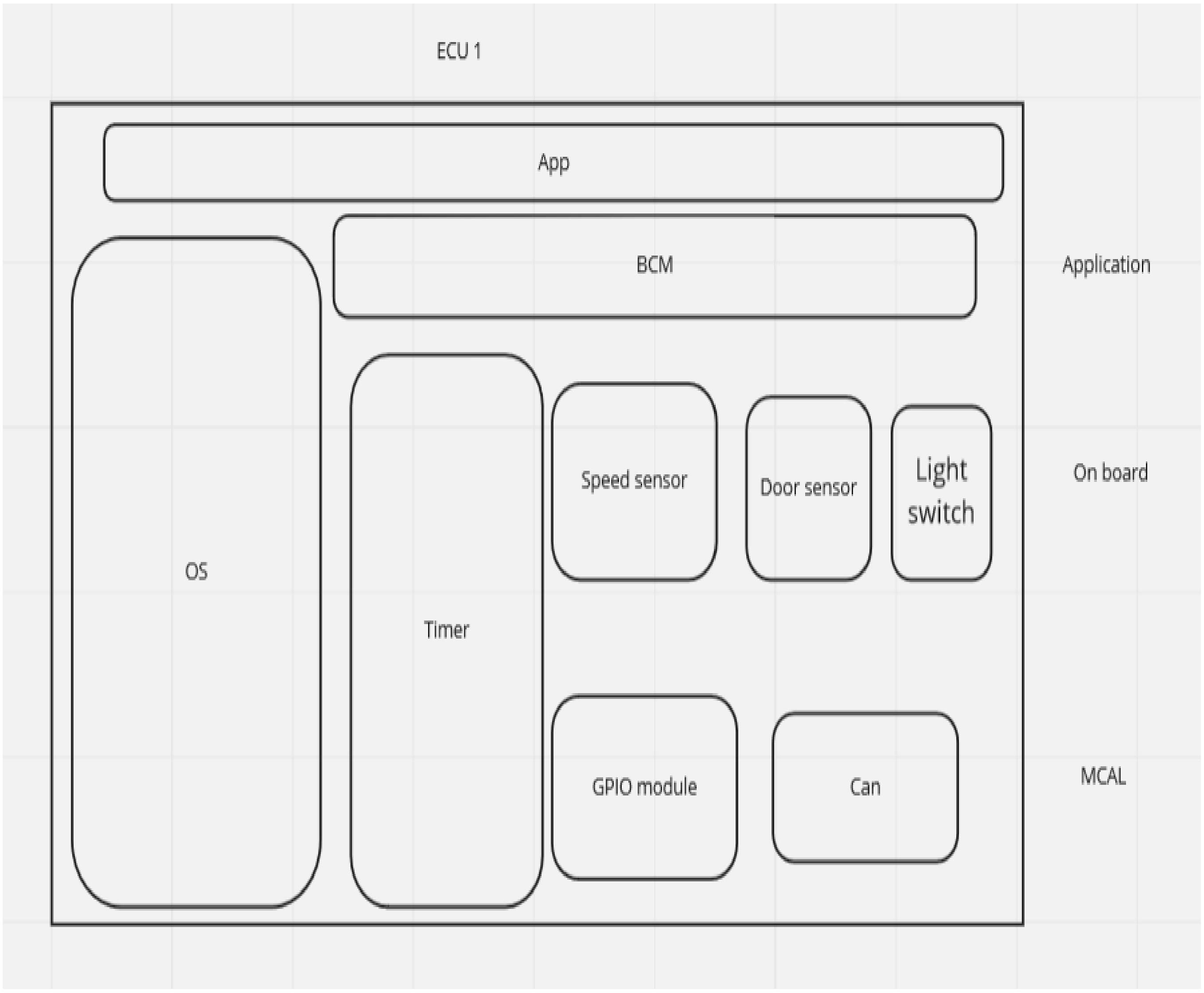
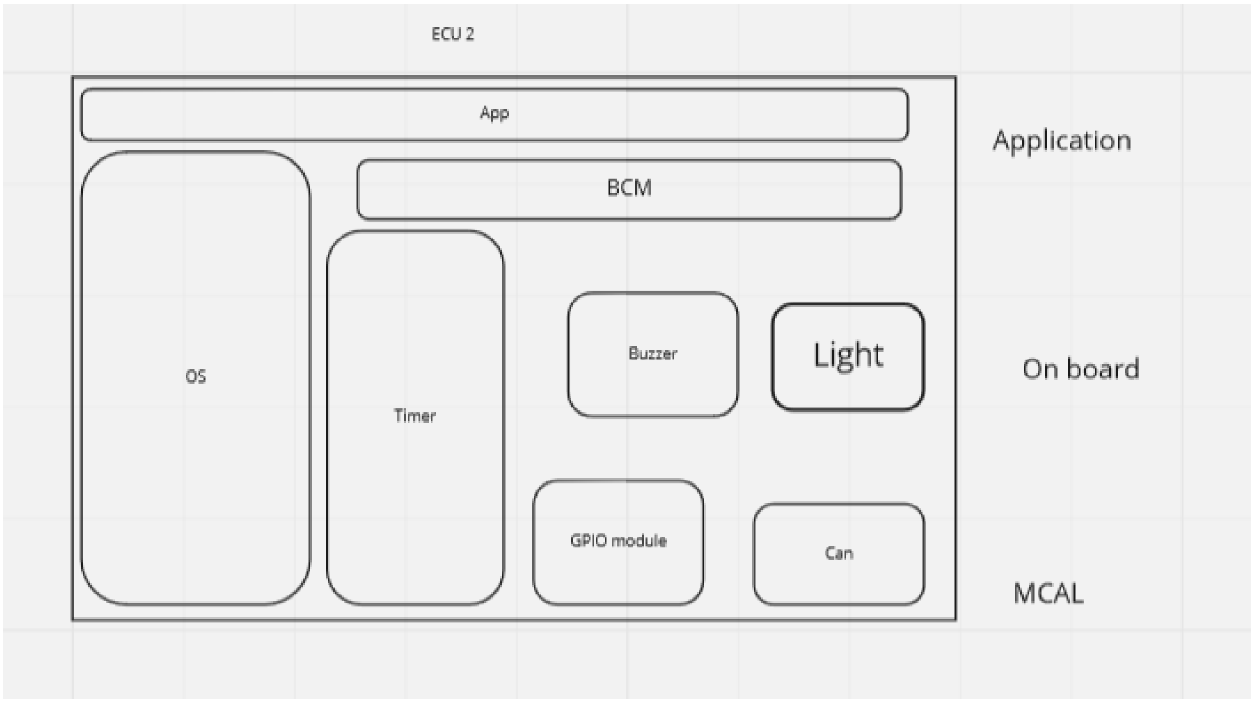
1-Layered architecture


ECU 1
App
BCM
Application
OS
Timer
Speed sensor
Door sensor
Light switch
On board
GPIO module
Can
MCAL


ECU 2
App
Application
BCM
OS
Timer
Buzzer
Light
On board
GPIO module
Can
MCAL

2-ECU components and modules for ECU1

2.1.ECU1

1-GPIO module
2-CANmodule
3-Door state module
4-Speedsensor module
5-Timer module
7-OS
8-Light switch modules

2.2.ECU2

1-GPIO module
2-CAN module
3-Light module
4-Buzzer module

3-APIS

3.1.ECU1

For GPIO:

Function	**GPIO_init**
syntax	**Void GPIO_init(GPIO_ConfigType * Config_Ptr)**
Input	**Config_Ptr**
Description	**Initialize GPIO driver**

Function	**GPIO_ReadPin**

Syntax	**GPIO_LevelType GPIO_ReadPin(GPIO_PinType GPIO_Pin)**
Input	**GPIO_Pin**
Output	**GPIO_LevelType**
Description	**Read state of pin (STD_high or STD_low)**

Function	**GPIO_WritePin**
Syntax	**void GPIO_ReadPin(GPIO_PinType GPIO_Pin, GPIO_LevelType Level_Type)**
Input	**GPIO_Pin , Level_Type**
Description	**Write state of pin (STD_high or STD_low)**

For BCM:

Function	**BCM_Writemsg**
Syntax	**Void BCM_Writemsg(CAN_messageType *message)**
Input	**message**
Desciption	**Send CAN message**

Function	**BCM_Readmsg**
Syntax	**Void BCM_Readmsg(CAN_messageType *message)**
Input	**message**
Desciption	**Recieve CAN message**

For CAN module:

Function	**Can_Init**
Syntax	**Void Can_Write(CAN_type *Config_Ptr)**

Input	**Config_Ptr**
Desciption	**Init CAN with Configuration required**

Function	**Can_Write**
Syntax	**Void Can_Write(CAN_messageType *message)**
Input	**message**
Desciption	**Send CAN message**

Function	**Can_Read**
Syntax	**Void Can_Read(CAN_messageType *message)**
Input	**message**
Desciption	**Recieve CAN message**

Function	**Can_TxConfirmation**
Syntax	**STD_RETURN Can_TxConfirmation(void)**
Input	**CAN_messageType *message**
Desciption	**Return OK or N_OK to indicate if msg sent succefully**

For Door Sensor Module:

Function	**Door_Init**

Syntax	**void Door_Init(DoorSensor_Type *Config_Ptr)**
Input	**Config_Ptr**
Desciption	**Initialize the Door**

Function	**Door_GetState**
Syntax	**DoorState_type Door_GetState(void)**
Output	**DoorState_type**
Desciption	**Read Door state(STD_high, STD_low)**

For SpeedSensor Module:

Function	**SpeedSensor_Init**
Syntax	**void SpeedSensor_Init(SpeedSensor_Type *Config_Ptr)**
Input	**Config_Ptr**
Desciption	**Initialize the SpeedSensor**

Function	**Speed_GetState**
Syntax	**SpeedState_type Speed_GetState(void)**
Output	**SpeedState_type**
Desciption	**Read Speed sensor state(STD_high, STD_low)**

For LightSwitch Module:

Function	**LightSwitch_Init**
Syntax	**void LightSensor_Init(LightSensor_type *config_ptr)**
Output	**config_ptr**

Desciption	**Initialize the light switch**

Function	**LightSwitch_GetState**
Syntax	**LightSwitchState_type LightSensor_GetState(void)**
Output	**LightSwitchState_type**
Desciption	**Read Light Switch state (STD_high, STD_low)**

For timer:

Function	**Timer_Init**
Syntax	**Void Timer_Init(TimerConfig_Type *Config_Ptr)**
Input	**Config_Ptr**
Desciption	**Initialize the timer**

Function	**Timer_Notification**
Syntax	**void Timer_Notification(void(*Ptr2Func)(void));**
Input	**Ptr2Func**
Desciption	**Set the CallBack function**

3.2.ECU2

For GPIO:

Function	**GPIO_init**

syntax	Void GPIO_init(GPIO_ConfigType * Config_Ptr)
Input	Config_Ptr
Description	Initialize GPIO driver

Function	GPIO_ReadPin
Syntax	GPIO_LevelType GPIO_ReadPin(GPIO_PinType GPIO_Pin)
Input	GPIO_Pin
Output	GPIO_LevelType
Description	Read state of pin (STD_high or STD_low)

Function	GPIO_WritePin
Syntax	void GPIO_ReadPin(GPIO_PinType GPIO_Pin, GPIO_LevelType Level_Type)
Input	GPIO_Pin , Level_Type
Description	Write state of pin (STD_high or STD_low)

For BCM:

Function	BCM_Writemsg
Syntax	Void BCM_Writemsg(CAN_messageType *message)
Input	message
Desciption	Send CAN message

Function	BCM_Readmsg
Syntax	Void BCM_Readmsg(CAN_messageType *message)

Input	**message**
Desciption	**Recieve CAN message**

For CAN module:

Function	**Can_Write**
Syntax	**Void Can_Write(CAN_messageType *message)**
Input	**message**
Desciption	**Send CAN message**

Function	**Can_Read**
Syntax	**Void Can_Read(CAN_messageType *message)**
Input	**message**
Desciption	**Recieve CAN message**

For timer:

Function	**Timer_Init**
Syntax	**Void Timer_Init(TimerConfig_Type *Config_Ptr)**
Input	**Config_Ptr**
Desciption	**Initialize the timer**

Function	**Timer_Notification**
Syntax	**void Timer_Notification(void(*Ptr2Func)(void));**

Input	**Ptr2Func**
Desciption	**Set the CallBack function**

For Buzzer Module

Function	**Buzzer_SetState**
Syntax	**void Buzzer _SetState(BuzzerState_Type Buzzer_Status)**
Input	**Buzzer_Status**
Desciption	**Set buzzer state (STD_high / STD_low)**

For Light Module

Function	**Light_SetState**
Syntax	**void Light _SetState(LightState_Type Light _Status)**
Input	**Light_Status**
Desciption	**Read Light Switch state (STD_high /STD_low)**

4-Typedefs

TypeDefs:

Name	**Type**	**Range**	**Description**
GPIO_LevelType	**uint8**	**STD_high 1U** **STD_LOW 0U**	**The possible level input or output channels can have**
GPIO_ConfigType	**Structure**	**the contents of the initialization data structure are**	**Type of the external data structure**

		specific to the microcontroller. (from autsar)	containing the initialization data for this module (inspired from autosar SWS)
CAN_messageType	Structure	the contents of the initialization data structure are specific to the microcontroller.	Data structure that contains message details to be sent
TimerConfig_Type	Structure	the contents of the initialization data structure are specific to the microcontroller.	ype of the external data structure containing the initialization data for this module (inspired from autosar SWS)
DoorState_type	uint8	STD_high 1U STD_LOW 0U	Possible levels for Door state STD_high when door is open while STD_LOW while door is closed
SpeedState_type	uint8	STD_high 1U STD_LOW 0U	Possible levels of speed STD_high when Speed is high while STD_Low when speed os low
BuzzerState_Type	uint8	STD_high 1U STD_LOW 0U	Possible levels of BuzzerState STD_high if Buzzer is On while STD_Low when buzzer is Off
LightState_Type	uint8	STD_high 1U STD_LOW 0U	Possible levels of LightState, STD_high when Light is On while STD_Low when light is off