AUTOSAR SIMULATION

Static and dynamic design

(Heba Ramadan Taha)

Project description:

-2 ECUs communicate with each other using BCM protocol over CAN BUS

-ECU1 will handle inputs (Sensors):

- Speed Sensor
- Door Sensor
- Light Sensor

-ECU1 Monitor system using observer pattern and send massages periodically Over CAN:

SpeedMsgs (5msec periodicity) / DoorMsg (10msec periodicity) /LightMsg (20 ms periodicity) .

-ECU2 will handle outputs (Actuators):

- Buzzer
- Light

-ECU2 Receive massages periodically and handle Actuators:

- Door open and car moving -> Buzzer on , Lights off
- Door open and car stopped -> Buzzer off , Lights on
- Door Close and lights on -> Set timer then Lights off
- Car stopped and lights on -> Buzzer on , Lights on

1-ECU_1

Static design:

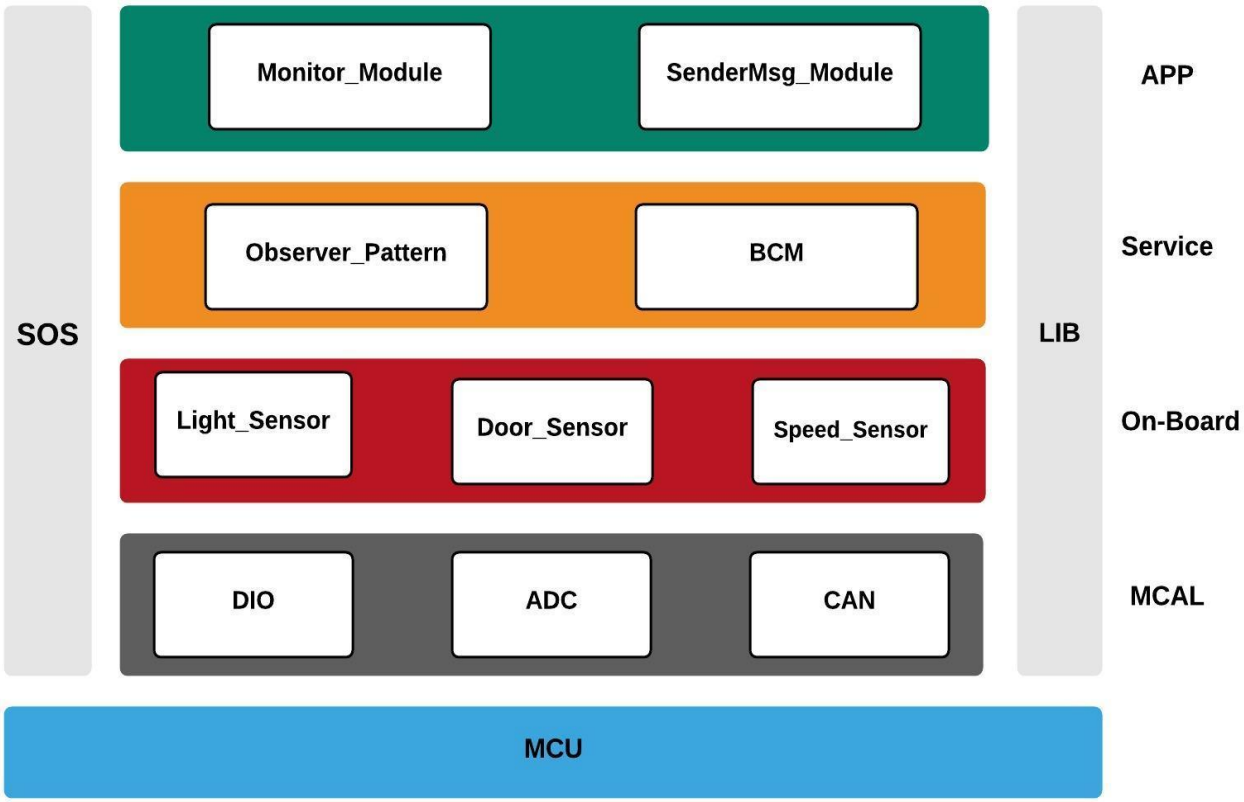
- Layered Architecture
- Modules APIs

dynamic design:

- Folder structure
- State machine
- Sequence diagram

Static design for ECU1

1- Layered Architecture:



2- Modules APIs:

- MCAL APIs

DIO APIs:

Function Name	DIO_eSetPinDirection(PinId_t PinIdCpy , PinDir_t PinDirCpy)		
Arguments	Inputs	PinIdCpy	enumeration
		description: Dio pin number to set direction	
		PinDirCpy	enumeration
		description: The direction of pin as input or output	
	Outputs	N/A	
		description:	
	Input/Output	N/A	
		description:	
Return	E_OK	0	
	E_NOK	1	
Description	Call this API to set pin direction as input or output		

Name	PinIdCpy		
Type	enumeration		
Rang	DIO_PIN0 DIO_PIN1 . . . DIO_PIN31	0 1 . . . 31	For pin 0 For pin 1 . . . For pin 31
Description	These values are to determine which pin in MC to be affected by the function		

Name	PinDirCpy		
Type	enumeration		
Rang	DIO_OUTPUT	1	To be output
	DIO_INPUT	0	To be input
Description	These values are to determine the direction of pin as output or input		

Function Name	DIO_eSetPinValue(PinId_t PinIdCpy , PinVal_t PinValCpy)		
Arguments	Inputs	PinIdCpy	enumeration
		description: The pin number to set value	
		PinValCpy	enumeration
		description: The direction of pin as high or low	
	Outputs	N/A	
		description:	
	Input/Output	N/A	
		description:	
Return	E_OK	0	
	E_NOK	1	
Description	Call this API to set pin value high or low		

Name	PinIdCpy		
Type	enumeration		
Rang	DIO_PIN0 DIO_PIN1 . . . DIO_PIN31	0 1 . . . 31	For pin 0 For pin 1 . . . For pin 31
Description	These values are to determine which pin in MC to be affected by the function		

Name	PinValCpy		
Type	enumeration		
Rang	DIO_HIGH	1	To make pin high
	DIO_LOW	0	To make pin low
Description	These values are to determine the value of pin as high or low		

Function Name	DIO_eGetPinValue(PinId_t PinIdCpy , u8 * pPinVal)		
Arguments	Inputs	PinIdCpy	enumeration
		description: The pin number to get value	
	Outputs	pPinVal	u8 *
		description: pointer to location which save value	
	Input/Output	N/A	
		description:	
Return	E_OK	0	
	E_NOK	1	
Description	Call this API to get pin value high or low		

Name	PinIdCpy		
Type	enumeration		
Rang	DIO_PIN0 DIO_PIN1 . . . DIO_PIN31	0 1 . . . 31	For pin 0 For pin 1 . . . For pin 31
Description	These values are to determine which pin in MC to be affected by the function		

ADC APIs:

Function Name	ADC_eInit(void)		
Arguments	Inputs	N/A	
		description:	
	Outputs	N/A	
		description:	
	Input/Output	N/A	
		description:	
Return	E_OK	0	
	E_NOK	1	
Description	Call this API to Initialize ADC module		

Function Name	ADC_eGetResult(ChannelId_t ChIdCpy , u16 * pResult)		
Arguments	Inputs	ChIdCpy	
		description: ADC hardware channel	
	Outputs	pResult	
		description: return digital result	
	Input/Output	N/A	
		description:	
Return	E_OK	0	
	E_NOK	1	
Description	Call this API to Initialize ADC module		

Name	ChIdCpy		
Type	enumeration		
Range	ADC_CHANNEL_0	0	For chID0
	ADC _CHANNEL_1	1	For chID1
	ADC _CHANNEL_2	2	For chID2
	ADC _CHANNEL_3	3	For chID3
	ADC _CHANNEL_4	4	For chID4
	ADC _CHANNEL_5	5	For chID5
	ADC _CHANNEL_6	6	For chID6
	ADC _CHANNEL_7	7	For chID7
Description	To determine which ADC hardware channel used to be affected by the function		

Name	pResult	
Type	u16	
Rang	0	Min value of digital result
	1023	Max(10 bit resolution adc)
Description	ADC digital value	

CAN APIs:

Function Name	CAN_eInit(void)		
Arguments	Inputs	N/A	
		description:	
	Outputs	N/A	
		description:	
	Input/Output	N/A	
		description:	
Return	E_OK	0	
	E_NOK	1	
Description	Call this API to Initialize CAN module		

Function Name	CAN_eSendByte(u8 ByteCpy)		
Arguments	Inputs	ByteCpy	u8
		description: Byte which sent by can	
	Outputs	N/A	
		description:	
	Input/Output	N/A	
		description:	
Return	E_OK	0	
	E_NOK	1	
Description	Call this API to send byte using can module		

Name	ByteCpy	
Type	u8	
Rang	0	Min value to send
	255	Max value to send
Description	Byte which sent by can module	

Function Name	CAN_eReceiveByte(u8 * pByteReceived)		
Arguments	Inputs	N/A	
		description:	
	Outputs	pByteReceived	u8 *
		description: return Received byte	
	Input/Output	N/A	
		description:	
Return	E_OK	0	
	E_NOK	1	
Description	Call this API to Receive byte using uart module		

- On-Board APIs

Light Sensor:

Datatype Table:

Name	SensorClass_t	
Type	struct	
element	Sensor_PinId	enumeration
	description: sensor dio pin	
	Sensor_value	u8
	description: sensor read value	
	ErrorState (*pfInit)(struct sensor *)	*ptf
	description: pointer to init function to initialize object_Sensor which created from this class type	
	ErrorState (*pfRead)(struct sensor * , u8* pValue)	*ptf
	description: pointer to Read function to read value of object_Sensor which created from this class type	
Description	Class type to create new sensor instance to be used with the Sensor interface	

Light Sensor APIs:

Global APIs

Function Name	LightSensor_eCreate(SensorClass_t * LightSensor, PinId_t PinCpy)		
Arguments	Inputs	LightSensor	SensorClass_t *
		description: Pointer to the sensor object (instance)	
		PinCpy	enumeration
		description: Dio pin to sensor object (instance)	
	Outputs	N/A	
		description:	
	Input/Output	N/A	
		description:	
Return	E_OK	0	
	E_NOK	1	
Description	Light Sensor constructor function to initialize instance sensor (Dio Pin or methods of sensor)		

Name	PinIdCpy		
Type	enumeration		
Rang	DIO_PIN0 . DIO_PIN31	0 . 31	Pin0 . Pin31
Description	These values are to determine which pin in MC to be affected by the function		

Name	LightSensor
Type	SensorClass_t *
Rang	
Description	instance Light sensor which created from class type

Function Name	LightSensor_eDelete(SensorClass_t * LightSensor)		
Arguments	Inputs	LightSensor	SensorClass_t *
		description: Pointer to the sensor object (instance)	
	Outputs	N/A	
		description:	
	Input/Output	N/A	
		description:	
Return	E_OK	0	
	E_NOK	1	
Description	Light Sensor destructor function to deinitialize instance sensor		

Name	LightSensor
Type	SensorClass_t *
Rang	
Description	instance Light sensor which created from class type

Private APIs

Function Name	LightSensor_eInit(SensorClass_t * LightSensor)		
Arguments	Inputs	LightSensor	SensorClass_t *
		description: Pointer to the sensor object (instance)	
	Outputs	N/A	
		description:	
	Input/Output	N/A	
		description:	
Return	E_OK	0	
	E_NOK	1	
Description	Definition of Init method in sensor class type to initialize instance light sensor hardware		

Function Name	LightSensor_eRead(SensorClass_t * LightSensor , u8* pValue)		
Arguments	Inputs	LightSensor	SensorClass_t *
		description: Pointer to the sensor object (instance)	
	Outputs	pValue	u8*
		description: pointer to value which read from light sensor instance	
	Input/Output	N/A	
		description:	
Return	E_OK	0	
	E_NOK	1	
Description	Call this Api to read light sensor value and return it		

Door Sensor:

Datatype Table:

Name	SensorClass_t	
Type	struct	
element	Sensor_PinId	enumeration
	description: sensor dio pin	
	Sensor_value	u8
	description: sensor read value	
	ErrorState (*pfInit)(struct sensor *)	*ptf
	description: pointer to init function to initialize object_Sensor which created from this class type	
	ErrorState (*pfRead)(struct sensor * , u8* pValue)	*ptf
	description: pointer to Read function to read value of object_Sensor which created from this class type	
Description	Class type to create new sensor instance to be used with the Sensor interface	

Door Sensor APIs:

Global APIs

Function Name	DoorSensor_eCreate(SensorClass_t * DoorSensor, PinId_t PinCpy)		
Arguments	Inputs	DoorSensor	SensorClass_t *
		description: Pointer to the sensor object (instance)	
		PinCpy	enumeration
		description: Dio pin to sensor object (instance)	
	Outputs	N/A	
		description:	
	Input/Output	N/A	
		description:	
Return	E_OK	0	
	E_NOK	1	
Description	Door Sensor constructor function to initialize instance sensor (Dio Pin or methods of sensor)		

Name	PinIdCpy		
Type	enumeration		
Rang	DIO_PIN0 . DIO_PIN31	0 . 31	Pin0 . Pin31
Description	These values are to determine which pin in MC to be affected by the function		

Name	DoorSensor
Type	SensorClass_t *
Rang	
Description	instance Door sensor which created from class type

Function Name	DoorSensor_eDelete(SensorClass_t * DoorSensor)		
Arguments	Inputs	DoorSensor	SensorClass_t *
		description: Pointer to the sensor object (instance)	
	Outputs	N/A	
		description:	
	Input/Output	N/A	
		description:	
Return	E_OK	0	
	E_NOK	1	
Description	Door Sensor destructor function to deinitialize instance sensor		

Name	DoorSensor
Type	SensorClass_t *
Rang	
Description	instance Light sensor which created from class type

Private APIs

Function Name	DoorSensor_eInit(SensorClass_t * LightSensor)		
Arguments	Inputs	Door	SensorClass_t *
		description: Pointer to the sensor object (instance)	
	Outputs	N/A	
		description:	
	Input/Output	N/A	
		description:	
Return	E_OK	0	
	E_NOK	1	
Description	Definition of Init method in sensor class type to initialize instance Door sensor hardware		

Function Name	DoorSensor_eRead(SensorClass_t * DoorSensor , u8* pValue)		
Arguments	Inputs	LightSensor	SensorClass_t *
		description: Pointer to the sensor object (instance)	
	Outputs	pValue	u8*
		description: pointer to value which read from door sensor instance	
	Input/Output	N/A	
		description:	
Return	E_OK	0	
	E_NOK	1	
Description	Call this Api to read door sensor value and return it		

Speed Sensor:

Datatype Table:

Name	SensorClass_t	
Type	struct	
element	Sensor_PinId	enumeration
	description: sensor dio pin	
	Sensor_value	u8
	description: sensor read value	
	ErrorState (*pfInit)(struct sensor *)	*ptf
	description: pointer to init function to initialize object_Sensor which created from this class type	
	ErrorState (*pfRead)(struct sensor * , u8* pValue)	*ptf
	description: pointer to Read function to read value of object_Sensor which created from this class type	
Description	Class type to create new sensor instance to be used with the Sensor interface	

Light Sensor APIs:

Global APIs

Function Name	SpeedSensor_eCreate(SensorClass_t * SpeedSensor, PinId_t PinCpy)		
Arguments	Inputs	SpeedSensor	SensorClass_t *
		description: Pointer to the sensor object (instance)	
		PinCpy	enumeration
		description: Dio pin to sensor object (instance)	
	Outputs	N/A	
		description:	
	Input/Output	N/A	
		description:	
Return	E_OK	0	
	E_NOK	1	
Description	Speed Sensor constructor function to initialize instance sensor (Dio Pin or methods of sensor)		

Name	PinIdCpy		
Type	enumeration		
Rang	DIO_PIN0 . DIO_PIN31	0 . 31	Pin0 . Pin31
Description	These values are to determine which pin in MC to be affected by the function		

Name	SpeedSensor
Type	SensorClass_t *
Rang	
Description	instance Speed sensor which created from class type

Function Name	SpeedSensor_eDelete(SensorClass_t * SpeedSensor)		
Arguments	Inputs	SpeedSensor	SensorClass_t *
		description: Pointer to the sensor object (instance)	
	Outputs	N/A	
		description:	
	Input/Output	N/A	
		description:	
Return	E_OK	0	
	E_NOK	1	
Description	Speed Sensor destructor function to deinitialize instance sensor		

Name	SpeedSensor
Type	SensorClass_t *
Rang	
Description	instance Light sensor which created from class type

Private APIs

Function Name	SpeedSensor_eInit(SensorClass_t * SpeedSensor)		
Arguments	Inputs	SpeedSensor	SensorClass_t *
		description: Pointer to the sensor object (instance)	
	Outputs	N/A	
		description:	
	Input/Output	N/A	
		description:	
Return	E_OK	0	
	E_NOK	1	
Description	Definition of Init method in sensor class type to initialize instance speed sensor hardware		

Function Name	SpeedSensor_eRead(SensorClass_t * SpeedSensor , u8* pValue)		
Arguments	Inputs	SpeedSensor	SensorClass_t *
		description: Pointer to the sensor object (instance)	
	Outputs	pValue	u8*
		description: pointer to value which read from speed sensor instance	
	Input/Output	N/A	
		description:	
Return	E_OK	0	
	E_NOK	1	
Description	Call this Api to read Speed sensor value and return it		

- Service layer APIs

Observer pattern:

Datatype Table:

Name	SensorObserver_t	
Type	struct	
element	Sensor_ID	enumeration
	description: Id of the sensor instance to be used with the specified sensor module	
	Sensor_Data	u8
	description: Reading of the sensor	
	NotificationHandle[]	Void()*
	description: Array of subscription list.	
Description	Used to add a new instance to the observer server	

Observer Sensor APIs:

Function Name	SensorObserver_eInit(SensorObserver_t * pSensorobserver)		
Arguments	Inputs	pSensorobserver	SensorObserver _t *
		description: Pointer to the observer object (instance)	
	Outputs	N/A	
		description:	
	Input/Output	N/A	
		description:	
Return	E_OK	0	
	E_NOK	1	
Description	observer constructor function to initialize instance sensor		

Function Name	SensorObserver_eSubscribe(SensorObserver_t * pSensorobserver , ErrorState (*pNotification)(void))		
Arguments	Inputs	pSensorobserver	SensorObserver _t *
		description: Pointer to the observer object (instance)	
		pNotification	*ptf
		description: The called function to notify an update	
	Outputs	N/A	
		description:	
	Input/Output	N/A	
		description:	
Return	E_OK	0	
	E_NOK	1	
Description	Call this api to add a new client to the subscription list, with the function to call to notify update.		

Function Name	SensorObserver_eUnsubscribe(SensorObserver_t * pSensorobserver)		
Arguments	Inputs	pSensorobserver	SensorObserver _t *
		description: Pointer to the observer object (instance)	
	Outputs	N/A	
		description:	
	Input/Output	N/A	
		description:	
Return	E_OK	0	
	E_NOK	1	
Description	Call this api to remove a client from the subscription list		

Function Name	SensorObserver_eNotify(void)		
Arguments	Inputs	N/A	
		description:	
	Outputs	N/A	
		description:	
	Input/Output	N/A	
		description:	
Return	E_OK	0	
	E_NOK	1	
Description	Call this api to notify subscribed clients of new readings		

BCM APIs:

Function Name	BCMeInit(void)		
Arguments	Inputs	N/A	
		description:	
	Outputs	N/A	
		description:	
	Input/Output	N/A	
		description:	
Return	E_OK	0	
	E_NOK	1	
Description	Call this api to initialize BCM communication module and initialize data buffer		

Function Name	BCM_eDispatcher(u32 * pBuffer)		
Arguments	Inputs	pBuffer	u32 *
		description: pointer to Buffer to send date or save data received	
	Outputs	N/A	
		description:	
	Input/Output	N/A	
		description:	
Return	E_OK	0	
	E_NOK	1	
Description	Call this api to Send data throw CAN bus.		

- App layer APIs

Monitor Module:

Function Name	Monitor_eInit(void)		
Arguments	Inputs	N/A	
		description:	
	Outputs	N/A	
		description:	
	Input/Output	N/A	
		description:	
Return	E_OK	0	
	E_NOK	1	
Description	Call this API to initialize the monitor module		

Function Name	Monitor_eLightClient(void)		
Arguments	Inputs	N/A	
		description:	
	Outputs	N/A	
		description:	
	Input/Output	N/A	
		description:	
Return	E_OK	0	
	E_NOK	1	
Description	Call this API when observer patten notify to update Light data		

Function Name	Monitor_eDoorClient(void)		
Arguments	Inputs	N/A	
		description:	
	Outputs	N/A	
		description:	
	Input/Output	N/A	
		description:	
Return	E_OK	0	
	E_NOK	1	
Description	Call this API when observer patten notify to update Door data		

Function Name	Monitor_eSpeedClient(void)		
Arguments	Inputs	N/A	
		description:	
	Outputs	N/A	
		description:	
	Input/Output	N/A	
		description:	
Return	E_OK	0	
	E_NOK	1	
Description	Call this API when observer patten notify to update Speed data		

Function Name	Monitor_eGetLightState(u8 * pLightState)		
Arguments	Inputs	N/A	
		description:	
	Outputs	pLightState	u8 *
		description:	
	Input/Output	N/A	
		description:	
Return	E_OK	0	
	E_NOK	1	
Description	Call this to get Light Mode		

Function Name	Monitor_eGetDoorState(u8 * pDoorState)		
Arguments	Inputs	N/A	
		description:	
	Outputs	pDoorState	u8 *
		description:	
	Input/Output	N/A	
		description:	
Return	E_OK	0	
	E_NOK	1	
Description	Call this to get Door Mode		

Function Name	Monitor_eGetSpeedState(u8 * pSpeedState)		
Arguments	Inputs	N/A	
		description:	
	Outputs	pSpeedState	u8 *
		description:	
	Input/Output	N/A	
		description:	
Return	E_OK	0	
	E_NOK	1	
Description	Call this to get Speed Mode		

Function Name	Monitor_eAddClient (void)		
Arguments	Inputs	N/A	
		description:	
	Outputs	pSpeedState	u8 *
		description:	
	Input/Output	N/A	
		description:	
Return	E_OK	0	
	E_NOK	1	
Description	Call this to add client to observe light sensor, door sensor and speed sensor		

SenderMsg Module:

Function Name	Sender_eInit(void)		
Arguments	Inputs	N/A	
		description:	
	Outputs	N/A	
		description:	
	Input/Output	N/A	
		description:	
Return	E_OK	0	
	E_NOK	1	
Description	Call this API to initialize Sendermsg module to send massages over CAN using BCM module		

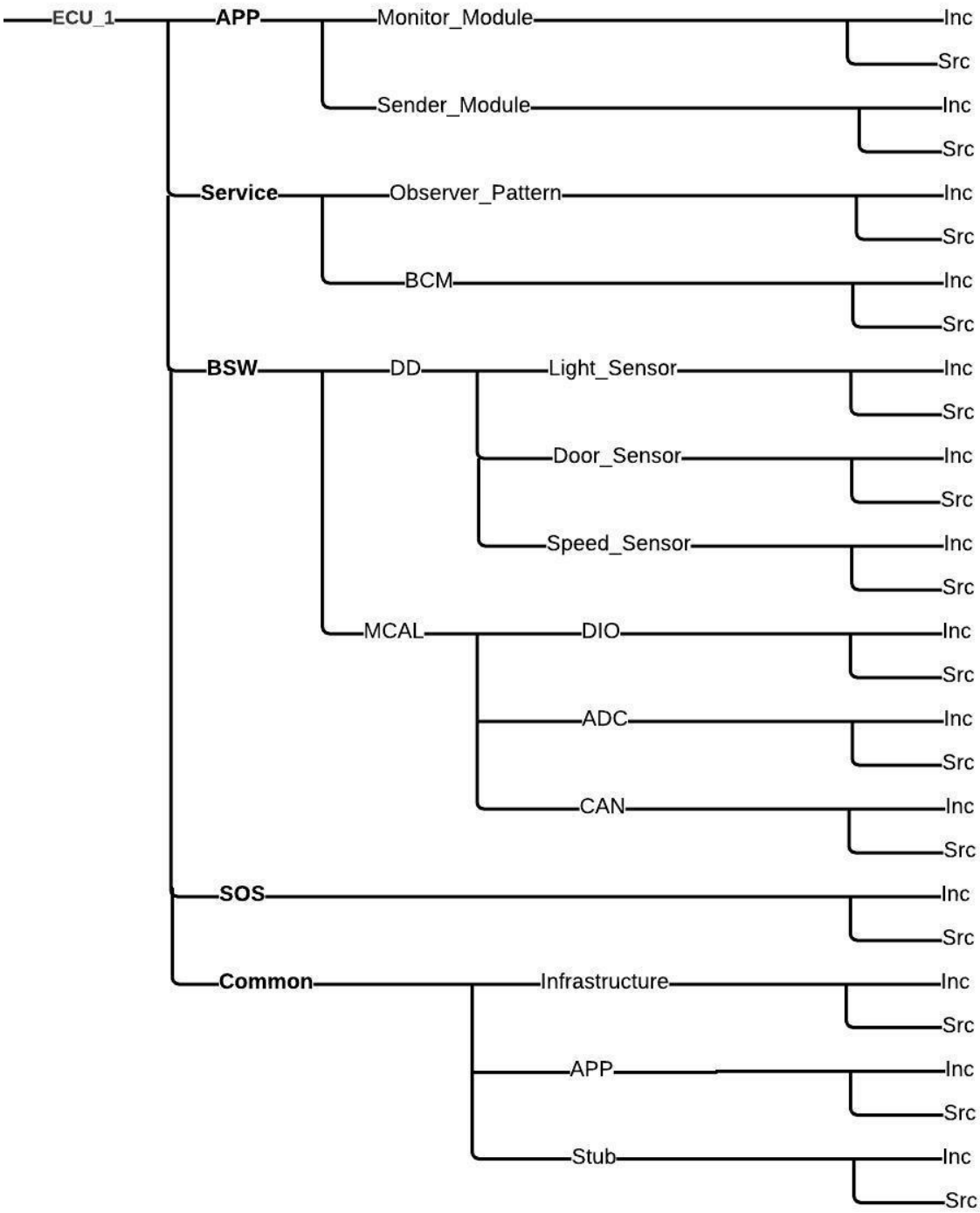
Function Name	Sender_eSpeedmsg(u8 * pSpeedmsg)		
Arguments	Inputs	N/A	
		description:	
	Outputs	pSpeedmsg	u8 *
		description: pointer to speed msg	
	Input/Output	N/A	
		description:	
Return	E_OK	0	
	E_NOK	1	
Description	Call this API to get speed massage from monitor module		

Function Name	Sender_eDoormsg(u8 * pDoormsg)		
Arguments	Inputs	N/A	
		description:	
	Outputs	pDoormsg	u8 *
		description: pointer to door msg	
	Input/Output	N/A	
		description:	
Return	E_OK	0	
	E_NOK	1	
Description	Call this API to get door massage from monitor module		

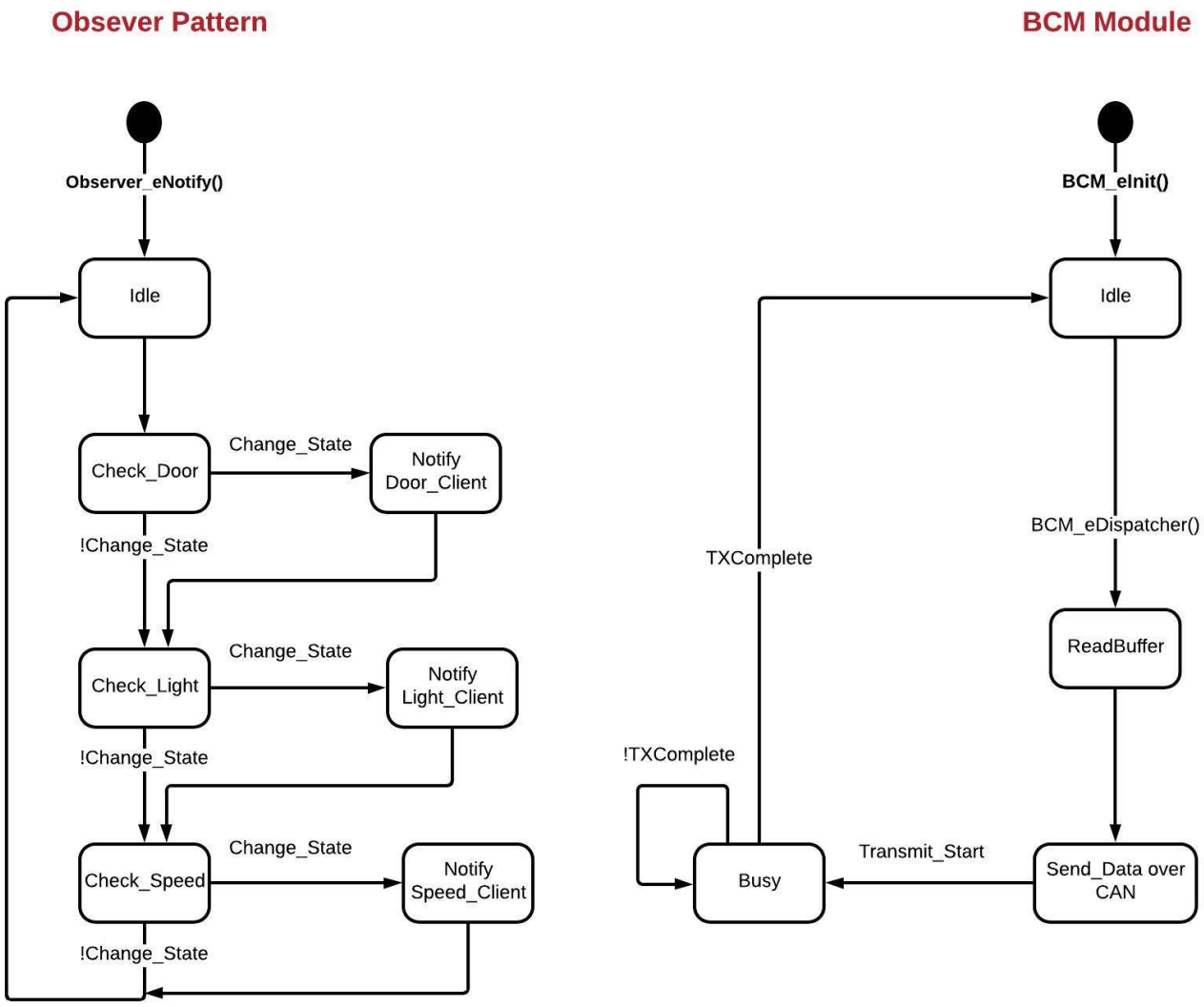
Function Name	Sender_eLightmsg(u8 * pSpeedmsg)		
Arguments	Inputs	N/A	
		description:	
	Outputs	pLightmsg	u8 *
		description: pointer to light msg	
	Input/Output	N/A	
		description:	
Return	E_OK	0	
	E_NOK	1	
Description	Call this API to get Light massage from monitor module		

Dynamic design for ECU1

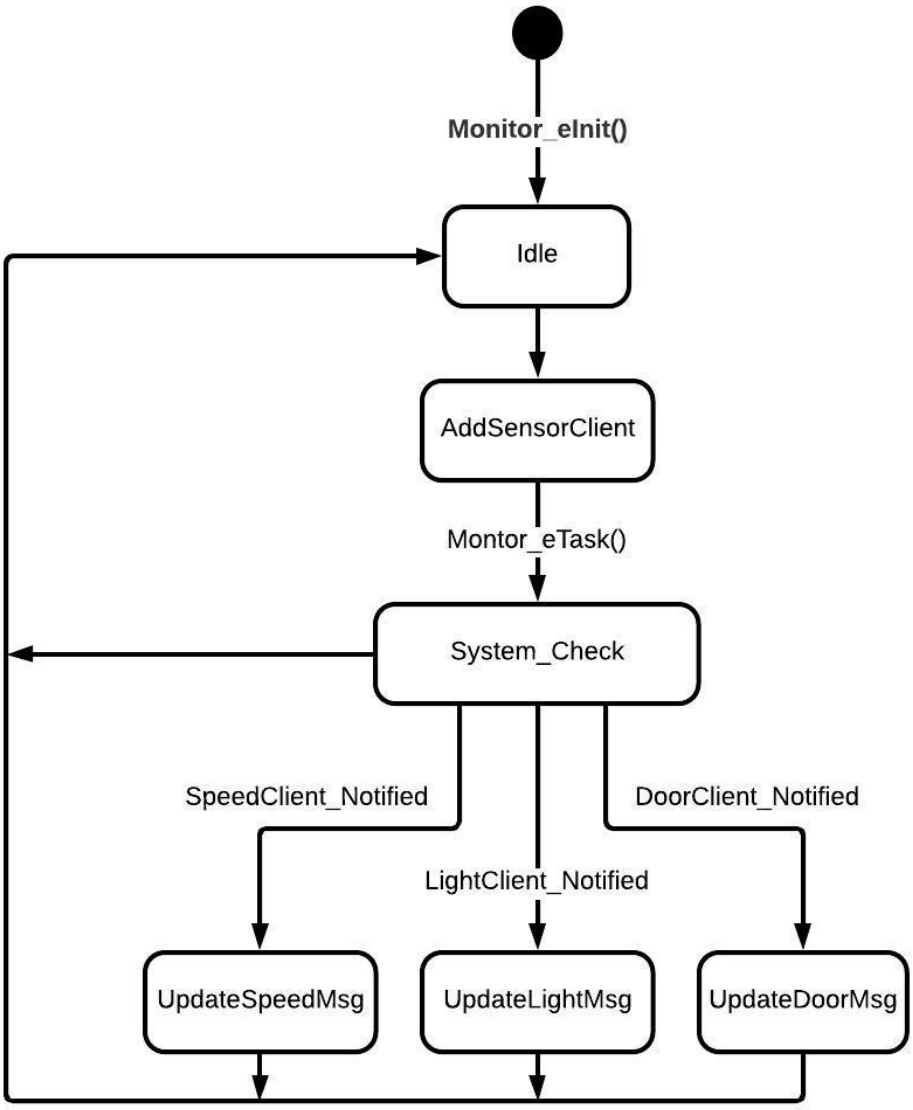
1- Folder Structure:



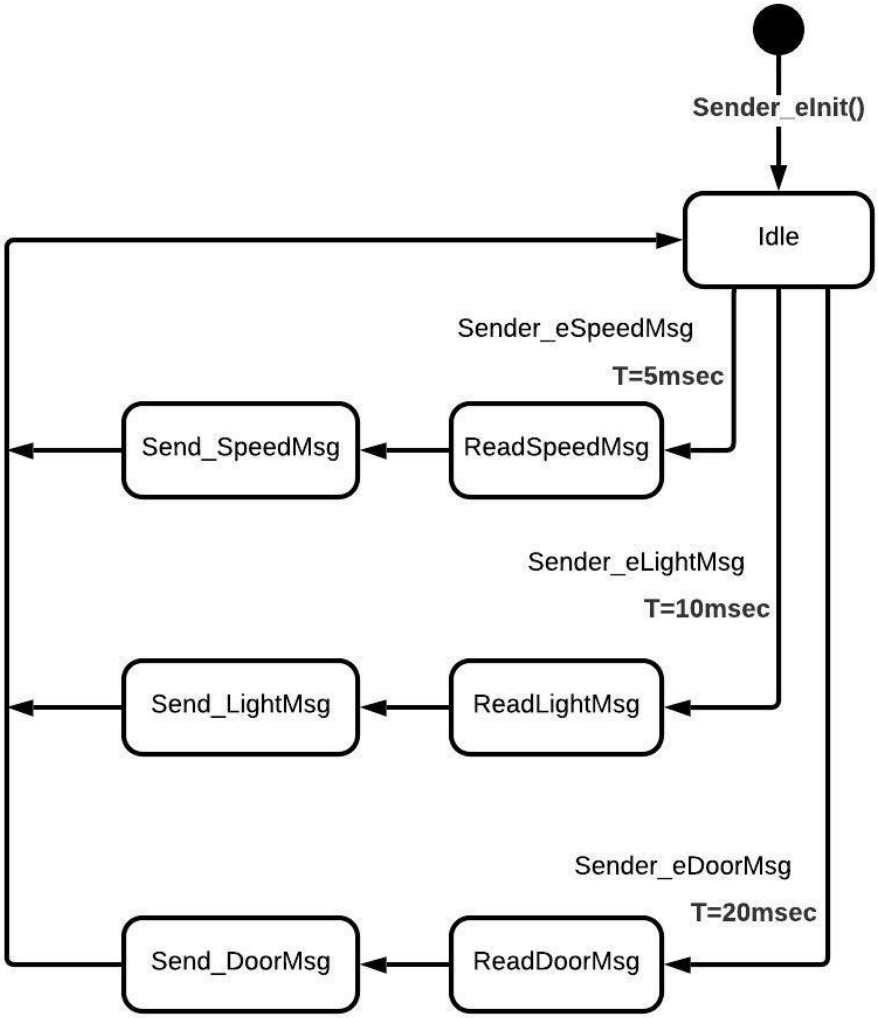
2- State machine:



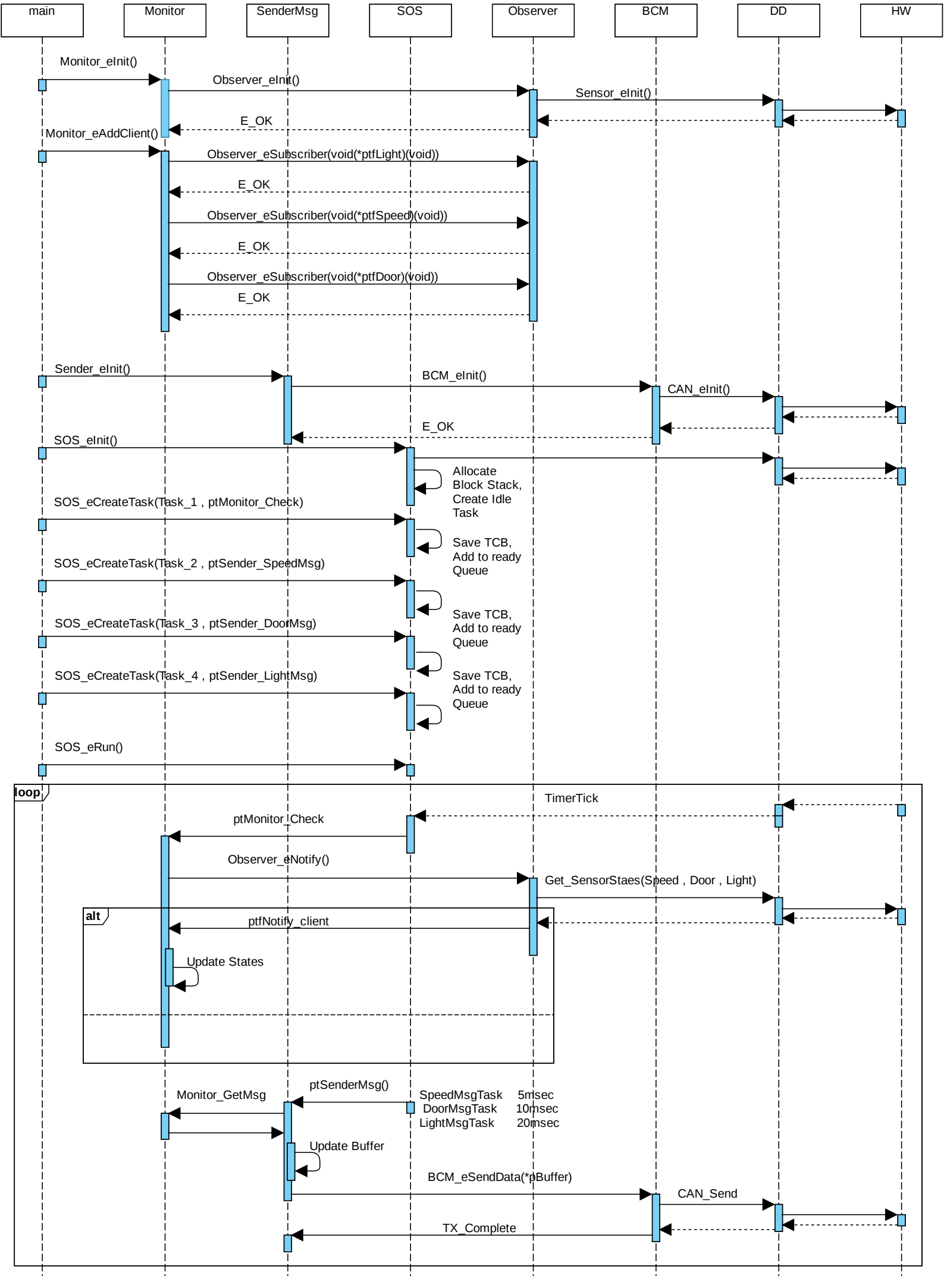
Monitor_Module



Sender_Module



3 - Sequence diagram



2- ECU_2

Static design:

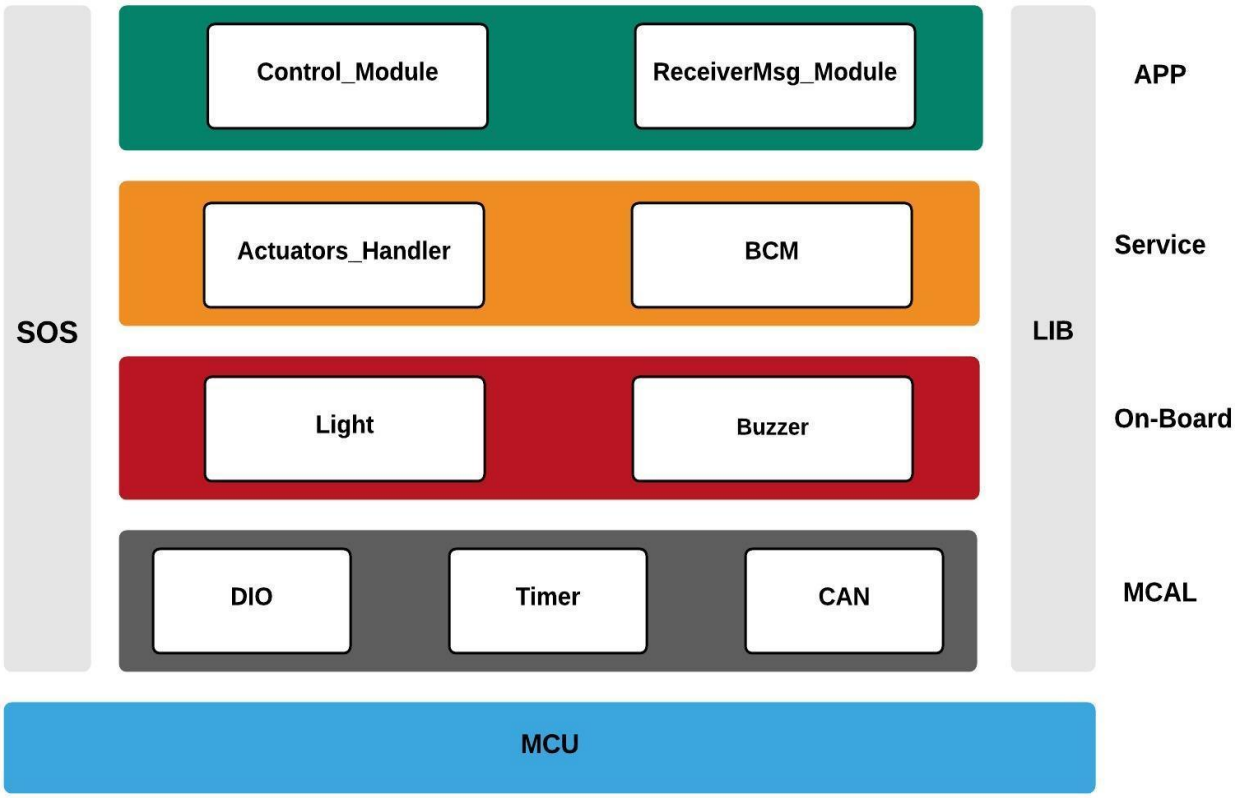
- Layered Architecture
- Modules APIs

dynamic design:

- Folder structure
- State machine
- Sequence diagram

Static design for ECU2

1- Layered Architecture:



2- Modules APIs:

- MCAL APIs

DIO APIs:

Function Name	DIO_eSetPinDirection(PinId_t PinIdCpy , PinDir_t PinDirCpy)		
Arguments	Inputs	PinIdCpy	enumeration
		description: Dio pin number to set direction	
		PinDirCpy	enumeration
		description: The direction of pin as input or output	
	Outputs	N/A	
		description:	
	Input/Output	N/A	
		description:	
Return	E_OK	0	
	E_NOK	1	
Description	Call this API to set pin direction as input or output		

Name	PinIdCpy		
Type	enumeration		
Rang	DIO_PIN0 DIO_PIN1 . . . DIO_PIN31	0 1 . . . 31	For pin 0 For pin 1 . . . For pin 31
Description	These values are to determine which pin in MC to be affected by the function		

Name	PinDirCpy		
Type	enumeration		
Rang	DIO_OUTPUT	1	To be output
	DIO_INPUT	0	To be input
Description	These values are to determine the direction of pin as output or input		

Function Name	DIO_eSetPinValue(PinId_t PinIdCpy , PinVal_t PinValCpy)		
Arguments	Inputs	PinIdCpy	enumeration
		description: The pin number to set value	
		PinValCpy	enumeration
		description: The direction of pin as high or low	
	Outputs	N/A	
		description:	
	Input/Output	N/A	
		description:	
Return	E_OK	0	
	E_NOK	1	
Description	Call this API to set pin value high or low		

Name	PinIdCpy		
Type	enumeration		
Rang	DIO_PIN0 DIO_PIN1 . . . DIO_PIN31	0 1 . . . 31	For pin 0 For pin 1 . . . For pin 31
Description	These values are to determine which pin in MC to be affected by the function		

Name	PinValCpy		
Type	enumeration		
Rang	DIO_HIGH	1	To make pin high
	DIO_LOW	0	To make pin low
Description	These values are to determine the value of pin as high or low		

Function Name	DIO_eGetPinValue(PinId_t PinIdCpy , u8 * pPinVal)		
Arguments	Inputs	PinIdCpy	enumeration
		description: The pin number to get value	
	Outputs	pPinVal	u8 *
		description: pointer to location which save value	
	Input/Output	N/A	
		description:	
Return	E_OK	0	
	E_NOK	1	
Description	Call this API to get pin value high or low		

Name	PinIdCpy		
Type	enumeration		
Rang	DIO_PIN0 DIO_PIN1 . . . DIO_PIN31	0 1 . . . 31	For pin 0 For pin 1 . . . For pin 31
Description	These values are to determine which pin in MC to be affected by the function		

Timer APIs:

Function Name	TIMER_eInit(ChannelId_t ChIdCpy)		
Arguments	Inputs	ChIdCpy	enumeration
		description: The Channel ID to initialize as Timer	
	Outputs	N/A	
		description:	
	Input/Output	N/A	
		description:	
Return	E_OK	0	
	E_NOK	1	
Description	Call this API to Initialize Timer as specified in the configuration file. Selecting timer hardware based on ChannelID.		

Name	ChIdCpy		
Type	enumeration		
Rang	TIMER_CHANNEL_0	0	For Timer0
	TIMER_CHANNEL_1	1	For Timer1
	TIMER_CHANNEL_2	2	For Timer2
Description	To determine which timer to be affected by the function		

Function Name	TIMER_eStart(ChannelId_t ChIdCpy , u16 TimeCountCpy)		
Arguments	Inputs	ChIdCpy	enumeration
		description: The Channel ID to Start Timer	
		TimeCountCpy	u16
		description: Number of counts in ms	
	Outputs	N/A	
		description:	
	Input/Output	N/A	
		description:	
Return	E_OK	0	
	E_NOK	1	
Description	Call this API to Start timer and count from 0 till number of counts in msec		

Name	ChIdCpy		
Type	enumeration		
Rang	TIMER_CHANNEL_0	0	For Timer0
	TIMER_CHANNEL_1	1	For Timer1
	TIMER_CHANNEL_2	2	For Timer2
Description	To determine which timer to be affected by the function		

Name	TimeCountCpy	
Type	u16	
Rang	1	Min number 1msec
	60000	Max 1 hour
Description	To determine number of mSec to count by timer	

Function Name	TIMER_eStop(ChannelId_t ChIdCpy)		
Arguments	Inputs	ChIdCpy	enumeration
		description: The Channel ID of Timer	
	Outputs	N/A	
		description:	
	Input/Output	N/A	
		description:	
Return	E_OK	0	
	E_NOK	1	
Description	Call this API to stop Timer		

Name	ChIdCpy		
Type	enumeration		
Rang	TIMER_CHANNEL_0	0	For Timer0
	TIMER_CHANNEL_1	1	For Timer1
	TIMER_CHANNEL_2	2	For Timer2
Description	To determine which timer to be affected by the function		

Function Name	TIMER_eGetStatus(ChannelId_t ChIdCpy , TimStat_t* pTimerState)		
Arguments	Inputs	ChIdCpy	enumeration
		description: The Channel ID to initialize as Timer	
	Outputs	pTimerState	TimStat_t*
		description: return state of timer	
	Input/Output	N/A	
		description:	
Return	E_OK	0	
	E_NOK	1	
Description	Call this API to get current state of timer		

Name	ChIdCpy		
Type	enumeration		
Rang	TIMER_CHANNEL_0	0	For Timer0
	TIMER_CHANNEL_1	1	For Timer1
	TIMER_CHANNEL_2	2	For Timer2
Description	To determine which timer to be affected by the function		

Name	pTimerState		
Type	enumeration		
Rang	TIMER_RUNNIG	0	
	TIMER_STOP	1	
	TIMER_EXPIRED	2	
Description	To describe current state of timer		

CAN APIs:

Function Name	CAN_eInit(void)		
Arguments	Inputs	N/A	
		description:	
	Outputs	N/A	
		description:	
	Input/Output	N/A	
		description:	
Return	E_OK	0	
	E_NOK	1	
Description	Call this API to Initialize CAN module		

Function Name	CAN_eSendByte(u8 ByteCpy)		
Arguments	Inputs	ByteCpy	u8
		description: Byte which sent by can	
	Outputs	N/A	
		description:	
	Input/Output	N/A	
		description:	
Return	E_OK	0	
	E_NOK	1	
Description	Call this API to send byte using can module		

Name	ByteCpy	
Type	u8	
Rang	0	Min value to send
	255	Max value to send
Description	Byte which sent by can module	

Function Name	CAN_eReceiveByte(u8 * pByteReceived)		
Arguments	Inputs	N/A	
		description:	
	Outputs	pByteReceived	u8 *
		description: return Received byte	
	Input/Output	N/A	
		description:	
Return	E_OK	0	
	E_NOK	1	
Description	Call this API to Receive byte using CAN module		

- On-Board APIs

Light Actuator:

Datatype Table:

Name	LightClass_t	
Type	struct	
element	Light_PinId	enumeration
	description: Light dio pin	
	ErrorState (*pfLight_Init)(struct Light *)	*ptf
	description: pointer to Light init function to initialize object (instance)Light which created from this class type	
	ErrorState (*pfLightOn)(struct Light *)	*ptf
	description: pointer to function to Light on object which created from this class type	
	ErrorState (*pfLightOff)(struct Light *)	*ptf
Description	Class type to create new Light instance to be used with the Light interface	

Light Actuator APIs:

Global APIs

Function Name	Light_eCreate(LightClass_t * pLight , PinId_t PinCpy)		
Arguments	Inputs	pLight	LightClass _t *
		description: Pointer to the Light object (instance)	
		PinCpy	enumeration
		description: Dio pin to light object (instance)	
	Outputs	N/A	
		description:	
	Input/Output	N/A	
		description:	
Return	E_OK	0	
	E_NOK	1	
Description	Light constructor function to initialize instance Light (Dio Pin or methods of Light)		

Name	PinIdCpy		
Type	enumeration		
Rang	DIO_PIN0 . DIO_PIN31	0 . 31	Pin0 . Pin31
Description	These values are to determine which pin in MC to be affected by the function		

Name	pLight
Type	LightClass _t *
Rang	
Description	instance Light which created from class type

Function Name	Light_eDelete(LightClass_t * pLight)		
Arguments	Inputs	pLight	LightClass _t *
		description: Pointer to the Light object (instance)	
	Outputs	N/A	
		description:	
	Input/Output	N/A	
		description:	
Return	E_OK	0	
	E_NOK	1	
Description	Call this api to destruct Light instance		

Private APIs

Function Name	Light_eInit(LightClass_t * pLight)		
Arguments	Inputs	pLight	LightClass _t *
		description: Pointer to the Light object (instance)	
	Outputs	N/A	
		description:	
	Input/Output	N/A	
		description:	
Return	E_OK	0	
	E_NOK	1	
Description	Definition of Init method in Light class type to initialize instance light hardware		

Function Name	Light_eLightON(LightClass_t * pLight)		
Arguments	Inputs	pLight	LightClass _t *
		description: Pointer to the Light object (instance)	
	Outputs	N/A	
		description:	
	Input/Output	N/A	
		description:	
Return	E_OK	0	
	E_NOK	1	
Description	Call this Api to light ON instance		

Function Name	Light_eLightOFF(LightClass_t * pLight)		
Arguments	Inputs	pLight	LightClass _t *
		description: Pointer to the Light object (instance)	
	Outputs	N/A	
		description:	
	Input/Output	N/A	
		description:	
Return	E_OK	0	
	E_NOK	1	
Description	Call this Api to light OFF instance		

Buzzer Actuator:

Datatype Table:

Name	BuzzerClass_t	
Type	struct	
element	Buzzer_PinId	enumeration
	description: Buzzer dio pin	
	ErrorState (* Buzzer_Init)(struct Buzzer *)	*ptf
	description: pointer to Light init function to initialize object (instance) Buzzer which created from this class type	
	ErrorState (*pf BuzzerOn)(struct Buzzer *)	*ptf
	description: pointer to function to Buzzer on object which created from this class type	
	ErrorState (*pfBuzzerOff)(struct Buzzer *)	*ptf
Description	Class type to create new Buzzer instance to be used with the Buzzer interface	

Buzzer Actuator APIs:

Global APIs

Function Name	Buzzer_eCreate(BuzzerClass_t * pBuzzer , PinId_t PinCpy)		
Arguments	Inputs	pBuzzer	BuzzerClass _t *
		description: Pointer to the Buzzer object (instance)	
		PinCpy	enumeration
		description: Dio pin to light object (instance)	
	Outputs	N/A	
		description:	
	Input/Output	N/A	
		description:	
Return	E_OK	0	
	E_NOK	1	
Description	Buzzer constructor function to initialize instance Buzzer (Dio Pin or methods of Buzzer)		

Name	PinIdCpy		
Type	enumeration		
Rang	DIO_PIN0 . DIO_PIN31	0 . 31	Pin0 . Pin31
Description	These values are to determine which pin in MC to be affected by the function		

Name	pBuzzer
Type	LightClass _t *
Rang	
Description	instance Buzzer which created from class type

Function Name	Buzzer_eDelete(BuzzerClass_t * pBuzzer)		
Arguments	Inputs	pBuzzer	BuzzerClass _t *
		description: Pointer to Buzzer object (instance)	
	Outputs	N/A	
		description:	
	Input/Output	N/A	
		description:	
Return	E_OK	0	
	E_NOK	1	
Description	Call this api to destruct Buzzer instance		

Private APIs

Function Name	Buzzer_eInit(BuzzerClass_t * pBuzzer)		
Arguments	Inputs	pBuzzer	BuzzerClass _t *
		description: Pointer to Buzzer object (instance)	
	Outputs	N/A	
		description:	
	Input/Output	N/A	
		description:	
Return	E_OK	0	
	E_NOK	1	
Description	Definition of Init method in Buzzer class type to initialize instance Buzzer hardware		

Function Name	Buzzer_eBuzzerON(BuzzerClass_t * pBuzzer)		
Arguments	Inputs	pBuzzer	BuzzerClass _t *
		description: Pointer to Buzzer object (instance)	
	Outputs	N/A	
		description:	
	Input/Output	N/A	
		description:	
Return	E_OK	0	
	E_NOK	1	
Description	Call this Api to light ON instance		

Function Name	Buzzer_eBuzzerOFF(BuzzerClass_t * pBuzzer)		
Arguments	Inputs	pBuzzer	BuzzerClass _t *
		description: Pointer to Buzzer object (instance)	
	Outputs	N/A	
		description:	
	Input/Output	N/A	
		description:	
Return	E_OK	0	
	E_NOK	1	
Description	Call this Api to Buzzer OFF instance		

- Service layer API

BCM APIs:

Function Name	BCMeInit(void)		
Arguments	Inputs	N/A	
		description:	
	Outputs	N/A	
		description:	
	Input/Output	N/A	
		description:	
Return	E_OK	0	
	E_NOK	1	
Description	Call this api to initialize BCM communication module and initialize data buffer		

Function Name	BCM_eDispatcher(u32 * pBuffer)		
Arguments	Inputs	pBuffer	u32 *
		description: pointer to Buffer to save data received	
	Outputs	N/A	
		description:	
	Input/Output	N/A	
		description:	
Return	E_OK	0	
	E_NOK	1	
Description	Call this api to Receive data throw CAN bus.		

Actuator Handler Module:

Function Name	ActuatorHandler_eInit(void)		
Arguments	Inputs	N/A	
		description:	
	Outputs	N/A	
		description:	
	Input/Output	N/A	
		description:	
Return	E_OK	0	
	E_NOK	1	
Description	Call this API to initialize the Actuator Handler module to init light and buzzer Actuator		

Function Name	ActuatorHandler _eSetActuators(DoorState_t DoorCpy , SpeedState_t SpeedCpy , LightState_t LightCpy)		
Arguments	Inputs	DoorCpy	enumeration
		description: Current state for door	
		SpeedCpy	enumeration
		description: Current state for speed	
		LightCpy	enumeration
		description: Current state for light	
	Outputs	N/A	
		description:	
	Input/Output	N/A	
		description:	
Return	E_OK	0	
	E_NOK	1	
Description	Call this API from periodic function in control module to set actuator (light-Buzzer) actions based on input states (Light Sensor- Door Sensor- Speed Sensor)		

Name	DoorCpy	
Type	enumeration	
Rang	0	For DOOR_OPEN
	1	For DOOR_CLOSE
Description	Current door state	

Name	SpeedCpy	
Type	enumeration	
Rang	0	For CAR_MOVE
	1	For CAR_STOP
Description	Current Speed state	

Name	LightCpy	
Type	enumeration	
Rang	0	For LIGHT_ON
	1	For LIGHT_OFF
Description	Current Light state	

- App layer APIs

ReceiverMsg Module:

Function Name	Receiver_eInit(void)		
Arguments	Inputs	N/A	
		description:	
	Outputs	N/A	
		description:	
	Input/Output	N/A	
		description:	
Return	E_OK	0	
	E_NOK	1	
Description	Call this API to initialize Receiver msg module to read massages from CAN using BCM module		

Function Name	Receiver_eRecievemsg(u32 * pBuffer)		
Arguments	Inputs	pBuffer	u32*
		description: pointer to Buffer to save data received from CAN	
	Outputs	N/A	
		description:	
	Input/Output	N/A	
		description:	
Return	E_OK	0	
	E_NOK	1	
Description	Call this API to receive massage periodically every 5msec		

Control Module:

Function Name	Control_eInit(void)		
Arguments	Inputs	N/A	
		description:	
	Outputs	N/A	
		description:	
	Input/Output	N/A	
		description:	
Return	E_OK	0	
	E_NOK	1	
Description	Call this API to initialize the control module to init Actuator Handler		

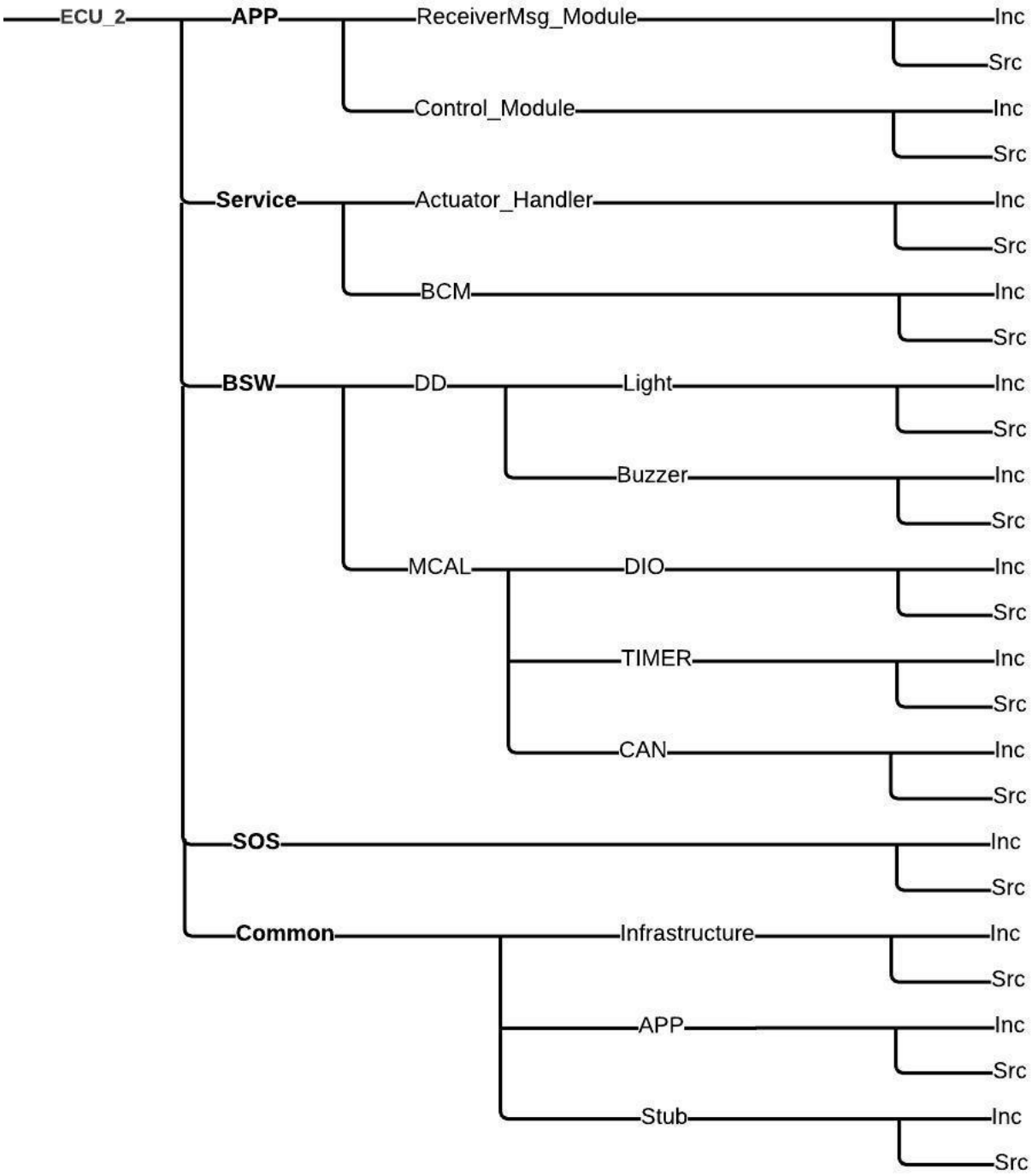
Function Name	Control_eReadSpeedmsg(void)		
Arguments	Inputs	N/A	
		description:	
	Outputs	N/A	
		description:	
	Input/Output	N/A	
		description:	
Return	E_OK	0	
	E_NOK	1	
Description	Call this API to Read speed msg which received from DataBuffer and update Speed State		

Function Name	Control_eReadDoormsg(void)		
Arguments	Inputs	N/A	
		description:	
	Outputs	N/A	
		description:	
	Input/Output	N/A	
		description:	
Return	E_OK	0	
	E_NOK	1	
Description	Call this API to Read door msg which received from DataBuffer and update door State		

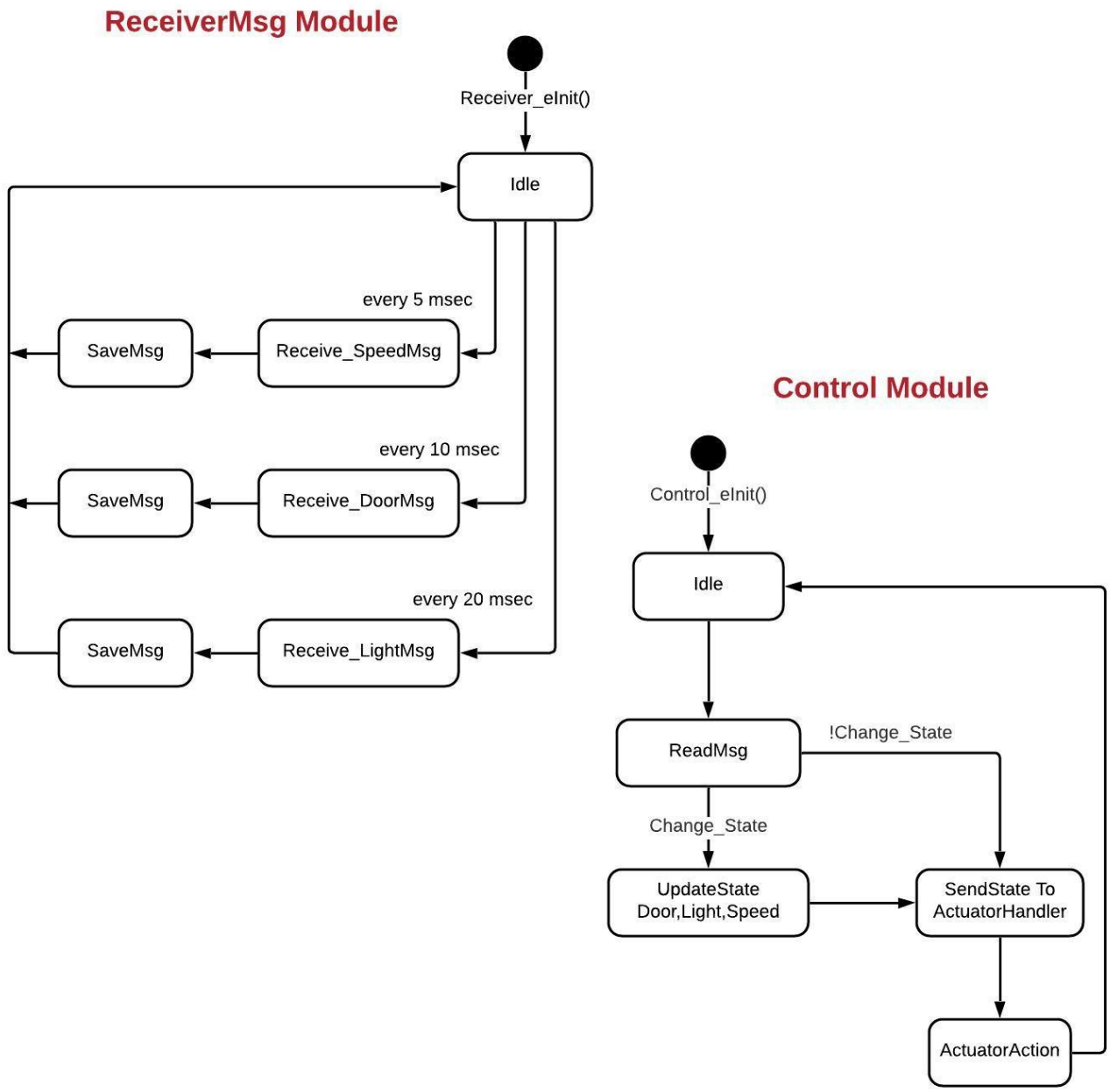
Function Name	Control_eReadLightmsg(void)		
Arguments	Inputs	N/A	
		description:	
	Outputs	N/A	
		description:	
	Input/Output	N/A	
		description:	
Return	E_OK	0	
	E_NOK	1	
Description	Call this API to Read light msg which received from DataBuffer and update light State		

Dynamic design for ECU2

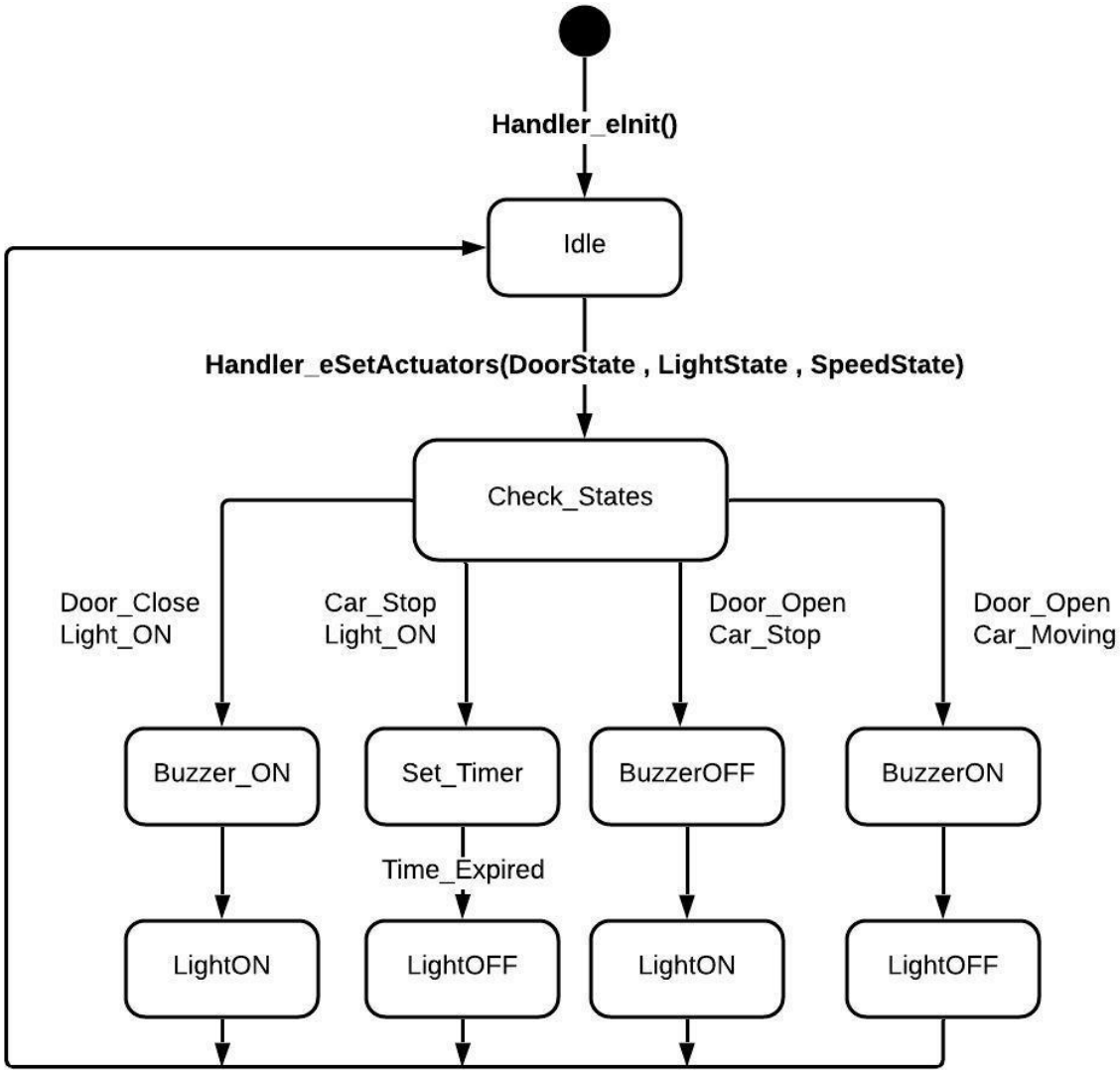
1- Folder Structure



2- State machine



Actuators_Handler



3 - Sequence diagram