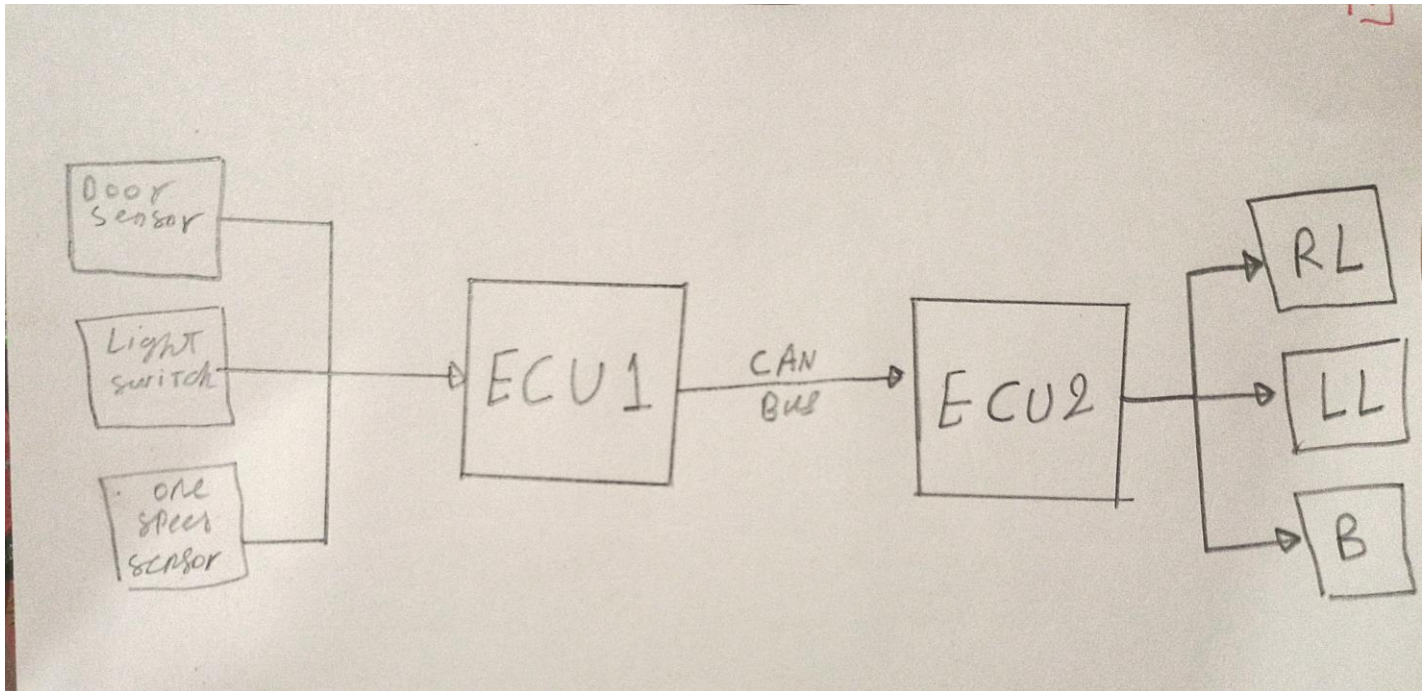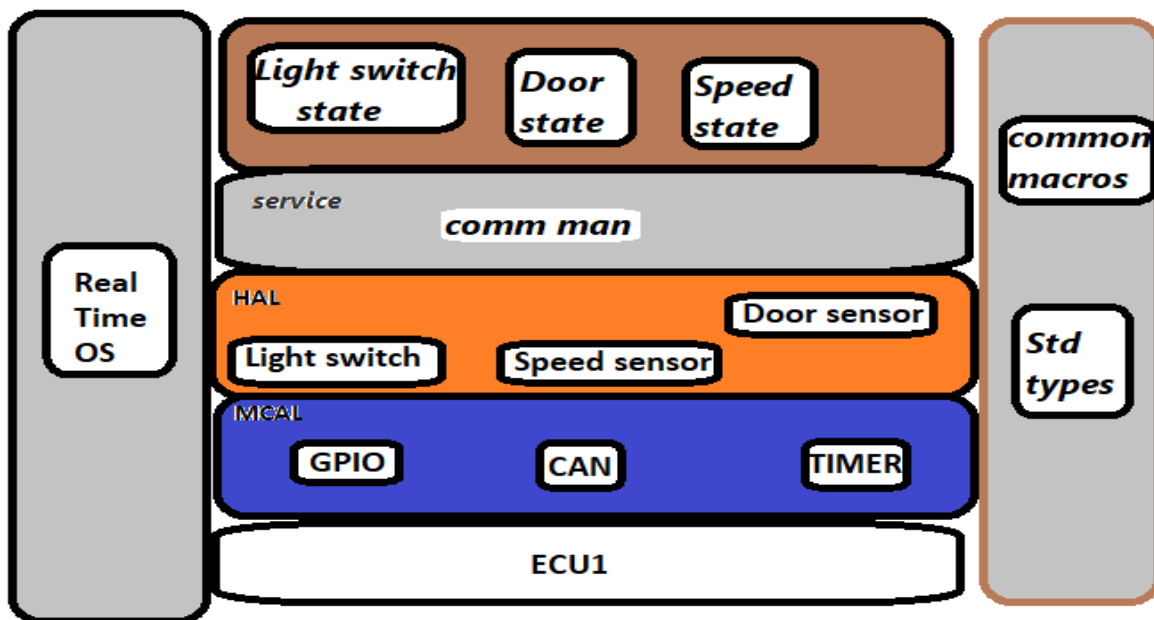# schematic (Block Diagram)

Static Design Analysis

ECU1

1. Make the layered architecture

2. Specify ECU components and modules

A. MCAL layer :

1- GPIO module   2-Timer Module   3- CAN Module

B. HAL layer:

1- Door Module   2-Light Switch Module   3-Speed Sensor

C. Service Layer:

1- comm man (communication manger)

D. App Layer:

1-Door State Module   2-Light Switch state Module   3-Speed Sensor State Module

3.Provide full detailed APIs for each module as well as a detailed description for the used typedefs

**** MCAL layer ****

1- GPIO

➔ init_DIO(Void)

Name	init_DIO
Inputs	---
Return Value	---
Reentrant Function ?	NO
Synchronous Function ?	Yes
Recursion Function ?	NO
Description	DIO initialization

➔ Dio_Write(Dio_port Port , Dio_PinNumber PinNumber , Dio_Level level)

Name	Dio_Write
Inputs	Port number & pin number & pin state
Return Value	---
Reentrant Function ?	Yes
Synchronous Function ?	Yes
Recursion Function ?	NO
Description	Write in pin

➔ Dio_Read (Dio_port Port , Dio_PinNumber PinNumber)

Name	init_Read
Inputs	Port number & pin number
Return Value	Dio_Level
Reentrant Function ?	Yes
Synchronous Function ?	Yes
Recursion Function ?	NO
Description	Read state of pin

detailed description for the used typedefs

*Typedef Enum{ LOW=0, HIGH=1} Dio_Level; // states of pins

* Typedef   char   Dio_PinNumber; // (save pin number in 8 bits variables)

*Typedef Enum{PORTA,PORTB,PORTC,PORTD,PORTE,PORTF} Dio_port;

2- CAN

➔ init_CAN(Void)

Name	init_CAN
Inputs	---
Return Value	---
Reentrant Function ?	NO
Synchronous Function ?	Yes
Recursion Function ?	NO
Description	CAN initialization

➔ CAN_transmit (int CAN_Num, char* Data)

Name	CAN_transmit
Inputs	Which channel & data
Return Value	Std_ReturnType
Reentrant Function ?	Yes
Synchronous Function ?	Yes
Recursion Function ?	NO
Description	Transmit using Can

➔ CAN_ Recieve (int CAN_Num)

Name	CAN_ Recieve
Inputs	Which channel
Return Value	Std_ReturnType
Reentrant Function ?	Yes
Synchronous Function ?	Yes
Recursion Function ?	NO
Description	receive using Can

3- Timer

➔ init_Timer(Void)

Name	init_Timer
Inputs	---
Return Value	---
Reentrant Function ?	NO
Synchronous Function ?	Yes

Recursion Function ?	NO
Description	Timer initialization

➜ **Timer_Start (Timer_Number Timer_Num, Timer_value Time)**

Name	Timer_Start
Inputs	Timer number & Time
Return Value	---
Reentrant Function ?	Yes
Synchronous Function ?	Yes
Recursion Function ?	NO
Description	Make timer count

➜ **Timer_Stop (Timer_Number Timer_Num)**

Name	Timer_Stop
Inputs	Timer number
Return Value	---
Reentrant Function ?	Yes
Synchronous Function ?	Yes
Recursion Function ?	NO
Description	Make timer stop count

➜ **Timer_GetTime (Timer_Number Timer_Num)**

Name	Timer_GetTime
Inputs	Timer number
Return Value	Timer_value
Reentrant Function ?	Yes
Synchronous Function ?	Yes
Recursion Function ?	NO
Description	Get time now

---

detailed description for the used typedefs

Typedef Enum{ T1 = T1PR, T2 = T2PR} Timer_Number;
// choose which timer we will work (timer 1 or timer 2 …)
* Typedef int Timer_Value; // (store the timer value in 32 bits)

**** HAL layer ****

1- Light switch

➔ **Init _Light_Switch (void)**

Name	Init _Light_Switch
Inputs	---
Return Value	---
Reentrant Function ?	NO
Synchronous Function ?	Yes
Recursion Function ?	NO
Description	Light Switch module initialization

➔ **Light_Switch_State (void)**

Name	Light_Switch_State
Inputs	-----
Return Value	Return state (ON or OFF)
Reentrant Function ?	Yes
Synchronous Function ?	Yes
Recursion Function ?	NO
Description	Check state of light switch if open or close

2- DOOR sensor

➔ **Init _DOOR_ sensor (void)**

Name	Init _DOOR_ sensor
Inputs	---
Return Value	---
Reentrant Function ?	NO
Synchronous Function ?	Yes
Recursion Function ?	NO
Description	Door sensor module initialization

➔ **DOOR_ sensor_State (void)**

Name	DOOR_ sensor_State
Inputs	-----
Return Value	Return state (ON or OFF)
Reentrant Function ?	Yes
Synchronous Function ?	Yes
Recursion Function ?	NO
Description	Check state of door if open or close

3- DOOR sensor

➔ *Init _Speed_Sensor (void)*

Name	Init _Speed_Sensor
Inputs	---
Return Value	---
Reentrant Function ?	NO
Synchronous Function ?	Yes
Recursion Function ?	NO
Description	Speed sensor module initialization

➔ *Speed_ sensor_State (void)*

Name	Speed_ sensor_State
Inputs	-----
Return Value	Return state (1 or 0)
Reentrant Function ?	Yes
Synchronous Function ?	Yes
Recursion Function ?	NO
Description	Check if car moving or not

**** Service layer ****

1- Comm manger

➔ *init_Comm_mang (void)*

Name	init_Comm_mang
Inputs	---
Return Value	---
Reentrant Function ?	NO
Synchronous Function ?	Yes
Recursion Function ?	NO
Description	Communication manger module initialization

➔ *Comm_mang_send (char* data)*

Name	Comm_mang_send
Inputs	Pointer to data
Return Value	Std_ReturnType
Reentrant Function ?	Yes
Synchronous Function ?	Yes
Recursion Function ?	NO
Description	Transmat data to can protocol

➜ Comm_mang_Recieve (void)

Name	Comm_mang_Recieve
Inputs	---
Return Value	Std_ReturnType
Reentrant Function ?	Yes
Synchronous Function ?	Yes
Recursion Function ?	NO
Description	Recieve data to can protocol

**** App layer ****

➜ Send_Door_state (char * data)

Name	Send_Door_state
Inputs	Pointer to data
Return Value	Std_ReturnType
Reentrant Function ?	Yes
Synchronous Function ?	Yes
Recursion Function ?	NO
Description	Send door state throw can protocol

➜ Read_Door_state (void)

Name	Read_Door_state
Inputs	Pointer to data
Return Value	State(open or close)
Reentrant Function ?	Yes
Synchronous Function ?	Yes
Recursion Function ?	NO
Description	Read door sensor state

➜ Send_Speed_Sensor_state (char * data)

Name	Send_Speed_Sensor_state
Inputs	Pointer to data
Return Value	Std_ReturnType
Reentrant Function ?	Yes
Synchronous Function ?	Yes
Recursion Function ?	NO
Description	Send car state throw can protocol

➔ *Read_Speed_Sensor_state (void)*

Name	Read_Speed_Sensor_state
Inputs	Pointer to data
Return Value	State (moving or not)
Reentrant Function ?	Yes
Synchronous Function ?	Yes
Recursion Function ?	NO
Description	Read speed sensor state

➔ *Send_Light_Switch_state (char * data)*

| Name	Send_Light_Switch_state
Inputs	Pointer to data
Return Value	Std_ReturnType
Reentrant Function ?	Yes
Synchronous Function ?	Yes
Recursion Function ?	NO
Description	Send light switch state throw can protocol

➔ *Read_Light_Switch_state (void)*

Name	Read_Light_Switch_state
Inputs	Pointer to data
Return Value	State(ON/OFF)
Reentrant Function ?	Yes
Synchronous Function ?	Yes
Recursion Function ?	NO
Description	Read light switch state

4- Prepare your folder structure according to the previous points
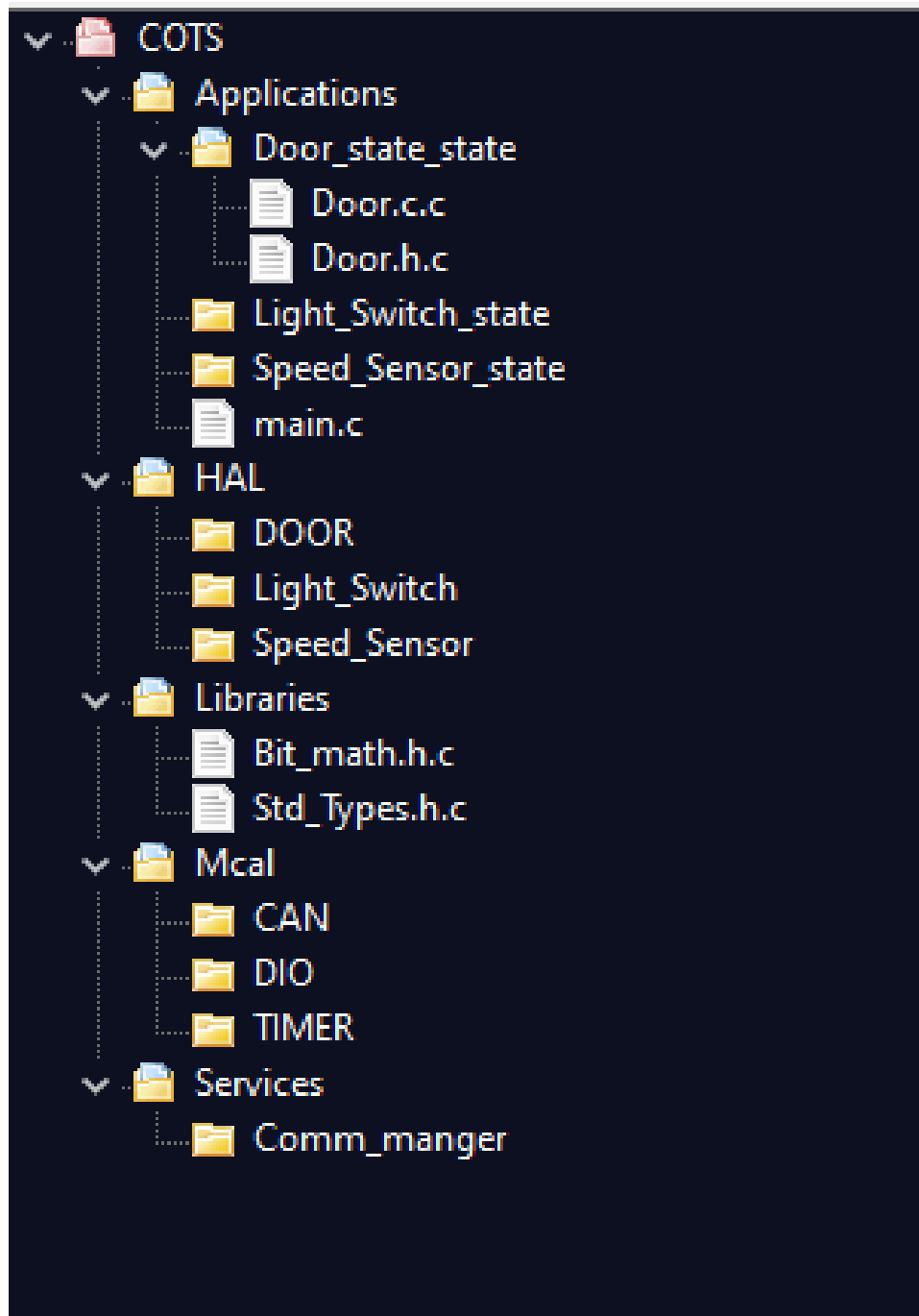
ECU2

1. Make the layered architecture
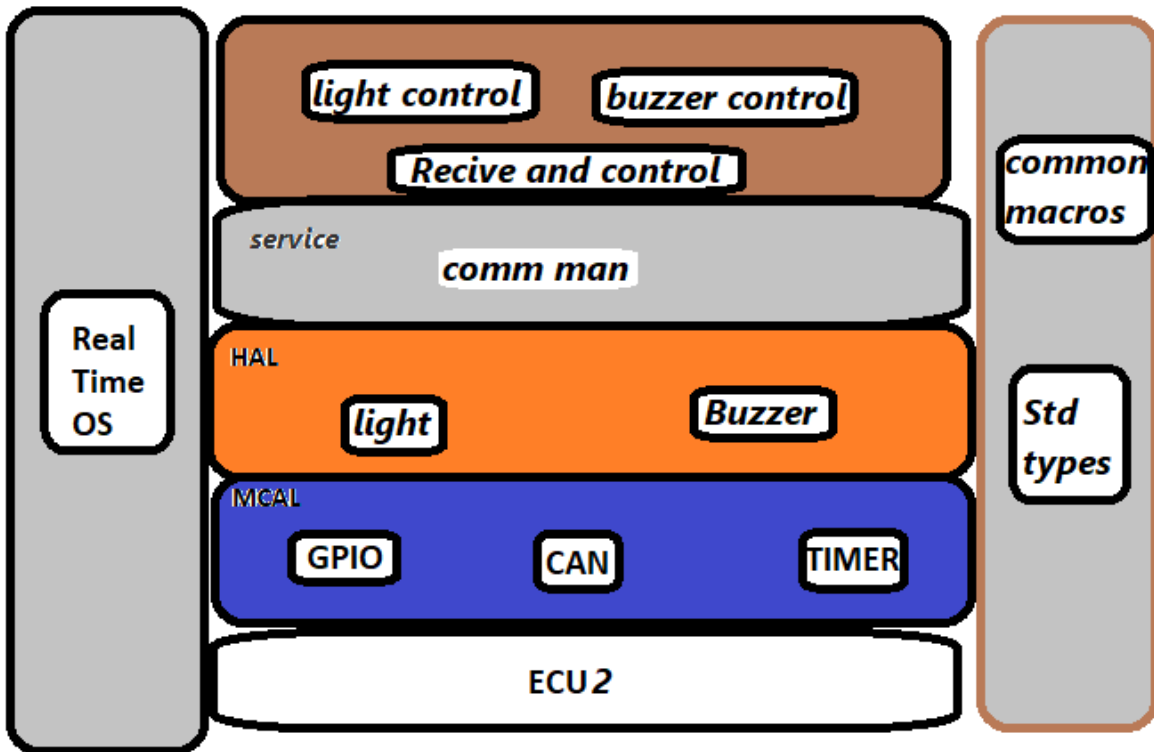
2. Specify ECU2 components and modules

A. MCAL layer :

1- GPIO module   2-Timer Module   3- CAN Module

B. HAL layer:

1- light Module   2- buzzer Module

C. Service Layer:

1- comm man (communication manger)

D. App Layer:

1-light control Module   2-buzzer control Module

3.Provide full detailed APIs for each module as well as a detailed description for the used typedefs

*** MCAL & Service layer in ECU1 and ECU2 are the same**

**** HAL layer ****

1- Light Module

➔ **Init _Light(void)**

Name	Init _Light
Inputs	---
Return Value	---
Reentrant Function ?	NO
Synchronous Function ?	Yes
Recursion Function ?	NO
Description	Light module initialization

➔ **Light_ switch (Light_Dir ID , Light_state mode)**

Name	Light_switch
Inputs	Direction(left or right) & mode(on or off)
Return Value	----
Reentrant Function ?	Yes
Synchronous Function ?	Yes
Recursion Function ?	NO
Description	Control light state

typedef Enum{left,right} Light_Dir; //choose left or right light

typedef Enum{Ligth_on,Light_off} Light_state; // state of the light

2- Buzzer Module

➔ **Init _Buzzer (void)**

Name	Init _Buzzer
Inputs	---
Return Value	---
Reentrant Function ?	NO
Synchronous Function ?	Yes
Recursion Function ?	NO
Description	Buzzer module initialization

➜ Buzzer_switch *(buzzer_state mode)*

Name	Buzzer_switch
Inputs	mode(on or off)
Return Value	----
Reentrant Function ?	Yes
Synchronous Function ?	Yes
Recursion Function ?	NO
Description	Control buzzer state

typedef Enum{buzzer_on,buzzer_off} buzzer_state; // state of the buzzer

**** App layer ****

1. Lights control module

➜ light_control *(Light_Dir ID , Light_state mode)*

Name	light_control
Inputs	mode(on or off)/which led
Return Value	-----
Reentrant Function ?	Yes
Synchronous Function ?	Yes
Recursion Function ?	NO
Description	Control light mode

2. Buzzer control module

➜ Buzzer_control *(buzzer_state mode)*

Name	Buzzer_control
Inputs	mode(on or off)
Return Value	----
Reentrant Function ?	Yes
Synchronous Function ?	Yes
Recursion Function ?	NO
Description	Control buzzer mode

3. Receive and control module

→ **Receive_mode (buzzer_state mode)**

Name	Buzzer_control
Inputs	mode(on or off)
Return Value	----
Reentrant Function ?	Yes
Synchronous Function ?	Yes
Recursion Function ?	NO
Description	Control buzzer mode

// this module receive state and choose what changes will happen in system and send messages for light control and buzzer control

4- Prepare your folder structure according to the previous points