Static Design

Basic design:

ECU1

ECU2

CAN Protocol

Door sensor

Light Switch

Speed sensor

Light source left

Light source right

buzzer

Pseudocode:

1. ECU1:

MCAL Layer:

For identifying GPIO pins control a GPIO module is needed to specify input and output pins, reading, and writing to pins.

For communication (sending status massages) a CAN module is needed.

To read the speed we need to know if the motor is moving or stopped, so an ADC module is not needed.

To time the massages sending a TIMER module is needed.

On-Board Layer:

Speed sensor module is needed which uses GPIO module.

Light Switch module is needed which uses GPIO module.

Door sensor module is needed which uses GPIO module.

Communication handler module is needed to connect service layer with MCAL layer (CAN module).

Service layer:

OS to handle all the layers.

Communication manager will be needed.

Standard lib layer:

STD Types.

STD Macros.

Hardware standards.

Application layer:

Main file to make the program.

1. ECU2:

For identifying GPIO pins control a GPIO module is needed to specify input and output pins, reading, and writing to pins.

For communication (sending status massages) a CAN module is needed.

To time the massages sending a TIMER module is needed and the time for closing the light source.

On-Board Layer:

Light source module is needed which uses GPIO module.

Buzzer module is needed that uses GPIO module.

Communication handler module is needed to connect service layer with MCAL layer (CAN module).

Service layer:

OS to handle all the layers.

Communication manager will be needed.

Standard lib layer:

STD Types.

STD Macros.

Hardware standards.

Application layer:

Main file to make the program.

Finale layer architecture:

Application

CAN GPIO Timer

OS

Comm.

manager

Speed

Light\_Switch

Comm. Manager Door

Application

CAN GPIO Timer

OS

Comm.

manager

Light\_Source

Buzzer

Comm. Manager

ECU 1

ECU 2

Modules Contents:

Common Modules:

1. CAN:

APIs:

|  |  |
| --- | --- |
|  |  |
| Description | Initialize CAN |
| Reentrancy | Reentrant |
| Synchronization | Synchronous |
| Type | Init function |
| Input | CAN\_config\_t \* |
| Output | None |
| Return | None |

* 1. CAN\_vidInit
  2. CAN\_u8SendData

|  |  |
| --- | --- |
|  |  |
| Description | Send data through the CAN |
| Reentrancy | Non Reentrant |
| Synchronization | Synchronous |
| Type | Sender |
| Input | CAN\_Data\_t \* |
| Output | Number of bytes sent |
| Return | 1. (if data is sent) 0(if some data is not sent) |

* 1. CAN\_xRecieveData

|  |  |
| --- | --- |
|  |  |
| Description | Receive data through the CAN |
| Reentrancy | Reentrant |
| Synchronization | Asynchronous |
| Type | Getter |
| Input | None |
| Output | None |
| Return | CAN\_Data\_t \* |

Typedefs:

1. CAN\_config\_t

A structure that contains all required data to configure CAN communication.

1. CAN\_Data\_t

The structure that contains the form of data sent.

1. Communication\_Manager

APIs:

* 1. CANM\_Init()

|  |  |
| --- | --- |
|  |  |
| Description | Initiate the CAN module |
| Reentrancy | Reentrant |
| Synchronization | synchronous |
| Type | Init |
| Input | None |
| Output | None |
| Return | None |

* 1. CANM\_SendStatus

|  |  |
| --- | --- |
|  |  |
| Description | Receive data through the CAN |
| Reentrancy | Reentrant |
| Synchronization | Asynchronous |
| Type | Setter |
| Input | 1. StatusSent 2- CurrentStatus |
| Output | None |
| Return | None |

1. GPIO

APIs:

* 1. GPIO\_vidInit

|  |  |
| --- | --- |
|  |  |
| Description | Initialize GPIO pins |
| Reentrancy | Reentrant |
| Synchronization | Synchronous |
| Type | Init function |
| Input | None (use an array of structures inside GPIO\_Config) |
| Output | None |
| Return | None |

* 1. GPIO\_u8GetPinVal

|  |  |
| --- | --- |
|  |  |
| Description | Get a pin status |
| Reentrancy | Reentrant |
| Synchronization | Synchronous |
| Type | Getter |
| Input | portNumber pinNumber |
| Output | None |
| Return | HIGH LOW |

* 1. GPIO\_vidSetPinVal

|  |  |
| --- | --- |
|  |  |
| Description | Get a pin status |
| Reentrancy | Reentrant |
| Synchronization | Synchronous |
| Type | Sender |
| Input | portNumber pinNumber Status |
| Output | None |
| Return | None |

* 1. GPIO\_vidSetPinDir

|  |  |
| --- | --- |
|  |  |
| Description | Choose pin direction |
| Reentrancy | Reentrant |
| Synchronization | Synchronous |
| Type | Sender |
| Input | portNumber pinNumber Direction |
| Output | None |
| Return | None |

* 1. GPIO\_vidInitPin

|  |  |
| --- | --- |
|  |  |
| Description | Choose pin direction and value |
| Reentrancy | Reentrant |
| Synchronization | Synchronous |
| Type | Sender |
| Input | portNumber pinNumber Direction Value |
| Output | None |
| Return | None |

* 1. GPIO\_vidEnablePin

|  |  |
| --- | --- |
|  |  |
| Description | Enable/Disable pin |
| Reentrancy | Reentrant |
| Synchronization | Synchronous |
| Type | Setter |
| Input | portNumber pinNumber EN(0 -> disable, 1 -> enable) |
| Output | None |
| Return | None |

Typedefs:

1. GPIO\_config\_t

A structure that contains all required data to configure a GPIO pin.

1. GPIO\_pinNumber

An enum that contains the available pins.

1. GPIO\_portNumber

An enum that contains the available ports.

1. GPIO\_pinValue

An enum that contains High and Low statuses.

1. GPIO\_pinDirection

An enum that contains Output and Input.

1. Timer:

APIs:

* 1. Timer\_vidInit

|  |  |
| --- | --- |
|  |  |
| Description | Initialize Timers |
| Reentrancy | Reentrant |
| Synchronization | Synchronous |
| Type | Init function |
| Input | None (use an array of structures inside Timer\_Config) |
| Output | None |
| Return | None |

* 1. Timer\_vidStart

|  |  |
| --- | --- |
|  |  |
| Description | Start a Timer |
| Reentrancy | Reentrant |
| Synchronization | Synchronous |
| Type |  |
| Input | Time in ms (or number of ticks) timer\_number |
| Output | Edit to active\_Timer array (put 1 in the array’s element that specifies the timer’s number) |
| Return | None |

* 1. Timer\_vidStop

|  |  |
| --- | --- |
|  |  |
| Description | Stop a Timer |
| Reentrancy | Reentrant |
| Synchronization | Synchronous |
| Type | Init function |
| Input | Time in ms (or number of ticks) timer\_number |
| Output | Edit to active\_Timer array (put zero) |
| Return | None |

* 1. Timer\_ISR

|  |  |
| --- | --- |
|  |  |
| Description | Timer ISR function |
| Reentrancy | Non-Reentrant |
| Synchronization | Synchronous |
| Type | Setter |
| Input | None |
| Output | Add 1 to tickCount |
| Return | None |

Typedefs:

1. Timer\_config\_t

A structure that contains all required data to configure a Timer pin.

1. Timer\_Id

An enum that contains all available timers.

ECU1 Modules:

1. Speed

APIs:

1. SpeedSensor\_vidInit

|  |  |
| --- | --- |
|  |  |
| Description | Initiate speed sensor |
| Reentrancy | Reentrant |
| Synchronization | Synchronous |
| Type | Init function |
| Input | None |
| Output | None |
| Return | None |

1. SpeedSensor\_ u8TurnOn

|  |  |
| --- | --- |
|  |  |
| Description | Turn on speed sensor |
| Reentrancy | Reentrant |
| Synchronization | Synchronous |
| Type |  |
| Input | None |
| Output | None |
| Return | 1(If done) 0(if not) |

1. SpeedSensor\_u8TurnOff

|  |  |
| --- | --- |
|  |  |
| Description | Turn off speed sensor |
| Reentrancy | Reentrant |
| Synchronization | Synchronous |
| Type |  |
| Input | None |
| Output | None |
| Return | 1(If done) 0(if not) |

1. SpeedSensor\_vidGetStatus

|  |  |
| --- | --- |
|  |  |
| Description | Get Motor Status |
| Reentrancy | Non-Reentrant |
| Synchronization | Synchronous |
| Type | Getter |
| Input | None |
| Output | MotorStatus (global variable) |
| Return | None |

1. SpeedSensor\_vidSendStatus

|  |  |
| --- | --- |
|  |  |
| Description | Send Motor Status |
| Reentrancy | Non-Reentrant |
| Synchronization | Asynchronous |
| Type | Sender |
| Input | None |
| Output | None |
| Return | None |

1. Light\_Switch

APIs:

1. LightSwitch\_vidInit

|  |  |
| --- | --- |
|  |  |
| Description | Initiate Light switch |
| Reentrancy | Reentrant |
| Synchronization | Synchronous |
| Type | Init function |
| Input | None |
| Output | None |
| Return | None |

1. LightSwitch\_vidGetStatus

|  |  |
| --- | --- |
|  |  |
| Description | Get status of Light switch |
| Reentrancy | Reentrant |
| Synchronization | Synchronous |
| Type | Init function |
| Input | None |
| Output | LightStatus |
| Return | None |

1. LightSwitch\_vidSendStatus

|  |  |
| --- | --- |
|  |  |
| Description | Send Light Status |
| Reentrancy | Non-Reentrant |
| Synchronization | Asynchronous |
| Type | Sender |
| Input | None |
| Output | None |
| Return | None |

1. Door

APIs:

1. DoorSensor\_vidInit

|  |  |
| --- | --- |
|  |  |
| Description | Initiate Door sensor |
| Reentrancy | Reentrant |
| Synchronization | Synchronous |
| Type | Init function |
| Input | None |
| Output | None |
| Return | None |

1. DoorSensor\_ u8TurnOn

|  |  |
| --- | --- |
|  |  |
| Description | Turn on Door sensor |
| Reentrancy | Reentrant |
| Synchronization | Synchronous |
| Type | init |
| Input | None |
| Output | None |
| Return | 1(if ok) 0(if not ok) |

1. DoorSensor\_u8TurnOff

|  |  |
| --- | --- |
|  |  |
| Description | Turn off Door sensor |
| Reentrancy | Reentrant |
| Synchronization | Synchronous |
| Type | De-init |
| Input | None |
| Output | None |
| Return | 1(if ok) 0(if not ok) |

1. DoorSensor\_vidGetStatus

|  |  |
| --- | --- |
|  |  |
| Description | Get Door Status |
| Reentrancy | Non-Reentrant |
| Synchronization | Synchronous |
| Type | Getter |
| Input | None |
| Output | DoorStatus (global variable) |
| Return | None |

1. DoorSensor\_vidSendStatus

|  |  |
| --- | --- |
|  |  |
| Description | Send Door Status |
| Reentrancy | Non-Reentrant |
| Synchronization | Asynchronous |
| Type | Sender |
| Input | None |
| Output | None |
| Return | None |

ECU2 Modules:

1. Light\_Source

APIs:

1. LightSource\_vidInit

|  |  |
| --- | --- |
|  |  |
| Description | Initiate Light source |
| Reentrancy | Reentrant |
| Synchronization | Synchronous |
| Type | Init function |
| Input | None |
| Output | None |
| Return | None |

1. LightSource\_u8ChangeStatus

|  |  |
| --- | --- |
|  |  |
| Description | Change status of a Light source |
| Reentrancy | Reentrant |
| Synchronization | Synchronous |
| Type | Setter |
| Input | The status you want to change the light source to (HIGH or LOW) |
| Output | None |
| Return | The current status of the changed Light source. |

1. Buzzer

APIs:

1. Buzzer\_vidInit

|  |  |
| --- | --- |
|  |  |
| Description | Initiate Buzzer |
| Reentrancy | Reentrant |
| Synchronization | Synchronous |
| Type | Init function |
| Input | None |
| Output | None |
| Return | None |

1. Buzzer\_u8ChangeStatus

|  |  |
| --- | --- |
|  |  |
| Description | Change status of the buzzer |
| Reentrancy | Reentrant |
| Synchronization | Synchronous |
| Type | Setter |
| Input | The status you want to change the Buzzer to. (HIGH or LOW) |
| Output | None |
| Return | The current status of the buzzer source. |