Automotive door control system design

By Youssef Mohamed Youssef

System Diagram

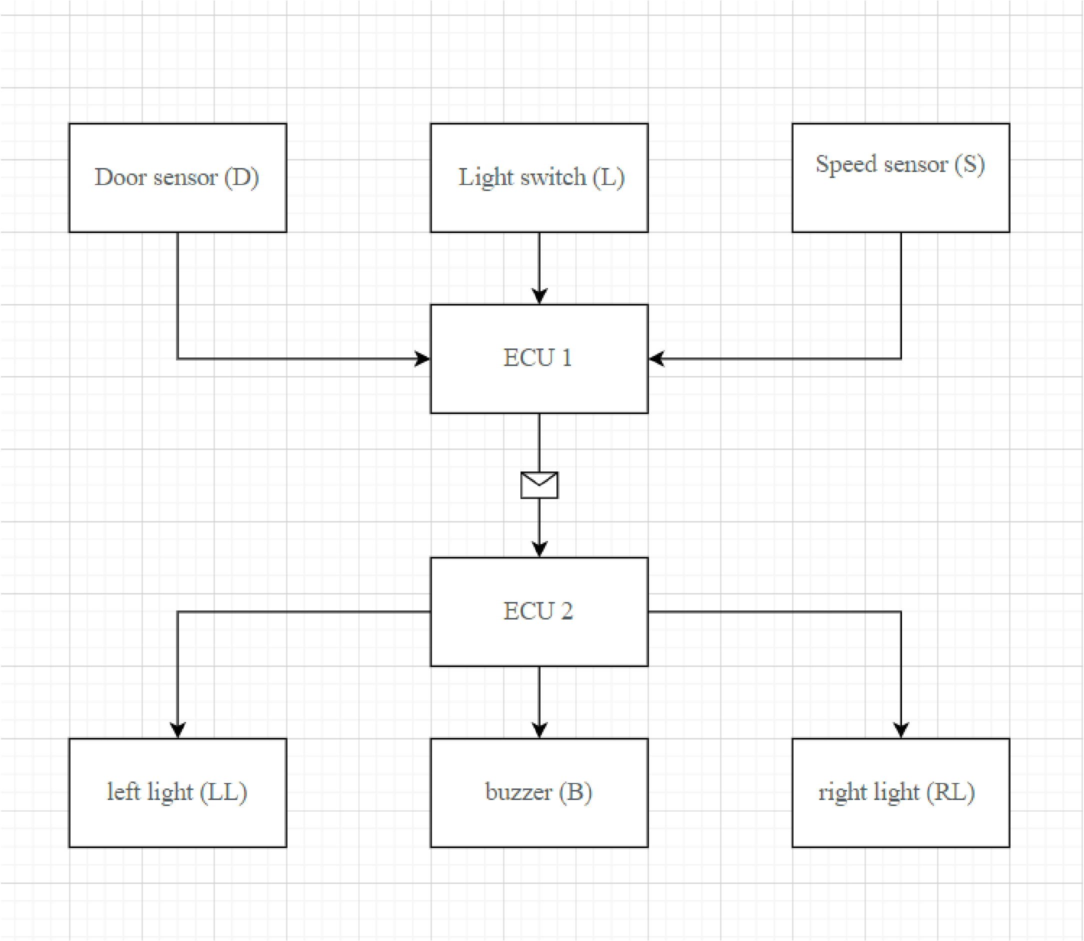


Figure 1

As shown in (figure 1) ECU 1 is connected to the sensors and ECU 2 is connected to the actuators.

ECU 1 send status periodically ECU 2 via CAN protocol.

Static design

The LIB, MCAL, SERVICES layers are the same in both ECUs. They will be disguised in detail.

ECU 1:

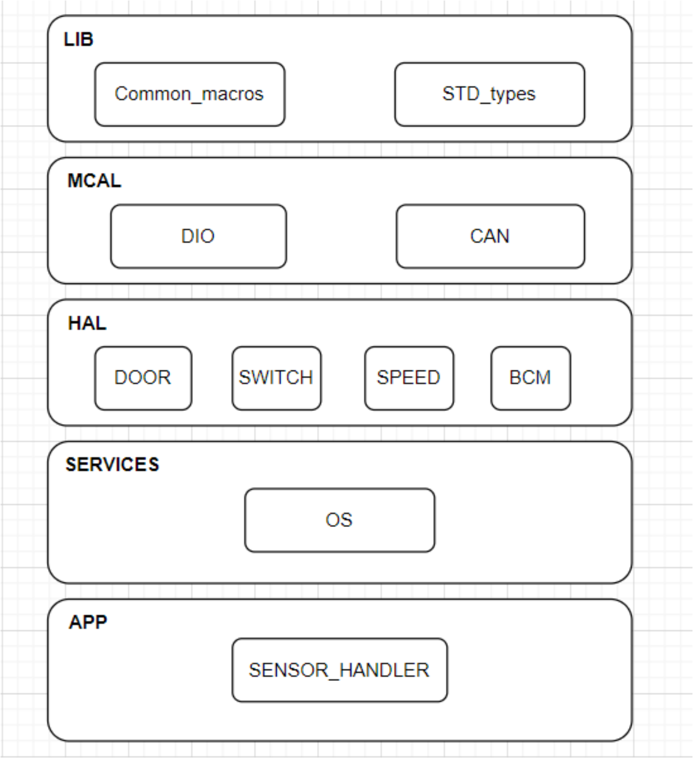


Figure 2

LIB Layer:

File	Description
Common_macros.h	**All commonly used macros**
STD_types.h	**Standard data types**

MCAL Layer:

Folder	Description
DIO	**DIO driver “DIO_registers.h”, “DIO_interface”, “DIO_program.c”**
CAN	**CAN driver “CAN_config.h”, “CAN_registers.h”, “CAN_interface”, “CAN_program.c”**

HAL Layer:

Folder	Description
DOOR	**Door sensor driver “DOOR_config.h”, “DOOR_interface”, “DOOR_program.c”**
SWITCH	**Light switch driver “SWITCH_config.h”, “SWITCH_interface”, “SWITCH_program.c”**
SPEED	**SPEED driver “SPEED_config.h”, “SPEED_interface”, “SPEED_program.c”**
BCM	**BCM driver “BCM_config.h”, “BCM_interface”, “BCM_program.c”**

SERVICES Layer:

Folder	Description
OS	**This folder contains all the OS files When using OS include “OS_interface.h” Also, configuration editing is done in “OS_config.h” Note: don’t change anything in other files**

APP Layer:

FILE	Description
SENSOR_HANDLER.c	**This file contains the main function of ECU 1**

ECU 2:

The difference here is in HAL and APP LAYERS.

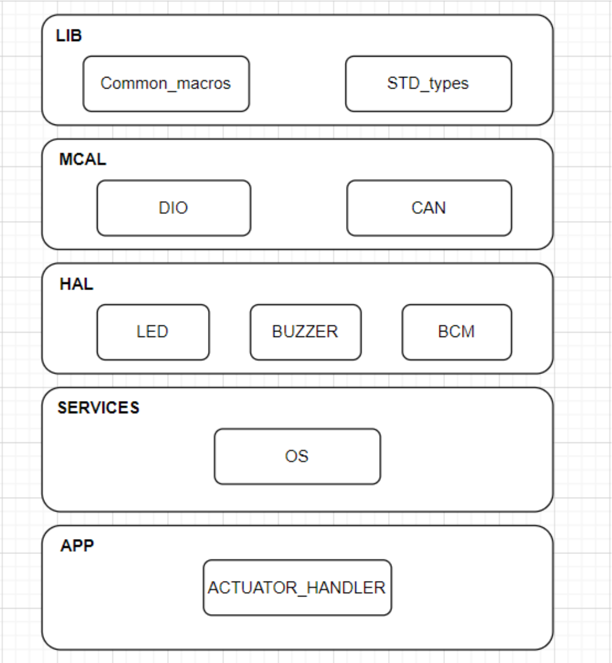


Figure 3

HAL Layer:

Folder	Description
LED	**LED driver used for ‘LL’ and ‘RL’ “LED_config.h”, “LED_interface”, “LED_program.c”**
BUZZER	**Buzzer driver “BUZZER_config.h”, “BUZZER_interface”, “BUZZER_program.c”**

APP Layer:

FILE	Description
ACTUATOR_HANDLER.c	**This file contains the main function of ECU 2**

APIs definition:

1. LIB layer:

1.1. Common_macros.h

Name	Type	value	description
E_OK	Macro	0	Used to define pin low state
E_NOK	Macro	1	Used to define pin high state
NULL	Macro	(void*)0	NULL pointer

1.2. STD_types.h:

Name	Type	Expansion	description
uint_8	Data type	Typedef unsigned char	standard type
sint_8	Data type	Typedef signed char	standard type
uint_16	Data type	Typedef unsigned short int	standard type
sint_16	Data type	Typedef signed short int	standard type
uint_32	Data type	Typedef unsigned long int	standard type
sint_32	Data type	Typedef signed long int	standard type
float_32	Data type	Typedef float	standard type
float_64	Data type	Typedef double	standard type

2. MCAL layer:

2.1. DIO:

Name	Return type	Return range	Parameters	Range	Description
DIO_init	Void	n/a	Void	N/a	Used to initialize DIO
DIO_write	uint_8	E_OK / E_NOK	(uint_8) Port_no	PORT_A / PORT_B	Change pin value
			(uint_8) Pin_no	PIN_0 -> PIN_15	
			(uint_8) Pin_value	PIN_HIGH / PIN_LOW	
DIO_read	uint_8	DIO_HIGH / DIO_LOW	(uint_8) Port_no	PORT_A / PORT_B	Return pin value
			(uint_8) Pin_no	PIN_0 -> PIN_15	

2.2. CAN:

Name	Return type	Return range	Parameters	Range	Description
CAN_init	Void	n/a	Void	N/a	Used to initialize CAN
CANMessageSet	void	n/a	(uint_32) BaseID	min to max base id	Send message
			(uint_32) ReceiverID	Min to max receiver id	
			(uint_32*)psMsgObject	Address of uint_32 variable	
CANMessageGet	void	n/a	(uint_32) BaseID	Min to max base id	Receive message And save it in specific address
			(uint_32) SenderID	Min to max receiver id	
			(uint_32*)psMsgObject	Address of uint_32 variable	

3. HAL Layer:

3.1. DOOR:

Name	Return type	Value	Parameters	Range	Description
DOOR_init	Void	n/a	Void	N/a	Used to initialize door sensor
DOOR_reading	uint_8	DOOR_OPENED/ DOOR_CLOSED	void	n/a	Return the state of the door

3.2. SWITCH

Name	Return type	Return range	Parameters	Range	Description
SWITCH_init	Void	n/a	Void	N/a	Used to initialize light switch sensor
SWITCH_reading	uint_8	SWITCH_OPENED/ SWITCH_CLOSED	Void	N/a	Return the state of the switch

3.3. SPEED

Name	Return type	Return range	Parameters	Range	Description
SPEED_init	Void	n/a	Void	N/a	Used to initialize Speed sensor
SPEED_reading	uint_8	CAR_MOVING/ CAR_STOPPED	Void	N/a	Return the state of the car speed

3.4. LED

Name	Return type	Return range	Parameters	Range	Description
LED_init	Void	n/a	Void	N/a	Used to initialize LED
LED_ChangeState	void	n/a	(unit_8) LED_ID	LED_LEFT/LED_RIGHT	Change the state of the led
			(unit_8) State	LED_ON/ LED_OFF	

3.5. BUZZER

Name	Return type	Return range	Parameters	Range	Description
BUZZER_init	Void	n/a	Void	N/a	Used to initialize tone
BUZZER_ChangeState	Void	n/a	(unit_8) state	BUZZER_ON/ BUZZER_OFF	Change the state of the Buzzer

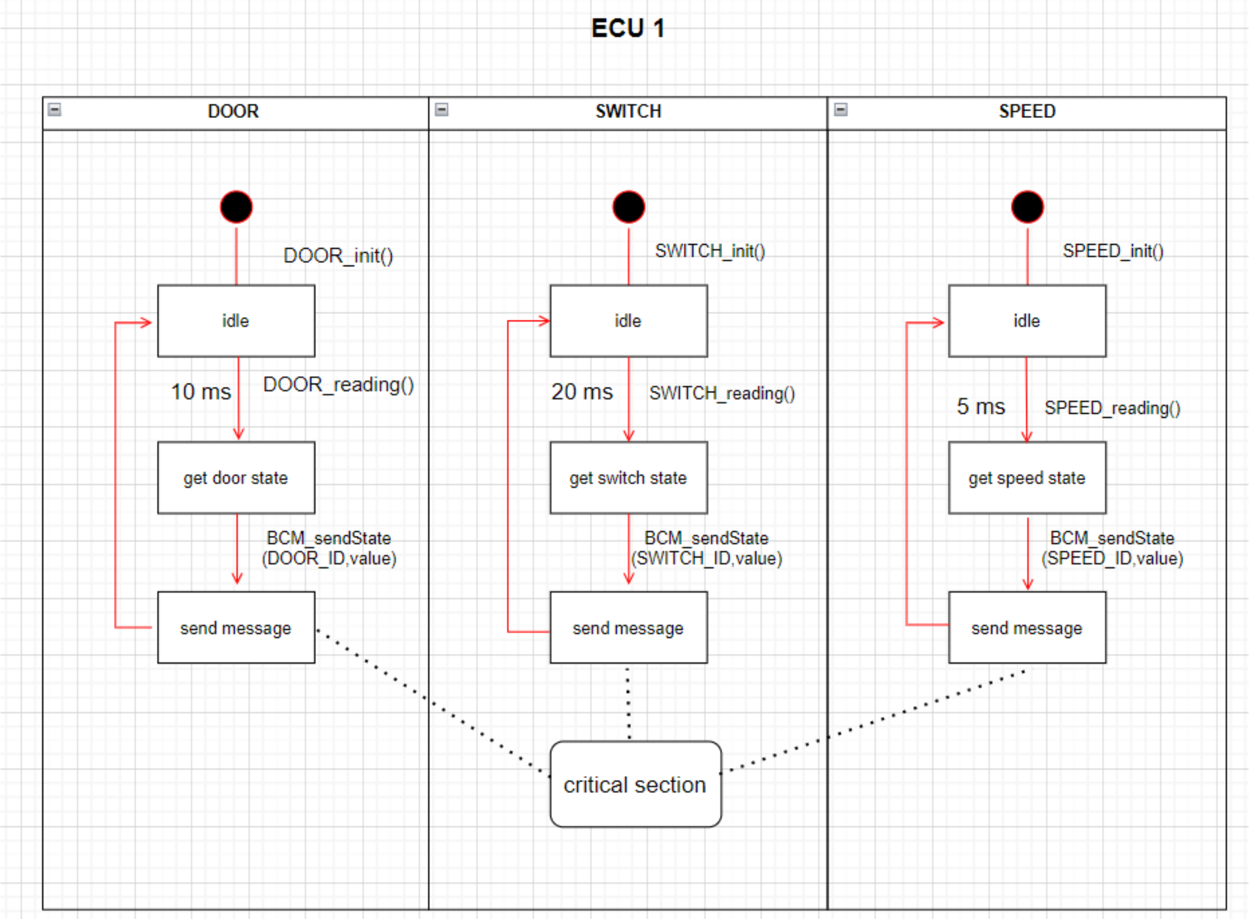
3.6. BCM

Name	Return type	Return range	Parameters	Range	Description
BCM_init	Void	n/a	Void	N/a	Used to initialize BCM
BCM_sendState	void	n/a	(uint_8) ID	DOOR_ID/SWITCH_ID/SPEED_ID	Used to send sensors' states from ecu 1 to ecu 2
			(uint_8) State	ID = DOOR_ID DOOR_OPENED/DOOR_CLOSED/ ID = SWITCH_ID SWITCH_ON/SWITCH_OFF/ ID =SPEED_ID CAR_MOVING/CAR_STOPPED	
BCM_receiveState	uint_8	ID = DOOR_ID DOOR_OPENED/DOOR_CLOSED/ ID = SWITCH_ID SWITCH_ON/SWITCH_OFF/ ID =SPEED_ID CAR_MOVING/CAR_STOPPED	(uint_8) ID	DOOR_ID/SWITCH_ID/SPEED_ID	Used to receive data from sensors' states from ecu 1 to ecu 2

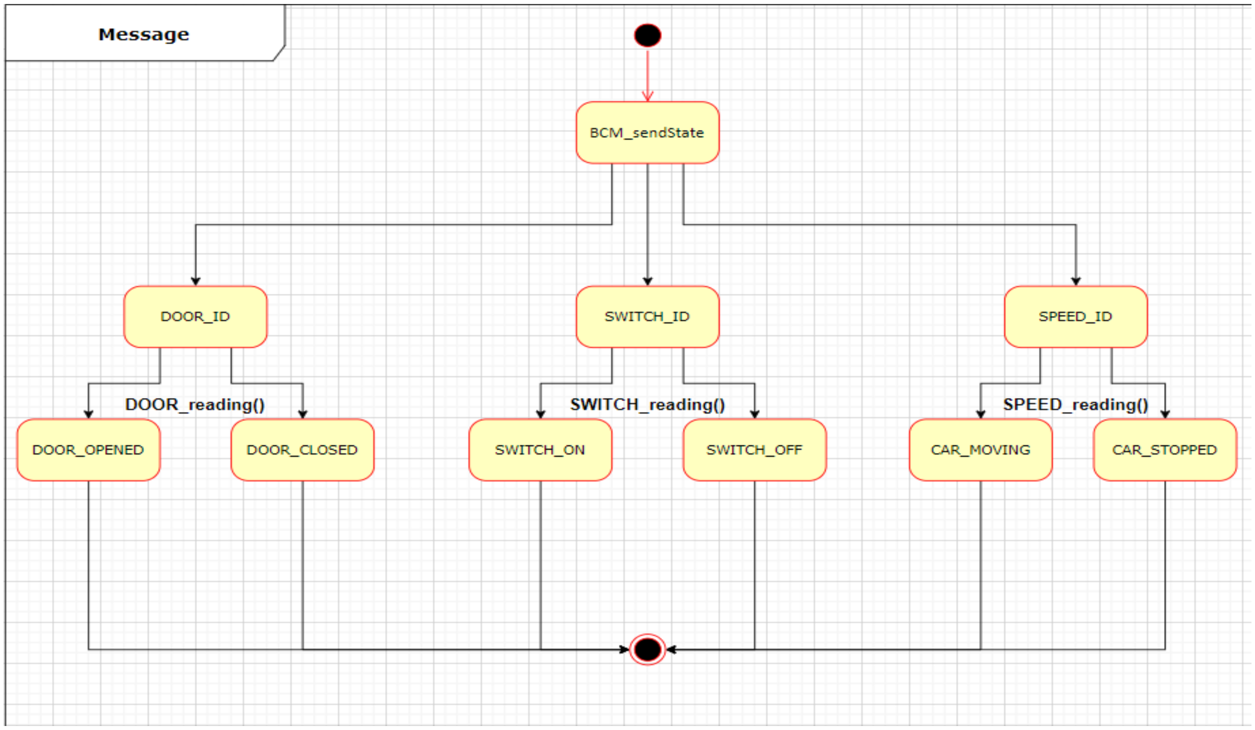
Dynamic design

Components are initialized before starting the OS.

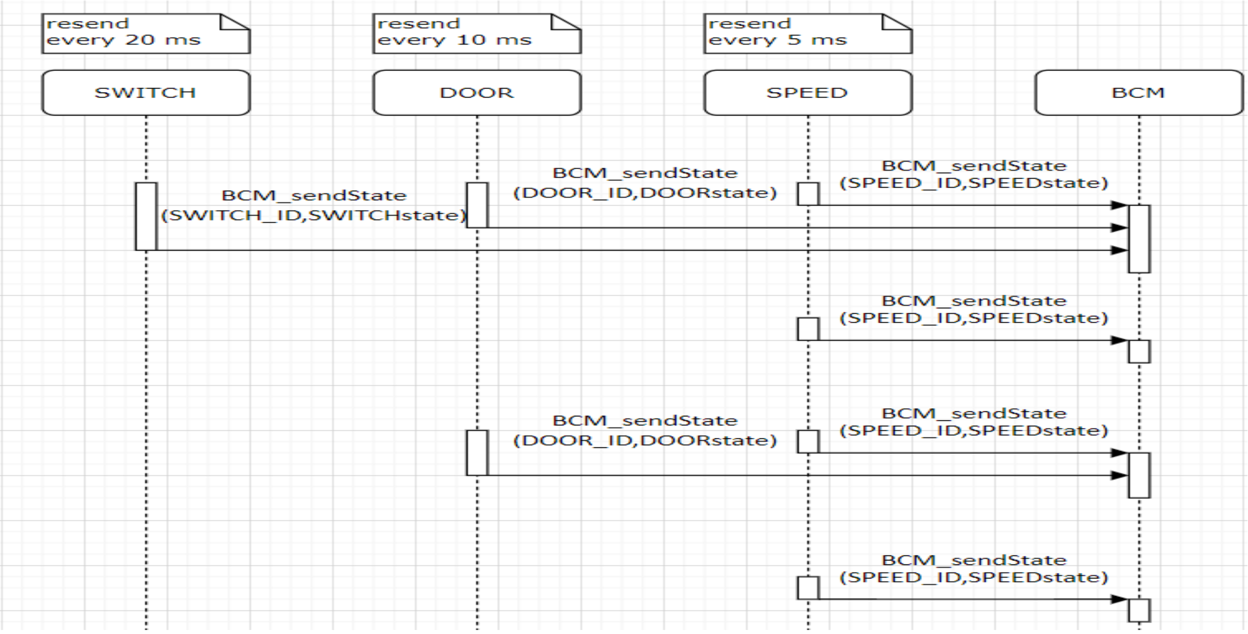
BCM is in critical section which is used to send data among ECUs.



BCM_SendState:



Sequence diagram:



CPU calculations:

Speed sensor task is 30us and P: 5 ms

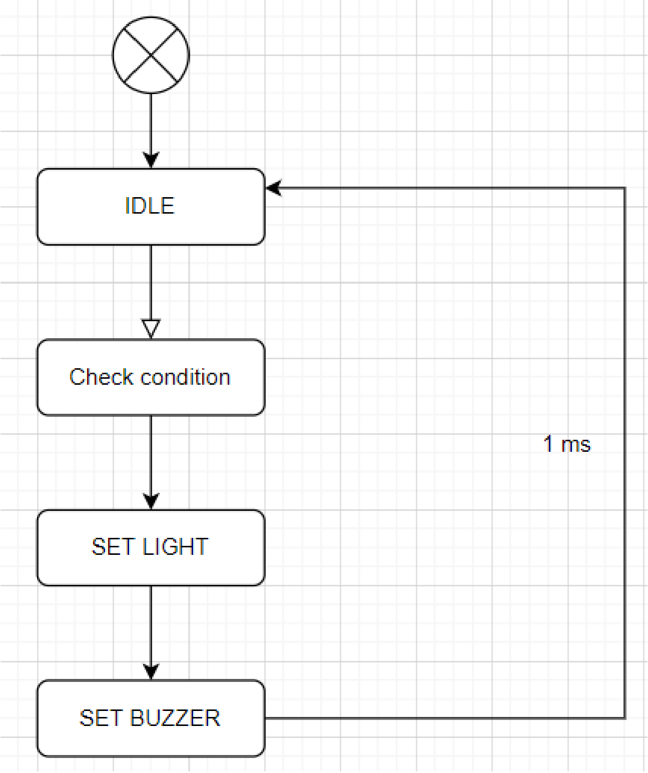
Door sensor task is 30us and P: 10 ms

Light switch task is 30us and P: 20 ms

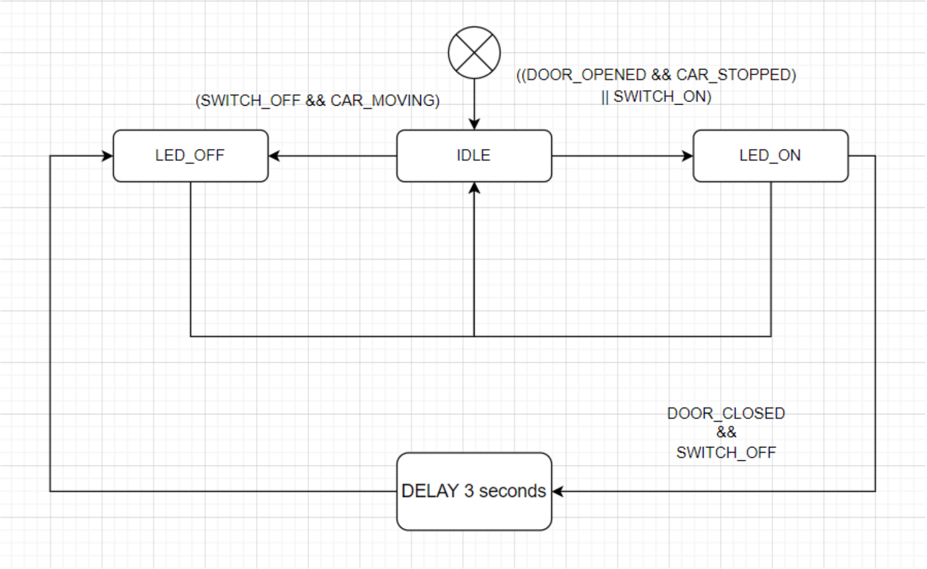
Hyper Period = 20 ms

CPU Load = ((0.03) + (0.03*2) + (0.03*4)) / 20 = 1.05%

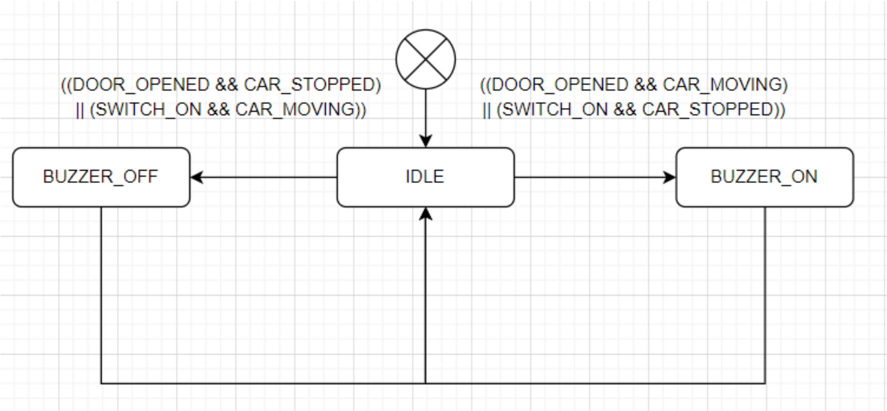
ECU 2



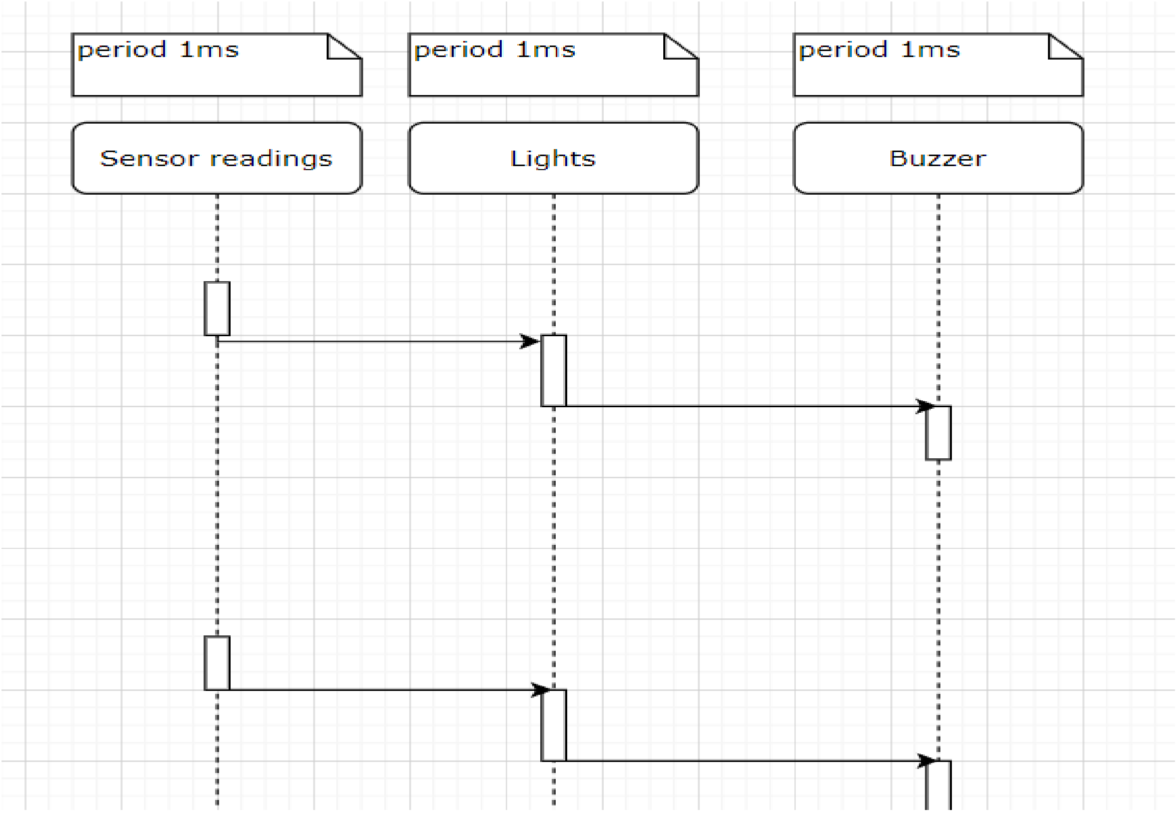
1. LL and RL (LEDs):



2. Buzzer:


((DOOR_OPENED && CAR_STOPPED)
|| (SWITCH_ON && CAR_MOVING))
((DOOR_OPENED && CAR_MOVING)
|| (SWITCH_ON && CAR_STOPPED))
BUZZER_OFF
IDLE
BUZZER_ON

Sequence diagram:



CPU calculations:

LED handling task is 100us and P: 1 ms

Buzzer handling task is 100us and P: 1 ms

BCM task is 50us and P: 1 ms

Hyper Period = 1 ms

CPU Load = ((0.05*5) + (0.1) + (0.1)) / 1 = 45%