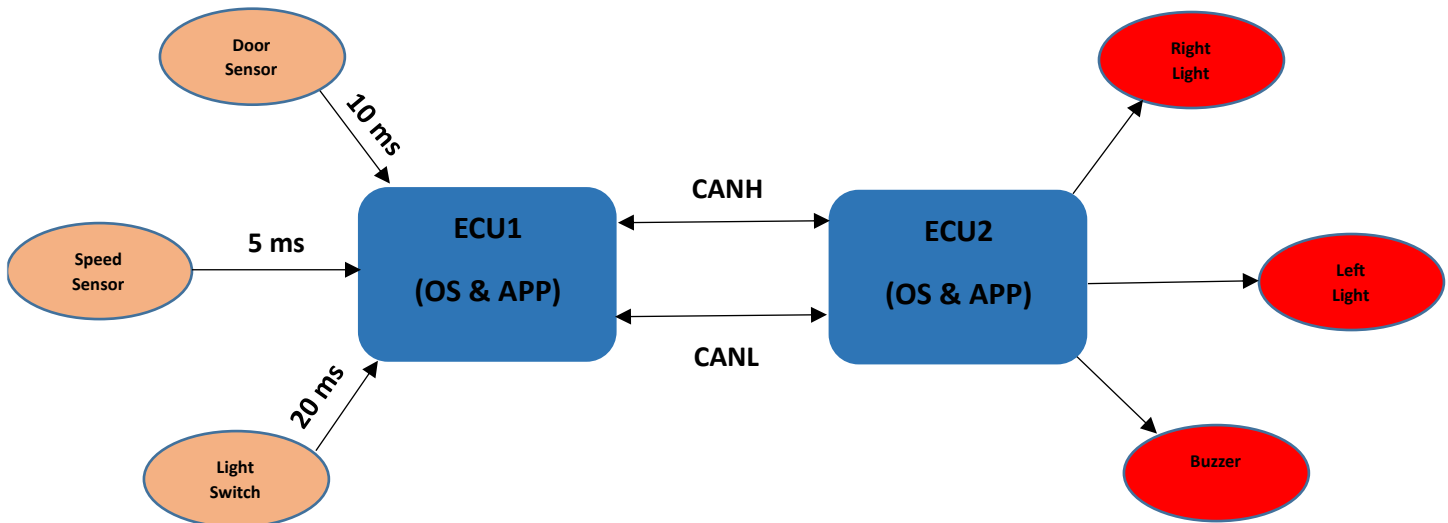


# Automotive door control system design

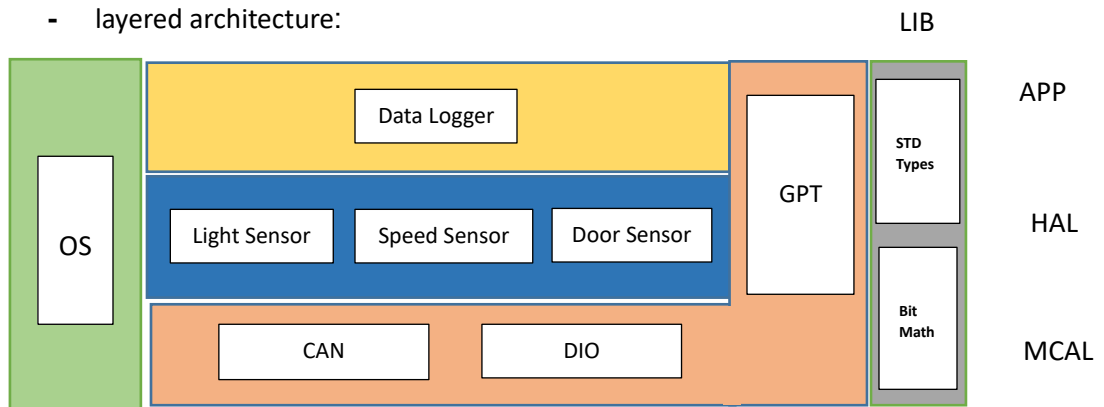
## ➤ system schematic:



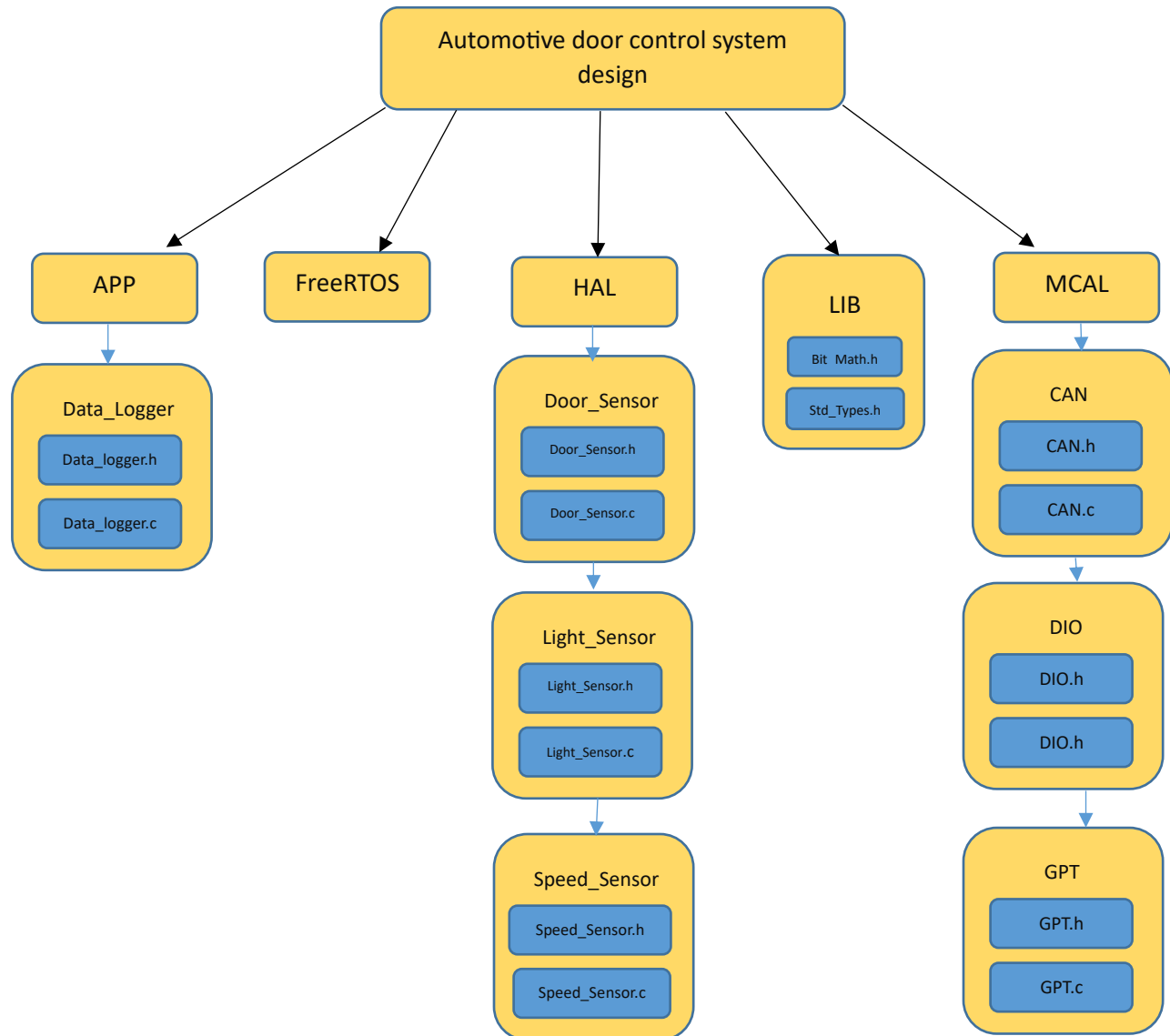
## ➤ Static design analysis:

- For ECU 1:

- layered architecture:



- folder structure :



- APIs:
- MCAL

**DIO.h**

```
typedef enum{
    PIN_0,
    PIN_1,
    PIN_2,
    PIN_3,
    PIN_4,
    PIN_5,
    PIN_6,
    PIN_7
}PIN_t;

typedef enum{
    PORT_A,
    PORT_B,
    PORT_C,
    PORT_D
}PORT_t;

typedef enum{
    INPUT,
    OUTPUT
}DIR_t;

typedef enum{
    LOW,
    HIGH
}STATUS_t;

typedef struct {
    PIN_t pin;
    PORT_t port;
    DIR_t dir;
}DIO_t;

void DIO_init(DIO_t);
void DIO_setDir(DIO_T);
void DIO_Write (DIO_t, STATUS_t);
uint_8 DIO_read(DIO_t);
```

**CAN.h**

```
#include "DIO.h"

typedef enum{
    CAN_0,
    CAN_1,
    CAN_2,
    CAN_3
}CAN_t;

typedef enum{
    HIGH_SPEED,
    LOW_SPEED
}CAN_TYPE_t;

typedef struct {
    CAN_t can;
    CAN_TYPE_t type;
}CAN_t;

void CAN_init(CAN_t);
void CAN_send (CAN_t, uint8_t);
uint8_t CAN_receive(CAN_t);
```

**GPT.h**

```
#include "DIO.h"

typedef enum{
    TIMER_0,
    TIMER_1,
    TIMER_2,
    TIMER_3
}TIMER_t;

typedef enum{
    NORMAL,
    INPUT_CAPTURE,
    PWM,
}MODE_t;

typedef enum{
    PRESCALLER_4,
    PRESCALLER_8,
    PRESCALLER_16,
    PRESCALLER_128,
    PRESCALLER_256
}PRESCALLER_t;

typedef struct {
    TIMER_t timer;
    PORT_t port;
    DIR_t dir;
}TIMER_t;

void TIMER_init(DIO_t);
void TIMER_setCallBackFunc(TIMER_T);
;
void Timer_Handler (void);
```

- HAL

**DoorSensor.h**

```
#include "DIO.h"
#include "GPT.h"
#include "CAN.h"

typedef enum{
    CLOSED,
    OPEN
}DOOR_STATE_t;

typedef struct{
    DIO_t doorPins;
    DOOR_STATE_t state;
}DOOR_t;

void DoorSensor_init(DOOR_t);
void DoorSensorGetStatus(DOOR_t);
```

**LightSensor.h**

```
#include "DIO.h"
#include "GPT.h"
#include "CAN.h"

typedef enum{
    CLOSED,
    OPEN
}Light_STATE_t;

typedef struct{
    DIO_t lightPins;
    Light_STATE_t state;
}LIGHT_t;

void lightSensor_init(LIGHT_t);
void lightSensorGetStatus(LIGHT_t);
```

**SpeedSensor.h**

```
#include "DIO.h"
#include "GPT.h"
#include "CAN.h"

typedef struct{
    DIO_t speedPins;
    Uint32_t speed;
}speed_t;

void speedSensor_init(speed_t);
void speedSensorGetStatus(speed_t);
```

- APP

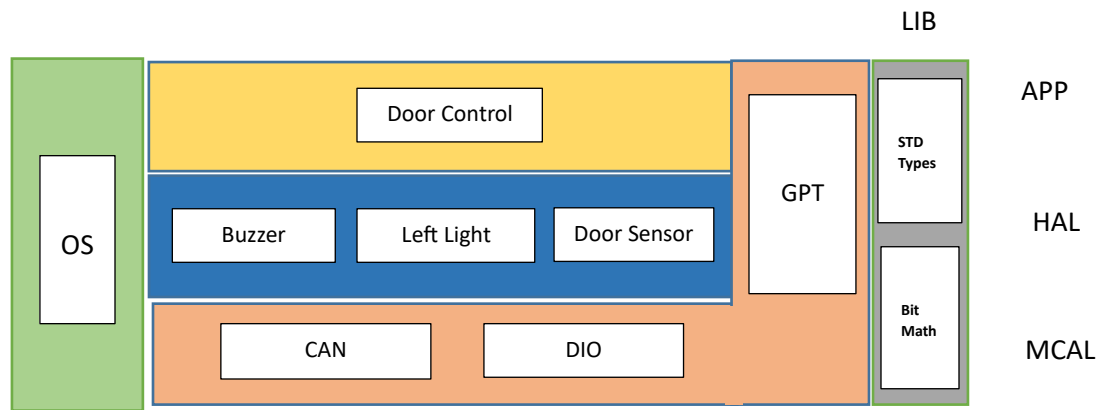
**DataLogger.h**

```
#include " DoorSensor.h"
#include " LightSensor.h"
#include " SpeedSensor.h"

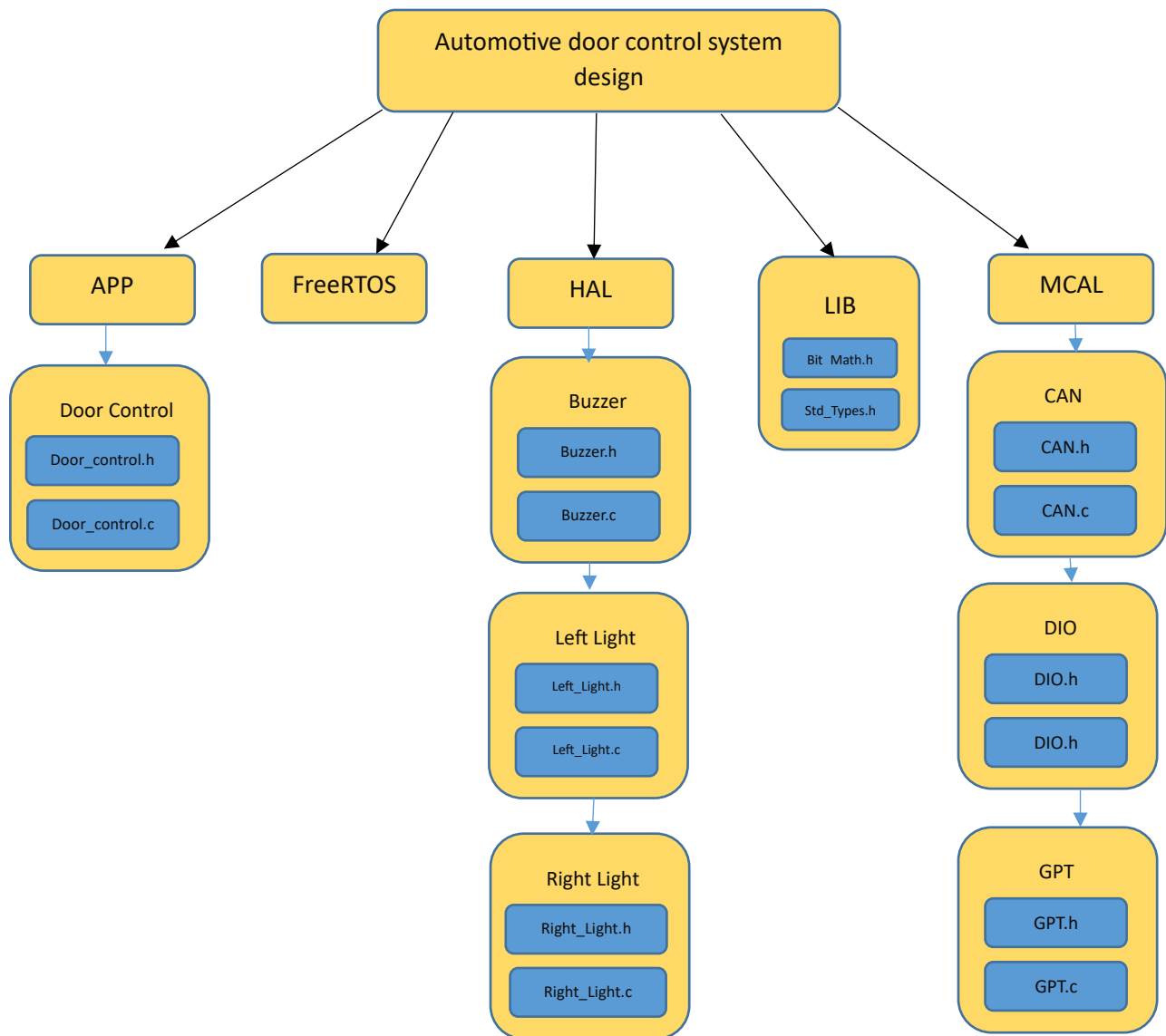
typedef struct{
    DOOR_t door;
    LIGHT_t light;
    speed_t light;
}dataLogger_t;

void DataLogger_init(dataLogger_t);
void GetSensorsReadings(dataLogger_t);
void Data_Send(dataLogger_t);
```

- For ECU 2:



- folder structure :



- APIs:
- MCAL

**DIO.h**

```
typedef enum{
    PIN_0,
    PIN_1,
    PIN_2,
    PIN_3,
    PIN_4,
    PIN_5,
    PIN_6,
    PIN_7
}PIN_t;

typedef enum{
    PORT_A,
    PORT_B,
    PORT_C,
    PORT_D
}PORT_t;

typedef enum{
    INPUT,
    OUTPUT
}DIR_t;

typedef enum{
    LOW,
    HIGH
}STATUS_t;

typedef struct {
    PIN_t pin;
    PORT_t port;
    DIR_t dir;
}DIO_t;

void DIO_init(DIO_t);
void DIO_setDir(DIO_T);
void DIO_Write (DIO_t, STATUS_t);
uint_8 DIO_read(DIO_t);
```

**CAN.h**

```
#include "DIO.h"

typedef enum{
    CAN_0,
    CAN_1,
    CAN_2,
    CAN_3
}CAN_t;

typedef enum{
    HIGH_SPEED,
    LOW_SPEED
}CAN_TYPE_t;

typedef struct {
    CAN_t can;
    CAN_TYPE_t type;
}CAN_t;

void CAN_init(CAN_t);
void CAN_send (CAN_t, uint8_t);
uint8_t CAN_receive(CAN_t);
```

**GPT.h**

```
#include "DIO.h"

typedef enum{
    TIMER_0,
    TIMER_1,
    TIMER_2,
    TIMER_3
}TIMER_t;

typedef enum{
    NORMAL,
    INPUT_CAPTURE,
    PWM,
}MODE_t;

typedef enum{
    PRESCALLER_4,
    PRESCALLER_8,
    PRESCALLER_16,
    PRESCALLER_128,
    PRESCALLER_256
}PRESCALLER_t;

typedef struct {
    TIMER_t timer;
    PORT_t port;
    DIR_t dir;
}TIMER_t;

void TIMER_init(DIO_t);
void TIMER_setCallBackFunc(TIMER_T);
;
void Timer_Handler (void);
```

- HAL

**Buzzer.h**

```
#include "DIO.h"
#include "GPT.h"
#include "CAN.h"

typedef struct{
    DIO_t BuzzerPins;
}BUZZER_t;

void Buzzer_init(BUZZER _t);
void Buzzer_ON(BUZZER _t);
void Buzzer_OFF(BUZZER_T);
```

**Left\_Light.h**

```
#include "DIO.h"
#include "GPT.h"
#include "CAN.h"

typedef struct{
    DIO_t LeftLightPins;
}LeftLight_t;

void LeftLight_init(LeftLight_t);
void LeftLight_ON(LeftLight_t);
void LeftLight_OFF(LeftLight_T);
```

**Right\_Light.h**

```
#include "DIO.h"
#include "GPT.h"
#include "CAN.h"

typedef struct{
    DIO_t LeftLightPins;
}RightLight_t;

void RightLight_init(RightLight _t);
void RightLight_ON(RightLight _t);
void RightLight_OFF(RightLight_T);
```

- APP

**Door\_control.**

```
#include " DoorSensor.h"
#include " LightSensor.h"
#include " SpeedSensor.h"

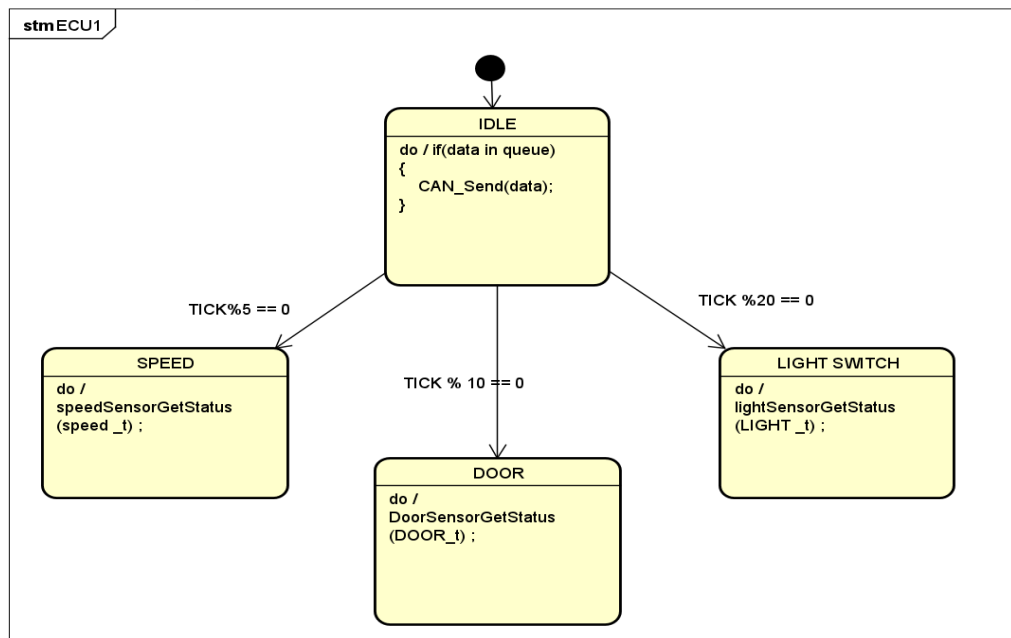
typedef struct{
    DOOR_t door;
    LIGHT_t light;
    speed _t light;
}dataLogger _t;

void DataLogger_init(dataLogger _t);
void GetSensorsReadings(dataLogger _t);
void Data_Send(dataLogger _t);
```

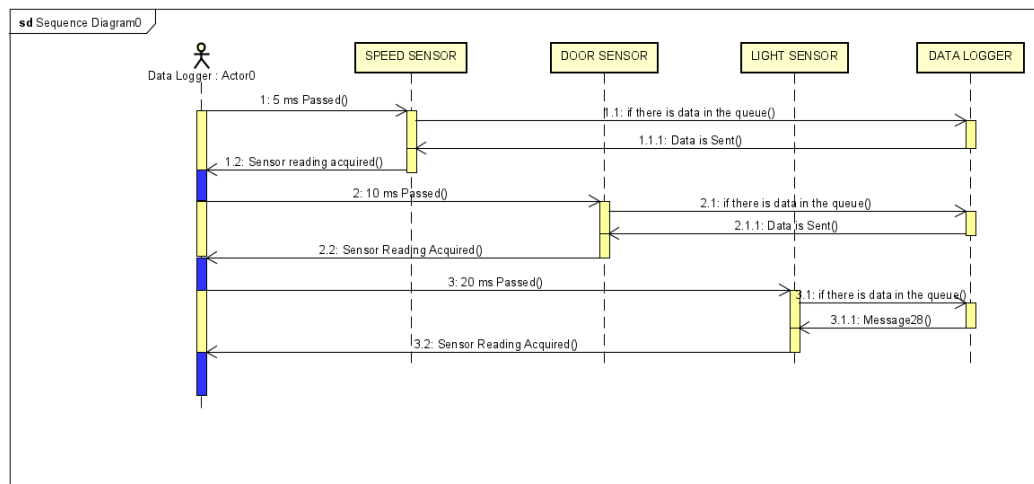
## ➤ Dynamic design analysis:

- ECU 1 :

- state machine diagram:



- sequence diagram:

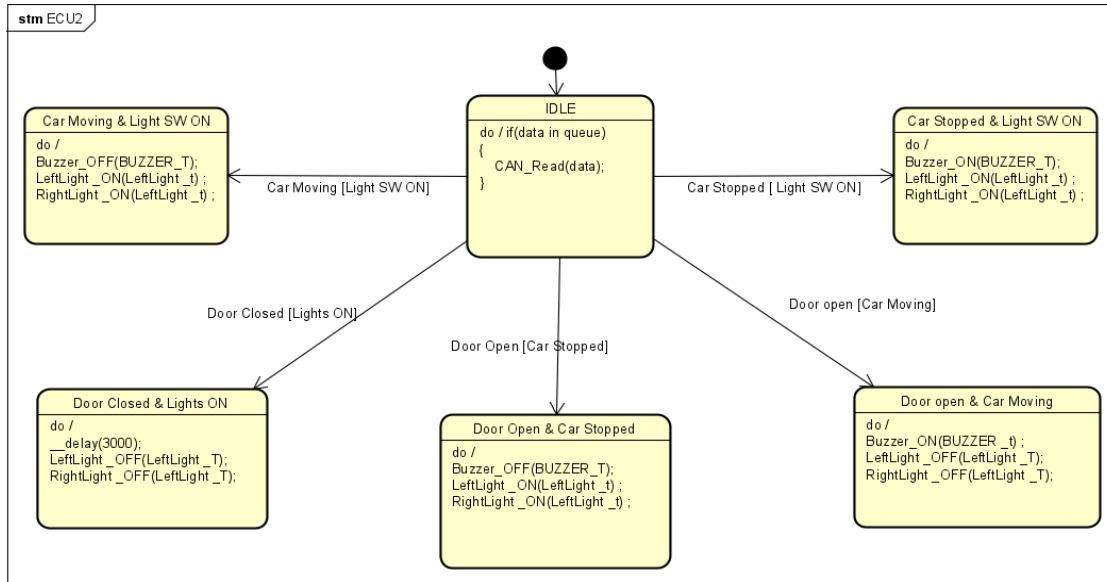


- CPU load : Assuming ( Task 1 period = 2ms , Task2 period = 2ms , Task3 period = 2ms)  

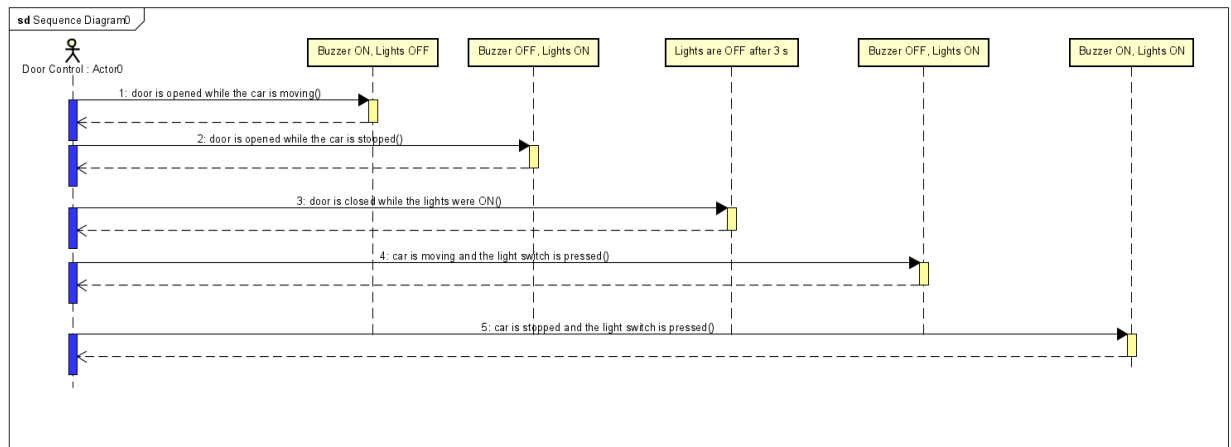
$$U = (2 + 2 + 2) / (\text{hyper-period} = 20) = 6/20 = 30\%$$

- ECU 2 :

- state machine diagram:



- sequence diagram:



- CPU load : Assuming ( each task has an execution time of 2 ms & each task will occur once in a hyper period of 20 ms)

$$U = (2 + 2 + 2 + 2 + 2) / (\text{hyper-period} = 20) = 10/20 = 50\%$$