



Automotive door control system design

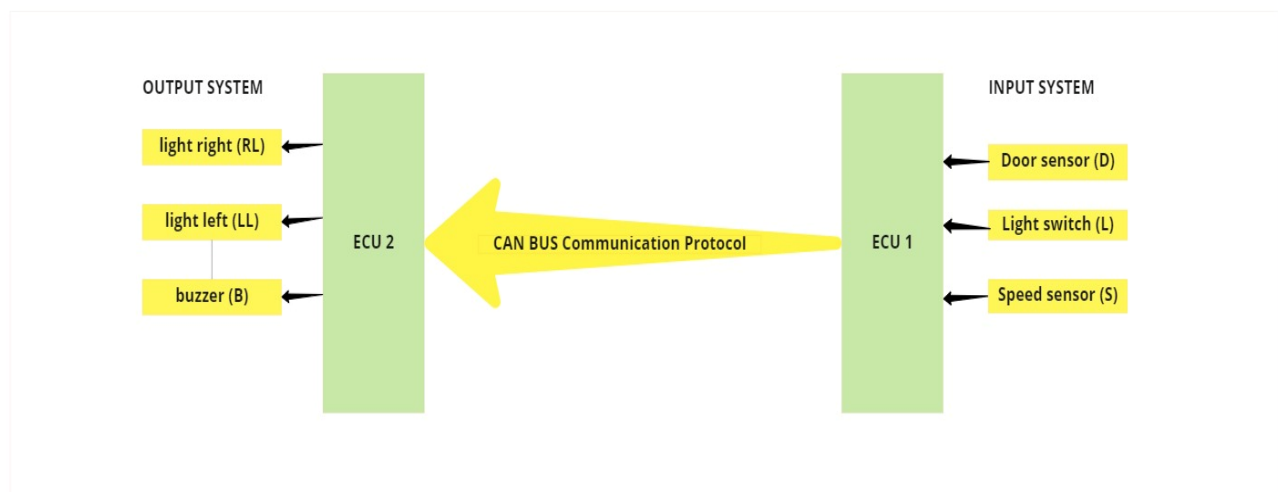
Static design Report

Name: Ahmed Mohamed Hussein Elshehry

Email : elshehry97@gmail.com

system schematic (Block Diagram) according to your requirements understanding.

system schematic (Block Diagram)

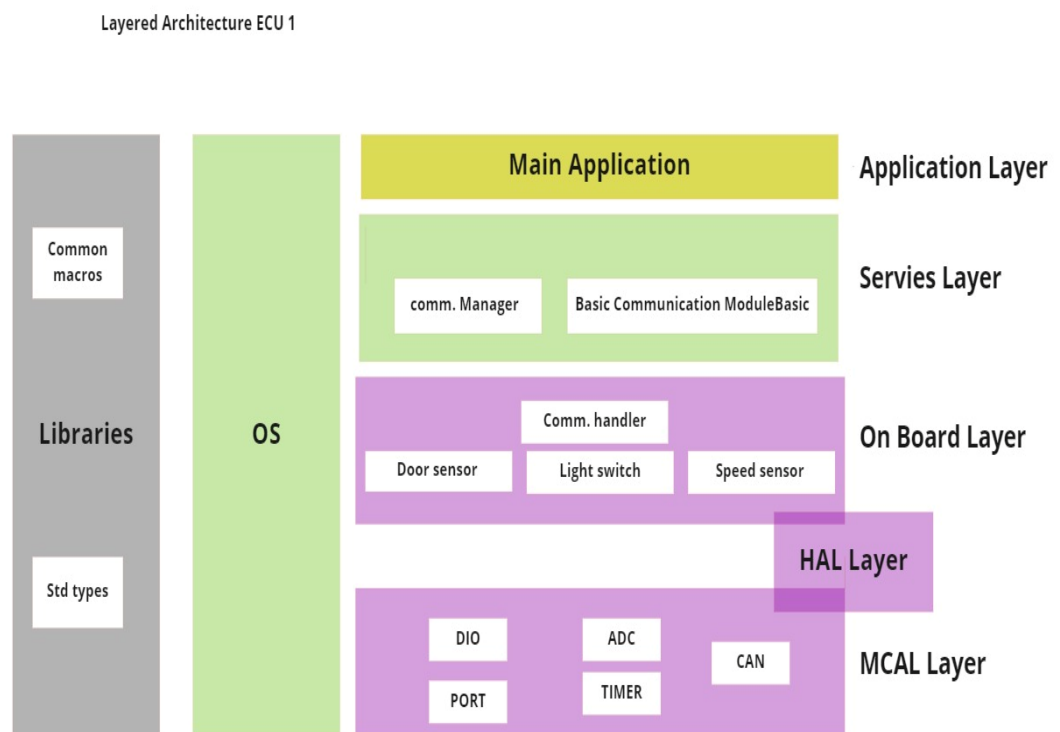


My Work Ahmed Elshehry

Static Design:

➤ For ECU 1:

1- the layered architecture:



My Work Ahmed Elskehry

2- Specify ECU components and modules

Components connected:

- 1. CAN BUS Communication Protocol (for communication between the two ECUs)
- 2. Light switch
- 3. Speed Sensor
- 4. Door Sensor

Modules:

External hardware:

- 1. CAN transiver module
- 2. Switch module
- 3. Speed Sensor module
- 4. Door Sensor module

Internal hardware:

- 1. Port Module (initialize all pins required with modes)
- 2. DIO Module (switch module, Door Sensor module)
- 3. TIMER module (timer for application)
- 4. ADC module (for speed sensor)
- 5. CAN Module (for can transiver data)

3- Provide full detailed APIs for each module as well as a detailed description

Layer	Module	APIs	API Details	
Application Layer	Main Application	DoorSensorTask		
			Syntax:	void DoorSensorTask(void);
			Sync/Async:	Synchronous
			Reentrancy:	Non-Reentrant
			Parameters:	None
			Return:	None
			Description:	Manage Door Sensor Task

Layer	Module	APIs	API Details	
Application Layer	Main Application	LightSwitchTask		
			Syntax:	void LightSwitchTask(void);
			Sync/Async:	Synchronous
			Reentrancy:	Non-Reentrant
			Parameters:	None
			Return:	None
			Description:	Manage Light Switch Task
		SpeedSensorTask		
			Syntax:	void SpeedSensorTask(void);
			Sync/Async:	Synchronous
			Reentrancy:	Non-Reentrant
			Parameters:	None
			Return:	None
			Description:	Manage Speed Sensor Task
Servies Layer	Basic Communication ModuleBasic (BCM Manager)	BCM_Manager		
			Syntax:	void BCM_Manager (uint8_t Id_Bus, uint64_t Data);
			Sync/Async:	Synchronous
			Reentrancy:	Non-Reentrant
			Parameters:	Data transmitter , Id Bus selection
			Return:	None
			Description:	Manage request the data Transmitter by CAN Bus W.R.T Id Bus selection
Servies Layer	comm. Manager	Sensor_Manager (do Monitoring Sensors)		
			Syntax:	uint8_t Sensor_Manager (uint8_t Id_Sensor);
			Sync/Async:	Synchronous
			Reentrancy:	Non-Reentrant
			Parameters:	Sensor selection want read states
			Return:	Date Read from sensor
			Description:	Manage request read states of data from sensor selection
On Board Layer	Comm. Handler	BCM_Handler	Syntax:	void BCM_Handler (uint8_t Id_Bus, uint64_t Data);
			Sync/Async:	Synchronous
			Reentrancy:	Non-Reentrant
			Parameters:	Data transmitter , Id Bus selection
			Return:	None
			Description:	Handler request the data Transmitter by CAN BUS but deals with Hardware directly

		Sensor_Handler	<table><tr><td>Syntax:</td><td>void Sensor_Handler (uint8_t Id_Sensor);</td></tr><tr><td>Sync/Async:</td><td>Synchronous</td></tr><tr><td>Reentrancy:</td><td>Non-Reentrant</td></tr><tr><td>Parameters:</td><td>Sensor selection want read states</td></tr><tr><td>Return:</td><td>None</td></tr><tr><td>Description:</td><td>Handler request read states of data from sensor selection but deals with Hardware directly</td></tr></table>	Syntax:	void Sensor_Handler (uint8_t Id_Sensor);	Sync/Async:	Synchronous	Reentrancy:	Non-Reentrant	Parameters:	Sensor selection want read states	Return:	None	Description:	Handler request read states of data from sensor selection but deals with Hardware directly												
Syntax:	void Sensor_Handler (uint8_t Id_Sensor);																										
Sync/Async:	Synchronous																										
Reentrancy:	Non-Reentrant																										
Parameters:	Sensor selection want read states																										
Return:	None																										
Description:	Handler request read states of data from sensor selection but deals with Hardware directly																										
On Board Layer	Door Sensor	<div>DoorSensor_Init</div> <div>DoorSensor_ReadStatus</div>	<table><tr><td>Syntax:</td><td>void DoorSensor_Init (void);</td></tr><tr><td>Sync/Async:</td><td>Synchronous</td></tr><tr><td>Reentrancy:</td><td>Non-Reentrant</td></tr><tr><td>Parameters:</td><td>None</td></tr><tr><td>Return:</td><td>None</td></tr><tr><td>Description:</td><td>Initialize the used DIO pins for digital input</td></tr></table> <table><tr><td>Syntax:</td><td>Status_door DoorSensor_ReadStatus (void);</td></tr><tr><td>Sync/Async:</td><td>Synchronous</td></tr><tr><td>Reentrancy:</td><td>Non-Reentrant</td></tr><tr><td>Parameters:</td><td>None</td></tr><tr><td>Return:</td><td>Status of the sensor door</td></tr><tr><td>Description:</td><td>Get the status of the sensor door (closed or open)</td></tr></table>	Syntax:	void DoorSensor_Init (void);	Sync/Async:	Synchronous	Reentrancy:	Non-Reentrant	Parameters:	None	Return:	None	Description:	Initialize the used DIO pins for digital input	Syntax:	Status_door DoorSensor_ReadStatus (void);	Sync/Async:	Synchronous	Reentrancy:	Non-Reentrant	Parameters:	None	Return:	Status of the sensor door	Description:	Get the status of the sensor door (closed or open)
Syntax:	void DoorSensor_Init (void);																										
Sync/Async:	Synchronous																										
Reentrancy:	Non-Reentrant																										
Parameters:	None																										
Return:	None																										
Description:	Initialize the used DIO pins for digital input																										
Syntax:	Status_door DoorSensor_ReadStatus (void);																										
Sync/Async:	Synchronous																										
Reentrancy:	Non-Reentrant																										
Parameters:	None																										
Return:	Status of the sensor door																										
Description:	Get the status of the sensor door (closed or open)																										
On Board Layer	Light Switch	<div>LightSwitch_Init</div> <div>LightSwitch_ReadStatus</div>	<table><tr><td>Syntax:</td><td>Void LightSwitch_Init (void);</td></tr><tr><td>Sync/Async:</td><td>Synchronous</td></tr><tr><td>Reentrancy:</td><td>Non-Reentrant</td></tr><tr><td>Parameters:</td><td>None</td></tr><tr><td>Return:</td><td>None</td></tr><tr><td>Description:</td><td>Initialize the used DIO pins for digital input</td></tr></table> <table><tr><td>Syntax:</td><td>Switch_status LightSwitch_ReadStatus (void);</td></tr><tr><td>Sync/Async:</td><td>Synchronous</td></tr><tr><td>Reentrancy:</td><td>Non-Reentrant</td></tr><tr><td>Parameters:</td><td>None</td></tr><tr><td>Return:</td><td>Status of the sensor door</td></tr><tr><td>Description:</td><td>Get the status of the Light Switch (Pressed or undressed)</td></tr></table>	Syntax:	Void LightSwitch_Init (void);	Sync/Async:	Synchronous	Reentrancy:	Non-Reentrant	Parameters:	None	Return:	None	Description:	Initialize the used DIO pins for digital input	Syntax:	Switch_status LightSwitch_ReadStatus (void);	Sync/Async:	Synchronous	Reentrancy:	Non-Reentrant	Parameters:	None	Return:	Status of the sensor door	Description:	Get the status of the Light Switch (Pressed or undressed)
Syntax:	Void LightSwitch_Init (void);																										
Sync/Async:	Synchronous																										
Reentrancy:	Non-Reentrant																										
Parameters:	None																										
Return:	None																										
Description:	Initialize the used DIO pins for digital input																										
Syntax:	Switch_status LightSwitch_ReadStatus (void);																										
Sync/Async:	Synchronous																										
Reentrancy:	Non-Reentrant																										
Parameters:	None																										
Return:	Status of the sensor door																										
Description:	Get the status of the Light Switch (Pressed or undressed)																										

On Board Layer	Speed Sensor	SpeedSensor_Init	
			Syntax: void SpeedSensor_Init (void);
			Sync/Async: Synchronous
			Reentrancy: Non-Reentrant
			Parameters: None
			Return: None
			Description: Initialize the used DIO pins for analog input For (ADC)
		SpeedSensor_ReadStatus	
			Syntax: Speed_status SpeedSensor_ReadStatus (void);
			Sync/Async: Synchronous
			Reentrancy: Non-Reentrant
			Parameters: None
			Return: Status of the sensor door
			Description: Read the value of the speed sensor (moving or stop)

MCAL Layer	DIO	DIO_Init	
			Syntax: Void DIO_Init (void);
			Sync/Async: Synchronous
			Reentrancy: Non-Reentrant
			Parameters: None
			Return: None
			Description: Initialize the used DIO pins with required configuration
		DIO_ReadChannel	
			Syntax: Dio_LevelTypeDIO_ReadChannel (Id_channelType Id_channel);
			Sync/Async: Synchronous
			Reentrancy: Non-Reentrant
			Parameters: Id channel want read
			Return: Status of pin High or low
			Description: Read the channel required
		DIO_WriteChannel	
			Syntax: void DIO_WriteChannel (Dio_LevelType Level);
			Sync/Async: Synchronous
			Reentrancy: Non-Reentrant
			Parameters: Level want to write channel
			Return: None
			Description: Write the level of the channel required
MCAL Layer	PORT	PORT_init	
			Syntax: void RORT_Init (void);
			Sync/Async: Synchronous
			Reentrancy: Non-Reentrant
			Parameters: None
			Return: None
			Description: Initialize the used Port with required configuration

MCAL Layer	Timer	Timer_Init	
			Syntax: void Timer_Init (void);
			Sync/Async: Synchronous
			Reentrancy: Non-Reentrant
			Parameters: None
			Return: None
			Description: Initialize timer required configuration
		Timer_Start	
			Syntax: void Timer_Start (uint8_t channel_Id, timer_ValueType value count);
			Sync/Async: Synchronous
			Reentrancy: Non-Reentrant
			Parameters: channel_Id / value count tick
			Return: None
			Description: Initialize timer required configuration
		Timer_Stop	
			Syntax: Void Timer_Stop (uint8_t channel_Id);
			Sync/Async: Synchronous
			Reentrancy: Non-Reentrant
			Parameters: Channel_Id of timer
			Return: None
			Description: Initialize timer required configuration
MCAL Layer	CAN	CAN_Init	Syntax: void CAN_Init (void);
			Sync/Async: Synchronous
			Reentrancy: Non-Reentrant
			Parameters: None
			Return: None
			Description: Initialize CAN bus required configuration and Hardware pin CAN
		CAN_Transmitter	
			Syntax: void CAN_Transmitter (uint8_t Pin_Id,uint64_t Data);
			Sync/Async: Synchronous
			Reentrancy: Non-Reentrant
			Parameters: Data transmitter , Pin_id
			Return: None
			Description: Transmitter data by CAN Bus

MCAL Layer	ADC	ADC_Init	<table><tr><td>Syntax:</td><td>void ADC_Init (void);</td></tr><tr><td>Sync/Async:</td><td>Synchronous</td></tr><tr><td>Reentrancy:</td><td>Non-Reentrant</td></tr><tr><td>Parameters:</td><td>None</td></tr><tr><td>Return:</td><td>None</td></tr><tr><td>Description:</td><td>Initialize ADC required configuration and Hardware pin ADC connect speed sensor</td></tr></table>	Syntax:	void ADC_Init (void);	Sync/Async:	Synchronous	Reentrancy:	Non-Reentrant	Parameters:	None	Return:	None	Description:	Initialize ADC required configuration and Hardware pin ADC connect speed sensor
			Syntax:	void ADC_Init (void);											
			Sync/Async:	Synchronous											
			Reentrancy:	Non-Reentrant											
			Parameters:	None											
			Return:	None											
			Description:	Initialize ADC required configuration and Hardware pin ADC connect speed sensor											
		ADC_ReadChannel	<table><tr><td>Syntax:</td><td>uint16_tADC_ReadChannel(uint8_tPin_Id);</td></tr><tr><td>Sync/Async:</td><td>Synchronous</td></tr><tr><td>Reentrancy:</td><td>Non-Reentrant</td></tr><tr><td>Parameters:</td><td>Pin_Id of ADC</td></tr><tr><td>Return:</td><td>The value of channel ADC</td></tr><tr><td>Description:</td><td>Read the value of channel ADC</td></tr></table>	Syntax:	uint16_tADC_ReadChannel(uint8_tPin_Id);	Sync/Async:	Synchronous	Reentrancy:	Non-Reentrant	Parameters:	Pin_Id of ADC	Return:	The value of channel ADC	Description:	Read the value of channel ADC
			Syntax:	uint16_tADC_ReadChannel(uint8_tPin_Id);											
			Sync/Async:	Synchronous											
			Reentrancy:	Non-Reentrant											
			Parameters:	Pin_Id of ADC											
			Return:	The value of channel ADC											
			Description:	Read the value of channel ADC											

Types define of argument some APIs :

Types	Define
typedef unsigned char uint8_t	Used in argument Id_Bus to select bus connect
typedef unsigned long long uint64_t	used because max width of data in CAN frame is 64 bits and used in argument Data transmitter API BCM_manager and Handler
Status_door	typedef enum{closed, opened} Status_door range{0,1} size 1bit
Status_switch	typedef enum{undressed ,pressed} Status_switch range{0,1} size 1bit
Status_speed	typedef enum{stopped, moving} Status_speed range{0,1} size 1bit after convert value adc
typedef uint32_t T timer_ValueType;	Value of tick range from 0 to $2^{32}-1$ size 32 bit
Typedef enum{T1 = T1PR,T2 = T2PR,Etc:} timer_ChannelType;	This enum types stores the identifier for the Channel like its name.
Dio_LevelType	typedef enum{LOW, HIGH} Dio_LevelType range{0,1} size 1bit
Id_channelType	typedef enum {Channel_1, Channel_2, Channel_3, Channel_4, Channel_5, Channel_6, Channel_7, Channel_8}Dio_LevelType range{0,8} size 1bit

4- folder structure according to the previous points:

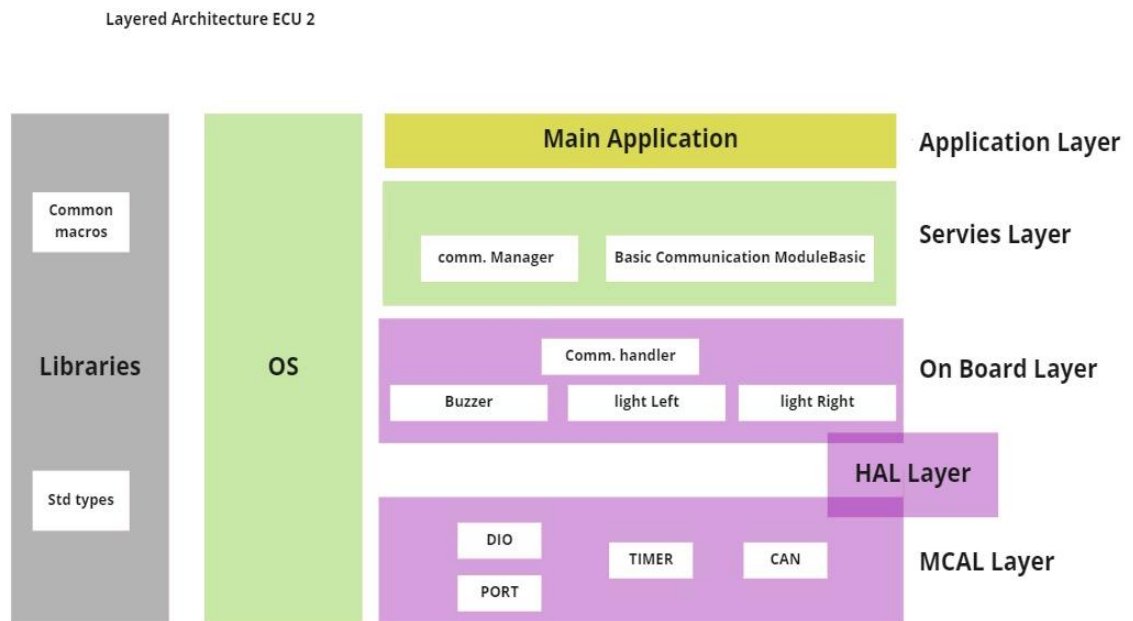
Application folder	Servies folder	On Board Layer
main.c	Operting_system.c	BCM_Handler.c
	BCM_Manager.c	Sensor_Handler.c
	Sensor_Manager.c	Door_sensor.c
		Light_switch.c
		Speed_sensor.c

MCAL folder	Configure folder
dio.c	Timer_config.c
port.c	Adc_config.c
adc.c	Can_config.c
Timer.c	Port_config.c
can.c	Dio_config.c
	Door_sensorconfig.c
	Light_switchconfig.c
	Speed_sensorconfig.c

Commen folder (all the header (name.h))
Mainapp.h / os.h / servies.h
BCS_manager.h/Sonser_manager.h
Light_switch.h / speed_sonser.h / Door_sensor.h
Dio.h / port.h / timer.h /can.h/adc.h
dio_config.h/port_config.h / timer_config.h /can_config.h /adc_config.h
Stdtypes.h /comman_macro.h /Hw.h

➤ For ECU 2:

1- the layered architecture:



My Work Ahmed Elshehry

2- Specify ECU components and modules

Components connected:

1. CAN BUS Communication Protocol (for communication between the two ECUs)
2. Light right
3. Light left
4. Buzzer

Modules:

External hardware:

1. CAN transiver module
2. Light left module
3. Light right module
4. Buzzer module

Internal hardware:

1. Port Module (initialize all pins required with modes)
2. DIO Module (switch module, Door Sensor module)
3. TIMER module (timer for application)
4. CAN Module (for can transiver data)

3- Provide full detailed APIs for each module as well as a detailed description

Layer	Module	APIs	API Details	
Application Layer	Main Application	PeriodicReceive_Status		
			Syntax:	Void PeriodicReceive_Status(uint64_t * data ,uint8_t* id_CAN);
			Sync/Async:	Synchronous
			Reentrancy:	Non-Reentrant
			Parameters:	Pointer to data act as buffer for data ,pointer of CAN bus id to id cheek it
			Return:	None
			Description:	Manage received data periodicity status of ECU1

Layer	Module	APIs	API Details	
Servies Layer	Basic Communication ModuleBasic (BCM Manager)	BCM_Manager		
			Syntax:	uint64_t BCM_Manager (uint8_t Id_Bus);
			Sync/Async:	Synchronous
			Reentrancy:	Non-Reentrant
			Parameters:	Id Bus selection want read data from bus
			Return:	Data received from ECU 1 By can bus
			Description:	Manage request the data received by CAN Bus W.R.T Id Bus selection
Servies Layer	comm. Manager	Actuator_Manager (do Monitoring Action)		
			Syntax:	Void Actuator_Manager (uint8_t actuator_id ,uint8_t action);
			Sync/Async:	Synchronous
			Reentrancy:	Non-Reentrant
			Parameters:	actuator_id selection want to do action states , action to do(on ,off)
			Return:	Nona
			Description:	Monitoring action request to do actuator selection
On Board Layer	Comm. Handler	BCM_Handler	Syntax:	uint64_t BCM_Handler (uint8_t Id_Bus);
			Sync/Async:	Synchronous
			Reentrancy:	Non-Reentrant
			Parameters:	Id Bus selection want received data from CAN BUS
			Return:	Data received from can bus
			Description:	Handler request the data Received by CAN BUS but deals with Hardware directly
Sync/Async:	Synchronous			
Reentrancy:	Non-Reentrant			
Parameters:	actuator_id selection want to do action states , action to do(on ,off)			
Return:	None			
Description:	Handler request to do action actuartor selection but deals with Hardware directly			

On Board Layer	Door Sensor	Buzzer_Init	
			Syntax: Void Buzzer_Init (void);
			Sync/Async: Synchronous
			Reentrancy: Non-Reentrant
			Parameters: None
			Return: None
		Buzzer_on	Description: Initialize the used DIO pins for digital output respect to configuration
			Syntax: void Buzzer_on(void);
			Sync/Async: Synchronous
			Reentrancy: Non-Reentrant
			Parameters: None
			Return: None
			Description: Set Buzzer to on states
		Buzzer_off	
			Syntax: void Buzzer_off(void);
			Sync/Async: Synchronous
			Reentrancy: Non-Reentrant
			Parameters: None
			Return: None
			Description: Set Buzzer to off states
On Board Layer	Light Switch	Light_Init	
			Syntax: Void Light_Init (void);
			Sync/Async: Synchronous
			Reentrancy: Non-Reentrant
			Parameters: None
			Return: None
		Light_off	Description: Initialize the used DIO pins for digital output base the configuration
			Syntax: void Light_off(void);
			Sync/Async: Synchronous
			Reentrancy: Non-Reentrant
			Parameters: None
			Return: None
			Description: Set Light to off states
		Light_on	
			Syntax: Void Light_on(void);
			Sync/Async: Synchronous
			Reentrancy: Non-Reentrant
			Parameters: None
			Return: None
			Description: Set light to on states

MCAL Layer	DIO	DIO_Init	
			Syntax: void DIO_Init (void);
			Sync/Async: Synchronous
			Reentrancy: Non-Reentrant
			Parameters: None
			Return: None
			Description: Initialize the used DIO pins with required configuration
		DIO_ReadChannel	
			Syntax: Dio_LevelTypeDIO_ReadChannel (Id_channelType id_channel);
			Sync/Async: Synchronous
			Reentrancy: Non-Reentrant
			Parameters: id_channel
			Return: Status of pin High or low
			Description: Read the channel required
		DIO_WriteChannel	
			Syntax: void DIO_WriteChannel (Dio_LevelType Level);
			Sync/Async: Synchronous
			Reentrancy: Non-Reentrant
			Parameters: Level want to write channel
			Return: None
			Description: Write the level of the channel required
MCAL Layer	PORT	PORT_init	
			Syntax: void RORT_Init (void);
			Sync/Async: Synchronous
			Reentrancy: Non-Reentrant
			Parameters: None
			Return: None
			Description: Initialize the used Port with required configuration

MCAL Layer	Timer	Timer_Init	
			Syntax: void Timer_Init (void);
			Sync/Async: Synchronous
			Reentrancy: Non-Reentrant
			Parameters: None
			Return: None
			Description: Initialize timer required configuration
		Timer_Start	
			Syntax: Void Timer_Start (uint8_t channel_Id, timer_ValueType value count);
			Sync/Async: Synchronous
			Reentrancy: Non-Reentrant
			Parameters: channel_Id / value count
			Return: None
			Description: Initialize timer required configuration
		Timer_Stop	
			Syntax: Void Timer_Stop (uint8_t channel_Id);
			Sync/Async: Synchronous
			Reentrancy: Non-Reentrant
			Parameters: Channel_Id of timer
			Return: None
			Description: Initialize timer required configuration
		CAN	
			Syntax: Void CAN_Init (void);
			Sync/Async: Synchronous
			Reentrancy: Non-Reentrant
			Parameters: None
			Return: None
			Description: Initialize CAN bus required configuration and Hardware pin CAN
			Syntax: Uint64_t CAN_ ReceivedData (uint8_t Pin_IdCAN);
			Sync/Async: Synchronous
			Reentrancy: Non-Reentrant
			Parameters: Pin_idcan
			Return: Data Recivered from Can bus
			Description: Received data from CAN Bus

Types define of argument some APIs:

Types	Define
typedef unsigned char uint8_t	Used in argument Id_Bus to select bus connect
typedef unsigned long long uint64_t	used because max width of data in CAN frame is 64 bits and used in argument Data transmitter API BCM_manager and Handler
Status_Bazzer	typedef enum {Low, High } Status_door range{0,1} size 1bit
Status_light	typedef enum{Low ,High } Status_switch range{0,1} size 1bit
typedef uint32_t T timer_ValueType;	Value of tick range from 0 to $2^{32} - 1$ size 32 bit
Typedef enum{T1 = T1PR,T2 = T2PR,Etc:} timer_ChannelType;	This enum types stores the identifier for the Channel like its name.
Dio_LevelType	typedef enum{LOW, HIGH} Dio_LevelType range{0,1} size 1bit
Id_channelType	typedef enum {Channel_1, Channel_2, Channel_3, Channel_4, Channel_5, Channel_6, Channel_7, Channel_8}Dio_LevelType range{0,8} size 1bit

4- folder structure according to the previous points:

Application folder	Servies folder	On Board Layer
main.c	Operting_system.c	BCM_Handler.c
	BCM_Manager.c	Actuator_Handler.c
	Actuator_Manager.c	Buzzer_sensor.c
		Light.c

MCAL folder	Configure folder
dio.c	Timer_config.c
port.c	Can_config.c
can.c	Dio_config.c
Timer.c	Port_config.c
	Light_config.c
	Buzzer_config.c

Commen folder (all the header (name.h))
Mainapp.h / os.h / servies.h
BCS_manager.h/ Actuator_manager.h
Light_.h / light.h
Dio.h / port.h / timer.h /can.h
dio_config.h/port_config.h / timer_config.h /can_config.h
Stdtypes.h /comman_macro.h /Hw.h