Bassant Ahmed Mohamed

Automotive Door Control System Design

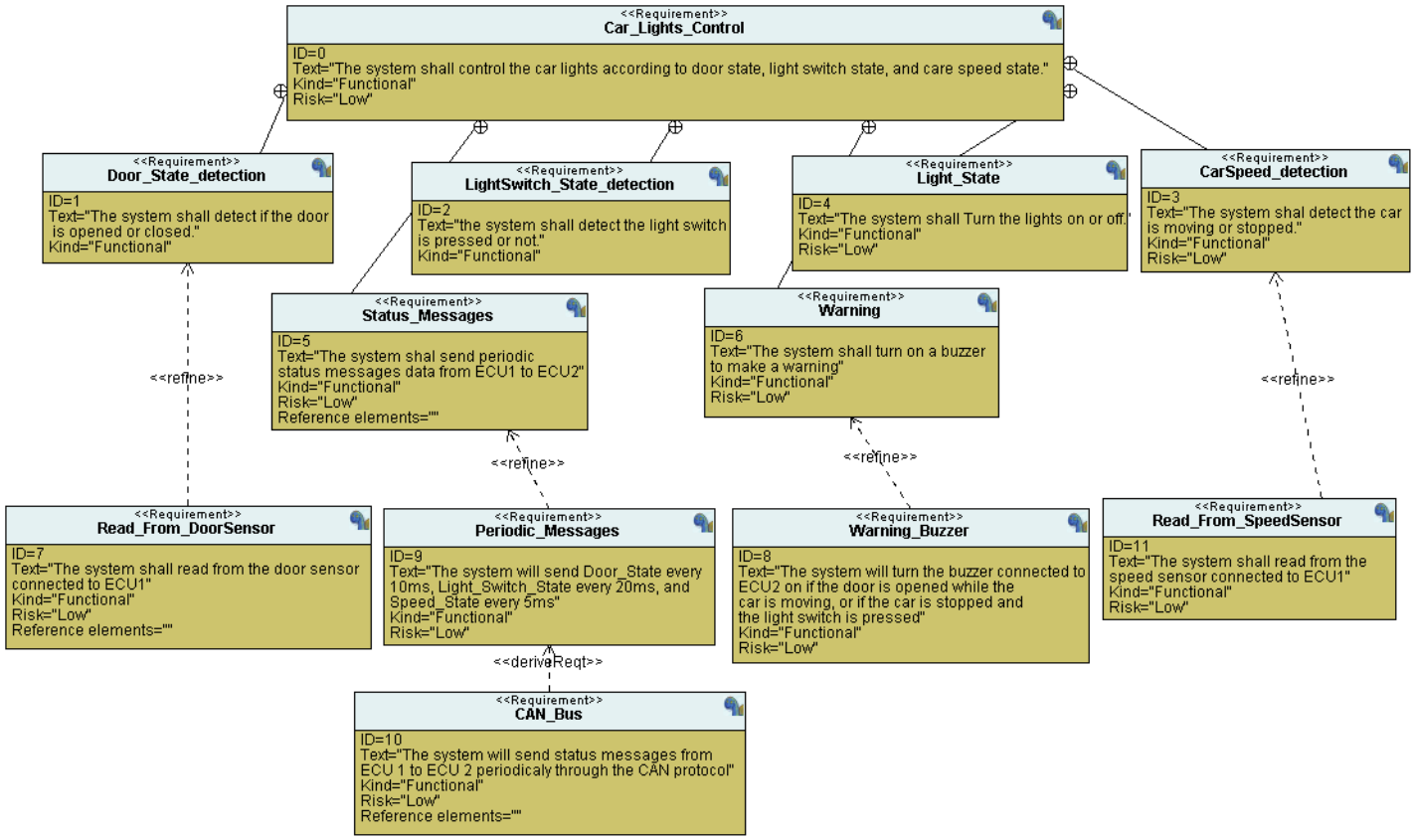
(Static Design)

Submitted to:

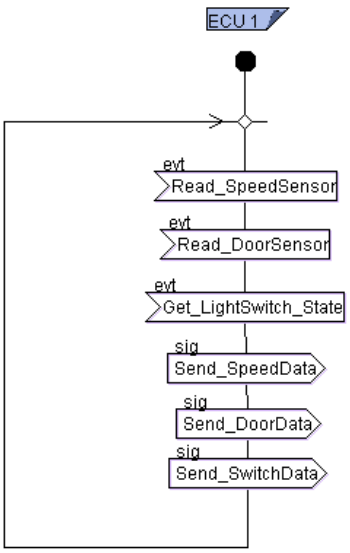
EgFwd Advanced Embedded System Track

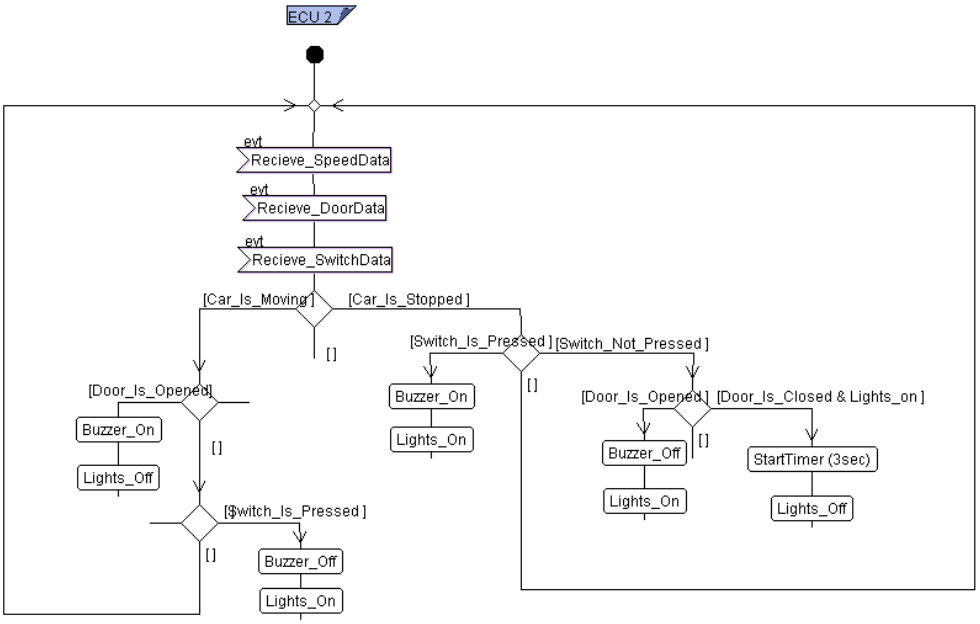
1. Reading Project Requirements

1.1 Requirements



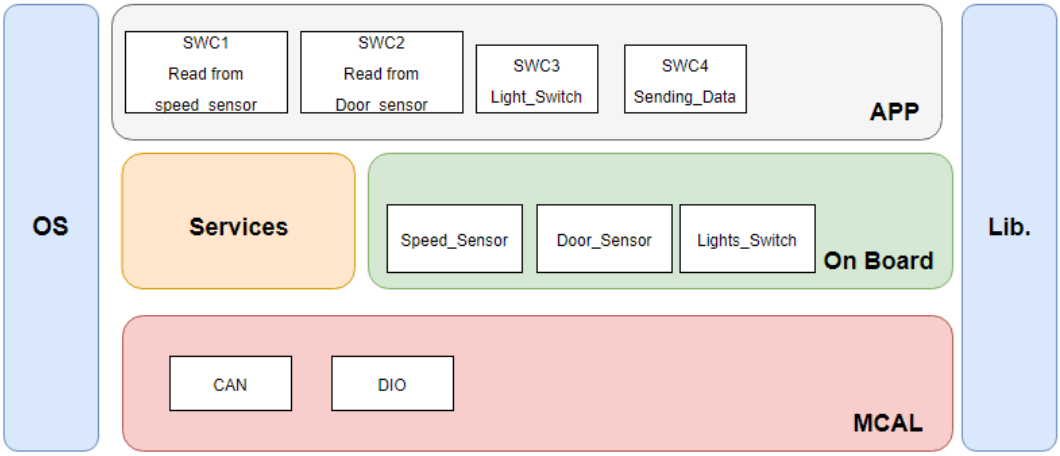
1.2 Block Diagram





2. Static Design Analysis

2.1 ECU1 Layered Architecture, Components, and Modules



2.2 ECU 1 APIs and Typedefs

2.2.1 Speed_Sensor

```
//Speed_Sensor.h

typedef struct
{
    Dio_ChannelType PinType;
}SpeedSensor_ConfigType;


void SpeedSensor_Init(void);
void Set_SpeedVal(int SpeedSensorVal);
```

```
//Speed_Sensor.c

int SpeedSensorVal = 0;
SpeedSensor_ConfigType SpeedSensor_Cfg ={//Pin_Type
                                         };

void SpeedSensor_Init(){}
void Set_SpeedVal(int SpeedSensorVal){}
```

Function name:	SpeedSensor_Init	
Arguments:	Inputs	non
	Outputs	non
	Input/Output	non
Return:	non	
Description:	Initialize the sensor module	

Function name:	Set_SpeedVal		
Arguments:	Inputs	SpeedSensorVal	int
		Description: keeps the speed value	
	Outputs	non	
	Input/Output	non	
Return:	non		
Description:	Getting the current speed value		

Name:	SpeedSensor_ConfigType	
Type:	Structure	
Elements:	PinType	Specifies the pin and the port used
Description:	A structure to configure the speed sensor elements	

2.2.2 Door_Sensor

```
//Door_Sensor.h

typedef struct
{
    Dio_ChannelType PinType;
}DoorSensor_ConfigType;

typedef Enum {
    Door_is_Opened,
    Door_is_Closed
}DoorState_Type;

void DoorSensor_Init(void);
DoorState_Type Set_DoorState(void);
```

```
//Door_Sensor.c

DoorState_Type DoorState = 0;
DoorSensor_ConfigType DoorSensor_Cfg ={//Pin_Type
                                       };

void DoorSensor_Init(void){}
DoorState_Type Set_DoorState(void){}
```

Function name:	DoorSensor_Init	
Arguments:	Inputs	non
	Outputs	non
	Input/Output	non
Return:	non	
Description:	Initialize the Door sensor module	

Function name:	Set_DoorState		
Arguments:	Inputs	non	
	Outputs	non	
	Input/Output	non	
Return:	DoorState_Type	Door_Is_Opened	0
		Door_Is_Closed	1
Description:	Get the state of the door		

Name:	DoorSensor_ConfigType	
Type:	Structure	
Elements:	PinType	Specifies the pin and the port used
Description:	A structure to configure the Door sensor elements	

Name:	DoorState_Type	
Type:	Enumeration	
Range:	Door_Is_Opened	0
	Door_Is_Closed	1
Description:	An Enum to specify the state of the door	

2.2.3 Lights Switch

```
//Lights_Switch.h

typedef struct
{
    Dio_ChannelType     PinType;
}Switch_ConfigType;

typedef Enum {
    Switch_Not_pressed,
    Switch_Is_pressed
}SwitchState_Type;

void Switch_Init(void);
SwitchState_Type Set_SwitchState(void);
```

Function name:	Switch_Init	
Arguments:	Inputs	non
	Outputs	non
	Input/Output	non
Return:	non	
Description:	Initialize the Lights_Switch module	

Function name:	Set_SwitchState		
Arguments:	Inputs	non	
	Outputs	non	
	Input/Output	non	
Return:	SwitchState_Type	Switch_Not_pressed	0
		Switch_is_pressed	1
Description:	Get the state of the Switch		

Name:	Switch_ConfigType	
Type:	Structure	
Elements:	PinType	Specifies the pin and the port used
Description:	A structure to configure the Light Switch elements	

Name:	SwitchState_Type	
Type:	Enumeration	
Range:	Switch_Not_Pressed	0
	Switch_Is_Pressed	1
Description:	An Enum to specify the state of the Switch	

2.2.4 DIO

```
//DIO.h

typedef enum
{
    Dio_PortA_PA0,
    Dio_PortA_PA1,
    Dio_PortA_PA2,
    Dio_PortA_PA3,
    Dio_PortA_PA4,
    Dio_PortA_PA5,
    Dio_PortA_PA6,
    Dio_PortA_PA7,
        .
        .
        .
}Dio_ChannelType;

typedef enum
{
    Low,
    High
}Dio_LevelType;

typedef struct
{
    Dio_ChannelType PortPinType;
    uint8           PortPinMode ;
    Dio_LevelType   PortPinLevelValue ;
    uint8           PortPinDirection ;
    uint8           PortPinInternalAttach ;
}Port_ConfigType;

void port_init (void);
Dio_LevelType Dio_ReadChannel (Dio_ChannelType ChannelId);
Dio_WriteChannel (Dio_ChannelType ChannelId,Dio_LevelType Level );
void Dio_FlipChannel (Dio_ChannelType ChannelId);
```

Function name:	Port_Init	
Arguments:	Inputs	non
	Outputs	non
	Input/Output	non
Return:	non	
Description:	Initialize the DIO module	

Function name:	Dio_ReadChannel		
Arguments:	Inputs	ChannelID	Dio_ChannelType to set the port and pin used
	Outputs	non	
	Input/Output	non	
Return:	Dio_LevelType	Low	0
		High	1
Description:	Reading specific pin		

Function name:	Dio_WriteChannel		
Arguments:	Inputs	ChannelID	Dio_ChannelType to set the port and pin used
		Level	Entering the level want to write on the pin
	Outputs	non	
	Input/Output	non	
Return:	non		
Description:	Write on specific pin		

Function name:	Dio_FlipChannel		
Arguments:	Inputs	ChannelID	Dio_ChannelType to set the port and pin used
	Outputs	non	
	Input/Output	non	
Return:	non		
Description:	Toggling specific pin		

Name:	Dio_LevelType	
Type:	Enumeration	
Range:	Low	0
	High	1
Description:	An Enum to specify the level of the Dio pin	

Name:	Port_ConfigType	
Type:	Structure	
Elements:	PortPinType	Specifies the pin and the port used
	PortPinMode	Specifies the pin mode GPIO or AF

	PortPinLevelValue	Specifies the pin level
	PortPinDirection	Specifies the pin direction input or output
	PortPinInternalAttach	Specifies the pin attachment PU or PD or OD
Description:	A structure to configure the Dio Pin elements	

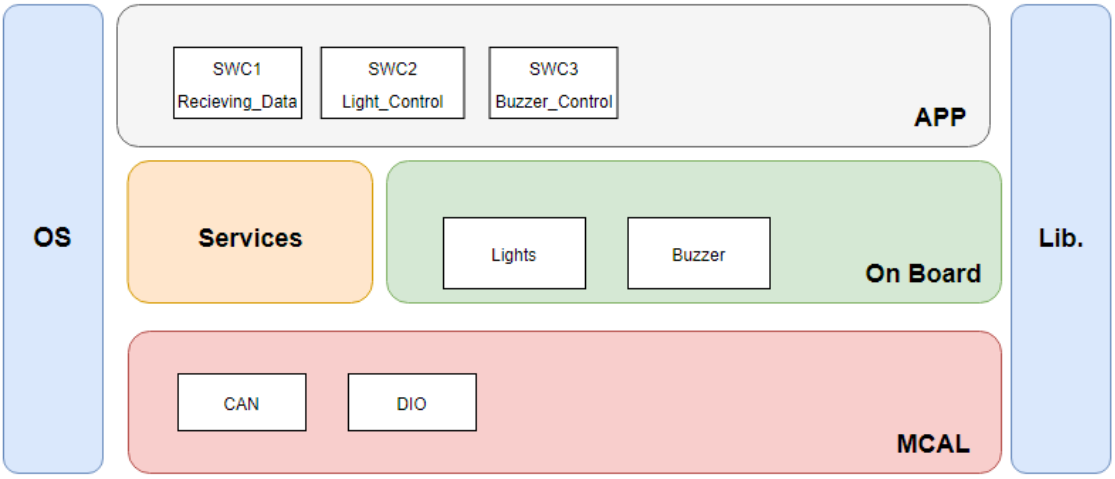
2.2.5 CAN

Function name:	CAN_Init	
Arguments:	Inputs	non
	Outputs	non
	Input/Output	non
Return:	non	
Description:	Initialize the CAN module	

Function name:	CAN_Tx		
Arguments:	Inputs	Data	Uint32 Data to be sent
	Outputs	non	
	Input/Output	non	
Return:	non		
Description:	Sending data		

Function name:	CAN_Rx		
Arguments:	Inputs	non	
	Outputs	non	
	Input/Output	non	
Return:	Std_ReturnType	E_OK	0
		E_NOK	1
Description:	Receiving data		

2.3 ECU 2 Layered Architecture, Components, and Modules



2.4 ECU 2 APIs and Typedefs

2.4.1 Lights

```
//Lights.h

typedef struct
{
    Light_Type      Light;
    Dio_ChannelType     PinType;
}Lights_ConfigType;

typedef enum
{
    R_Light,
    L_Light
}Light_Type;

void Lights_Init(void);
void Lights_On(void);
void Lights_Off(void);
```

Function name:	Lights_Init	
Arguments:	Inputs	non
	Outputs	non
	Input/Output	non
Return:	non	
Description:	Initialize the Lights module	

Function name:	Lights_On	
Arguments:	Inputs	non
	Outputs	non
	Input/Output	non
Return:	non	
Description:	Turn on the Lights	

Function name:	Lights_Off	
Arguments:	Inputs	non
	Outputs	non
	Input/Output	non
Return:	non	
Description:	Turn off the Lights	

Name:	Lights_ConfigType	
Type:	Structure	
Elements:	PinType	Specifies the pin and the port used
	Light	Light_Type to specify witch light to configure
Description:	A structure to configure the lights elements	

Name:	Light_Type	
Type:	Enumeration	
Range:	R_Light	0
	L_Light	1
Description:	An Enum to specify the Light direction	

2.4.2 Buzzer

```
//Buzzer.h

typedef struct
{
    Dio_ChannelType     PinType;
}Buzzer_ConfigType;

void Buzz_Init(void);
void Buzz_On(void);
void Buzz_Off(void);
```

Function name:	Buzz_Init	
Arguments:	Inputs	non
	Outputs	non
	Input/Output	non
Return:	non	
Description:	Initialize the Buzzer module	

Function name:	Buzz_On	
Arguments:	Inputs	non
	Outputs	non
	Input/Output	non
Return:	non	
Description:	Turn on the buzzer	

Function name:	Buzz_Off	
Arguments:	Inputs	non
	Outputs	non
	Input/Output	non
Return:	non	
Description:	Turn off the buzzer	

Name:	Buzzer_ConfigType	
Type:	Structure	
Elements:	PinType	Specifies the pin and the port used
Description:	A structure to configure the Buzzer elements	

2.4.3 DIO

```
//DIO.h

typedef enum
{
    Dio_PortA_PA0,
    Dio_PortA_PA1,
    Dio_PortA_PA2,
    Dio_PortA_PA3,
    Dio_PortA_PA4,
    Dio_PortA_PA5,
    Dio_PortA_PA6,
    Dio_PortA_PA7,
        .
        .
        .
}Dio_ChannelType;

typedef enum
{
    Low,
    High
}Dio_LevelType;

typedef struct
{
    Dio_ChannelType PortPinType;
    uint8           PortPinMode ;
    Dio_LevelType   PortPinLevelValue ;
    uint8           PortPinDirection ;
    uint8           PortPinInternalAttach ;
}Port_ConfigType;

void port_init (void);
Dio_LevelType Dio_ReadChannel (Dio_ChannelType ChannelId);
Dio_WriteChannel (Dio_ChannelType ChannelId,Dio_LevelType Level );
void Dio_FlipChannel (Dio_ChannelType ChannelId);
```

Function name:	Port_Init	
Arguments:	Inputs	non
	Outputs	non
	Input/Output	non
Return:	non	
Description:	Initialize the DIO module	

Function name:	Dio_ReadChannel		
Arguments:	Inputs	ChannelID	Dio_ChannelType to set the port and pin used
	Outputs	non	
	Input/Output	non	
Return:	Dio_LevelType	Low	0
		High	1
Description:	Reading specific pin		

Function name:	Dio_WriteChannel		
Arguments:	Inputs	ChannelID	Dio_ChannelType to set the port and pin used
		Level	Entering the level want to write on the pin
	Outputs	non	
	Input/Output	non	
Return:	non		
Description:	Write on specific pin		

Function name:	Dio_FlipChannel		
Arguments:	Inputs	ChannelID	Dio_ChannelType to set the port and pin used
	Outputs	non	
	Input/Output	non	
Return:	non		
Description:	Toggling specific pin		

Name:	Dio_LevelType	
Type:	Enumeration	
Range:	Low	0
	High	1
Description:	An Enum to specify the level of the Dio pin	

Name:	Port_ConfigType	
Type:	Structure	
Elements:	PortPinType	Specifies the pin and the port used
	PortPinMode	Specifies the pin mode GPIO or AF

	PortPinLevelValue	Specifies the pin level
	PortPinDirection	Specifies the pin direction input or output
	PortPinInternalAttach	Specifies the pin attachment PU or PD or OD
Description:	A structure to configure the Dio Pin elements	

2.4.4 CAN

Function name:	CAN_Init		
Arguments:	Inputs	non	
	Outputs	non	
	Input/Output	non	
Return:	non		
Description:	Initialize the CAN module		

Function name:	CAN_Tx		
Arguments:	Inputs	Data	Uint32 Data to be sent
	Outputs	non	
	Input/Output	non	
Return:	non		
Description:	Sending data		

Function name:	CAN_Rx		
Arguments:	Inputs	non	
	Outputs	non	
	Input/Output	non	
Return:	Std_ReturnType	E_OK	0
		E_NOK	1
Description:	Receiving data		