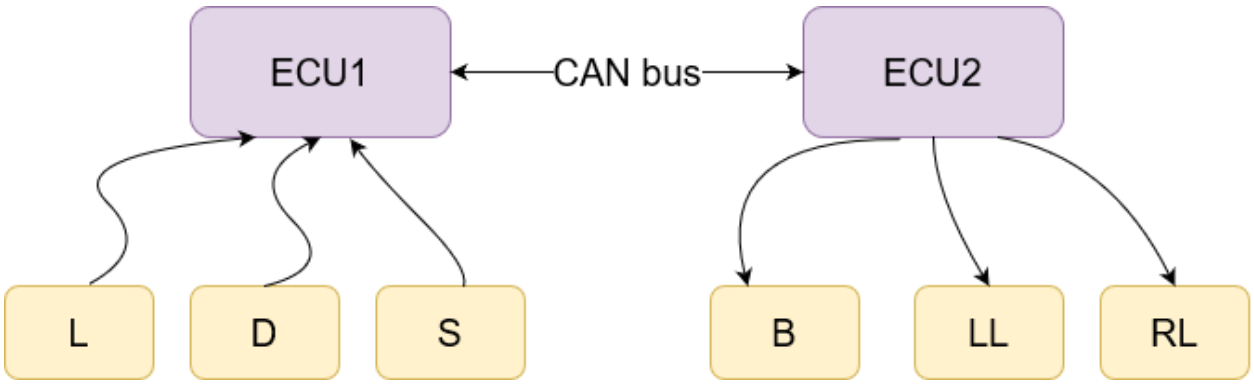
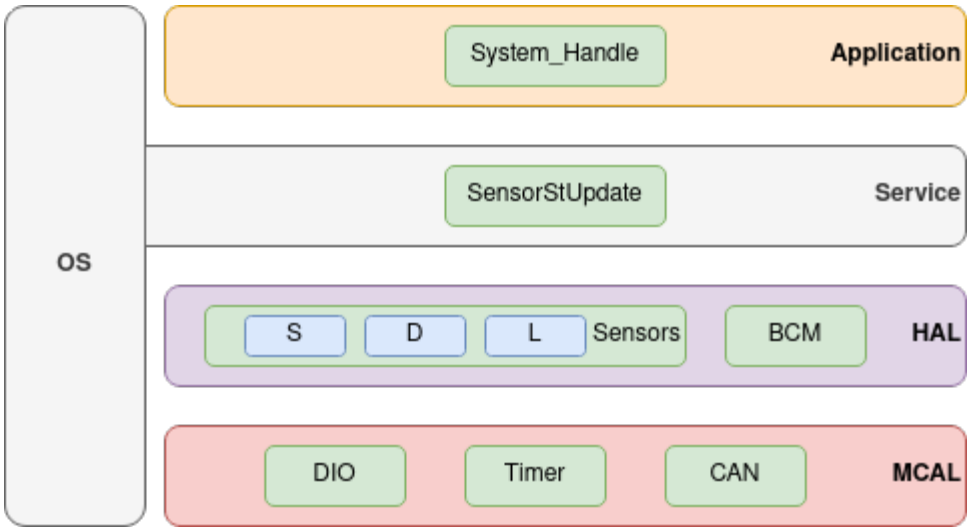
Automotive door control system design

System schematic



Static Design

ECU 1



ECU1 Components :

- DIO.
- Timer.
- CAN .

Components APIs :

- DIO.h :

```
#include "Std_Types.h"
enum DataDir { INPUT, OUTPUT};
```

```
    enum PORT { portA, portB,portC,portD,portE,portF};
    void DIO_Init( enum DataDir c , enum PORT port , uint32 PIN);

    uint32 DigitalRead(enum PORT port, uint32 PIN);
```

- DIO.c :

```
    void DIO_Init( enum DataDir c , enum PORT port , uint32 PIN){

  /*  Initialize the wanted PIN in the desired PORT as Input or Output
based on the usr requirement and the hardware used   */

/* example of that code can be taken from the ARM project -GPIO driver-
if it is needed   */


}

    uint32 DigitalRead(enum PORT port, uint32 PIN){

/* Read from the wanted PIN that assigned as an input from the desired
Port */

/* example of that code can be taken from the ARM project -GPIO driver-
if it is needed   */


}
```

- Timer.h :

```
      #include "Std_Types.h"
      void (*USER_CallBack_Fn)(void);

      void Timer_Init(void);
      void Timer_SetPeriod(uint32 Period);
      void Timer_Handler(void);
      void Timer_usrCallbackFn(void(*UserIsr)(void));

    Timer.c :
      void Timer_Init(void){
```

```
/* Initialize the timer */
/* example of that code can be taken from the ARM project -Timer driver-
if it is needed   */

/*In ARM project we used the SYSTICK timer*/




}
      void Timer_SetPeriod(uint32 Period){
/* Set Period for the timer */
/* example of that code can be taken from the ARM project -Timer driver-
if it is needed   */

/*In ARM project we used the SYSTICK timer*/

/* In our case the timer would be set to 5ms as the most frequent sensor
needs to be checked every 5 ms */

}

void Timer_usrCallbackFn(void(*UserIsr)(void)){
/*Links the USER ISR function with the timer ordinary handler , In our
case the user ISR will be the method that takes the periodic reading and
then send it to ECU2 -that method will be provided in the System_Handle
driver -*/

/* example of that code can be taken from the ARM project -Interrupt
driver- if it is needed   */

/*In ARM project we used the SYSTICK timer*/

             USER_CallBack_Fn= UserIsr;

}

void Timer_Handler(void)
{
/* System Timer ISR Handler */
/* example of that code can be taken from the ARM project -Interrupt
driver- if it is needed   */

/*In ARM project we used the SYSTICK timer*/
```

```
    (*USER_CallBack_Fn)();
}
```

- CAN.h :

```
Typedef CAN_Config ;
Typedef CAN_Message;

void CAN_Init(uint32_t baud_rate, CAN_Config config);
void CAN_Transmit(CAN_Message message);

CAN_Message CAN_Receive(void);
```

- CAN.c :

```
void CAN_Init(uint32_t baud_rate, CAN_Config config)
{
    /* Initialize CAN module with given baud rate and configuration
*/
     */ Configure CAN module based on given baud rate and
configuration */
}
void CAN_Transmit(CAN_Message message)
{
    /* Transmit a CAN message*/
    /* Copy message data to CAN transmit buffer
     Trigger transmission of message*/
}

CAN_Message CAN_Receive(void)
{
    /* Receive a CAN message */

    /* Check if a message has been received*/
    /* If a message has been received, read message data from CAN
receive buffer */
    /* Return received message */
}
```

- BCM.h :

```c
#include "CAN.h"
#include "Timer.h"
#include "Sensors.h"

void BCM_Init(uint32_t Period, enum BUS bus);
void BCM_Callback_ISR();
void BCM_Send(enum ID ECU_ID, uint32_t Data);
```

- BCM.c :

```c
#include "BCM.h"

void BCM_Init(uint32_t Period, enum BUS bus){

    CAN_Init(baud_rate,config);
}
void BCM_Send(enum ID ECU_ID, uint32_t Data){

CAN_Message message = Data;

void CAN_Transmit(message);
/* Sends the desired data. */
}
```

- Sensors.h :

```c
#include "Std_Types.h"
#include "DIO.h"


void Sensors_Init(void);
uint32 D-Sensor_Read();
uint32 L-Sensor_Read();
uint32 S-Sensor_Read();
```

Sensors.c :

```c
#include "Sensors.h"

void Sensors_Init(void){
/* Initialize the D sensor DIO as input. */
```

```
       DIO_Init( INPUT , portF , 1);
    /* Initialize the L sensor DIO as input. */
       DIO_Init( INPUT , portF , 2);
    /* Initialize the S sensor DIO as input. */
       DIO_Init( INPUT , portF , 3);


}
      uint32 D-Sensor_Read(){
    /* Returns the input value of the DIO pin assigned to the D sensor.
*/
    return  DigitalRead(portF, 1);

}

      uint32 L-Sensor_Read(){
  /* Returns the input value of the DIO pin assigned to the L sensor. */

 return  DigitalRead(portF, 1);

}
      uint32 S-Sensor_Read(){
 /* Returns the input value of the DIO pin assigned to the S sensor. */

  return  DigitalRead(portF, 1);

}
```

- SensorStUpdate.h :

```
      #include "Std_Types.h"
      #include "SensorStUpdate.h"
      #include "BCMtUpdate.h"


      void SensorStUpdate._Init(void);
      void Sensor_Callback_ISR()
```

System_Handle.c :

```
      #include "SensorStUpdate.h"
```

```
    void SensorStUpdate(void){


 /* Initiate the Sensors DIOs. */
            Sensors_Init();
           Timer_Init();
          /* Initiate the timer that sends the data periodically. */
          Timer_SetPeriod(5);
          /* Sets the timer period which is 5 milli seconds. */
          Timer_usrCallbackFn(*(Sensor_Callback_ISR()));
          /* Links the timer interrupt handler with the BCM that sends
the data periodically the S reading every 5 ms , L reading every 10 ms
and the D reading every 15 ms . */

}




void Sensor_Callback_ISR(){

* the handler sends the data periodically the S reading every 5 ms , L
reading every 10 ms  and the D reading every 15 ms . */
/* i represents the value that indicate which data will be sent at a time
and every 1 of i represents 1 ms period */

 if(i==1 || i==3 ){

/* Send Door sensor Data to ECU2 through CAN bus. */
BCM_Send(ECU2_ID, D-Sensor_Read());

/*  Update the i value. */
i++;

}esle if(i==2){

/* Send Door sensor Data to ECU2  through CAN bus. */
/* Send Lamp sensor Data to ECU2 through CAN bus. */
BCM_Send(ECU2_ID, D-Sensor_Read());
BCM_Send(ECU2_ID, L-Sensor_Read());

/*  Update the i value. */
```

```
i++;

}else if(i==4){


/* Send Door sensor Data to ECU2 through CAN bus. */
/* Send Lamp sensor Data to ECU2 through CAN bus. */
/* Send Door sensor Data to ECU2 through CAN bus. */

BCM_Send(ECU2_ID, D-Sensor_Read());
BCM_Send(ECU2_ID, L-Sensor_Read());
BCM_Send(ECU2_ID, S-Sensor_Read());
 /* Resets the i value. */
i=1;


}


}
```

- System_Handle.h :

```
#include "Std_Types.h"
#include "SensorStUpdate.h"


 void System_Init(void);
```

System_Handle.c :

```
  #include "System_Handle.h"

void System_Init(void){
 /* Initialize the Sensor Updating monitor which is taking the
periodic reading. */
   SensUpdate_Init();
```

```
}
```

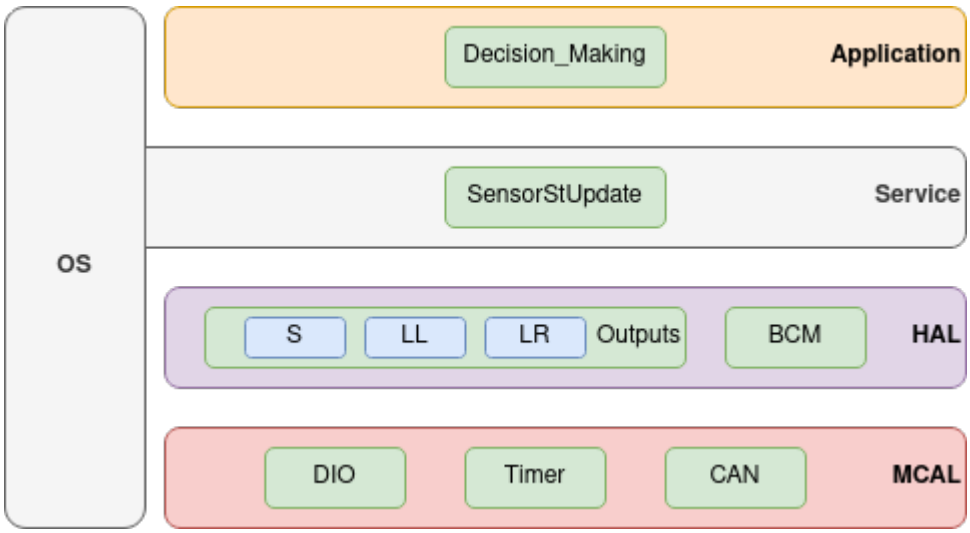
- App
  -System_Handle.h
  -System_Handle.c

- Service
  -SensorStUpdate.h
  -SensorStUpdate.c

- HAL
  -BCM.h
  -BCM.c
  -Sensors.h
  -Sensors.c

- MCAL
  -DIO.h
  -DIO.c
  -Timer.h
  -Timer.c
  -CAN.h
  -CAN.c

ECU 2



ECU2 Components :

- DIO.
- Timer.
- CAN .

Components APIs :

- DIO.h :

```c
#include "Std_Types.h"

enum DataDir { INPUT, OUTPUT};

enum PORT { portA, portB,portC,portD,portE,portF};
void DIO_Init( enum DataDir c , enum PORT port , uint32 PIN);

uint32 DigitalWrite(enum PORT port, uint32 PIN , uint32  value);
```

- DIO.c :

```c
void DIO_Init( enum DataDir c , enum PORT port , uint32 PIN){

  /*  Initialize the wanted PIN in the desired PORT as Input or Output
based on the user requirement and the hardware used   */

/* example of that code can be taken from the ARM project -GPIO driver-
if it is needed   */


}

uint32 DigitalWrite(enum PORT port, uint32 PIN, uint32  value){

/* Write to the wanted PIN that assigned as an output from the desired
Port */

/* example of that code can be taken from the ARM project -GPIO driver-
if it is needed   */


}
```

- Timer.h :

```c
#include "Std_Types.h"
void (*USER_CallBack_Fn)(void);

void Timer_Init(void);
void Timer_SetPeriod(uint32 Period);
```

```
      void Timer_Handler(void);
      void Timer_usrCallbackFn(void(*UserIsr)(void));
```

Timer.c :

```
      void Timer_Init(void){

/* Initialize the timer */
/* example of that code can be taken from the ARM project -Timer driver-
if it is needed   */

/*In ARM project we used the SYSTICK timer*/




}
      void Timer_SetPeriod(uint32 Period){
/* Set Period for the timer */
/* example of that code can be taken from the ARM project -Timer driver-
if it is needed   */

/*In ARM project we used the SYSTICK timer*/

/* In our case the timer would be set to 5ms as the most frequent sensor
needs to be checked every 5 ms */

}

void Timer_usrCallbackFn(void(*UserIsr)(void)){
/*Links the USER ISR function with the timer ordinary handler , In our
case the user ISR will be the method that takes the periodic reading and
then send it to ECU2 -that method will be provided in the System_Handle
driver -*/

/* example of that code can be taken from the ARM project -Interrupt
driver- if it is needed   */

/*In ARM project we used the SYSTICK timer*/

             USER_CallBack_Fn= UserIsr;

}

void Timer_Handler(void)
{
```

```
/* System Timer ISR Handler */
/* example of that code can be taken from the ARM project -Interrupt
driver- if it is needed   */

/*In ARM project we used the SYSTICK timer*/


      (*USER_CallBack_Fn)();
}
```

- CAN.h :

```
Typedef CAN_Config ;
Typedef CAN_Message;

void CAN_Init(uint32_t baud_rate, CAN_Config config);
void CAN_Transmit(CAN_Message message);

CAN_Message CAN_Receive(void);
```

- CAN.c :

```
void CAN_Init(uint32_t baud_rate, CAN_Config config)
{
    /* Initialize CAN module with given baud rate and configuration
*/
     */ Configure CAN module based on given baud rate and
configuration */
}
void CAN_Transmit(CAN_Message message)
{
    /* Transmit a CAN message*/
    /* Copy message data to CAN transmit buffer
     Trigger transmission of message*/
}

CAN_Message CAN_Receive(void)
{
    /* Receive a CAN message */
```

```
        /* Check if a message has been received*/
        /* If a message has been received, read message data from CAN
    receive buffer */
        /* Return received message */
    }
```

- BCM.h :

```
    #include "CAN.h"
    #include "Timer.h"
    #include "Outputs.h"


        /* variables to hold the sensors statues . */

    uint32_t D-value;
    uint32_t S-value;
    uint32_t L-value;

    void BCM_Init(uint32_t Period, enum BUS bus);
    void BCM_Callback_ISR();
    uint32_t BCM_Receive(enum ID ECU_ID);
```

- BCM.c :

```
    #include "BCM.h"

    void BCM_Init(uint32_t Period, enum BUS bus){


        CAN_Init(baud_rate,config);


}
    uint32_t BCM_Receive(enum ID ECU_ID){

    uint32_t message =CAN_Receive();
    return  message;



    }
```

- Outputs.h :

```c
#include "Std_Types.h"
#include "DIO.h"


void Outputs_Init(void);
void B-Sensor_Write(uint32  value);
void LL-Sensor_Write(uint32  value);
void LR-Sensor_Write(uint32  value);
```

Outouts.c :

```c
  void Sensors_Init(void){
 /* Initialize the D sensor DIO as output. */
   DIO_Init( OUTPUT , portF , 1);
/* Initialize the L sensor DIO as output. */
   DIO_Init( OUTPUT , portF , 2);
/* Initialize the S sensor DIO as output . */
   DIO_Init( OUTPUT , portF , 3);


}
 void B-Sensor_Write(uint32  value){
/* Sets the output value of the DIO pin assigned to the Buzzer. */
      DigitalWrite(portF, 1 , value);

}

   void LL-Sensor_Write(uint32  value){
  /* Sets the output value of the DIO pin assigned to the LL . */

   DigitalWrite(portF, 1 , value );

}
  void LR-Sensor_Write(uint32  value){
 /* Sets the output value of the DIO pin assigned to the LR . */

    DigitalWrite(portF, 1, value);

}
```

- Decision_Making.h :

```c
#include "Std_Types.h"
#include "Outputs.h"
#include "BCM.h"
void Outputs_Update(void);
```

Decision_Making.c :

```c
void Outputs_Update(void){

if( D-value==1 && S-value>0 ){

/*If the door is opened while the car is moving → Buzzer ON, Lights OFF */
B-Sensor_Write( 1 );
LL-Sensor_Write( 0 );
LR-Sensor_Write( 0 );


}esle if( D-value==1 && S-value==0 ){

/* If the door is opened while the car is stopped → Buzzer OFF, Lights ON */
B-Sensor_Write( 0 );
LL-Sensor_Write( 1 );
LR-Sensor_Write( 1 );


}esle if( L-value==1 && S-value>0 ){

/*If the car is moving and the light switch is pressed → Buzzer OFF, Lights ON*/
B-Sensor_Write( 0 );
LL-Sensor_Write( 1 );
LR-Sensor_Write( 1 );


}esle if( L-value==1 && S-value==0 ){

/*If the car is stopped and the light switch is pressed → Buzzer ON, Lights ON */
B-Sensor_Write( 1 );
LL-Sensor_Write( 1 );
```

```
LR-Sensor_Write( 1 );

}

}
```

- SensorStUpdate.h :

```
#include "Std_Types.h"
#include "SensorStUpdate.h"
#include "BCMtUpdate.h"

void SensorStUpdate._Init(void);
void Sensor_Callback_ISR()
```

System_Handle.c :

```
#include "SensorStUpdate.h"

void SensorStUpdate(void){

/* Initiate the Sensors DIOs. */
           Sensors_Init();
          Timer_Init();
         /* Initiate the timer that sends the data periodically. */
         Timer_SetPeriod(5);
         /* Sets the timer period which is 5 milli seconds. */
         Timer_usrCallbackFn(*(Sensor_Callback_ISR()));
         /* Links the timer interrupt handler with the BCM that sends
the data periodically the S reading every 5 ms , L reading every 10 ms
and the D reading every 15 ms . */

                                    }
```

```
void Sensor_Callback_ISR(){

* the handler sends the data periodically the S reading every 5 ms , L
reading every 10 ms  and the D reading every 15 ms . */
/* i represents the value that indicate which data will be sent at a time
and every 1 of i represents 1 ms period */

 if(i==1 || i==3 ){

/* Read Door sensor Data to ECU2 through CAN bus. */
 D-value = BCM_Receive(ECU_ID);

/*  Update the i value. */
i++;

}esle if(i==2){

/* Read Door sensor Data to ECU2  through CAN bus. */
/* Read Lamp sensor Data to ECU2 through CAN bus. */
 D-value = BCM_Receive(ECU_ID);
 L-value = BCM_Receive(ECU_ID);

/*  Update the i value. */

i++;

}else if(i==4){


/* Read Door sensor Data to ECU2 through CAN bus. */
/* Read Lamp sensor Data to ECU2 through CAN bus. */
/* Read Door sensor Data to ECU2 through CAN bus. */

 D-value = BCM_Receive(ECU_ID);
 L-value = BCM_Receive(ECU_ID);
 S-value = BCM_Receive(ECU_ID);
 /* Resets the i value. */
i=1;


}
```

```
}
```

- App
  -Decision_Making.h
  -Decision_Making .c

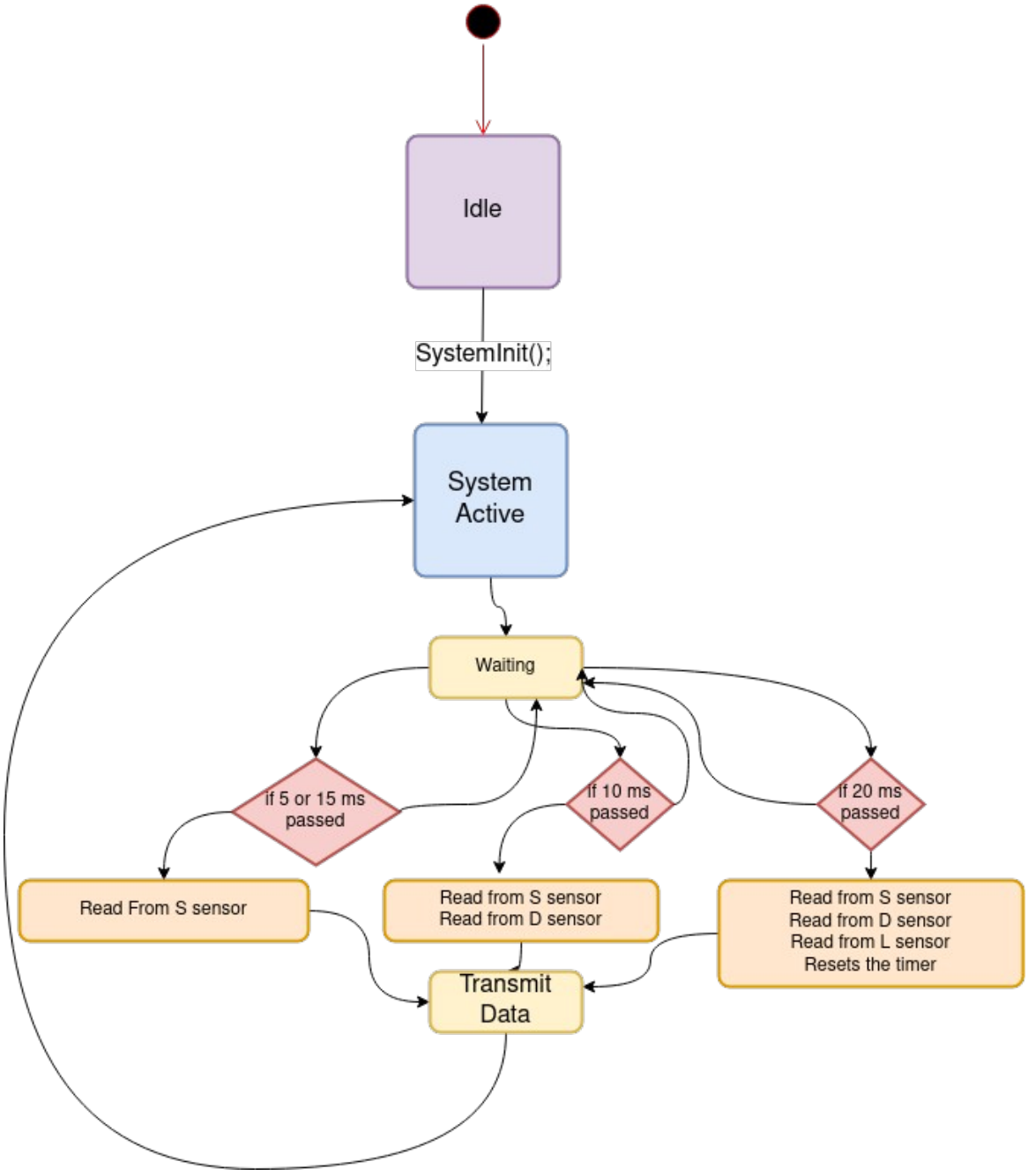
- Service
  -SensorStUpdate.h
  -SensorStUpdate.c

- HAL
  -BCM.h
  -BCM.c
  -Outputs.h
  -Outputs.c

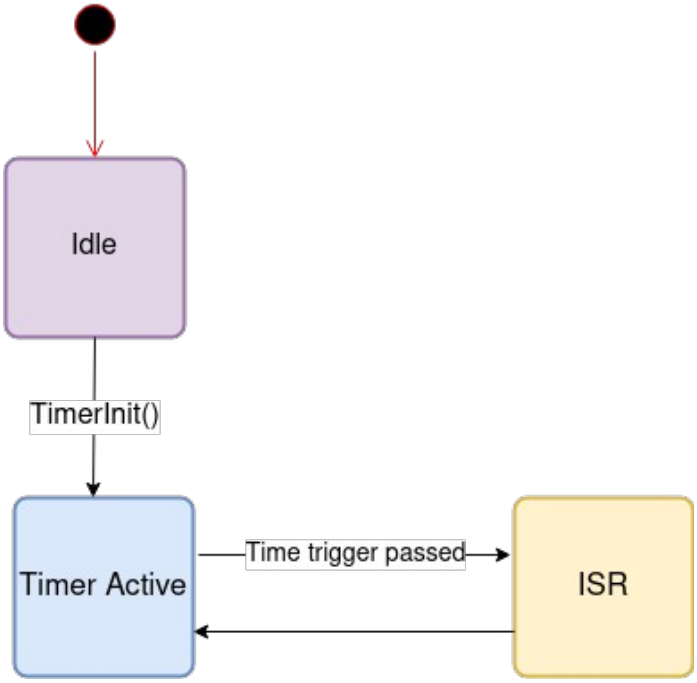
- MCAL
  -DIO.h
  -DIO.c
  -Timer.h
  -Timer.c
  -CAN.h
  -CAN.c

Dynamic Design

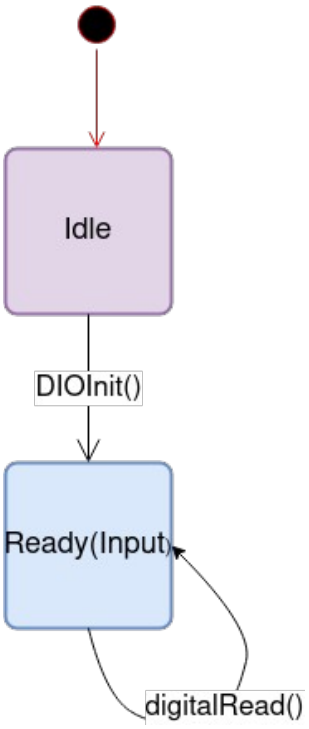
ECU 1



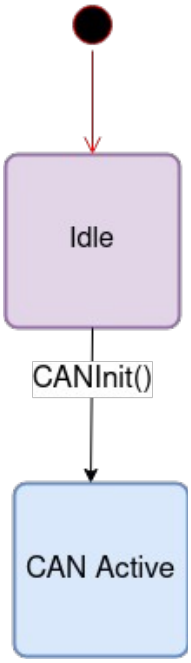
Timer

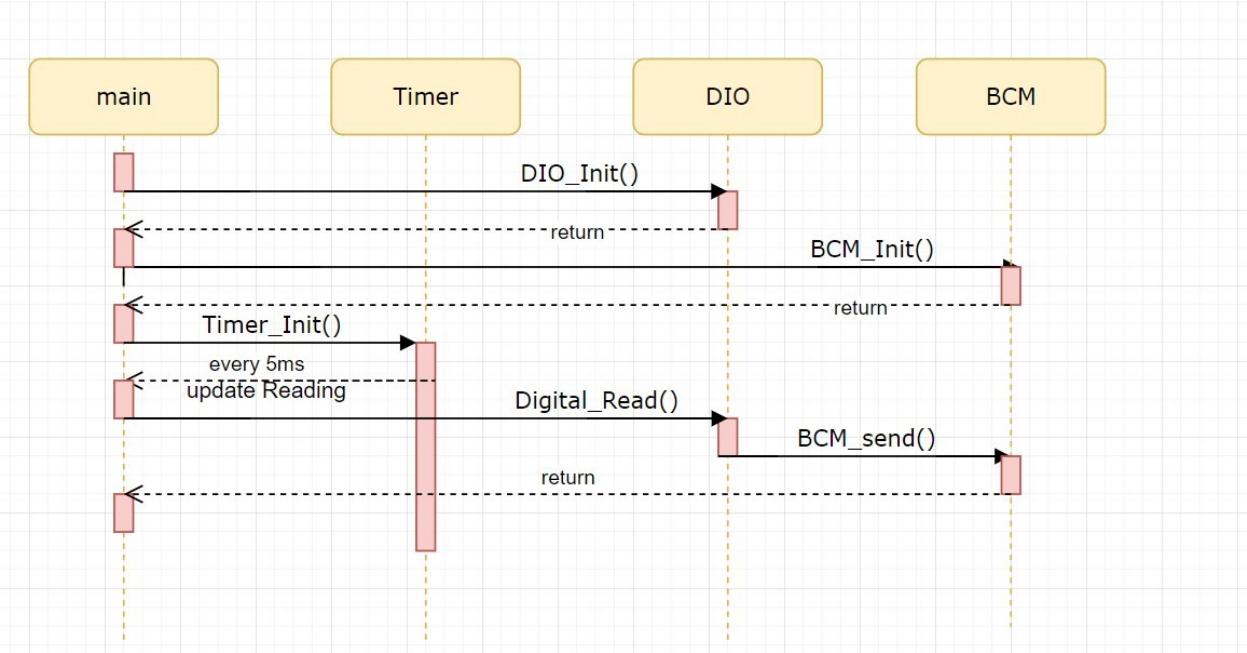

Idle
TimerInit()
Timer Active
Time trigger passed
ISR

DIO


Idle
DIOInit()
Ready(Input)
digitalRead()

CAN



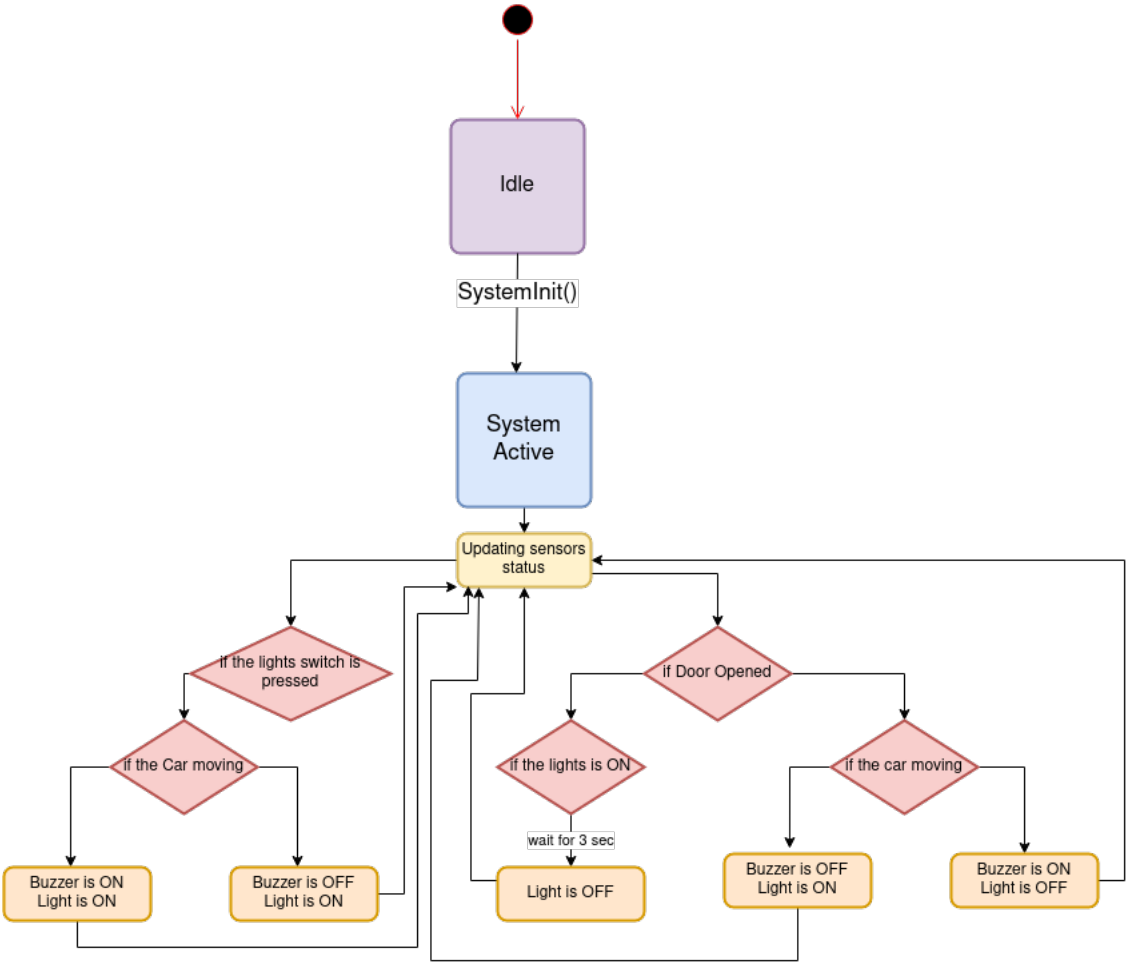


CPU Load

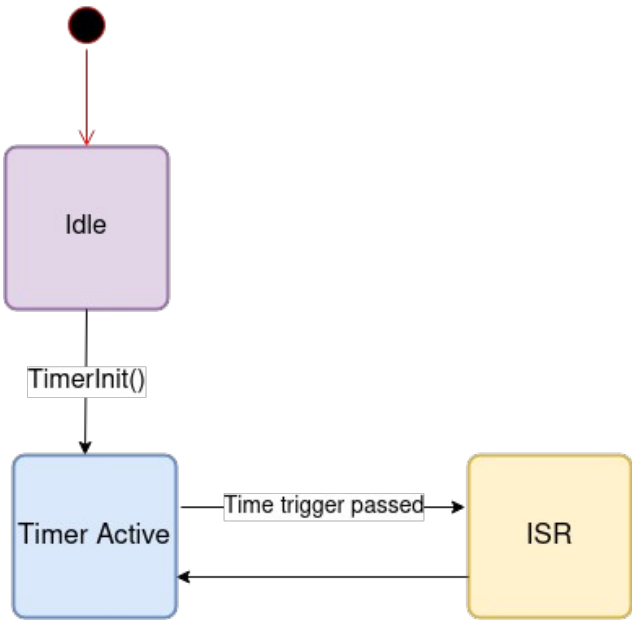
ECU1 Load = (Reading_S_task*4+Reading_D_task*2+Reading_L_task+ CANSend_task*7)/ ECU1_Hyperperiod .

Since ECU1_Hyperperiod = 20ms .

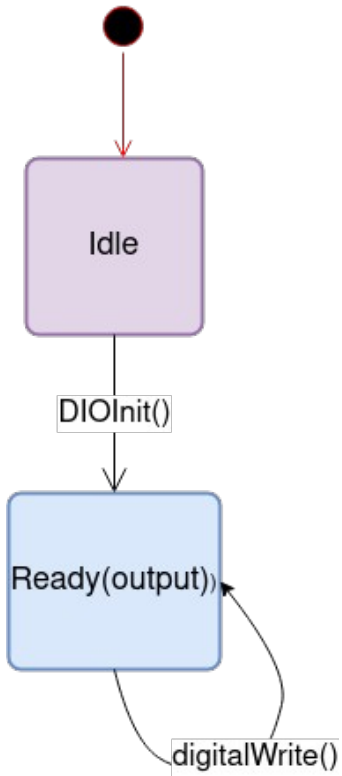
ECU 2


Idle
SystemInit()
System Active
Updating sensors status
if the lights switch is pressed
if the Car moving
Buzzer is ON
Light is ON
Buzzer is OFF
Light is ON
if Door Opened
if the lights is ON
wait for 3 sec
Light is OFF
if the car moving
Buzzer is OFF
Light is ON
Buzzer is ON
Light is OFF

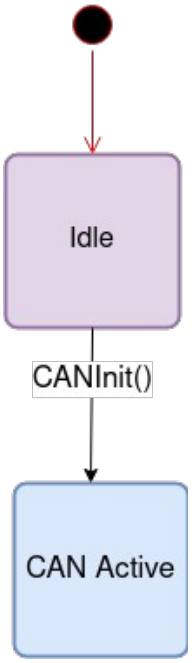
Timer

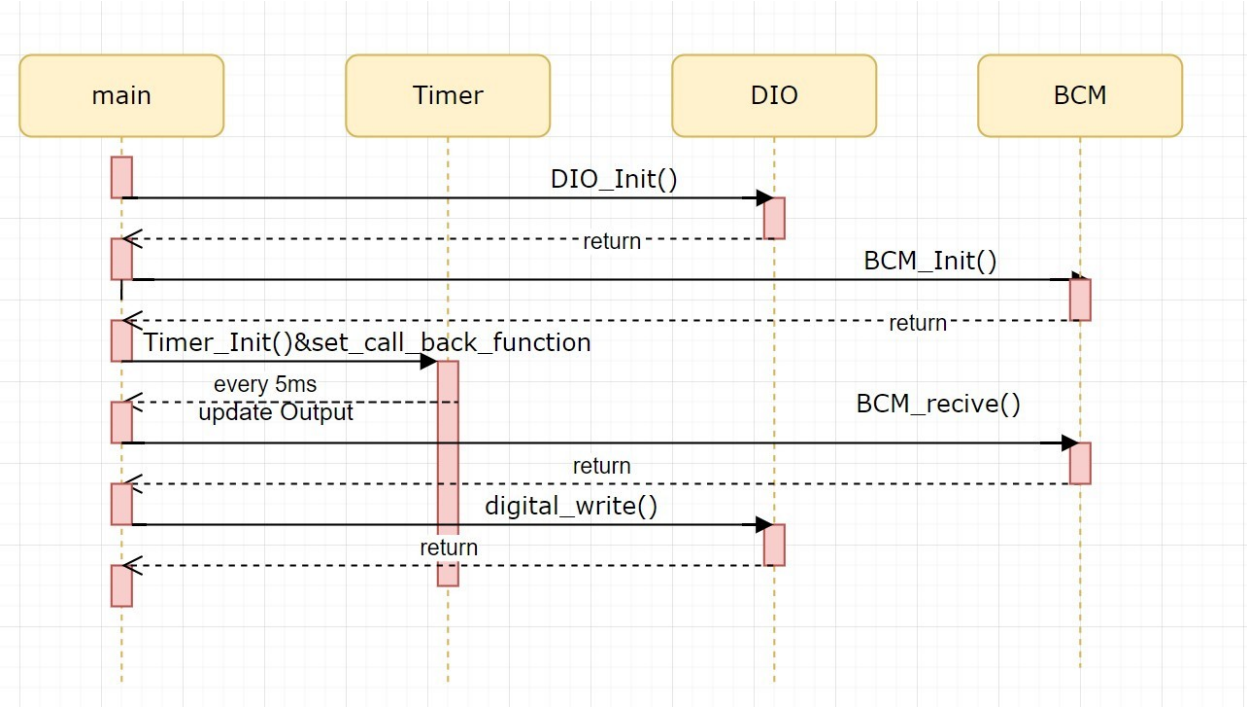

Idle
TimerInit()
Timer Active
Time trigger passed
ISR

DIO


Idle
DIOInit()
Ready(output)
digitalWrite()

CAN





CPU Load

ECU2 Load = (Update_Outpur_task+ CANReceive_task*3)/ ECU2_Hyperperiod .

Since ECU2_Hyperperiod = 5ms .