core/Src/pinpad.c, [52](#)
- Core/Src/servoMotor.c, [53](#)
- Core/Src/statusPage.c, [54](#)
- CurrPage
  - MAIN\_Enum, [15](#)
- deleteDigit
  - PIN\_PAD\_PAGE\_Private\_Functions, [27](#)
- DISCARD
  - STATUS\_PAGE\_Enum, [35](#)
- dispCorrectPassword
  - PIN\_PAD\_PAGE\_Private\_Functions, [27](#)
- dispDigit
  - PIN\_PAD\_PAGE\_Private\_Functions, [28](#)
- dispDigitSpaces
  - PIN\_PAD\_PAGE\_Private\_Functions, [28](#)
- DISPLAY, [38](#)
- displayLockScreen
  - PIN\_PAD\_PAGE\_Public\_Functions, [30](#)
- displayStatusPage
  - STATUS\_PAGE\_Public\_Functions, [37](#)
- dispWrongPassword
  - PIN\_PAD\_PAGE\_Private\_Functions, [28](#)
- DOOR\_LOCK
  - STATUS\_PAGE\_Enum, [35](#)
- drawDigitBox
  - PIN\_PAD\_PAGE\_Private\_Functions, [29](#)
- drawNumPad
  - PIN\_PAD\_PAGE\_Private\_Functions, [29](#)
- enableClocks
  - Common\_Methods\_Public\_Functions, [13](#)
- Error\_Handler
  - MAIN\_Private\_Functions, [17](#)
- getMembraneNum
  - MEMBRANE\_KEY\_PAD\_Private\_Functions, [21](#)
- gpio
  - pin, [39](#)
- GPIOx
  - pin, [39](#)
- initializeAdcSensors
  - ADC\_Public\_Functions, [12](#)
- initializeDisplay
  - PIN\_PAD\_PAGE\_Public\_Functions, [31](#)
- initializeMembrane
  - MEMBRANE\_KEY\_PAD\_Private\_Functions, [21](#)
- initializeMembranePins
  - MEMBRANE\_KEY\_PAD\_Public\_Functions, [23](#)
- initServoMotor
  - SERVO\_MOTOR\_Public\_Functions, [32](#)
- isInRange

- Common\_Methods\_Public\_Functions, 13
- LOCK
  - STATUS\_PAGE\_Enum, 35
- lockDoor
  - SERVO\_MOTOR\_Public\_Functions, 33
- MAIN, 14
- main
  - MAIN\_Private\_Functions, 17
- MAIN\_Enum, 14
  - CurrPage, 15
  - PINPAD, 15
  - STATUS, 15
- MAIN\_Private\_Constants, 15
- MAIN\_Private\_FunctionPrototypes, 16
- MAIN\_Private\_Functions, 16
  - Error\_Handler, 17
  - main, 17
  - SystemClock\_Config, 17
- MAIN\_Private\_Variables, 15
- MEMBRANE\_KEY\_PAD, 19
- MEMBRANE\_KEY\_PAD\_Private\_Constants, 19
- MEMBRANE\_KEY\_PAD\_Private\_Functions, 20
  - getMembraneNum, 21
  - initializeMembrane, 21
  - readMembrane, 22
  - resetMembranePins, 22
- MEMBRANE\_KEY\_PAD\_Private\_Variables, 20
- MEMBRANE\_KEY\_PAD\_Public\_Functions, 22
  - checkMemPin, 23
  - initializeMembranePins, 23
- MEMBRANE\_KEY\_PAD\_Structs, 20
- MUTE
  - STATUS\_PAGE\_Enum, 35
- NONE
  - STATUS\_PAGE\_Enum, 35
- numPadCoordinates
  - PIN\_PAD\_PAGE\_Private\_Constants, 24
- PB4\_Init
  - SERVO\_MOTOR\_Private\_Functions, 32
- PERIPHERALS, 38
- pin, 39
  - gpio, 39
  - GPIOx, 39
- PIN\_PAD\_PAGE, 24
- PIN\_PAD\_PAGE\_Private\_Constants, 24
  - numPadCoordinates, 24
- PIN\_PAD\_PAGE\_Private\_Functions, 25
  - checkCoords, 25
  - checkPassword, 26
  - clearDigits, 26
  - clearText, 26
  - deleteDigit, 27
  - dispCorrectPassword, 27
  - dispDigit, 28
  - dispDigitSpaces, 28
  - dispWrongPassword, 28
  - drawDigitBox, 29
  - drawNumPad, 29
- PIN\_PAD\_PAGE\_Private\_Variables, 25
- PIN\_PAD\_PAGE\_Public\_Functions, 29
  - checkPin, 30
  - displayLockScreen, 30
  - initializeDisplay, 31
- PINPAD
  - MAIN\_Enum, 15
- readMembrane
  - MEMBRANE\_KEY\_PAD\_Private\_Functions, 22
- resetMembranePins
  - MEMBRANE\_KEY\_PAD\_Private\_Functions, 22
- SERVO\_MOTOR, 31
- SERVO\_MOTOR\_Private\_Functions, 31
  - PB4\_Init, 32
- SERVO\_MOTOR\_Public\_Functions, 32
  - initServoMotor, 32
  - lockDoor, 33
  - unlockDoor, 33
- STATUS
  - MAIN\_Enum, 15
- STATUS\_PAGE, 34
- STATUS\_PAGE\_Enum, 35
  - DISCARD, 35
  - DOOR\_LOCK, 35
  - LOCK, 35
  - MUTE, 35
  - NONE, 35
  - StatusPageButtons, 35
- STATUS\_PAGE\_Private\_Constants, 34
- STATUS\_PAGE\_Private\_Functions, 35
  - statusPage\_checkButtonCoords, 36
- STATUS\_PAGE\_Private\_Variables, 34
- STATUS\_PAGE\_Public\_Functions, 36
  - checkStatusPageTouch, 36
  - displayStatusPage, 37
  - updateStatusPage, 37
- statusPage\_checkButtonCoords
  - STATUS\_PAGE\_Private\_Functions, 36
- StatusPageButtons
  - STATUS\_PAGE\_Enum, 35
- SystemClock\_Config
  - MAIN\_Private\_Functions, 17
- unlockDoor
  - SERVO\_MOTOR\_Public\_Functions, 33
- updateStatusPage
  - STATUS\_PAGE\_Public\_Functions, 37
- wait
  - Common\_Methods\_Public\_Functions, 13