Automotive door control system design

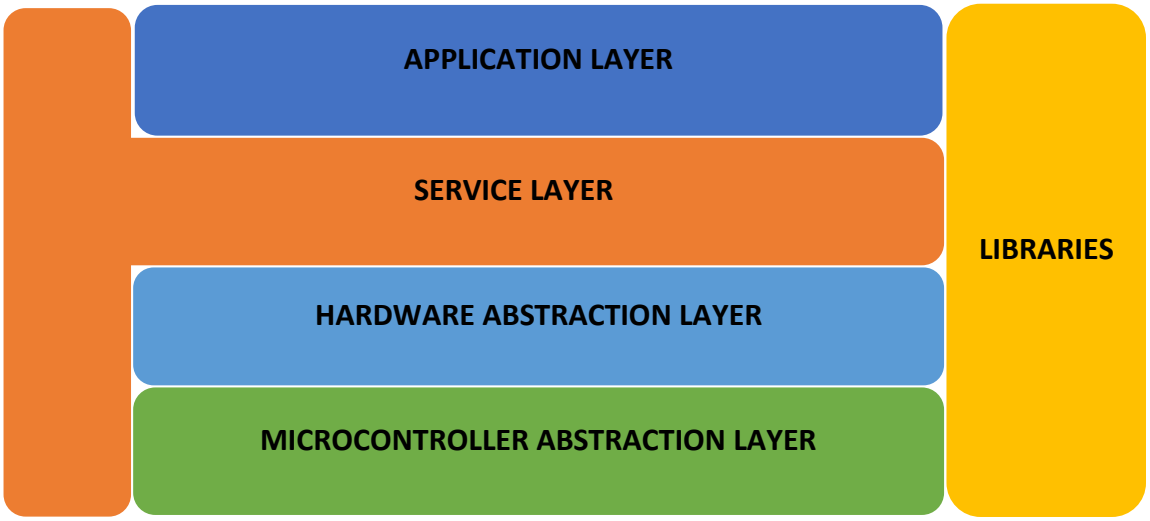
Static design analysis

Prepared by

Esmael Ehab Esmael

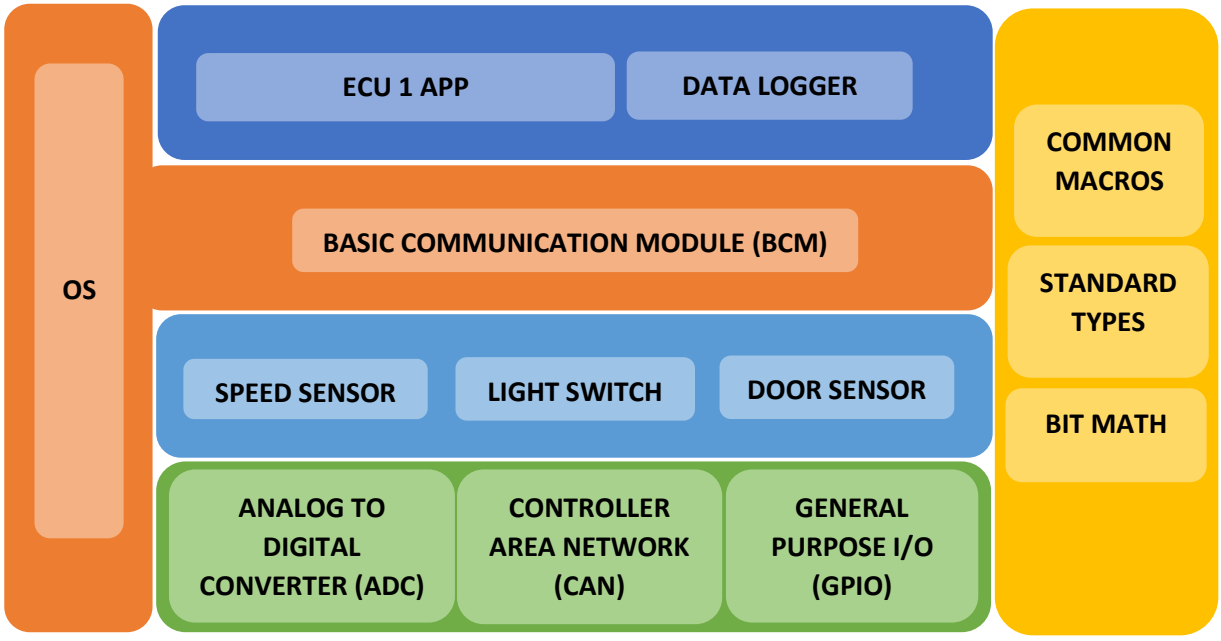
ECU 1:

1- The layered architecture for ECU 1



2- ECU 1 components and modules

- Automotive door app 1, data logger.
- Basic communication module (BCM) and Operation system (OS).
- Speed sensor, light switch and door sensor components.
- General purpose I/O (GPIO), Analog to digital converter (ADC) and Controller area network (CAN) modules.



3- APIs and typedefs for ECU 1

GPIO MODULE			
API	`ERROR_STATE GPIO_init (void);`		
Description	Initialize the GPIO with the required configuration		
Synchronization	Synchronous	**Reentrancy**	Non-reentrant
Parameters	None	**Return**	ERROR-STATE
API	`ERROR_STATE GPIO_Deinit (void);`		
Description	uninitialize the GPIO		
Synchronization	Synchronous	**Reentrancy**	Non-reentrant
Parameters	None	**Return**	ERROR-STATE
API	`void GPIO_write (uint32 a_pinId, uint8 a_value);`		
Description	Write the required GPIO pin with the required value		
Synchronization	Synchronous	**Reentrancy**	Non-reentrant
Parameters	Pin number – pin value	**Return**	None
API	`STD_VALUE GPIO_read (uint32 a_pinId);`		
Description	read the required GPIO pin value		
Synchronization	Synchronous	**Reentrancy**	Non-reentrant
Parameters	Pin number	**Return**	STD_VALUE

ADC MODULE			
API	`void ADC_init (void);`		
Description	Initialize the ADC with the required configuration		
Synchronization	Synchronous	**Reentrancy**	Non-reentrant
Parameters	Pin number - pin value	**Return**	None

API	`void ADC_Deinit (void);`		
Description	uninitialize the ADC		
Synchronization	Synchronous	**Reentrancy**	Non-reentrant
Parameters	Pin number - pin value	**Return**	None
API	`uint32 ADC_readChannel (uint8 a_chId);`		
Description	Write the required GPIO pin with the required value		
Synchronization	Synchronous	**Reentrancy**	Non-reentrant
Parameters	channel number	**Return**	uint32

CAN MODULE			
API	`ERROR_STATE CAN_init (void);`		
Description	Initialize CAN bus with the required configuration		
Synchronization	Synchronous	**Reentrancy**	Non-reentrant
Parameters	None	**Return**	ERROR_STATE
API	`ERROR_STATE CAN_Deinit (void);`		
Description	uninitialize CAN bus		
Synchronization	Synchronous	**Reentrancy**	Non-reentrant
Parameters	None	**Return**	ERROR_STATE
API	`void CAN_transmit (uint8 a_canPinId, uint64 a_message);`		
Description	Transmit messages via CAN bus		
Synchronization	Synchronous	**Reentrancy**	Non-reentrant
Parameters	Can Pin number - Message	**Return**	None

SPEED SENSOR			
API	`ERROR_STATE SpeedSensor_init (void);`		
Description	Initialize the speed sensor pin via ADC		
Synchronization	Synchronous	**Reentrancy**	Non-reentrant
Parameters	None	**Return**	ERROR_STATE
API	`Uint16 SpeedSensor_getSpeed (void);`		
Description	Get the speed from the speed sensor via ADC		
Synchronization	Synchronous	**Reentrancy**	Non-reentrant
Parameters	None	**Return**	Car speed

DOOR SENSOR			
API	`ERROR_STATE DoorSensor_init (void);`		
Description	Initialize the door sensor pin via GPIO		
Synchronization	Synchronous	**Reentrancy**	Non-reentrant
Parameters	None	**Return**	ERROR_STATE
API	`uint8 DoorSensor_getStatus (void);`		
Description	Initialize the door sensor pin via GPIO		
Synchronization	Synchronous	**Reentrancy**	Non-reentrant
Parameters	None	**Return**	uint8

LIGHT SWITCH			
API	`ERROR_STATE LightSwitch_init (void);`		
Description	Initialize the door sensor pin via GPIO		
Synchronization	Synchronous	**Reentrancy**	Non-reentrant
Parameters	None	**Return**	ERROR_STATE

API	uint8 LightSwitch_getStatus (void);		
Description	Read the light swich status		
Synchronization	Synchronous	Reentrancy	Non-reentrant
Parameters	None	Return	uint8

BCM			
API	Void BCM_mananger(uint64 a_ManagerMessage,uint8 a_bus);		
Description	Allow the application to choose which bus to send the message to		
Synchronization	Synchronous	Reentrancy	Non-reentrant
Parameters	Message – the bus sent to	Return	None

DATA LOGGER			
API	void DataLogger_saveData (uint64 a_data);		
Description	Save the required data sent to it		
Synchronization	Synchronous	Reentrancy	Non-reentrant
Parameters	Data to be saved	Return	None

ECU 1 APP			
API	void SendDoorState_task (void);		
Description	Send the door sensor state to ECU2 via CAN bus		
Synchronization	Synchronous	Reentrancy	Non-reentrant
Parameters	None	Return	None
API	void SendSpeed_task (void);		
Description	Send the speed sensor value to ECU2 via CAN bus		
Synchronization	Synchronous	Reentrancy	Non-reentrant
Parameters	None	Return	None

API	`void SendLightSwitchState_task (void);`		
Description	Send the light switch state to ECU2 via CAN bus		
Synchronization	Synchronous	**Reentrancy**	Non-reentrant
Parameters	None	**Return**	None

Typedefs used:

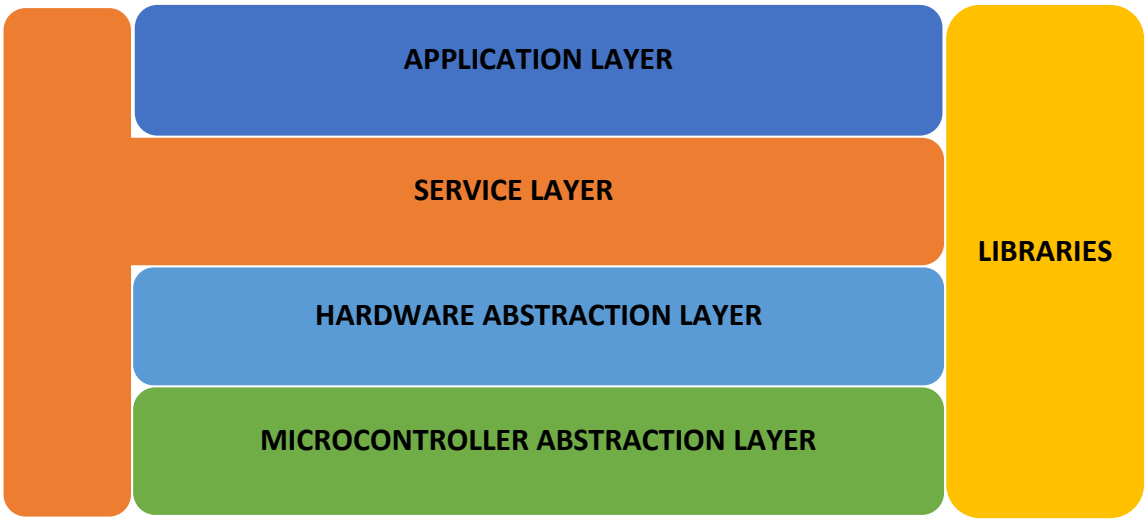
- `typedef unsigned long uint32`
- `typedef unsigned short uint16`
- `typedef unsigned char uint8`
- `typedef unsigned long long uint64`

4- Folder structure for ECU 1

folder	MCAL folder	HAL folder	SERVICE folder	APP folder	LIBRARY folder
implementation (.C) files	GPIO.c	DoorSensor.c	BCM.c	ECU1App.c	N/A
	CAN.c	SpeedSensor.c	OS.c	DataLogger.c	N/A
	ADC.c	LightSwitch.c	OS_timer.c	N/A	N/A
Configuration (.C) files	GPIO_config.c	DoorSensor_config.c	BCM_config.c	DataLogger_config.c	N/A
	CAN_config.c	SpeedSensor_config.c	OS_config.c	N/A	N/A
	ADC_config.c	LightSwitch_config.c	OS_timerconfig.c	N/A	N/A
Header (.h) files	GPIO.h	DoorSensor.h	BCM.h	ECU1App.h	Common_macros.h
	CAN.h	SpeedSensor.h	OS.h	DataLogger.h	Std_types.h
	ADC.h	LightSwitch.h	OS_timer.h	DataLogger_config.h	Bit_math.h
	GPIO_config.h	DoorSensor_config.h	BCM_config.h	N/A	N/A
	CAN_config.h	SpeedSensor_config.h	OS_config.h	N/A	N/A
	ADC_config.h	LightSwitch_config.h	OS_timerconfig.h	N/A	N/A

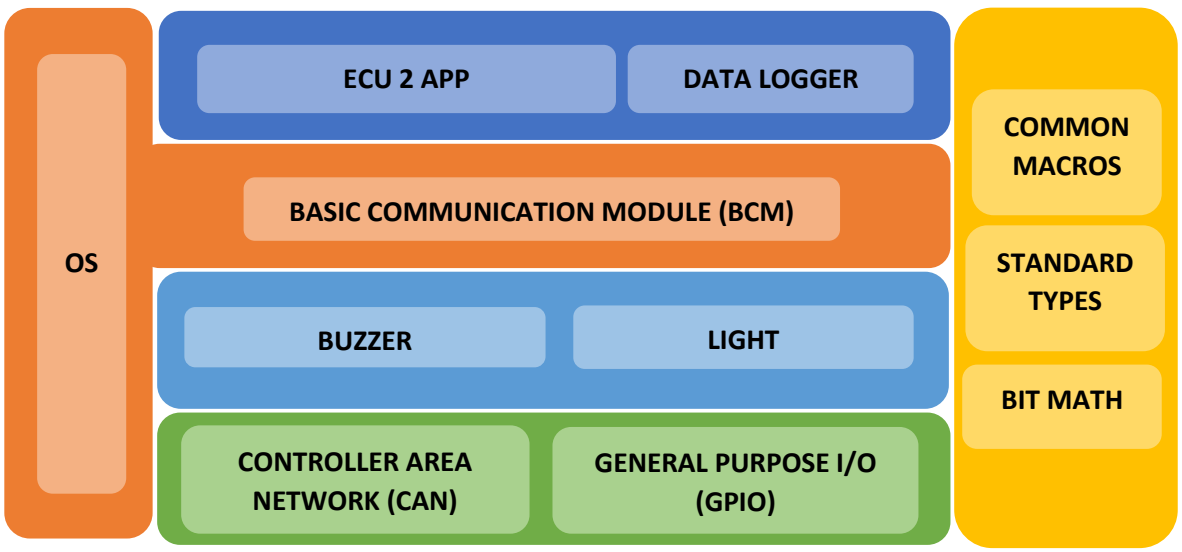
ECU 2:

1- The layered architecture for ECU 2



2- ECU 2 components and modules

- Automotive door app 2, data logger.
- Basic communication module (BCM) and Operation system (OS).
- Buzzer and light components.
- General purpose I/O (GPIO) and Controller area network (CAN) modules.



3- APIs and typedefs for ECU 2

GPIO MODULE			
API	`ERROR_STATE GPIO_init (void);`		
Description	Initialize the GPIO with the required configuration		
Synchronization	Synchronous	**Reentrancy**	Non-reentrant
Parameters	None	**Return**	ERROR-STATE
API	`ERROR_STATE GPIO_Deinit (void);`		
Description	uninitialize the GPIO		
Synchronization	Synchronous	**Reentrancy**	Non-reentrant
Parameters	None	**Return**	ERROR-STATE
API	`void GPIO_write (uint32 a_pinId, uint8 a_value);`		
Description	Write the required GPIO pin with the required value		
Synchronization	Synchronous	**Reentrancy**	Non-reentrant
Parameters	Pin number – pin value	**Return**	None
API	`STD_VALUE GPIO_read (uint32 a_pinId);`		
Description	read the required GPIO pin value		
Synchronization	Synchronous	**Reentrancy**	Non-reentrant
Parameters	Pin number	**Return**	STD_VALUE

CAN MODULE			
API	`ERROR_STATE CAN_init (void);`		
Description	Initialize CAN bus with the required configuration		
Synchronization	Synchronous	**Reentrancy**	Non-reentrant
Parameters	None	**Return**	ERROR_STATE

API	`ERROR_STATE CAN_Deinit (void);`		
Description	uninitialize CAN bus		
Synchronization	Synchronous	**Reentrancy**	Non-reentrant
Parameters	None	**Return**	ERROR_STATE
API	`Uint64 CAN_receive (uint8 a_canPinId);`		
Description	Receiving messages via CAN bus		
Synchronization	Synchronous	**Reentrancy**	Non-reentrant
Parameters	Can Pin number	**Return**	message
BUZZER			
API	`ERROR_STATE BUZZER_init (void);`		
Description	Initialize the buzzer via GPIO		
Synchronization	Synchronous	**Reentrancy**	Non-reentrant
Parameters	None	**Return**	ERROR_STATE
API	`void BUZZER_on (void);`		
Description	Set the buzzer on via GPIO		
Synchronization	Synchronous	**Reentrancy**	Non-reentrant
Parameters	None	**Return**	None
API	`void BUZZER_off (void);`		
Description	Set the buzzer off via GPIO		
Synchronization	Synchronous	**Reentrancy**	Non-reentrant
Parameters	None	**Return**	None

LIGHT			
API	`ERROR_STATE LIGHT_init (void);`		
Description	Initialize the light via GPIO		
Synchronization	Synchronous	**Reentrancy**	Non-reentrant
Parameters	None	**Return**	ERROR_STATE
API	`void LIGHT_on (void);`		
Description	Set the light on via GPIO		
Synchronization	Synchronous	**Reentrancy**	Non-reentrant
Parameters	None	**Return**	None
API	`void LIGHT_off (void);`		
Description	Set the light off via GPIO		
Synchronization	Synchronous	**Reentrancy**	Non-reentrant
Parameters	None	**Return**	None

BCM			
API	`uint64 BCM_mananger (uint8 a_bus);`		
Description	Allow the application to choose which bus to read the message from		
Synchronization	Synchronous	**Reentrancy**	Non-reentrant
Parameters	the bus to read from	**Return**	message

DATA LOGGER			
API	`void DataLogger_saveData (uint64 a_data);`		
Description	Save the required data sent to it		
Synchronization	Synchronous	**Reentrancy**	Non-reentrant
Parameters	Data to be saved	**Return**	None

ECU 2 APP			
API	`void ReceiveMessage_task (void);`		
Description	Receive the message periodically to take actions		
Synchronization	Synchronous	**Reentrancy**	Non-reentrant
Parameters	None	**Return**	None

Typedefs used:

- `typedef unsigned long uint32`
- `typedef unsigned short uint16`
- `typedef unsigned char uint8`
- `typedef unsigned long long uint64`

4- Folder structure for ECU 2

folder	MCAL folder	HAL folder	SERVICE folder	APP folder	LIBRARY folder
implementation (.C) files	GPIO.c	Buzzer.c	BCM.c	ECU2App.c	N/A
	CAN.c	Light.c	OS.c	DataLogger.c	N/A
	N/A	N/A	OS_timer.c	N/A	N/A
Configuration (.C) files	GPIO_config.c	buzzer_config.c	BCM_config.c	DataLogger_config.c	N/A
	CAN_config.c	light_config.c	OS_config.c	N/A	N/A
	N/A	N/A	OS_timerconfig.c	N/A	N/A
Header (.h) files	GPIO.h	buzzer.h	BCM.h	ECU2App.h	Common_macros.h
	CAN.h	light.h	OS.h	DataLogger.h	Std_types.h
	GPIO_config.h	Buzzer_config.h	OS_timer.h	DataLogger_config.h	Bit_math.h
	CAN_config.h	Light_config.h	BCM_config.h	N/A	N/A
	N/A	N/A	OS_config.h	N/A	N/A
	N/A	N/A	OS_timerconfig.h	N/A	N/A