Implementation of Parking System with FSM

Embedded Systems Programming

6/3/2019

Department of Computer Engineering, University of Denver

Submitted By: Siddhesh Suresh Padwal

DU ID (873305527)

Submitted To: Dr. Goncalo Martins

**Table of Contents**

**Abstract ……………………………………………...……..… 1**

**Introduction …………………..……………........................... 2**

**Working ………………….……………………..…………….. 4**

**BLOCK Diagram ………………………………………...……6**

**Finite State Machine ...…………………………..………….....7**

**State Explanation with code ………...………….…………..... 8**

**Conclusion ………..…………………………………………… 12**

**References …………………………………………………….. 13**

**Appendixes …………………………………………………..... 14**

Link : <https://github.com/SiddheshPadwal10/Smart-Parking-System-with-Embedded>

**Abstract**

The industrialization of the world, increase in population, slow paced city development and mismanagement of the available parking space has resulted in parking related problems. Due to the proliferation in the number of vehicles on the road, traffic problems are bound to exist. There is a dire need for a secure, intelligent, efficient and reliable system which can be used for guidance towards the parking facility, negotiation of the parking fee, along with the proper management of the parking facility. This is due to the fact that the current transportation infrastructure and car park facility developed are unable to cope with the influx of vehicles on the road. To alleviate the aforementioned problems, the smart parking system has been developed.

This paper reviews Parking System using Atollic for STM32 used for parking guidance, parking facility management. The discussed system will be able to reduce the problems which are arising due to unavailability of a reliable, efficient and modern parking system. . Subsequently, the various sensor systems used in developing the systems in addition to the recent research and commercial system on the market are examined as vehicle detection plays a crucial role in the smart parking system.

**Introduction**

The industrial growth of the world is reflected by the increase in the number of automobiles on the streets throughout the world, which has caused a lot of parking related problems. The slow paced city planning has increased the problem even more. The search for the parking space is a time consuming process which not only affects the economic activities’ efficiency, but also the social interactions and cost. Network companies cannot provide updated information of the parking facilities on the internet as the parking facilities do not cooperate with the companies. Certain big cars are not able to fit into the normally available parking spaces. Hence there is a need for a system; which can take all relevant information into consideration, for finding the parking vacancy.

Human errors are the major source of traffic accidents, therefore building in-car technologies for checking the parking lot, avoiding accidents and guidance to the parking facility is turning out to be an integral area for research. The objective of such technologies is the reduction of the burden on driver, improvement of the traffic capacity, and provision of reliable and secure car functions.

The parking meters which rely on coins or tokens is an inefficient system as it requires man power for management of the parking and exact change for paying the parking charges. Parking control and enforcement systems provide efficient and effective monitoring of meter and it also keeps a check on any violations of the parking lot. This results in best possible use of the parking space for increasing the revenue.

The Parking service, a part of Intelligent Transportation System, gives rise to different parking facilities on the basis of new functions they provide. This service not only manages the internal operations of the parking facility, but it is also designed to work with different aspects related to the parking facility.

The services which the Parking System should provide in the future are

* The parking availability information system and parking reservation system should provide advanced navigation services.
* The mobile electric commerce system and a continuously working gate system should collect the toll charges electrically.
* Provision of strong functions for facilitating administrators and managers in management of the parking facility.

The information related to the availability of unoccupied lot; before the driver enters the facility is provided on the display. An empty parking lot can be reserved by the driver through entering the password. The continuous entry and exit system facilitates a driver by getting rid of time consuming processes such as getting a ticket, and the freedom of selecting any payment method.

Further modifications result in even better systems, such as reserving the parking space online and using a smart card with it will help the driver find the destination quickly, safely and easily. Despite the system requiring no man power, it will still be able to know about the entrance and exit of the vehicles as well as the occupancy rate of the facility. These systems will also decrease the traffic congestion as the number of vehicles parked on the street will decrease. These new systems will boost the parking business by the increase in the number of customers.

The latest advancement in intelligent parking service is the parking space negotiation system which is much different than the parking information system. Parking space negotiation system uses the linking and integration of the parking facilities which results in negotiation and coordinates between the in-vehicle information system and parking facility. This system initializes the negotiation process for the parking charges, the advance reservation of the parking lot, search for the best possible path from the current position to the parking facility and then to the destination. Coordination work is an important task for the negotiation corporation. Negotiation is just like a business where both sellers and buyers decide the terms of business, for getting the best possible deal for both parties.

**Working**

In the entrance of the parking system, there is a scanner to scan the smartcard which will detect a vehicle coming. Scanning the smartcard will generate interrupt service routine to the main code (actual code). Once the ISR is triggered, a password will be sent to the system and if it matched with the correct password, the gate will open to let the vehicle get in. Otherwise, the gate is still locked. If the vehicle goes in the parking, the count of the parking space will get updated. If there is no parking space available, the system will send the message as Parking is not available so that people will save time instead of going in the parking system.

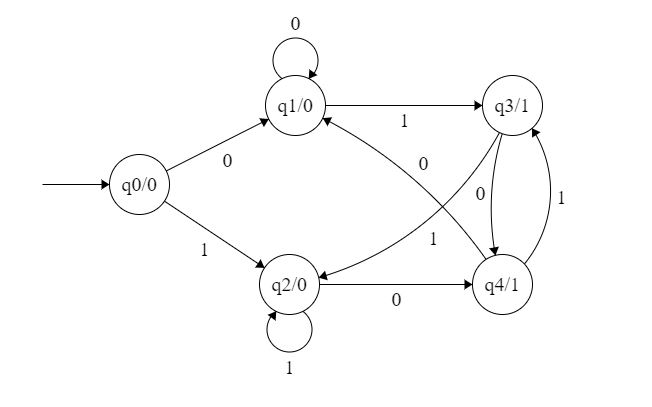
**Implementation with Finite State Machine**

A finite-state machine (FSM) or finite-state automaton, finite automaton, or simply a state machine, is a mathematical model of computation. It is an abstract machine that can be in exactly one of a finite number of states at any given time. The FSM can change from one state to another in response to some external inputs; the change from one state to another is called a transition. An FSM is defined by a list of its states, its initial state, and the conditions for each transition. Finite state machines are of two types - deterministic finite state machines and non-deterministic finite state machines. A deterministic finite-state machine can be constructed equivalent to any non-deterministic one.

**Moore Machines:**

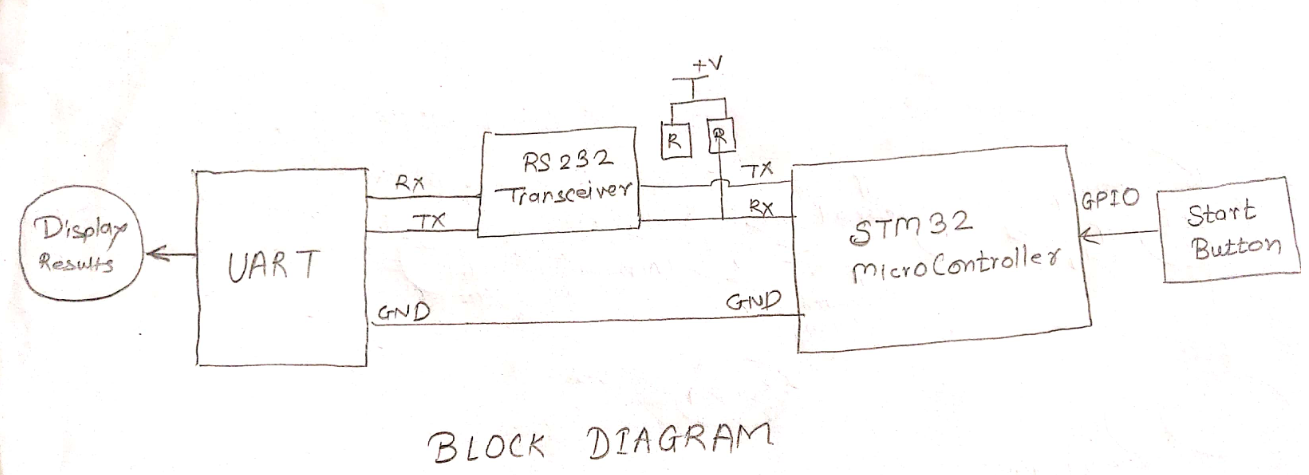
 Moore machines are finite state machines with output value and its output depends only on present state. It can be defined as (Q, q0, **∑,**O, δ, λ) where:

* Q is finite set of states.
* q0 is the initial state.
* **∑**is the input alphabet.
* O is the output alphabet.
* δ is transition function which maps Q×**∑**→ Q.
* λ is the output function which maps Q → O.



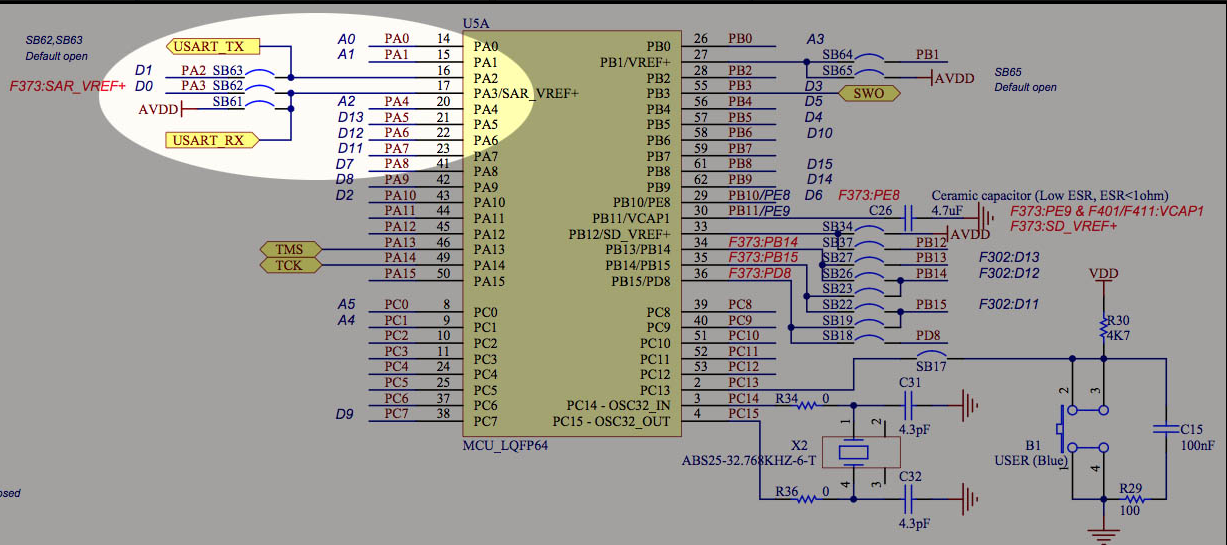
**Example diagram for Moore machine**

**BLOCK Diagram for Parking System**



**Correction:** It’s LC234X circuit used for UART in this project

**Schematic Diagram**



**Assignments**

To implement the parking system we need to have sensor (button), LED’s and UART.

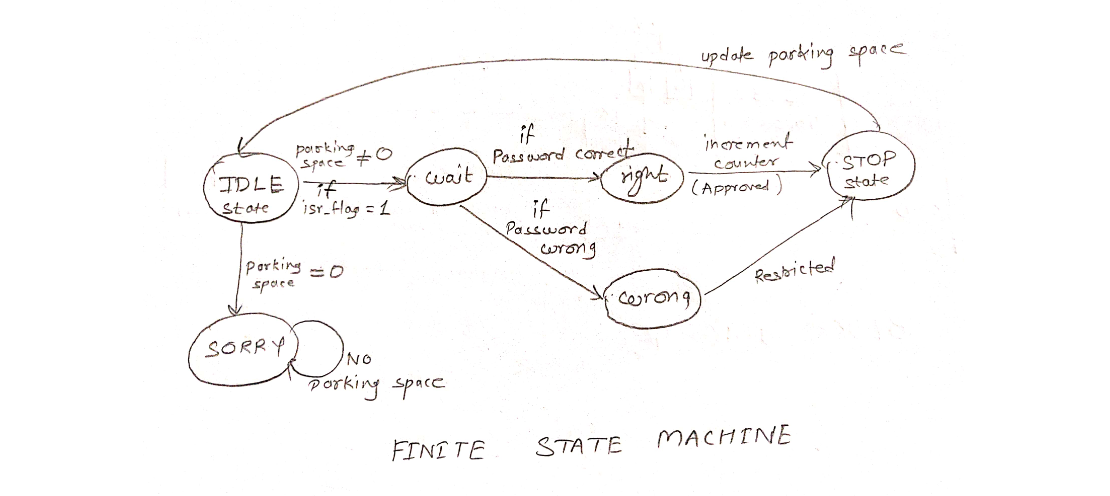
Sensors:

1. sensor\_entrance (Button): To detect the car is entered in the parking lot.

LED’s:

1. Red LED: To display the door is closed and vehicle has to stop.
2. Green LED: To display that car can pass through the parking system.
3. Blue LED: To display that the vehicle is restricted.
4. Orange LED: Please wait while processing the information.

UART: To display messages for each conditions to communicate with the car driver.

.

**FSM diagram for Parking System**

**State explanation with the code and output:**

1. **IDLE STATE :**

**Code:**

**case** *IDLE*:

GPIO\_WriteBit(GPIOD, RED | GREEN | BLUE | ORANGE, *Bit\_RESET*);

**if**(parking\_space == 0){

state = *SORRY*;

}

sprintf(sparking\_space, "%i\n\r",parking\_space);

USART\_PutString("Parking Space:");

USART\_PutString(sparking\_space);

**if**( isr\_flag\_0 == 1 ){

isr\_flag\_0 = 0;

password\_1 = 1;

password\_2 = 1;

USART\_PutString("WELCOME!\n");

GPIO\_WriteBit(GPIOD, RED, *Bit\_SET*);

state = *WAIT\_PASSWORD*;

}

**break**;

**Explanation:**

**IDLE** is the default state, where Parking space availability is shown to drivers so that if it is full they will not enter into the system and the code will enter into the **SORRY** state. Once you scanned through the smartcard which is represented by button in my project, it will activate ISR and the password can be send to the system. It will show WELCOME message to driver and it will go to the next state which is **WAIT\_PASSWORD**.

1. **WAIT STATE**

**Code:**

**case** *WAIT\_PASSWORD*:

GPIO\_WriteBit(GPIOD, ORANGE, *Bit\_SET*);

GPIO\_WriteBit(GPIOD, RED | GREEN | BLUE, *Bit\_RESET*);

USART\_PutString("Checking Authorization!\n");

**if**((password\_1 == 1)&&(password\_2== 1))

{

state = *RIGHT\_PASS*;

}

**else**

{

state = *WRONG\_PASS*;

}

**break**;

**Explanation:**

In **WAIT\_PASSWORD** state, ORANGE light will turned ON and display will show ‘Checking Authorization’.

If the password received is wrong it will go to the **WRONG\_PASS** state and if the password is correct it will go to the **RIGHT\_PASS** state.

1. **RIGHT\_PASS**

**Code:**

**case** *RIGHT\_PASS*:

GPIO\_WriteBit(GPIOD, RED | BLUE | ORANGE, *Bit\_RESET*);

GPIO\_WriteBit(GPIOD, GREEN, *Bit\_SET*);

USART\_PutString("Approved\n");

counter = 1;

state = *STOP*;

**break**;

**Explanation:**

If the password entered correctly in wait state, Green light will show in **RIGHT\_PASS** state and it will display ‘Approved’ message indicating that you can enter in the parking system. After crossing the car through the parking system, It will go to the **STOP state** counting the car has entered in the system.

1. **WRONG\_PASS**

**Code:**

**case** *WRONG\_PASS*:

**if**((password\_1== 1)&&(password\_2== 1)){

state = *RIGHT\_PASS*;

}

**else**{

GPIO\_WriteBit(GPIOD, BLUE, *Bit\_SET*);

GPIO\_WriteBit(GPIOD, RED | GREEN| ORANGE , *Bit\_RESET*);

USART\_PutString("Restricted!\n");

state = *STOP*;

}

**break**;

**Explanation:**

If the password is wrong, **wrong password** state will show BLUE light and will display that driver is restricted and it will go to **STOP** state. If driver enter right password after the first failed trial, it will go to the **right password** state, where driver can pass through this system.

1. **STOP**

**Code:**

**case** *STOP*:

parking\_space = parking\_space - counter;

GPIO\_WriteBit(GPIOD, RED, *Bit\_SET*);

GPIO\_WriteBit(GPIOD, ORANGE| BLUE | GREEN , *Bit\_RESET*);

counter = 0;

state = *IDLE*;

**break**;

**Explanation:**

In the STOP state, it will update the parking space and it will complete the whole process.

1. **SORRY state**

**Code:**

**case** *SORRY*:

GPIO\_WriteBit(GPIOD, RED, *Bit\_SET*);

USART\_PutString("NO Parking Space!\n");

state = *SORRY*;

**break**;

**Explanation:**

If the parking is full, it will show RED light and send message that No Parking Space in a parking lot.

**CONCLUSION**

This system can counter the parking problems that arise due to the unavailability of a reliable, efficient and modern Parking system.

Human errors can be avoided by installing this parking system.

There are several advantages of employing a car park system for urban planners, business owners and vehicle drivers. They offer convenience for vehicle users and efficient usage of space for urban-based companies.

Automated car park systems save time, money, space and simplify the often tedious task of parking.

Future work should be done for integrating different technologies together in order to achieve a system which is the most efficient, reliable, secure and inexpensive.

**References:**

1. <https://www.geeksforgeeks.org/mealy-and-moore-machines/>
2. <https://skyline-parking.com/>
3. <https://cityliftparking.com/solutions/case-studies>
4. https://www.westfaliausa.com/products/parking

**Appendixes:**

The complete code for the project has written below:

/\* Includes Libraries\*/

**#include** "stm32f4xx.h"

**#include** "stm32f4\_discovery.h"

**#include** "stm32f4xx.h"

**#include** "stm32f4\_discovery.h"

**#include** "stm32f4xx\_rcc.h"

**#include** "stm32f4xx\_gpio.h"

**#include** "stm32f4xx\_usart.h"

//Define delay for processing the informations

**#define** DELAY 48000000

//LED's for each conditions

**#define** RED GPIO\_Pin\_14 // PORT D

**#define** GREEN GPIO\_Pin\_12 // PORT D

**#define** BLUE GPIO\_Pin\_15 // PORT D

**#define** ORANGE GPIO\_Pin\_13 // PORT D

**volatile** **int** parking\_space = 3; //total parking space

**volatile** **char** sparking\_space[50]; //displace integer value of parking space

**volatile** **int** counter = 0; //to update number of cars going in the parking lot

//Button to trigger the ISR

**#define** BUTTON GPIO\_Pin\_0 // PORT A

GPIO\_InitTypeDef GPIO\_LED;

GPIO\_InitTypeDef GPIO\_BUTTON;

//To handle FSM

**enum** Patt {*IDLE*, *WAIT\_PASSWORD*, *RIGHT\_PASS*, *WRONG\_PASS*,*STOP*, *SORRY*, *NONE*};

**int** state = *NONE*;

**volatile** **char** isr\_flag\_0 = 0; //enable the ISR routine

//Password for each driver

**volatile** **char** password\_1 = 0;

**volatile** **char** password\_2 = 0;

**void** **InitializeIO\_2**(){

//Enable the clock on GPIO D

RCC\_AHB1PeriphClockCmd(RCC\_AHB1Periph\_GPIOD, *ENABLE*);

GPIO\_LED.GPIO\_Pin = GREEN | RED | BLUE | ORANGE; //PORT D

GPIO\_LED.GPIO\_Mode = *GPIO\_Mode\_OUT*;

GPIO\_LED.GPIO\_OType = *GPIO\_OType\_PP*; //push/pull mode

GPIO\_LED.GPIO\_Speed = *GPIO\_Speed\_2MHz*;

GPIO\_Init(GPIOD, &GPIO\_LED);

// Turn Off LEDs

GPIO\_WriteBit(GPIOD, RED | GREEN | BLUE| ORANGE, *Bit\_RESET*);

//Enable the clock on GPIO A

RCC\_AHB1PeriphClockCmd(RCC\_AHB1Periph\_GPIOA, *ENABLE*);

GPIO\_BUTTON.GPIO\_Pin = BUTTON;

GPIO\_BUTTON.GPIO\_Mode = *GPIO\_Mode\_IN*;

GPIO\_BUTTON.GPIO\_OType = *GPIO\_OType\_PP*; //push/pull mode

GPIO\_BUTTON.GPIO\_Speed = *GPIO\_Speed\_50MHz*;

GPIO\_Init(GPIOA, &GPIO\_BUTTON);

}

**void** **configure\_PA0**(**void**){

EXTI\_InitTypeDef EXTI\_InitStruct;

NVIC\_InitTypeDef NVIC\_InitStruct;

// Enable clock for SysCFG

RCC\_APB2PeriphClockCmd(RCC\_APB2Periph\_SYSCFG, *ENABLE*);

// system using PA0 for exti\_line0

SYSCFG\_EXTILineConfig(EXTI\_PortSourceGPIOA, EXTI\_PinSource0);

EXTI\_InitStruct.EXTI\_Line = EXTI\_Line0;

EXTI\_InitStruct.EXTI\_LineCmd = *ENABLE*;

EXTI\_InitStruct.EXTI\_Mode = *EXTI\_Mode\_Interrupt*;

EXTI\_InitStruct.EXTI\_Trigger = *EXTI\_Trigger\_Rising\_Falling*;

EXTI\_Init(&EXTI\_InitStruct);

// Add IRQ vector to NVIC

NVIC\_InitStruct.NVIC\_IRQChannel = *EXTI0\_IRQn*;

NVIC\_InitStruct.NVIC\_IRQChannelCmd = *ENABLE*;

NVIC\_InitStruct.NVIC\_IRQChannelPreemptionPriority = 0x00;

NVIC\_InitStruct.NVIC\_IRQChannelSubPriority = 0x00;

NVIC\_Init(&NVIC\_InitStruct);

}

//set interrupt handler

//handle PA0 interrupt

**void** **EXTI0\_IRQHandler**(**void**)

{

**if**(EXTI\_GetITStatus(EXTI\_Line0) != *RESET*){

isr\_flag\_0 = 1;

//clear flag

EXTI\_ClearITPendingBit(EXTI\_Line0);

}

}

**void** **config\_UART**()

{

GPIO\_InitTypeDef GPIO\_Struct;

USART\_InitTypeDef UART\_Struct;

// Enable clock for GPIOA

RCC\_AHB1PeriphClockCmd(RCC\_AHB1Periph\_GPIOA, *ENABLE*);

// Enable clock for USART 2

RCC\_APB1PeriphClockCmd(RCC\_APB1Periph\_USART2, *ENABLE*);

// Connect PA2 to USART2\_TX

GPIO\_PinAFConfig(GPIOA, GPIO\_PinSource2, GPIO\_AF\_USART2);

// Connect PA3 to USART2\_RX

GPIO\_PinAFConfig(GPIOA, GPIO\_PinSource3, GPIO\_AF\_USART2);

// Initialize GPIOA

GPIO\_Struct.GPIO\_Pin = GPIO\_Pin\_2 | GPIO\_Pin\_3;

GPIO\_Struct.GPIO\_Mode = *GPIO\_Mode\_AF*;

GPIO\_Struct.GPIO\_OType = *GPIO\_OType\_PP*;

GPIO\_Struct.GPIO\_PuPd = *GPIO\_PuPd\_UP*;

GPIO\_Struct.GPIO\_Speed = *GPIO\_Speed\_50MHz*;

GPIO\_Init(GPIOA, &GPIO\_Struct);

// Initialize UART

UART\_Struct.USART\_BaudRate = 9600;

UART\_Struct.USART\_HardwareFlowControl = USART\_HardwareFlowControl\_None;

UART\_Struct.USART\_Mode = USART\_Mode\_Rx | USART\_Mode\_Tx;

UART\_Struct.USART\_Parity = USART\_Parity\_No;

UART\_Struct.USART\_StopBits = USART\_StopBits\_1;

UART\_Struct.USART\_WordLength = USART\_WordLength\_8b;

USART\_Init(USART2, &UART\_Struct);

// Enable USART 2

USART\_Cmd(USART2, *ENABLE*);

USART\_ITConfig(USART2, USART\_IT\_RXNE, *ENABLE*);

}

**void** **USART\_PutChar**(**char** c)

{

**while**( !USART\_GetFlagStatus(USART2, USART\_FLAG\_TXE) );

USART\_SendData(USART2, c);

}

**void** **USART\_PutString**(**char** \*s)

{

// Send a string

**while**(\*s)

{

USART\_PutChar(\*s++);

}

}

**int** **main**(**void**){

**int** i;

// Initialization of GPIO, UART

InitializeIO\_2();

configure\_PA0();

config\_UART();

state = *IDLE*;

**while**(1){

// In the report each state is explain in detailed.

**switch**(state){

**case** *IDLE*:

GPIO\_WriteBit(GPIOD, RED | GREEN | BLUE | ORANGE, *Bit\_RESET*);

**if**(parking\_space == 0){

state = *SORRY*;

}

sprintf(sparking\_space, "%i\n\r",parking\_space);

USART\_PutString("Parking Space:");

USART\_PutString(sparking\_space);

**if**( isr\_flag\_0 == 1 ){

// if interrupt is called

isr\_flag\_0 = 0;

password\_1 = 1;

password\_2 = 1;

USART\_PutString("WELCOME!\n");

GPIO\_WriteBit(GPIOD, RED, *Bit\_SET*);

state = *WAIT\_PASSWORD*;

}

**break**;

**case** *WAIT\_PASSWORD*:

GPIO\_WriteBit(GPIOD, ORANGE, *Bit\_SET*);

GPIO\_WriteBit(GPIOD, RED | GREEN | BLUE, *Bit\_RESET*);

USART\_PutString("Checking Authorization!\n");

**if**((password\_1 == 1)&&(password\_2== 1))

{

state = *RIGHT\_PASS*;

}

**else**

{

state = *WRONG\_PASS*;

}

**break**;

**case** *RIGHT\_PASS*:

GPIO\_WriteBit(GPIOD, RED | BLUE | ORANGE, *Bit\_RESET*);

GPIO\_WriteBit(GPIOD, GREEN, *Bit\_SET*);

USART\_PutString("Approved\n");

counter = 1;

state = *STOP*;

**break**;

**case** *WRONG\_PASS*:

**if**((password\_1== 1)&&(password\_2== 1)){

state = *RIGHT\_PASS*;

}

**else**{

GPIO\_WriteBit(GPIOD, BLUE, *Bit\_SET*);

GPIO\_WriteBit(GPIOD, RED | GREEN| ORANGE , *Bit\_RESET*);

USART\_PutString("Restricted!\n");

state = *STOP*;

}

**break**;

**case** *STOP*:

parking\_space = parking\_space - counter;

GPIO\_WriteBit(GPIOD, RED, *Bit\_SET*);

GPIO\_WriteBit(GPIOD, ORANGE| BLUE | GREEN , *Bit\_RESET*);

counter = 0;

state = *IDLE*;

**break**;

**case** *SORRY*:

GPIO\_WriteBit(GPIOD, RED, *Bit\_SET*);

USART\_PutString("NO Parking Space!\n");

state = *SORRY*;

**break**;

**default**:

GPIO\_WriteBit(GPIOD, RED | GREEN | BLUE| ORANGE, *Bit\_RESET*);

**break**;

}

**for**(i=0; i<DELAY; i++);

}

}

/\*

\* Callback used by stm32f4\_discovery\_audio\_codec.c.

\* Refer to stm32f4\_discovery\_audio\_codec.h for more info.

\*/

**void** **EVAL\_AUDIO\_TransferComplete\_CallBack**(uint32\_t pBuffer, uint32\_t Size){

/\* **TODO**, implement your code here \*/

**return**;

}

/\*

\* Callback used by stm324xg\_eval\_audio\_codec.c.

\* Refer to stm324xg\_eval\_audio\_codec.h for more info.

\*/

uint16\_t **EVAL\_AUDIO\_GetSampleCallBack**(**void**){

/\* **TODO**, implement your code here \*/

**return** -1;

}