Static Design

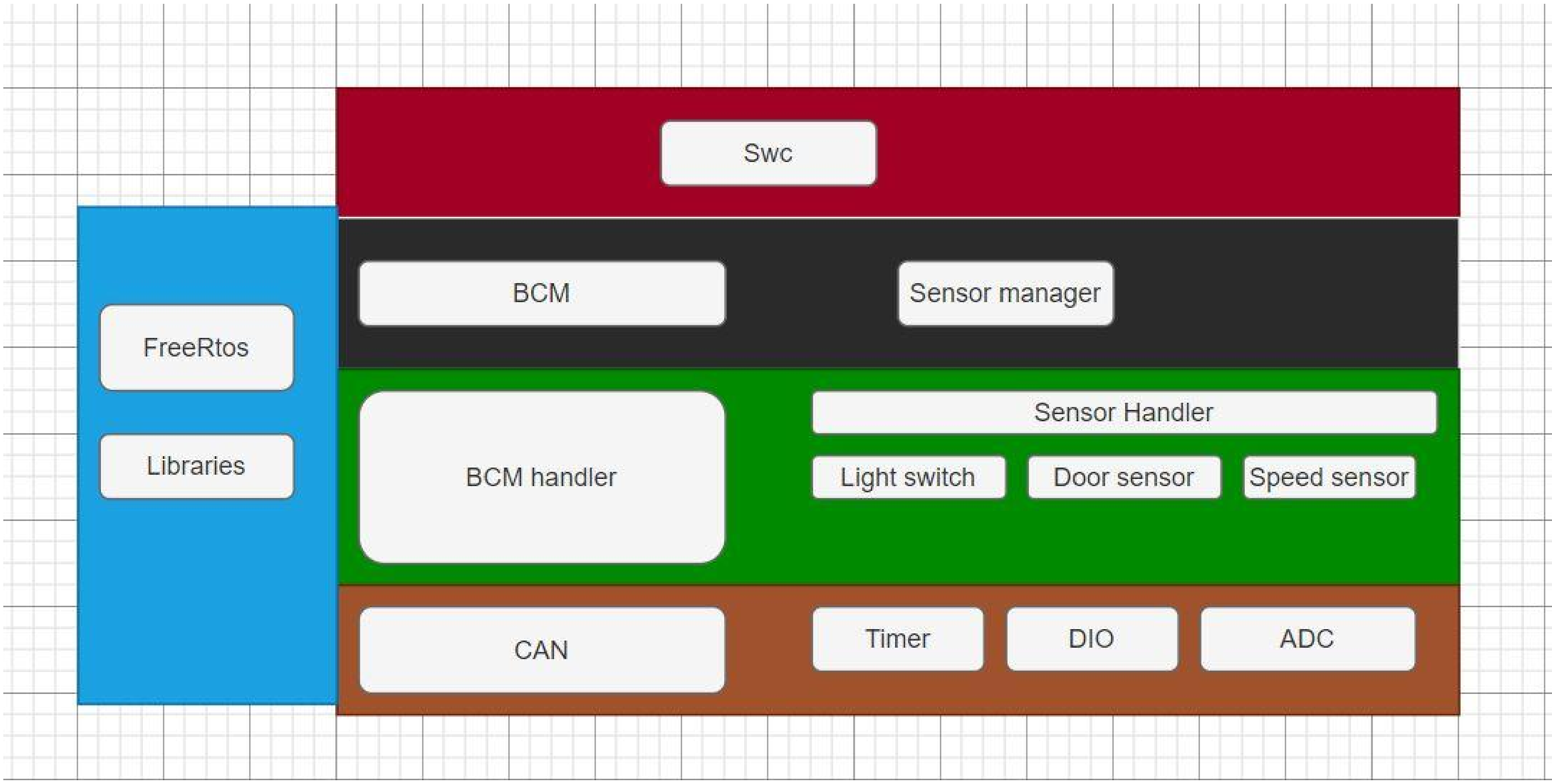
Name: **Ahmed Elkhateeb**

Email : Ahmedelkhateeb.asurt@gmail.com

Static Design:

For ECU 1:

1- the layered architecture:



2- Specify ECU components modules

ECU 1 Components:

1. **Microcontroller with following peripherals (ADC, Timer, DIO, CAN)**
2. **Door sensor**
3. **Light switch**
4. **Speed sensors**

ECU 1 Modules:

1. **FreeRTOS OS**
2. **Drivers for ADC, Timer and DIO, PORT and CAN**
3. **Separate HAL modules for Door sensor, Light switch and speed sensor**
4. **Basic communication Manager module**

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

3.1 That three Tasks to creation in Application Layer

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

3.2 That module in Servies Layer

Layer	Module	APIs	API Details
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: **Id_Bus**: that the ID commutation protocol want to connect it, **Data** :that the data want to send by BCM manager 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:** Level_States Sensor_Manager (Id_sensor Id_Sensor_read); **Sync/Async:** Synchronous **Reentrancy:** Non-Reentrant **Parameters:** **Id_Sensor_read** : that id Sensor selection want to read states **Return:** Date of states Read from sensor **Description:** Manage request read states of data from sensor selection

Types define of argument of APIs :

Types	Define
typedef unsigned char uint8_t	Used in armament Id_Bus to select bus connect range{0,255 } that range depended commutation to managed by BCM ,size 8bit
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
Level_States	typedef enum {Low, High } Level_States range{0,1} size 1bit
Id_sensor	typedef enum {Sensor_1, sensor_2, sensor_3} Id_sensor range{0,2 max sensor in project } size 2 bit

3.3 That module in On Board Layer

Layer	Module	APIs	API Details	
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:	**Id_Bus**: that the ID commutation protocol want to connect it, **Data** :that the data want to send by BCM manager
			Return:	None
			Description:	Manage request the data Transmitter by CAN Bus W.R.T Id Bus selection but deals with Hardware directly
On Board Layer	Comm. Handler	Sensor Handler	**Syntax:**	Level_States Sensor_Handler (Id_sensor Id_Sensor_read);
			Sync/Async:	Synchronous
			Reentrancy:	Non-Reentrant
			Parameters:	**Id_Sensor_read** : that id Sensor selection want to read states
			Return:	Date of states Read from sensor
			Description:	Manage request read states of data from sensor selection but deals with Hardware directly
On Board Layer	Door Sensor	DoorSensor_Init	**Syntax:**	void DoorSensor_Init (void);
			Sync/Async:	Synchronous
			Reentrancy:	Non-Reentrant
			Parameters:	None
			Return:	None
			Description:	Initialize the used DIO pins for digital input
		DoorSensor_ReadStatus	**Syntax:**	Status_door DoorSensor_ReadStatus (void);
			Sync/Async:	Synchronous
			Reentrancy:	Non-Reentrant
			Parameters:	None
			Return:	Status of the sensor door closed or opened
			Description:	Get the status of the sensor door (closed or not)

On Board Layer	Light Switch	LightSwitch_Init	**Syntax:**	Void LightSwitch_Init (void);
			Sync/Async:	Synchronous
			Reentrancy:	Non-Reentrant
			Parameters:	None
			Return:	None
			Description:	Initialize the used DIO pins for digital input
		LightSwitch_ReadStatus	**Syntax:**	Status_switch LightSwitch_ReadStatus (void);
			Sync/Async:	Synchronous
			Reentrancy:	Non-Reentrant
			Parameters:	None
			Return:	Status of the light switch Pressed or unpressed)
			Description:	Get the status of the Light Switch (Pressed or unpressed)
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:**	Status_speed SpeedSensor_ReadStatus (void);
			Sync/Async:	Synchronous
			Reentrancy:	Non-Reentrant
			Parameters:	None
			Return:	Status of the sensor speed of car that can be moving or stopped
			Description:	Read the Status value of the speed sensor (moving or stop)

Types define of argument of APIs:

Types	Define
typedef unsigned char uint8_t	Used in armament Id_Bus to select bus connect

	range{0,255 } that range depended commutation to managed by BCM ,size 8bit
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
Level_States	typedef enum {Low, High} Level_States range{0,1} size 1bit
Id_sensor	typedef enum {Sensor_1, sensor_2, sensor_3} Id_sensor range{0,2 max sensor in project } size 2 bit
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

3.4 That module in MCAL Layer

Layer	Module	APIs	API Details
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 file .
		DIO_ReadChannel	**Syntax:** LevelType DIO_ReadChannel(Id_channel channel); **Sync/Async:** Synchronous **Reentrancy:** Non-Reentrant **Parameters:** **Channel**: the value of channel want to read it the value of enum **Id_channel** **Return:** Status of pin High or low that value from Dio_LevelType **Description:** Read the channel required
		DIO_WriteChannel	**Syntax:** void DIO_WriteChannel (LevelType Level); **Sync/Async:** Synchronous **Reentrancy:** Non-Reentrant **Parameters:** **Level** : Level want to write channel high level or low level **Return:** None **Description:** Write the level of the channel required

MCAL Layer	PORT	Port_init(*Port_cfg_ptr)	**Syntax:** void Port_init(*Port_cfg_ptr) **Sync/Async:** Synchronous **Reentrancy:** Non-Reentrant **Parameters:** This API takes pointer to the configuration container of the port driver to initialize the configured pins **Return:** None **Description:** Initialize the used Port with required configuration of the pointer
MCAL Layer	PORT	void SetPinValue(port_of_Id port_Id,Pin_of_num Pin_num, Dio_LevelType level)	**Syntax:** void SetPinValue(port_of_Id port_Id,Pin_of_num Pin_num, Dio_LevelType level) **Sync/Async:** Synchronous **Reentrancy:** Non-Reentrant **Parameters:** This API takes to the configuration **port_Id** that type of port_of_Id to port_1 or port_2 …. , **Pin_num** the number of pin want to configure, **level** that initiation of level of pin high or low **Return:** None **Description:** Initialize the used Port with required configuration of the Parameters.
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 (timer_ChannelType channel, timer_ValueType value_count); **Sync/Async:** Synchronous **Reentrancy:** Non-Reentrant **Parameters:** **Channel**: that the channel wanted to start timer , **value_count** value of counter to count tick the mix value depend of over flow timer count **Return:** None **Description:** Initialize timer required configuration of Parameters to start count

MCAL Layer	Timer	Timer_Stop	**Syntax:**	Void Timer_Stop (timer_ChannelType channel);
			Sync/Async:	Synchronous
			Reentrancy:	Non-Reentrant
			Parameters:	**Channel:** channel Id of timer wanted to stopped
			Return:	None
			Description:	Stop timer required configuration id channel
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_Transmiter	**Syntax:**	void CAN_Transmiter (uint8_t Pin_Id,uint64_t Data);
			Sync/Async:	Synchronous
			Reentrancy:	Non-Reentrant
			Parameters:	Data transmitter by the can bus , Pin_id the agreement to selection the id of bus wanted connected
			Return:	None
			Description:	Transmitter data by CAN Bus
MCAL Layer	ADC	ADC_Init	**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	**Syntax:**	uint16_tADC_ReadChannel(Pin_of_num Pin_Id);
			Sync/Async:	Synchronous
			Reentrancy:	Non-Reentrant
			Parameters:	Pin_Id of ADC to read value
			Return:	The value of channel ADC

			Description:	Read the value of channel ADC

Types define of argument of APIs:

<table>
<tr><th>Types</th><th>Define</th></tr>
<tr><td>LevelType</td><td>typedef enum {LOW, HIGH} Dio_LevelType
range{0,1} size 1bit</td></tr>
<tr><td>Id_channel</td><td>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</td></tr>
<tr><td>Port_cfg_ptr that of struct to configuration
Typedef struct{uint8_t Port_Pin_Direction,
uint8_t PORT_PIN_INTERNAL_ATTACH,
uint8_t PORT_PIN_LEVEL_VALUE ,
uint8_t PORT_def_PORTx,
uint8_t PORT_def_PINx,
uint8_t PORT_def_Mode_x}port_config;</td><td>
<table>
<tr><td>Port_Pin_Direction</td><td>Used to set the direction input or output</td></tr>
<tr><td>PORT_PIN_INTERNAL_ATTACH</td><td>Used to select the internal resistance</td></tr>
<tr><td>PORT_PIN_LEVEL_VALUE</td><td>Used to specify the initial value</td></tr>
<tr><td>PORT_def_PORTx</td><td>This typedef used to point to specific port , if x equal A then this is portA</td></tr>
<tr><td>PORT_def_PINx</td><td>This typedef used to point to specific pin , if x equal 0 then this is pin0</td></tr>
<tr><td>PORT_def_Mode_x</td><td>This typedef used to point to specific mode , if x equal adc then this is adc mode</td></tr>
</table>
</td></tr>
<tr><td>port_of_Id</td><td>typedef enum {Port_1, Port _2 Port _3, Port l_4, Port _5, Port _6 Port _7} port_of_Id range{0,8} size 1bit</td></tr>
<tr><td>,Pin_of_num</td><td>typedef enum {Pin_1, Pin _2 Pin _3, Pin l_4, Pin _5, Pin_6, Pin _7,Pin_8} Pin_of_num range{0,8} size 1bit</td></tr>
<tr><td>typedef uint32_t T timer_ValueType;</td><td>Value of tick range from 0 to 2^32 -1 size 32 bit</td></tr>
</table>

Typedef enum {T1 = T1PR,T2 = T2PR,Etc:} timer_ChannelType;	This enum types stores the identifier for the Channel like its name.

4- folder structure according to the previous points:

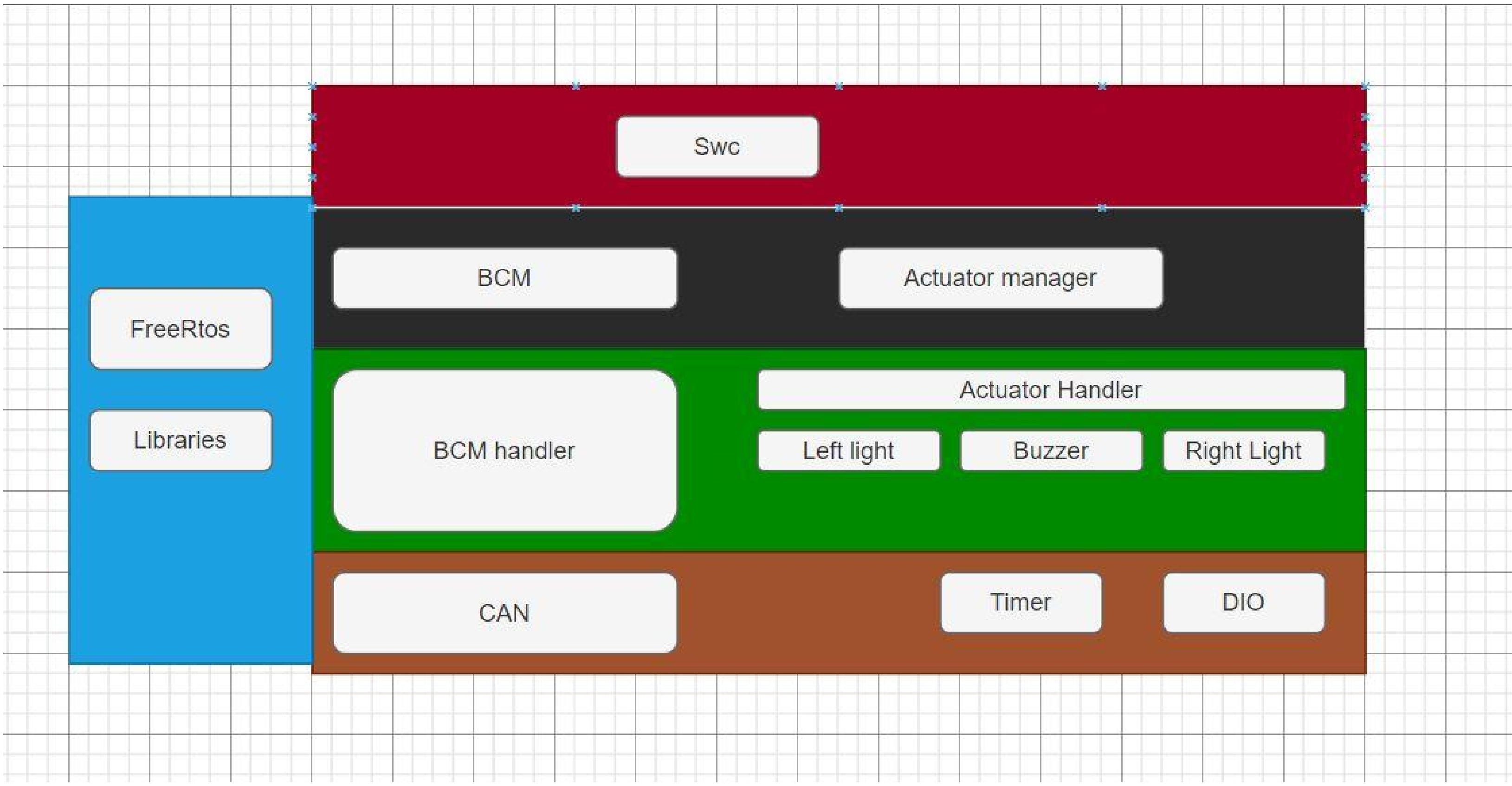
Application folder	Servies folder	On Board Layer
main.c	Operting_system.c	BCM_Handler.c
	BCM_Manager.c	Sensor_Handler.c
	Sensor_Manager.c	Door_sensor.c
		Light_switch.c
		Speed_sensor.c

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:



2- Specify ECU components &modules

ECU 2 Components

- **Microcontroller with following peripherals (Timer, DIO, CAN)**
- **Lights for left side**
- **Lights for right side**
- **Buzzer**

ECU 2 Modules

- **FreeRTOS OS**
- **Drivers for Timer and DIO, PORT and CAN**
- **Separate HAL modules for Door sensor, Light switch and speed sensor**
- **Basic communication Manager module**

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

3.1 <u>That module in Application Layer</u>

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

3.2 That module in Servies Layer

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**: that the ID commutation protocol want to connect it to received data
			Return:	Return Data frame of CAN bus that the data want to receive by CAN bus from ECU1
			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 (actuator_Id actuator , action_status_action);
			Sync/Async:	Synchronous
			Reentrancy:	Non-Reentrant
			Parameters:	**actuator_id** selection want to do action states , **action** want to do(on ,off) Actuator
			Return:	None
			Description:	Monitoring action request to do actuator selection

Types define of argument of APIs:

Types	Define
typedef unsigned char uint8_t	Used in armament Id_Bus to select bus connect range{0,255 } that range depended commutation to managed by BCM ,size 8bit
typedef unsigned long long uint64_t	used because max width of data in CAN frame is 64 bits and used in argument Data received API BCM_manager and Handler
Status_action	typedef enum {OFF,ON } status_action range{0,1} size 1bit
actuator_Id	typedef enum { actuator_1, actuator_2} actuator_Id range{0,1} max actuator in project Buzzer and light } size 1 bit

3.3 That module in On Board Layer

Layer	Module	APIs	API Details	
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**: that the ID commutation protocol want to connect it to received data
			Return:	Return Data frame of CAN bus that the data want to receive by CAN bus from ECU1
			Description:	Handler request the data received by CAN Bus W.R.T Id Bus selection but deals with Hardware directly
On Board Layer	Comm. Handler	Actuator_Handler	Syntax:	Void Actuator_Handler (actuator_Id actuator , action_status_action);
			Sync/Async:	Synchronous
			Reentrancy:	Non-Reentrant
			Parameters:	**actuator_id** selection want to do action states , **action** want to do(on ,off) Actuator
			Return:	None
			Description:	Handler request to do action actuartor selection but deals with Hardware directly
On Board Layer	Buzzer	Buzzer_Init	**Syntax:**	**Void Buzzer_Init (void);**
			Sync/Async:	Synchronous
			Reentrancy:	Non-Reentrant
			Parameters:	None
			Return:	None
			Description:	Initialize the used DIO pins for digital output respect to configuration
		Buzzer_on	**Syntax:**	**void Buzzer_on(void);**
			Sync/Async:	Synchronous

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

Types define of argument of APIs:

Types	Define
typedef unsigned char uint8_t	Used in armament Id_Bus to select bus connect range{0,255 } that range depended commutation to managed by BCM ,size 8bit

typedef unsigned long long uint64_t	used because max width of data in CAN frame is 64 bits and used in argument Data received API BCM_manager and Handler
Status_action	typedef enum {OFF,ON } status_action range{0,1} size 1bit
actuator_Id	typedef enum { actuator_1, actuator_2} actuator_Id range{0,1} max actuator in project Buzzer and light } size 1 bit

3.4 That module in MCAL Layer

Layer	Module	APIs	API Details	
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 file .
		DIO_ReadChannel	**Syntax:**	LevelType DIO_ReadChannel(Id_channel channel);
			Sync/Async:	Synchronous
			Reentrancy:	Non-Reentrant
			Parameters:	**Channel**: the value of channel want to read it the value of enum **Id_channel**
			Return:	Status of pin High or low that value from Dio_LevelType
			Description:	Read the channel required
		DIO_WriteChannel	**Syntax:**	void DIO_WriteChannel (LevelType Level);
			Sync/Async:	Synchronous
			Reentrancy:	Non-Reentrant
			Parameters:	**Level** : Level want to write channel high level or low level
			Return:	None
			Description:	Write the level of the channel required

MCAL Layer	PORT	Port_init(*Port_cfg_ptr)	**Syntax:** void Port_init(*Port_cfg_ptr) **Sync/Async:** Synchronous **Reentrancy:** Non-Reentrant **Parameters:** This API takes pointer to the configuration container of the port driver to initialize the configured pins **Return:** None **Description:** Initialize the used Port with required configuration of the pointer
MCAL Layer	PORT	void SetPinValue(port_of_Id port_Id,Pin_of_num Pin_num, Dio_LevelType level)	**Syntax:** void SetPinValue(port_of_Id port_Id,Pin_of_num Pin_num, Dio_LevelType level) **Sync/Async:** Synchronous **Reentrancy:** Non-Reentrant **Parameters:** This API takes to the configuration **port_Id** that type of port_of_Id to port_1 or port_2 …. , **Pin_num** the number of pin want to configure, **level** that initiation of level of pin high or low **Return:** None **Description:** Initialize the used Port with required configuration of the Parameters.
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 (timer_ChannelType channel, timer_ValueType value_count); **Sync/Async:** Synchronous **Reentrancy:** Non-Reentrant **Parameters:** **Channel**: that the channel wanted to start timer , **value_count** value of counter to count tick the mix value depend of over flow timer count **Return:** None **Description:** Initialize timer required configuration of Parameters to start count

MCAL Layer	Timer	Timer_Stop	**Syntax:**	Void Timer_Stop (timer_ChannelType channel);
			Sync/Async:	Synchronous
			Reentrancy:	Non-Reentrant
			Parameters:	**Channel:** channel Id of timer wanted to stopped
			Return:	None
			Description:	Stop timer required configuration id channel
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_ReceivedData	**Syntax:**	uint64_t CAN_ReceivedData (uint8_t Pin_Id);
			Sync/Async:	Synchronous
			Reentrancy:	Non-Reentrant
			Parameters:	**Pin_id** the agreement to selection the id of bus wanted connected to received Data
			Return:	Data Received by the can bus
			Description:	Received data from CAN Bus

Types define of argument of APIs:

Types	Define	
LevelType	typedef enum {LOW, HIGH} Dio_LevelType range{0,1} size 1bit	
Id_channel	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	
Port_cfg_ptr that of struct to configuration Typedef struct{uint8_t Port_Pin_Direction, uint8_t PORT_PIN_INTERNAL_ATTACH, uint8_t PORT_PIN_LEVEL_VALUE , uint8_t PORT_def_PORTx, uint8_t PORT_def_PINx,	Port_Pin_Direction	Used to set the direction input or output
	PORT_PIN_INTERNAL_ATTACH	Used to select the internal resistance

<table>
<tr><td rowspan="5">uint8_t PORT_def_Mode_x}port_config;</td><td>PORT_PIN_LEVEL_VALUE</td><td>Used to specify the initial value</td></tr>
<tr><td>PORT_def_PORTx</td><td>This typedef used to point to specific port , if x equal A then this is portA</td></tr>
<tr><td>PORT_def_PINx</td><td>This typedef used to point to specific pin , if x equal 0 then this is pin0</td></tr>
<tr><td>PORT_def_Mode_x</td><td>This typedef used to point to specific mode , if x equal adc then this is adc mode</td></tr>
<tr><td colspan="2"></td></tr>
<tr><td>port_of_Id</td><td colspan="2">typedef enum {Port_1, Port _2 Port _3, Port l_4, Port _5, Port _6 Port _7} port_of_Id range{0,8} size 1bit</td></tr>
<tr><td>,Pin_of_num</td><td colspan="2">typedef enum {Pin_1, Pin _2 Pin _3, Pin l_4, Pin _5, Pin_6, Pin _7,Pin_8} Pin_of_num range{0,8} size 1bit</td></tr>
<tr><td>typedef uint32_t T timer_ValueType;</td><td colspan="2">Value of tick range from 0 to 2^32 -1 size 32 bit</td></tr>
<tr><td>Typedef enum {T1 = T1PR, T2 = T2PR,Etc:} timer_ChannelType;</td><td colspan="2">This enum types stores the identifier for the Channel like its name.</td></tr>
</table>

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