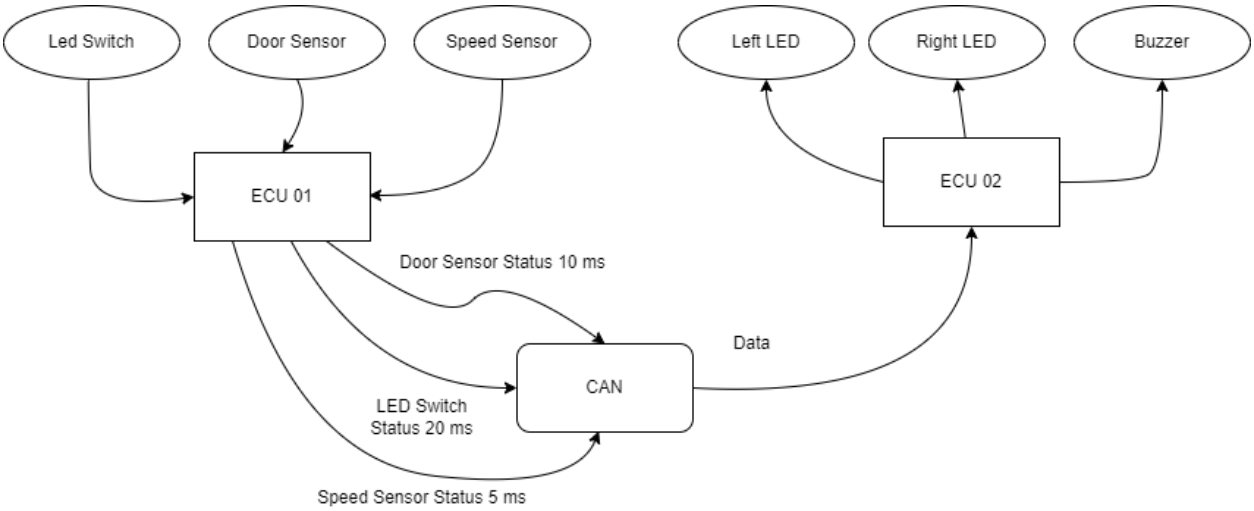
Automotive Door Control System Design

Email: msmb.mail@gmail.com

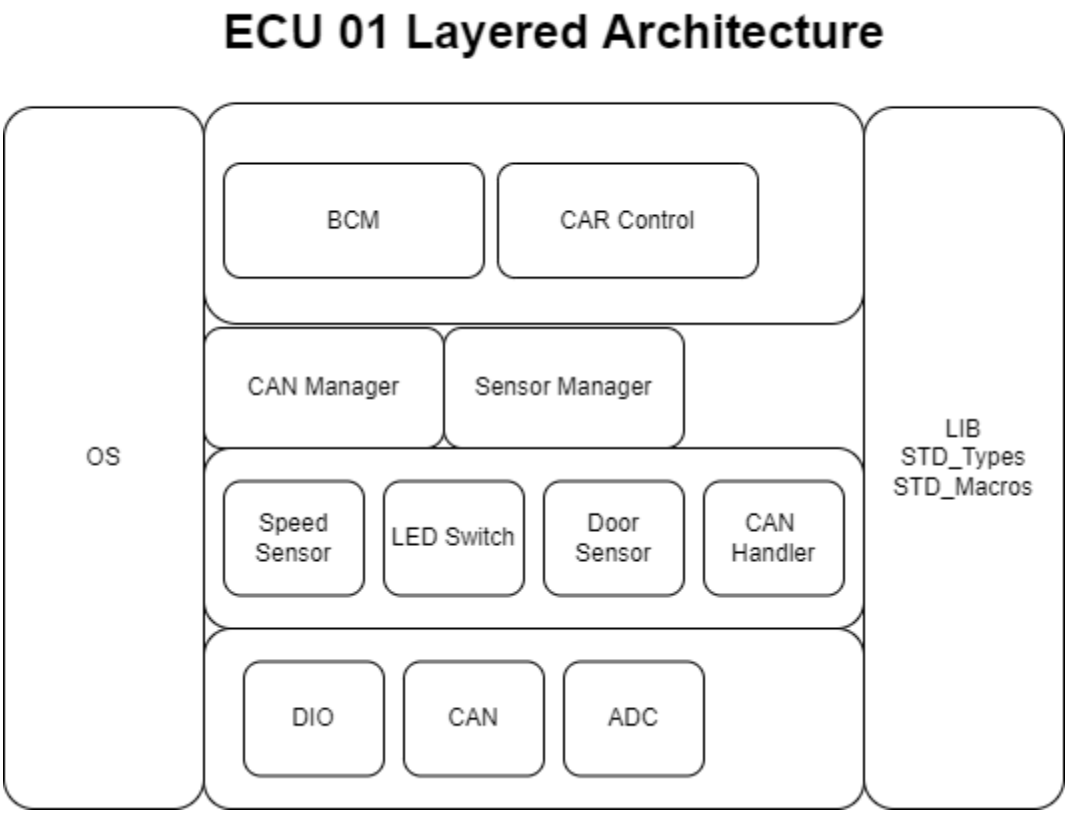
Name: Michael Samir

Static Design:

Block Diagram:



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

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

Name: DIO_WriteChannel
Arguments:
- Name: Channel Id
- 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:

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

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”

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

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:

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

- 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 ())

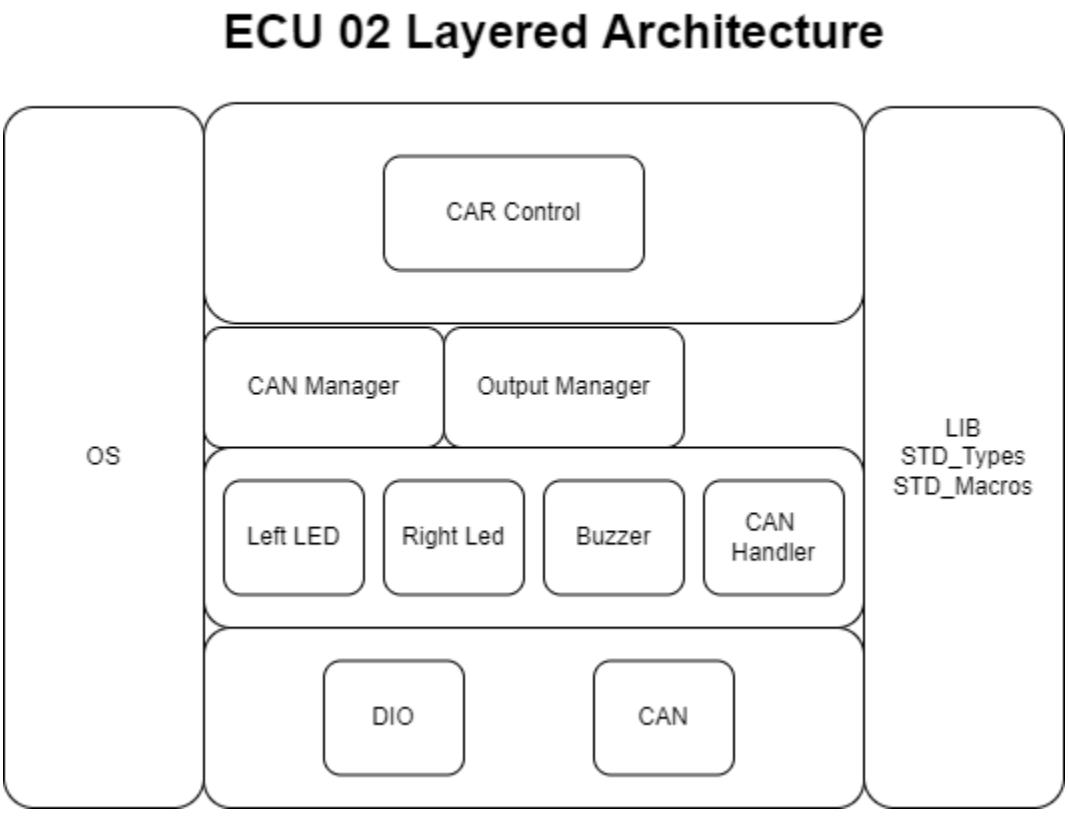
Car Control:

- Name: InputDevices_Init
- Return type: void

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

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

ECU 02 Layered Architecture



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.

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.

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

Name: CAN_Read
Return type: u16
Description: Receive data from CAN

Right LED Driver:

- void RL_Init (u8 ChannelId)

Name: RL_Init
Arguments:
- Name: ChannelId
- Type: u8
- Range: 0:10
- 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:
- Name: ChannelId
- Type: u8
- Range: 0:10
- 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:
- Name: ChannelId
- Type: u8
- Range: 0:10
- 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

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:

- Name: CANManager_Init
- Return type: void
- Description: API for initialization of communication

- void CANManager_Receive(void)

- Name: CANManager_Receive

- Return type: void
- 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())

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:

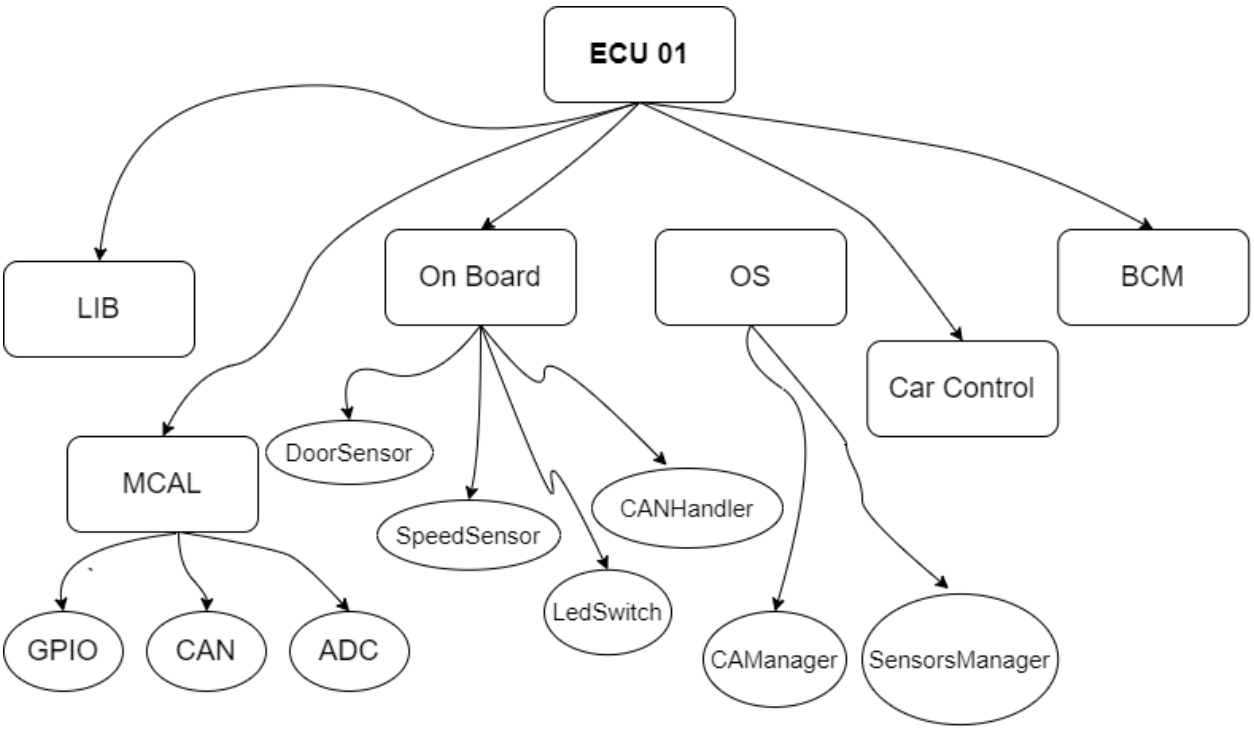
- Name: CommunicationManager_Init
- Return type: void
- Description: this API Call the OS API to initialize Communication (CANManager_Init ())

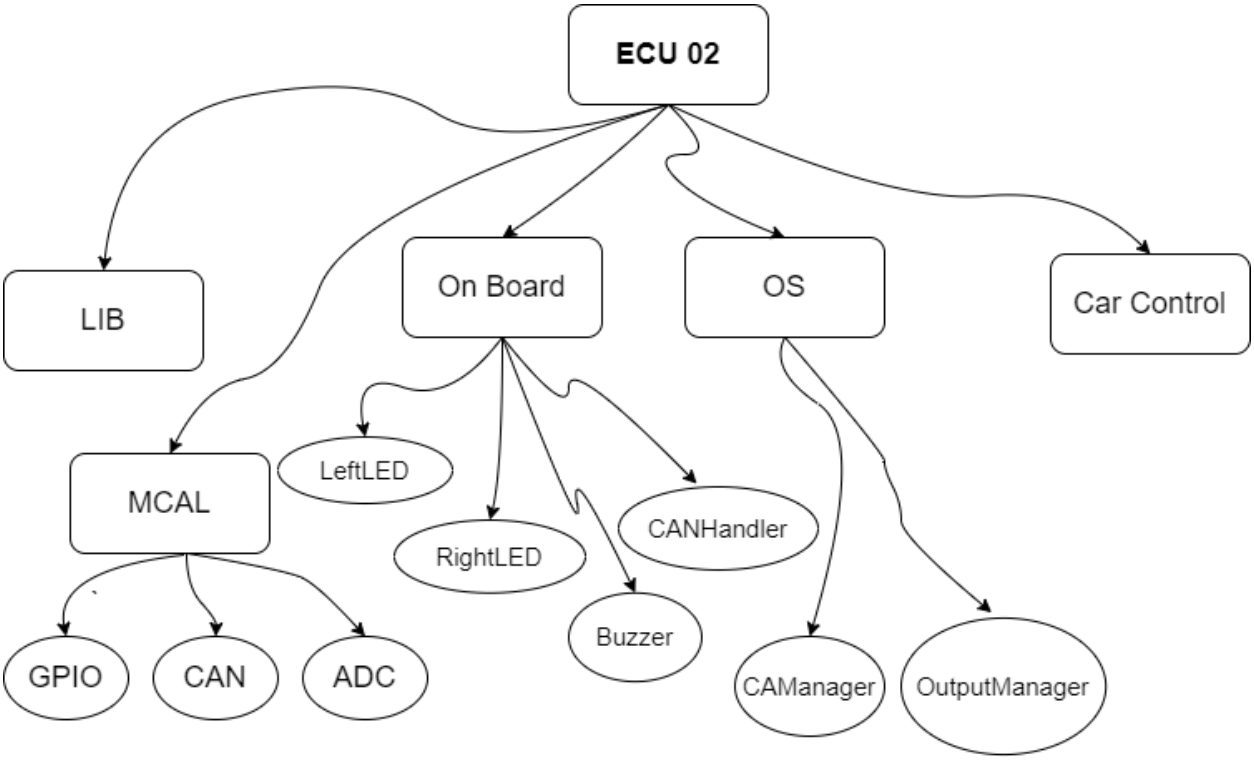
- Void ReceivingMesseges_Control(void)
- Name: ReceivingMesseges_Control
- Return type: void

- 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())

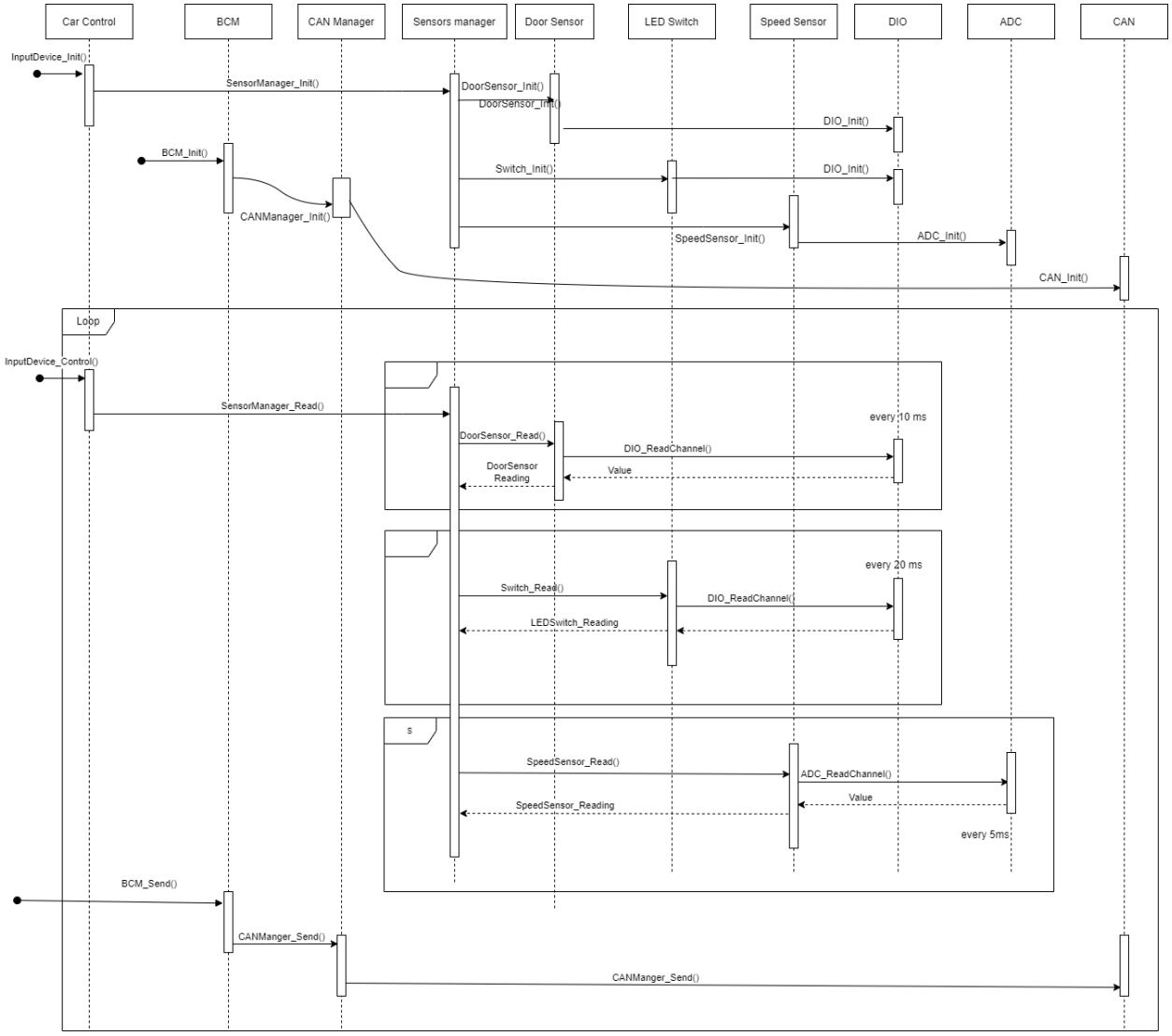
2: Folder Structure




ECU 02
LIB
On Board
OS
Car Control
MCAL
LeftLED
RightLED
CANHandler
Buzzer
GPIO
CAN
ADC
CAManager
OutputManager

3 Dynamic Design

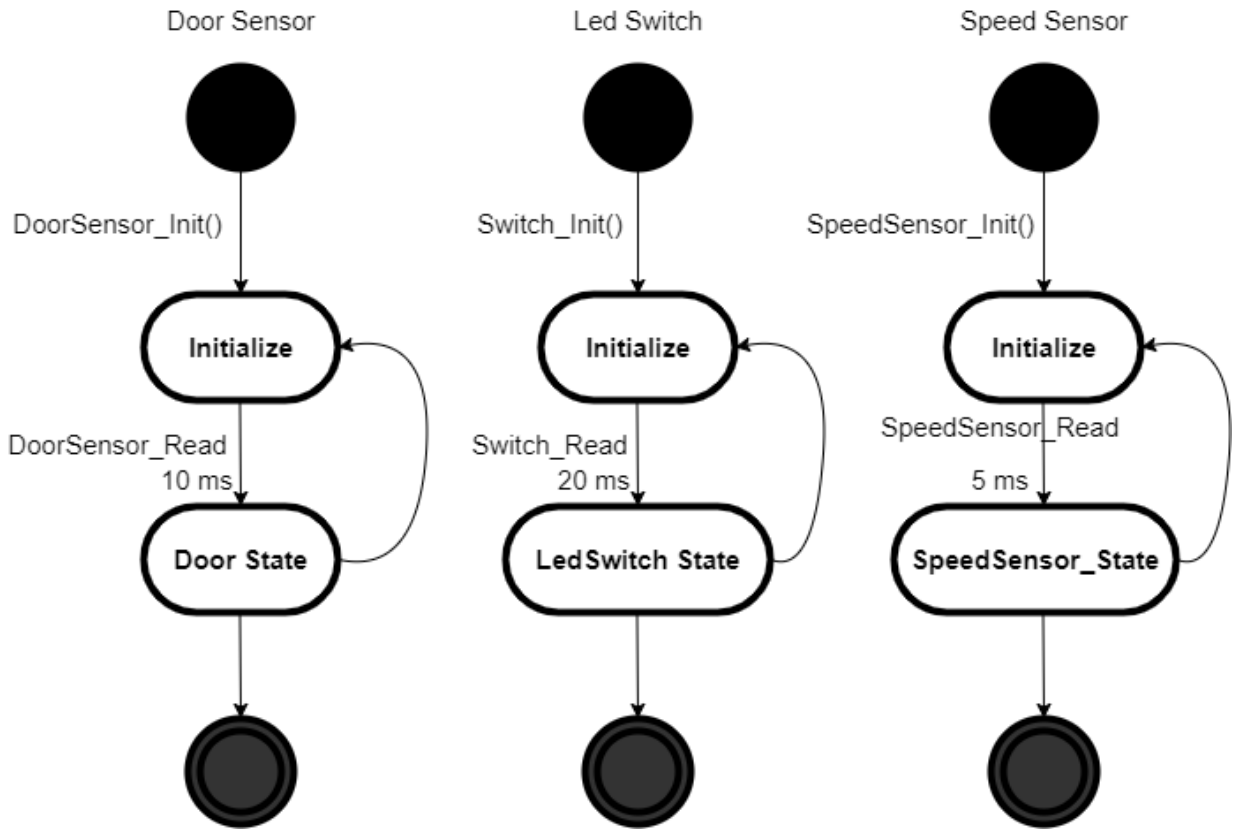
ECU 1 Sequence diagram

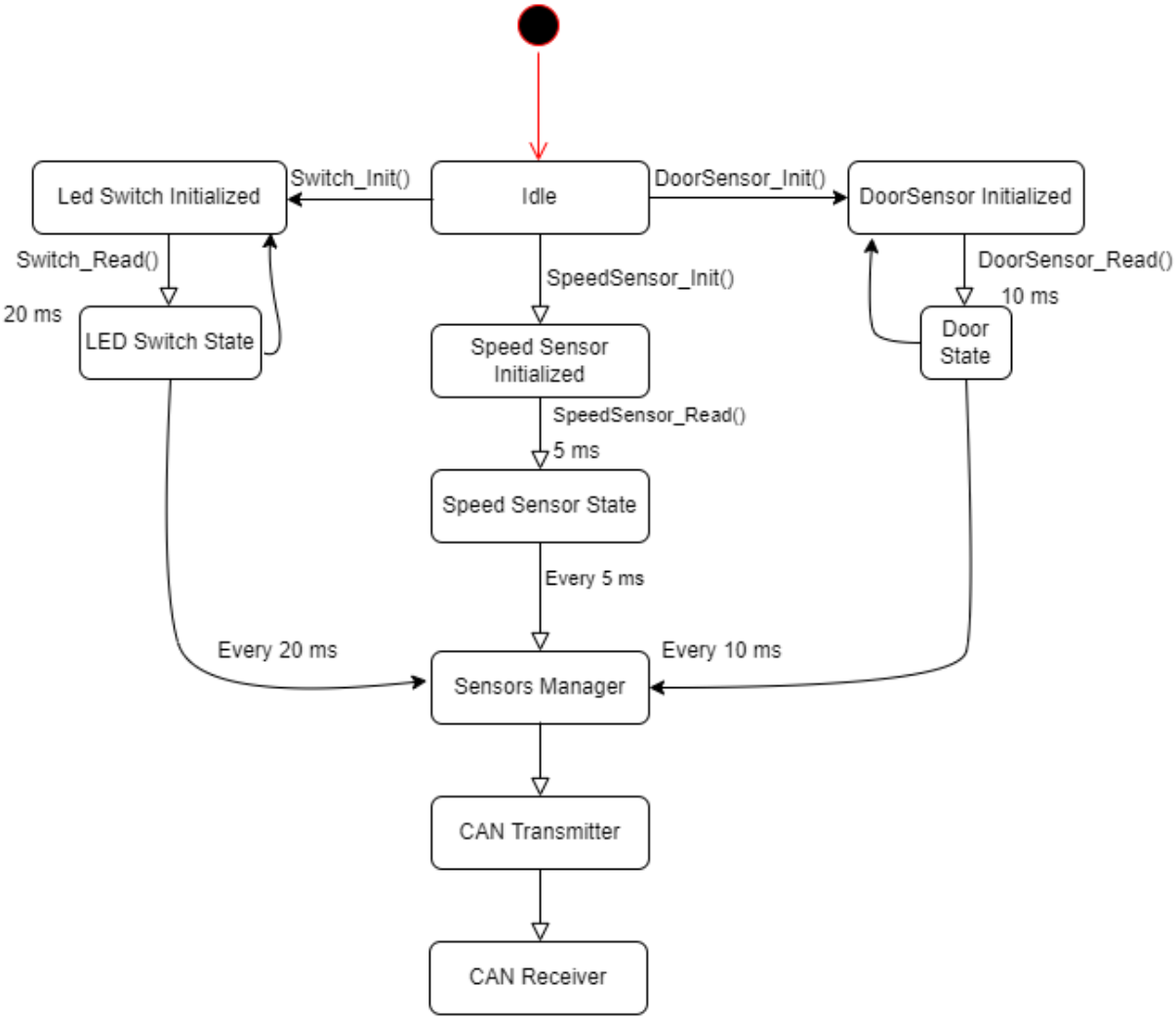


CPU Load

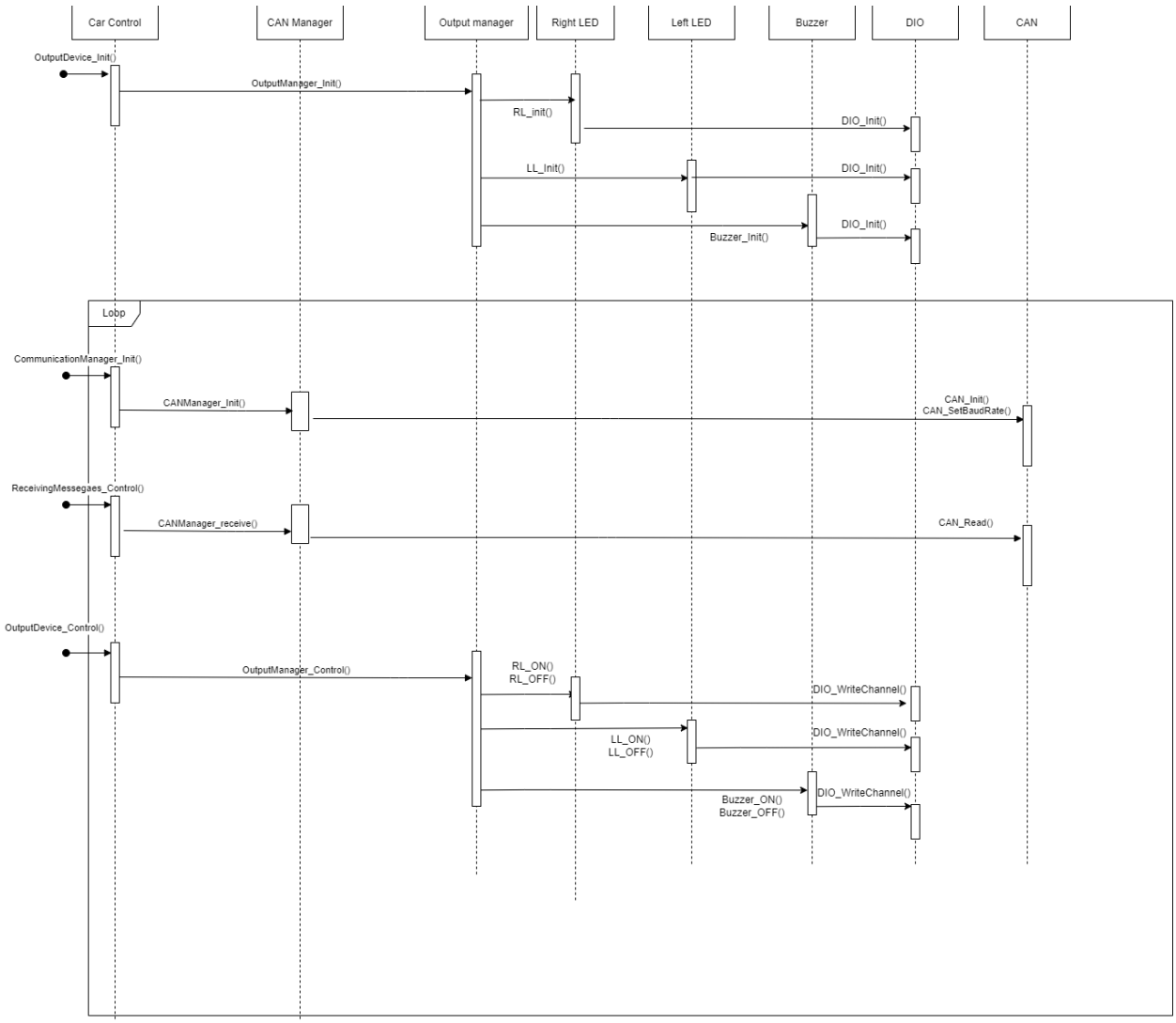
CPU Utilization = 100 - idle time = 100 - 65
CPU Utilization = 35%

State Machine




Led Switch Initialized
Switch_Init()
Idle
DoorSensor_Init()
DoorSensor Initialized
Switch_Read()
20 ms
LED Switch State
SpeedSensor_Init()
Speed Sensor Initialized
DoorSensor_Read()
10 ms
Door State
SpeedSensor_Read()
5 ms
Speed Sensor State
Every 5 ms
Every 20 ms
Every 10 ms
Sensors Manager
CAN Transmitter
CAN Receiver

ECU 2: Sequence diagram

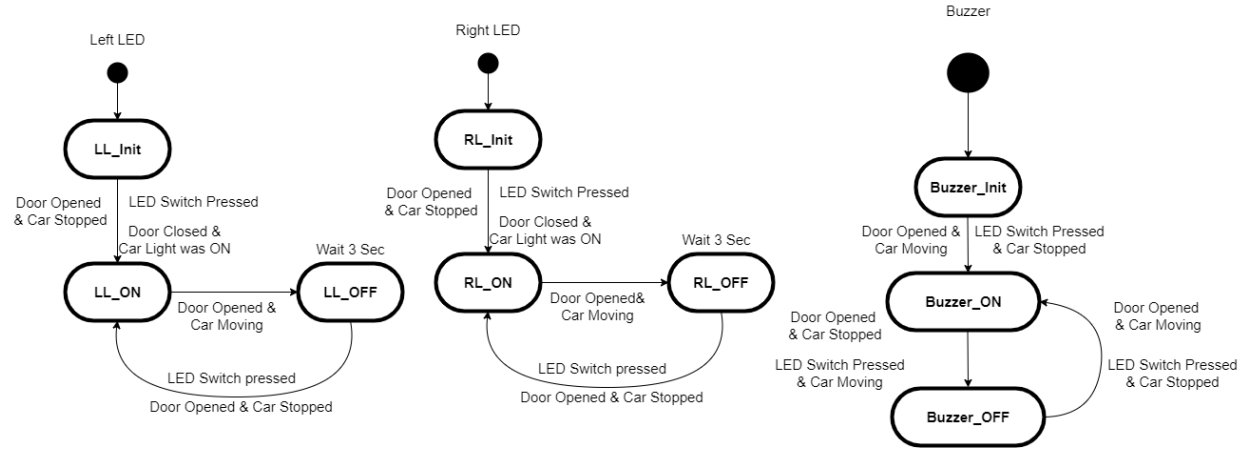


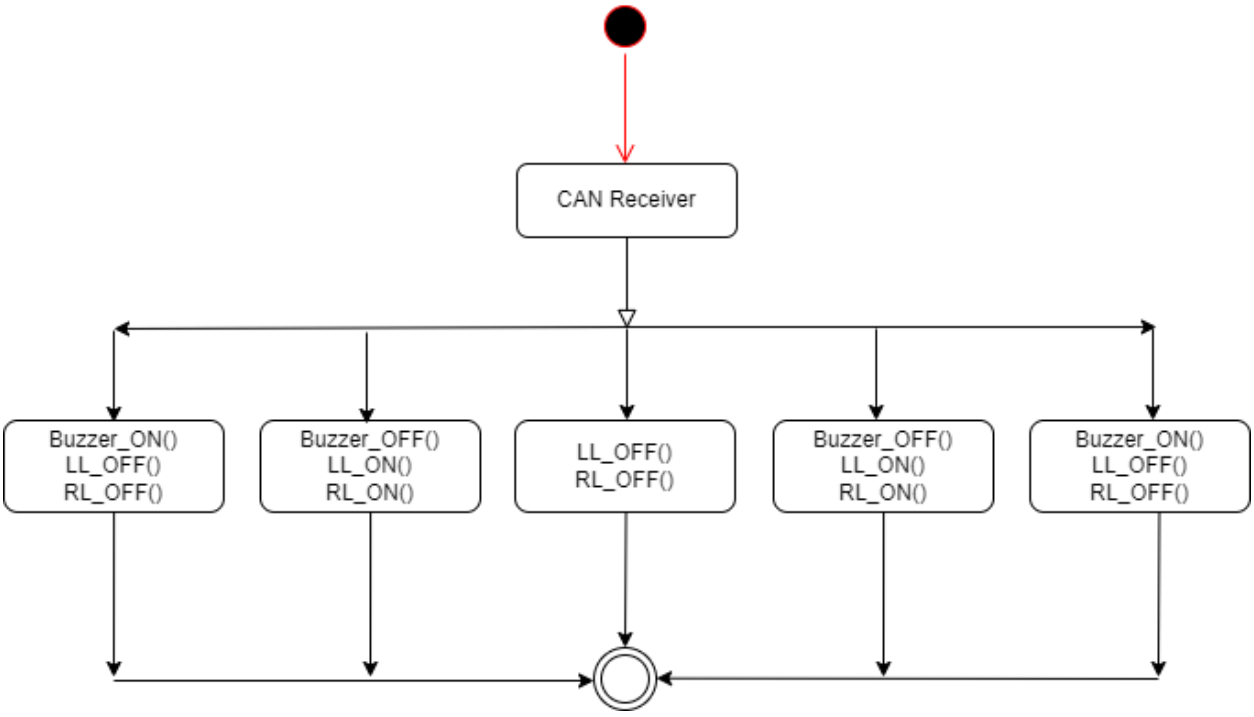
CPU Load

CPU Utilization = 100 - idle time = 100 - 65

CPU Utilization = 35%

State Machine





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 %