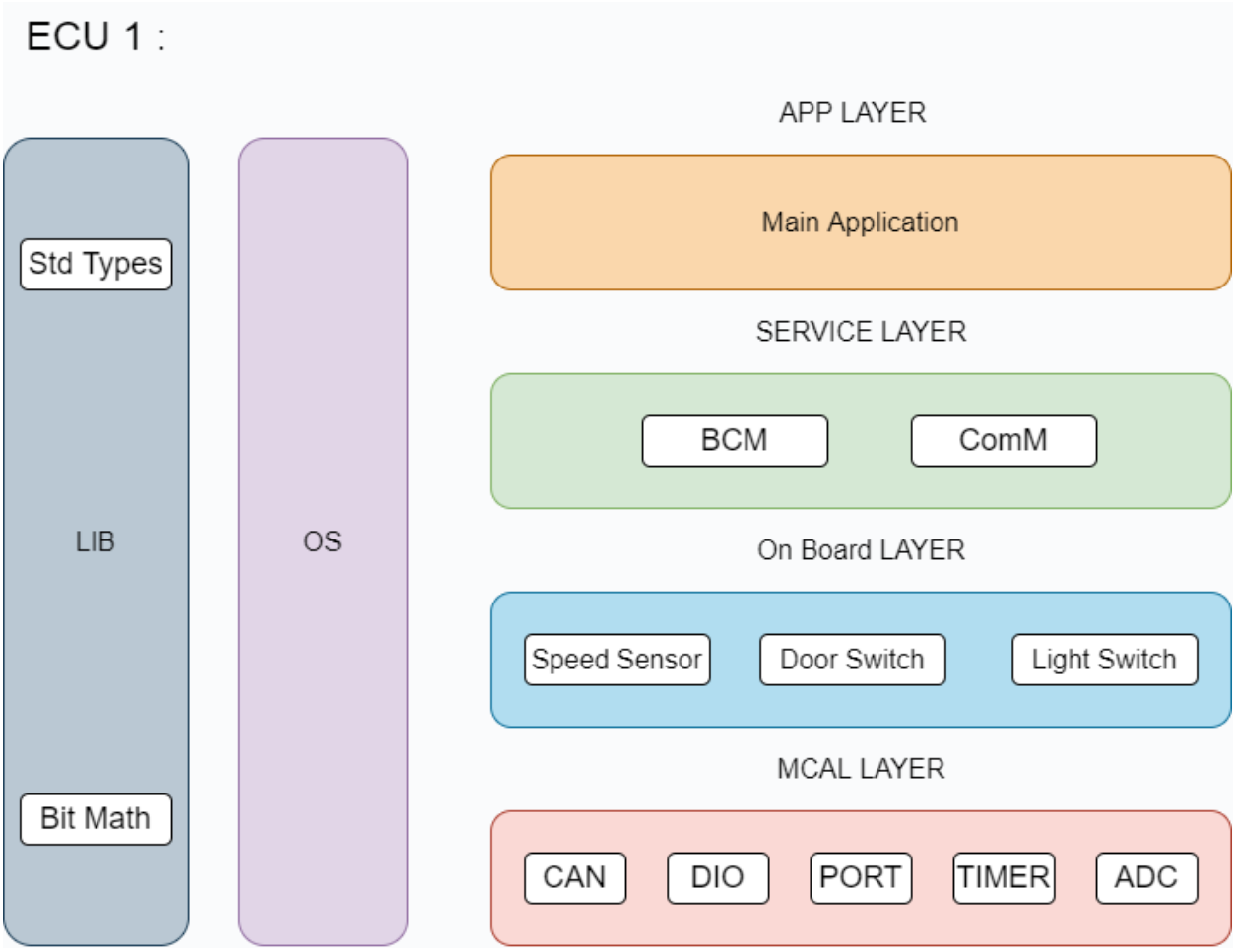
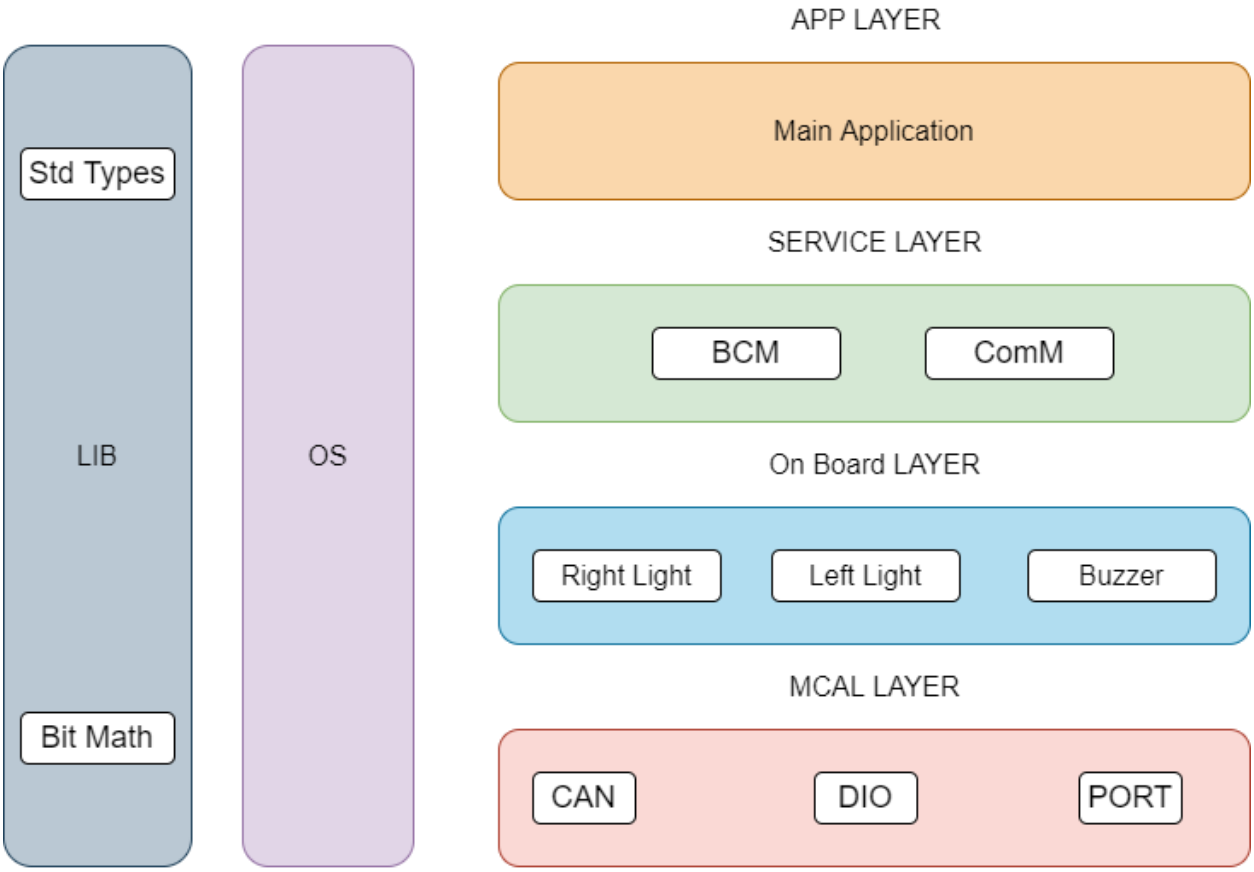
Automotive Door Control System Design

1- The Layered Architecture:



ECU 2 :


APP LAYER
Main Application
SERVICE LAYER
BCM
ComM
On Board LAYER
Right Light
Left Light
Buzzer
MCAL LAYER
CAN
DIO
PORT
Std Types
LIB
Bit Math
OS

2- ECU Components, Modules and Folder Structure:

- **ECU 1 :**

  1- MCAL:

  - **PORT (PORT.c – PORT.h – PORT_CFG.c)**
  - **DIO (DIO.c – DIO.h)**
  - **CAN (CAN.c – CAN.h)**
  - **ADC (ADC.c – ADC.h)**
  - **TIMER (TIMER.c – TIMER.h)**

  2- On Board:

  - **Speed Sensor (SPEED_SENSOR.h – SPEED_SENSOT.c)**
  - **Door Switch (DOOR_SWITCH.h – DOOR_SWITCH.c)**
  - **Light Switch(LIGHT_SWITCH.h – LIGHT_SWITCH.c)**

  3- OS (OS.c – OS.h – OS_CFG.c)

  4- Libraries:

  - **STD Types (STD_TYPES.h)**
  - **Bit Math (BIT_MATH.h)**

  5- Service Layer:

  - **BCM (BCM.c – BCM.h)**

- **ComM (ComM.c – ComM.h)**

- **ECU 2 :**

1- MCAL:

- **PORT (PORT.c – PORT.h – PORT_CFG.c)**
- **DIO (DIO.c – DIO.h)**
- **CAN (CAN.c – CAN.h)**

2- On Board:

- **Right Light (Light.h – Light.c)**
- **Left Light ((Light.h – Light.c)**
- **Buzzer(Buzzer.h - Buzzer.c)**

3- OS (OS.c – OS.h – OS_CFG.c

4- Libraries:

- **STD Types (STD_TYPES.h)**
- **Bit Math (BIT_MATH.h)**

5- Service Layer:

- **BCM (BCM.c – BCM.h)**

3- Full Detailed APIs and Typedefs:

ECU1:

```
/*
 *  Description : Initialize the used Port driver with required configuration
 *  Input       : Array that contains the configuration of ports
 *  Output      : None
 *  Type        : Synchronous - Non Reentrant
 */
void Port_Init( const Port_ConfigType* ConfigPtr );

/*
 *  Description : Set The Direction of an Hardware Pin
 *  Input       : 1- The pin shall to be modified   2- The desired Direction
 *  Output      : None
 *  Type        : Synchronous - Reentrant
 */
void Port_SetPinDirection( Port_PinType Pin,Port_PinDirectionType Direction );

/*
 *  Description : Set The Mode of an Hardware Pin
 *  Input       : 1- The pin shall to be modified   2- The desired Mode
 *  Output      : None
 *  Type        : Synchronous - Reentrant
 */
void Port_SetPinMode( Port_PinType Pin,Port_PinModeType Mode );

/*
 *  Description : Read The State of an Hardware Pin
 *  Input       : The pin shall to be read
 *  Output      : The State of the Pin
 *  Type        : Synchronous - Reentrant
 */
Dio_LevelType Dio_ReadChannel( Dio_ChannelType ChannelId );
```

```c
/*
 *  Description : Set The State of an Hardware Pin
 *  Input       : 1- The pin shall to be Written    2- The Desired State
 *  Output      : None
 *  Type        : Synchronous - Reentrant
 */
void Dio_WriteChannel( Dio_ChannelType ChannelId, Dio_LevelType Level );

/*
 *  Description : Initialize ADC driver
 *  Input       : None
 *  Output      : None
 *  Type        : Synchronous - Non Reentrant
 */
void ADC_Init ( void );

/*
 *  Description : Start ADC conversion
 *  Input       : The number of Channel that to be converted
 *  Output      : None
 *  Type        : Synchronous - Non Reentrant
 */
void ADC_Start ( Channel_ID channel );

/*
 *  Description : Stop ADC conversion
 *  Input       : The number of Channel that to be converted
 *  Output      : None
 *  Type        : Synchronous - Non Reentrant
 */
void ADC_Stop ( Channel_ID channel );

/*
 *  Description : Read the result of ADC conversion
 *  Input       : The number of Channel that converted
 *  Output      : The result of conversion
 *  Type        : Synchronous - Non Reentrant
 */
int ADC_Read (Channel_ID channel );

/*
 *  Description : Initialize Timer driver
 *  Input       : None
 *  Output      : None
 *  Type        : Synchronous - Non Reentrant
 */
void Timer_Init( void );

/*
 *  Description : Start Timer counting
 *  Input       : The number of Channel that to start counting
 *  Output      : None
 *  Type        : Synchronous - Non Reentrant
 */
void Timer_Start ( Timer_Channel channel);

/*
 *  Description : Stop Timer counting
 *  Input       : The number of Channel that to Stop counting
 *  Output      : None
 *  Type        : Synchronous - Non Reentrant
 */
void Timer_Stop ( Timer_Channel channel);
```

```c
/*
 *  Description : Initialize CAN driver
 *  Input       : None
 *  Output      : None
 *  Type        : Synchronous - Non Reentrant
 */
void CAN_Init(void);

/*
 *  Description : Send a byte through CAN bus
 *  Input       : Data to be sent
 *  Output      : None
 *  Type        : Synchronous - Non Reentrant
 */
void CAN_Transmit ( u8 data );

/*
 *  Description : Read the conversion of speed sensor
 *  Input       : none
 *  Output      : the result of speed sensor
 *  Type        : Synchronous - Non Reentrant
 */
u16 Read_Speed_Sensor ( void );

/*
 *  Description : Read the state of light switch
 *  Input       : none
 *  Output      : the state of the switch
 *  Type        : Synchronous - Non Reentrant
 */
Dio_LevelType Read_Light_Switch ( void ) ;

/*
 *  Description : Send a byte through CAN bus
 *  Input       : Data to be sent
 *  Output      : None
 *  Type        : Synchronous - Non Reentrant
 */
void CAN_Transmit ( u8 data );

/*
 *  Description : Read the conversion of speed sensor
 *  Input       : none
 *  Output      : the result of speed sensor
 *  Type        : Synchronous - Non Reentrant
 */
u16 Read_Speed_Sensor ( void );

/*
 *  Description : Read the state of light switch
 *  Input       : none
 *  Output      : the state of the switch
 *  Type        : Synchronous - Non Reentrant
 */
Dio_LevelType Read_Light_Switch ( void ) ;

/*
 *  Description : Read the state of the door switch
 *  Input       : none
 *  Output      : the state of the switch
 *  Type        : Synchronous - Non Reentrant
 */
Dio_LevelType Read_Door_Switch ( void ) ;
```

```c
#define DOOR_SWITCH_PIN         PIN_0
#define LIGHT_SWITCH_PIN        PIN_1
#define SPEED_SENSOR_CH         ADC_CH0

/* enum to describe the state of pins*/
typedef enum {
    PIN_IS_LOW ,
    PIN_IS_HIGH
}Dio_LevelType;

/*enum to describe the number of each pin*/
typedef enum {
    PIN_0 , PIN_1 , PIN_2 , PIN_3 , PIN_4 , PIN_5 , PIN_6 , PIN_7
}Dio_ChannelType;

/* enum to describe ADC channels */
typedef enum {
    ADC_CH0 , ADC_CH1 , ADC_CH2 , ADC_CH3 , ADC_CH4
}Channel_ID;

/* enum to describe TIMER channels */
typedef enum {
    TIMER_0 , TIMER_1 , TIMER_2
}Timer_Channel;
```

ECU2:

```c
#define LEFT_LIGHT_PIN          PIN_0
#define RIGHT_LIGHT_PIN         PIN_1
#define BUZZER_PIN              ADC_CH0

/* enum to describe the state of pins*/
typedef enum {
    PIN_IS_LOW ,
    PIN_IS_HIGH
}Dio_LevelType;

/* enum to describe Control state*/
typedef enum {
    OFF,
    ON
}Control_State;

/* enum to describe Light Switch*/
 typedef enum {
     RIGHT_LIGHT ,
     LEFT_LIGHT
 }Light_Switch;


/*enum to describe the number of each pin*/
typedef enum {
    PIN_0 , PIN_1 , PIN_2 , PIN_3 , PIN_4 , PIN_5 , PIN_6 , PIN_7
}Dio_ChannelType;
```

```c
/*
 *  Description : Set The State of an Hardware Pin
 *  Input       : 1- The pin shall to be Written    2- The Desired State
 *  Output      : None
 *  Type        : Synchronous - Reentrant
 */
void Dio_WriteChannel( Dio_ChannelType ChannelId, Dio_LevelType Level );

/*
 *  Description : Initialize CAN driver
 *  Input       : None
 *  Output      : None
 *  Type        : Synchronous - Non Reentrant
 */
void CAN_Init(void);

/*
 *  Description : Read a byte through CAN bus
 *  Input       : void
 *  Output      : The Data to be read
 *  Type        : Synchronous - Non Reentrant
 */
u8 CAN_Read (void );

/*
 *  Description : Control the state of the buzzer
 *  Input       : the state to be modified
 *  Output      : none
 *  Type        : Synchronous - Non Reentrant
 */
void Buzzer_Control ( Control_State state );

/*
 *  Description : Initialize the used Port driver with required configuration
 *  Input       : Array that contains the configuration of ports
 *  Output      : None
 *  Type        : Synchronous - Non Reentrant
 */
void Port_Init( const Port_ConfigType* ConfigPtr );

/*
 *  Description : Set The Direction of an Hardware Pin
 *  Input       : 1- The pin shall to be modified   2- The desired Direction
 *  Output      : None
 *  Type        : Synchronous - Reentrant
 */
void Port_SetPinDirection( Port_PinType Pin,Port_PinDirectionType Direction );

/*
 *  Description : Set The Mode of an Hardware Pin
 *  Input       : 1- The pin shall to be modified   2- The desired Mode
 *  Output      : None
 *  Type        : Synchronous - Reentrant
 */
void Port_SetPinMode( Port_PinType Pin,Port_PinModeType Mode );

/*
 *  Description : Read The State of an Hardware Pin
 *  Input       : The pin shall to be read
 *  Output      : The State of the Pin
 *  Type        : Synchronous - Reentrant
 */
Dio_LevelType Dio_ReadChannel( Dio_ChannelType ChannelId );
```

```c
/*
 *  Description : Control the state of the buzzer
 *  Input       : the state to be modified
 *  Output      : none
 *  Type        : Synchronous - Non Reentrant
 */
void Buzzer_Control ( Control_State state );

/*
 *  Description : Control the state of the Light
 *  Input       : 1- The light component ( right or left ) 2- the state to be modified
 *  Output      : none
 *  Type        : Synchronous - Non Reentrant
 */
void Light_Control (Light_Switch light_component , Control_State state );
```