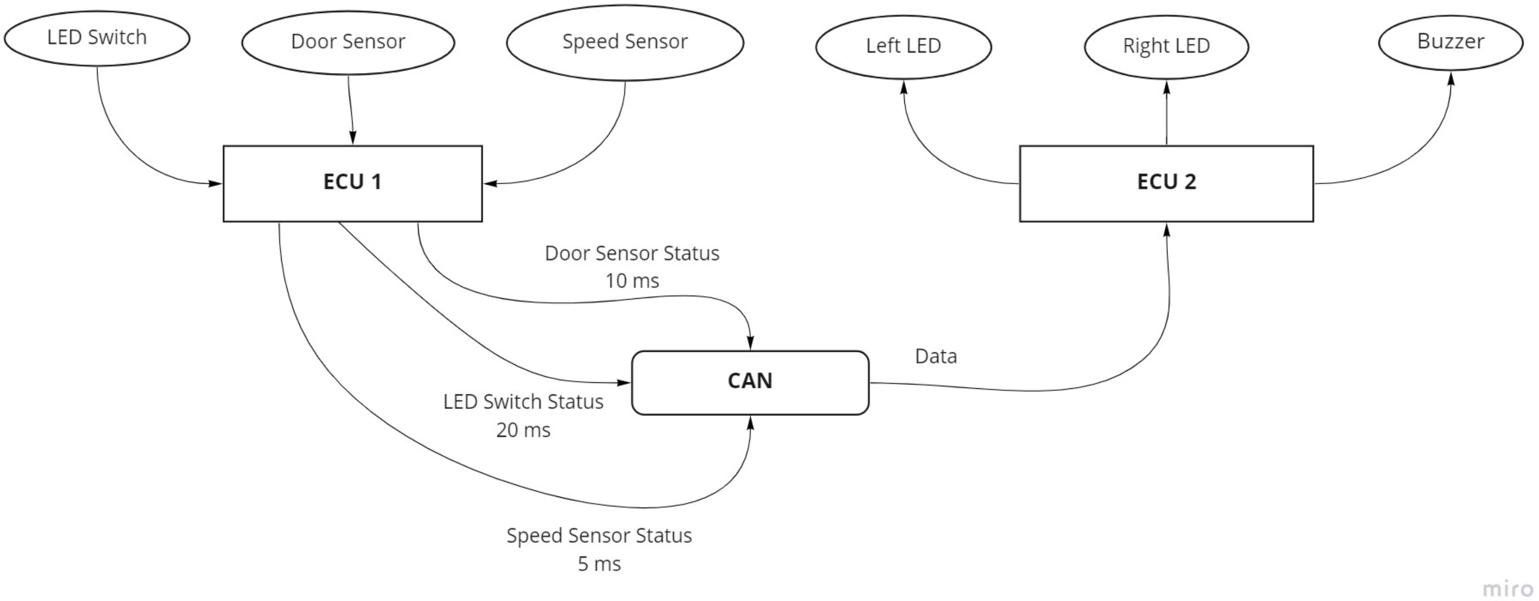
Name: Tarek Alaa Gaber

Email: tarekalaa799@gmail.com

Topic: Automotive Door Control System Design

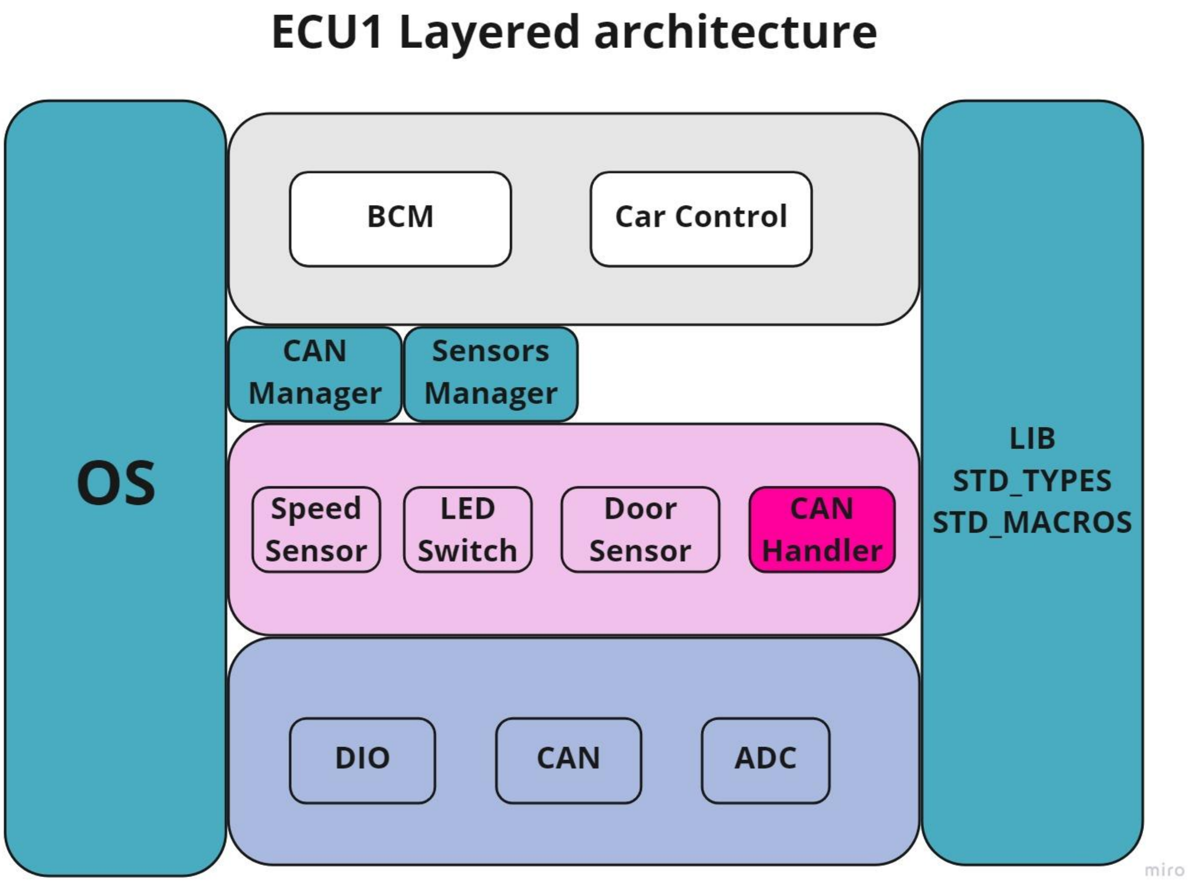
1. Static Design

1.1. Block Diagram



1.2. ECU 1

- Layered architecture



- APIs and typedefs

DIO Driver:

typedef unsigned char u8
typedef struct DIO_ConfigType

- void DIO_Init(const DIO_ConfigType * ConfigPtr, u8 size)
  Name: DIO_Init
  Arguments:
  - Name: ConfigPtr
  - Type: pointer to DIO_ConfigType
  - Range: structure size

- Description: pointer to array that has all configurations to the selected pins passed by use (ex: pin number, type, speed…)

- Name: size
- Type: u8
- Range: 0:10
- Description: argument that has size of array of used pins

Return type: void
Description: This API called to configure GPIO pins in the ECU using array of struct => typedef struct DIO_ConfigType;

- u8 DIO_ReadChannel(u8 ChannelId)

Name: DIO_ReadChannel
Arguments:

- Name: ChannelId
- Type: u8
- Range: 0:10
- Description: Channel number to be read

Return type: u8
Description: API to read the value of GPIO Channel.

- Void DIO_WriteChannel(u8 ChannelId,u8 Value)

Name: DIO_WriteChannel
Arguments:

- Name: ChannelId
- Type: u8
- Range: 0:10
- Description: Channel number to be written

- Name: Value
- Type: u8
- Range: 0:1
- Description: Value to be written

Return type: void

Description: API to write the value of GPIO Channel.

- u8 DIO_ReadPort (u8 PortId)

Name: DIO_ReadPort

Arguments:

- Name: PortId
- Type: u8
- Range: 0:10
- Description: Port to be read

Return type: u8

Description: API to read the value of GPIO Port.

- Void DIO_WritePort (u8 PortId, u8 Value)

Name: DIO_WritePort

Arguments:

- Name: PortId
- Type: u8
- Range: 0:10
- Description: Port to be written

- Name: Value
- Type: u8
- Range: 0:1
- Description: Value to be written

Return type: void

Description: API to write the value of GPIO Port.

ADC Driver:

- void ADC_Init(u8 channels)

Name: ADC_Init

Arguments:

  - Name: channels
  - Type: u8
  - Range: 0:10
  - Description: the channel number to work as ADC

Return type: void
Description: This API called to Initialize the needed GPIO pin as ADC pins

- u8 ADC_ReadChannel (u8 channel)

Name: ADC_ ReadChannel
Arguments:

  - Name: channel
  - Type: u8
  - Range: 0:10
  - Description: the channel number to work as ADC

Return type: u8 ->the value read by ADC
Description: This API to read Value of ADC channel

##

CAN Driver:

- void CAN_Init(void)
  Name: CAN_Init
  Return type: void
  Description: API to initializes CAN module.

- void CAN_SetBaudrate (u16 Baudrate)

Name: CAN_SetBaudrate

Arguments:

  - Name: Baudrate

- Type: u16
- Range: 0: 65535
- Description: the new baud rate

Return type: void

Description: This API to set the baud rate configuration of the CAN controller.

- ➢ void CAN_Write (u16 data);

Name: CAN_Write

Arguments:

- Name: data
- Type: u16
- Range: 0: 65535
- Description: data would be sent

Return type: void

Description: API to send Data via CAN

- ➢ u16 CAN_Read(void)

Name: CAN_Read

Return type: u16

Description: Receive data from CAN

###

Door Sensor:

Must include “DIO Driver”

- ➢ void DoorSensor_Init (u8 ChannelId)

Name: DoorSensor_Init

Arguments:

- Name: Channel
- Type: u8
- Range: 0:10
- Description: Channel connected to Door Sensor

Return type: void

Description: this API to Initialize Channel as Door Sensor

- u8 DoorSensor_Read (u8 ChannelId)

Name: DoorSensor_Read

Arguments:

- Name: ChannelId
- Type: u8
- Range: 0:10
- Description: Channel connected to Door Sensor

Return type: u8 ->the State of Door Sensor

Description: this API to Read Channel of GPIO for Door Sensor

##

Speed Sensor:

Must include “ADC Driver”

- Void SpeedSensor_Init (u8 ChannelId)

Name: SpeedSensor_Init

Arguments:

- Name: ChannelId
- Type: u8
- Range: 0:10
- Description: Channel connected to Speed Sensor

Return type: void

Description: This API to initialize Channel of GPIO as Speed Sensor

- u8 SpeedSensor_Read (u8 ChannelID)

Name: SpeedSensor_Read

Arguments:

- Name: ChannelId
- Type: u8
- Range: 0:10
- Description: Channel connected to Speed Sensor

Return type: u8 -> value of Speed Sensor

Description: This API to Read the state of Speed Sensor for the specified ADC channel

##

LED Switch:

Must include "DIO Driver"

- void Switch_Init (u8 ChannelId)

Name: Switch_Init

Arguments:

- Name: ChannelId
- Type: u8
- Range: 0:10
- Description: Channel connected to LED Switch

Return type: void

Description: This API to initialize Channel of GPIO as LED Switch

- u8 Switch_Read (u8 ChannelId)

Name: Switch_Read

Arguments:

- Name: ChannelId
- Type: u8
- Range: 0:10
- Description: Channel connected to LED Switch

Return type: u8-> state of Switch

Description: This API to Read the specified GPIO pin for LED Switch

###

CAN Handler:

To enable total abstraction, a handler will be added as a point of contact between the CAN manager and the can Protocol.

CAN Manager:

- void CANManager_Init(void)
- Name: CANManager_Init
- Return type: void
- Description: API for initialization of communication(using CAN_Init())

- void CANManager_Send(void)
- Name: CANManager_Send
- Return type: void
- Description: API for sending messages between layers (using CAN_Write())

Sensor Manager:

- ➢ void SensorsManager_Init (void)
  - Name: SensorsManager_Init
  - Return type: void
  - Description: API to initialization of all sensors by calling (Switch_Init (u8 ChannelId), DoorSensor_Init (u8 ChannelId), SpeedSensor_Init (u8 ChannelId))

- ➢ Void SensorsManager_Read(void)
  - Name: SensorsManager_Read
  - Return type: void
  - Description: API to get sensors readings (DoorSensor_Read (u8 ChannelId), SpeedSensor_Read (u8 ChannelId), Switch_Read (u8 ChannelId))

###

BCM:

- ➢ void BCM_Init ()
  - Name: BCM_Init
  - Return type: void
  - Description: this API call the API in OS to establish CAN connection (CANManager_Init ())

- ➢ void BCM_Send ()
  - Name: BCM_ Send
  - Return type: void
  - Description: this API call the API in OS to Send the status messages to ECU2 (CANManager_Send ())

###

Car Control:

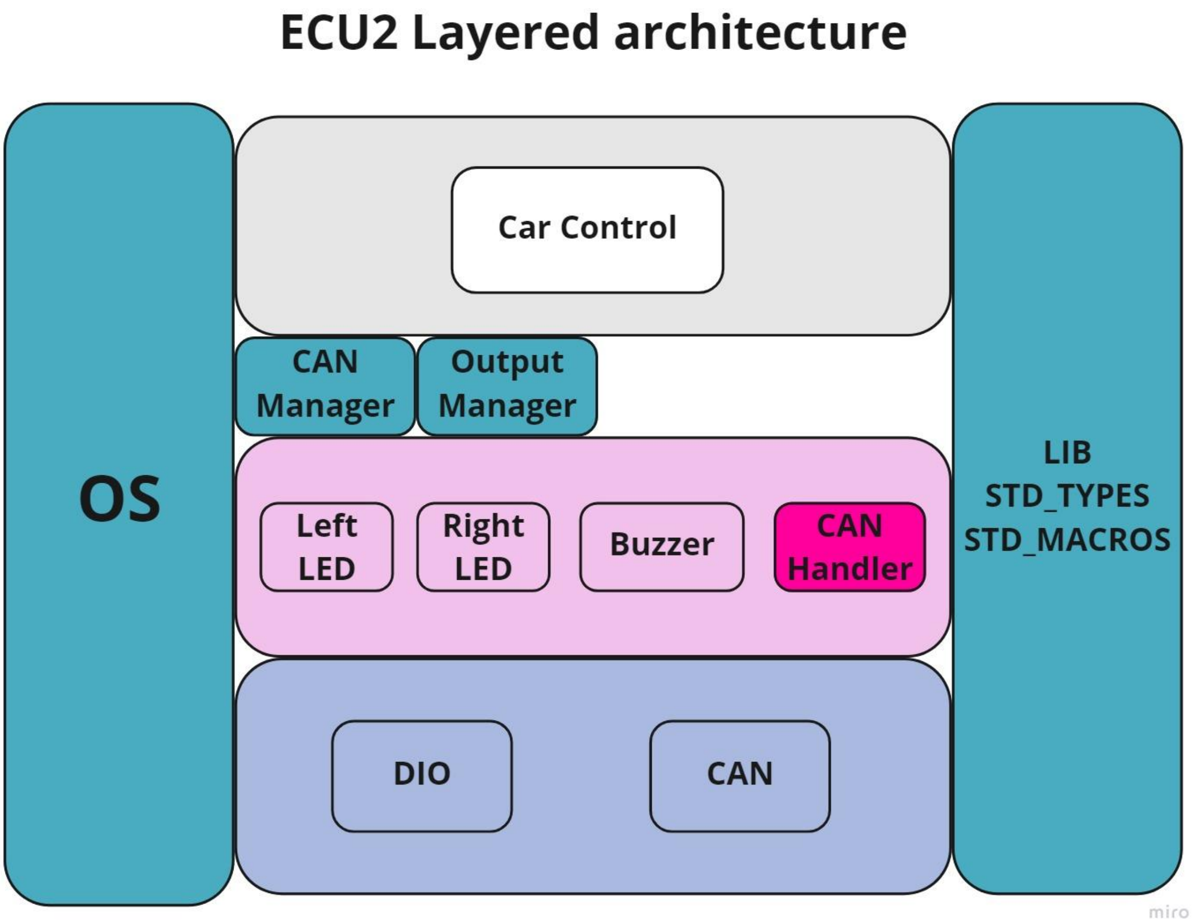
- ➢ void InputDevices_Init (void)
  - Name: InputDevices_Init
  - Return type: void

- Description: API to call the OS API to initialize input devices (SensorsManager_Init ())

  - void InputDevices_Control(void)
- Name: InputDevices_Control
- Return type: void
- Description: API to Call OS API to read input devices readings (SensorsManager_Read ())

1.3. ECU2

- Layered architecture



- APIs and typedefs

DIO Driver:

typedef unsigned char u8

typedef struct DIO_ConfigType

- void DIO_Init (const DIO_ConfigType * ConfigPtr, u8 size)

  Name: DIO_Init

  Arguments:

  - Name: ConfigPtr
  - Type: pointer to DIO_ConfigType
  - Range: structure size

- Description: pointer to array that has all configurations to the selected pins passed by use (ex: pin number, type, speed…)

- Name: size
- Type: u8
- Range: 0:10
- Description: argument that has size of array of used pins

Return type: void
Description: This API called to configure GPIO pins in the ECU using array of struct => typedef struct DIO_ConfigType;

- u8 DIO_ReadChannel (u8 ChannelId)

Name: DIO_ReadChannel
Arguments:

- Name: ChannelId
- Type: u8
- Range: 0:10
- Description: Channel number to be read

Return type: u8
Description: API to read the value of GPIO Channel.

- Void DIO_WriteChannel (u8 ChannelId, u8 Value)

Name: DIO_WriteChannel
Arguments:

- Name: ChannelId
- Type: u8
- Range: 0:10
- Description: Channel number to be written

- Name: Value
- Type: u8
- Range: 0:1
- Description: Value to be written

Return type: void
Description: API to write the value of GPIO Channel.

- u8 DIO_ReadPort (u8 PortId)

Name: DIO_ReadPort
Arguments:

- Name: PortId
- Type: u8
- Range: 0:10
- Description: Port to be read

Return type: u8
Description: API to read the value of GPIO Port.

- Void DIO_WritePort (u8 PortId, u8 Value)

Name: DIO_WritePort
Arguments:

- Name: PortId
- Type: u8
- Range: 0:10
- Description: Port to be written

- Name: Value
- Type: u8
- Range: 0:1
- Description: Value to be written

Return type: void
Description: API to write the value of GPIO Port.

CAN Driver:

- void CAN_Init(void)
  Name: CAN_Init
  Return type: void
  Description: API to initializes CAN module.

- void CAN_SetBaudrate (u16 Baudrate)

Name: CAN_SetBaudrate

Arguments:

- Name: Baudrate
- Type: u16
- Range: 0: 65535
- Description: the new baud rate

Return type: void

Description: This API to set the baud rate configuration of the CAN controller.

- void CAN_Write (u16 data);

Name: CAN_Write

Arguments:

- Name: data
- Type: u16
- Range: 0: 65535
- Description: data would be sent

Return type: void

Description: API to send Data via CAN

- u16 CAN_Read(void)

Name: CAN_Read

Return type: u16

Description: Receive data from CAN

Right LED Driver:

- ➢ void RL_Init (u8 ChannelId)

Name: RL_Init
Arguments:

- o Name: ChannelId
- o Type: u8
- o Range: 0:10
- o Description: Channel connected to Right LED

Return type: void
Description: API to Initialize Channel of GPIO as Right LED

- ➢ void RL_ON (u8 ChannelId)

Name: RL_ON
Arguments:

- o Name: ChannelId
- o Type: u8
- o Range: 0:10
- o Description: Channel connected to Right LED

Return type: void
Description: API to make Right LED ON

- ➢ void RL_OFF (u8 ChannelId)

Name: RL_OFF
Arguments:

- o Name: ChannelId
- o Type: u8
- o Range: 0:10
- o Description: Channel of GPIO connected to Right LED

Return type: void
Description: API to make Right LED OFF

Left LED Driver:

- void LL_Init (u8 ChannelId)

Name: LL_Init

Arguments:

- Name: ChannelId
- Type: u8
- Range: 0:10
- Description: Channel connected to Left LED

Return type: void

Description: API to Initialize Channel of GPIO as Left LED

- void LL_ON (u8 ChannelId)

Name: LL_ON

Arguments:

- Name: ChannelId
- Type: u8
- Range: 0:10
- Description: Channel of GPIO connected to Left LED

Return type: void

Description: API to make Left LED ON

- void LL_OFF (void)

Name: LL_OFF

Arguments:

- Name: ChannelId
- Type: u8
- Range: 0:10
- Description: Channel of GPIO connected to Left LED

Return type: void

Description: API to make Left LED OFF

Buzzer Driver:

- void Buzzer_Init (u8 ChannelId)

Name: Buzzer_Init

Arguments:

- Name: ChannelId
- Type: u8
- Range: 0:10
- Description: Channel of GPIO connected to Buzzer

Return type: void

Description: API to initialize Channel of GPIO as Buzzer

- void Buzzer_ON (u8 ChannelId)

Name: Buzzer_ON

Arguments:

- Name: ChannelId
- Type: u8
- Range: 0:10
- Description: Channel of GPIO connected to Buzzer

Return type: void

Description: API to make Buzzer ON

- void Buzzer_OFF (u8 ChannelId)

Name: Buzzer_OFF

Arguments:

- Name: ChannelId
- Type: u8
- Range: 0:10
- Description: Channel of GPIO connected to Buzzer

Return type: void

Description: API to make Buzzer ON

CAN Handler:

To enable total abstraction, a handler will be added as a point of contact between the CAN manager and the can Protocol.

CAN Manager:

- ➢ void CANManager_Init(void)
- o Name: CANManager_Init
- o Return type: void
- o Description: API for initialization of communication

- ➢ void CANManager_Receive(void)
- o Name: CANManager_Receive
- o Return type: void
- o Description: API for Receiving messages (using CAN_Read ())

###

Output Manager:

- ➢ void OutputManager_Init(void)

Name: OutputManager_Init

Description: API for initialization of all Output devices by calling (LL_Init(), RL_Init(),Buzzer_Init())

- ➢ Void OutputManager_Control(void)

Name: OutputManager_Control

Description: API for controlling of all Output devices by calling (LL_ON (), RL_ON (), RL_OFF (), LL_OF (), Buzzer_ON (), Buzzer_OFF ())

###

Car Control:

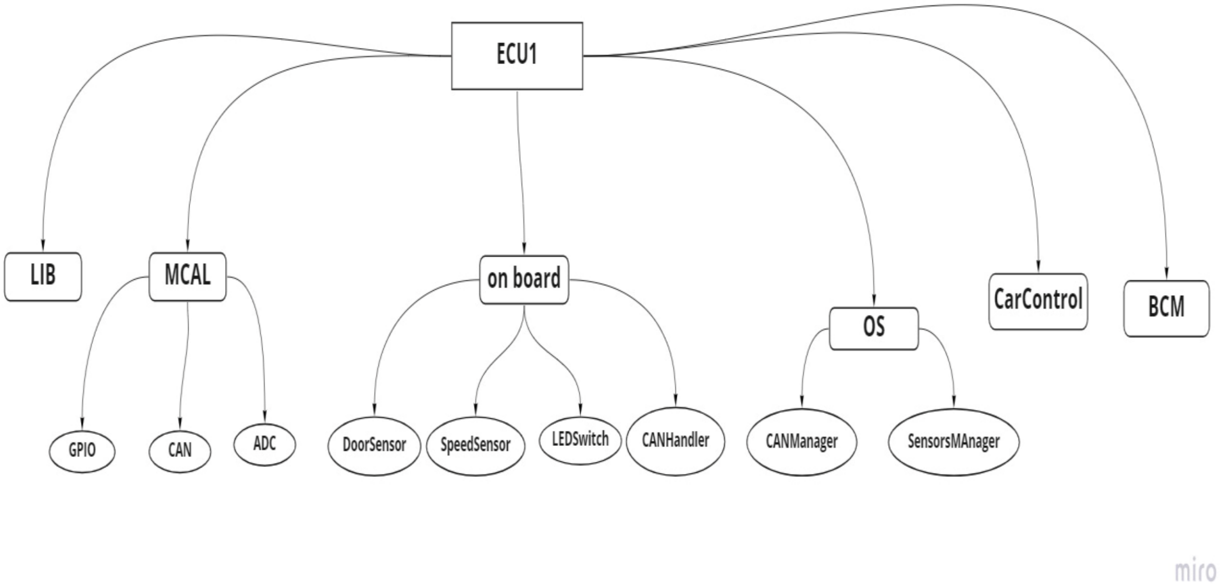
- ➢ void CommunicationManager_Init(void)
- o Name: CommunicationManager_Init
- o Return type: void
- o Description: this API Call the OS API to initialize Communication (CANManager_Init ())

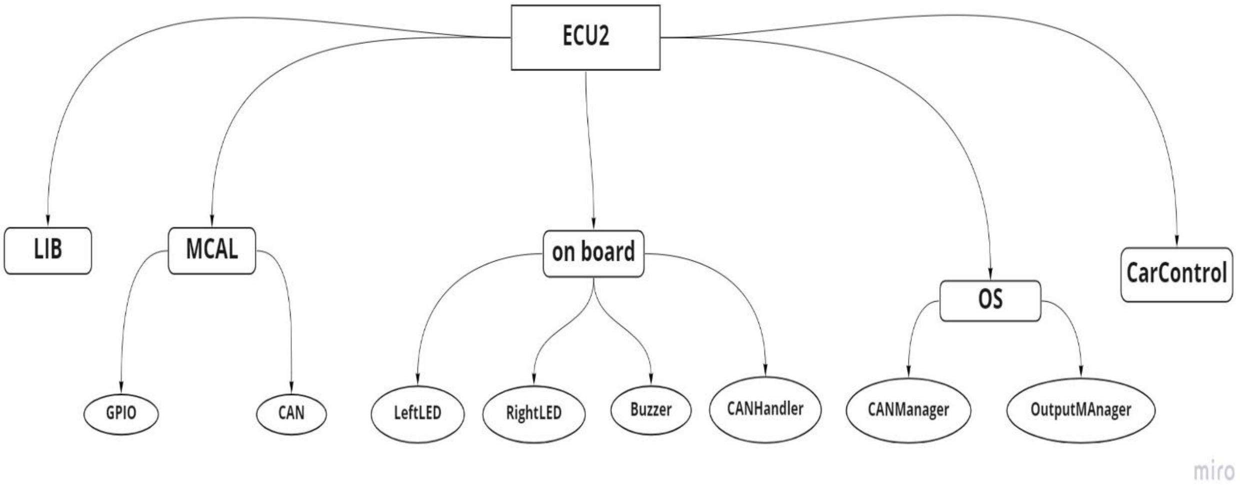
- ➢ Void ReceivingMesseges_Control(void)
- o Name: ReceivingMesseges_Control
- o Return type: void

- Description: this API Call OS API to start receiving can messages (CANManager_Receive ())

- void OutputDevices_Init(void)
- Name: OutputDevices_Init
- Return type: void
- Description: this API Call the OS API to initialize Output devices (OutputManager_Init ())
- void OutputDevices_Control(void)
- Name: OutputDevices_Control
- Return type: void
- Description: this API call the OS API to control output devices(using OutputManager_Control())

2. Folder Structure



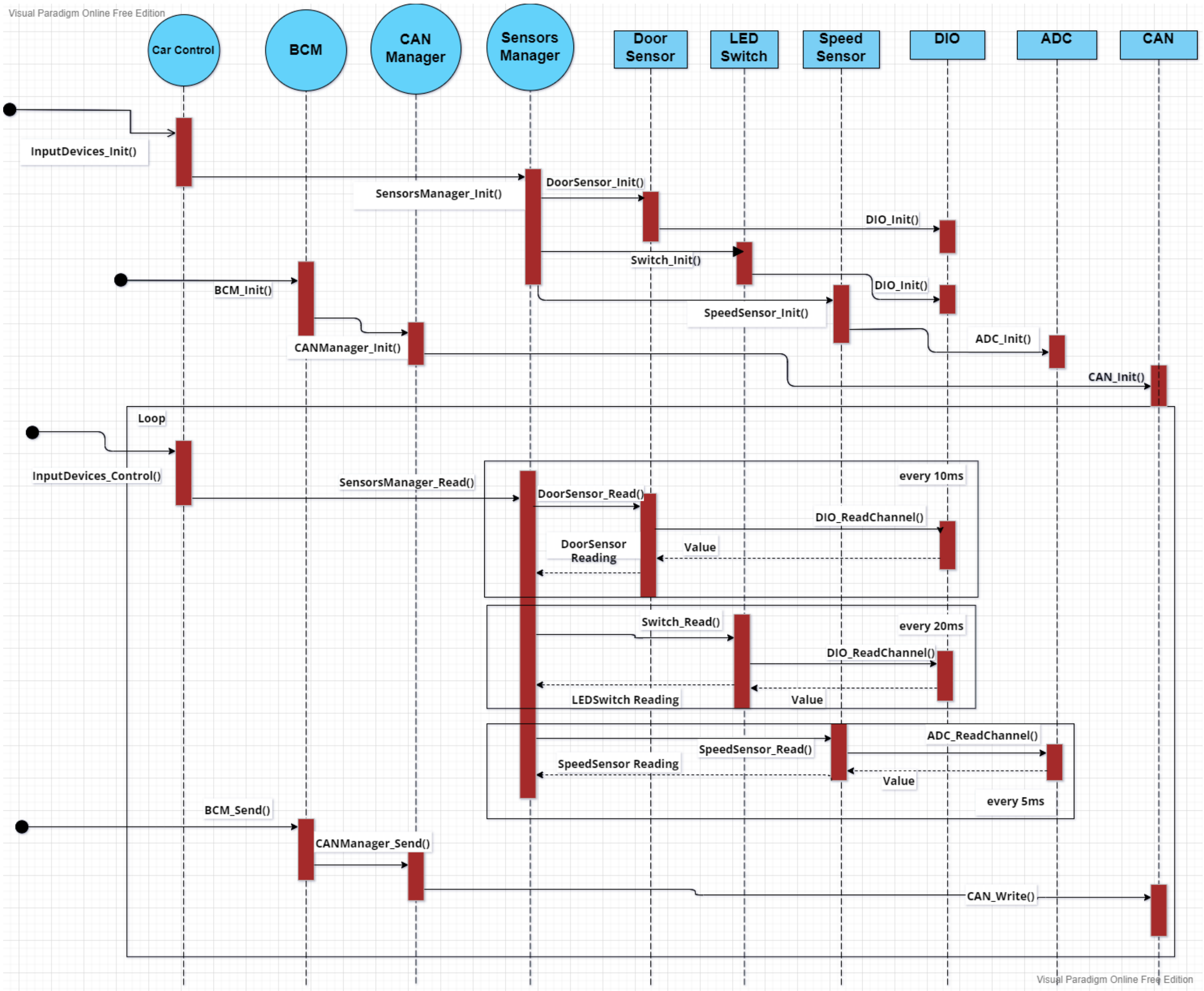


3. Dynamic Design

3.1. ECU1

- **Sequence diagram**

ECU1 Sequence diagram



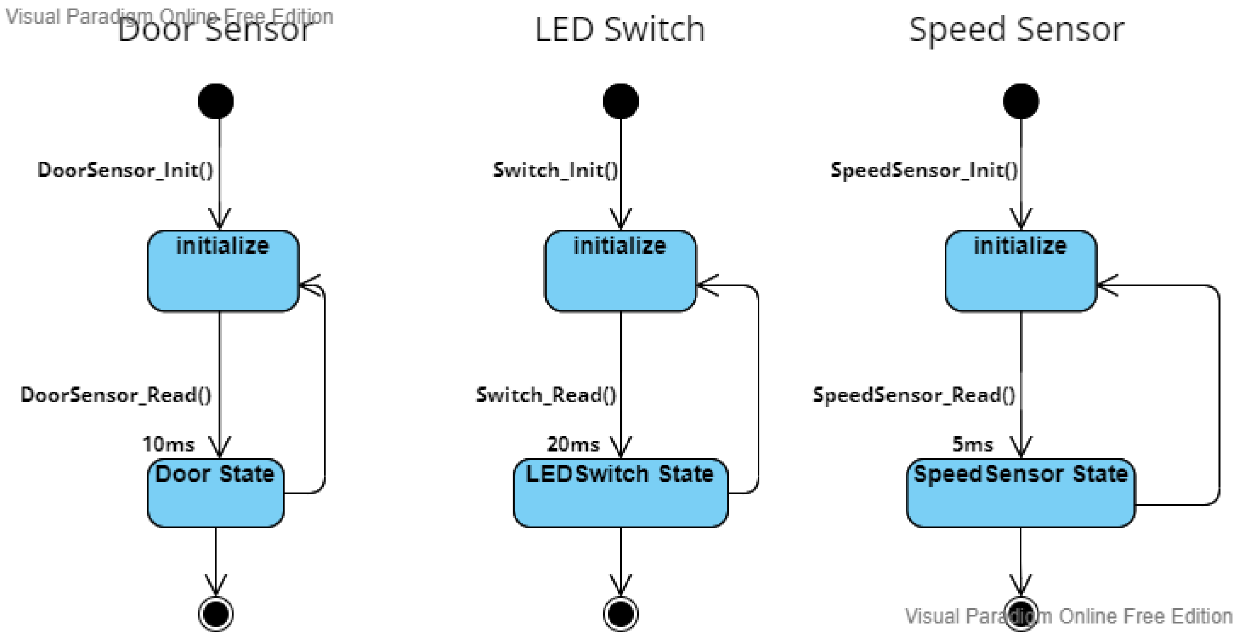
- **CPU load**

CPU Utilization = 100 - idle time = 100 - 65

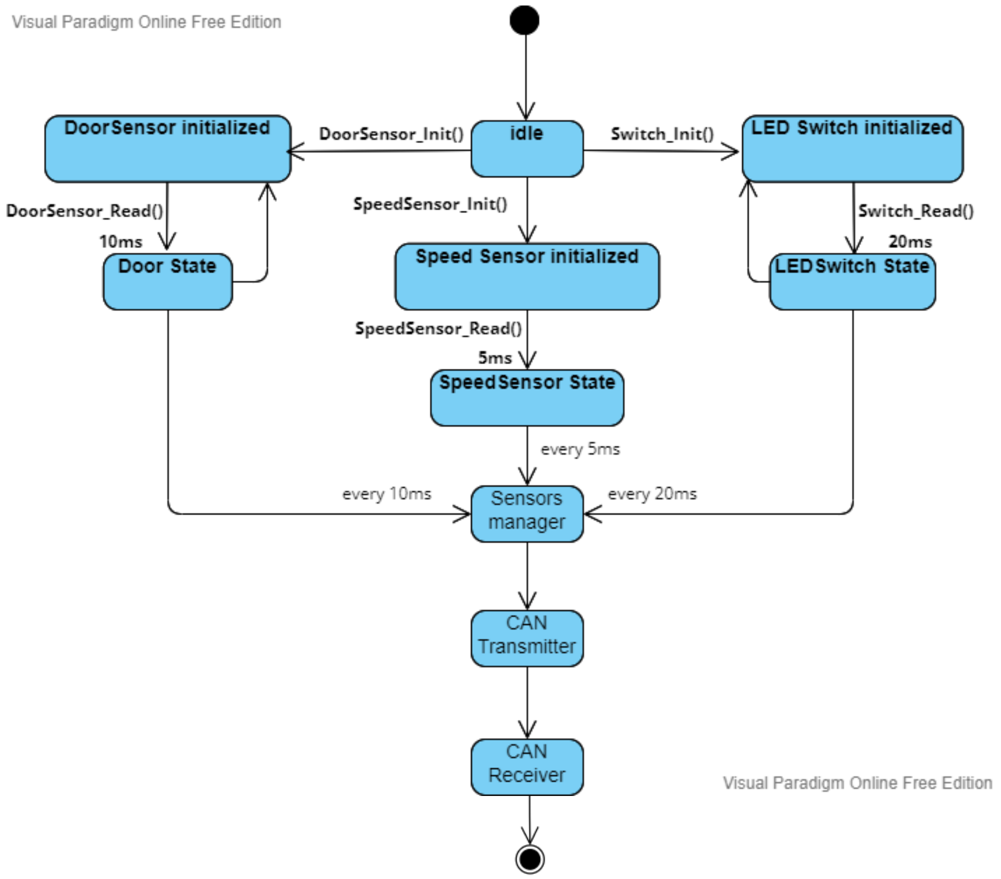
CPU Utilization = 35%

- **State machine**

  - **Each component**



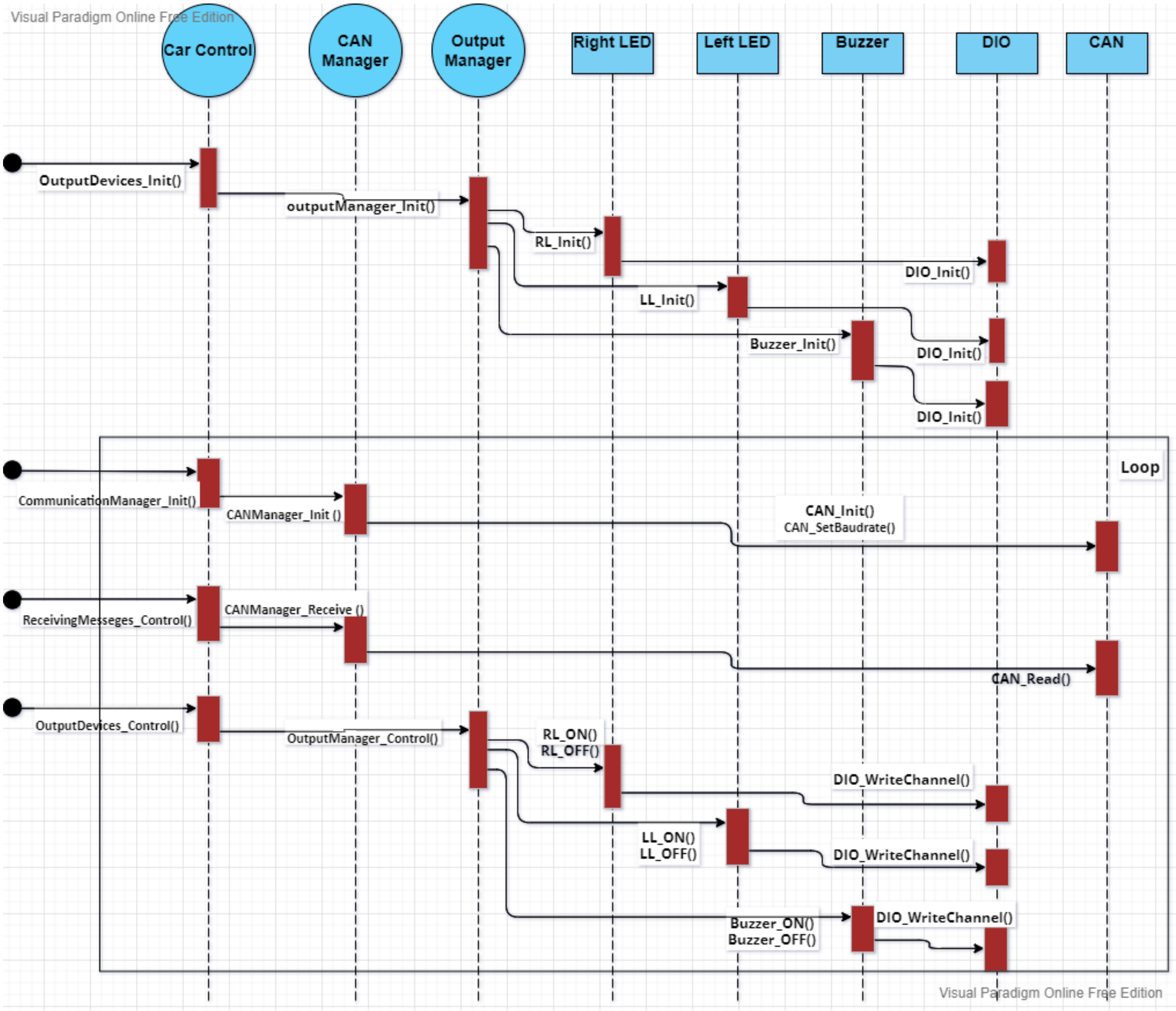
  - **ECU1 operation**



3.2. ECU2

- **Sequence diagram**

ECU2 Sequence diagram

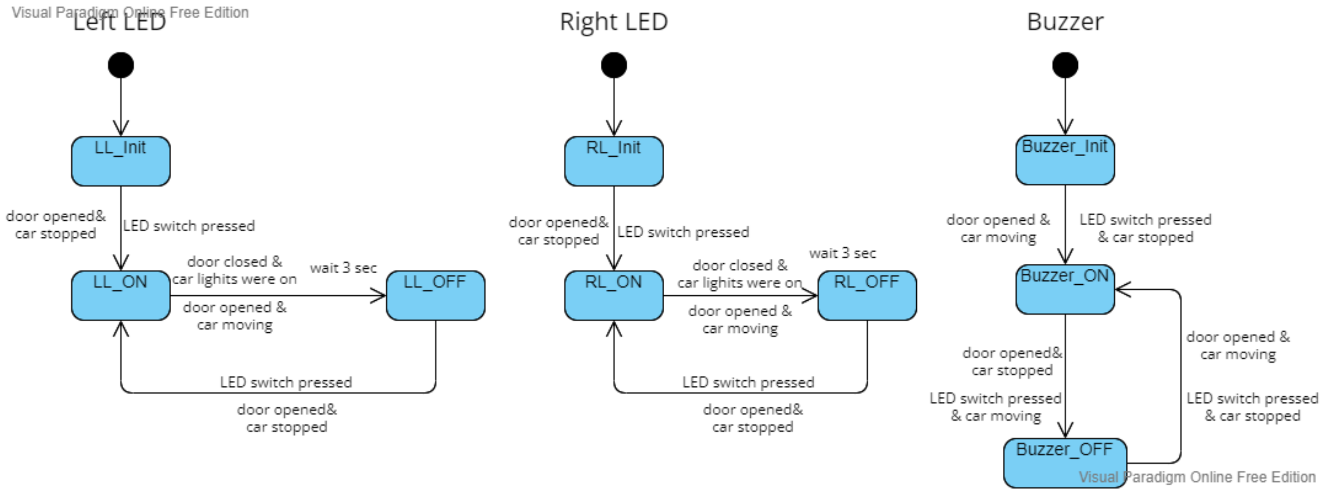


- **CPU load**

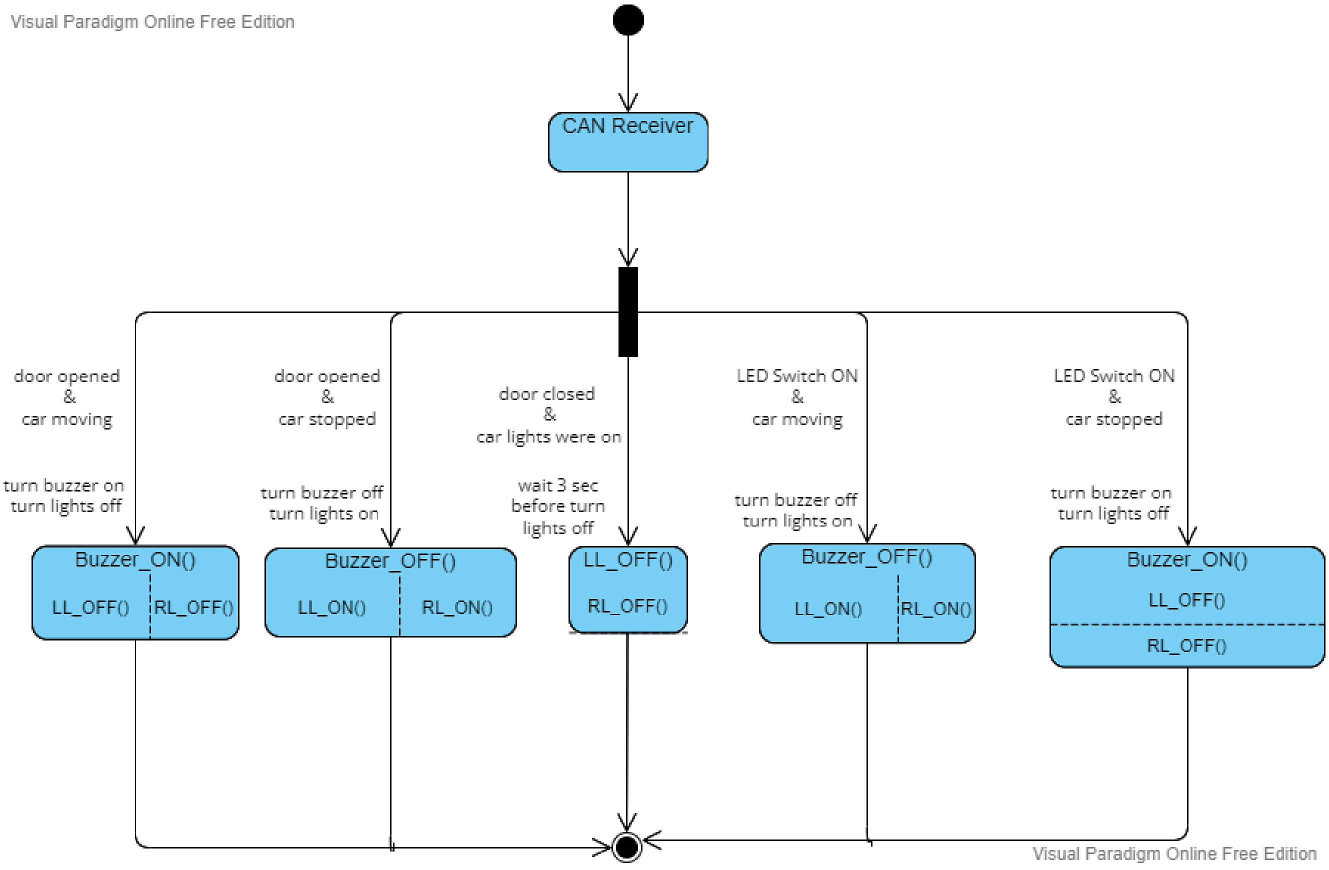
CPU Utilization = 100 - idle time = 100 - 65

CPU Utilization = 35%

- **State machine**
  - ➢ **Each component**



- ➢ **ECU2 operation**



4. Bus load

A CAN frame has approximately 125 bits

Assume that we are using 500 kBit/s bit rate:

bit time = 1 / bit rate = 1 / (500 * 1000) s = 2 * 10^{-6} s = 2 µs

This means 1 bit will take 2 µs to transfer on bus when using 500 kBit/s

So the approximate time to transfer 1 frame is (2 µs/bit * 125 bit) = 250 µs

Three messages are:

- Door sensor message = 1 frame every 10 ms
- Light switch message = 1 frame every 20 ms
- Speed sensor message = 1 frame every 5 ms

1 frame every 10 ms = 100 frames every 1000 ms

1 frame every 20 ms = 50 frames every 1000 ms

1 frame every 5 ms = 200 frame every 1000 ms

Total frames = 350 frames every 1000 ms

Total time on bus = 350 * 250 µs

Total time = 1000 ms

Bus load = ((350 * 250) / (1000 * 1000)) * 100 % = 8.75 %