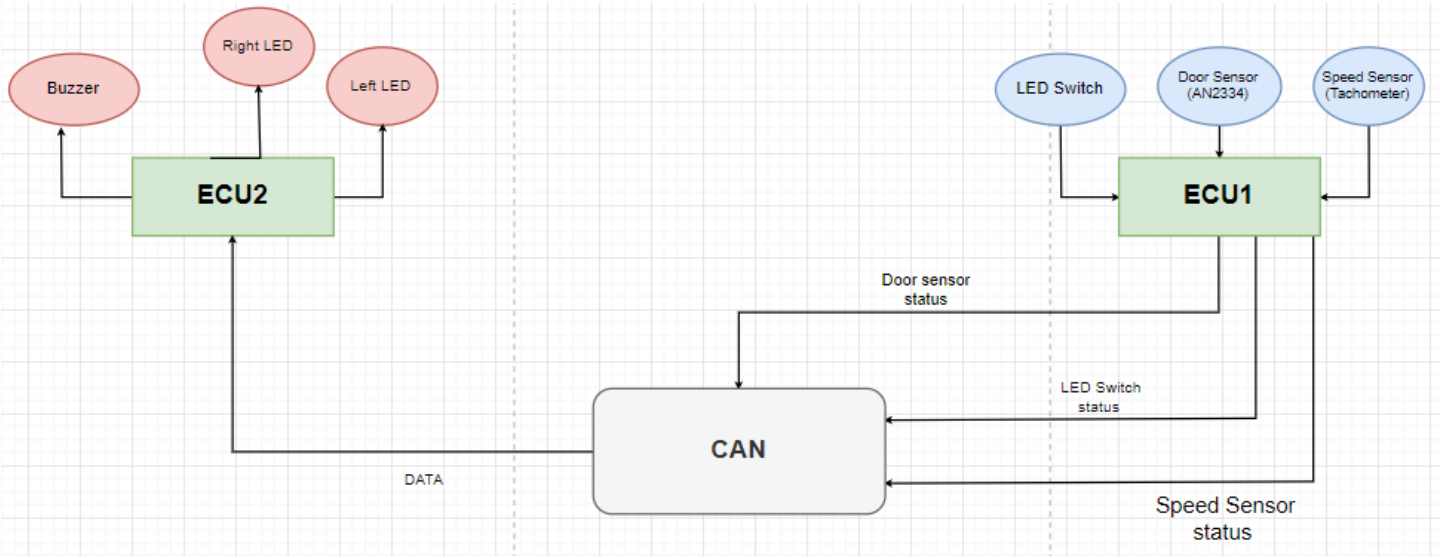
Automotive door control system design

Tarek Mahmoud

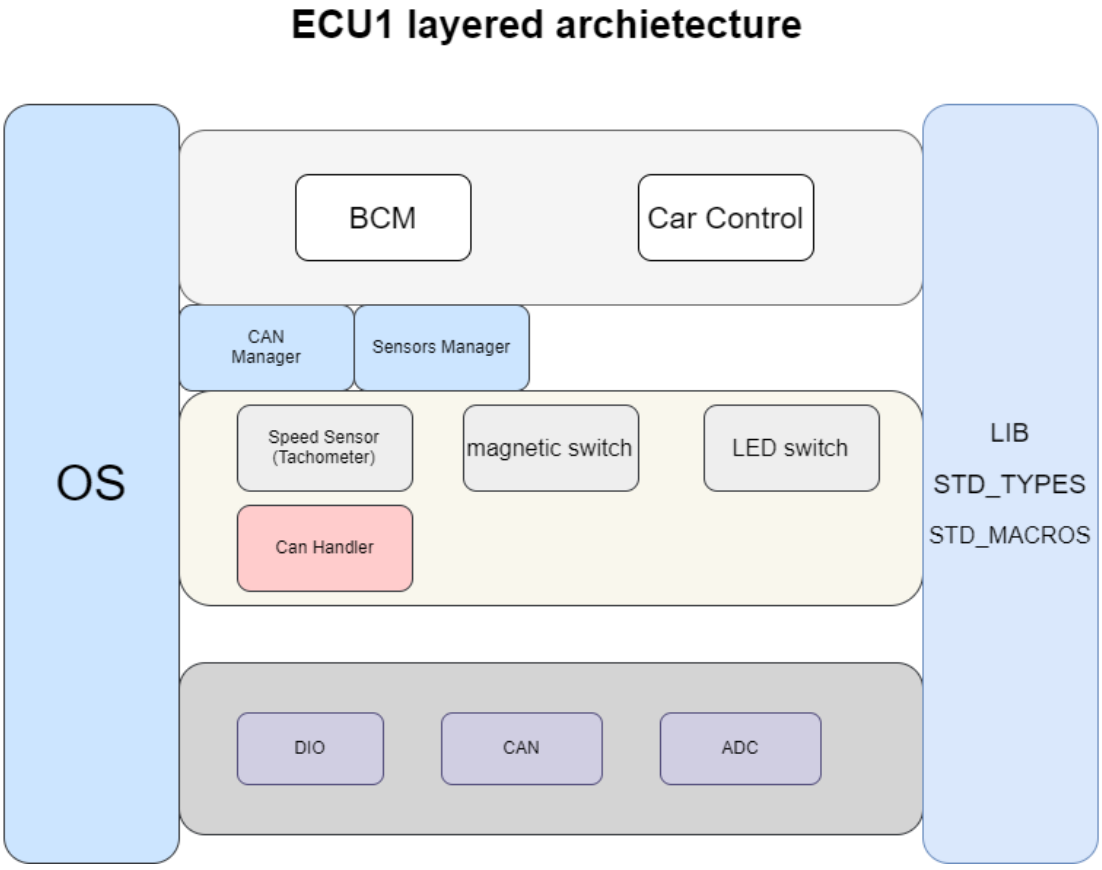
1. Static design

1.1 Block diagram



1.2 For ECU1

- Layered architecture



- ECU components and modules

DIO for (LED switch, Magnetic switch for door)
ADC for (Speed sensor)
CAN for sending sensors data to ECU2

- APIs and typedefs

DIO Module:

typedef unsigned char u8

typedef struct pinInfo

- void DIO_Init(pinInfo* Pins, u8 size)

  Name: DIO_Init

  Arguments:

  - Name: Pins
  - Type: pointer to pinInfo
  - 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 pinInfo;

- u8 DIO_Read(u8 pin)

  Name: DIO_Read

  Arguments:

  - Name: pin
  - Type: u8
  - Range: 0:10

- Description: pin number to be read

Return type: u8

Description: API to read the value of GPIO pin.

ADC Module:

- 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 configure the needed GPIO pin as ADC pins
- u8 ADC_Read(u8 channel)

  Name: ADC_ Read

  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 ADC channel

CAN Module:

- 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 buad rate buadrate

  Return type: void

  Description: This service shall set the baud rate configuration of the CAN controller.

- Can_Write(u16 data);

  Name: Can_Write

  Arguments:

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

  Return type: void

  Description: API to send Data via CAN

Switch module:

- void Switch_Init**(u8 pin)**

  Name: Switch _Init

  - Arguments: Name: pin
  - Type: u8
  - Range: 0: 10
  - Description: pin connected to switch

Return type: void

Description: This API called to Configure the Switch to the GPIO pin

- u8 Switch_Read(u8 pin)

  Name: Switch _ Read

`

Arguments:

- Name: pin
- Type: u8
- Range: 0: 10
- Description: pin connected to switch

Return type: u8 →state of GPIO pin

Description: This API called to Read the specified GPIO pin for the switch

Speed Sensor module:

- void SpeedSensor_Init(u8 pin);

  Name: SpeedSensor _Init

Arguments:

- Name: pin
- Type: u8
- Range: 0: 10
- Description: pin connected to speed sensor

Return type: void

Description: This API called to configure the Speed Sensor to the GPIO pin

- U8 SpeedSensor_GetReading(u8 pin)

  Name: SpeedSensor_GetReading

Arguments:

- Name: pin
- Type: u8
- Range: 0: 10
- Description: pin connected to speed sensor

Return type: u8 value of speed sensor

Description Get the current reading for the specified ADC channel

Magnetic Switch:

- void MagneticSwitch_Init**(u8 pin)**
  Name: MagneticSwitch_Init
    - Arguments: Name: pin
    - Type: u8
    - Range: 0: 10
    - Description: pin connected to MagneticSwitch

  Return type: void

  Description: This API called to configure the magnetic Switch to the GPIO pin
- u8 MagneticSwitch_Read (u8 pin)
  Name: MagneticSwitch_Read

  `Arguments:
    - Name: pin
    - Type: u8
    - Range: 0: 10
    - Description: pin connected to switch

  Return type: u8 →state of GPIO pin

  Description: This API called to Read the specified GPIO pin for the magnetic switch

CAN Handler:

- To provide total abstraction the handler will be add as a middle of communication between CAN manager and can Protocol

CAN Manager:

- Void Can_ManagerInit(void)
  Name: Can_ ManagerInit

  Return type: void

  Description: Responsible for initialization of communication
- Void Can_ManagerSendData(void)
  Name: Can_ ManagerSendData

  Return type: void

Description: Responsible for sending and receiving messages between layers

Parameters passed to the API from configuration files

Sensor Mannager:

- Void Sensors_ManagerInit(void)

  Name: Sensors_ManagerInit

  Return type: void

  Description:Responsible for initialization of all sensors by calling (Switch_Init(), MagneticSwitch_Init(), SpeedSensor_Init()).
- Void Sensor_ManagerRead(void)

  Name: Sensors_ManagerInit

  Return type: void

  Description:Responsible for get sensors readings (MagneticSwitch_Read(), SpeedSensor_Read(), Switch_Read()).

Parameters passed to the API from configuration files

BCM Module:

- BCM_Init()

  Name: BCM_Init

  Return type: void

  Description: call the API in OS to establish CAN connection (Can_ManagerInit())
- BCM_Handler()

  Name: BCM_ Handler

  Return type: void

  Description: call the API in OS to Send the status messeges to ECU2 (Can_ManagerSendData())

Car Control Module:

- Void Init_Inputdevices(void)

  Name: *Init_Inputdevices*

  Return type: void

  Description: API to call the OS API to initialize input devices (*Sensors_ManagerInit()*)
- Void Control_Inputdevices(void)

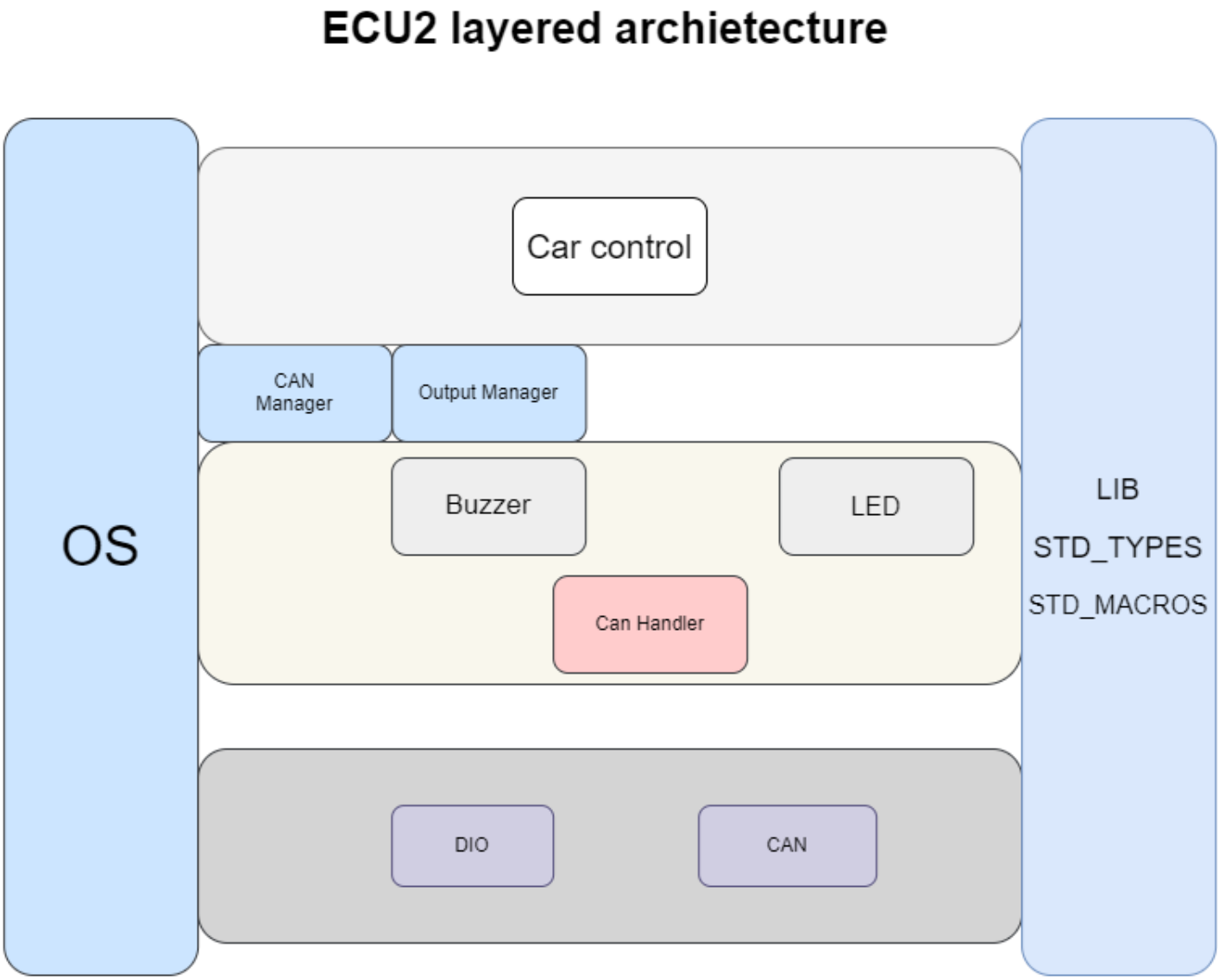
  Name: *Init_Inputdevices*

Return type: void

Description: *API to Call OS API to read input devices readings (Sensors_ManagerRead()) .*

For ECU2

- Layered architecture



- ECU components and modules

DIO for (LED, and buzzer)

CAN for receiving sensors data from ECU1

- APIs and typedefs

DIO Module:

  - void DIO_Init(pinInfo* Pins, u8 size)

Name: DIO_Init

Arguments:

- Name: Pins
- Type: pointer to pinInfo
- 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 pinInfo;

- void DIO_Write(u8 pin, u8 value)

Name: DIO_Write

Arguments:

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

- Name: value
- Type: u8
- Range: 0:1
- Description: value of gpio pin

Return type: void

Description: API to write the value of GPIO pin.

CAN Module:

- 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 service shall set the baud rate configuration of the CAN controller.
- U16 Can_Read(void);

  Name: Can_Read

  Arguments: void

  Return type: u16→ data in can bus

  Description: This service shall read the value in can bus

LED module:

- Led_Init(u8 pin)

  Name: Led_Init

  Arguments:

  - Name: pin
  - Type: u8
  - Range: 0: 10
  - Description: pin to be connected to led

  Return type: void

Description: Configure LED to the needed GPIO Pin

- Void Led_High(u8 pin)

  Name: Led_High

  Arguments:

  - Name: pin
  - Type: u8
  - Range: 0: 10
  - Description: pin to be connected to led

Return type: void

Description: Output high to the specified GPIO pin for the LEDs

- Led_Low(u8 pin)

  Name: Led_ Low

  Arguments:

  - Name: pin
  - Type: u8
  - Range: 0: 10
  - Description: pin to be connected to led

Return type: void

Description: Output low to the specified GPIO pin for the LEDs

Buzzer module:

- Buzzer_Init(u8 pin)

  Name: Buzzer_Init

  Arguments:

  - Name: pin
  - Type: u8
  - Range: 0: 10
  - Description: pin to be connected to Buzzer

Return type: void

Description: Configure Buzzer to the needed GPIO Pin

- Void Buzzer _High(u8 pin)

Name: Buzzer _High

Arguments:

- Name: pin
- Type: u8
- Range: 0: 10
- Description: pin to be connected to Buzzer

Return type: void

Description: Output high to the specified GPIO pin for the Buzzer

- Buzzer _Low(u8 pin)

Name: Buzzer _ Low

Arguments:

- Name: pin
- Type: u8
- Range: 0: 10
- Description: pin to be connected to Buzzer

Return type: void

Description: Output low to the specified GPIO pin for the Buzzer

CAN Handler:

To provide total abstraction the handler will be add as a middle of communication between CAN manager and can Protocol

Output Mannager:

- *Output_ManagerInit()*

Name: *Output_ManagerInit*

Return type: void

Description: Responsible for initialization of all Output devices by calling (Led_Init(), Buzzer_Init()).

- *Output_ManagerControl* ()

Name: *Output_ManagerControl*

Return type: void

Description:

Responsible for controlling output devices *(*Buzzer_High(), Buzzer_Low(), Led_High(), Led_Low(),*)*

CAN Manager:

- Can_ManagerInit()

Name: Can_ ManagerInit

Return type: void

Description: Responsible for initialization of communication

- Can_ManagerRecieveData()

Name: Can_ ManagerRecieveData

Return type: void

Description: Responsible for receiving messages (using Can_Read() API Parameters passed to the API from configuration files

Car Control Module:

- Void Init_CommunicationMannager**(void)**

Name: Init_CommunicationMannager

Return type: void

Description: Call the OS API to initialize Communication (Can_ManagerInit*()*)

- Void Control_RecievingMesseges(void)

Name: Control_RecievingMesseges

Return type: void

Description: Call the OS API to initialize Communication

Call OS API to start receiving can messages (Can_ManagerRecieveData())

- Init_Outputdevices()

Name: Control_RecievingMesseges

Return type: void

Description: Call the OS API to initialize Output devices (*Output_ManagerInit()*)

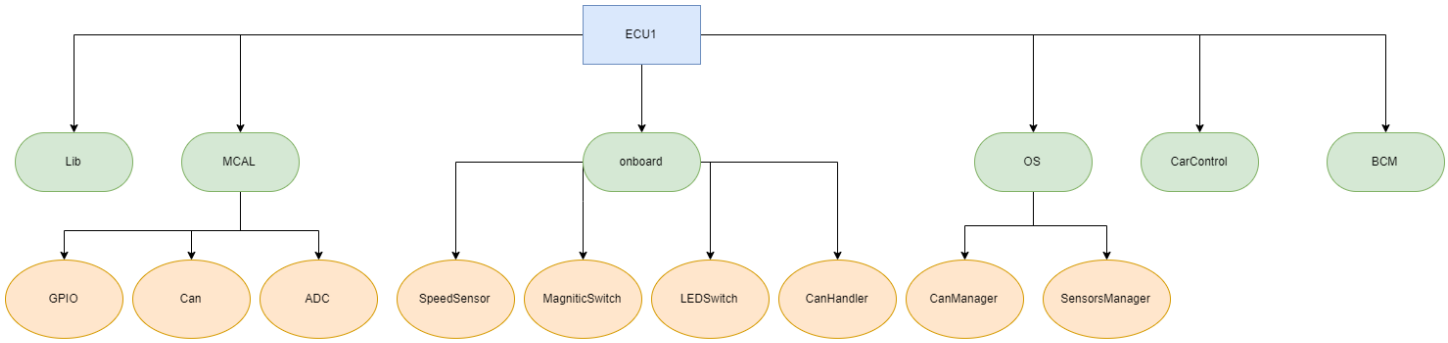
- Control_Outputdevices()

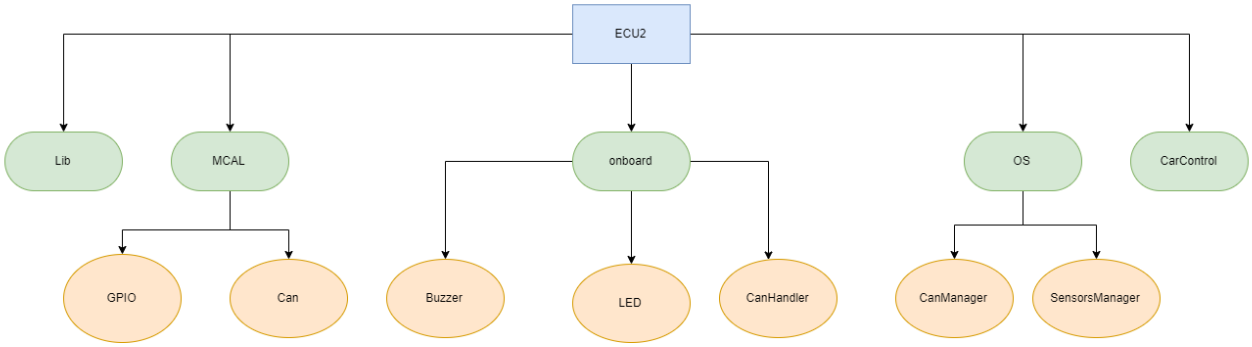
Name: Control_RecievingMesseges

Return type: void

Description: Call OS API to control Output devices *(Output_ManagerControl())*.

Folder structure

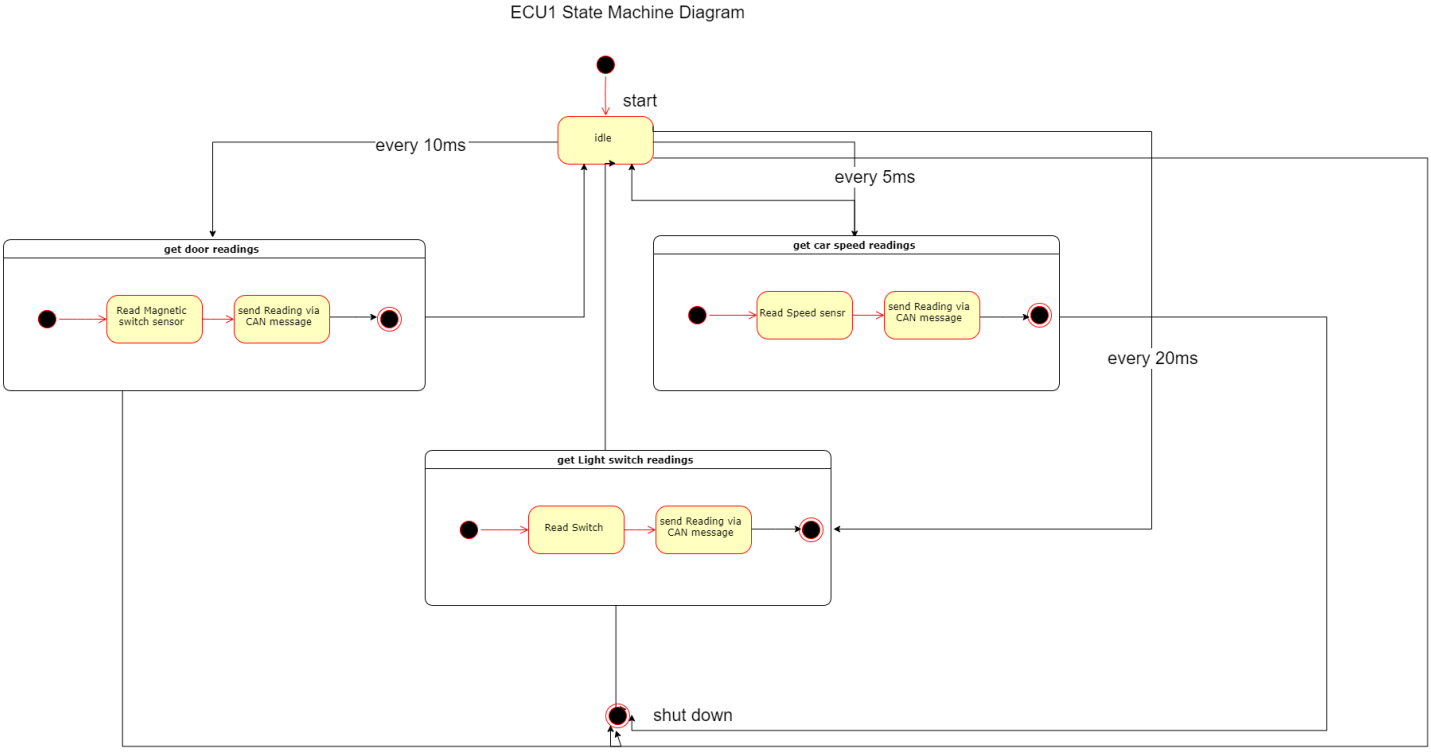




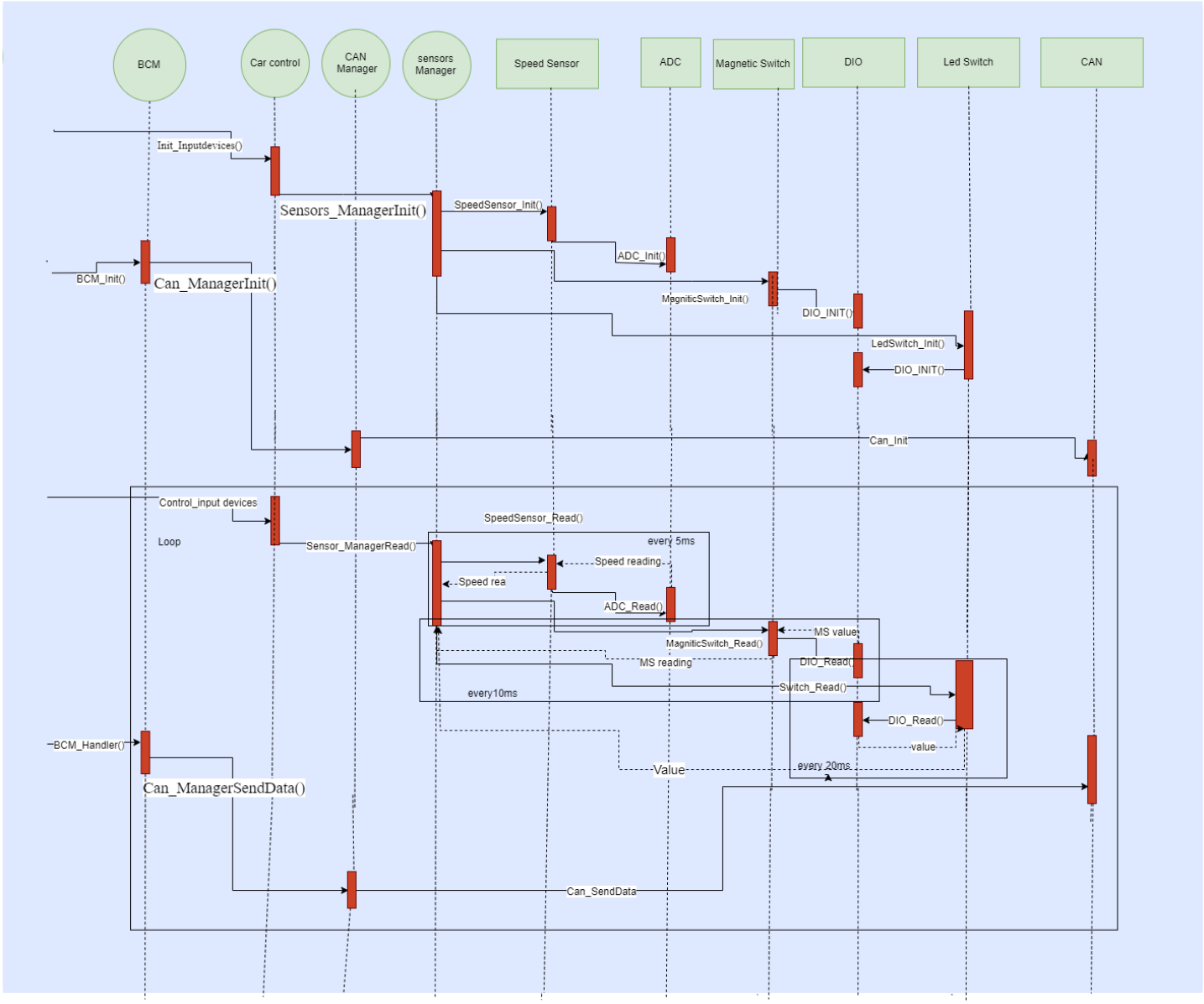
Dynamic Design

For ECU1

- State Machine

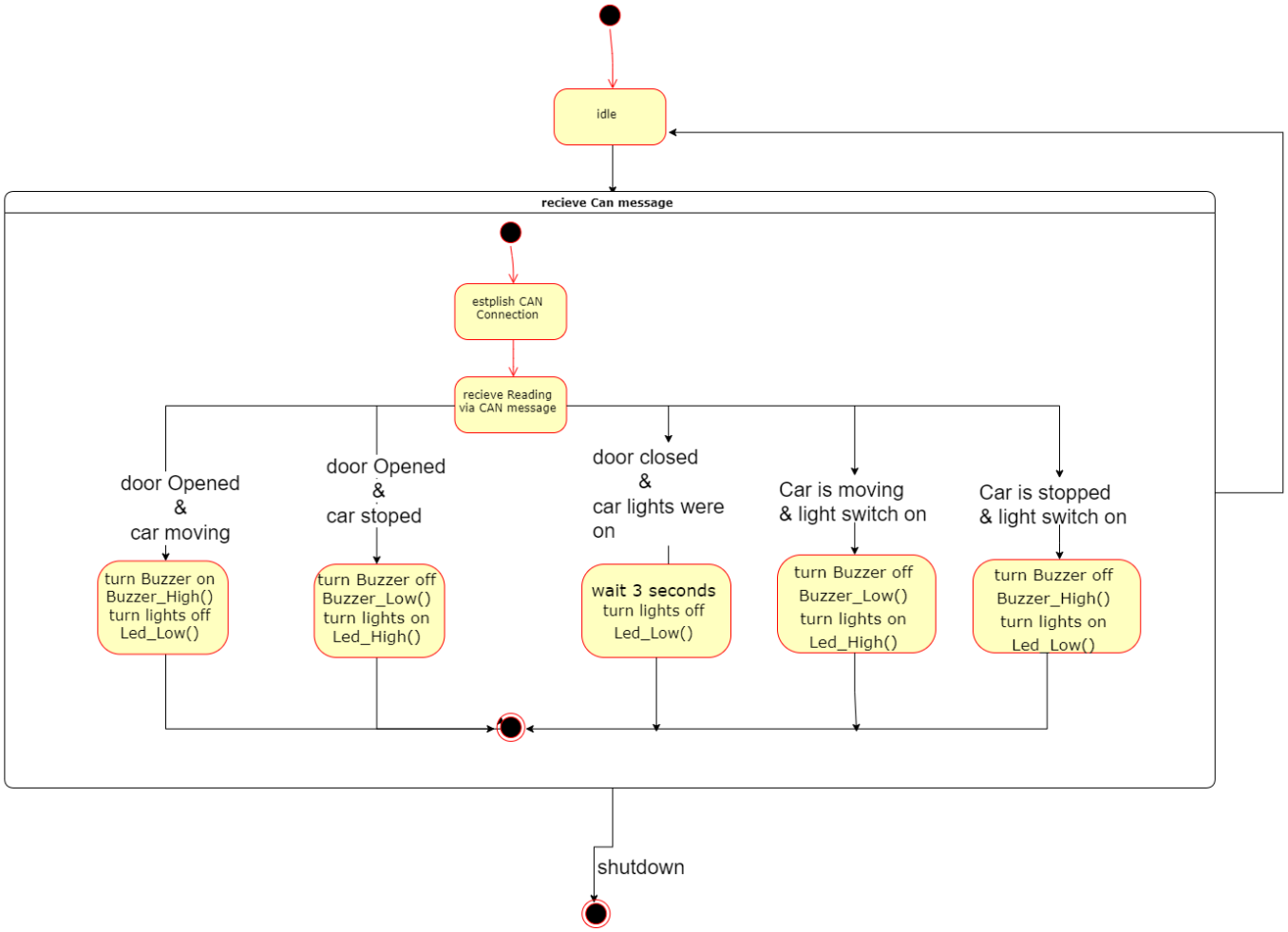


- Sequence diagram



For ECU2

- State Machine diagram



- Sequence diagram