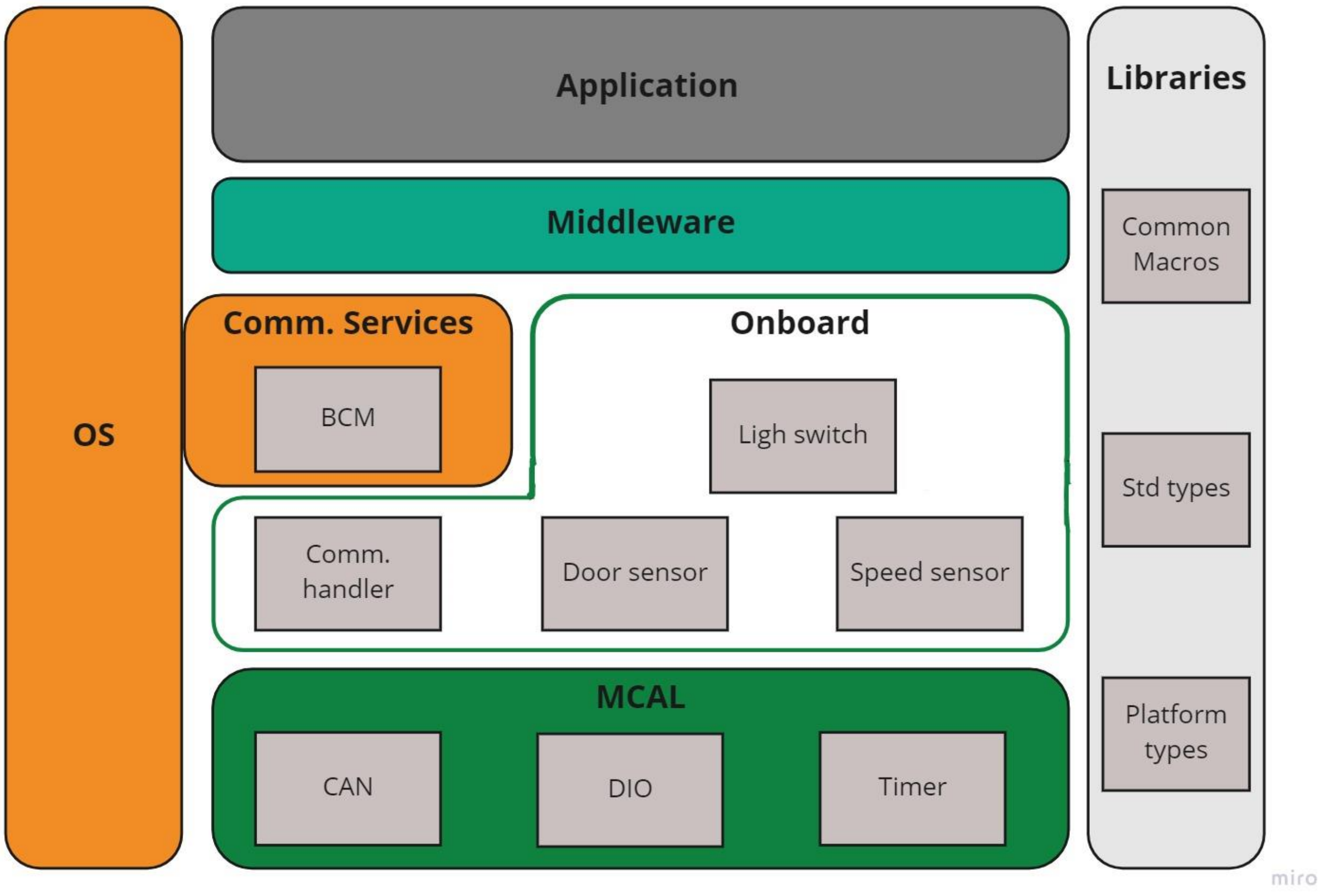
[AUTOMOTIVE DOOR CONTROL SYSTEM DESIGN]

Static Design

ECU1

- Layered architecture with ECU components and modules:



- Full detailed APIs for each module:

DIO data types

Name:	**Dio_ChannelType**
Type:	enum
Range:	Pin_0Port0, Pin_1port0, …..Pin_0Port1, Pin_1Port1,…..
Description:	Data type for the channel number.

Name:	**Dio_LevelType**
Type:	uint8
Range:	0 channel low , 1 channel high
Description:	Data type for the channel level.

Name:	**Dio_ConfigType**
Type:	structure
Elements:	Channels – An array contain all the channels to configure them.
Description:	Type defining structure to use in Dio_Init API.

DIO Functions

Function Name:	**Dio_Init**	
Arguments:	Input:	ConfigPtr - Pointer to configure data.
	Output:	None
	Input/output:	None
Sync\Async:	Synchronous	
Reentrancy:	Non reentrant	
Return:	None	
Description:	Function to Initialize the Dio module.	

Function Name:	**Dio_Read**	
Arguments:	Input:	ChannelId - ID of DIO channel.
	Output:	None
	Input/output:	None
Sync\Async:	Synchronous	
Reentrancy:	Reentrant	
Return:	Level - Level of DIO channel	
Description:	Function to read the level a given pin.	

Function Name:	**Dio_Write**	
Arguments:	Input:	ChannelId - ID of DIO channel. Level - Level of DIO channel.
	Output:	None
	Input/output:	None
Sync\Async:	Synchronous	
Reentrancy:	Reentrant	
Return:	None	
Description:	Function to write the level on a given pin.	

CAN data types

Name:	**CAN_ConfigType**
Type:	Structure
Elements:	CAN_Channel_Id, Bouad_Rate, Message_Rx_Id, Message_Tx_Id
Description:	containing the overall initialization data for the CAN driver.

Name:	**Can_HwHandleType**
Type:	uint8, uint16
Range:	Standard 0..0x0FF Extended 0..0xFFFF
Description:	Represents the hardware object handles of a CAN hardware unit. For CAN hardware units with more than 255 HW objects use extended range.

Name:	**Can_PduType**
Type:	Structure
Elements:	swPduHandle, Lenght, Id, SDU
Description:	Used to provide ID, SDU and DLC from CAN interface to CAN driver.

CAN Functions

Function Name:	**Can_Init**	
Arguments:	Input:	ConfigPtr - Pointer to configure data.
	Output:	None
	Input/output:	None
Sync\Async:	Synchronous	
Reentrancy:	Non reentrant	
Return:	None	
Description:	Function to Initialize the Can module.	

Function Name:	**Can_Write**	
Arguments:	Input:	Hth - information which HW-transmit handle shall be used for transmit. Implicitly this is also the information about the controller to use because the Hth numbers are unique inside one hardware unit. PduInfo - Pointer to SDU user memory, DLC and Identifier.
	Output:	None
	Input/output:	None
Sync\Async:	Synchronous	
Reentrancy:	Reentrant	
Return:	E_OK	0
	E_NOK	1
Description:	This function write commands.	

Function Name:	**Can_MainFunction_Write**	
Arguments:	Input:	None
	Output:	None
	Input/output:	None
Return:	None	
Description:	This function performs the polling of TX confirmation and TX cancellation confirmation when CAN_TX_PROCESSING is set to POLLING.	

Function Name:	**Can_MainFunction_Read**	
Arguments:	Input:	None
	Output:	None
	Input/output:	None
Return:	None	
Description:	This function performs the polling of RX indications when CAN_RX_PROCESSING is set to POLLING.	

Bcm data types

Name:	**Bcm_HwHandleType**	
Type:	uint8	
Range:	Standard	0..255
Description:	Represents the hardware objects. (sender or receiver)	

BCM Functions

Function Name:	**Bcm_SetCommRequest**	
Arguments:	Input:	Bcm_HwHandleType SenderId, TimeStamp, Message
	Output:	None
	Input/output:	None
Sync\Async:	Synchronous	
Reentrancy:	Reentrant	
Return:	None	
Description:	Function to select which sender will use CAN and send data now.	

Function Name:	**Bcm_GetCommRequest**	
Arguments:	Input:	Bcm_HwHandleType ReceiverId,
	Output:	None
	Input/output:	None
Sync\Async:	Synchronous	
Reentrancy:	Reentrant	
Return:	Message	
Description:	Function to select which receiver will get data now.	

CommH data types

Name:	**CommH_HwHandleType**	
Type:	uint8, uint16	
Range:	Standard	0..0x0FF
	Extended	0..0xFFFF
Description:	Represents the hardware object handles of a CAN hardware unit. For CAN hardware units with more than 255 HW objects use extended range.	

CommH Functions

Function Name:	**CommH_SetMode**	
Arguments:	Input:	CommH_HwHandleType CanId
	Output:	None
	Input/output:	None
Sync\Async:	Synchronous	
Reentrancy:	Reentrant	
Return:	None	

Description:	Function to select CAN driver internal or external.

Timer data types

Name:	**Timer_ValueType**
Type:	uint16
Range:	0 : 65535
Description:	Data type for the tick value.

Timer Functions

Function Name:	**Timer_Init**	
Arguments:	Input:	None
	Output:	None
	Input/output:	None
Sync\Async:	Synchronous	
Reentrancy:	Non reentrant	
Return:	None	
Description:	Function to initialize the Timer module.	

Function Name:	**Timer_Start**	
Arguments:	Input:	Timer_ValueType tick_time
	Output:	None
	Input/output:	None
Sync\Async:	Synchronous	
Reentrancy:	Non reentrant	
Return:	None	
Description:	Function to start the timer.	

Function Name:	**Timer_Stop**	
Arguments:	Input:	None
	Output:	None
	Input/output:	None
Sync\Async:	Synchronous	
Reentrancy:	Reentrant	
Return:	None	
Description:	Function to stop the timer.	

Function Name:	**Timer_SetCallBack**	
Arguments:	Input:	None
	Output:	void(*ptr2Fun)(void)
	Input/output:	None
Sync\Async:	Synchronous	
Reentrancy:	Reentrant	
Return:	None	
Description:	Function to assign function when ISR trigger.	

Function Name:	**Timer_Handler**	
Arguments:	Input:	None
	Output:	None
	Input/output:	None
Sync\Async:	Synchronous	
Reentrancy:	Reentrant	
Return:	None	
Description:	ISR triggerd every tick.	

Speed sensor(S) data types

Name:	**S_ValueType**
Type:	Uint8
Range:	0, 1
Description:	Data type the speed value.

Speed sensor(S) Functions

Function Name:	**S_Init**	
Arguments:	Input:	None
	Output:	None
	Input/output:	None
Sync\Async:	Synchronous	
Reentrancy:	Reentrant	
Return:	None	
Description:	Initialize HW.	

Function Name:	**S_Read**	
Arguments:	Input:	Dio_ChannelType S_Id
	Output:	None
	Input/output:	None
Sync\Async:	Synchronous	
Reentrancy:	Reentrant	
Return:	S_ValueType Speed 0 stoped 1 moving	
Description:	Read sensor output.	

Door sensor(D) data types

Name:	**D_ValueType**
Type:	Uint8
Range:	0, 1
Description:	Data type for the door state value.

Door sensor(D) Functions

Function Name:	**D_Init**	
Arguments:	Input:	None
	Output:	None
	Input/output:	None
Sync\Async:	Synchronous	
Reentrancy:	Reentrant	
Return:	None	
Description:	Initialize HW.	

Function Name:	**D_Read**	
Arguments:	Input:	Dio_ChannelType D_Id
	Output:	None
	Input/output:	None
Sync\Async:	Synchronous	
Reentrancy:	Reentrant	
Return:	D_ValueType Door_State 0 closed 1 opened	
Description:	Read sensor output.	

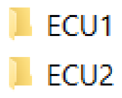
Light switch(L) data types

No need

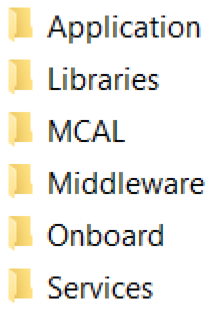
Light switch (L) Functions

Function Name:	**L_Read**	
Arguments:	Input:	Dio_ChannelType L_Id
	Output:	None
	Input/output:	None
Sync\Async:	Synchronous	
Reentrancy:	Reentrant	
Return:	Dio_LevelType Switch_State 0 released 1 pressed	
Description:	Read switch output.	

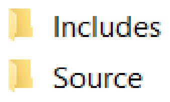
- ## Folders structure:


ECU1
ECU2

ECU1


Application
Libraries
MCAL
Middleware
Onboard
Services

Each file


Includes
Source

Application source

Appl.c
main.c

Application header

Appl.h

Libraries

Common_Macros.h
Platform_Types.h
Std_types.h

MCAL source

Can.c
Dio.c
Timer.c

MCAL header

Can.h
Dio.h
Timer.h

Middleware source

Midd.c

Middleware header

Midd.h

Onboard source

- Comm_Handler.c
- Door_Sensor.c
- Light_Switch.c
- Speed_Sensor.c

Onboard header

- Comm_Handler.h
- Door_Sensor.h
- Light_Switch.h
- Speed_Sensor.h

Services source

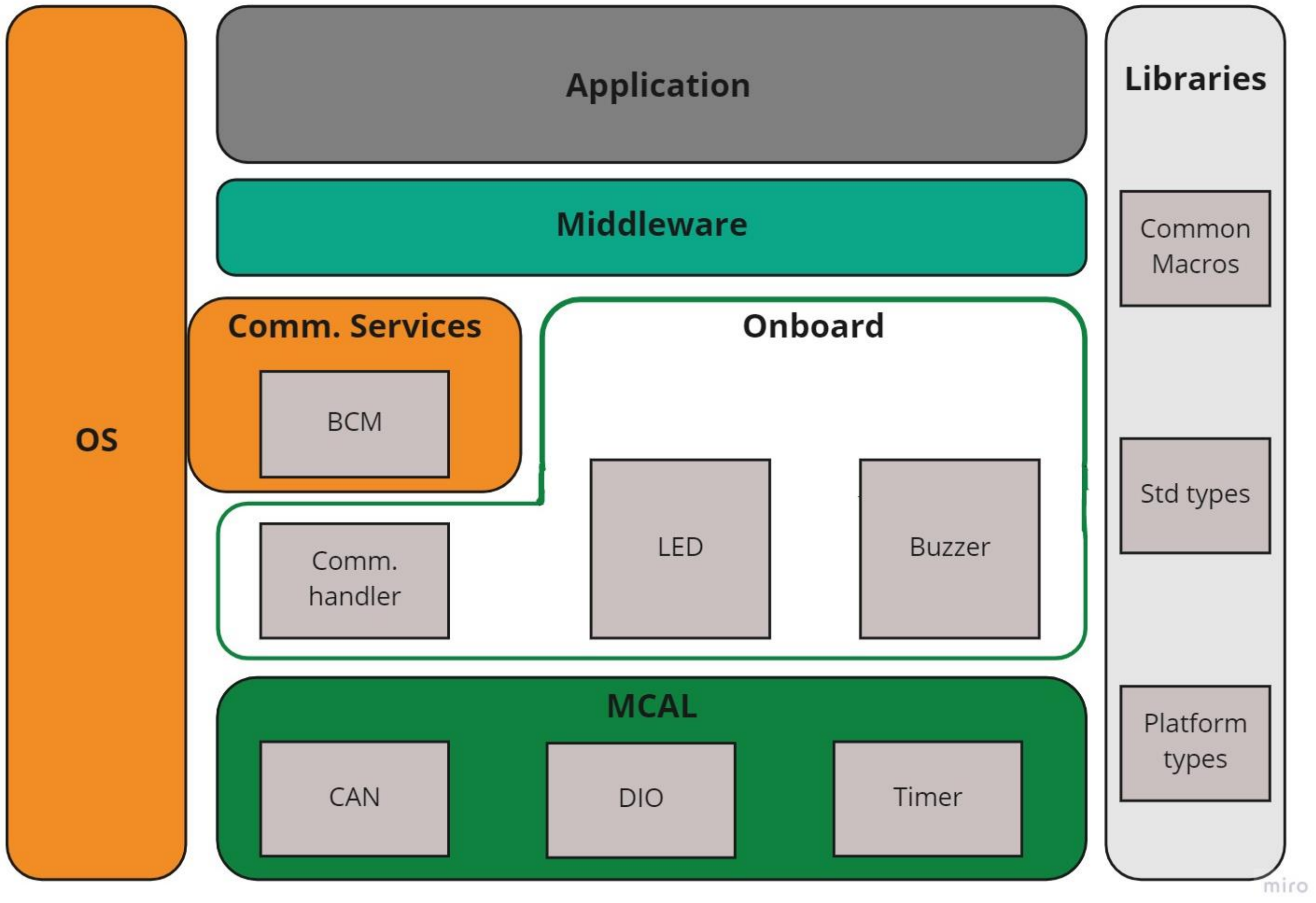
- Bcm.c
- Os.c

Services header

- Bcm.h
- Os.h

ECU2

- Layered architecture with ECU components and modules:



- Full detailed APIs for each module:

DIO, CAN, BCM, CommH and Timer same as ECU 1

Buzzer(B) data types

No need

Buzzer(B) Functions

Function Name:	**B_Init**	
Arguments:	Input:	None
	Output:	None
	Input/output:	None
Sync\Async:	Synchronous	
Reentrancy:	Non-Reentrant	
Return:	None	
Description:	Initialize HW.	

Function Name:	**B_SetOn**	
Arguments:	Input:	None
	Output:	None
	Input/output:	None
Sync\Async:	Synchronous	
Reentrancy:	Reentrant	
Return:	None	
Description:	Set the Buzzer on.	

Function Name:	**B_SetOff**	
Arguments:	Input:	None
	Output:	None
	Input/output:	None
Sync\Async:	Synchronous	
Reentrancy:	Reentrant	
Return:	None	
Description:	Set the Buzzer off.	

LEDs(RL, LL) data types

No need

LEDs(RL, LL) Functions

Function Name:	**LED_Init**	
Arguments:	Input:	None
	Output:	None
	Input/output:	None
Sync\Async:	Synchronous	

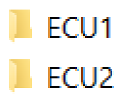
Reentrancy:	Non-Reentrant	
Return:	None	
Description:	Initialize HW.	

Function Name:	**LED_SetOn**	
Arguments:	Input:	None
	Output:	None
	Input/output:	None
Sync\Async:	Synchronous	
Reentrancy:	Reentrant	
Return:	None	
Description:	Set the LED on.	

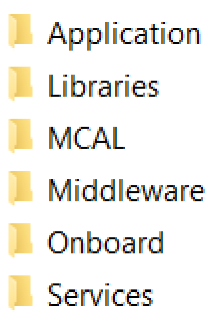
Function Name:	**LED_SetOff**	
Arguments:	Input:	None
	Output:	None
	Input/output:	None
Sync\Async:	Synchronous	
Reentrancy:	Reentrant	
Return:	None	
Description:	Set the LED off.	

Function Name:	**LED_Toggel**	
Arguments:	Input:	None
	Output:	None
	Input/output:	None
Sync\Async:	Synchronous	
Reentrancy:	Reentrant	
Return:	None	
Description:	Toggel the LED.	

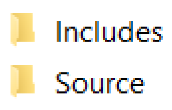
- ## Folders structure:


ECU1
ECU2

ECU2

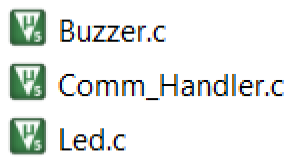

Application
Libraries
MCAL
Middleware
Onboard
Services

Each file

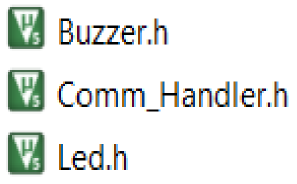

Includes
Source

All same as ECU1 except

Onboard source


Buzzer.c
Comm_Handler.c
Led.c

Onboard header


Buzzer.h
Comm_Handler.h
Led.h