# Control de acceso

1.0

Generado por Doxygen 1.11.0

# Capítulo 1

# Índice de archivos

## 1.1. Lista de archivos

Lista de todos los archivos con breves descripciones:

# Capítulo 2

# Documentación de archivos

## 2.1. Referencia del archivo main.cpp

```
#include "mbed.h"
#include <cstdint>
#include <string.h>
#include "arm_book_lib.h"
#include "mis_funciones.hpp"
```

### Funciones

- BusOut keypadRows (PC_6, PC_7, PC_8, PC_9)

  *Declaración de GPIOs asociados a las filas del keypad.*
- BusIn keypadCols (PC_0, PC_1, PC_2, PC_3)

  *Declaración de GPIOs asociados a las Columnas del keypad.*
- DigitalOut workingLed (PA_5)

  *Declaración de GPIO asociado a LED que indica funcionamiento del sistema.*
- DigitalOut accessGrantedLed (PA_6)

  *Declaración de GPIO asociado a LED que indica acceso concedido.*
- DigitalOut accessDeniedLed (PA_7)

  *Declaración de GPIO asociado a LED que indica acceso denegado.*
- UnbufferedSerial uartUSB (USBTX, USBRX)

  *Declaración de UART para mostrar los mensajes del control de acceso.*
- int main ()

### Variables

- bool sequenceDetectedState

  *Flag indicador de secuencia completa. Flag en true indica secuencia completa.*
- bool accessState

  *Flag indicador de estado de acceso. Flag en true indica acceso concedido.*
- bool timeoutState

  *Flag indicador de timeout. Flag en true indica timeout alcanzado.*
- bool startTimeout

- int colReading

  *Respaldo del estado de las columnas del keypad luego de detectar tecla presionada.*
- int rowReading

  *Respaldo del estado de las filas del keypad luego de detectar tecla presionada.*
- uint8_t currentIndex

  *Indice actual de la secuencia.*
- uint16_t accumulatedTime

  *Tiempo acumulado utilizado para comparar con tiempo de timeout.*
- char inputSequence [SEQUENCE_LEN+1]

  *Array utilizado para guardar la secuencia de caracteres ingresados.*
- char keypadAllowedEntries [ROW_LEN][COL_LEN]

  *Array que contiene los caracteres asociados a las teclas del keypad.*

### 2.1.1. Descripción detallada

**Versión**

1.0

**Fecha**

18/7/24

**Autor**

Ramos Leonardo

Definición en el archivo main.cpp.

### 2.1.2. Documentación de funciones

#### 2.1.2.1. accessDeniedLed()

```
DigitalOut accessDeniedLed (
            PA_7 )
```

Declaración de GPIO asociado a LED que indica acceso denegado.

#### 2.1.2.2. accessGrantedLed()

```
DigitalOut accessGrantedLed (
            PA_6 )
```

Declaración de GPIO asociado a LED que indica acceso concedido.

#### 2.1.2.3. keypadCols()

```
BusIn keypadCols (
            PC_0 ,
            PC_1 ,
            PC_2 ,
            PC_3 )
```

Declaración de GPIOs asociados a las Columnas del keypad.

#### 2.1.2.4. keypadRows()

```
BusOut keypadRows (
            PC_6 ,
            PC_7 ,
            PC_8 ,
            PC_9 )
```

Declaración de GPIOs asociados a las filas del keypad.

#### 2.1.2.5. main()

```
int main ()
```

Definición en la línea 35 del archivo main.cpp.

```
00035           {
00037   keypadInit();
00038   uartInit();
00039   workingLed = 0;
00040   while (true) {
00041     if (accumulatedTime < TIMEOUT) {
00042       keypadSweepUpdate();
00043       if (sequenceDetectedState == true) {
00044         if (areEqual()) {
00045           accessState = true;
00046         } else {
00047           accessState = false;
00048         }
00049         if (accessState == true) {
00050           displayAccessGrantedMsg();
00051           accessGrantedLed = ON;
00052           delay(3000);
00053           accessGrantedLed = OFF;
00054           accessState = false;
00055         } else {
00056           displayAccessDeniedMsg();
00057           accessDeniedLed = ON;
00058           delay(1000);
00059           accessDeniedLed = OFF;
00060         }
00061         clearSequence();
00062         sequenceDetectedState = false;
00063         timeoutState = false;
00064         accumulatedTime = 0;
00065       }
00066       delay(10);
00067       accumulatedTime += 10;
00068     }
00069     else{
00070         if(timeoutState == true){
00071             displayTimeoutMsg();
00072             clearSequence();
00073             timeoutState = false;
00074         }
00075         accumulatedTime = 0;
00076     }
00077     if((accumulatedTime % BLINKING_TIME) == 0){
00078         workingLed = !workingLed;
00079     }
00080   }
00082 }
```

#### 2.1.2.6. uartUSB()

```
UnbufferedSerial uartUSB (
            USBTX ,
            USBRX )
```

Declaración de UART para mostrar los mensajes del control de acceso.

#### 2.1.2.7. workingLed()

```
DigitalOut workingLed (
            PA_5 )
```

Declaración de GPIO asociado a LED que indica funcionamiento del sistema.

### 2.1.3. Documentación de variables

#### 2.1.3.1. accessState

```
bool accessState
```

Flag indicador de estado de acceso. Flag en true indica acceso concedido.

Definición en la línea 22 del archivo main.cpp.

#### 2.1.3.2. accumulatedTime

```
uint16_t accumulatedTime
```

Tiempo acumulado utilizado para comparar con tiempo de timeout.

Definición en la línea 28 del archivo main.cpp.

#### 2.1.3.3. colReading

```
int colReading
```

Respaldo del estado de las columnas del keypad luego de detectar tecla presionada.

Definición en la línea 25 del archivo main.cpp.

#### 2.1.3.4. currentIndex

```
uint8_t currentIndex
```

Indice actual de la secuencia.

Definición en la línea 27 del archivo main.cpp.

#### 2.1.3.5. inputSequence

```
char inputSequence[SEQUENCE_LEN+1]
```

Array utilizado para guardar la secuencia de caracteres ingresados.

Definición en la línea 29 del archivo main.cpp.

#### 2.1.3.6. keypadAllowedEntries

```
char keypadAllowedEntries[ROW_LEN][COL_LEN]
```

**Valor inicial:**

```
= {{'1', '2', '3', 'A'},
                                                {'4', '5', '6', 'B'},
                                                {'7', '8', '9', 'C'},
                                                {'*', '0', '#', 'D'}}
```

Array que contiene los caracteres asociados a las teclas del keypad.

Definición en la línea 30 del archivo main.cpp.

```
00030                                             {{'1', '2', '3', 'A'},
00031                                              {'4', '5', '6', 'B'},
00032                                              {'7', '8', '9', 'C'},
00033                                              {'*', '0', '#', 'D'}};
```

#### 2.1.3.7. rowReading

```
int rowReading
```

Respaldo del estado de las filas del keypad luego de detectar tecla presionada.

Definición en la línea 26 del archivo main.cpp.

#### 2.1.3.8. sequenceDetectedState

```
bool sequenceDetectedState
```

Flag indicador de secuencia completa. Flag en true indica secuencia completa.

Definición en la línea 21 del archivo main.cpp.

#### 2.1.3.9. startTimeout

```
bool startTimeout
```

Definición en la línea 24 del archivo main.cpp.

#### 2.1.3.10. timeoutState

```
bool timeoutState
```

Flag indicador de timeout. Flag en true indica timeout alcanzado.

Definición en la línea 23 del archivo main.cpp.

## 2.2. main.cpp

Ir a la documentación de este archivo.

```
00001
00007 #include "mbed.h"
00008 #include <cstdint>
00009 #include <string.h>
00010 #include "arm_book_lib.h"
00011 #include "mis_funciones.hpp"
00012
00013 // main() runs in its own thread in the OS
00014 BusOut keypadRows(PC_6, PC_7, PC_8, PC_9);
00015 BusIn keypadCols(PC_0, PC_1, PC_2, PC_3);
00016 DigitalOut workingLed(PA_5);
00017 DigitalOut accessGrantedLed(PA_6);
00018 DigitalOut accessDeniedLed(PA_7);
00019 UnbufferedSerial uartUSB(USBTX, USBRX);
00020
00021 bool sequenceDetectedState;
00022 bool accessState;
00023 bool timeoutState;
00024 bool startTimeout;
00025 int colReading;
00026 int rowReading;
00027 uint8_t currentIndex;
00028 uint16_t accumulatedTime;
00029 char inputSequence[SEQUENCE_LEN + 1];
00030 char keypadAllowedEntries[ROW_LEN][COL_LEN] = {{'1', '2', '3', 'A'},
00031                                                {'4', '5', '6', 'B'},
00032                                                {'7', '8', '9', 'C'},
00033                                                {'*', '0', '#', 'D'}};
00034
00035 int main() {
00037   keypadInit();
00038   uartInit();
00039   workingLed = 0;
00040   while (true) {
00041     if (accumulatedTime < TIMEOUT) {
00042       keypadSweepUpdate();
00043       if (sequenceDetectedState == true) {
00044         if (areEqual()) {
00045           accessState = true;
00046         } else {
00047           accessState = false;
00048         }
00049         if (accessState == true) {
00050           displayAccessGrantedMsg();
00051           accessGrantedLed = ON;
00052           delay(3000);
00053           accessGrantedLed = OFF;
00054           accessState = false;
00055         } else {
00056           displayAccessDeniedMsg();
00057           accessDeniedLed = ON;
00058           delay(1000);
00059           accessDeniedLed = OFF;
00060         }
00061         clearSequence();
00062         sequenceDetectedState = false;
00063         timeoutState = false;
00064         accumulatedTime = 0;
00065       }
00066       delay(10);
00067       accumulatedTime += 10;
00068     }
00069     else{
00070         if(timeoutState == true){
00071             displayTimeoutMsg();
00072             clearSequence();
00073             timeoutState = false;
00074         }
00075         accumulatedTime = 0;
00076     }
00077     if((accumulatedTime % BLINKING_TIME) == 0){
00078         workingLed = !workingLed;
00079     }
00080   }
00082 }
```

## 2.3. Referencia del archivo mis_funciones.cpp

```
#include "mis_funciones.hpp"
```

### Funciones

- void keypadInit ()

  *keypadInit Inicializa el keypad 4x4 para su uso. Principalmente configura los GPIO de columnas como input PULL DOWN y da valores iniciales a ciertos flags.*
- void uartInit ()

  *uartInit Configura velocidad de transmisión y formato de la trama de datos de UART*
- void keypadSweepUpdate ()

  *keypadSweepUpdate Barre el keypad 4x4 hasta encontrar una tecla presionada, guarda el caracter asociado a esa tecla e indica mediante flag sequenceDetectedState que se ha ingresado la secuencia completa*
- char getKeyPressed ()

  *getKeyPressed Realiza la conversion de posicion en keypad (fila y columna) a caracter asociado*
- bool areEqual ()

  *areEqual Compara igualdad elemento a elemento entre el array de la secuencia ingresada y el string de usuario guardado en memoria*
- void displayAccessGrantedMsg ()

  *displayAccessGrantedMsg Muestra mensajes asociados a un acceso concedido en terminal*
- void displayAccessDeniedMsg ()

  *displayAccessDeniedMsg Muestra mensajes asociados a un acceso denegado en terminal*
- void displayTimeoutMsg ()

  *displayTimeoutMsg Muestra mensaje de timeout en terminal*
- void clearSequence ()

  *clearSequence Llena de ceros el array que recibe la secuencia desde el keypad y reinicia su índice.*

### 2.3.1. Descripción detallada

**Versión**

1.0

**Fecha**

18/7/24

**Autor**

Ramos Leonardo

Definición en el archivo mis_funciones.cpp.

## 2.3.2. Documentación de funciones

### 2.3.2.1. areEqual()

```
bool areEqual ()
```

areEqual Compara igualdad elemento a elemento entre el array de la secuencia ingresada y el string de usuario guardado en memoria

**Devuelve**

true si son iguales, false si no lo son.

Definición en la línea 111 del archivo mis_funciones.cpp.

```
00111                {
00112     bool equalState = true;
00113     if(strcmp(inputSequence, ACCESS_SEQUENCE) != 0){
00114         equalState = false;
00115     }
00116     return equalState;
00117 }
```

### 2.3.2.2. clearSequence()

```
void clearSequence ()
```

clearSequence Llena de ceros el array que recibe la secuencia desde el keypad y reinicia su índice.

Definición en la línea 153 del archivo mis_funciones.cpp.

```
00153                    {
00154     for(uint8_t i = 0; i < SEQUENCE_LEN; i++){
00155         inputSequence[i] = '\0';
00156     }
00157     currentIndex = 0;
00158 }
```

### 2.3.2.3. displayAccessDeniedMsg()

```
void displayAccessDeniedMsg ()
```

displayAccessDeniedMsg Muestra mensajes asociados a un acceso denegado en terminal

Definición en la línea 134 del archivo mis_funciones.cpp.

```
00134                                {
00135   uartUSB.write("Acceso denegado\n", 16);
00136   uartUSB.write("Usuario desconocido\n", 20);
00137   uartUSB.write("Clave:", 6);
00138   uartUSB.write(inputSequence, SEQUENCE_LEN);
00139   uartUSB.write("\n", 1);
00140 }
```

### 2.3.2.4. displayAccessGrantedMsg()

```
void displayAccessGrantedMsg ()
```

displayAccessGrantedMsg Muestra mensajes asociados a un acceso concedido en terminal

Definición en la línea 122 del archivo mis_funciones.cpp.

```
00122                                {
00123   uartUSB.write("Acceso concedido\n", 17);
00124   uartUSB.write("Usuario:", 8);
00125   uartUSB.write(USERNAME, strlen(USERNAME));
00126   uartUSB.write("Clave:", 6);
00127   uartUSB.write(ACCESS_SEQUENCE, strlen(ACCESS_SEQUENCE));
00128   uartUSB.write("\n", 1);
00129 }
```

#### 2.3.2.5. displayTimeoutMsg()

```
void displayTimeoutMsg ()
```

displayTimeoutMsg Muestra mensaje de timeout en terminal

Definición en la línea 145 del archivo mis_funciones.cpp.

```
00145                           {
00146     uartUSB.write("\nTimeout!!!\n", 12);
00147 }
```

#### 2.3.2.6. getKeyPressed()

```
char getKeyPressed ()
```

getKeyPressed Realiza la conversion de posicion en keypad (fila y columna) a caracter asociado

**Devuelve**

Caracter asociado a la tecla presionada

Definición en la línea 87 del archivo mis_funciones.cpp.

```
00087                           {
00088   uint8_t currentRow;
00089   uint8_t currentCol;
00090   for (uint8_t i = 0; i < ROW_LEN; i++) {
00091     if ((rowReading & (1 << i)) != 0) {
00092       currentRow = i;
00093       i = ROW_LEN;
00094     }
00095   }
00096   for (uint8_t i = 0; i < COL_LEN; i++) {
00097     if ((colReading & (1 << i)) != 0) {
00098       currentCol = i;
00099       i = COL_LEN;
00100     }
00101   }
00102   uartUSB.write("*", 1);
00103   return keypadAllowedEntries[currentRow][currentCol];
00104 }
```

#### 2.3.2.7. keypadInit()

```
void keypadInit ()
```

keypadInit Inicializa el keypad 4x4 para su uso. Principalmente configura los GPIO de columnas como input PULL DOWN y da valores iniciales a ciertos flags.

Definición en la línea 33 del archivo mis_funciones.cpp.

```
00033                           {
00034   keypadCols.mode(PullDown);
00035   inputSequence[SEQUENCE_LEN] = '\0';
00036   sequenceDetectedState = false;
00037   currentIndex = 0;
00038   accumulatedTime = 0;
00039 }
```

#### 2.3.2.8. keypadSweepUpdate()

```
void keypadSweepUpdate ()
```

keypadSweepUpdate Barre el keypad 4x4 hasta encontrar una tecla presionada, guarda el caracter asociado a esa tecla e indica mediante flag sequenceDetectedState que se ha ingresado la secuencia completa

Definición en la línea 58 del archivo mis_funciones.cpp.

```
00058                           {
00059   for (uint8_t i = 0; i < ROW_LEN; i++) {
00060     keypadRows = (1 << i);
00061     if (keypadCols != 0) {
00062       delay(100);
00063       if (keypadCols != 0) {
00064         timeoutState = true;
00065         accumulatedTime = 0;
00066         rowReading = keypadRows.read();
00067         colReading = keypadCols.read();
00068         inputSequence[currentIndex] = getKeyPressed();
00069         currentIndex++;
00070         if (currentIndex >= SEQUENCE_LEN) {
00071           uartUSB.write("\n", 1);
00072           sequenceDetectedState = true;
00073         }
00074         i = ROW_LEN;
00075       }
00076     }
00077   }
00078   keypadRows = 0;
00079   return;
00080 }
```

#### 2.3.2.9. uartInit()

```
void uartInit ()
```

uartInit Configura velocidad de transmisión y formato de la trama de datos de UART

Definición en la línea 45 del archivo mis_funciones.cpp.

```
00045                 {
00046   uartUSB.baud(9600);
00047   uartUSB.format(/*bits*/ 8,
00048                  /*paridad*/ SerialBase::None,
00049                  /*bits de stop*/ 1);
00050 }
```

## 2.4. mis_funciones.cpp

Ir a la documentación de este archivo.

```
00001
00007 #include "mis_funciones.hpp"
00008
00010 extern BusOut keypadRows;
00011 extern BusIn keypadCols;
00012 extern DigitalOut workingLed;
00013 extern DigitalOut accessGrantedLed;
00014 extern DigitalOut accessDeniedLed;
00015 extern UnbufferedSerial uartUSB;
00016
00017 extern bool sequenceDetectedState;
00018 extern bool accessState;
00019 extern bool timeoutState;
00020 extern int colReading;
00021 extern int rowReading;
00022 extern uint8_t currentIndex;
00023 extern uint16_t accumulatedTime;
00024 extern char inputSequence[SEQUENCE_LEN + 1];
00025 extern char keypadAllowedEntries[ROW_LEN][COL_LEN];
00027
00033 void keypadInit() {
```

```
00034   keypadCols.mode(PullDown);
00035   inputSequence[SEQUENCE_LEN] = '\0';
00036   sequenceDetectedState = false;
00037   currentIndex = 0;
00038   accumulatedTime = 0;
00039 }
00040
00045 void uartInit() {
00046   uartUSB.baud(9600);
00047   uartUSB.format(/*bits*/ 8,
00048                  /*paridad*/ SerialBase::None,
00049                  /*bits de stop*/ 1);
00050 }
00051
00058 void keypadSweepUpdate() {
00059   for (uint8_t i = 0; i < ROW_LEN; i++) {
00060     keypadRows = (1 << i);
00061     if (keypadCols != 0) {
00062       delay(100);
00063       if (keypadCols != 0) {
00064         timeoutState = true;
00065         accumulatedTime = 0;
00066         rowReading = keypadRows.read();
00067         colReading = keypadCols.read();
00068         inputSequence[currentIndex] = getKeyPressed();
00069         currentIndex++;
00070         if (currentIndex >= SEQUENCE_LEN) {
00071           uartUSB.write("\n", 1);
00072           sequenceDetectedState = true;
00073         }
00074         i = ROW_LEN;
00075       }
00076     }
00077   }
00078   keypadRows = 0;
00079   return;
00080 }
00081
00087 char getKeyPressed() {
00088   uint8_t currentRow;
00089   uint8_t currentCol;
00090   for (uint8_t i = 0; i < ROW_LEN; i++) {
00091     if ((rowReading & (1 << i)) != 0) {
00092       currentRow = i;
00093       i = ROW_LEN;
00094     }
00095   }
00096   for (uint8_t i = 0; i < COL_LEN; i++) {
00097     if ((colReading & (1 << i)) != 0) {
00098       currentCol = i;
00099       i = COL_LEN;
00100     }
00101   }
00102   uartUSB.write("*", 1);
00103   return keypadAllowedEntries[currentRow][currentCol];
00104 }
00105
00111 bool areEqual(){
00112     bool equalState = true;
00113     if(strcmp(inputSequence, ACCESS_SEQUENCE) != 0){
00114         equalState = false;
00115     }
00116     return equalState;
00117 }
00118
00122 void displayAccessGrantedMsg() {
00123   uartUSB.write("Acceso concedido\n", 17);
00124   uartUSB.write("Usuario:", 8);
00125   uartUSB.write(USERNAME, strlen(USERNAME));
00126   uartUSB.write("Clave:", 6);
00127   uartUSB.write(ACCESS_SEQUENCE, strlen(ACCESS_SEQUENCE));
00128   uartUSB.write("\n", 1);
00129 }
00130
00134 void displayAccessDeniedMsg() {
00135   uartUSB.write("Acceso denegado\n", 16);
00136   uartUSB.write("Usuario desconocido\n", 20);
00137   uartUSB.write("Clave:", 6);
00138   uartUSB.write(inputSequence, SEQUENCE_LEN);
00139   uartUSB.write("\n", 1);
00140 }
00141
00145 void displayTimeoutMsg(){
00146     uartUSB.write("\nTimeout!!!\n", 12);
00147 }
00148
00153 void clearSequence(){
```

```
00154     for(uint8_t i = 0; i < SEQUENCE_LEN; i++){
00155         inputSequence[i] = '\0';
00156     }
00157     currentIndex = 0;
00158 }
```

## 2.5. Referencia del archivo mis_funciones.hpp

```
#include "mbed.h"
#include <cstdint>
#include <string.h>
#include "arm_book_lib.h"
```

### defines

- #define ROW_LEN 4

  *Cantidad de filas del keypad.*
- #define COL_LEN 4

  *Cantidad de columnas del keypad.*
- #define SEQUENCE_LEN 4

  *Largo de la secuencia ingresada.*
- #define ACCESS_SEQUENCE "ABCD"

  *Secuencia de acceso guardada en memoria.*
- #define USERNAME "Leonardo\n"

  *Usuario asociado a la secuencia de acceso en memoria.*
- #define TIMEOUT 5000

  *Tiempo de timeout en ms.*
- #define BLINKING_TIME 500

  *Tiempo de parpadeo (ms) del led de funcionamiento del sistema.*

### Funciones

- void keypadInit ()

  *keypadInit Inicializa el keypad 4x4 para su uso. Principalmente configura los GPIO de columnas como input PULL DOWN y da valores iniciales a ciertos flags.*
- void uartInit ()

  *uartInit Configura velocidad de transmisión y formato de la trama de datos de UART*
- void keypadSweepUpdate ()

  *keypadSweepUpdate Barre el keypad 4x4 hasta encontrar una tecla presionada, guarda el caracter asociado a esa tecla e indica mediante flag sequenceDetectedState que se ha ingresado la secuencia completa*
- char getKeyPressed ()

  *getKeyPressed Realiza la conversion de posicion en keypad (fila y columna) a caracter asociado*
- void displayAccessGrantedMsg ()

  *displayAccessGrantedMsg Muestra mensajes asociados a un acceso concedido en terminal*
- void displayAccessDeniedMsg ()

  *displayAccessDeniedMsg Muestra mensajes asociados a un acceso denegado en terminal*
- void displayTimeoutMsg ()

  *displayTimeoutMsg Muestra mensaje de timeout en terminal*
- bool areEqual ()

  *areEqual Compara igualdad elemento a elemento entre el array de la secuencia ingresada y el string de usuario guardado en memoria*
- void clearSequence ()

  *clearSequence Llena de ceros el array que recibe la secuencia desde el keypad y reinicia su índice.*

### 2.5.1. Descripción detallada

**Versión**

1.0

**Fecha**

18/7/24

**Autor**

Ramos Leonardo

Definición en el archivo mis_funciones.hpp.

### 2.5.2. Documentación de «define»

#### 2.5.2.1. ACCESS_SEQUENCE

```
#define ACCESS_SEQUENCE "ABCD"
```

Secuencia de acceso guardada en memoria.

Definición en la línea 18 del archivo mis_funciones.hpp.

#### 2.5.2.2. BLINKING_TIME

```
#define BLINKING_TIME 500
```

Tiempo de parpadeo (ms) del led de funcionamiento del sistema.

Definición en la línea 21 del archivo mis_funciones.hpp.

#### 2.5.2.3. COL_LEN

```
#define COL_LEN 4
```

Cantidad de columnas del keypad.

Definición en la línea 16 del archivo mis_funciones.hpp.

#### 2.5.2.4. ROW_LEN

```
#define ROW_LEN 4
```

Cantidad de filas del keypad.

Definición en la línea 15 del archivo mis_funciones.hpp.

#### 2.5.2.5. SEQUENCE_LEN

```
#define SEQUENCE_LEN 4
```

Largo de la secuencia ingresada.

Definición en la línea 17 del archivo mis_funciones.hpp.

#### 2.5.2.6. TIMEOUT

```
#define TIMEOUT 5000
```

Tiempo de timeout en ms.

Definición en la línea 20 del archivo mis_funciones.hpp.

#### 2.5.2.7. USERNAME

```
#define USERNAME "Leonardo\n"
```

Usuario asociado a la secuencia de acceso en memoria.

Definición en la línea 19 del archivo mis_funciones.hpp.

### 2.5.3. Documentación de funciones

#### 2.5.3.1. areEqual()

```
bool areEqual ()
```

areEqual Compara igualdad elemento a elemento entre el array de la secuencia ingresada y el string de usuario guardado en memoria

**Devuelve**

true si son iguales, false si no lo son.

Definición en la línea 111 del archivo mis_funciones.cpp.

```
00111                {
00112     bool equalState = true;
00113     if(strcmp(inputSequence, ACCESS_SEQUENCE) != 0){
00114         equalState = false;
00115     }
00116     return equalState;
00117 }
```

#### 2.5.3.2. clearSequence()

```
void clearSequence ()
```

clearSequence Llena de ceros el array que recibe la secuencia desde el keypad y reinicia su índice.

Definición en la línea 153 del archivo mis_funciones.cpp.

```
00153                   {
00154     for(uint8_t i = 0; i < SEQUENCE_LEN; i++){
00155         inputSequence[i] = '\0';
00156     }
00157     currentIndex = 0;
00158 }
```

#### 2.5.3.3. displayAccessDeniedMsg()

```
void displayAccessDeniedMsg ()
```

displayAccessDeniedMsg Muestra mensajes asociados a un acceso denegado en terminal

Definición en la línea 134 del archivo mis_funciones.cpp.

```
00134                                {
00135   uartUSB.write("Acceso denegado\n", 16);
00136   uartUSB.write("Usuario desconocido\n", 20);
00137   uartUSB.write("Clave:", 6);
00138   uartUSB.write(inputSequence, SEQUENCE_LEN);
00139   uartUSB.write("\n", 1);
00140 }
```

#### 2.5.3.4. displayAccessGrantedMsg()

```
void displayAccessGrantedMsg ()
```

displayAccessGrantedMsg Muestra mensajes asociados a un acceso concedido en terminal

Definición en la línea 122 del archivo mis_funciones.cpp.

```
00122                                 {
00123   uartUSB.write("Acceso concedido\n", 17);
00124   uartUSB.write("Usuario:", 8);
00125   uartUSB.write(USERNAME, strlen(USERNAME));
00126   uartUSB.write("Clave:", 6);
00127   uartUSB.write(ACCESS_SEQUENCE, strlen(ACCESS_SEQUENCE));
00128   uartUSB.write("\n", 1);
00129 }
```

#### 2.5.3.5. displayTimeoutMsg()

```
void displayTimeoutMsg ()
```

displayTimeoutMsg Muestra mensaje de timeout en terminal

Definición en la línea 145 del archivo mis_funciones.cpp.

```
00145                           {
00146     uartUSB.write("\nTimeout!!!\n", 12);
00147 }
```

#### 2.5.3.6. getKeyPressed()

```
char getKeyPressed ()
```

getKeyPressed Realiza la conversion de posicion en keypad (fila y columna) a caracter asociado

**Devuelve**

Caracter asociado a la tecla presionada

Definición en la línea 87 del archivo mis_funciones.cpp.

```
00087                         {
00088   uint8_t currentRow;
00089   uint8_t currentCol;
00090   for (uint8_t i = 0; i < ROW_LEN; i++) {
00091     if ((rowReading & (1 << i)) != 0) {
00092       currentRow = i;
00093       i = ROW_LEN;
00094     }
00095   }
00096   for (uint8_t i = 0; i < COL_LEN; i++) {
00097     if ((colReading & (1 << i)) != 0) {
00098       currentCol = i;
00099       i = COL_LEN;
00100     }
00101   }
00102   uartUSB.write("*", 1);
00103   return keypadAllowedEntries[currentRow][currentCol];
00104 }
```

#### 2.5.3.7. keypadInit()

```
void keypadInit ()
```

keypadInit Inicializa el keypad 4x4 para su uso. Principalmente configura los GPIO de columnas como input PULL DOWN y da valores iniciales a ciertos flags.

Definición en la línea 33 del archivo mis_funciones.cpp.

```
00033                   {
00034   keypadCols.mode(PullDown);
00035   inputSequence[SEQUENCE_LEN] = '\0';
00036   sequenceDetectedState = false;
00037   currentIndex = 0;
00038   accumulatedTime = 0;
00039 }
```

#### 2.5.3.8. keypadSweepUpdate()

```
void keypadSweepUpdate ()
```

keypadSweepUpdate Barre el keypad 4x4 hasta encontrar una tecla presionada, guarda el caracter asociado a esa tecla e indica mediante flag sequenceDetectedState que se ha ingresado la secuencia completa

Definición en la línea 58 del archivo mis_funciones.cpp.

```
00058                           {
00059   for (uint8_t i = 0; i < ROW_LEN; i++) {
00060     keypadRows = (1 << i);
00061     if (keypadCols != 0) {
00062       delay(100);
00063       if (keypadCols != 0) {
00064         timeoutState = true;
00065         accumulatedTime = 0;
00066         rowReading = keypadRows.read();
00067         colReading = keypadCols.read();
00068         inputSequence[currentIndex] = getKeyPressed();
00069         currentIndex++;
00070         if (currentIndex >= SEQUENCE_LEN) {
00071           uartUSB.write("\n", 1);
00072           sequenceDetectedState = true;
00073         }
00074         i = ROW_LEN;
00075       }
00076     }
00077   }
00078   keypadRows = 0;
00079   return;
00080 }
```

#### 2.5.3.9. uartInit()

```
void uartInit ()
```

uartInit Configura velocidad de transmisión y formato de la trama de datos de UART

Definición en la línea 45 del archivo mis_funciones.cpp.

```
00045                 {
00046   uartUSB.baud(9600);
00047   uartUSB.format(/*bits*/ 8,
00048                  /*paridad*/ SerialBase::None,
00049                  /*bits de stop*/ 1);
00050 }
```

# 2.6. mis_funciones.hpp

Ir a la documentación de este archivo.

```
00001
00007 #ifndef MIS_FUNCIONES_H
00008 #define MIS_FUNCIONES_H
00009
00010 #include "mbed.h"
00011 #include <cstdint>
00012 #include <string.h>
00013 #include "arm_book_lib.h"
00014
00015 #define ROW_LEN 4
00016 #define COL_LEN 4
00017 #define SEQUENCE_LEN 4
00018 #define ACCESS_SEQUENCE "ABCD"
00019 #define USERNAME "Leonardo\n"
00020 #define TIMEOUT 5000
00021 #define BLINKING_TIME 500
00022
00023 /*Defino prototipos de funciones*/
00024 void keypadInit();
00025 void uartInit();
00026 void keypadSweepUpdate();
00027 char getKeyPressed();
00028 void displayAccessGrantedMsg();
00029 void displayAccessDeniedMsg();
00030 void displayTimeoutMsg();
00031 bool areEqual();
00032 void clearSequence();
00033
00034 #endif // MIS_FUNCIONES_H
```

# Índice alfabético