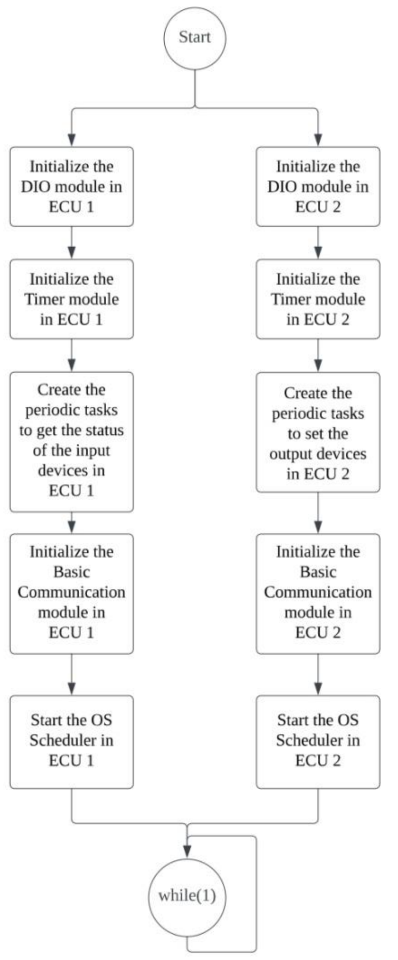
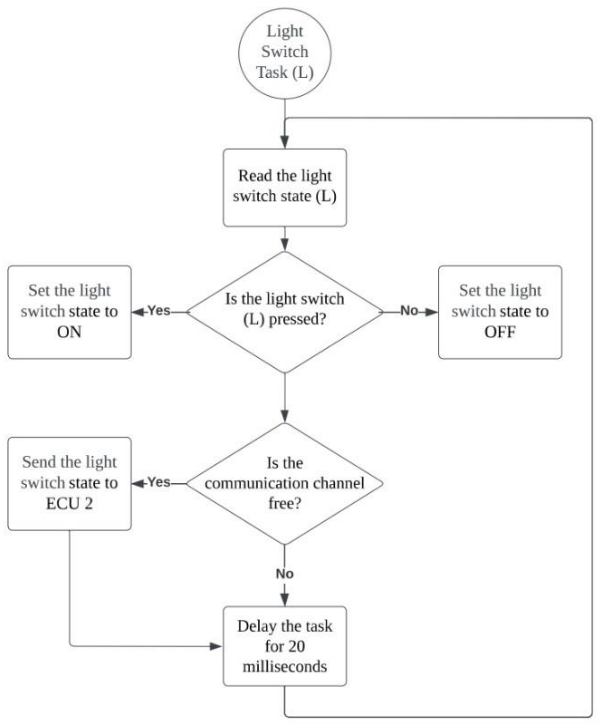
<u>Static Design Analysis of the Car System</u>

1. *<u>Project Requirements:</u>*

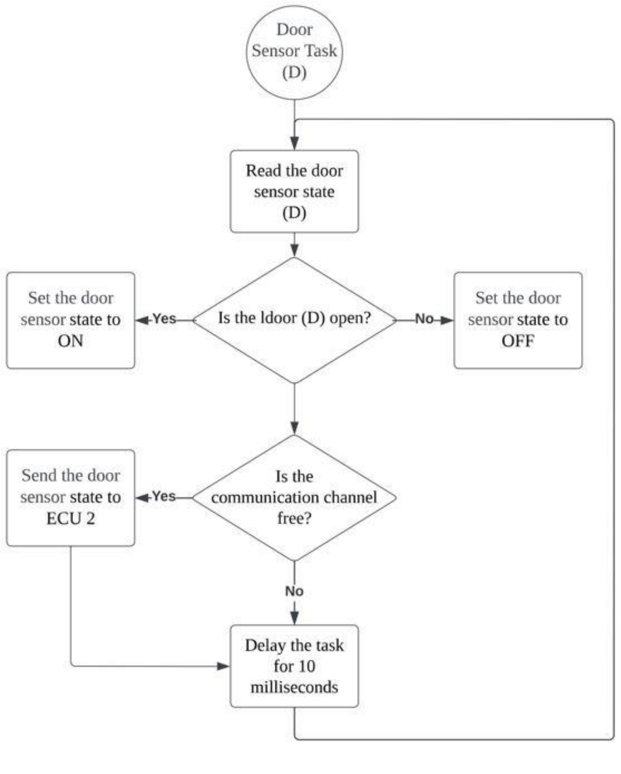
- *"You should draw and deliver the system schematic (Block Diagram) according to your requirements understanding, a screenshot is required."*



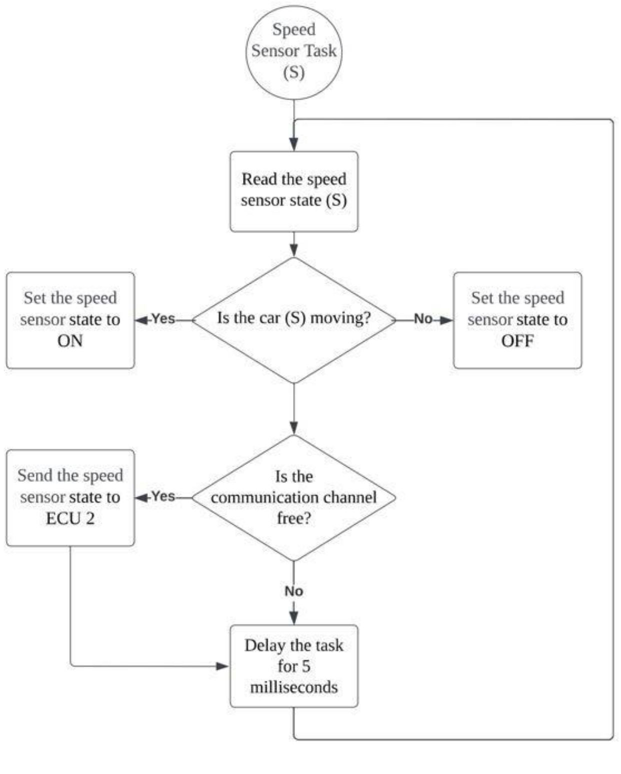
Start
Initialize the DIO module in ECU 1
Initialize the Timer module in ECU 1
Create the periodic tasks to get the status of the input devices in ECU 1
Initialize the Basic Communication module in ECU 1
Start the OS Scheduler in ECU 1
Initialize the DIO module in ECU 2
Initialize the Timer module in ECU 2
Create the periodic tasks to set the output devices in ECU 2
Initialize the Basic Communication module in ECU 2
Start the OS Scheduler in ECU 2
while(1)



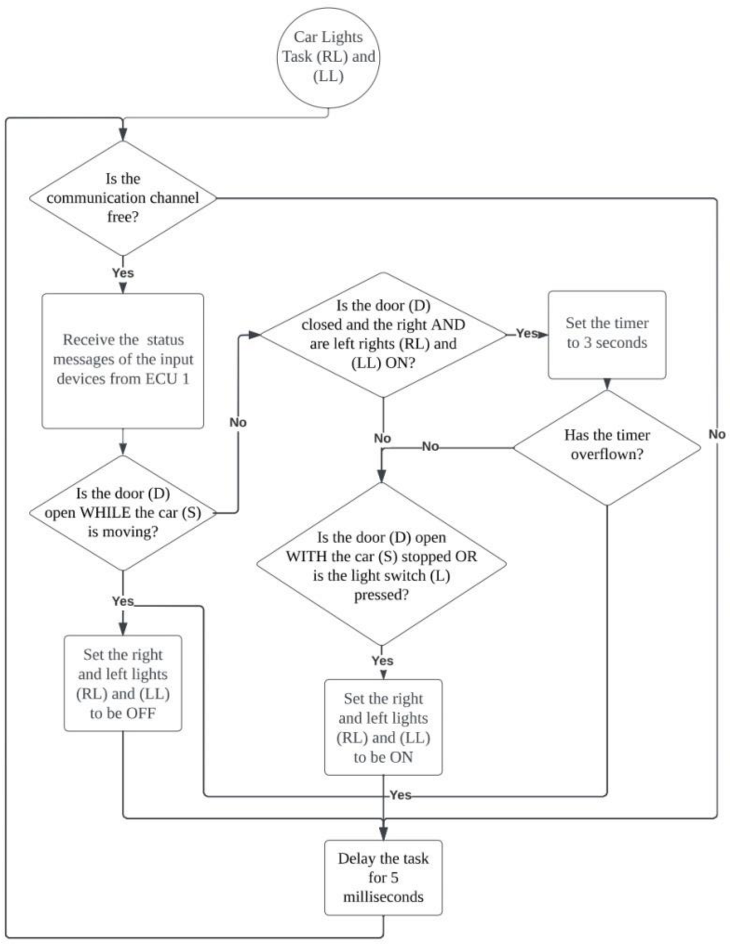
Light Switch Task (L)
Read the light switch state (L)
Is the light switch (L) pressed?
Yes
Set the light switch state to ON
No
Set the light switch state to OFF
Is the communication channel free?
Yes
Send the light switch state to ECU 2
No
Delay the task for 20 milliseconds



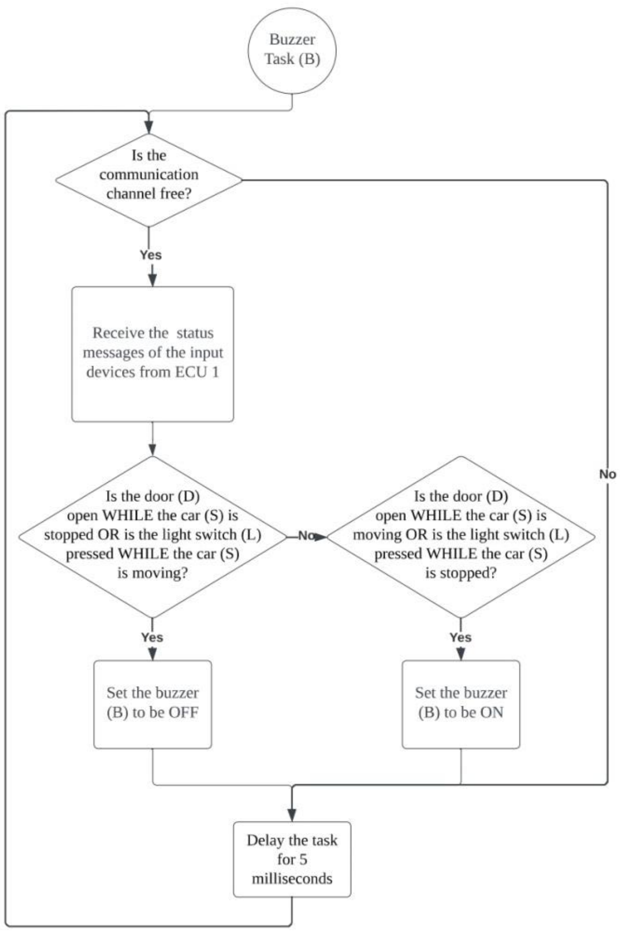
Door Sensor Task (D)
Read the door sensor state (D)
Is the ldoor (D) open?
Yes
Set the door sensor state to ON
No
Set the door sensor state to OFF
Is the communication channel free?
Yes
Send the door sensor state to ECU 2
No
Delay the task for 10 milliseconds



Speed Sensor Task (S)
Read the speed sensor state (S)
Is the car (S) moving?
Yes
Set the speed sensor state to ON
No
Set the speed sensor state to OFF
Is the communication channel free?
Yes
Send the speed sensor state to ECU 2
No
Delay the task for 5 milliseconds



Car Lights Task (RL) and (LL)
Is the communication channel free?
Yes
No
Receive the status messages of the input devices from ECU 1
Is the door (D) open WHILE the car (S) is moving?
Yes
No
Set the right and left lights (RL) and (LL) to be OFF
Is the door (D) closed and the right AND are left rights (RL) and (LL) ON?
Yes
No
Set the timer to 3 seconds
Has the timer overflown?
No
Yes
Is the door (D) open WITH the car (S) stopped OR is the light switch (L) pressed?
Yes
Set the right and left lights (RL) and (LL) to be ON
Delay the task for 5 milliseconds

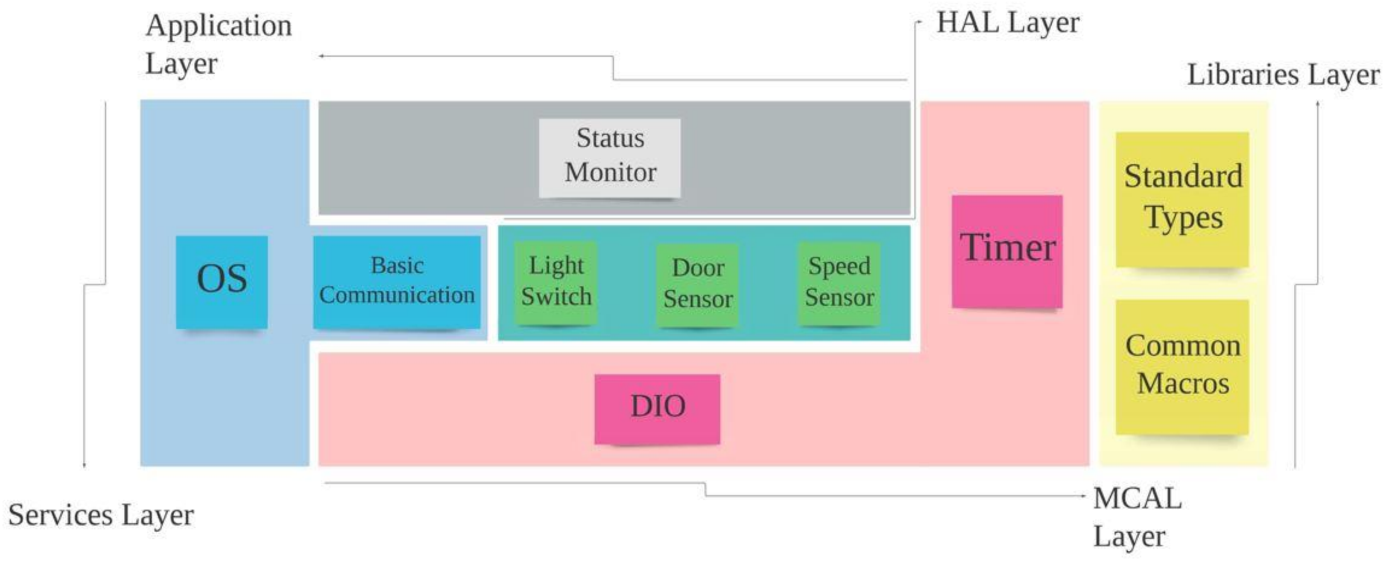


Buzzer Task (B)
Is the communication channel free?
Yes
No
Receive the status messages of the input devices from ECU 1
Is the door (D) open WHILE the car (S) is stopped OR is the light switch (L) pressed WHILE the car (S) is moving?
No
Yes
Is the door (D) open WHILE the car (S) is moving OR is the light switch (L) pressed WHILE the car (S) is stopped?
Yes
Set the buzzer (B) to be OFF
Set the buzzer (B) to be ON
Delay the task for 5 milliseconds

2. *Static Design Analysis:*

For ECU 1:

- *"Make the layered architecture."*



- *“Specify ECU components and modules.”*

    - “**DIO_Init**”, “**DIO_Read**”, “**DIO_Write**” components → “**DIO**” module
    - “**Timer_Init**”, “**Timer_Start**”, “**Timer_Stop**” components → “**Timer**” module
    - “**Task_Create**”, “**Scheduler_Start**” → “**OS**” module
    - “**CAN_Init**”, “**CAN_Send**”, “**CAN_Receive**”, “**CAN_isFree**” → “**Basic Communication**” module
    - “**LightSwitch_Read**” → “**Light Switch**” module
    - “**SpeedSensor_Read**” → “**Speed Sensor**” module
    - “**DoorSensor_Read**” → “**Door Sensor**” module
    - “**LightSwitch_isPressed**”, “**Car_isMoving**”, “**Door_isOpen**” → “**Status Monitor**” module
    - “**DIO_StdType**”, “**Timer_StdType**”, “**CAN_StdType**”, “**LightSwitch_StdType**”, “**SpeedSensor_StdType**”, “**DoorSensor_StdType**” → “**Standard Types**” module
    - “**Platform_Types**”, “**MCU_HW**”, “**Compiler_Config**” → “**Common Macros**” module

- *“Provide full detailed APIs for each module as well as a detailed description for the used typedefs.”*

APIs

Service Name:	DIO_Init
Syntax:	void DIO_Init(const DIO_ConfigType* DIOConfig)
Sync/Async:	Synchronous
Reentrancy:	Reentrant
Parameters (in):	DIOConfig | a pointer to DIO configurations
Parameters (out):	None
Return Value:	None
Description:	This function initializes the DIO module

Service Name:	DIO_Read
Syntax:	DIO_LevelType DIO_Read(DIO_PortType DIOPort, DIO_PinType DIOPin)
Sync/Async:	Synchronous
Reentrancy:	Reentrant
Parameters (in):	DIOPort | the port of the DIO to be read / DIOPin | the pin of the DIO to be read
Parameters (out):	DIOLevel | the level of the DIO read
Return Value:	DIO_LevelType
Description:	This function reads the level of the DIO requested

Service Name:	DIO_Write
Syntax:	void DIO_Write(DIO_PortType DIOPort, DIO_PinType DIOPin, DIO_LevelType DIOLevel)
Sync/Async:	Synchronous
Reentrancy:	Non-reentrant
Parameters (in):	DIOPort | the port of the DIO to be read / DIOPin | the pin of the DIO to be read / DIOLevel | the level of the DIO to write to the DIO
Parameters (out):	None
Return Value:	None

Description:	This function writes the level requested to the DIO

Service Name:	Timer_Init
Syntax:	void Timer_Init(const Timer_ConfigType* TimerConfig)
Sync/Async:	Synchronous
Reentrancy:	Reentrant
Parameters (in):	TimerConfig | a pointer to the timer configuration
Parameters (out):	None
Return Value:	None
Description:	This function initializes the timer module

Service Name:	Timer_Start
Syntax:	void Timer_Start(Timer_LevelType TimerValue)
Sync/Async:	Synchronous
Reentrancy:	Reentrant
Parameters (in):	TimerValue | the value of the timer count
Parameters (out):	None
Return Value:	None
Description:	This function starts the timer/counter with the value requested

Service Name:	Timer_Stop
Syntax:	void Timer_Stop(void)
Sync/Async:	Synchronous
Reentrancy:	Reentrant
Parameters (in):	None
Parameters (out):	None
Return Value:	None
Description:	This function stops the timer/counter

Service Name:	Task_Create
Syntax:	void Task_Create(const Task_ConfigType* TaskConfig)
Sync/Async:	Synchronous
Reentrancy:	Reentrant
Parameters (in):	TaskConfig | a pointer to the task configurations
Parameters (out):	None
Return Value:	None
Description:	This function initializes the OS task

Service Name:	Scheduler_Start
Syntax:	void Scheduler_Start(void)
Sync/Async:	Synchronous
Reentrancy:	Reentrant
Parameters (in):	None
Parameters (out):	None
Return Value:	None
Description:	This function starts the OS scheduler to select the task to run

Service Name:	CAN_Init
Syntax:	void CAN_Init(const CAN_ConfigType* CANConfig)
Sync/Async:	Synchronous
Reentrancy:	Reentrant
Parameters (in):	CANConfig | a pointer to the CAN channel configurations

Parameters (out):	None
Return Value:	None
Description:	This function initializes the CAN channel of the basic communication module

Service Name:	CAN_Send
Syntax:	void CAN_Send(CAN_BufferType* CANBuffer, CAN_LevelType CANBufferLength)
Sync/Async:	Synchronous
Reentrancy:	Reentrant
Parameters (in):	CANBuffer | a pointer to the buffer to be sent through the CAN channel / CANBufferLength | the length of the buffer to be sent through the CAN channel
Parameters (out):	None
Return Value:	None
Description:	This function sends the buffer with the specified length through the CAN channel

Service Name:	CAN_Receive
Syntax:	void CAN_Receive(CAN_BufferType* CANBuffer)
Sync/Async:	Synchronous
Reentrancy:	Reentrant
Parameters (in):	CANBuffer | a pointer to the buffer to be received through the CAN channel
Parameters (out):	None
Return Value:	None
Description:	This function receives the buffer through the CAN channel

Service Name:	LightSwitch_Read
Syntax:	LightSwitch_LevelType LightSwitch_Read(LightSwitch_PortType LightSwitchPort, LightSwitch_PinType LightSwitchPin)
Sync/Async:	Synchronous
Reentrancy:	Reentrant
Parameters (in):	LightSwitchPort | the port of the light switch to be read / LightSwitchPin | the pin of the light switch to be read
Parameters (out):	LightSwitchLevel | the level of the light switch read
Return Value:	LightSwitch_LevelType
Description:	This function reads the level of the light switch requested to see if pressed or not

Service Name:	SpeedSensor_Read
Syntax:	SpeedSensor_LevelType SpeedSensor_Read (SpeedSensor_PortType SpeedSensorPort, SpeedSensor_PinType SpeedSensorPin)
Sync/Async:	Synchronous
Reentrancy:	Reentrant
Parameters (in):	SpeedSensorPort | the port of the speed sensor to be read / SpeedSensorPin | the pin of the speed sensor to be read
Parameters (out):	SpeedSensorLevel | the level of the speed sensor read
Return Value:	SpeedSensor_LevelType
Description:	This function reads the level of the speed sensor requested to see if the car is moving or not

Service Name:	DoorSensor_Read
Syntax:	DoorSensor_LevelType DoorSensor_Read(DoorSensor_PortType DoorSensorPort, DoorSensor_PinType DoorSensorPin)
Sync/Async:	Synchronous
Reentrancy:	Reentrant
Parameters (in):	DoorSensorPort | the port of the door sensor to be read / DoorSensorPin | the pin of the door sensor to be read
Parameters (out):	DoorSensorLevel | the level of the door sensor read
Return Value:	DoorSensor_LevelType

Description:	This function reads the level of the door sensor requested to see if open or not

Service Name:	LightSwitch_isPressed
Syntax:	LightSwitch_LevelType LightSwitch_isPressed(void)
Sync/Async:	Synchronous
Reentrancy:	Reentrant
Parameters (in):	None
Parameters (out):	LightSwitchLevel | the level of the light switch read
Return Value:	LightSwitch_LevelType
Description:	This function reads the level of the light switch configured to see if pressed or not

Service Name:	Car_isMoving
Syntax:	SpeedSensor_LevelType Car_isMoving(void)
Sync/Async:	Synchronous
Reentrancy:	Reentrant
Parameters (in):	None
Parameters (out):	SpeedSensorLevel | the level of the speed sensor read
Return Value:	SpeedSensor_LevelType
Description:	This function reads the level of the speed sensor configured to see if the car is moving or not

Service Name:	Door_isOpen
Syntax:	DoorSensor_LevelType Door_isOpen(void)
Sync/Async:	Synchronous
Reentrancy:	Reentrant
Parameters (in):	None
Parameters (out):	DoorSensorLevel | the level of the door sensor read
Return Value:	DoorSensor_LevelType
Description:	This function reads the level of the door sensor configured to see if the door is open or not

Typedefs

Name:	DIO_ConfigType
Type:	Structure
Range:	Implementation Specific
Description:	The data structure containing the overall initialization data for the DIO module
Available via:	DIO.h

Name:	DIO_LevelType
Type:	Char
Range:	0 to 1
Description:	The boolean value indicating if the DIO pin is LOW (0) or HIGH (1)
Available via:	DIO.h

Name:	DIO_PortType
Type:	Char
Range:	0 to Number of Ports
Description:	The number of the port containing the DIO pin stored as an enum member
Available via:	DIO.h

Name:	DIO_PinType
Type:	Char
Range:	0 to 7

Description:	The number of the DIO pin specified stored as an enum member
Available via:	DIO.h

Name:	Timer_ConfigType
Type:	Structure
Range:	Implementation Specific
Description:	The data structure containing the overall initialization data for the timer module
Available via:	Timer.h

Name:	Timer_LevelType
Type:	Int
Range:	0 to 65,535
Description:	The count value given to the timer/counter register
Available via:	Timer.h

Name:	Task_ConfigType
Type:	Structure
Range:	Implementation Specific
Description:	The data structure containing the overall initialization data for the tasks
Available via:	OS.h

Name:	CAN_ConfigType
Type:	Structure
Range:	Implementation Specific
Description:	The data structure containing the overall initialization data for the CAN module
Available via:	CAN.h

Name:	CAN_BufferType
Type:	Char
Range:	0 to 255
Description:	The CAN frame holding the message sent/received
Available via:	CAN.h

Name:	CAN_LevelType
Type:	Char
Range:	0 to 255
Description:	The length of the message to store in the CAN frame
Available via:	CAN.h

Name:	LightSwitch_LevelType
Type:	Char
Range:	0 to 1
Description:	The boolean value indicating if the light switch is RELEASED (0) or PRESSED (1)
Available via:	LightSwitch.h

Name:	LightSwitch_PortType
Type:	Char
Range:	0 to Number of Ports
Description:	The number of the port containing the light switch pin based on the DIO enum numbers
Available via:	LightSwitch.h

Name:	LightSwitch_PinType
Type:	Char

Range:	0 to 7
Description:	The number of the light switch pin based on the DIO enum members
Available via:	LightSwitch.h

Name:	DoorSensor_LevelType
Type:	Char
Range:	0 to 1
Description:	The boolean value indicating if the door is CLOSED (0) or OPEN (1)
Available via:	DoorSensor.h

Name:	DoorSensor_PortType
Type:	Char
Range:	0 to Number of Ports
Description:	The number of the port containing the door sensor pin based on the DIO enum numbers
Available via:	DoorSensor.h

Name:	DoorSensor_PinType
Type:	Char
Range:	0 to 7
Description:	The number of the door sensor pin based on the DIO enum members
Available via:	DoorSensor.h

Name:	SpeedSensor_LevelType
Type:	Char
Range:	0 to 1
Description:	The boolean value indicating if the car is STOPPED (0) or MOVING (1)
Available via:	SpeedSensor.h

Name:	SpeedSensor_PortType
Type:	Char
Range:	0 to Number of Ports
Description:	The number of the port containing the speed sensor pin based on the DIO enum numbers
Available via:	SpeedSensor.h

Name:	SpeedSensor_PinType
Type:	Char
Range:	0 to 7
Description:	The number of the speed sensor pin based on the DIO enum members
Available via:	SpeedSensor.h

- *“Prepare your folder structure according to the previous points.”*

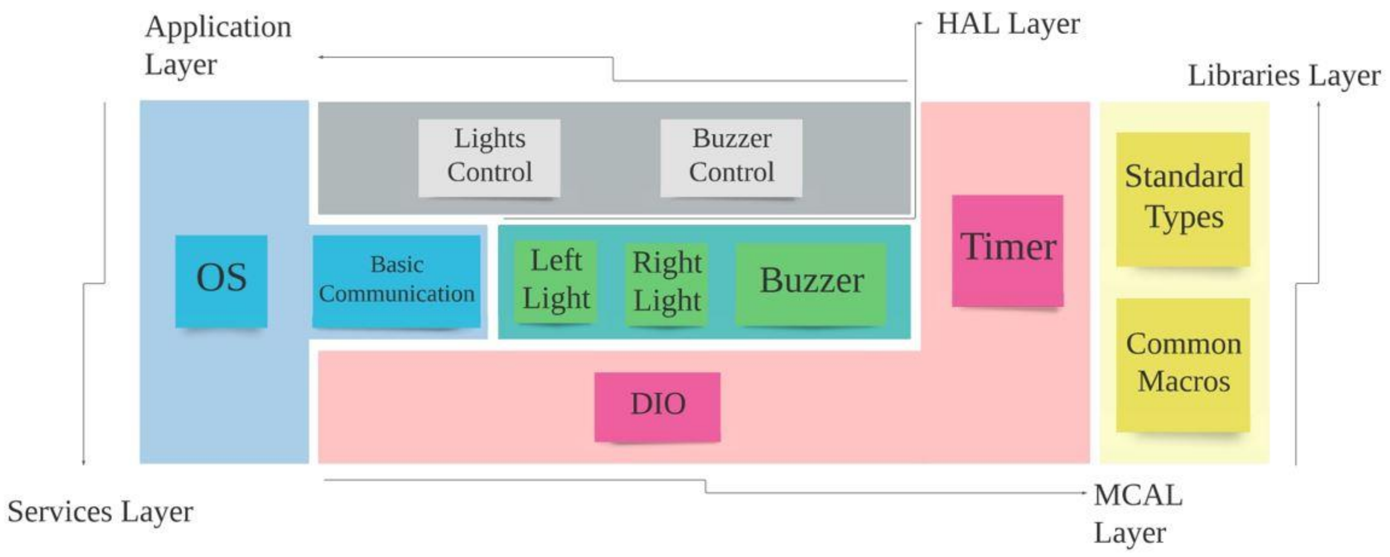
.

```
├── MCAL/
│   ├── Configuration/
│   │   ├── inc/
│   │   │   ├── DIO_Cfg.h
│   │   │   ├── DIO_Lcfg.h
│   │   │   ├── Timer_Cfg.h
│   │   │   └── Timer_Lcfg.h
│   │   └── src/
│   │       ├── DIO_Lcfg.c
│   │       └── Timer_Lcfg.c
│   │
│   ├── DIO/
│   │   ├── inc/
│   │   │   └── DIO.h
│   │   └── src/
│   │       └── DIO.c
│   └── Timer/
│       ├── inc/
│       │   └── Timer.h
│       └── src/
│           └── Timer.c
├── HAL/
│   ├── Configuration/
│   │   ├── inc/
│   │   │   ├── LightSwitch_Cfg.h
│   │   │   ├── LightSwitch_Lcfg.h
│   │   │   ├── SpeedSensor_Cfg.h
│   │   │   ├── SpeedSensor_Lcfg.h
│   │   │   ├── DoorSensor_Cfg.h
│   │   │   └── DoorSensor_Lcfg.h
│   │   └── src/
│   │       ├── LightSwitch_Lcfg.c
│   │       ├── SpeedSensor_Lcfg.c
│   │       └── DoorSensor_Lcfg.c
│   ├── LightSwitch/
│   │   ├── inc/
│   │   │   └── LightSwitch.h
│   │   └── src/
│   │       └── LightSwitch.c
│   ├── SpeedSensor/
│   │   ├── inc/
│   │   │   └── SpeedSensor.h
│   │   └── src/
│   │       └── SpeedSensor.c
│   └── DoorSensor/
│       ├── inc/
│       │   └── DoorSensor.h
│       └── src/
│           └── DoorSensor.c
├── Service/
│   ├── Configuration/
│   │   ├── inc/
│   │   │   ├── OS_Cfg.h
│   │   │   ├── OS_Lcfg.h
│   │   │   ├── CAN_Cfg.h
│   │   │   └── CAN_Lcfg.h
│   │   └── src/
│   │       ├── OS_Lcfg.c
│   │       └── CAN_Lcfg.c
│   ├── OS/
│   │   ├── inc/
│   │   │   └── OS.h
│   │   └── src/
│   │       └── OS.c
│   └── CAN/
│       ├── inc/
│       │   └── CAN.h
│       └── src/
│           └── CAN.c
│
```

```
├── Library/
│   ├── StdTypes/
│   │   └── inc/
│   │       └── StdTypes.h
│   └── Common/
│       └── inc/
│           ├── PlatformTypes.h
│           ├── MCU_HW.h
│           └── Compiler_Cfg.h
└── Application/
    └── main.c
    └── StatusMonitor/
        ├── inc/
        │   └── StatusMonitor.h
        └── src/
            └── StatusMonitor.c
```

<u>For ECU 2:</u>

- *"Make the layered architecture."*



- *"Specify ECU components and modules."*
    - "**DIO_Init**", "**DIO_Read**", "**DIO_Write**" components → "**DIO**" module
    - "**Timer_Init**", "**Timer_Start**", "**Timer_Stop**" components → "**Timer**" module
    - "**Task_Create**", "**Scheduler_Start**" → "**OS**" module
    - "**CAN_Init**", "**CAN_Send**", "**CAN_Receive**", "**CAN_isFree**" → "**Basic Communication**" module
    - "**LeftLight_Write**" → "**Left Light**" module
    - "**RightLight_Write**" → "**Right Light**" module
    - "**Buzzer_Write**" → "**Buzzer**" module
    - "**LeftLight_TurnOn**", "**LeftLight_TurnOff**",

“**RightLight_TurnOn**”, “**RightLight_TurnOff**” → “**Lights Control**” module

- “**Buzzer_TurnOn**”, “**Buzzer_TurnOff**” → “**Buzzer Control**” module
- “**DIO_StdType**”, “**Timer_StdType**”, “**CAN_StdType**”, “**LightLevel_StdType**”, “**BuzzerLevel_StdType**” → “**Standard Types**” module
- “**Platform_Types**”, “**MCU_HW**”, “**Compiler_Config**” → “**Common Macros**” module

• *“Provide full detailed APIs for each module as well as a detailed description for the used typedefs.”*

APIs

Service Name:	DIO_Init
Syntax:	void DIO_Init(const DIO_ConfigType* DIOConfig)
Sync/Async:	Synchronous
Reentrancy:	Reentrant
Parameters (in):	DIOConfig | a pointer to DIO configurations
Parameters (out):	None
Return Value:	None
Description:	This function initializes the DIO module

Service Name:	DIO_Read
Syntax:	DIO_LevelType DIO_Read(DIO_PortType DIOPort, DIO_PinType DIOPin)
Sync/Async:	Synchronous
Reentrancy:	Reentrant
Parameters (in):	DIOPort | the port of the DIO to be read / DIOPin | the pin of the DIO to be read
Parameters (out):	DIOLevel | the level of the DIO read
Return Value:	DIO_LevelType
Description:	This function reads the level of the DIO requested

Service Name:	DIO_Write
Syntax:	void DIO_Write(DIO_PortType DIOPort, DIO_PinType DIOPin, DIO_LevelType DIOLevel)
Sync/Async:	Synchronous
Reentrancy:	Non-reentrant
Parameters (in):	DIOPort | the port of the DIO to be read / DIOPin | the pin of the DIO to be read / DIOLevel | the level of the DIO to write to the DIO
Parameters (out):	None
Return Value:	None
Description:	This function writes the level requested to the DIO

Service Name:	Timer_Init
Syntax:	void Timer_Init(const Timer_ConfigType* TimerConfig)
Sync/Async:	Synchronous
Reentrancy:	Reentrant
Parameters (in):	TimerConfig | a pointer to the timer configuration
Parameters (out):	None
Return Value:	None
Description:	This function initializes the timer module

Service Name:	Timer_Start
Syntax:	void Timer_Start(Timer_LevelType TimerValue)
Sync/Async:	Synchronous
Reentrancy:	Reentrant
Parameters (in):	TimerValue | the value of the timer count
Parameters (out):	None
Return Value:	None
Description:	This function starts the timer/counter with the value requested

Service Name:	Timer_Stop
Syntax:	void Timer_Stop(void)
Sync/Async:	Synchronous
Reentrancy:	Reentrant
Parameters (in):	None
Parameters (out):	None
Return Value:	None
Description:	This function stops the timer/counter

Service Name:	Task_Create
Syntax:	void Task_Create(const Task_ConfigType* TaskConfig)
Sync/Async:	Synchronous
Reentrancy:	Reentrant
Parameters (in):	TaskConfig | a pointer to the task configurations
Parameters (out):	None
Return Value:	None
Description:	This function initializes the OS task

Service Name:	Scheduler_Start
Syntax:	void Scheduler_Start(void)
Sync/Async:	Synchronous
Reentrancy:	Reentrant
Parameters (in):	None
Parameters (out):	None
Return Value:	None
Description:	This function starts the OS scheduler to select the task to run

Service Name:	CAN_Init
Syntax:	void CAN_Init(const CAN_ConfigType* CANConfig)
Sync/Async:	Synchronous
Reentrancy:	Reentrant
Parameters (in):	CANConfig | a pointer to the CAN channel configurations
Parameters (out):	None
Return Value:	None
Description:	This function initializes the CAN channel of the basic communication module

Service Name:	CAN_Send
Syntax:	void CAN_Send(CAN_BufferType* CANBuffer, CAN_LevelType CANBufferLength)
Sync/Async:	Synchronous
Reentrancy:	Reentrant
Parameters (in):	CANBuffer | a pointer to the buffer to be sent through the CAN channel / CANBufferLength | the length of the buffer to be sent through the CAN channel
Parameters (out):	None
Return Value:	None

Description:	This function sends the buffer with the specified length through the CAN channel

Service Name:	CAN_Receive
Syntax:	void CAN_Receive(CAN_BufferType* CANBuffer)
Sync/Async:	Synchronous
Reentrancy:	Reentrant
Parameters (in):	CANBuffer | a pointer to the buffer to be received through the CAN channel
Parameters (out):	None
Return Value:	None
Description:	This function receives the buffer through the CAN channel

Service Name:	LeftLight_Write
Syntax:	void LeftLight_Write(Light_PortType LeftLightPort, Light_PinType LeftLightPin, Light_LevelType LeftLightLevel)
Sync/Async:	Synchronous
Reentrancy:	Non-reentrant
Parameters (in):	LeftLightPort | the port of the left light to be written / LeftLightPin | the pin of the left light to be written / LeftLightLevel | the level of the left light to be written
Parameters (out):	None
Return Value:	None
Description:	This function writes the level specified to the left light

Service Name:	RightLight_Write
Syntax:	void RightLight_Write(Light_PortType RightLightPort, Light_PinType RightLightPin, Light_LevelType RightLightLevel)
Sync/Async:	Synchronous
Reentrancy:	Non-reentrant
Parameters (in):	RightLightPort | the port of the right light to be written / RightLightPin | the pin of the right light to be written / RightLightLevel | the level of the right light to be written
Parameters (out):	None
Return Value:	None
Description:	This function writes the level specified to the right light

Service Name:	Buzzer_Write
Syntax:	void Buzzer_Write(Buzzer_PortType BuzzerPort, Buzzer_PinType BuzzerPin, Buzzer_LevelType BuzzerLevel)
Sync/Async:	Synchronous
Reentrancy:	Non-reentrant
Parameters (in):	BuzzerPort | the port of the buzzer to be written / BuzzerPin | the pin of the buzzer to be written / BuzzerLevel | the level of the buzzer to be written
Parameters (out):	None
Return Value:	None
Description:	This function writes the level specified to the buzzer

Service Name:	LeftLight_TurnOn
Syntax:	void LeftLight_TurnOn(void)
Sync/Async:	Synchronous
Reentrancy:	Non-reentrant
Parameters (in):	None
Parameters (out):	None
Return Value:	None

Description:	This function turns on the left light

Service Name:	LeftLight_TurnOff
Syntax:	void LeftLight_TurnOff(void)
Sync/Async:	Synchronous
Reentrancy:	Non-reentrant
Parameters (in):	None
Parameters (out):	None
Return Value:	None
Description:	This function turns off the left light

Service Name:	RightLight_TurnOn
Syntax:	void RightLight_TurnOn(void)
Sync/Async:	Synchronous
Reentrancy:	Non-reentrant
Parameters (in):	None
Parameters (out):	None
Return Value:	None
Description:	This function turns on the right light

Service Name:	RightLight_TurnOff
Syntax:	void RightLight_TurnOff(void)
Sync/Async:	Synchronous
Reentrancy:	Non-reentrant
Parameters (in):	None
Parameters (out):	None
Return Value:	None
Description:	This function turns off the right light

Service Name:	Buzzer_TurnOn
Syntax:	void Buzzer_TurnOn(void)
Sync/Async:	Synchronous
Reentrancy:	Non-reentrant
Parameters (in):	None
Parameters (out):	None
Return Value:	None
Description:	This function turns on the buzzer

Service Name:	Buzzer_TurnOff
Syntax:	void Buzzer_TurnOff(void)
Sync/Async:	Synchronous
Reentrancy:	Non-reentrant
Parameters (in):	None
Parameters (out):	None
Return Value:	None
Description:	This function turns off the buzzer

Typedefs

Name:	DIO_ConfigType
Type:	Structure
Range:	Implementation Specific
Description:	The data structure containing the overall initialization data for the DIO module

Available via:	DIO.h

Name:	DIO_LevelType
Type:	Char
Range:	0 to 1
Description:	The boolean value indicating if the DIO pin is LOW (0) or HIGH (1)
Available via:	DIO.h

Name:	DIO_PortType
Type:	Char
Range:	0 to Number of Ports
Description:	The number of the port containing the DIO pin stored as an enum member
Available via:	DIO.h

Name:	DIO_PinType
Type:	Char
Range:	0 to 7
Description:	The number of the DIO pin specified stored as an enum member
Available via:	DIO.h

Name:	Timer_ConfigType
Type:	Structure
Range:	Implementation Specific
Description:	The data structure containing the overall initialization data for the timer module
Available via:	Timer.h

Name:	Timer_LevelType
Type:	Int
Range:	0 to 65,535
Description:	The count value given to the timer/counter register
Available via:	Timer.h

Name:	Task_ConfigType
Type:	Structure
Range:	Implementation Specific
Description:	The data structure containing the overall initialization data for the tasks
Available via:	OS.h

Name:	CAN_ConfigType
Type:	Structure
Range:	Implementation Specific
Description:	The data structure containing the overall initialization data for the CAN module
Available via:	CAN.h

Name:	CAN_BufferType
Type:	Char
Range:	0 to 255
Description:	The CAN frame holding the message sent/received
Available via:	CAN.h

Name:	CAN_LevelType
Type:	Char
Range:	0 to 255

Description:	The length of the message to store in the CAN frame
Available via:	CAN.h

Name:	LightSwitch_LevelType
Type:	Char
Range:	0 to 1
Description:	The boolean value indicating if the light switch is RELEASED (0) or PRESSED (1)
Available via:	LightSwitch.h

Name:	LightSwitch_PortType
Type:	Char
Range:	0 to Number of Ports
Description:	The number of the port containing the light switch pin based on the DIO enum numbers
Available via:	LightSwitch.h

Name:	LightSwitch_PinType
Type:	Char
Range:	0 to 7
Description:	The number of the light switch pin based on the DIO enum members
Available via:	LightSwitch.h

Name:	DoorSensor_LevelType
Type:	Char
Range:	0 to 1
Description:	The boolean value indicating if the door is CLOSED (0) or OPEN (1)
Available via:	DoorSensor.h

Name:	DoorSensor_PortType
Type:	Char
Range:	0 to Number of Ports
Description:	The number of the port containing the door sensor pin based on the DIO enum numbers
Available via:	DoorSensor.h

Name:	DoorSensor_PinType
Type:	Char
Range:	0 to 7
Description:	The number of the door sensor pin based on the DIO enum members
Available via:	DoorSensor.h

Name:	SpeedSensor_LevelType
Type:	Char
Range:	0 to 1
Description:	The boolean value indicating if the car is STOPPED (0) or MOVING (1)
Available via:	SpeedSensor.h

Name:	SpeedSensor_PortType
Type:	Char
Range:	0 to Number of Ports
Description:	The number of the port containing the speed sensor pin based on the DIO enum numbers
Available via:	SpeedSensor.h

Name:	SpeedSensor_PinType
Type:	Char
Range:	0 to 7
Description:	The number of the speed sensor pin based on the DIO enum members
Available via:	SpeedSensor.h

Name:	Light_LevelType
Type:	Char
Range:	0 to 1
Description:	The boolean value setting the car light OFF (0) or ON (1)
Available via:	LeftLight.h/RightLight.h

Name:	Light_PortType
Type:	Char
Range:	0 to Number of Ports
Description:	The number of the port containing the car light pin based on the DIO enum numbers
Available via:	LeftLight.h/RightLight.h

Name:	Light_PinType
Type:	Char
Range:	0 to 7
Description:	The number of the car light pin based on the DIO enum members
Available via:	LeftLight.h/RightLight.h

Name:	Buzzer_LevelType
Type:	Char
Range:	0 to 1
Description:	The boolean value setting the buzzer OFF (0) or ON (1)
Available via:	Buzzer.h

Name:	Buzzer_PortType
Type:	Char
Range:	0 to Number of Ports
Description:	The number of the port containing the buzzer pin based on the DIO enum numbers
Available via:	Buzzer.h

Name:	Buzzer_PinType
Type:	Char
Range:	0 to 7
Description:	The number of the buzzer pin based on the DIO enum members
Available via:	Buzzer.h

- *"Prepare your folder structure according to the previous points."*

```
.
├── MCAL/
│   ├── Configuration/
│   │   ├── inc/
│   │   │   ├── DIO_Cfg.h
│   │   │   ├── DIO_Lcfg.h
│   │   │   ├── Timer_Cfg.h
│   │   │   └── Timer_Lcfg.h
│   │   └── src/
│   │       ├── DIO_Lcfg.c
│   │       └── Timer_Lcfg.c
│   ├── DIO/
│   │   ├── inc/
│   │   │   └── DIO.h
│   │   └── src/
│   │       └── DIO.c
│   └── Timer/
│       ├── inc/
│       │   └── Timer.h
│       └── src/
│           └── Timer.c
├── HAL/
│   ├── Configuration/
│   │   ├── inc/
│   │   │   ├── LeftLight_Cfg.h
│   │   │   ├── LeftLight_Lcfg.h
│   │   │   ├── RightLight_Cfg.h
│   │   │   ├── RightLight_Lcfg.h
│   │   │   ├── Buzzer_Cfg.h
│   │   │   └── Buzzer_Lcfg.h
│   │   └── src/
│   │       ├── Buzzer_Lcfg.c
│   │       ├── Buzzer_Lcfg.c
│   │       └── Buzzer_Lcfg.c
│   ├── LeftLight/
│   │   ├── inc/
│   │   │   └── LeftLight.h
│   │   └── src/
│   │       └── LeftLight.c
│   ├── RightLight/
│   │   ├── inc/
│   │   │   └── RightLight.h
│   │   └── src/
│   │       └── RightLight.c
│   └── Buzzer/
│       ├── inc/
│       │   └── Buzzer.h
│       └── src/
│           └── Buzzer.c
├── Service/
│   ├── Configuration/
│   │   ├── inc/
│   │   │   ├── OS_Cfg.h
│   │   │   ├── OS_Lcfg.h
│   │   │   ├── CAN_Cfg.h
│   │   │   └── CAN_Lcfg.h
│   │   └── src/
│   │       ├── OS_Lcfg.c
│   │       └── CAN_Lcfg.c
│   ├── OS/
│   │   ├── inc/
│   │   │   └── OS.h
│   │   └── src/
│   │       └── OS.c
│   └── CAN/
│       ├── inc/
│       │   └── CAN.h
│       │
│       └── src/
│           └── CAN.c
├── Library/
│   ├── StdTypes/
│   │   └── inc/
│   │       └── StdTypes.h
│   └── Common/
│       └── inc/
│           ├── PlatformTypes.h
│           ├── MCU_HW.h
│           └── Compiler_Cfg.h
└── Application/
    ├── main.c
    ├── LightsControl/
    │   ├── inc/
    │   │   └── LightsControl.h
    │   └── src/
    │       └── LightsControl.c
    └── BuzzerControl/
        ├── inc/
        │   └── BuzzerControl.h
        └── src/
            └── BuzzerControl.c
```