SMART HOME

Youssef Mohamed

IMT School

Table of Contents

Abstract

Different types of devices can be connected to smart-home networks. Smart light switches and dimmers can be used to control existing light fixtures, turning lights on or off or connecting them to a scheduler for timed activation. A smart-home owner might program their lights to coincide with the sunrise/sunset cycle, for example. The use of smart plugs also allows for the control of existing, ordinary devices. Smart kitchen appliances and entertainment units are becoming available as well.

Introduction

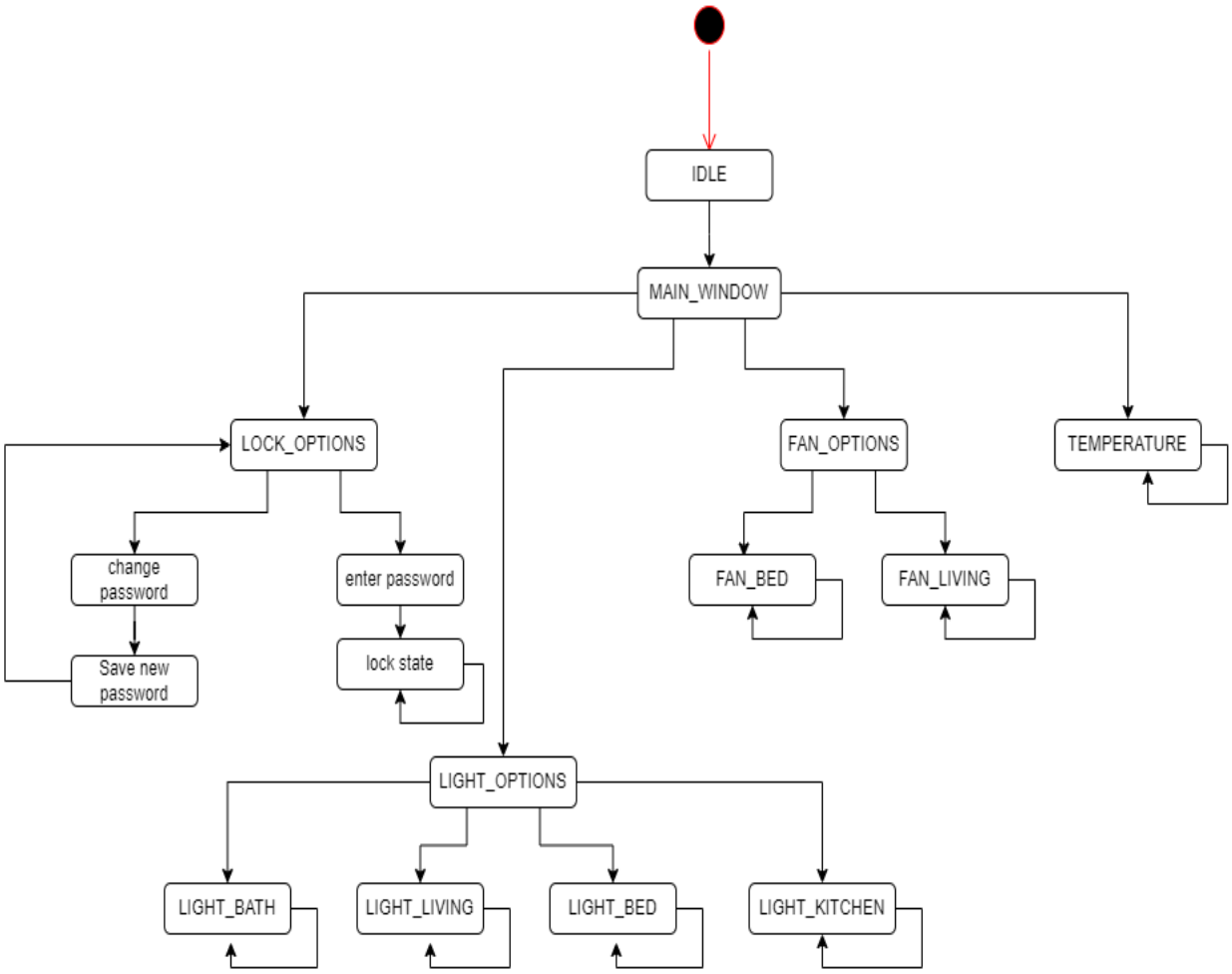
Home automation is a growing trend in the 2010s that allows owners to monitor their homes from afar. This project is about design a prototype for smart home submitted to IMT School to get certification for completing the Embedded Systems Diploma. We used ATmega32 as a microcontroller with LCD, Keypad, LEDs, Fans, IR Remote, Temperature sensor and EEPROM. The system has Door Lock, Fans control, Lights control, Temperature Warning. Tasks are handled using Free RTOS.

Activity Diagram:

1.Normal system activity:

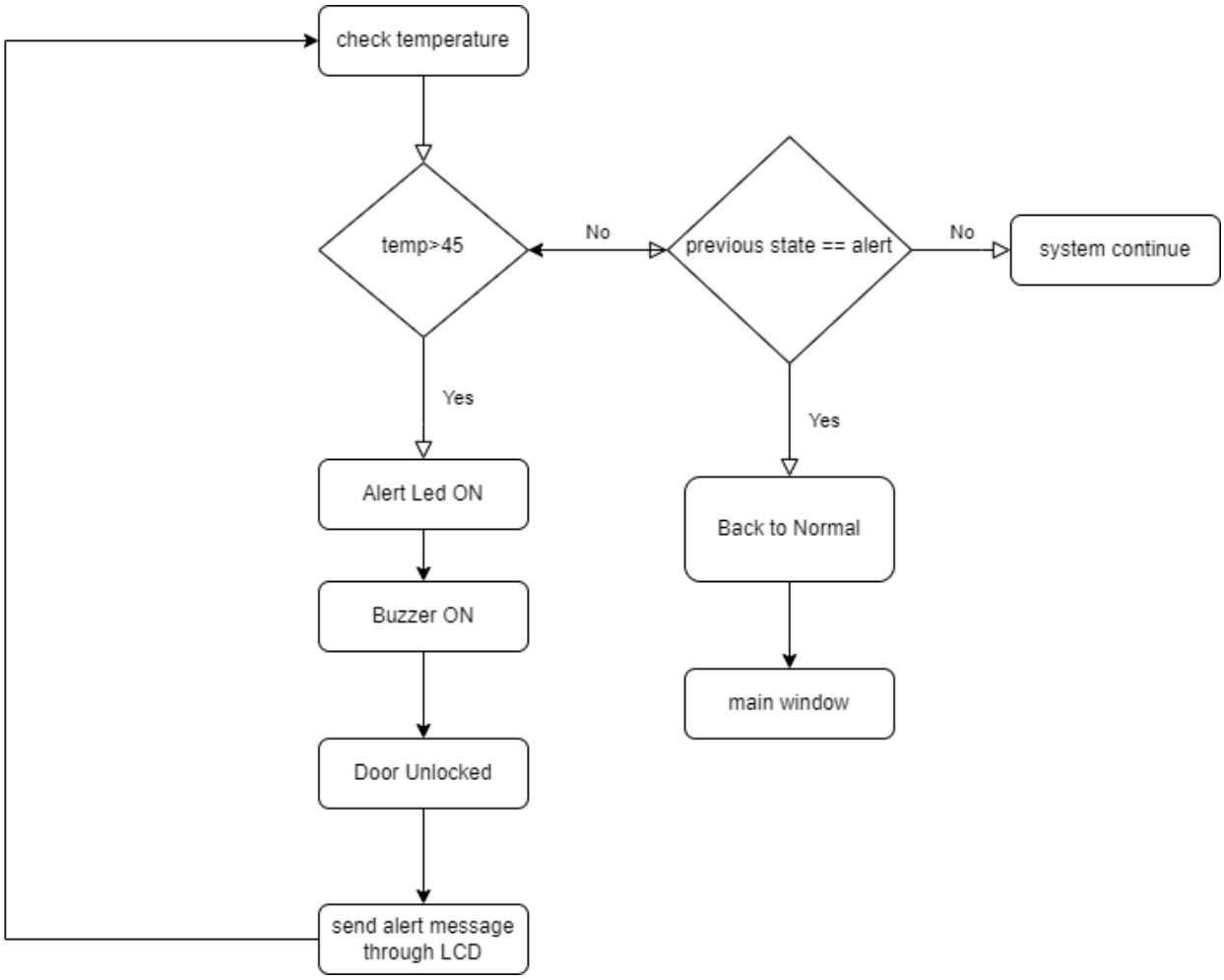
The system is divided into states each state has its own function. The user can switch between states using keypad. Critical case has its own handler and can interrupt the system in any case and when critical case is done the system well send a message and go to main window.

User can go to the previous window by clicking ‘B’ button, also can go directly to the main window by clicking ‘S’ button.



2.Alert check activity:

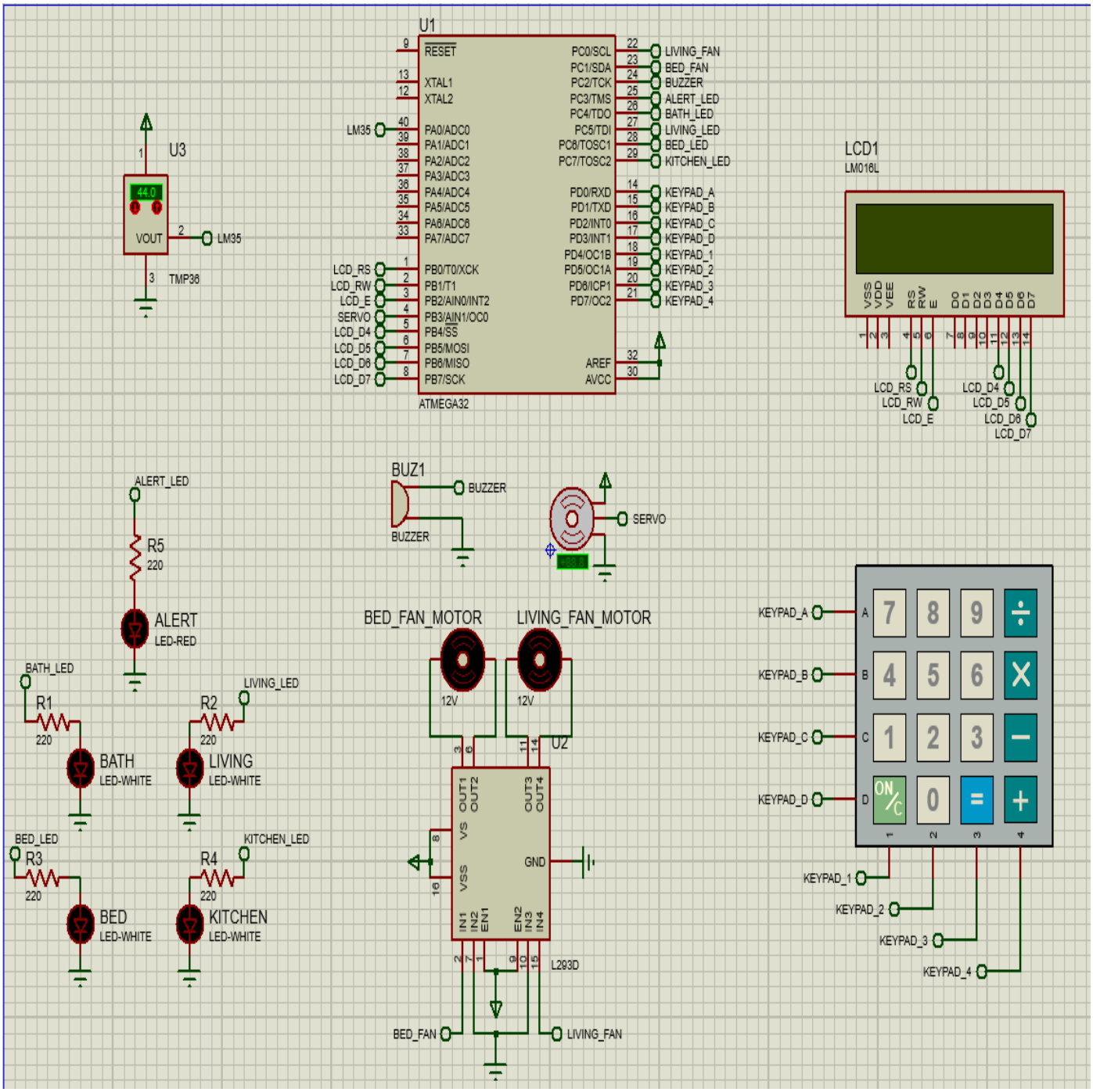
This check is done before managing keypad. If the temperature is more than 45 the critical temperature an alert led will turn on as well as a buzzer and the lcd will show alert message. The door will open to facilitate the exit. When the temperature goes back to normal the alert led and the buzzer will turn off and the door will go back to the state before alert. Then welcome back message will be sent on lcd and when the user presses any key the system will go to the main window.



Connections:

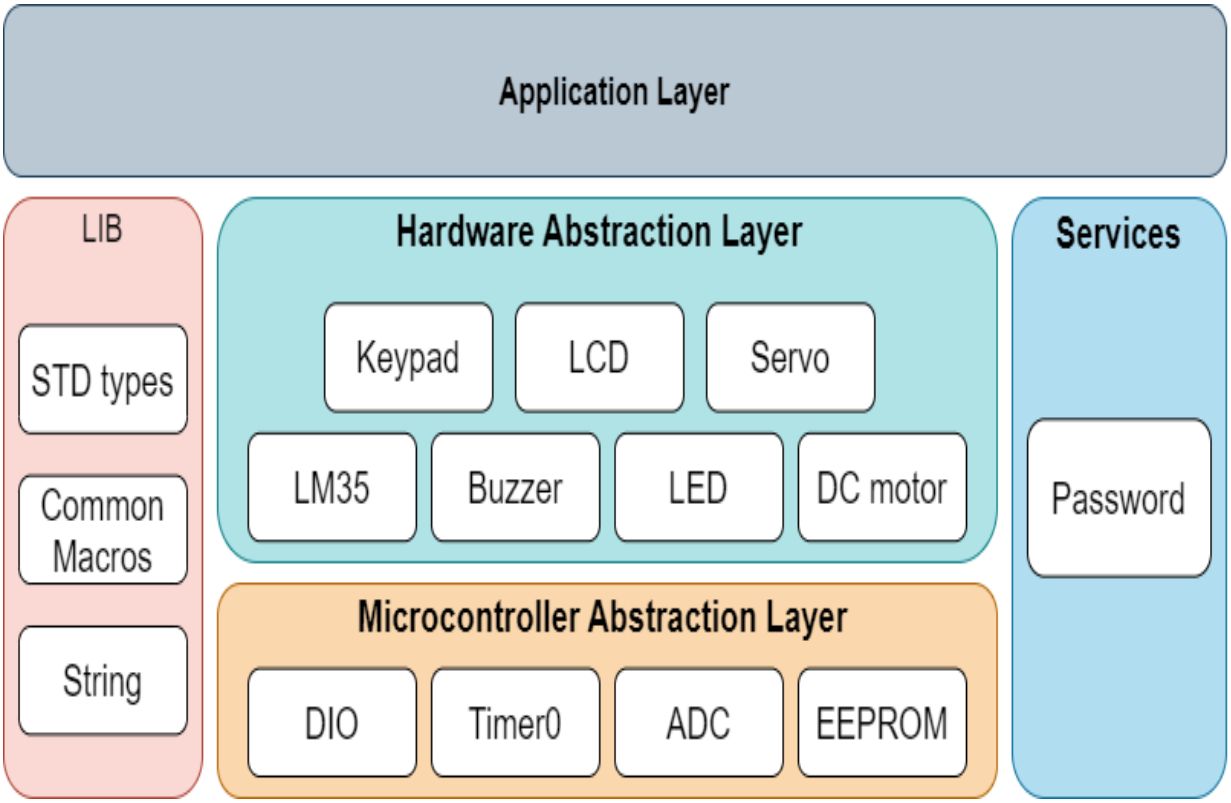
ATMEGA32	COMPONENT
A0	Temperature_Sensor
B0	LCD_RS
B1	LCD_RW
B2	LCD_E
B3	Servo
B4	LCD_D4
B5	LCD_D5
B6	LCD_D6
B7	LCD_D7
C0	Living_Fan
C1	Bed_Fan
C2	Buzzer
C3	Alert_LED
C4	Bath_LED
C5	Living_LED
C6	Kitchen_LED
C7	Bath _LED
D0	Keypad_R1
D1	Keypad_R2
D2	Keypad_R3
D3	Keypad_R4
D4	Keypad_C1
D5	Keypad_C2
D6	Keypad_C3
D7	Keypad_C4

System Diagram



Static design:

Layered architecture:



Preprocessor Configurations:

1.Microcontroller Configurations:

1.1. Timer0 Configurations:

Macros	Value
TIMER0_MODE	TIMER0_MODE_PWM
TIMER0_CLOCK_SOURCE	TIMER0_CLOCK_SOURCE_PRESCALER_64
TIMER0_CTC_TOP_VALUE	50
TIMER0_PRELOAD_VALUE	0
TIMER0_OC0_PIN_CONFIG	CLEAR_ON_COMPARE_MATCH
TIMER0_AUTO_RESTART	1

2.Hardware Configurations:

2.1. LCD Configurations:

Macros	Value
LCD_COMMAND_PORT	PORTB
LCD_EN	PIN2
LCD_RW	PIN1
LCD_RS	PIN0
LCD_DL	LCD_4_BIT_MODE
LCD_DATA_PORT	PORTB
LCD_DB4	PIN4
LCD_DB5	PIN5
LCD_DB6	PIN6
LCD_DB7	PIN7
LCD_NO_OF_LINES	LCD_2_LINE_MODE
LCD_FONT_SIZE	LCD_FONT_5x7

2.2. Keypad Configurations:

Macros	Value
KEYPAD_ROW_PORT	PORTD
START_ROW	PIN0
END_ROW	PIN3
KEYPAD_COL_PORT	PORTD
START_COL	PIN4
END_COL	PIN7
Global_u8KeypadKeys [4][4]	Row0 = {'7', '8', '9', 'S'}
	Row1 = {'4', '5', '6', 'C'}
	Row2 = {'1', '2', '3', 'B'}
	Row3 = {'*', '0', '#', '='}

2.3. Servo Configurations:

Macros	Value
SERVO_DUTYCYCLE_MIN	32
SERVO_DUTYCYCLE_MAX	158
SERVO_TIMER_USED	TIMER0

3.Services Configurations:

3.1. Password Configurations:

Macros	Value
PASSWORD_LENGTH	((uint_8)4)
PASSWORD_SAVE_DIRECTORY	INTERNAL_EEPROM
PASSWORD_START_ADDRESS	((uint_16)0x0000)
PASSWORD_INTIAL_VALUE	"0000"

APIs:

1.Microcontroller APIs:

1.1. DIO APIs:

Name	MDIO_voidSetPinDirection	void	
Arguments	Inputs	copy_u8PortId	uint_8
		Range: PORTA: PORTD	
		copy_u8PinId	uint_8
		Range: PIN_0: PIN_7	
		copy_u8PinDirection	uint_8
		Range: OUTPUT / INPUT	
Description	Set pin either output or input		

Name	MDIO_voidSetPinValue	void	
Arguments	Inputs	copy_u8PortId	uint_8
		Range: PORTA: PORTD	
		copy_u8PinId	uint_8
		Range: PIN_0: PIN_7	
		copy_u8PinValue	uint_8
		Range: HIGH / LOW / TOGGLE	
Description	Set pin either high or low or toggle its state		

Name	MDIO_u8GetPinValue	uint_8	
Arguments	Inputs	copy_u8PortId	uint_8
		Range: PORTA: PORTD	
		copy_u8PinId	uint_8
		Range: PIN_0: PIN_7	
Description	Return pin state either high or low		

1.2. GIE APIs:

Name	MGIE_voidEnable	void
Arguments	Inputs	**void**
Description	Enable global interrupt flag	

Name	MGIE_voidDisable	void
Arguments	Inputs	**void**
Description	Disable global interrupt flag	

1.3. ADC APIs

Name	MADC_voidInit	**void**
Arguments	Inputs	**void**
Description	Enable ADC	

Name	MADC_uint_16GetChannelValue	uint_16
Arguments	Inputs	copy_Channel_ID uint_8
		Range: see ADC channels macros
Range	0: 1023	
Description	Return the value of ADC channel	

1.4. Timer APIs:

Name	M_TIMER0_void_Init	void
Arguments	Inputs	void
Description	Initialize timer0 See timer configurations	

Name	M_TIMER0_void_SetDutyCycle	void	
Arguments	Inputs	copy_DutyCycle	uint_8
		Range: 0: 255	
Description	Change the duty cycle of the timer signal		

1.5. EEPROM APIs:

Name	M_EEPROM_voidWriteByte	void	
Arguments	Inputs	copy_address	uint_16
		Range: 0: 65535	
		copy_data	uint_8
		Range: 0: 255	
Description	Write a byte to the given address		

Name	M_EEPROM_u8ReadByte	uint_8	
Arguments	Inputs	copy_address	uint_16
		Range: 0: 65535	
Description	Return value of a byte from the given address		

2.Hardware APIs:

2.1. LED APIs:

Name	H_LED_void_init	**void**	
Arguments	Inputs	LED_Object	ST_LED
		See datatypes description	
Description	Initialize led object		

Name	H_LED_voidChangeState	**void**	
Arguments	Inputs	LED_Object	ST_LED
		See datatypes description	
		NewState	EN_LED
		Range: LED_OFF / LED_ON / LED_TOGGLE	
Description	Change led state		

Name	H_LED_EnGetState	**EN_LED**	
Arguments	Inputs	LED_Object	ST_LED
		See datatypes description	
Range	LED_OFF / LED_ON		
Description	Return led state		

2.2. LCD APIs:

Name	HLCD_voidInitialize	**void**
Arguments	Inputs	**void**
Description	Initialize LCD with specified configuration in LCD_config.h	

Name	HLCD_voidSendChar	**void**	
Arguments	Inputs	copy_s8Data	char
		Character to be printed	
Description	Print a character on LCD		

Name	HLCD_voidSendString	**void**	
Arguments	Inputs	copy_s8Data	char *
		String to be printed In the format: “String”	
Description	Print a string on LCD		

Name	HLCD_voidSendNumber	**void**	
Arguments	Inputs	copy_number	float_32
		Range: any number in the range of 16 characters	
Description	Print a number on LCD either float or integer		

Name	HLCD_voidCreateChar	**void**	
Arguments	Inputs	copy_CharLocation	uint_8
		Address where the character is saved Range: 0:16	
		copy_RowsOfChar	uint_8[]
		Range: 8 elements: Each element range: 0:255	
Description	Create a new character and specify an address for it To print this character use HLCD_voidSendChar and send the character address to it		

Name	HLCD_voidClearScreen	**void**
Arguments	Inputs	**void**
Description	Clear LCD screen and return cursor to initial position (0,0)	

Name	HLCD_voidSetCursor	**void**	
Arguments	Inputs	copy_u8Raw	uint_8
		Range: 0 / 1	
		copy_u8Col	uint_8
		Range: 0:15	
Description	Set cursor position The lcd has 2 rows and 16 columns Origin (0,0) at top left position Can be changed from configuration		

2.3. Keypad APIs:

Name	HKEYPAD_voidInitialize	**void**
Arguments	Inputs	**void**
Description	Initialize Keypad with specified configuration in KEYPAD_config.h	

Name	HKEYPAD_voidManage	**void**	
Arguments	Inputs	local_CallOutHandler	void (*) (void)
		Assign callback function address	
Description	Called in while loop to handle keypad and call callback function when key is pressed		

Name	HKEYPAD_u8GetPressedKey	**sint_8**
Arguments	Inputs	void
Description	Return pressed key value located in Global_u8KeypadKeys array in KEYPAD_config.h	

2.4. LM35 APIs

Name	HLM35_Init	**void**
Arguments	Inputs	**void**
Description	Initialize ADC	

Name	**HLM35_Reading**	**float_32**	
Arguments	Inputs	copy_Channel_ID	uint_8
		Range: 0:7	
Range	-40:125		
Description	Return the value of ADC channel value multiplied by a factor to be converted to temperature value		

2.5. Servo APIs:

Name	H_Servo_init	**void**
Arguments	Inputs	**void**
Description	Initialize Servo from SERVO_config.h	

Name	M_EEPROM_u8ReadByte	**void**	
Arguments	Inputs	copy_angle	uint_8
		Range: 0:180	
Description	Change duty cycle of the PWM to change servo angle		

2.6. DC motor APIs:

Name	H_DCmotorInit	**void**	
Arguments	Inputs	dcmotor_object	ST_DCmotor
		See datatypes description	
Description	Initialize DC motor object		

Name	H_DCmotor_voidChangeState	**void**	
Arguments	Inputs	dcmotor_object	ST_DCmotor
		See datatypes description	
		NewState	EN_DCMotorstate
		Range: MOTOR_STOP / MOTOR_MOVE	
Description	Change led state		

Name	H_DCmotor_EnGetState	**EN_DCMotorstate**	
Arguments	Inputs	dcmotor_object	ST_DCmotor
		See datatypes description	
Range	MOTOR_STOP / MOTOR_MOVE		
Description	Return DC motor state		

2.7. Buzzer APIs:

Name	H_DCmotorInit	**void**	
Arguments	Inputs	Buzzer_object	EN_Buzzerstate
		See datatypes description	
Description	Initialize buzzer object		

Name	H_DCmotor_voidChangeState	**void**	
Arguments	Inputs	Buzzer_object	EN_Buzzerstate
		See datatypes description	
		NewState	EN_Buzzerstate
		Range: BUZZER_OFF / BUZZER_ON	
Description	Change buzzer state		

Name	H_DCmotor_EnGetState	**EN_Buzzerstate**	
Arguments	Inputs	Buzzer_object	EN_Buzzerstate
		See datatypes description	
Range	BUZZER_OFF / BUZZER_ON		
Description	Return buzzer state		

3.Services APIs:

3.1. Password APIs:

Name	S_PASSWORD_voidInit	**void**
Arguments	Inputs	**void**
Description	Initialize memory where the password will be saved. with specified configuration in PASSWORD_config.h	

Name	S_PASSWORD_u8_Check	**void**	
Arguments	Inputs	copy_password	char []
		length is configured by PASSWORD_LENGTH in PASSWORD_config.h	
Description	Check if the password entered is matching the password in the memory		

Name	S_PASSWORD_u8_CheckMatching	**void**	
Arguments	Inputs	copy_password	char []
		length is configured by PASSWORD_LENGTH in PASSWORD_config.h	
		copy_ConfirmPassword	char []
		length is configured by PASSWORD_LENGTH in PASSWORD_config.h	
Description	Check if the 2 arrays match		

Name	S_PASSWORD_u8_Change		void
Arguments	Inputs	copy_NewPassword	char []
		length is configured by PASSWORD_LENGTH in PASSWORD_config.h	
Description	Change the password saved in the memory		

4.Application APIs:

Name	APP_hardware_Init	void
Arguments	Inputs	void
Description	Initialize all hardware and services components and set initial values for some components: • LEDs: turn off • DC motors: stop • Buzzer: turn off • Door lock(servo): lock • LCD: create characters (LOCKED, UNLOCKED, HEART). Send string “Welcome ” then character (HEART). Then send string “press any key” in second row.	

Name	APP_CheckTemperature	void
Arguments	Inputs	void
Description	Update temperature and go to ALERT_WINDOW if temperature exceeded critical value. If system was in ALERT_WINDOW and temperature came back to normal, go to BACK_TO_NORMAL. If the system was in TEMP_WINDOW, then update temperature and print it in second row and print “C “ After.	

Name	APP_SetSystemWindow	void
Arguments	Inputs	void
Description	Setup lcd window according to the current state	

Name	APP_CheckTemperature	void
Arguments	Inputs	void
Description	Check temperature from lm35 If the temperature exceeds critical value, the servo will open the door. alert led and buzzer will be turned on and system will go to ALERT_WINDOW. If the temperature returned to normal state, The system will go to BACK_TO_NORMAL window. If the system is temperature window and temperature is normal, this function will update the temperature value and print it on screen.	

Name	APP_keypad_Handler	void
Arguments	Inputs	void
Description	Handle what happens when a key is pressed	

Name	APP_Password_handler	void
Arguments	Inputs	void
Description	Handle pressed keys in Password windows	

Name	APP_Light_handler		void
Arguments	Inputs	copy_LED_ID	ST_LED
Description	Handle pressed keys in light windows		

Name	APP_Fan_handler		void
Arguments	Inputs	copy_fan_ID	ST_DCmotor
Description	Handle pressed keys in fan windows		

Name	main	void
Arguments	Inputs	void
Description	Call APP_hardware_Init before loop Call APP_CheckTemperature then HKEYPAD_voidManage in the loop. Send address of APP_keypad_Handler as callback function of HKEYPAD_voidManage	