وزارة الاتصالات
وتكنولوجيا المعلومات

Report on:

Automotive door control system design

Prepared by:

Abanoub Salah Mitry

Submitted To:

Udacity Team

Submitted Date:

Contents

1. Project Requirements

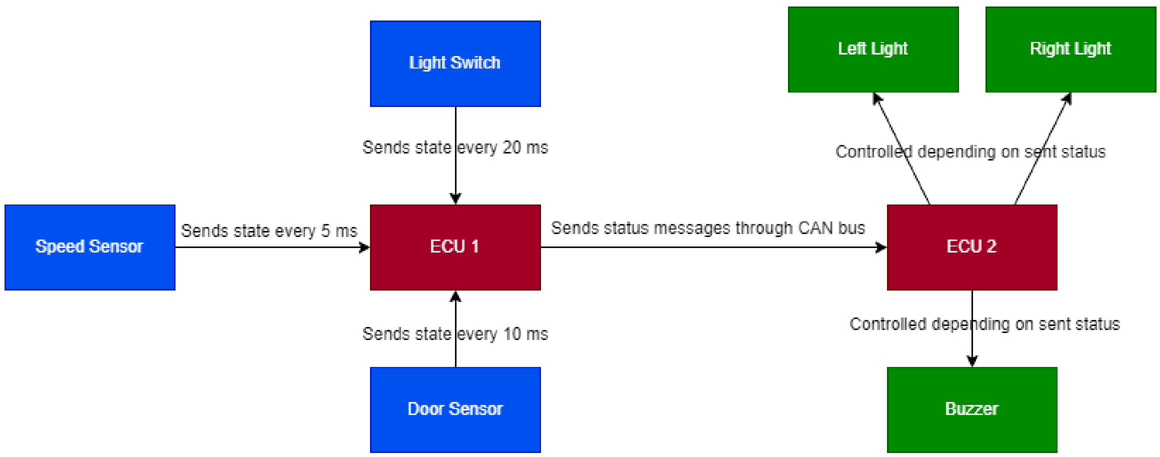
1.1. Hardware Requirements

1. Two microcontrollers connected via CAN bus
2. One Door sensor (D)
3. One Light switch (L)
4. One Speed sensor (S)
5. ECU 1 connected to D, S, and L, all input devices
6. Two lights, right (RL) and left (LL)
7. One buzzer (B)
8. ECU 2 connected to RL, LL, and B, all output devices

1.2. Software Requirements

1. ECU 1 will send status messages periodically to ECU 2 through the CAN protocol
2. Status messages will be sent using Basic Communication Module (BCM)
3. Door state message will be sent every 10ms to ECU 2
4. Light switch state message will be sent every 20ms to ECU 2
5. Speed state message will be sent every 5ms to ECU 2
6. Each ECU will have an OS and application SW components
7. If the door is opened while the car is moving → Buzzer ON, Lights OFF
8. If the door is opened while the car is stopped → Buzzer OFF, Lights ON
9. If the door is closed while the lights were ON → Lights are OFF after 3 seconds
10. If the car is moving and the light switch is pressed → Buzzer OFF, Lights ON
11. If the car is stopped and the light switch is pressed → Buzzer ON, Lights ON

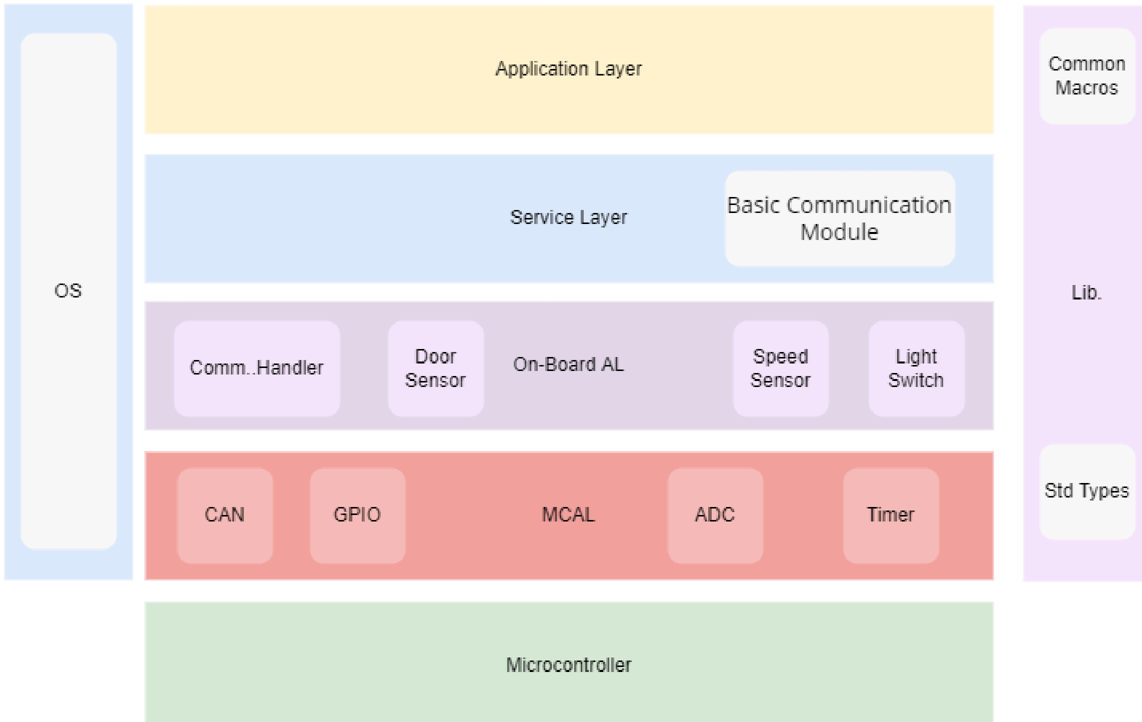
2. Block Diagram



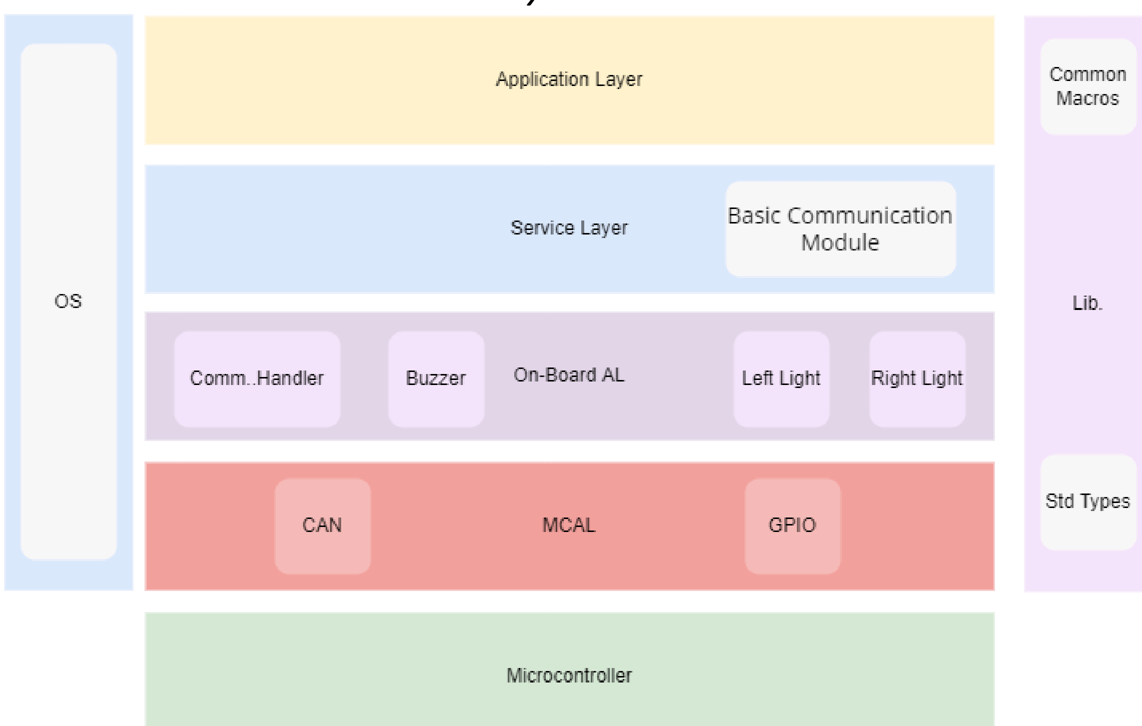
Block Diagram

3. Static Design

3.1. Layered Architecture



ECU 1 Layered architecture



ECU 2 Layered architecture

3.2. Components and Modules

- For ECU 1:

  ADC
  CAN
  GPIO
  Door Sensor
  Light Switch
  Speed Sensor
  Lib.
  Comm. Handler
  Basic Communication Module

- For ECU 2:

  CAN

  GPIO

  Buzzer

  Left Light

  Right Light

  Lib.

  Comm. Handler

  Basic Communication Module

3.3. APIs Details

- ADC

Service Name	ADC_init(void)
Parameters(in)	None
Return Value	None
Description	Initiates the ADC peripheral

Service Name	ADC_StartConversion(ADC_ChannelType ChannelID)	
Parameters(in)	ChannelId	ADC channel ID
Return Value	None	
Description	Start an ADC Channel conversion	

Service Name	ADC_StopConversion(ADC_ChannelType ChannelID)	
Parameters(in)	ChannelId	ADC channel ID
Return Value	None	
Description	Stop an ADC channel conversion	

Service Name	ADC_ReadChannel(ADC_ChannelType ChannelID, ADC_ValueChannelType *DataBufferPtr)	
Parameters(in)	ChannelId	ADC channel ID
	DataBufferPtr	ADC reading stored there
Return Value	Std_ReturnType	E_OK
		E_NOT_OK
Description	Read an ADC channel value	

Name	ADC_ChannelType
Kind	enum
Range	Range is µC specific
Description	ID of the ADC channel

Name	ADC_ValueChannelType
Kind	int
Range	Implementation specific
Description	ADC converted channel value

- CAN

Service Name	CAN_init(void)
Parameters(in)	None
Return Value	None
Description	Initiates the CAN peripheral

<table>
<tr><td>Service Name</td><td colspan="3">CAN_ReadChannel(CAN_ChannelType ChannelID, CAN_ValueChannelType *DataBufferPtr)</td></tr>
<tr><td rowspan="2">Parameters(in)</td><td>ChannelId</td><td colspan="2">CAN channel ID</td></tr>
<tr><td>DataBufferPtr</td><td colspan="2">CAN Rx stored there</td></tr>
<tr><td rowspan="2">Return Value</td><td colspan="2" rowspan="2">Std_ReturnType</td><td>E_OK</td></tr>
<tr><td>E_NOT_OK</td></tr>
<tr><td>Description</td><td colspan="3">Read a CAN channel value</td></tr>
</table>

<table>
<tr><td>Service Name</td><td colspan="2">CAN_WriteChannel(CAN_ChannelType ChannelId, CAN_ ValueChannelType Level)</td></tr>
<tr><td rowspan="2">Parameters(in)</td><td>ChannelId</td><td>GPIO channel ID</td></tr>
<tr><td>Level</td><td>Value to be written</td></tr>
<tr><td>Return Value</td><td colspan="2">None</td></tr>
<tr><td>Description</td><td colspan="2">Tx using a CAN Channel</td></tr>
</table>

Name	CAN_ChannelType
Kind	enum
Range	cover all available CAN channels
Description	Numeric ID of a CAN channel

Name	CAN_ValueChannelType
Kind	int
Range	Implementation specific
Description	CAN channel value

- GPIO

Service Name	GPIO_init(void)
Parameters(in)	None
Return Value	None
Description	Initiates the GPIO peripheral

Service Name	GPIO_ReadChannel(GPIO_ChannelType ChannelId)	
Parameters(in)	ChannelId	GPIO channel ID
Return Value	GPIO_LevelType	STD_HIGH
		STD_LOW
Description	Read a GPIO Channel	

Service Name	GPIO_WriteChannel(GPIO_ChannelType ChannelId, GPIO_LevelType Level)	
Parameters(in)	ChannelId	GPIO channel ID
	Level	Value to be written
Return Value	None	
Description	Write to a GPIO Channel	

Name	GPIO_ChannelType
Kind	enum
Range	cover all available GPIO channels
Description	Numeric ID of a GPIO channel

Name	GPIO_LevelType	
Kind	uint8	
Range	STD_LOW	0x0
	STD_HIGH	0x1
Description	Possible levels of the GPIO channel	

- Lib.

Name	Std_ReturnType	
Kind	uint8	
Range	E_OK	0x0
	E_NOT_OK	0x1
Description	Standard status return type	

- Basic Communication Module

Service Name	BCM_Write(BCM_ValueType Level)	
Parameters(in)	Level	Value to be written
Return Value	None	
Description	Tx using a BCM layer	

Service Name	BCM_Read(BCM_ValueType *Level)	
Parameters(in)	Level	Value to be written to
Return Value	None	
Description	Rx using a BCM layer	

Name	BCM_ValueType
Kind	Int
Range	Implementation specific
Description	BCM layer value

- Comm. Handler

Service Name	CommHandler_Write(BCM_ValueType Level)	
Parameters(in)	Level	Value to be written
Return Value	None	
Description	Tx using CAN bus	

Service Name	CommHandler_Read(BCM_ValueType *Level)	
Parameters(in)	Level	Value to be written to
Return Value	None	
Description	Rx using CAN bus	

Name	CommHandler_ValueType
Kind	Int
Range	Implementation specific
Description	Comm. handler layer value

- Door Sensor

Service Name	DoorSensor_Read(DoorSensor_ValueType *Level)	
Parameters(in)	Level	Value to be written to
Return Value	None	
Description	Read door sensor value	

Name	DoorSensor_ValueType
Kind	Int
Range	Implementation specific
Description	Door sensor value

- Light Switch

Service Name	LightSwitch_Read(LightSwitch_ValueType *Level)	
Parameters(in)	Level	Value to be written to
Return Value	None	
Description	Read light switch value	

Name	LightSwitch_ValueType
Kind	Int
Range	Implementation specific
Description	Light switch value

- Speed Sensor

Service Name	SpeedSensor_Read(SpeedSensor_ValueType *Level)	
Parameters(in)	Level	Value to be written to
Return Value	None	
Description	Read Speed sensor value	

Name	SpeedSensor_ValueType
Kind	Int
Range	Implementation specific
Description	Speed sensor value

- Buzzer

Service Name	Buzzer_Write(Buzze_ValueType *Level)	
Parameters(in)	Level	Value to be written to
Return Value	None	
Description	Write buzzer value	

Name	Buzzer_ValueType
Kind	Int
Range	Implementation specific
Description	Buzzer value

- ## Left Light

Service Name	LeftLight_Write(LeftLight_ValueType *Level)	
Parameters(in)	Level	Value to be written to
Return Value	None	
Description	Write left light value	

Name	LeftLight_ValueType
Kind	Int
Range	Implementation specific
Description	Left light value

- ## Right Light

Service Name	RightLight_Write(RightLight_ValueType *Level)	
Parameters(in)	Level	Value to be written to
Return Value	None	
Description	Write right light value	

Name	RightLight_ValueType
Kind	Int
Range	Implementation specific
Description	Right light value

3.4. Folder Structure

```
├───APPLICATION
│   ├───config
│   └───ECU1
│       ├───inc
│       └───src
├───HAL
│   ├───config
│   ├───DoorSensor
│   │   ├───inc
│   │   └───src
│   ├───LightSwitch
│   │   ├───inc
│   │   └───src
│   └───SpeedSensor
│       ├───inc
│       └───src
├───LIBRARIES
├───MCAL
│   ├───ADC
│   │   ├───inc
│   │   └───src
│   ├───CAN
│   │   ├───inc
│   │   └───src
│   ├───config
│   ├───GPIO
│   │   ├───inc
│   │   └───src
│   ├───GPT
│   │   ├───inc
│   │   └───src
│   ├───IntCtrl
│   │   ├───inc
│   │   └───src
│   ├───MCU
│   │   ├───inc
│   │   └───src
│   └───Port
│       ├───inc
│       └───src
└───SERVICE
```

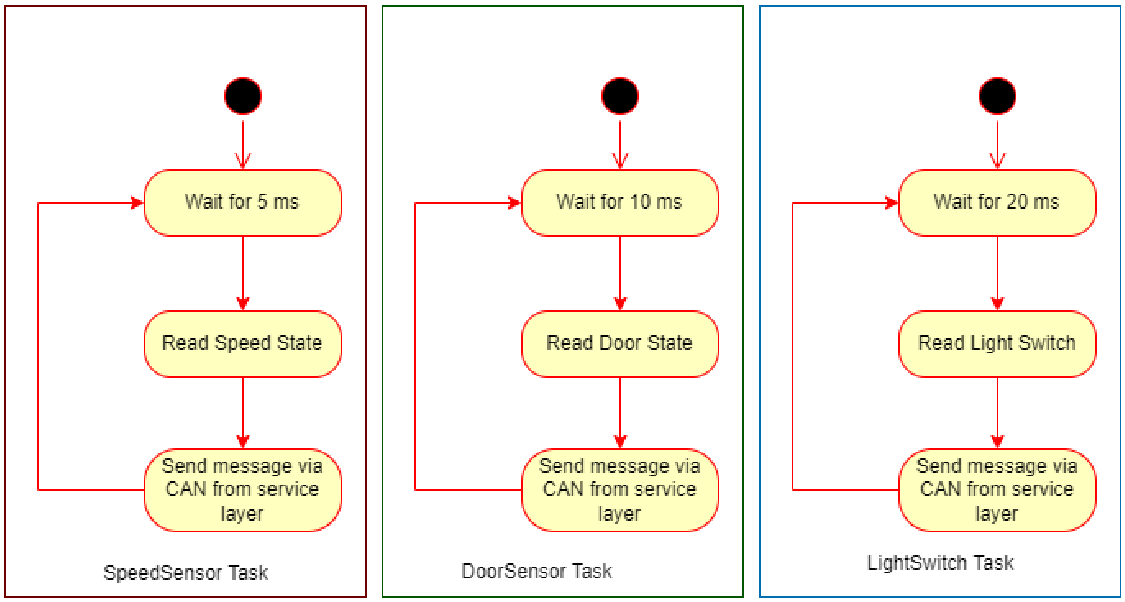
ECU 1

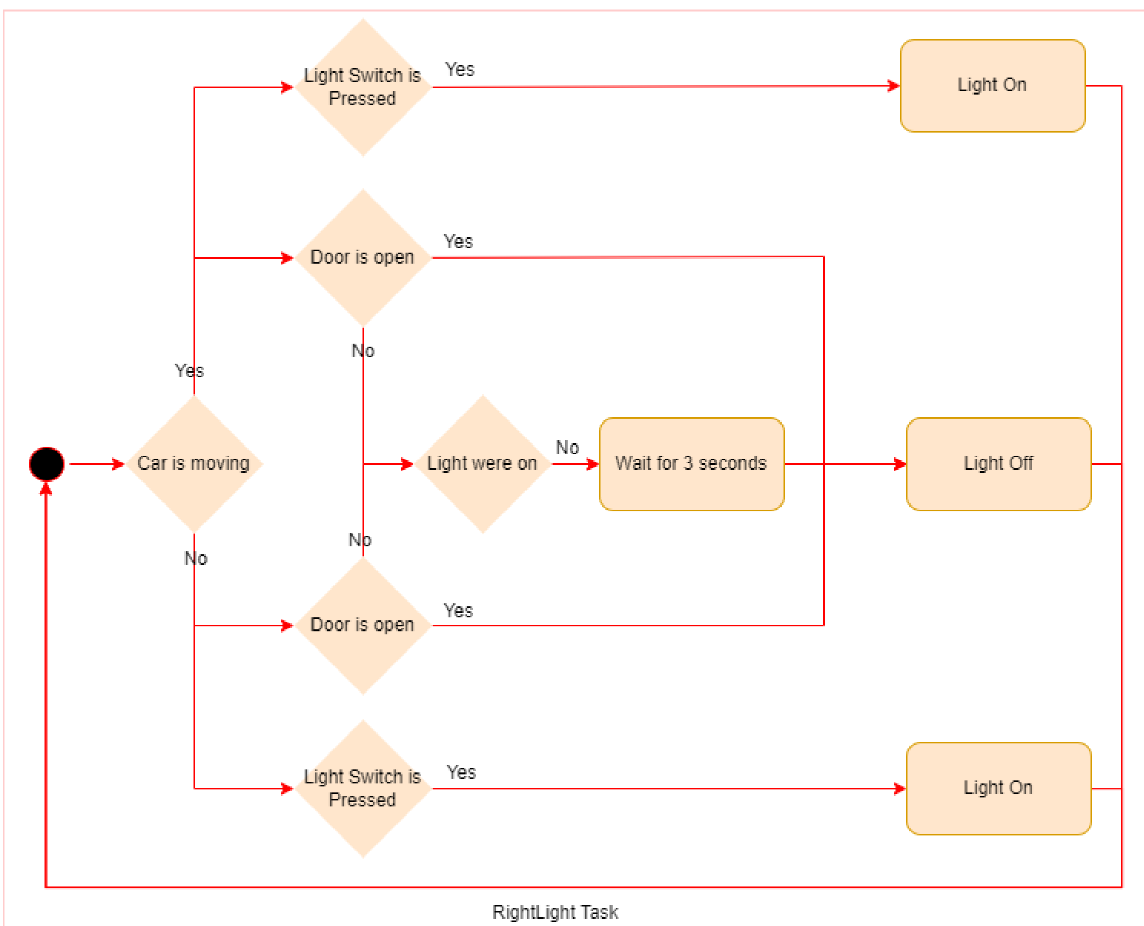
```
├───APPLICATION
│   ├───config
│   └───ECU2
│       ├───inc
│       └───src
├───HAL
│   ├───Buzzer
│   │   ├───inc
│   │   └───src
│   ├───config
│   ├───LeftLight
│   │   ├───inc
│   │   └───src
│   └───RightLight
│       ├───inc
│       └───src
├───LIBRARIES
├───MCAL
│   ├───CAN
│   │   ├───inc
│   │   └───src
│   ├───config
│   ├───GPIO
│   │   ├───inc
│   │   └───src
│   ├───IntCtrl
│   │   ├───inc
│   │   └───src
│   ├───MCU
│   │   ├───inc
│   │   └───src
│   └───Port
│       ├───inc
│       └───src
└───SERVICE
```

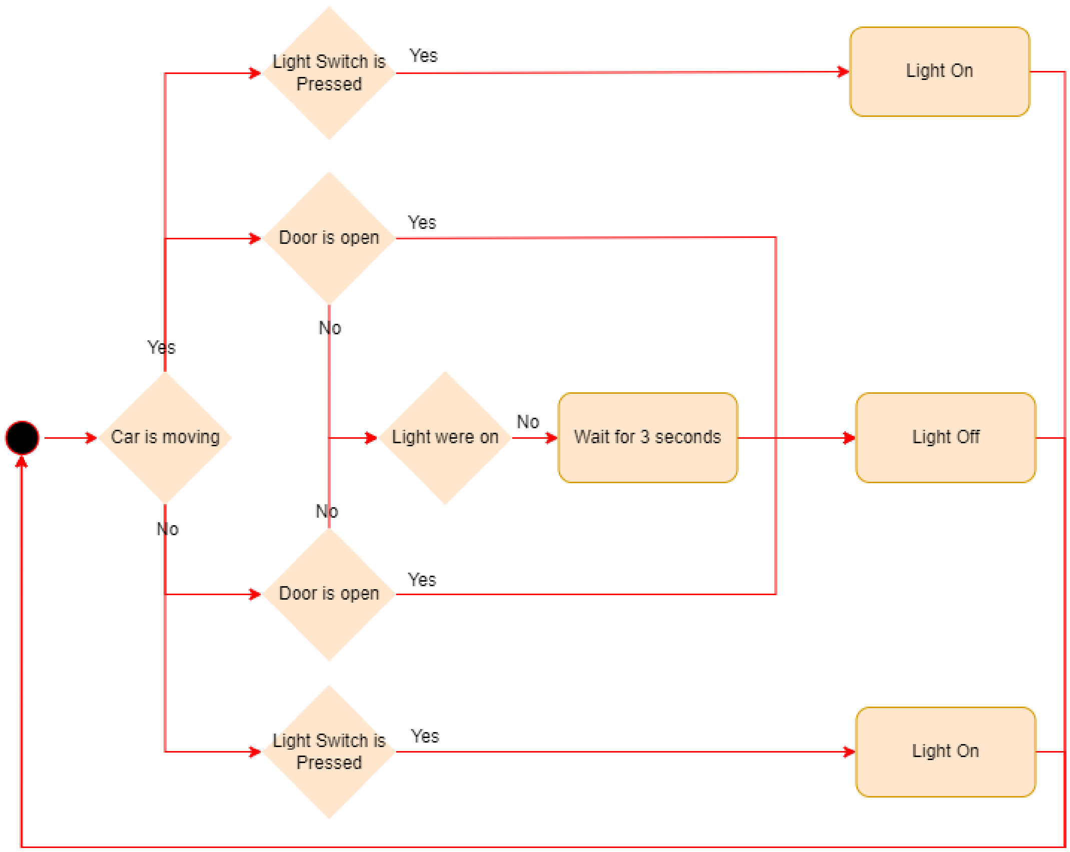
ECU 2

4. Dynamic Design

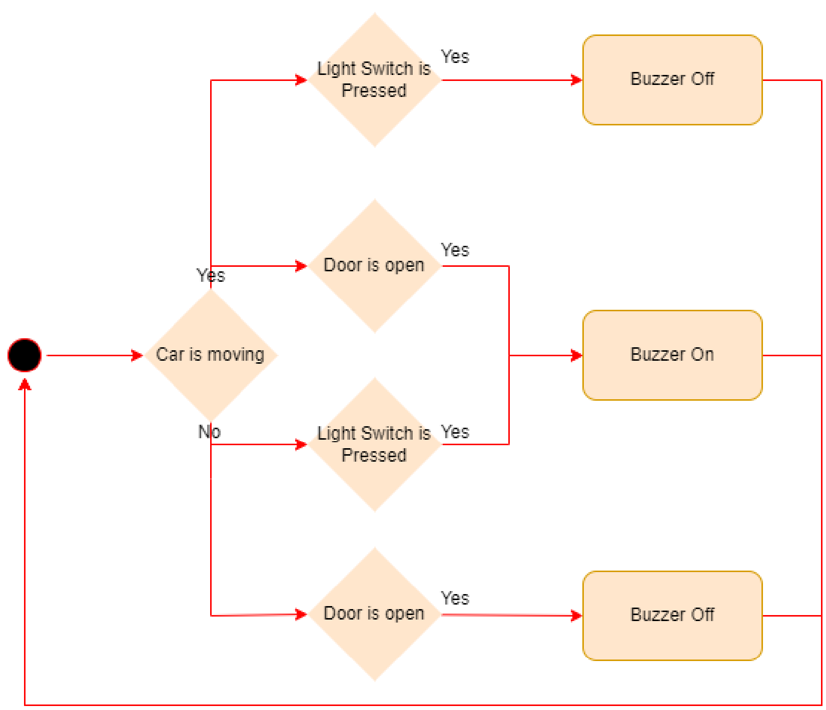
4.1. State Machine Diagrams





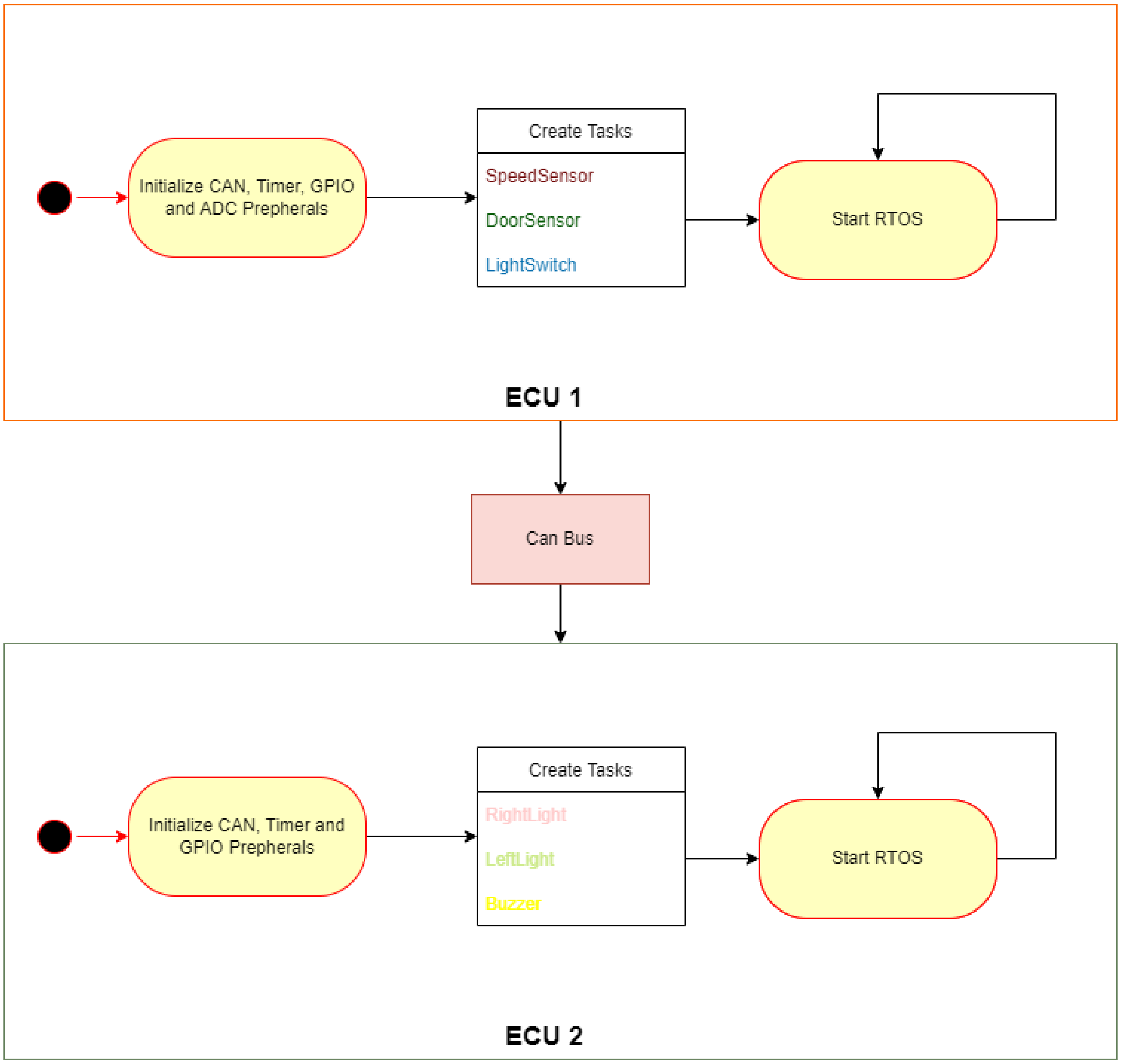


LeftLight Task

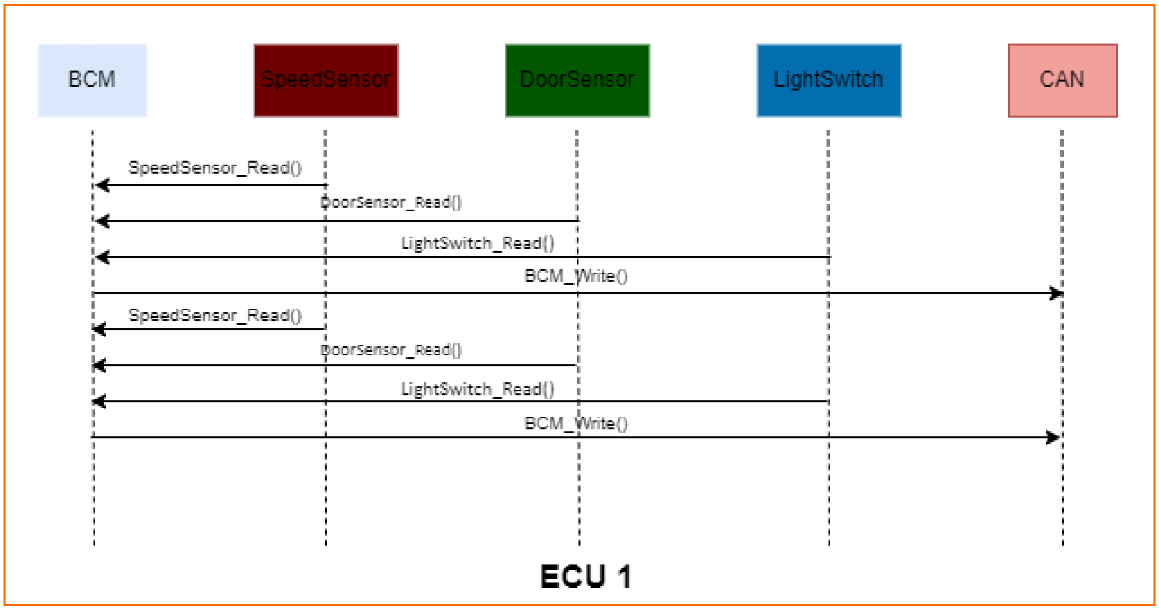


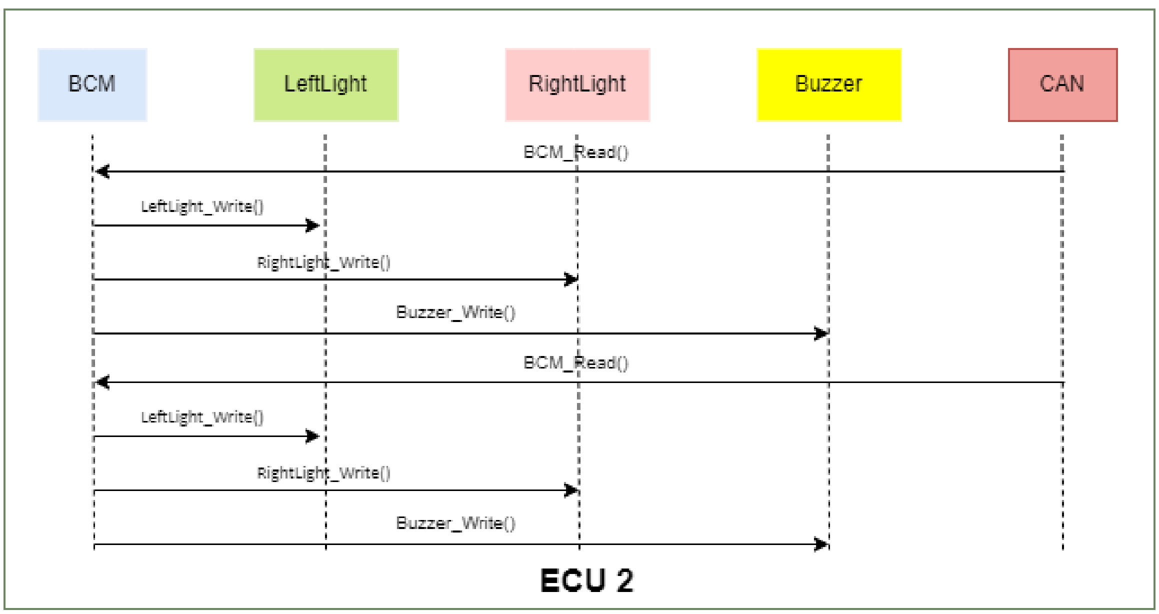
Buzzer Task

4.2. State Machine Diagram for ECU Operation



4.3. Sequence Diagram





4.4. CPU Load Calculation

ECU 1 Tasks	Period	Deadline	WCET
T1	X1	x1	y1
T2	X2	x2	y2
T3	X3	x3	y3

$$utilization(U) = \frac{y1}{x1} + \frac{y2}{x2} + \frac{y3}{x3}$$

ECU 2 Tasks	Period	Deadline	WCET
T1	u1	u1	v1
T2	u2	u2	v2
T3	u3	u3	v3

$$utilization(U) = \frac{v1}{u1} + \frac{v2}{u2} + \frac{v3}{u3}$$

4.5. Bus Load Calculation

Assume CAN frame of 64 bit and 1 Mbit/s

$$\therefore Frame\ Time = \frac{64\ bit}{1Mbit/s} \times \frac{1Mbit/s}{1000000\ bit/s} = 64\ \mu s$$

$$\therefore No.of\ frames\ in\ 1\ second$$

$$= \frac{1000\ ms}{5\ ms/frame} + \frac{1000\ ms}{10\ ms/frame} + \frac{1000\ ms}{20\ ms/frame} = \ 350\ frames$$

$$\therefore Total\ time\ on\ bus = 350\ frames\ * 64\ \mu s$$
$$= 22400\ \mu s$$

$$\therefore Bus\ load\ (\%) = \frac{44800\ \mu s}{1\ s} = 2.24\ \%$$