National Telecomunication insitute

Automotive Embedded System track

Appraid Team

Group C

Project Documentaion

Under supervision:

Dr. Mahmoud Hussien

Eng. Aya Abdelwahab Anan

Prepared By:

No.	Name
1	Ahmed Mahmoud
2	Ahmd Hassan Ali
3	Alaa Ashraf Mahmoud
4	Mohammed Hassan Elnomos
5	Nour Alaa Eldin

Smart Home System Documentation

Table of Contents

Table of figure

System Overview

The system is divided into two Microcontrollers the home security system, and the home control system. Each system has separate tasks and integrates with each other to develop a full smart home system.

In this project we have developed a smart home software project using two ATMEGA 32 microcontroller with oscillator frequency of 8 Khz. Each peripheral and external hardware has its unique driver has been built by our team from zero to ensure the high quality and configurability of the system application. Both systems have been built using a high abstraction level as shown in figure 1.

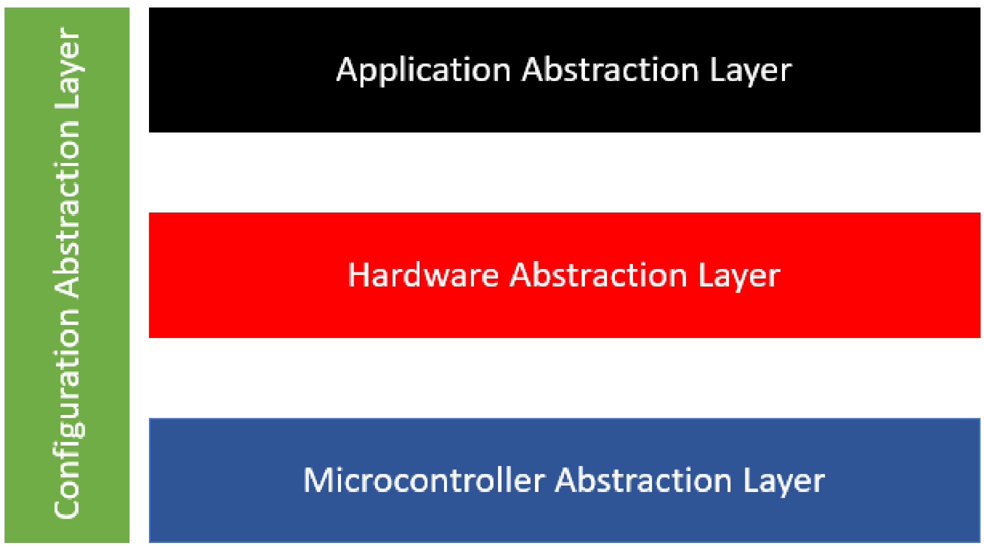


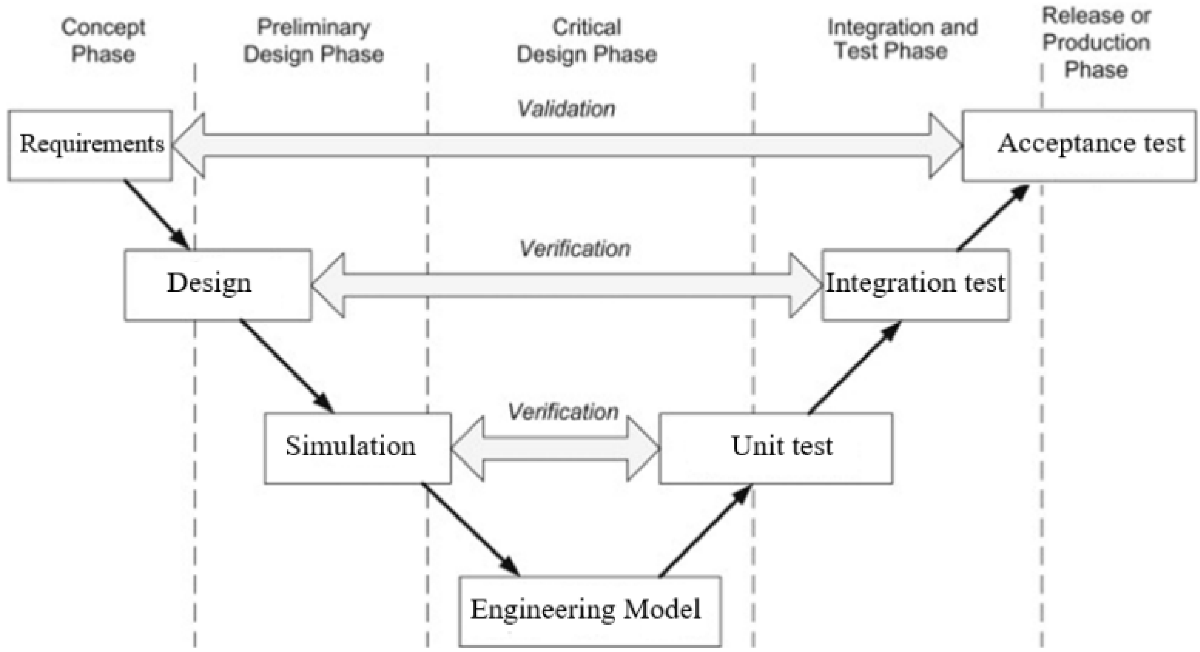
FIGURE 1: ABSTRACTION LAYERS

First, the home security system is responsible to secure the home and does not allow any outsider to enter the house without a valid identification. This system has been built using different types of drivers such as DIO, GIE, I2C, and USRTS in MCAL layer and Buzzer, Relay LCD, Keypad, EEPROM, RTC, and Fingerprint in the HAL layer.

Second, the home control system is responsible to allow user to control his/her own home system using mobile phone such as fan speed and light intensity and shows the temperature and the current light intensity in the LCD. In this system we have used DIO, Timer1, Timer0, USARTS, and ADC in the MCAL layer and Dc motor, temperature sensor, LDR sensor, WIFI module, LED, LCD in the MCAL layer.

➢ System Design Approach

To create a fully organized and clear project, V model will be implemented in this project. The V-Model demonstrates the relationship between each phase of the progress life cycle and its associated phase of testing. The V-Model improves project simplicity and project control by specifying standardized methods and defining the corresponding results and responsible roles. Figure 2 consists of five phases, that shows how verification and validation between the various activities.



In **requirements**, stage is a primary step to specify the requirements to create a full product, which is the Smart home system in our case. In this phase, we have decided all the systems prototypes and its functionality used in this project.

In the **Design** stage, will cover the preliminary system design of project details of algorithm, tool, and drivers selections.

The **simulation** stage will contain algorithm development, and proteus simulation tests.

The **Engineering model** includes the implementation stage which contains the system application and drivers coding.

The **Unit test** stage, contain the testing of each ECU individually.

The **integration testing** includes testing both ECUs together to prepare to the acceptance test by fixing the errors.

➢ System Block diagram

The following block diagram shows both systems are integrated with each other to develop full smart home project. ECU1 which is the home security system is responsible to take two types of user verification method from the fingerprint the keypad then shows the result in the LCD. EEPROM is used to allow the user to save its own password in ROM memory, while the current time is displayed using RTC. When the user is successfully authorized by the two verification steps then the door will open by the relay signal and an interrupt signal will be sent to the ECU2. In the other hand, if the user is not authorized then the buzzer will fire on and the system will off for 5 min.

In addition, ECU2 which is the home control system will control the house devices using tcp from the mobile. Both the mobile phone and the wifi module is connected at same wifi. Once the system is on the user then can choose from the menu list located in the application to control either led intensity or the fan speed and in same time the current values of the led intensity, fan speed, and the temperature. Moreover, the user chooses to control the device manually or to allow it runs automatically.

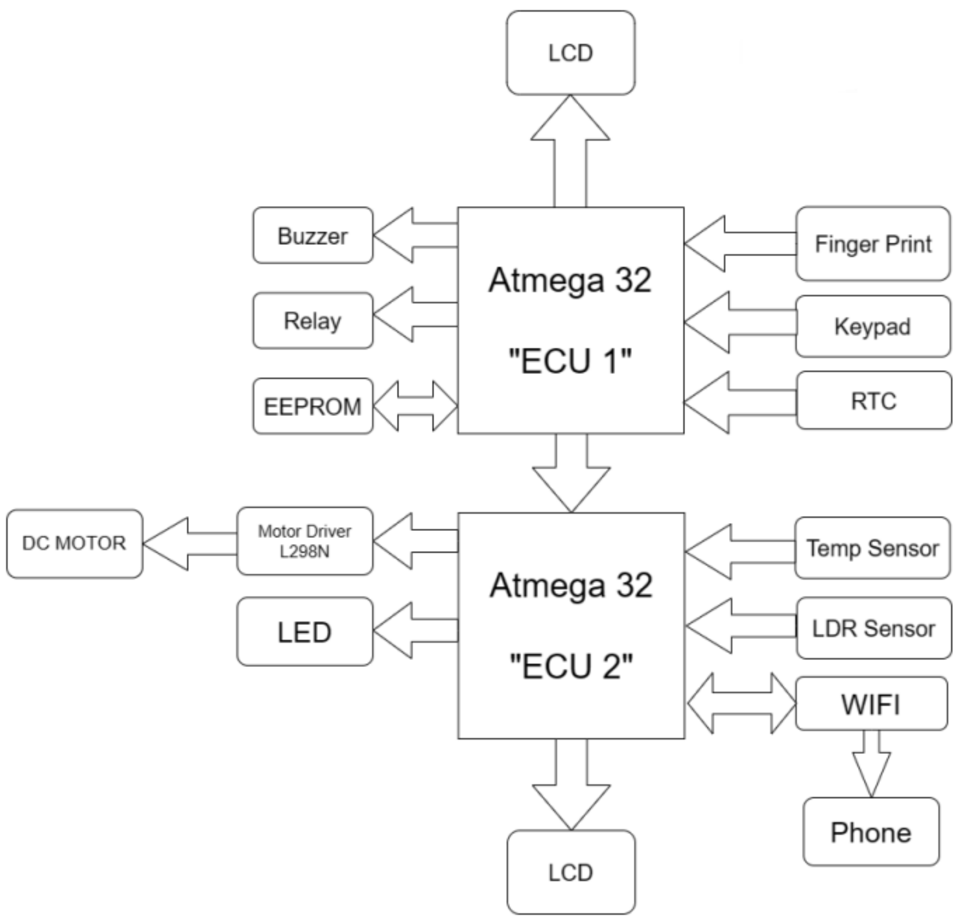


FIGURE 2: SYSTEM BLOCK DIAGRAM

➢ List of all drivers

The following table show all drivers used in the project software in both HAL and MCAL layers:

Hardware abstraction layer drivers:

Dc Motor				
Return Value	**Arguments**	**Sync/Async**	**Description**	**Syntax**
void	**void**	**Sync**	**function make dc motor working in forward**	**void H_Dc_Motor_Void_FORWARD ()**
void	**void**	**Sync**	**function make dc motor working in backward**	**void H_Dc_Motor_Void_BACKWARD(**
void	**void**	**Sync**	**function make dc motor stop working**	**void H_Dc_Motor_Void_STOP();**
void	**Duty**	**Sync**	**function make dc motor working with value of Duty**	**void H_Dc_Motor_Void_PWM();**
Buzzer				
Return Value	**Arguments**	**Sync/Async**	**Description**	**Syntax**
void	**void**	**Sync**	function make Buzzer working	void H_BUZZER_Void_TurnOn();
void	**void**	**Sync**	function make Buzzer stop	void H_BUZZER_Void_TurnOff();
KeyPad				
Return Value	**Arguments**	**Sync/Async**	**Description**	**Syntax**
u8 val of button	void	Sync	the function get the value of pressed key	u8 H_KEYPAD_u8_getPressedKey();

LCD				
Return Value	**Arguments**	**Sync/Async**	**Description**	**Syntax**
void	void	Sync	Init LCD in 4 Mode and 8 Mode	void H_LCD_void_Init();
void	u8 copy_u8data	Sync	write char on LCD	void H_LCD_void_sendData( );
void	u8 copy_u8comma	Sync	write commands in LCD	void H_LCD_void_sendCommand
void	const char * pstr	Sync	Write String in LCD	void H_LCD_void_sendString();
void	s32 copy_s32Num	Sync	convert char to num and Write int numbers	void H_LCD_void_sendIntNum();
void	u8 copy_u8Row,u8	Sync	put cursor in x, y position	void H_LCD_void_gotoXY();
void	const u8 * ArrPatte	Sync	create custom in LCD	void H_LCD_void_creatCustomCl
void	u8 copy_u8charCo	Sync	display custom in LCD	void H_LCD_void_displayCustom
void	void	Sync	Clear LCD	void H_LCD_void_ClearDisplay();
EEPROM				
Return Value	**Arguments**	**Sync/Async**	**Description**	**Syntax**
void	void	Sync	Init EEPROM	void H_EEPROM_Void_Init();
error state	u16 Address, u8 D	Sync	Read Byte from eeprom	ES_t H_EEPROM_Void_WriteByt
error state	u16 Copy_ByteAd	Sync	Write Byte to eeprom	ES_t H_EEPROM_Void_ReadBy

RTC				
Return Value	**Arguments**	**Sync/Async**	**Description**	**Syntax**
void	void	Sync	Initliaze function	void H_RTC_Void_Init();
void	RTC_CONFIG_TIME ,	Sync	Set Time	void H_RTC_Void_SetTime();
RTC_CONFIG_TIM	void	Sync	Get Time	RTC_CONFIG_TIME H_RTC_Void_GetT
void	RTC_CONFIG_DATE	Sync	Set Date	void H_RTC_Void_SetDate(RTC_CONF
RTC_CONFIG_DA	void	Sync	Get Date	RTC_CONFIG_DATE H_RTC_Void_Get
BCD_Nuber	u8 Copy_Number	Sync	Convert from DECEMIAL TO BCD	u8 H_RTC_DEC_TO_BCD(u8 Copy_Nu
Decimal Number	u8 Copy_Number	Sync	Convert from BCD TO DECEMIAL	u8 H_RTC_DEC_TO_BCD();

Relay				
Return Value	**Arguments**	**Sync/Async**	**Description**	**Syntax**
void	**Realy_Type**	**Sync**	**function turn on Relay 1 or Relay 2**	**void H_Relay_Void_WorkOn();**
void	**Realy_Type**	**Sync**	**function turn off Relay 1 or Relay 3**	**void H_Relay_Void_WorkOff();**

Microcontroller abstraction layer drivers:

DIO

Return Value	Arguments	Sync/Async	Description	Syntax
Dio_HIGH /LOW	Dio_ChannelType	SYNC	read the pin state	Dio_LevelType M_Dio_en_getPinValue();
void	Dio_ChannelType,Dio_L	SYNC	write the pin	void M_Dio_void_setPinValue();
Dio_PortLevelType	Dio_PortType	SYNC	Read the whole port	Dio_PortLevelType M_Dio_en_getPortValue(
void	Dio_PortType ,Dio_Port	SYNC	write the whole port	void M_Dio_void_setPortValue(l);
void	Dio_ChannelType	SYNC	toggle the pin state	Dio_LevelType M_Dio_void_togglePinValue(

PORT

Return Value	Arguments	Sync/Async	Description	Syntax
void	const Port_ConfigType	Sync	Init port values high / low or input / output	void Port_Init();

TWI

Return Value	Arguments	Sync/Async	Description	Syntax
void	TWI_SCL_FREQUAN	Sync	Initlize	void M_TWI_Void_InitMaster();
TWI_STATUS	None	Sync	Send Start Condition	void M_TWI_Void_InitMaster();
TWI_STATUS	u8 Copy_DataByte	Sync	Send Byte	void M_TWI_Void_InitMaster();
void	u8 Copy_SlaveAddres	Sync	Set Slave Address	void M_TWI_Void_InitMaster();
void	u8 *PTR_RecevedDat	Sync	Recevie With ACK	void M_TWI_Void_InitMaster();
void	u8 *PTR_RecevedDat	Sync	Recevie With NOT ACK	void M_TWI_Void_InitMaster();
void	u8* status	Sync	Get Status	void M_TWI_Void_InitMaster();

TIMER 0

Return Value	Arguments	Sync/Async	Description	Syntax
void	TimerMode, TimerWave	Sync	Init Timer 0	void M_Timer0_Void_Init();
void	TimerScaler	Sync	Start Timer by prescaler	void M_Timer0_Void_start();
void	TimerScaler	Sync	Stop Timer	void M_Timer0_Void_stop();
void	TimerMode	Sync	inturrept of timer 0 enable	void M_Timer0_Void_EnableInt();
void	TimerMode	Sync	inturrept of timer 0 disnable	void M_Timer0_Void_DisableInt();
Error state	void (*pf) (void)	Async	timer 0 callback function fo CTC	u8 M_Timer0_U8_CTCsetCallBack();
Error state	void (*pf) (void)	Async	timer 0 callback function fo OV	u8 M_Timer0_U8_OVsetCallBack ();
void	TimemSec,Timer0 Mode,Tir	Sync	function delay time by micro sec	void M_Timer0_Void_setDelayTimeMilliSec();
void	duty,Timer0Scaler, Timer0S	Sync	function calculate duty and save the value in OCR0	void M_Timer0_Void_setFastPWM();
void	duty,Timer0Scaler, Timer0S	Sync	function calculate duty and save the value in OCR1	void M_Timer0_Void_setphaseCorrectPWM();
u32	void	Sync	function calculate num of counts in timer0	u32 M_Timer0_U32_GetCounts();

ADC

Return Value	Arguments	Sync/Async	Description	Syntax
Error State	ADC_cfg_type	Sync	Init ADC	ES_t M_ADC_enu_init();
Error State	ADC_CHNL_TYPE	Sync	selet channel	ES_t M_ADC_enu_selectChannel();
Error State	ADC_cfg_type , float*	Sync	Get ADC Digital value Sync	ES_t M_ADC_enu_getDigitalValueSynch
Error State	float* voltage	ASync	Get ADC Digital value ASync	ES_t M_ADC_enu_getDigitalValueAsync
Error State	*Copy_pfunAppFun, vo	Async	Set call back function	ES_t M_ADC_enu_setCallBack();
Error State	void	Sync	Start Convertion	ES_t M_ADC_enu_startConversion();

ECU1: Security Control System

In the security control system, the welcome screen will be appeared once the system start. The user can choose either to sing in or sign up or to rest all passwords. Resting all password will overnight on the previous user's data.

The application algorithm is illustrated in the flow chart and state diagram section.

➢ ECU1 Flowchart

The following figure shows the flow chart of ECU1 algorithm. Once the user chose the signup the system will ask to enter any valid password to ensure that no outsider can sign up in the home without permission. The default Password of the system will be 1234 the user can over right it after signing up. After entering a valid password, the user will be asked to scan his/her finger print and entering a personal password, the data will be save and giving the user a unique ID for identification.

If the user chose to sign in the system will ask to enter fingerprint, then enter the password for this fingerprint's user. if either the fingerprint or the password is incorrect the system will give the user three trials then the system will close, and fire alarm will on for 5 min.

If the user is authorized the system will open the door and send signal to the ECU2 to enter the house.

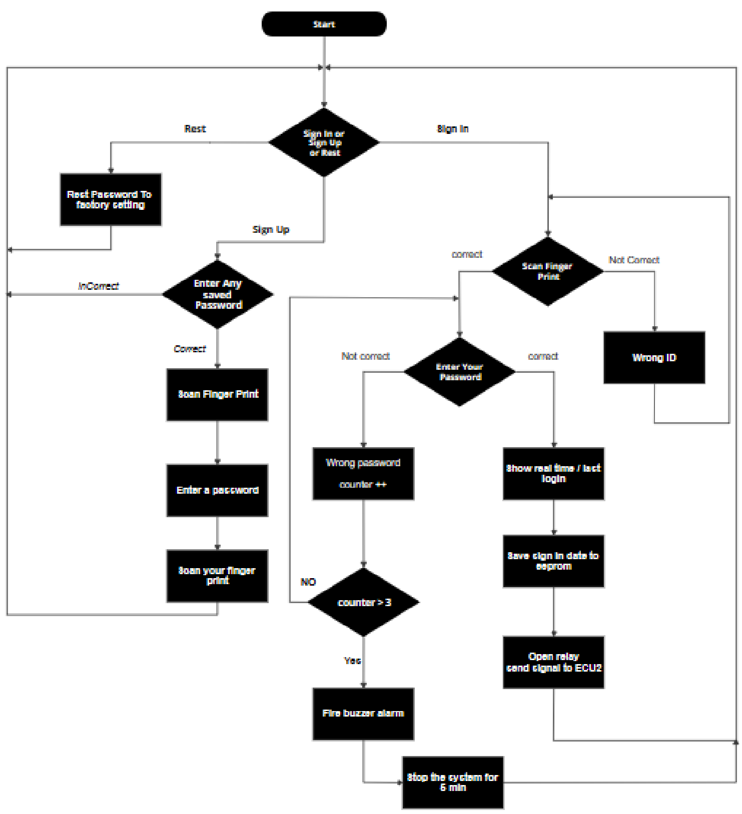


FIGURE 3: ECU1 FLOW CHART

➢ State diagram

The states in this ECU are seven states, when the system start it enter state one displays the options to user (sign in, sign up, reset).
When user choose one of the options the system exit from state 1 and enter state 2 or 3.
if the user choose sign up , the system enter state 2 and save the new pass, ID, finger print, when sign up complete the system exit from this state and return to start point.
if the user choose sign in , the system enter state3 and ask user to enter pass, ID, Finger Print then check and exit from this state . if the check is true the system enters in state five because it is successful login so it will display "welcome", save the log in time, display it, relay open for 5 sec then enter state six and send signal to ECU2.
but. if the check is false the system enter in state 4 and increase counter and check num of trying if it less than 3 it return to state 3 and ask user to log in, but if it greater than 3 the system enter in state 7 and display "system stopped", the buzzer work and system closed for 1 min then return to start point.

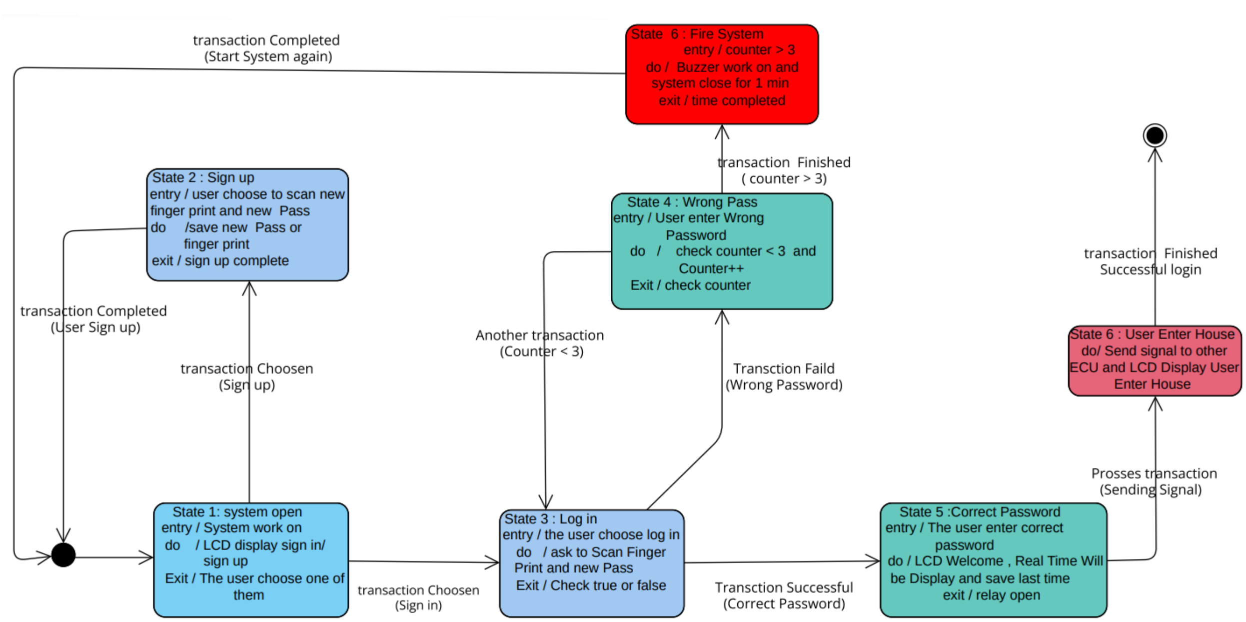


FIGURE 4: ECU1 STATE DIAGRAM

➢ List of modules

The following table shows all drivers modules used in the MCAL and HAL layer for ECU1 system:

List of all Modules	
HAL	
Module Name	**Function**
Keypad	Allow user to enter password
LCD	Display different operation for the user
RTC	Real Time Clock, used to get the current time
EEPROM	Electrically erasable programable ROM, used as small storge for password and last login
Buzzer	To fire alarm when wrong password
Relay	Open Door
Fingerprint	To identify the user identity
MCAL	
Module Name	**Function**
DIO	Used for deciding the Pin status
PORT	To configure the PINS
I2C	Communication RTC and EEPROM with MC1
USART	Communication Fingerprint and MC1
TIMER0	Delay when stop the system

➢ Software Components:

The following table shows all functions implemented in the application layer to develop the security system software.

Sign In User Authentication	
Syntax	LOGIN_STATE User_Authentication_SignIn();
Description	User will enter Fingerprint and password
Sync/Async	Sync
Reentrancy	Reentrant
Arguments	NONE
Return Value	Login State

Sign Up User Authentication	
Syntax	void User_Authentication_SignUp();
Description	User will enter new finger Print and password save password to new location
Sync/Async	Sync
Reentrancy	Reentrant
Arguments	NONE
Return Value	VOID

Successful Authentication	
Syntax	void Successful_Authentication(void);
Description	Welcome screen, Display Real time, save last login, Relay open, and Send signal to ECU2
Sync/Async	Sync
Reentrancy	Reentrant
Arguments	NONE
Return Value	void

Time processing	
Syntax	void Time_Processing();
Description	Show Real time and save it to the EEPROM
Sync/Async	Sync
Reentrancy	Reentrant
Arguments	NONE
Return Value	void

Wrong Authentication	
Syntax	Pass_State Wrong_Authentication();
Description	Increase Global counter, re enter password, Call Sign in function
Sync/Async	Async
Reentrancy	Non-Reentrant
Arguments	NONE
Return Value	PASS_STATE

Check Password	
Syntax	Static PASS_STATE Check_Password(u8 *Copy_Password, u8 Copy_ID);
Description	Compare ID and check Password from EEPROM DATA if correct or not
Sync/Async	Sync
Reentrancy	Reentrant

Arguments	u8 *Copy_Password, u8 Copy_ID
Return Value	PASS_STATE

System Off	
Syntax	Void System_OFF();
Description	Fire Buzzer, Delay 1 min, LCD print System off
Sync/Async	Sync
Reentrancy	Reentrant
Arguments	NONE
Return Value	VOID

Save New Password	
Syntax	Void Save_NewPassword(u8 *Copy_NewPassword, u8 Copy_ID);
Description	Save password to new location in the EEPROM
Sync/Async	Sync
Reentrancy	Reentrant
Arguments	NONE
Return Value	VOID

Global Variables		
U8	Password counter	Used for check number of unsuccessful login

➢ Memory Organization

EEPROM Memory Organization	
Address Number	Memory Description
0x00 – 0x03	Password characters for user 1
0x04 – 0x07	Password characters for user 2
0x20 – 0x23	Real Time (Hours, min, sec, and PM/AM) (LAST LOGIN)
0x24 – 0x26	5Real Time (Day) (LAST LOGIN)
0x255	Number of IDs

Microcontroller2: Home control system

In the Home control system, the ECU2 will wait until it receives an interrupt signal from ECU1 to start working. Once the system is on the WIFI module will connect to the Wi-Fi, the user will be able send command to the controller and the controller will send a verification message to the user to ensure all data is transferred correctly.
The application algorithm is illustrated in the flow chart and state diagram section.

➤ ECU2 Flow chart

The system starts if the login done successfully, and signal received from ECU1.
It starts by the system initialization prosses, then displays control option on user's smart phone and read WIFI commands.
LM35 and LDR start to sense temperature and Light intensity, then and based on the measured values and user's preferences the fan speed and LED intensity is adjusted through PWM signals generated from two different timer modules in the MCU.
User have two parameters to manipulate; first to control fan speed and user has 3 options:

- High speed.
- Low speed.
- Automatic (depending on LM35 measurements).

Second to control LED Intensity and user has 2 options:

- Turn off.
- Automatic (depending on LDR measurements).

Also, user can choose to make a logout, and in this case the system stops until new successful login signal comes from the ECU1

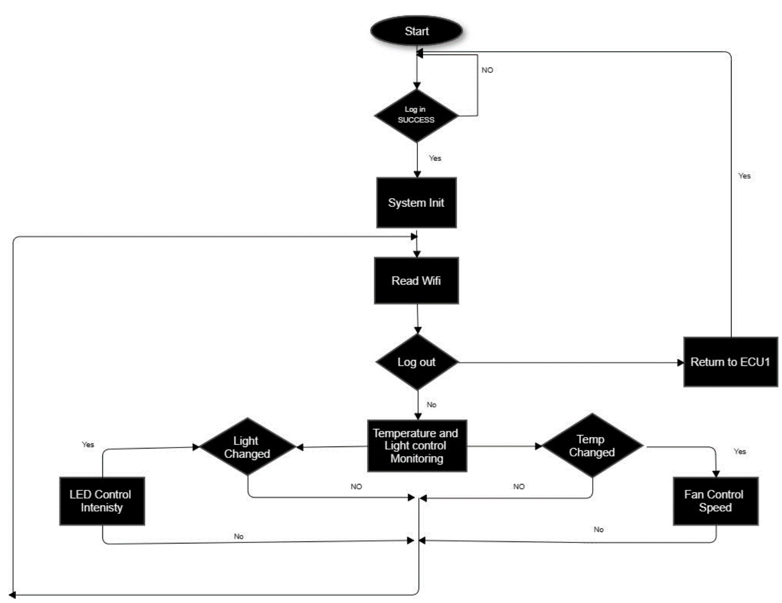


FIGURE 5: ECU2 BLOCK DIAGRAM

➢ State diagram

The states in this ECU are eight states:

when the signal come from ECU1 the ECU2 enter state one to in it the system and display the menu of options then Wi-Fi signal come so the system out of state one and enter state 2 to read the Wi-Fi commands get from the user the user have 3 choices (fan speed- led intensity – log out) if user choose log out the system enter in state 5 and stop the system but if not the system enter state 7 and gets the reading of Temp , LDR Sensors. If user choose fan speed the system enter state 4, adjusted the fan speed according to temp , the user also have another option to control the motor automatic by using state 8.

If user choose Led intensity the system enter state 3, adjusted the LED Intensity according to LDR Reading , the user also have another option to control the LED automatic by using state 6

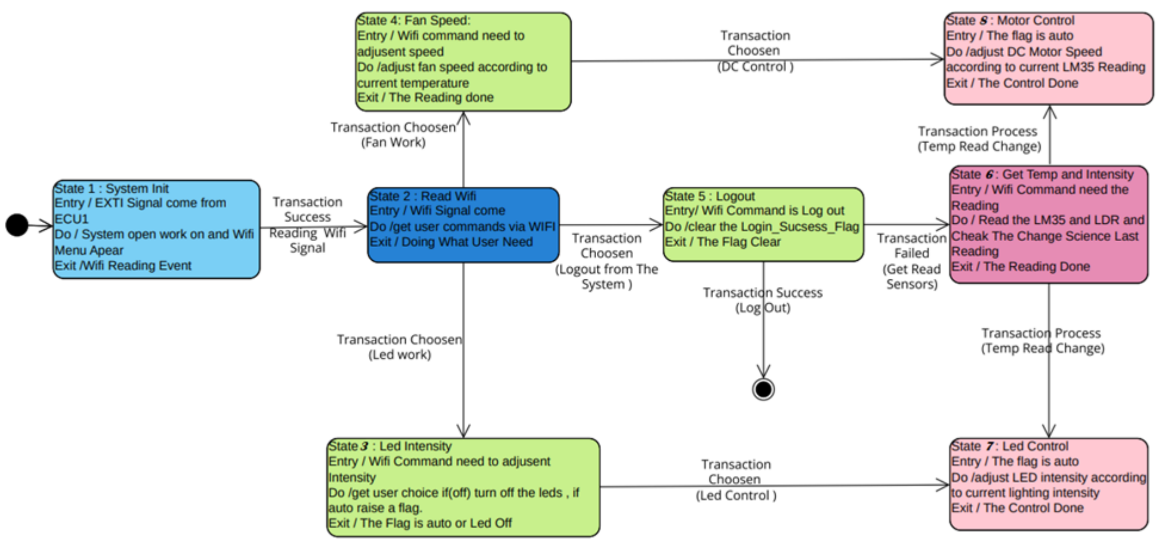


FIGURE 6: ECU2 STATE DIAGRAM

➢ List of modules

The following table shows all drivers modules used in the MCAL and HAL layer for ECU2 system:

List of all Modules	
HAL	
Module Name	**Function**
Keypad	Allow user to enter password
WIFI module	To connect the mobile phone with the microcontroller
DC Motor	To control fan speed
LED/LDR	Used to control light intensity
Temperature sensor	To read the temperature value
MCAL	
Module Name	**Function**
DIO	Used for deciding the Pin status
PORT	To configure the PINS
I2C	Communication RTC and EEPROM with MC1
USART	Communication Fingerprint and MC1
TIMER0	To generate PWM signal for the dc motor
TIMER1	To generate PWM signal for led intensity
ADC	To read the temperature sensor convert it to digital value

➢ System Components

main	
Syntax	void main()
Description	main function to run all tasks
Sync/Async	sync
Reentrancy	reentrant
Arguments	void
Return Value	void

System initialize	
Syntax	ES_t system_init(System_cfg_t copy_system_cfg)
Description	initialize all system modules(wifi - Ports - LM35- ADC- Motor- Timer0 - Timer 1- EXTI - DIO - LDR -uart)
Sync/Async	sync
Reentrancy	reentrant
Arguments	System_cfg_t copy_system_cfg
Return Value	enum (ES_t)
Read_Wifi	
Syntax	ES_t read_userWIFI()
Description	check and update state of (fan,LED, and login flags)
Sync/Async	sync
Reentrancy	NonReentrant
Arguments	void
Return Value	enum(ES_t)
Get temperature AndIntens	
Syntax	Es_t get_tempAndIntens(u32* ptr_temp, u32* ptr_intens)
Description	read temperature and intensity using LM35 and LDR respectively and check if there's a change since last reading
Sync/Async	sync
Reentrancy	NonReentrant
Arguments	u32* ptr_temp, u32* ptr_intens
Return Value	enum(ES_t)
Adjust fan Speed	
Syntax	ES_t adjust_fanSpeed(u8 speed)
Description	adjust fan speed according to current temperature
Sync/Async	sync
Reentrancy	NonReentrant
Arguments	u8 speed
Return Value	enum(ES_t)
Adjust Led Intensity	
Syntax	ES_t adjust_LedIntensity(u8 intensity)
Description	adjust LED intensity according to current lighting intensity
Sync/Async	sync
Reentrancy	NonReentrant
Arguments	u8 intensity
Return Value	enum(ES_t)

WIFI send State	
Syntax	ES_t WIFI_sendState()
Description	send temperature via WIFI to mobile APP
Sync/Async	sync
Reentrancy	NonReentrant
Arguments	void
Return Value	enum(ES_t)
return_controlToECCU1	
Syntax	void return_controlToECCU1()
Description	send trigger to ECU1 whenever user logs out
Sync/Async	sync
Reentrancy	NonReentrant
Arguments	void
Return Value	void

Global Variable

TYPE	IDENTIFIER	DESCRIPTION
u8	Login_Sucsess_Flag	if the user passed the authentication successfully this flag will be set by an external interrupt.
u8	Fan_Auto_Flag	if the user selected auto mode for the fan this flag will be set and fan speed will change according to LM35 reading.
u8	LED_Auto_Flag	if the user selected auto mode for LED this flag will be set and LED intensity will change according to LDR reading.
u8	Fan_User_State	This variable will contain fan speed set by the user.
u8	LED_User_State	This variable will contain LED state set by the user.
u8	Temp_Changed_Flag	if the current reading of LM35 doesn't lay in the same range of previous reading, this flag will be set.
u8	Intnens_Changed_Flag	if the current reading of LDR doesn't lay in the same range of previous reading, this flag will be set.

Conclusion:

In conclusion, the system can be used with high security level by using the two-verification method as discussed in the above documentation. After the authorization the system will open the door and send the control to the other ECU. The user will be able to control the devices inside the house using mobile application through the Wi-Fi module. To ensure the project organization we have used the software abstraction level with decent amount of drivers configurability.