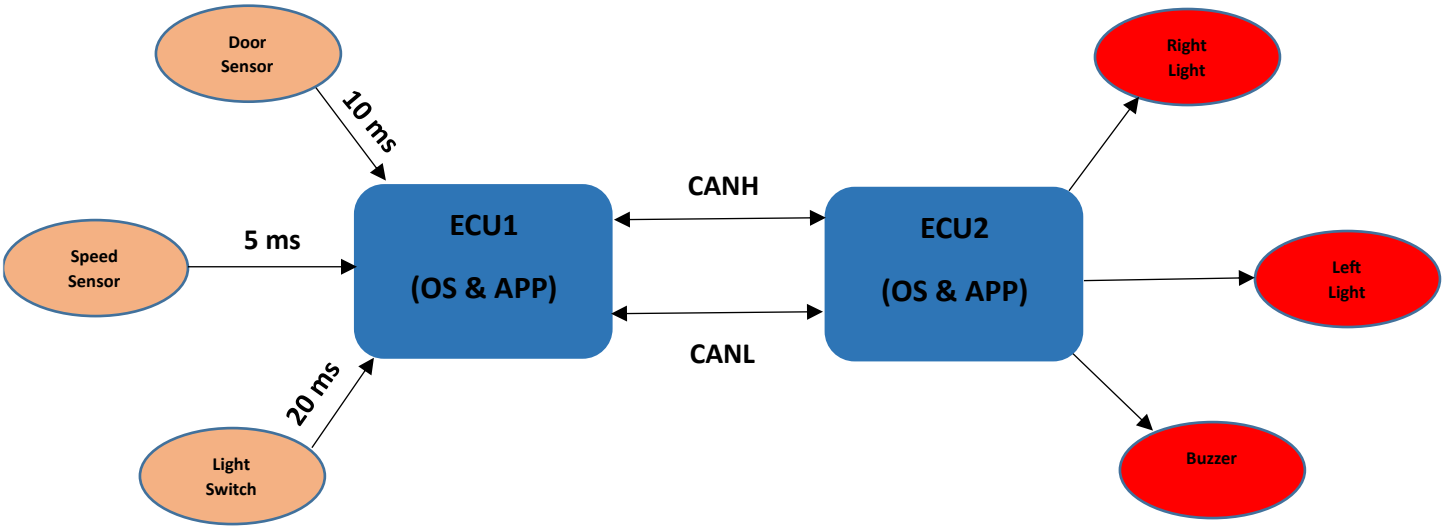
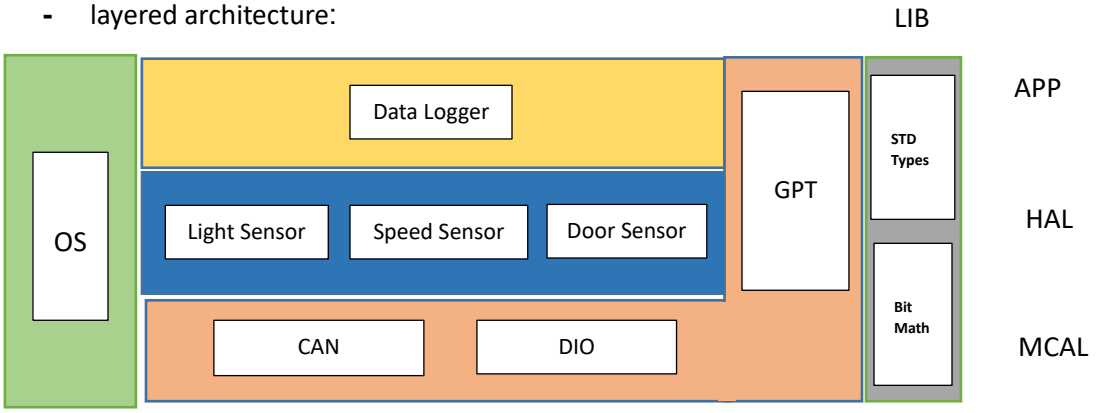
Automotive door control system design

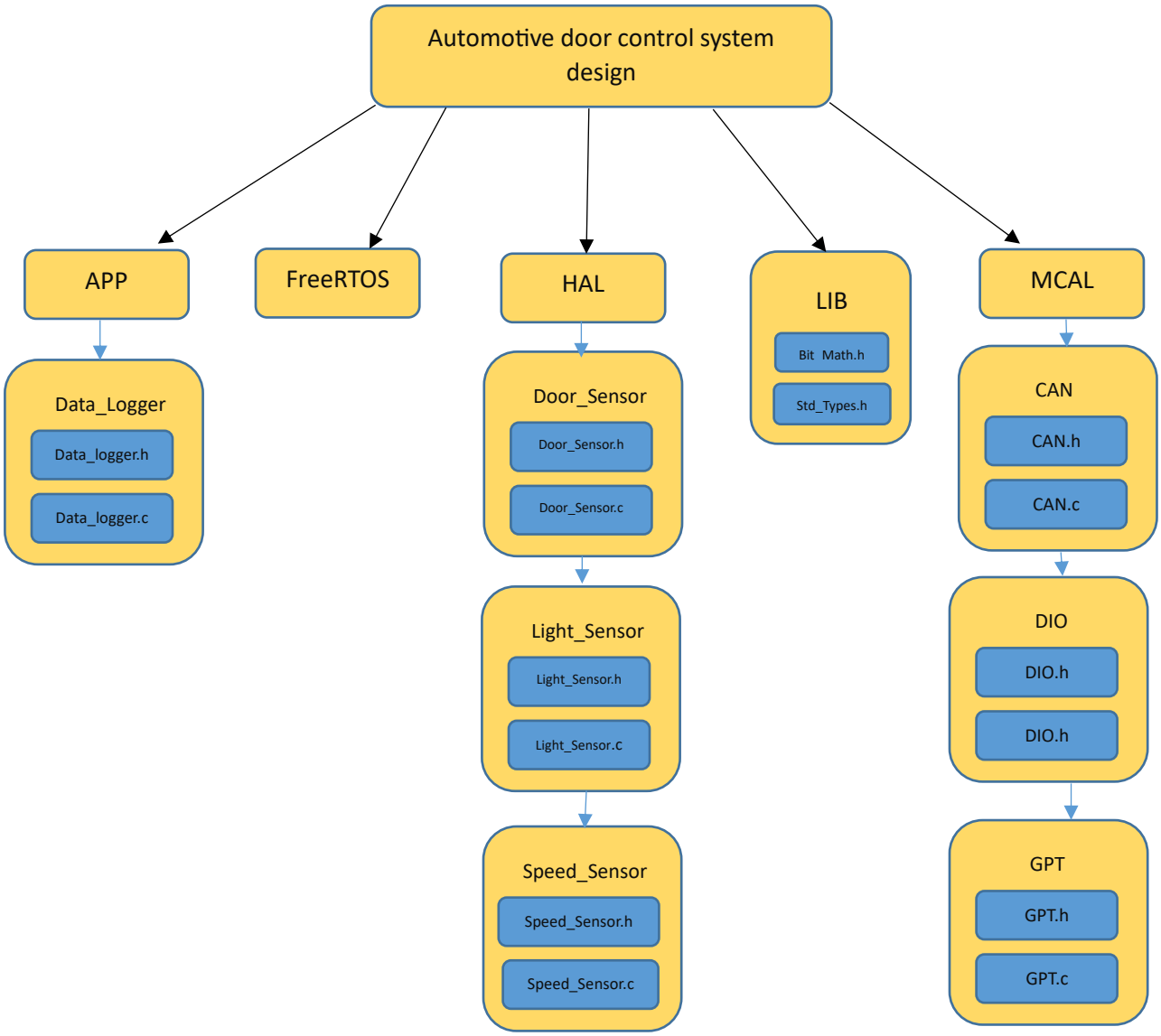
- system schematic:



- 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; // port pins

typedef enum{
   PORT_A,
   PORT_B,
   PORT_C,
   PORT_D
}PORT_t;  // port names

typedef enum{
   INPUT,
   OUTPUT
}DIR_t; // pin direction

typedef enum{
   LOW,
   HIGH
}STATUS_t; // pin status
typedef struct  {
   PIN_t pin;
   PORT_t port ;
   DIR_t dir ;
   }DIO_t;   // pin configuration

/* Function to initialize the a specific pin
It takes struct of type DIO_t and return
nothing */
void DIO_init(DIO_t); //pin initialization

/* Function to  write high or low to
specific pin It takes struct of type DIO_t
and return nothing */
void DIO_Write (DIO_t, STATUS_t);

/* Function to  read a pin status It takes
struct of type DIO_t and return uint_8 */
uint_8 DIO_read(DIO_t) ;
```

CAN.h

```
#include "DIO.h"

typedef enum{
   CAN_0,
   CAN_1,
   CAN_2,
   CAN_3
}CAN_t;  // CAN channel

typedef enum{
   HIGH_SPEED,
   LOW_SPEED
}CAN_TYPE_t; // CAN operating mode

typedef struct  {
   CAN_t can;
   CAN_TYPE_t type ;
   }CAN_t;  // CAN configuration

/* Function to initialize the can bus It
takes struct of type CAN_t and return
nothing */
void CAN_init(CAN_t); // initialization

/* Function to send single byte using
CAN bus It takes struct of type CAN _t
and return nothing */
void CAN_send (CAN_t, uint8_t);

/* Function to initialize the a specific
pin It takes struct of type DIO_t and
return nothing */
uint8_t CAN_receive(CAN_t) ;
```

GPT.h

```
#include "DIO.h"

typedef enum{
   TIMER_0,
   TIMER_1,
   TIMER_2,
   TIMER_3
}TIMER_t; //timer channel

typedef enum{
   NORMAL,
   INPUT_CAPTURE,
   PWM,
}MODE_t;  // timer mode

typedef enum{
   PRESCALLER_4,
   PRESCALLER_8,
   PRESCALLER_16,
   PRESCALLER_128,
   PRESCALLER_256

}PRESCALLER_t;  // timer prescaller
typedef struct  {
   TIMER_t timer;
   PORT_t port ;
   DIR_t dir ;
   }TIMER_t;  // timer configuration

/* Function to initialize the timer  It takes
struct of type TIMER _t and return
nothing */
void TIMER_init(DIO_t);

/* Function to  set the callback function It
takes struct of type TIMER _t and return
nothing */
void TIMER_setCallBackFunc(TIMER_T) ;

/* Timer interrupt handler nothing  and
return nothing */
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;

/* Function initialize the door sensor
It takes struct of type DOOR _t and
return nothing */
void DoorSensor_init(DOOR_t);

/* Function to get the door status It
takes nothing and return enum type
DOOR_STATE_t */

DOOR_STATE_t
DoorSensorGetStatus(void) ;
```

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;

/* Function initialize the light sensor  It
takes struct of type LIGHT _t and
return nothing */
void lightSensor_init(LIGHT _t);

/* Function to get the light status It
takes struct of type LIGHT _t and
return enum type Light_STATE_t
Light_STATE_t
lightSensorGetStatus(LIGHT _t) ;
```

SpeedSensor.h

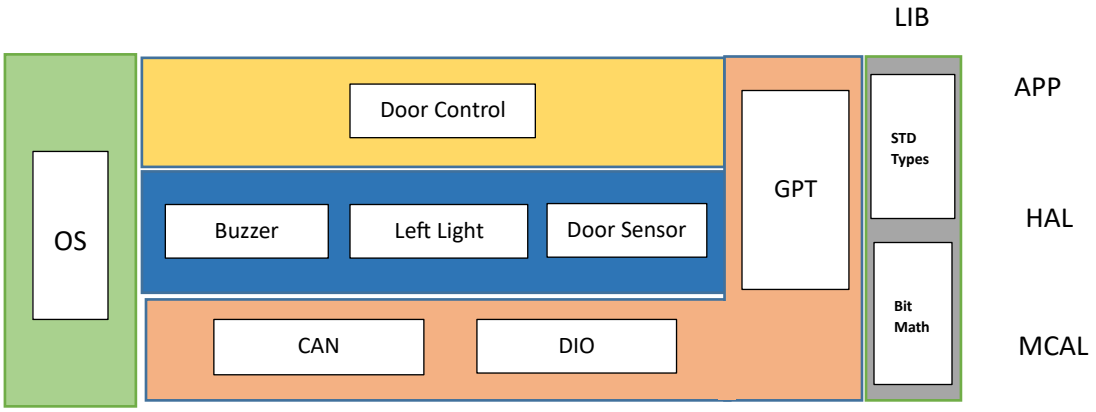
```
#include "DIO.h"
#include "GPT.h"
#include "CAN.h"


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

/* Function initialize the speed sensor
It takes struct of type speed _t and
return nothing */
void speedSensor_init(speed _t);

/* Function initialize the speed sensor
It takes struct of type speed _t and
return speed value */
uint32_T speedSensorGetval(speed _t)
```

- APP

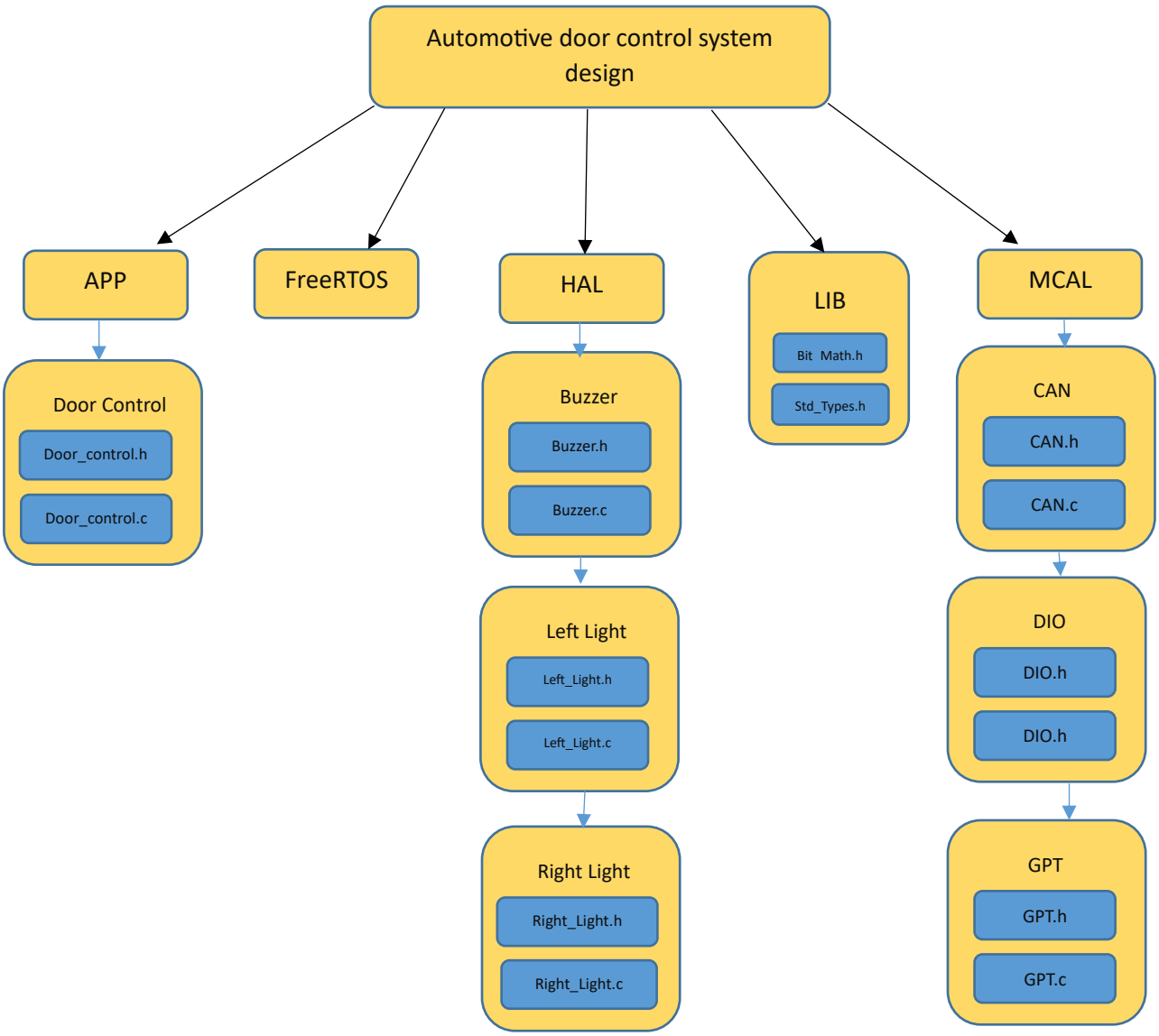
DataLogger.h

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


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


/* Function initialize the data logger It takes struct of
type dataLogger _t and return nothing */
void DataLogger_init(dataLogger _t);

/* Function to get the data logger readings It takes
struct of type dataLogger _t and return nothing */
void GetSensorsReadings(dataLogger _t) ;

/* Function to send the data logger readings It takes
struct of type dataLogger _t and return nothing */
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; // port pins

typedef enum{
   PORT_A,
   PORT_B,
   PORT_C,
   PORT_D
}PORT_t;  // port names

typedef enum{
   INPUT,
   OUTPUT
}DIR_t; // pin direction

typedef enum{
   LOW,
   HIGH
}STATUS_t; // pin status
typedef struct  {
   PIN_t pin;
   PORT_t port ;
   DIR_t dir ;
   }DIO_t;   // pin configuration

/* Function to initialize the a specific pin
It takes struct of type DIO_t and return
nothing */
void DIO_init(DIO_t); //pin initialization

/* Function to  write high or low to
specific pin It takes struct of type DIO_t
and return nothing */
void DIO_Write (DIO_t, STATUS_t);

/* Function to  read a pin status It takes
struct of type DIO_t and return uint_8 */
uint_8 DIO_read(DIO_t) ;
```

CAN.h

```
#include "DIO.h"

typedef enum{
   CAN_0,
   CAN_1,
   CAN_2,
   CAN_3
}CAN_t;  // CAN channel

typedef enum{
   HIGH_SPEED,
   LOW_SPEED
}CAN_TYPE_t; // CAN operating mode

typedef struct  {
   CAN_t can;
   CAN_TYPE_t type ;
   }CAN_t;  // CAN configuration

/* Function to initialize the can bus It
takes struct of type CAN_t and return
nothing */
void CAN_init(CAN_t); // initialization

/* Function to send single byte using
CAN bus It takes struct of type CAN _t
and return nothing */
void CAN_send (CAN_t, uint8_t);

/* Function to initialize the a specific
pin It takes struct of type DIO_t and
return nothing */
uint8_t CAN_receive(CAN_t) ;
```

GPT.h

```
#include "DIO.h"

typedef enum{
   TIMER_0,
   TIMER_1,
   TIMER_2,
   TIMER_3
}TIMER_t; //timer channel

typedef enum{
   NORMAL,
   INPUT_CAPTURE,
   PWM,
}MODE_t;  // timer mode

typedef enum{
   PRESCALLER_4,
   PRESCALLER_8,
   PRESCALLER_16,
   PRESCALLER_128,
   PRESCALLER_256

}PRESCALLER_t;  // timer prescaller
typedef struct  {
   TIMER_t timer;
   PORT_t port ;
   DIR_t dir ;
   }TIMER_t;  // timer configuration

/* Function to initialize the timer  It takes
struct of type TIMER _t and return
nothing */
void TIMER_init(DIO_t);

/* Function to  set the callback function It
takes struct of type TIMER _t and return
nothing */
void TIMER_setCallBackFunc(TIMER_T) ;

/* Timer interrupt handler nothing  and
return nothing */
void Timer_Handler (void);
```

  - HAL

Buzzer.h

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

typedef struct{
   DIO_t BuzzerPins;
}BUZZER_t;

/* Function to  intialize the Buzzer It
takes struct of type BUZZER _t and
return nothing */
void Buzzer_init(BUZZER _t);

/* Function to  turn  the Buzzer on It
takes struct of type BUZZER _t and
return nothing */
void Buzzer_ON(BUZZER _t) ;

/* Function to  turn  the Buzzer off It
takes struct of type BUZZER _t and
return nothing */
void Buzzer_OFF(BUZZER_T);
```

Left_Light.h

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


typedef struct{
   DIO_t LeftLightPins;
}LeftLight_t;

/* Function to  intialize the LeftLight It
takes struct of type LeftLight _t and
return nothing */
void LeftLight_init(LeftLight_t);

/* Function to  turn  the LeftLight on It
takes struct of type LeftLight _t and
return nothing */
void LeftLight _ON(LeftLight _t) ;

/* Function to  turn  the LeftLight off It
takes struct of type LeftLight _t and
return nothing */
void LeftLight _OFF(LeftLight _T);
```

Right_Light.h

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

typedef struct{
   DIO_t LeftLightPins;
}RightLight_t;

/* Function to  intialize the RightLight
It takes struct of type RightLight _t and
return nothing */
void RightLight _init(RightLight _t);

/* Function to  turn  the RightLight on
It takes struct of type RightLight _t and
return nothing */
void RightLight _ON(RightLight _t) ;

/* Function to  turn  the RightLight off
It takes struct of type RightLight _t and
return nothing */
void RightLight _OFF(RightLight _T);
```

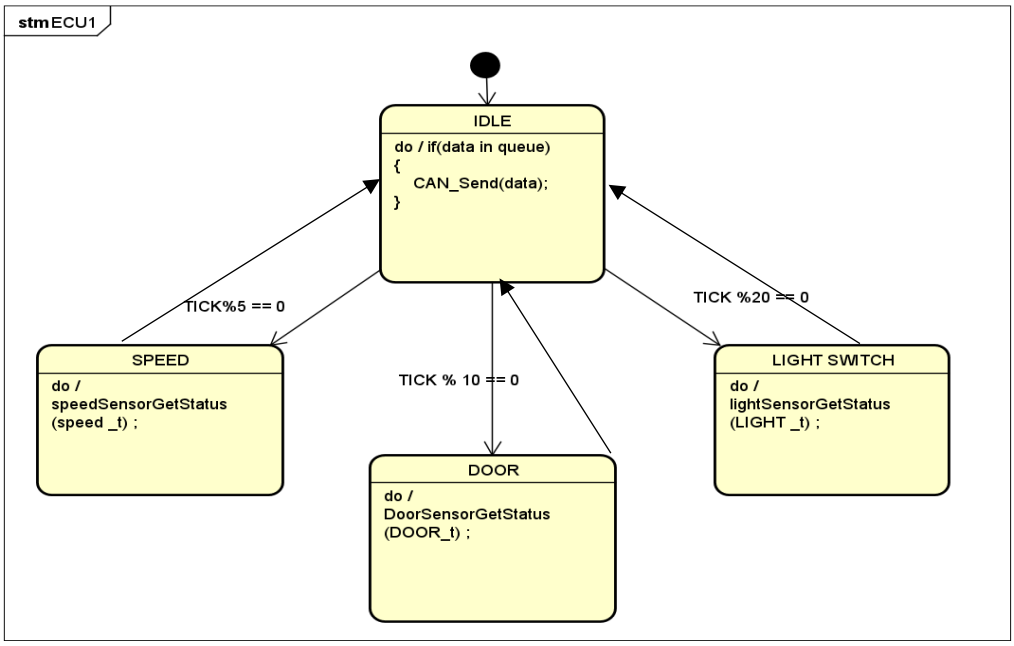
- APP

DataLogger.h

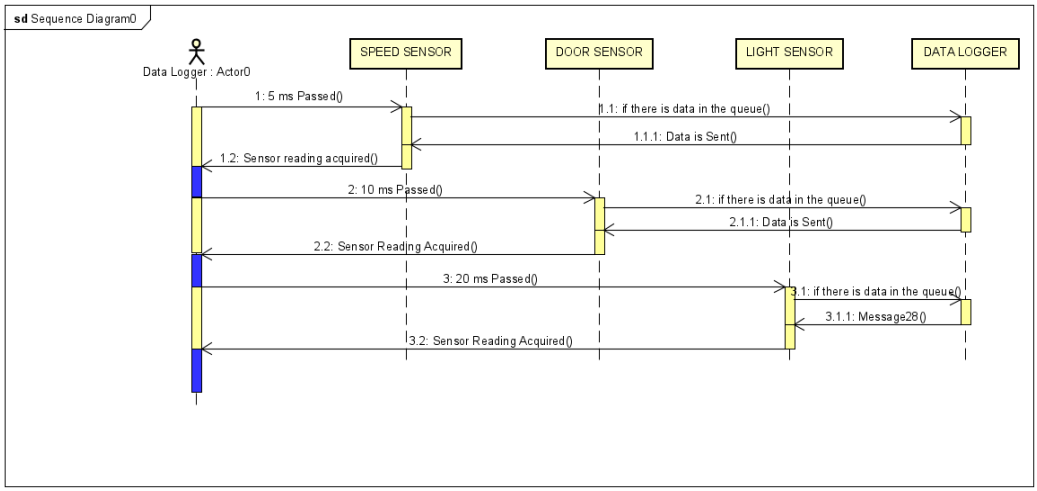
```
#include " DoorSensor.h"
#include " LightSensor.h"
#include " SpeedSensor.h"


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


/* Function initialize the data logger It takes struct of
type dataLogger _t and return nothing */
void DataLogger_init(dataLogger _t);

/* Function to get the data logger readings It takes
struct of type dataLogger _t and return nothing */
void GetSensorsReadings(dataLogger _t) ;

/* Function to send the data logger readings It takes
struct of type dataLogger _t and return nothing */
void Data_Send(dataLogger _t) ;
```

➢ Dynamic design analysis:

- ECU 1 :
  - state machine diagram:



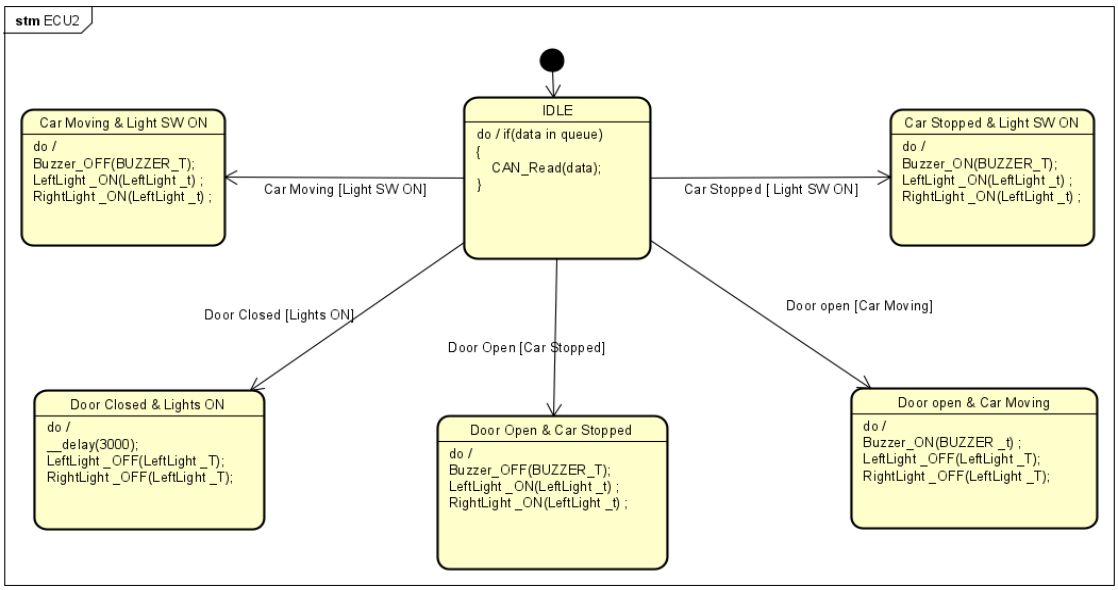
- sequence diagram:



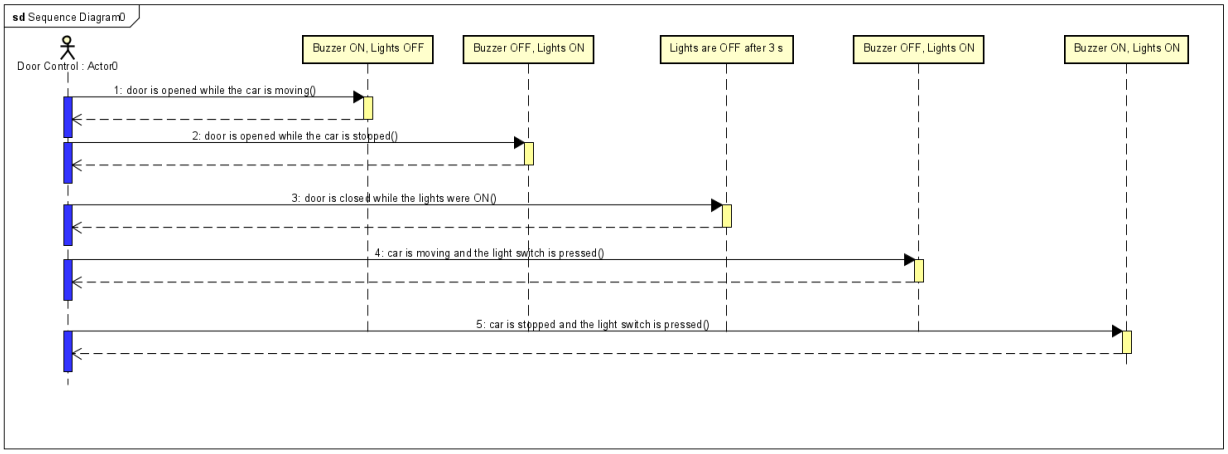
- CPU load : Assuming (Task 1 period = 2ms , Task2 period = 2ms , Task3 period = 2ms)

  U = (2 + 2 +2) / (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) / (hyper-period = 20) = 10/20 =50%