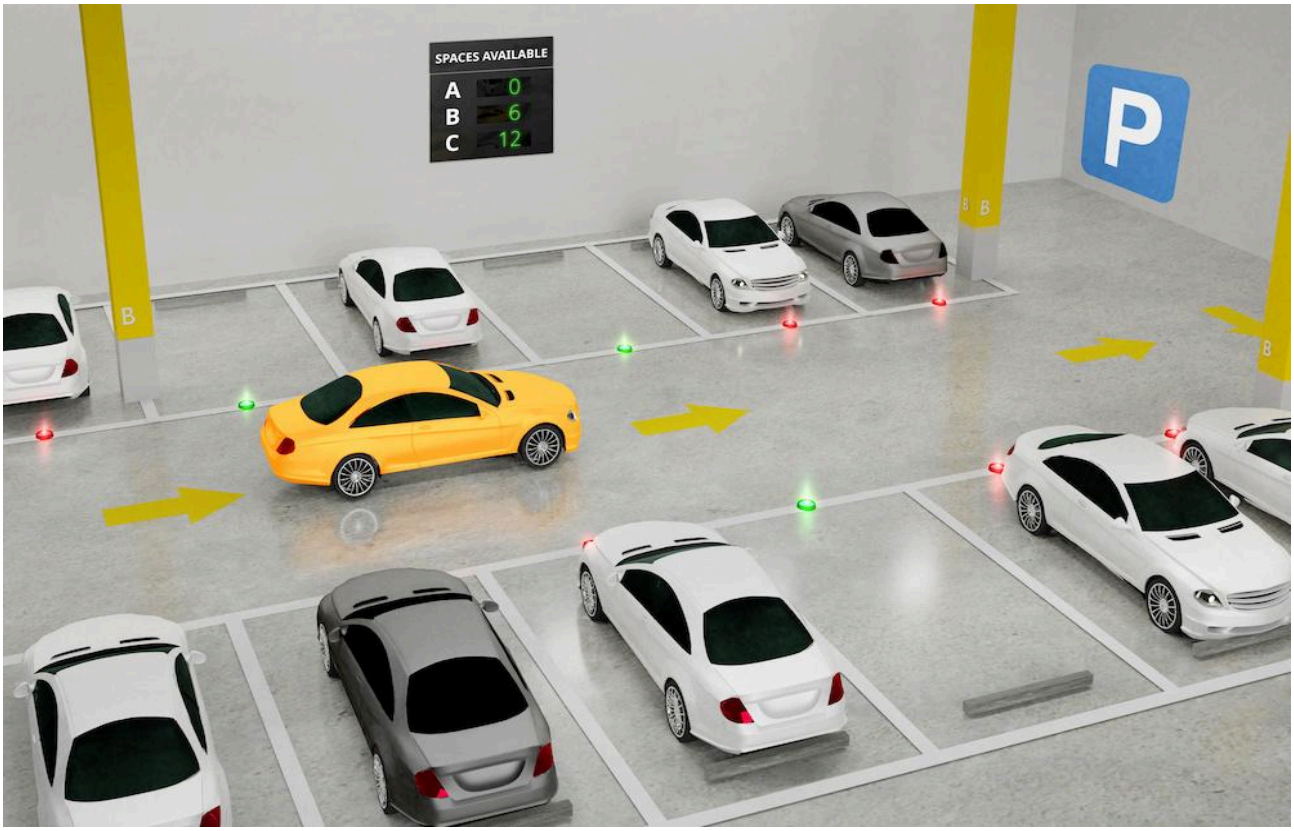
Automated Private Parking Management System



Eng : Belal Hani Sabha

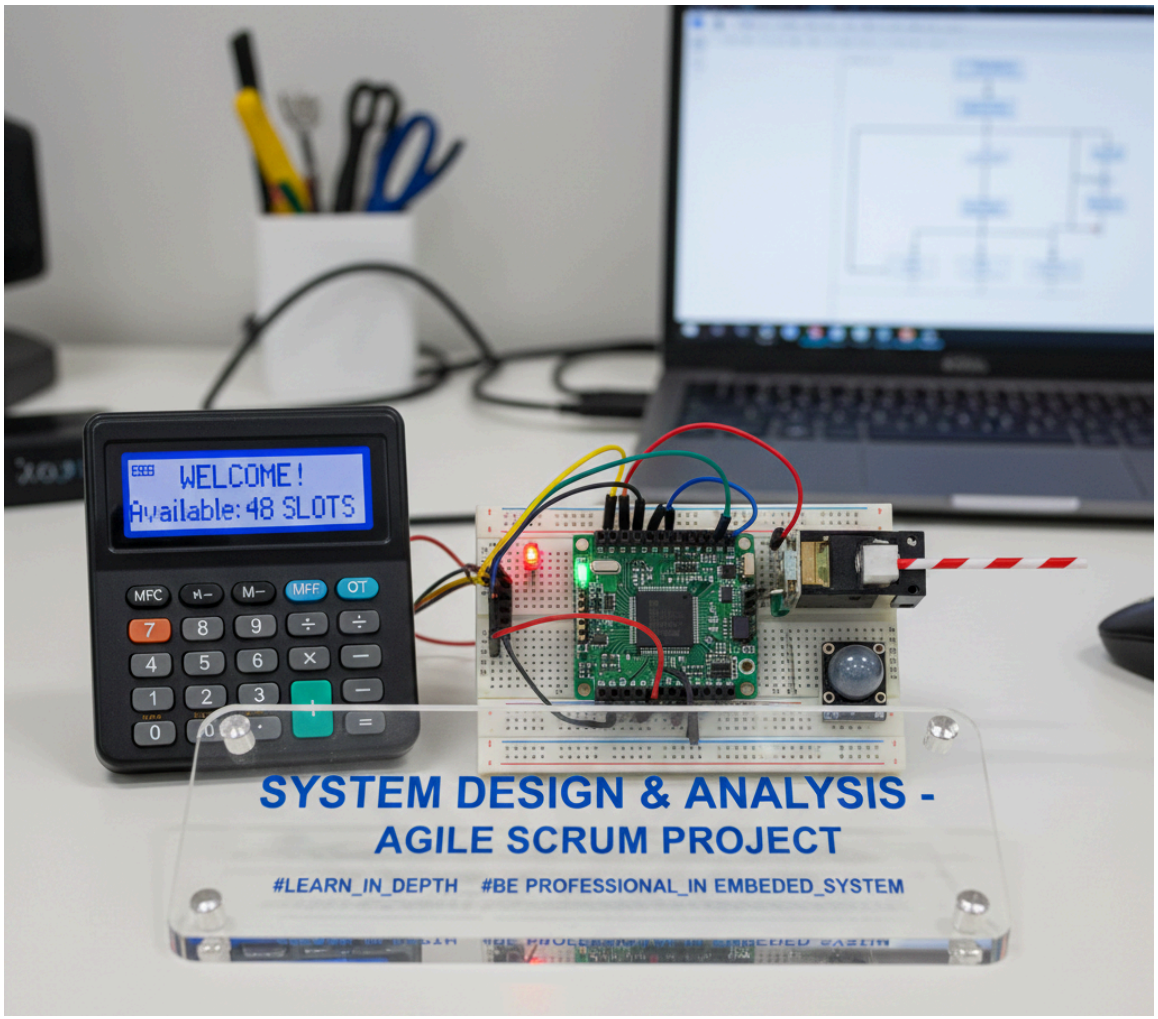
GitRepo:https://github.com/belalsabha/Master_Embedded_Systems/tree/main/Second%20Term%20%20Project%20-%20SPVPS

Table of Contents:

Project Description

The Smart Private Vehicle Parking System {SPVPS} is an automated solution for controlling vehicle access in private parking Slots. The system uses an STM32F103C6 microcontroller to manage entry and exit gates through a whitelist based authentication system.

The system integrates multiple hardware peripherals, including LCD displays , Keypads , Servo Motors , PIR sensors , and LEDs , to ensure seamless and safe parking experience . Only authorized vehicles can enter or exit and the system monitors the number of available parking slots in real-time.



Case Study:

Specifications:

1-Display welcome message and available parking slots.
2-Verify vehicle IDs against authorized list.
3-Open and close gates for authorized vehicles.
4-Control vehicle exit and update slot count.
5-Admin interface to add or remove vehicle IDs.
6-Optional logging of entries or exits.

Assumptions:

1-Authorized vehicle list is stored in the MCU.
2-Each vehicle has a unique ID .
3-LCD and Keypad are connected and powered.
4-Entry and exit gates use Servo Motors .
5-PIR sensors used to detect anything under the gate.
6-Drivers enter the vehicle number via RFID Card(Terminal).
7-System operates in any environments.
8-Hardware malfunctions are indicated via LEDs .
9-Maximum authorized vehicles less than three .
10-Gate opening/closing time less than five seconds to Ensure safety .

Method :

Agile Scrum :

I used Jira Software to manage this project using Agile Scrum which emphasizes iterative development and delivering functional parts of the system in short cycles called Sprints .

I organized the project as follows:

1-**Epics :** Major system modules such as MCAL Drivers , HAL Drivers , Logic , Safety and Feedback , Verification and Validation}

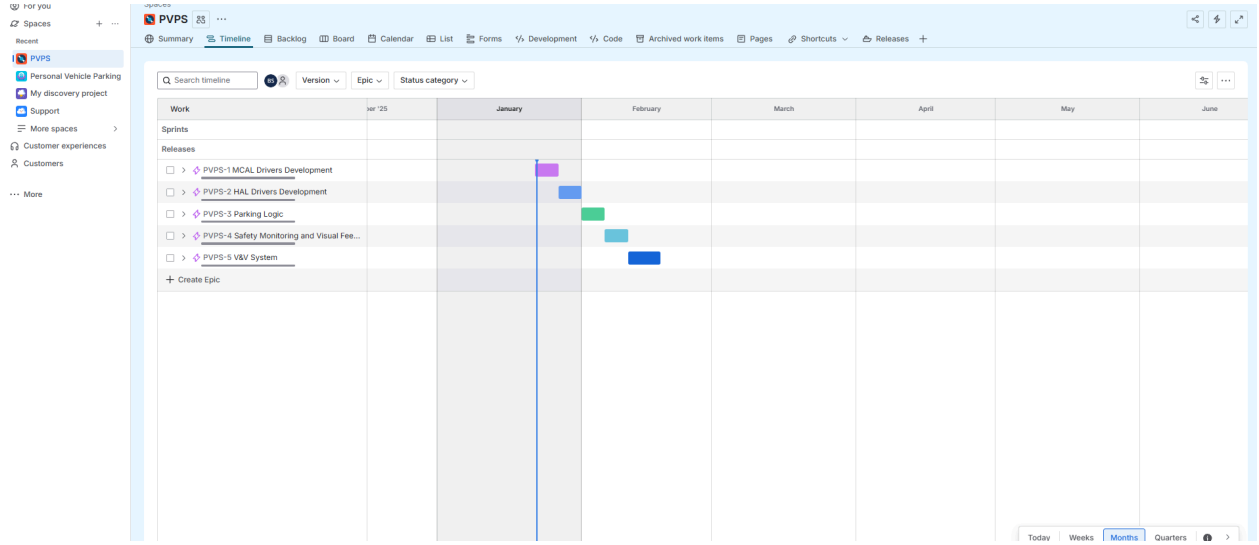
2-**User Stories :** Specific every features under each Epic.

3-**Story Points :** Each User Story was assigned points to estimate its relative time.

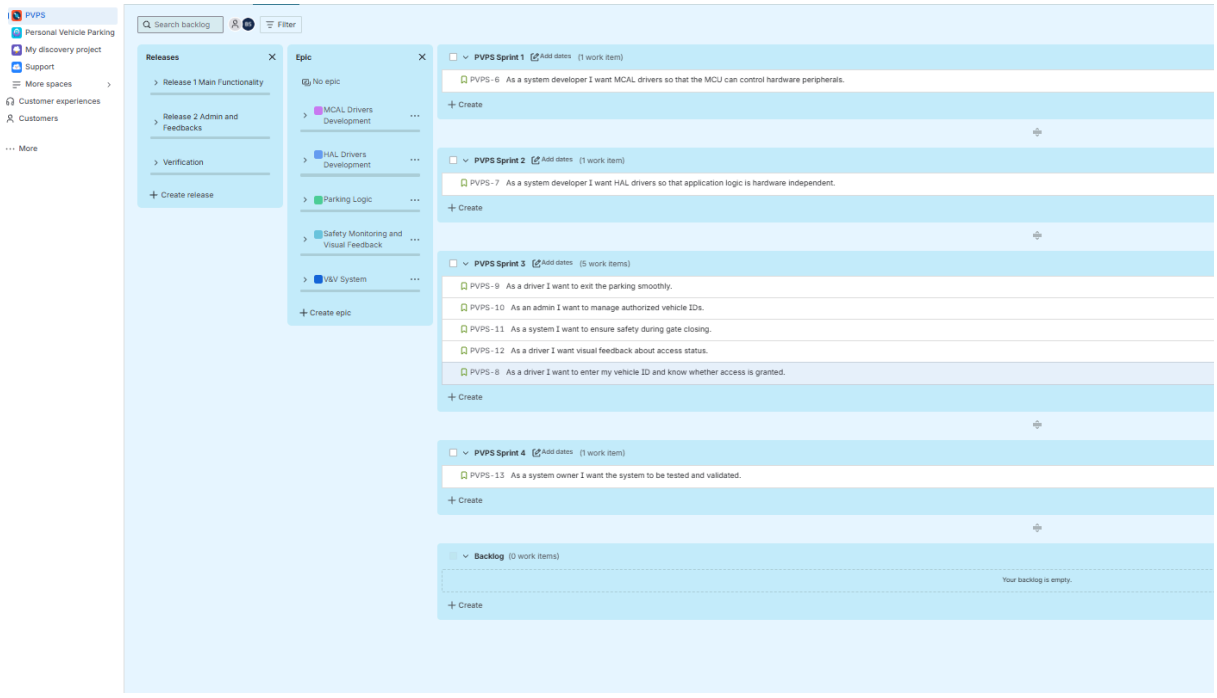
4-**Four Sprints :** Each Sprint focused on implementing one or more related User Stories .

5-**Three Releases :** planed sprints and put in releases based on functionality

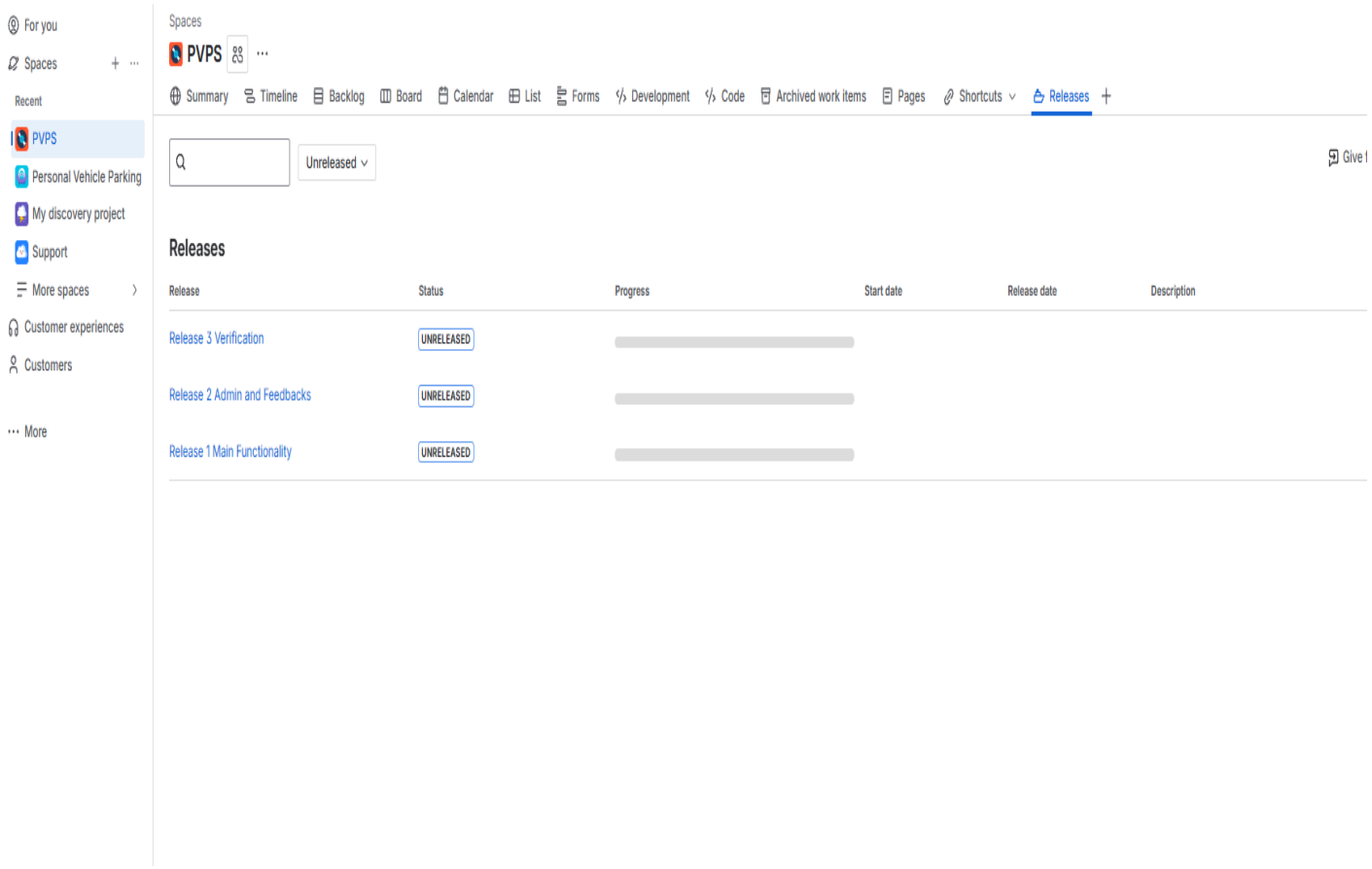
Epics:



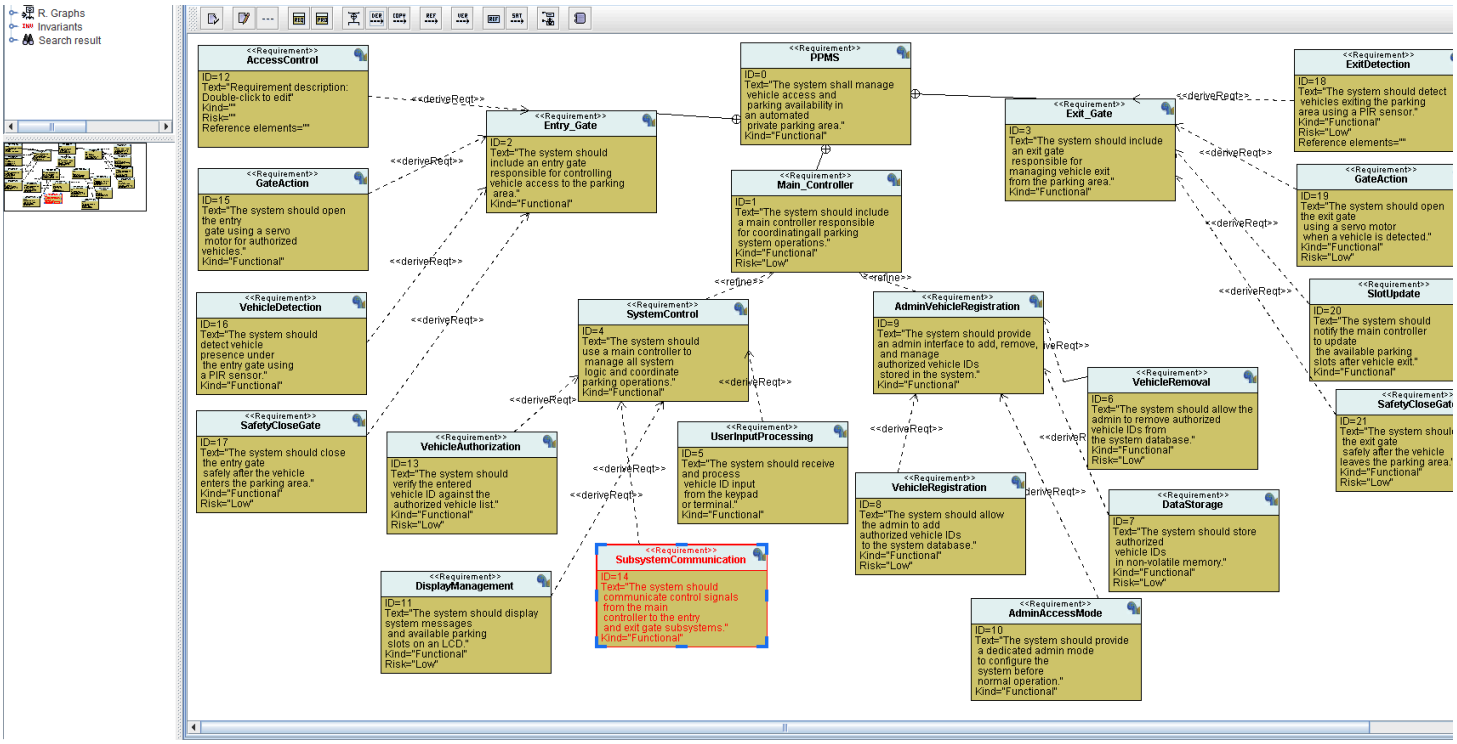
User Stories & Sprints :



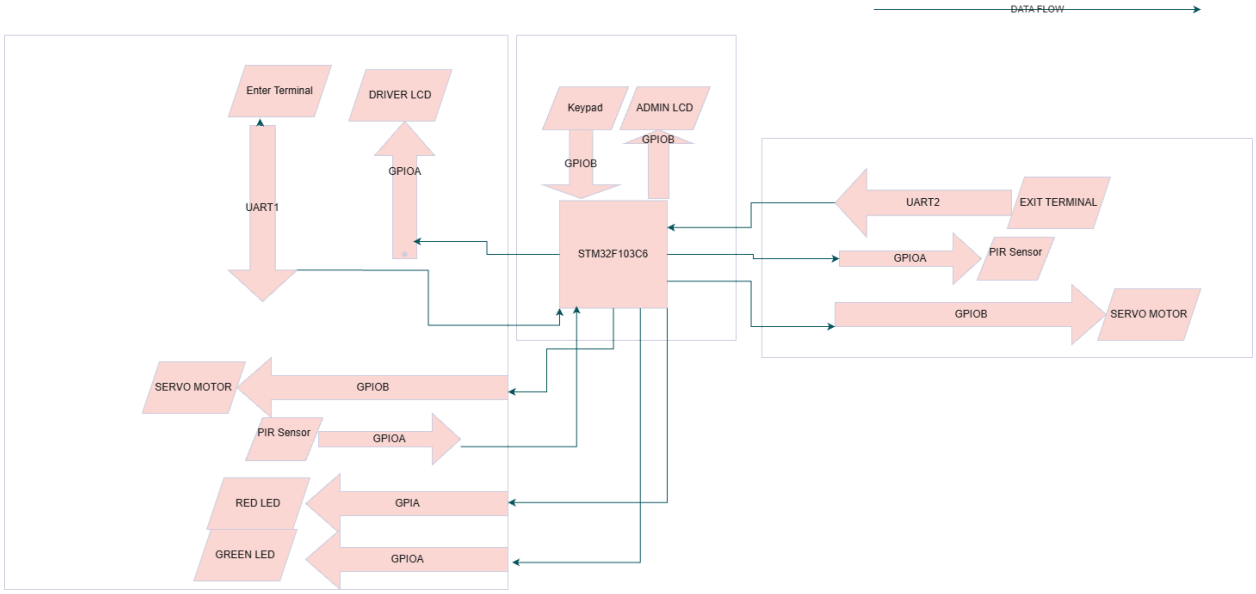
Releases :


For you
Spaces
Recent
PVPS
Personal Vehicle Parking
My discovery project
Support
More spaces
Customer experiences
Customers
More
Spaces
PVPS
Summary
Timeline
Backlog
Board
Calendar
List
Forms
Development
Code
Archived work items
Pages
Shortcuts
Releases
Unreleased
Releases
Release
Status
Progress
Start date
Release date
Description
Release 3 Verification
UNRELEASED
Release 2 Admin and Feedbacks
UNRELEASED
Release 1 Main Functionality
UNRELEASED

Requirements:



Space Exploration:



We will use **STM32F103C6 MCU** responsible for reading sensor controlling actuators and managing the system logic.

RFID Sensors : Detect and identify authorized vehicles at entry and exit.

PIR Sensors : Detect vehicle presence under the gate to prevent collisions.

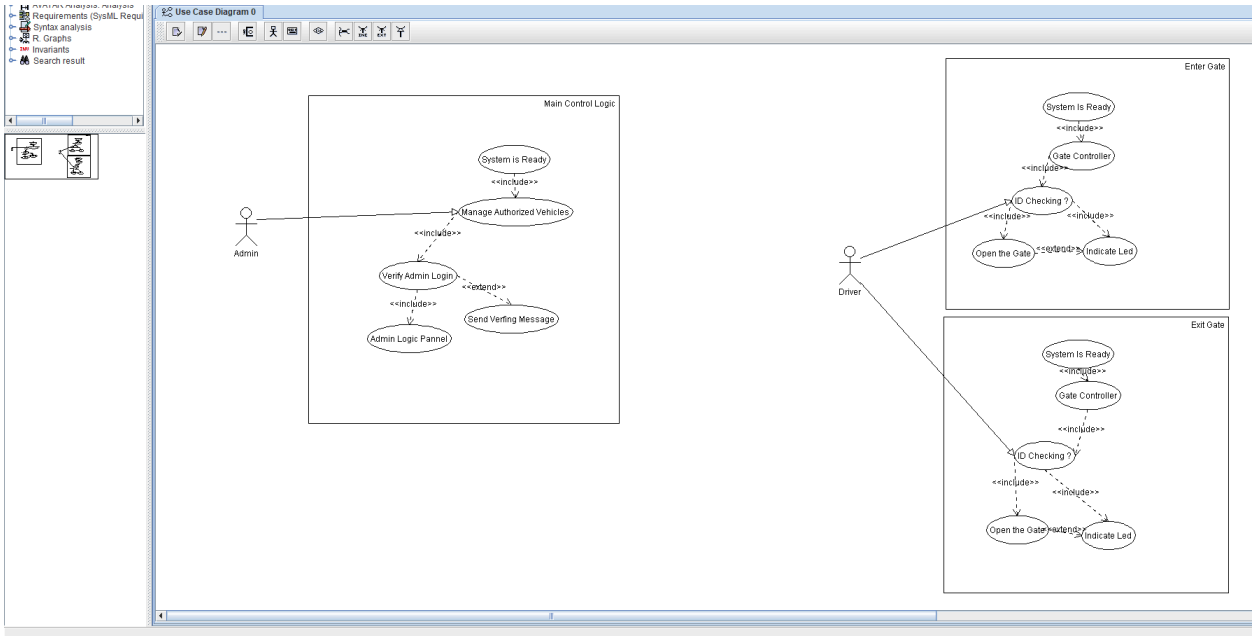
Servo Motors : Control the opening and closing gates.

LCD : Display welcome messages, available slots, and alerts.

Keypad : enter vehicle ID manually.

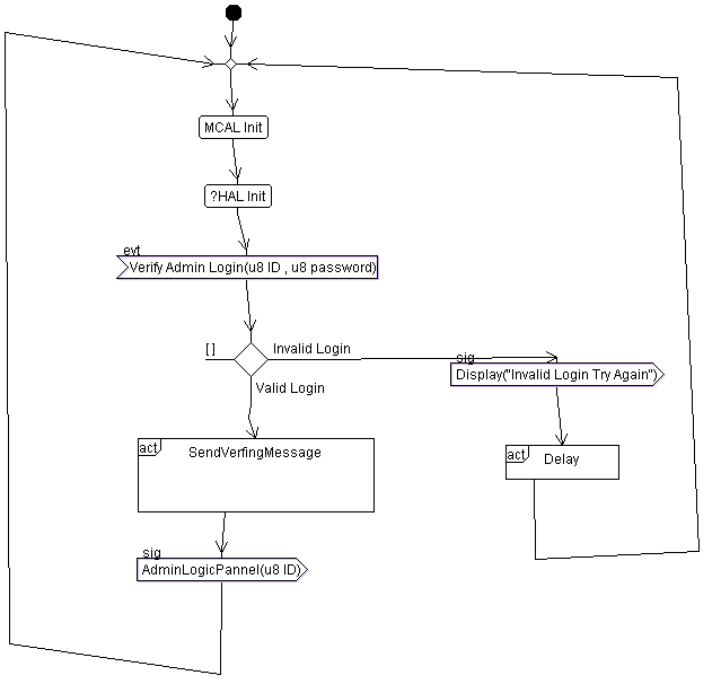
System Analysis:

1- Use Case Diagram:

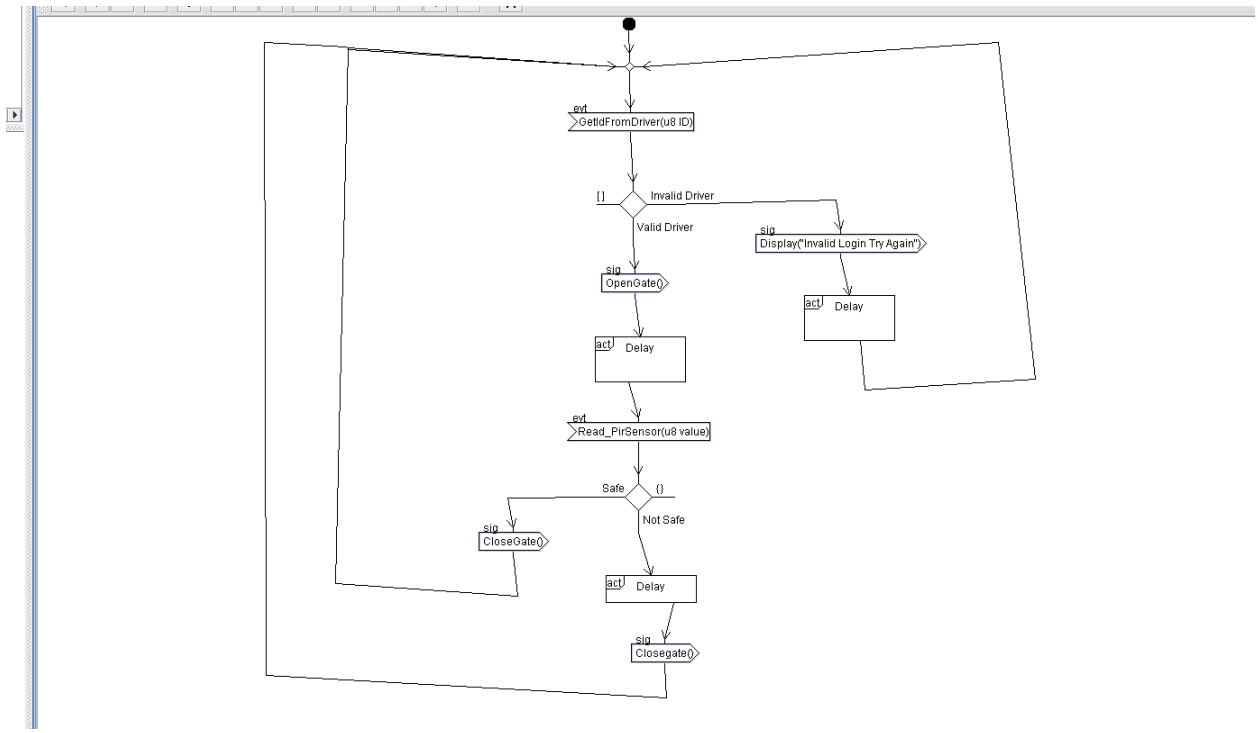


2-Activity Diagram:

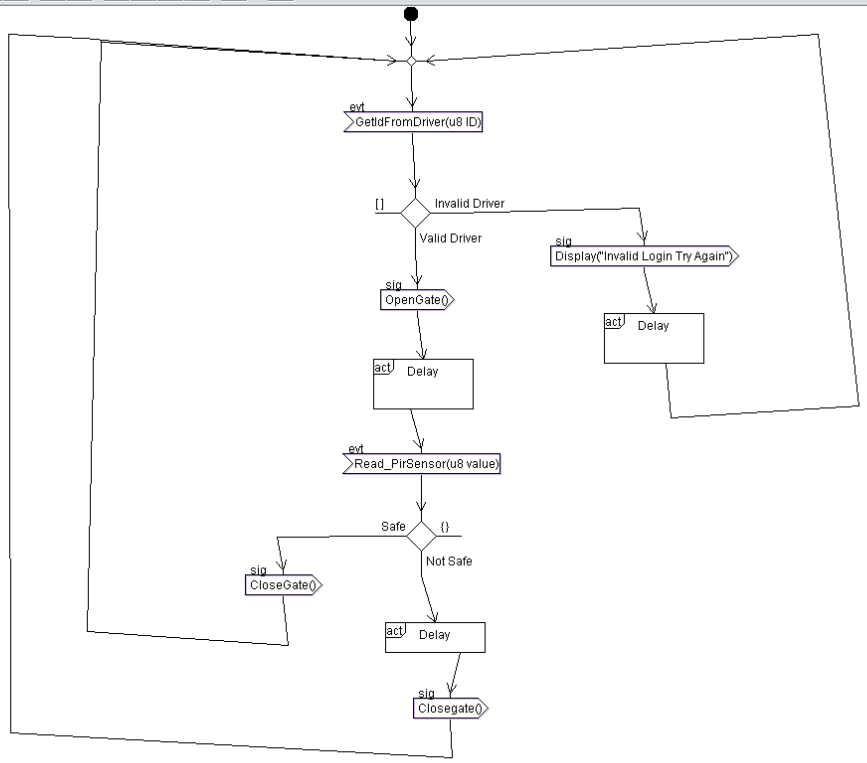
Main Control



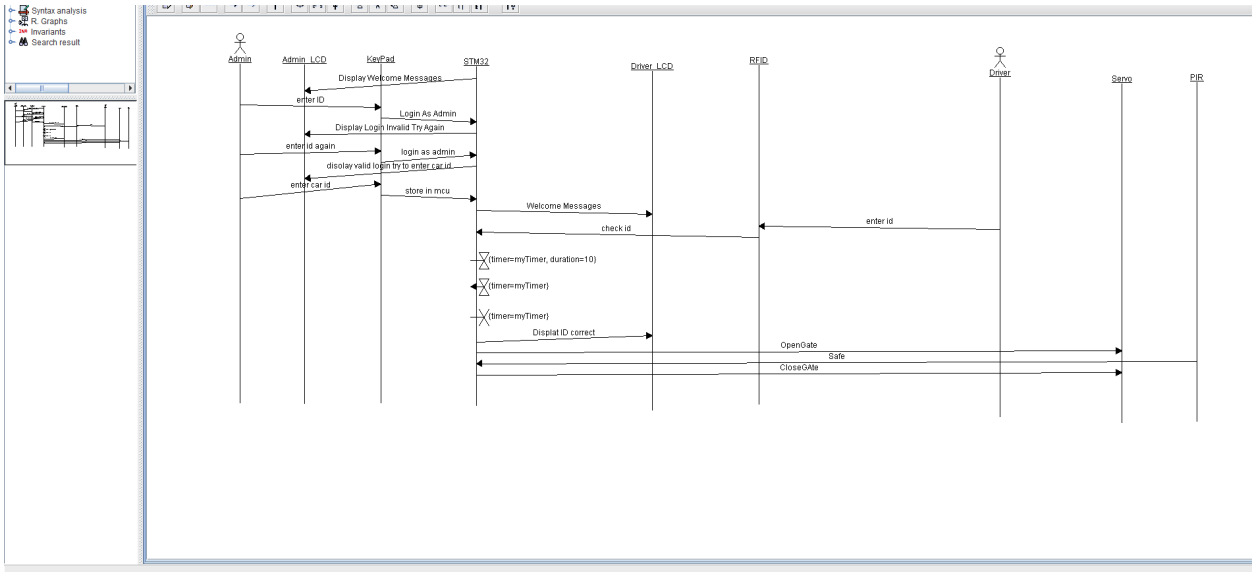
Enter Gate


evt
GetIdFromDriver(u8 ID)
[]
Invalid Driver
Valid Driver
sig
Display("Invalid Login Try Again")
sig
OpenGate()
act
Delay
act
Delay
evt
Read_PirSensor(u8 value)
Safe
()
Not Safe
sig
CloseGate()
act
Delay
sig
Closegate()

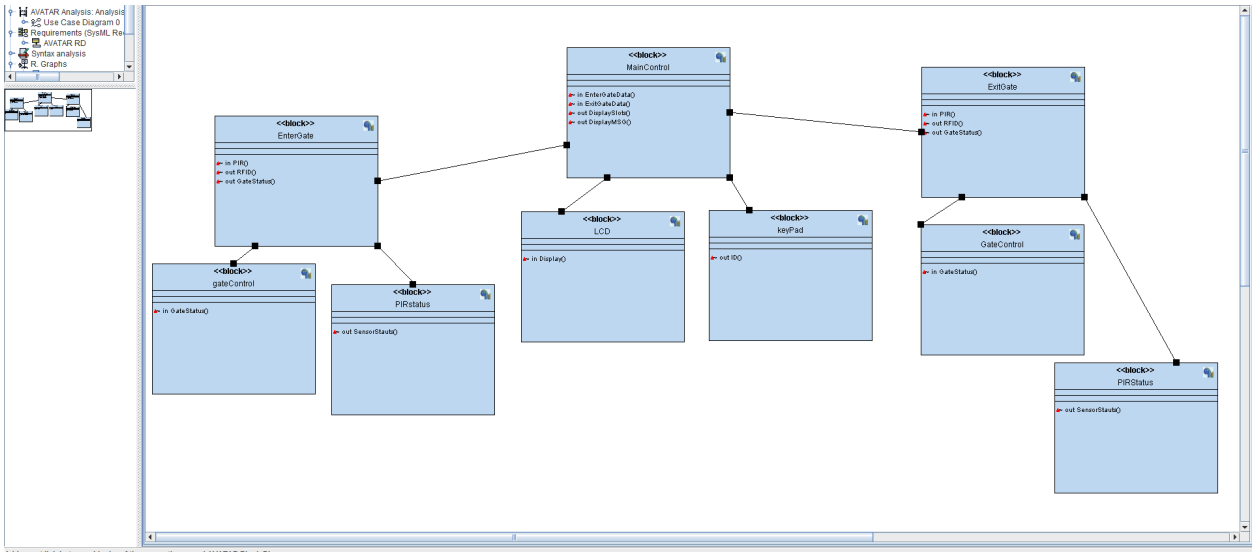
Exit Gate


evt
GetIdFromDriver(u8 ID)
[]
Invalid Driver
Valid Driver
sig
Display("Invalid Login Try Again")
sig
OpenGate()
act
Delay
act
Delay
evt
Read_PirSensor(u8 value)
Safe
{}
Not Safe
sig
CloseGate()
act
Delay
sig
Closegate()

3-Sequence Diagram:

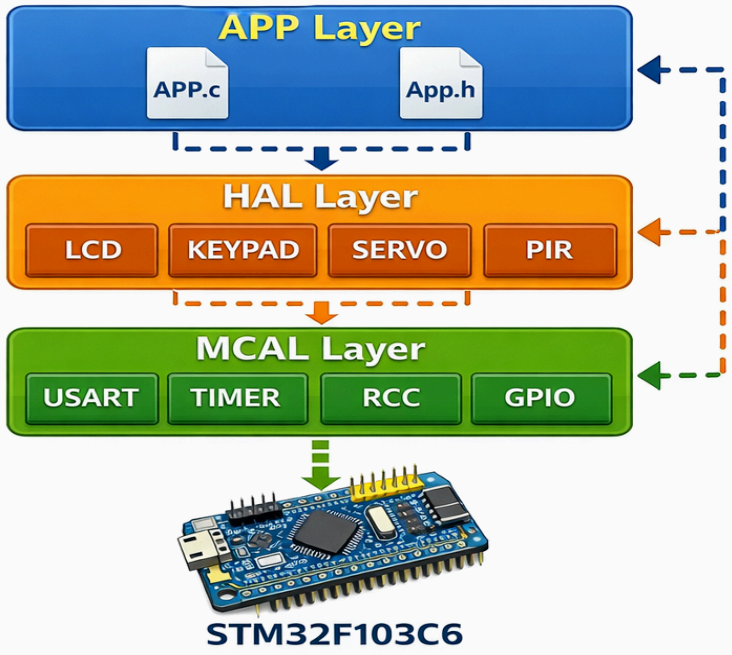


System Design:



Coding & System Architecture :

The system architecture of the parking management system was designed using a **Three layers** for ensuring modularity , scalability and hardware abstraction .



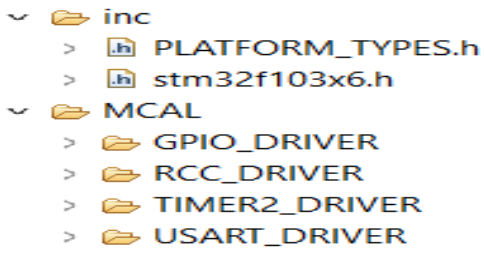
The layers are as follows:

1-MCAL (Microcontroller Abstraction Layer) :

The **MCAL layer** is responsible for direct interaction with the STM32F103C6 microcontroller . It provides low level drivers and services that manage the hardware without exposing details to higher layers . The implemented drivers include :

-GPIO : input/output control for LEDs , sensors and switches.
-USART : Serial communication for car ID validation.
-TIMER2 : Accurate timing for delay and servo control.
-RCC (Clock) : System clock configuration and peripheral clock enabling.

These drivers provide the foundation for upper layers, ensuring precise timing and reliable hardware access.

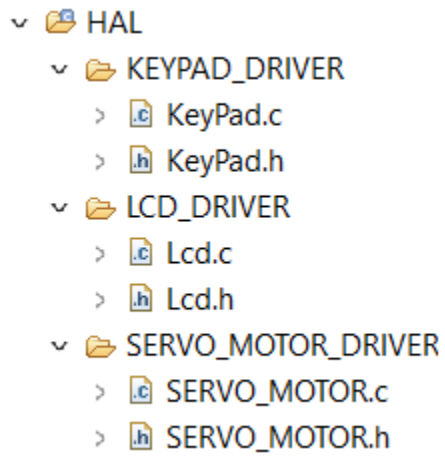


2-HAL (Hardware Abstraction Layer) :

The **HAL layer** builds on top of MCAL to represent complete hardware modules. Each module uses MCAL drivers internally, providing an abstraction to the application layer. The implemented HAL modules include:

-**LCD Display** : For driver and admin interfaces.
-**Keypad** : For user input and admin ID entry.
-**Servo Motors** : Control Entry & Exit Gates .
-**LEDs** : Status indicators.
-**PIR Sensors** : Vehicle detection at gates.

HAL ensures that the application layer can control hardware modules without dealing with low level details.

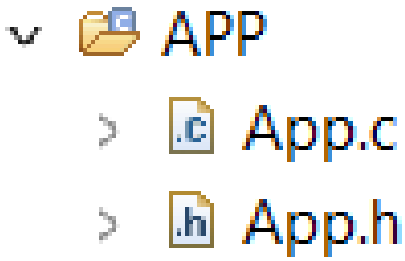


3-APP (Application Layer) :

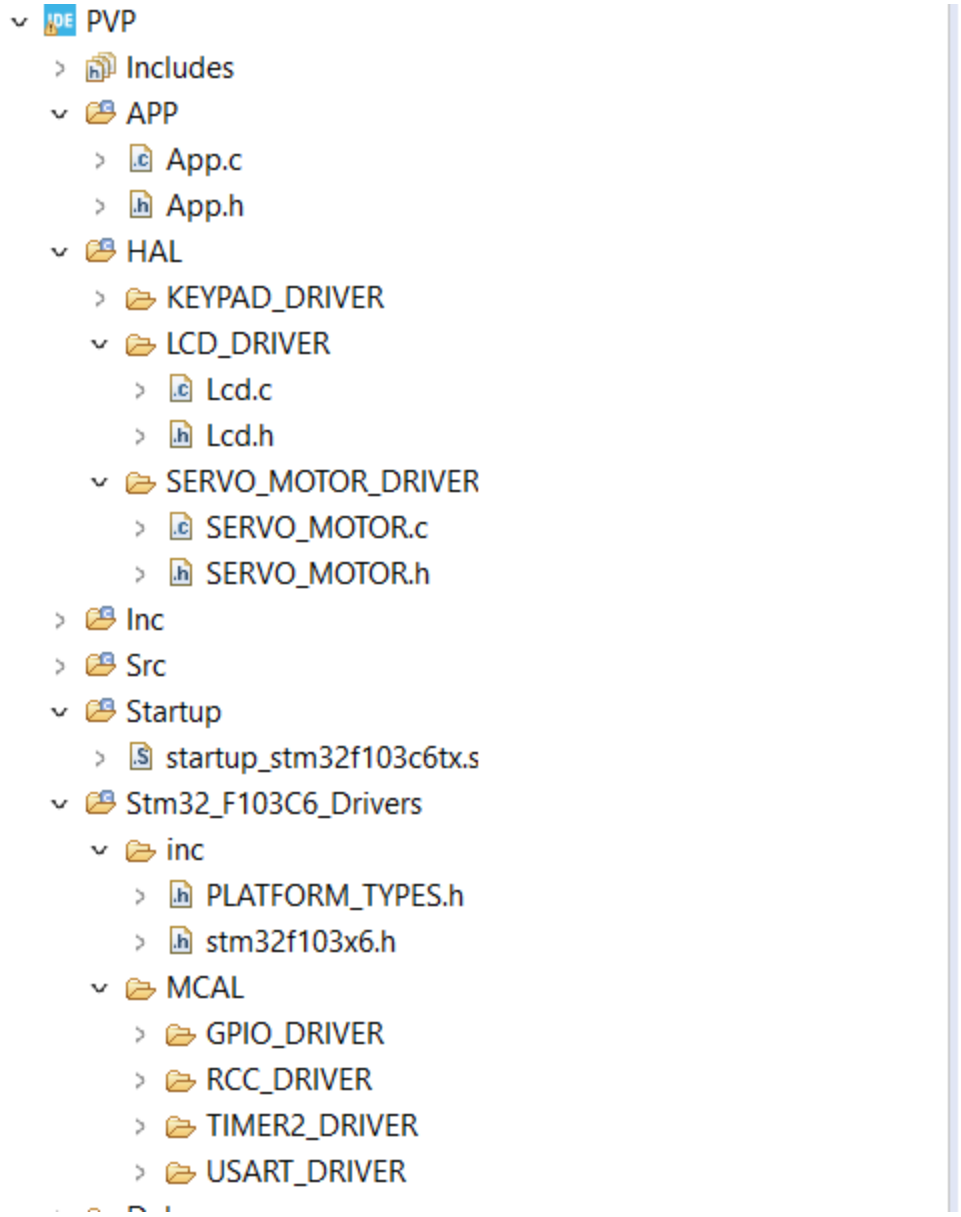
The APP layer was designed to implement the main system logic , contains the main parking system logic including :

-Managing parking slots and authorized car IDs
-Handling entry and exit gates using servo motors
-Displaying information on LCDs for admin and drivers
-Processing system events triggered by PIR sensors or USART interrupts

APP uses HAL APIs to interact with hardware, ensuring a clear separation between logic and hardware.



This layered approach allows easy maintenance , system scalability and hardware independence while ensuring all components work together seamlessly .


PVP
Includes
APP
App.c
App.h
HAL
KEYPAD_DRIVER
LCD_DRIVER
Lcd.c
Lcd.h
SERVO_MOTOR_DRIVER
SERVO_MOTOR.c
SERVO_MOTOR.h
Inc
Src
Startup
startup_stm32f103c6tx.s
Stm32_F103C6_Drivers
inc
PLATFORM_TYPES.h
stm32f103x6.h
MCAL
GPIO_DRIVER
RCC_DRIVER
TIMER2_DRIVER
USART_DRIVER

System Advantages:

-Modular Architecture: The system is divided into three layers which simplifies maintenance and future upgrades.

-Hardware Abstraction: Upper layers do not directly access the microcontroller hardware making the system more portable.

-Real Time Response: Using interrupts for USART and PIR sensors ensures fast and reliable response to vehicle entry and exit.

-User Friendly Interface: LCD displays provide clear instructions for both admin and drivers.

-Access Control: Only authorized car IDs are allowed improving security.

-Parking Slot Management: Real time tracking of available parking slots ensures efficient space utilization.

Future Improvements:

-Increase Capacity: Support for more cars and multiple gates simultaneously.

-Remote Monitoring: Integration with mobile or web applications for remote management.

-Database Storage: Storing car IDs and logs in a non volatile memory for record keeping.

-Additional Sensors: Adding cameras or IR sensors to enhance vehicle detection and safety.

-Automatic Alerts: Send notifications when parking is full or unauthorized access is attempted.

Testing:

-Static Testing: The code was reviewed and analyzed without executing the program to verify logical correctness, proper function definitions, and consistent use of global variables.

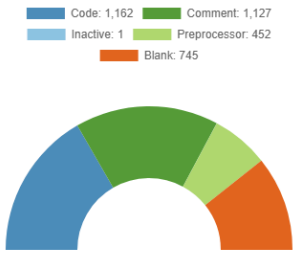
-Functional Testing: Each module (MCAL drivers, HAL modules and APP logic) was tested individually to ensure correct behavior.

-Integration Testing: The complete system was tested to verify communication between layers, gate operation, sensor detection, and user interface performance.

3,158	24	65
Lines	Files	Functions

Line Breakdown

Categorization by Line Type

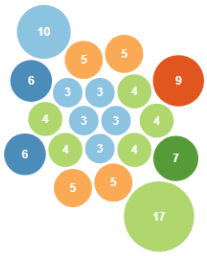


Directory Structure

Segment Size by Lines of Code

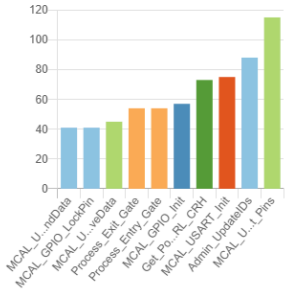
Most Complex Functions

Complexity by the McCabe Cyclomatic Metric



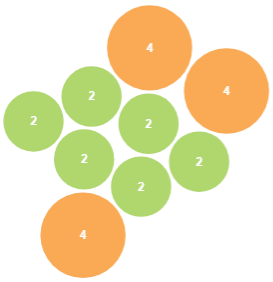
Largest Functions

Height by Total Lines



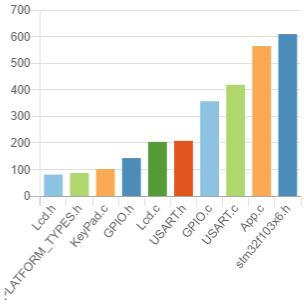
Most Complex Files

Complexity by the McCabe Cyclomatic Metric

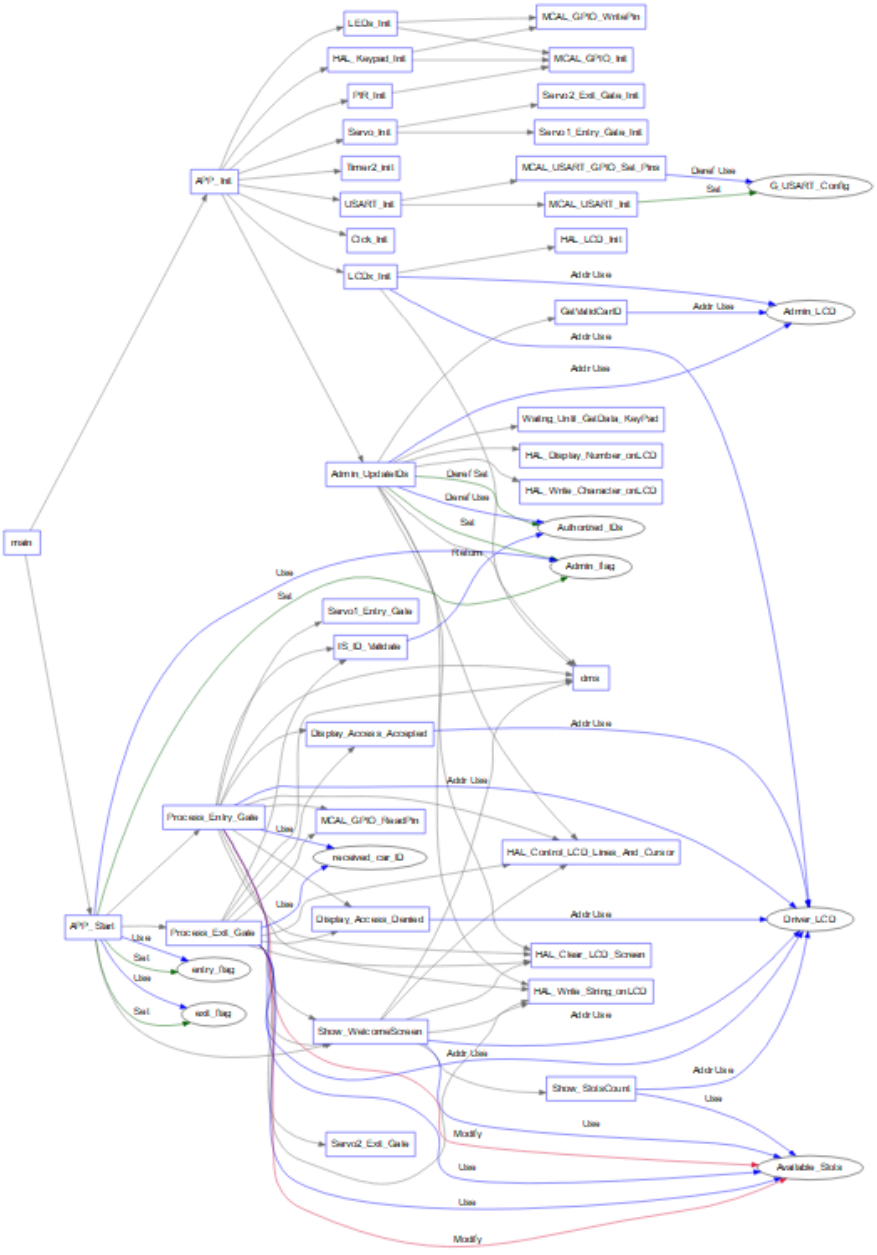


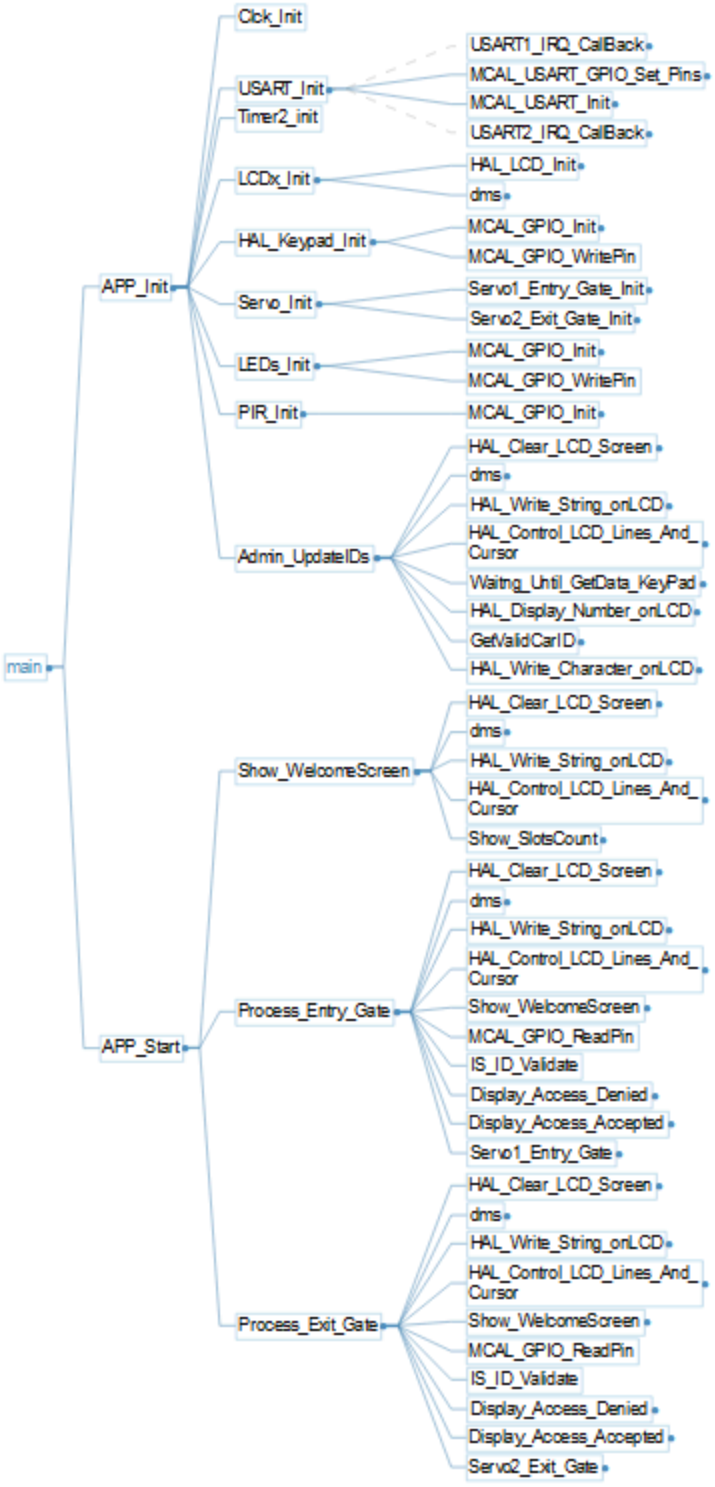
Largest Files

Height by Total Lines

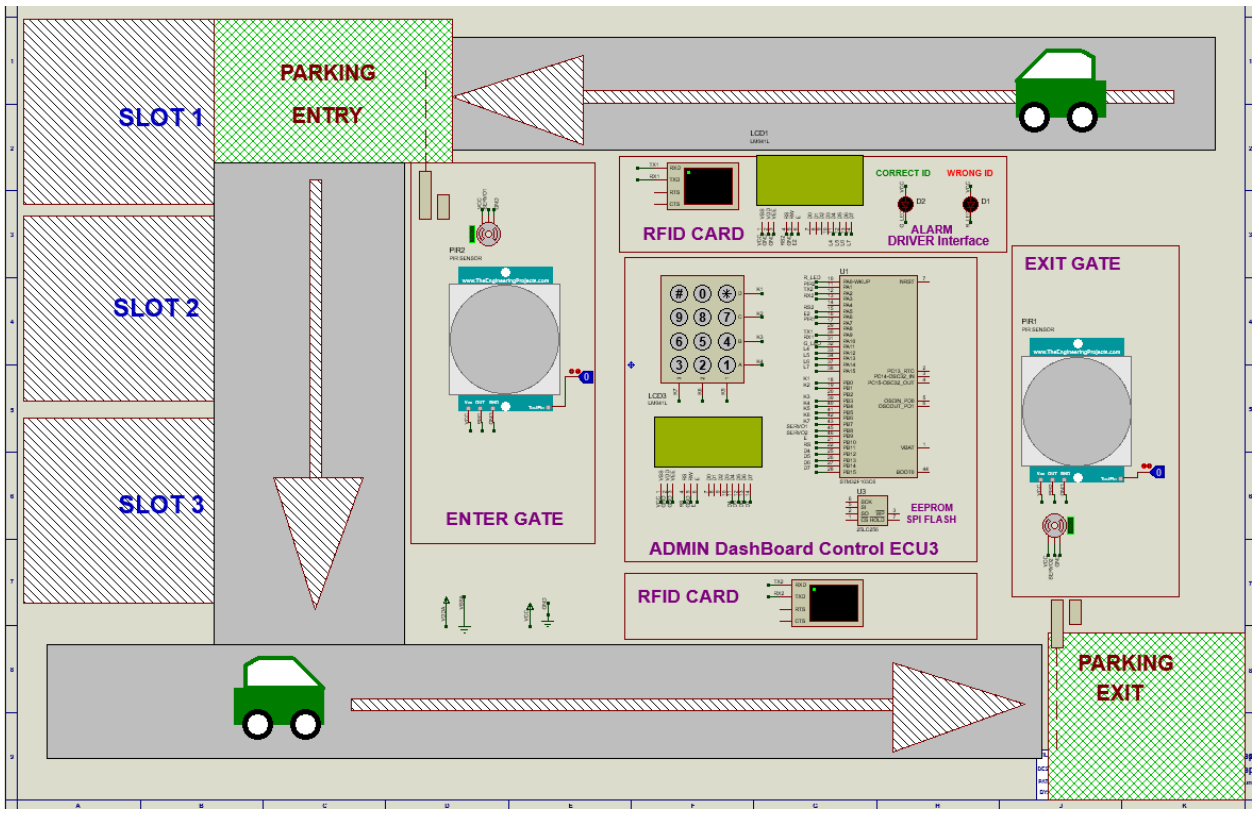


```
--------------------------------
rototypes
--------------------------------
d Clck_Init(void);                                          //  >> Test is Done
d USART_Init(void);                                         //  >> Test is Done
d LCDx_Init(void);                                          //  >> Test is Done
d Servo_Init(void);                                         //  >> Test is Done
d LEDs_Init(void);                                          //  >> Test is Done
d PIR_Init(void);                                           //  >> Test is Done

d Show_WelcomeScreen(void);                                 //  >> Test is Done
d Show_SlotsCount(void);                                    //  >> Test is Done

d Admin_UpdateIDs(void);                                    //  >> Test is Done
t8_t GetValidCarID(void);                                   //  >> Test is Done
t8_t Waitng_Until_GetData_KeyPad(void);                     //  >> Test is Done

d USART1_IRQ_CallBack(void);                                //  >> Test is Done
d USART2_IRQ_CallBack(void);                                //  >> Test is Done


t8_t IS_ID_Validate(uint8_t id);                            //  >> Test is Done
d Flashing_LED(GPIOx_t* GPIOx, uint16_t pin, uint8_t times); //  >> Test is Done
d Display_Access_Accepted(uint8_t id, uint8_t is_entry);    //  >> Test is Done
d Display_Access_Denied(uint8_t id);                        //  >> Test is Done
d Process_Entry_Gate(void);                                 //  >> Test is Done
d Process_Exit_Gate(void);                                  //  >> Test is Done
```


LEDs_Init
MCAL_GPIO_WritePin
HAL_Keypad_Init
MCAL_GPIO_Init
PIR_Init
Servo2_Exit_Gate_Init
Servo_Init
Servo1_Entry_Gate_Init
Timer2_Init
MCAL_USART_GPIO_Set_Pins
Uses/ Uses
Set
G_USART_Config
USART_Init
MCAL_USART_Init
APP_Init
Click_Init
HAL_LCD_Init
LCDs_Init
Addr Uses
GetValidCardID
Addr Uses
Admin_LCD
Addr Uses
Addr Uses
Waiting_Until_GetData_KeyPad
HAL_Display_Number_onLCD
Admin_UpdateIDs
Uses/ Set
HAL_Write_Character_onLCD
Uses/ Uses
Set
Authorized_IDs
main
Return
Admin_flag
Uses
Set
Servo1_Entry_Gate
IS_ID_Validate
dms
Display_Access_Accepted
Addr Uses
Addr Uses
Process_Entry_Gate
MCAL_GPIO_ReadPin
Uses
received_car_ID
HAL_Control_LCD_Lines_And_Cursor
Uses
Display_Access_Denied
Addr Uses
Driver_LCD
APP_Start
Process_Exit_Gate
Uses
Set
entry_flag
Uses
Set
exit_flag
HAL_Clear_LCD_Screen
HAL_Write_String_onLCD
Addr Uses
Show_WelcomeScreen
Addr Uses
Addr Uses
Show_SlotsCount
Uses
Uses
Modify
Servo2_Exit_Gate
Available_Slots
Uses
Uses
Modify


main
APP_Init
Clck_Init
USART_Init
USART1_IRQ_CallBack
MCAL_USART_GPIO_Set_Pins
MCAL_USART_Init
USART2_IRQ_CallBack
Timer2_init
LCDx_Init
HAL_LCD_Init
dms
HAL_Keypad_Init
MCAL_GPIO_Init
MCAL_GPIO_WritePin
Servo_Init
Servo1_Entry_Gate_Init
Servo2_Exit_Gate_Init
LEDs_Init
MCAL_GPIO_Init
MCAL_GPIO_WritePin
PIR_Init
MCAL_GPIO_Init
Admin_UpdateIDs
HAL_Clear_LCD_Screen
dms
HAL_Write_String_onLCD
HAL_Control_LCD_Lines_And_Cursor
Waitng_Until_GetData_KeyPad
HAL_Display_Number_onLCD
GetValidCarID
HAL_Write_Character_onLCD
APP_Start
Show_WelcomeScreen
HAL_Clear_LCD_Screen
dms
HAL_Write_String_onLCD
HAL_Control_LCD_Lines_And_Cursor
Show_SlotsCount
Process_Entry_Gate
HAL_Clear_LCD_Screen
dms
HAL_Write_String_onLCD
HAL_Control_LCD_Lines_And_Cursor
Show_WelcomeScreen
MCAL_GPIO_ReadPin
IS_ID_Validate
Display_Access_Denied
Display_Access_Accepted
Servo1_Entry_Gate
Process_Exit_Gate
HAL_Clear_LCD_Screen
dms
HAL_Write_String_onLCD
HAL_Control_LCD_Lines_And_Cursor
Show_WelcomeScreen
MCAL_GPIO_ReadPin
IS_ID_Validate
Display_Access_Denied
Display_Access_Accepted
Servo2_Exit_Gate

Hardware Circuit:



Conclusion :

-The parking management system was successfully designed and implemented using a three layer architecture The system ensures secure access control real time slot management and user friendly interaction.

-Testing confirmed correct operation of all modules and integration between layers.

-Future improvements can further enhance capacity, automation and remote monitoring.