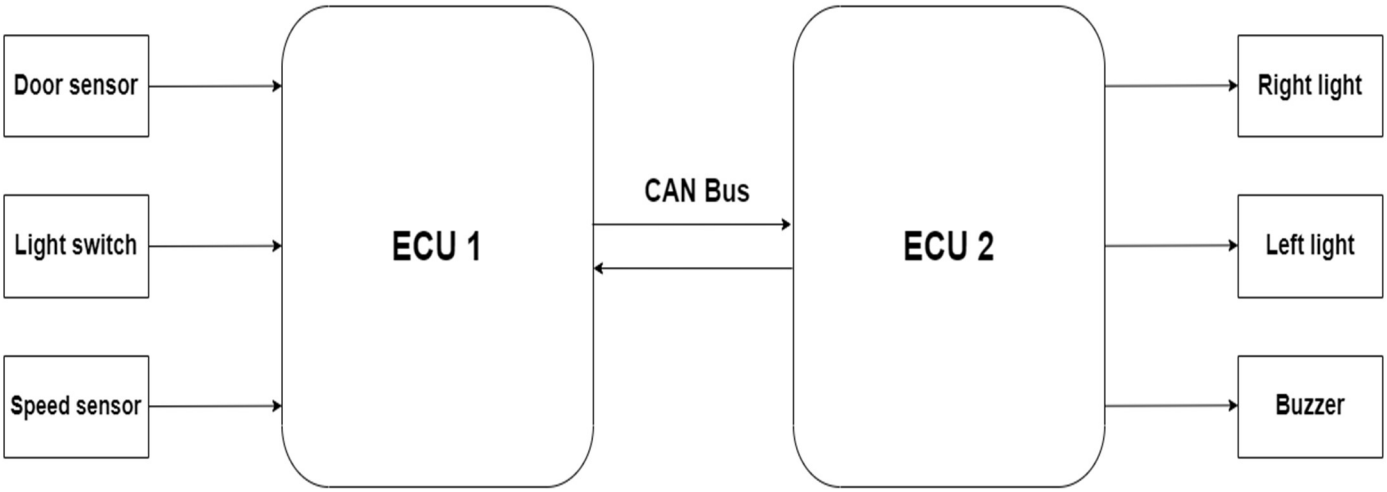
❖ Automotive door control system design

(Static Design)

- **System Blockdiagram :**

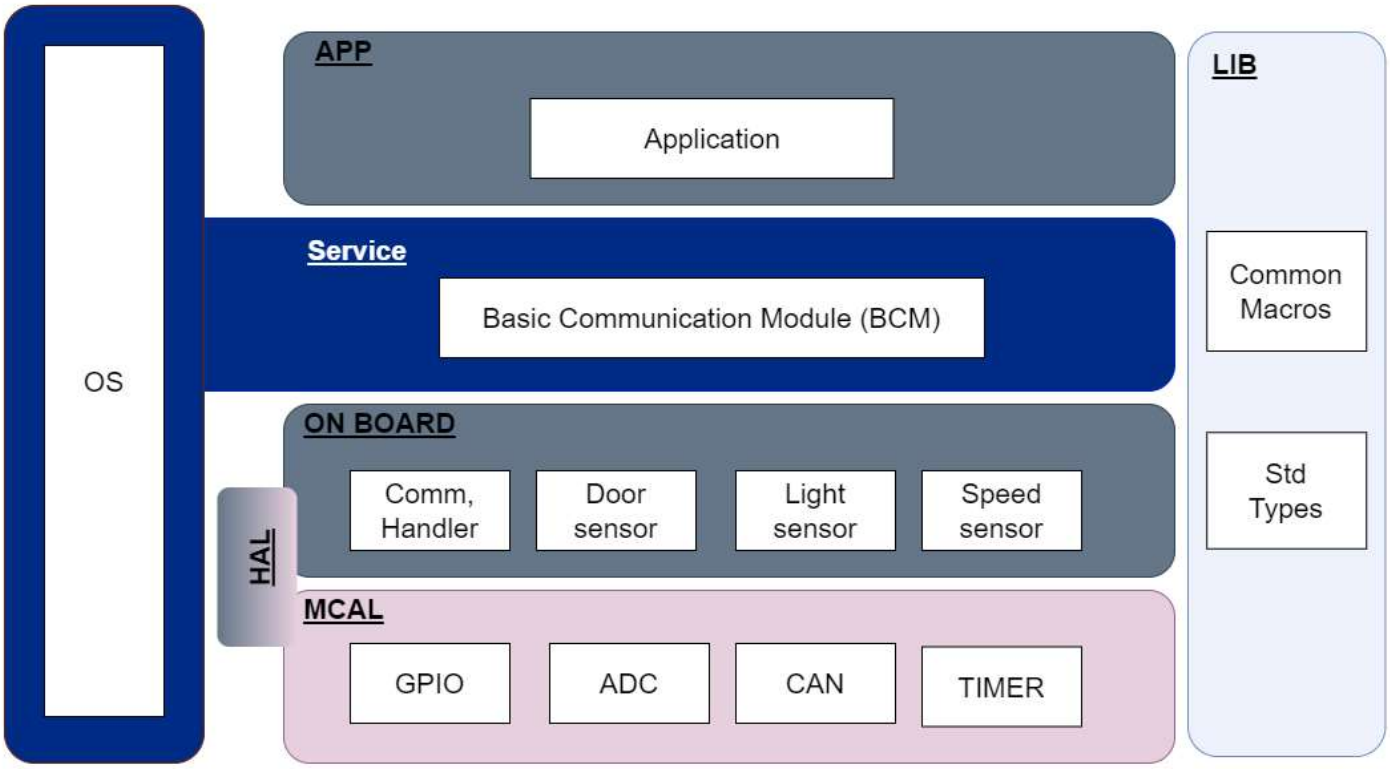
requirements:

1. Two microcontrollers connected via CAN bus
2. One Door sensor (D)
3. One Light switch (L)
4. One Speed sensor (S)
5. ECU 1 connected to D, S, and L, all input devices
6. Two lights, right (RL) and left (LL)
7. One buzzer (B)
8. ECU 2 connected to RL, LL, and B, all output devices



- **Layered Architecture :**

ECU 1 :



ECU 1 Components and Modules :

* Connection with Door/Speed sensor and Light switch ------ > so we need GPIO module "MCAL dirver" .

* Speed sensor ----- > se we need ADC modeule "MCAL driver"

* CAN Communacation ------ > CAN module "MCAL dirver"

* we need module to each on board hardware

--> Door sensor module - Light sensor Module - Speed sensor Module "ON board layer"

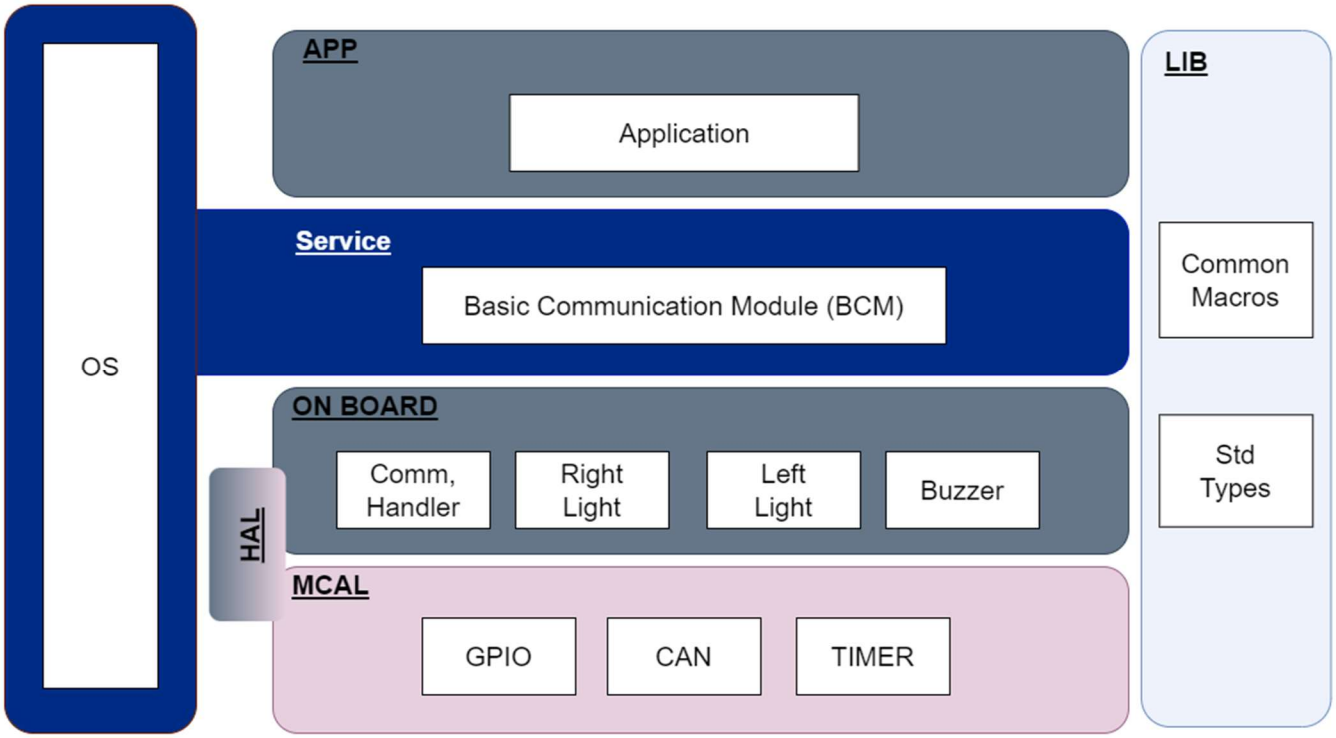
* we need manager (BCM) -----> Basic Communication Module (BCM) "Service layer"

* all layers should be closed so we need a handled between service layer and MCAL layer ----------> Communication handler module "ON BOARD layer"

* OS ----- > RTOS

* peridic transmition ----> Timer module (Which in our case systick as a source for RTOS Ticks) "MCAL driver"

ECU 2 :



ECU 2 Components and Modules :

* Connection with Right/Left Light and Buzzer ------ > so we need GPIO module "MCAL dirver".

* CAN Communacation ------ > CAN module "MCAL dirver"

* We need module to each on board hardware ----> Right Light Module - Left Light Module - Buzzer Module "ON board layer"

* we need manager (BCM) -----> Basic Communication Module (BCM) "Service layer"

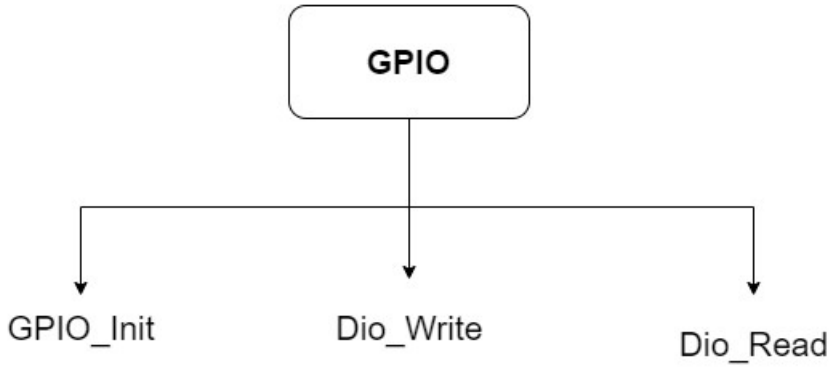
* all layers should be closed so we need a handled between service layer and MCAL layer ----------> Communication handler module "ON BOARD layer"

* OS ----- > RTOS

* Periodic transmition ----> Timer module (Which in our case systick as a source for RTOS Ticks) "MCAL driver" .

APIs for each module & description for the used typedefs :

- **Common Modules/APIS in ECU1&ECU2 :-**



Function name :	**GPIO_Init**	
Arguments	**Inputs**	**N/A**
	Outputs	**N/A**
Return	**E_OK**	**0**
	E_NOK	**1**
Description :	This function Responsible for Initialize GPIO pins (pinmode , pindirection , Internal attach , alt func, .. etc)	

Function name :	**DIO_Read**		
Arguments	**Inputs**	PortID	PORT_t
		Description : Port Number	
		Pin_Num	PIN_t
		Description : Pin Number	
Return	GPIO_LOW	0	
	GPIO_HIGH	1	
Description :	This function used to get pin State (High/Low)		

Function name :	DIO_Write	
Arguments	Inputs	PortID : PORT_t Description : Port Number Pin_Num : PIN_t Description : Pin Number Value : Level_t Description : Pin State
Return	Void	

Description : This function used to set pin High/Low

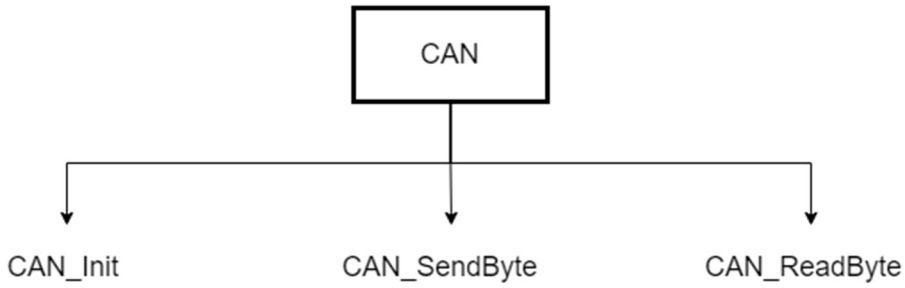
Name :	Level_t		
Type:	Enumeration		
Range :	GPIO_LOW	0	Description : Low state
	GPIO_PIN_1	1	Description : High state

Description : Enumeration for representing DIO State .

Name :	PIN_t		
Type:	Enumeration		
Range :	GPIO_PIN_0	0	Description : select Pin 0
	GPIO_PIN_1	1	Description : select Pin 1
	GPIO_PIN_1	2	Description : select Pin 2
	GPIO_PIN_2	3	Description : select Pin 3
	GPIO_PIN_4	4	Description : select Pin 4
	GPIO_PIN_5	5	Description : select Pin 5
	GPIO_PIN_6	6	Description : select Pin 6
	GPIO_PIN_7	7	Description : select Pin 7

Description : Enumeration for representing ECU port's pins .

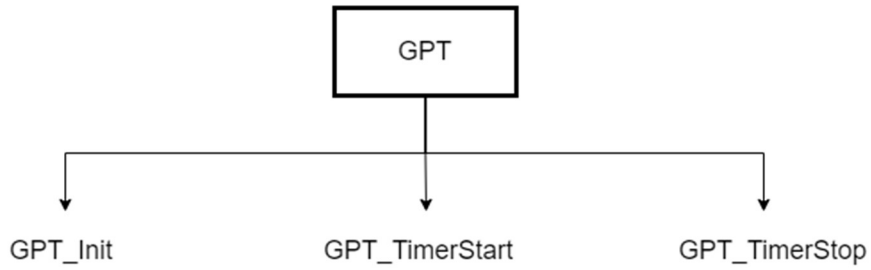
Name :	PORT_t		
Type:	Enumeration		
Range :	GPIO_PORTA	0	Description : select Port A
	GPIO_PORTB	1	Description : select Port B
	GPIO_PORTC	2	Description : select Port C
	GPIO_PORTD	3	Description : select Port D
Description :	Enumeration for representing ECU ports .		



Function name :	CAN_Init	
Arguments	Inputs	N/A
Return	E_OK	0
	E_NOK	1
Description :	This function responsible for initializing CAN communication hardware.	

Function name :	CAN_SendByte	
Arguments	Inputs	Date u8 Description: Byte to be sent
Return	E_OK	0
	E_NOK	1
Description :	This function used to send a byte over can bus .	

Function name :	CAN_ReadByte	
Arguments	Inputs	N/A
Return	u8	
Description :	This function used to read byte over can bus (polling) .	



Function name :	**GPT_Init**	
Arguments	**Inputs**	copy_Config_Ptr | GPT_Config_t* Description : Pointer to GPT_Config_t(struct) which holds selected channel .
Return	E_OK	0
	E_NOK	1
Description :	This function used to initialize selected channel/Timer hardware.	

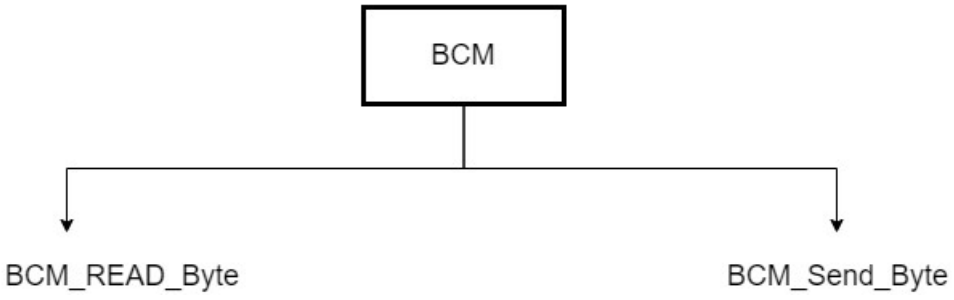
Function name :	**GPT_StartTimer**	
Arguments	**Inputs**	Channel | GPT_Channel_t Description : Selected channel Value | uint32 Description : Starting value
Return	E_OK	0
	E_NOK	1
Description :	This function used to Start selected channel/Timer with starting value.	

Function name :	GPT_StopTimer	
Arguments	Inputs	Channel GPT_Channel_t Description : Selected channel
Return	E_OK	0
	E_NOK	1
Description :	This function used to Stop selected channel/Timer .	

Name :	GPT_Config_t		
Type:	Structure		
Elements:	Channel_id Channel_Mode Channel_Dir_t void(* Call_Back)(void)	GPT_Channel_t GPT_Mode_t GPT_Dir_t	Desc,: select channel Desc,:select Mode Desc,:select Dir, Up/Down counting. Desc,: Call back function to be called every ISR
Description :	Structure to be passed to GPT_Init to initialize specific channel .		

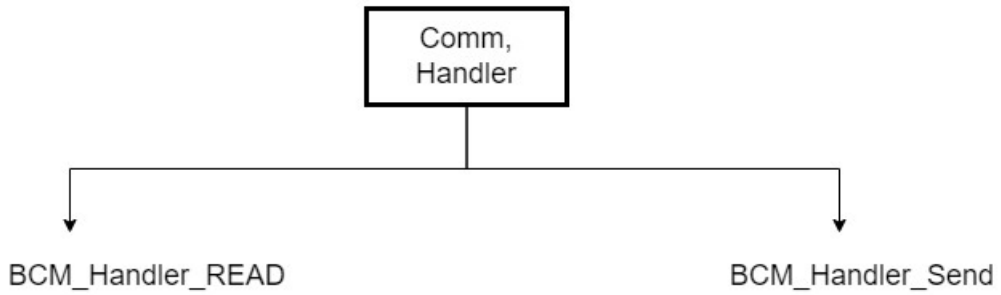
Name :	GPT_Channel_t		
Type:	Enumeration		
Range :	GPT_TIM0 GPT_TIM1 GPT_TIM2	0 1 2	Description : Select TIM 0 Description : Select TIM 1 Description : Select TIM 2
Description :	Enumeration for representing Timers .		

Name :	GPT_Mode_t		
Type:	Enumeration		
Range :	One_Shot_Tim_mode Periodic_Tim_mode	0 1	Description : only one shot Description : Periodic
Description :	Enumeration for GPT Modes .		



Function name :	BCM_Read_Byte	
Arguments	Inputs	N/A
Return	uint8	
Description :	This function Read status message from CAN module .	

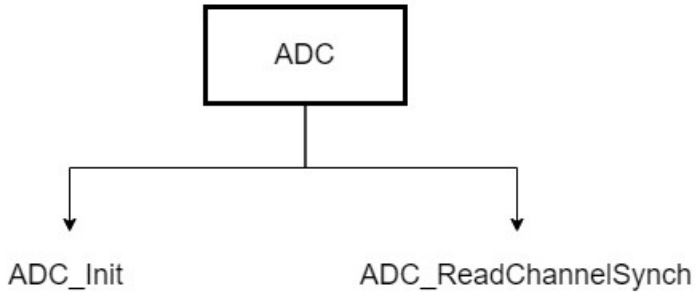
Function name :	**BCM_Send_Byte**	
Arguments	**Inputs**	Data : uint8 Description : Byte to be send
Return	E_OK	0
	E_NOK	1
Description :	This function send status message .	



Function name :	**BCM_Handler_Read**	
Arguments	**Inputs**	N/A
Return	uint8	
Description :	This function Read message from CAN module (works as interface between Service/MCAL layers to avoid open layer) .	

Function name :	**BCM_Handler_Send**	
Arguments	**Inputs**	Data uint8 Description : Byte to be send
Return	E_OK	0
	E_NOK	1
Description :	This function send status message to CAN module (works as interface between Service/MCAL layers to avoid open layer) .	

○ Modules/APIS in ECU1 ONLY :-



Function name :	**ADC_Init**	
Arguments	**Inputs**	N/A
Return	E_OK	0
	E_NOK	1
Description :	This function used to initialize ADC. (mode.Volt_ref,Prescaler,Resolution , ..)	

Function name :	**ADC_ReadChannelSynch**	
Arguments	**Inputs**	Channel | Channel_t Description: Channel Number
Return	u16	
Description :	This function used to get value of selected ADC channel	

Name :	Channel_t		
Type:	Enumeration		
Range :	ADC_0	0	Description : Select ch 0
	ADC_1	1	Description : Select ch 1
	ADC_2	2	Description : Select ch 2
	ADC_3	3	Description : Select ch 3
	ADC_4	4	Description : Select ch 4
	ADC_5	5	Description : Select ch 5
	ADC_6	6	Description : Select ch 6
	ADC_7	7	Description : Select ch 7
Description :	Enumeration for representing ADC Channels.		

Name :	Mode_t		
Type:	Enumeration		
Range :	SINGLE_CONVERSION	0	Description :only single conv,
	CONTINOUS_CONVERSION	1	Description : Select ch 1
Description :	Enumeration for representing ADC Modes.		

Name :	ADC_PRE_t		
Type:	Enumeration		
Range :	ACD_2_PRE	1	Description : prescaler 2
	ACD_4_PRE	2	Description : prescaler 4
	ACD_8_PRE	3	Description : prescaler 8
	.....		
	ACD_64_PRE	6	Description : prescaler 64
	ACD_128_PRE	7	Description : prescaler 128
Description :	Enumeration for representing ADC Prescalers..		

Name :	ADC_VOLT_REF		
Type:	Enumeration		
Range :	ADC_AREF ADC_AVCC ADC_INTERNA	0 1 2	Description : Select AREF Description : Select AVCC Description : Select provided internal voltage .
Description :	Enumeration for representing ADC reference voltages.		

Name :	ADC_RESOLUTION		
Type:	Enumeration		
Range :	ADC_10Bit ADC_8Bit	0 1	Description : 10 bit mode Description : 8 bit mode
Description :	Enumeration for representing ADC Resoultions..		

ECU1 Sensors APIS :-

Function name	**Door_Get_State**	
Arguments	**Inputs**	N/A
Return	LOW	0 "Closed"
	HIGH	1 "Opened"
Description :	This function used to get door state (Opened/Closed) .	

Function name	**Light_SW_Get_State**	
Arguments	**Inputs**	N/A
Return	LOW	0 "Not Pressed"
	HIGH	1 "Pressed"
Description :	This function used to get light switch state (Pressed/Not) .	

Function name	Speed_Get_Value	
Arguments	Inputs	N/A
Return	uint16	
Description : This function used to get speed value from speed sensor .		

- **ECU2 Sensors APIS :-**

Function name	Left_Light_Set_State	
Arguments	Inputs	Light_State bool Description : LOW ---> turning off HIGH -->Turning on
Return	N/A	
Description : This function used to turn the left light ON/OFF .		

Function name	**Right_Light_Set_State**		
Arguments	**Inputs**	Light_State	bool
		Description : LOW ---> turning off HIGH -->Turning on	
Return	N/A		
Description :	This function used to turn the right light ON/OFF .		

Function name	**Buzzer_Set_State**		
Arguments	**Inputs**	state	bool
		Description : LOW ---> Turn OFF HIGH---> Turn ON	
Return	N/A		
Description :	This function used to turn buzzer ON/OFF .		