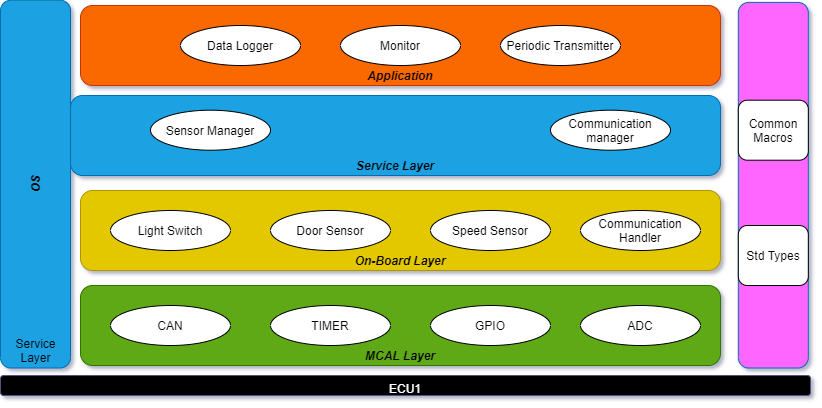
Static Design Analysis

## ECU 1: -

1. Layered Architecture



Notes:

* Door sensor receives digital value, high if closed and low if it’s open.
* Speed sensor receives an analog value of car speed and then converts it to digital high or low.
* The sensor manager controls the light switcher to receive digital value.
* We have three tasks (door sensor, speed sensor, and light switch); each does the following monitor (Sensor Manager), log readings and periodic transmission via CAN bus (Communication Manager).

1. Modules

Diagram

Description automatically generated

1. ECU 1 APIs

|  |  |
| --- | --- |
| Module | APIs |
| Application Tasks | doorSensorTask  speedSensorTask  lightSwitchTask |

|  |  |
| --- | --- |
| Syntax | void doorSensorTask (void) |
| Sync/Async | Synchronous |
| Reentrancy | Non-Reentrant |
| Parameters | None |
| Return | None |
| Description | Door Sensor Task |

|  |  |
| --- | --- |
| Syntax | void speedSensorTask (void) |
| Sync/Async | Synchronous |
| Reentrancy | Non-Reentrant |
| Parameters | None |
| Return | None |
| Description | Speed Sensor Task |

|  |  |
| --- | --- |
| Syntax | void lightSwitchTask (void) |
| Sync/Async | Synchronous |
| Reentrancy | Non-Reentrant |
| Parameters | None |
| Return | None |
| Description | Light Switch Task |

|  |  |
| --- | --- |
| Module | APIs |
| DATA LOGGER | dataLog\_Save |

|  |  |
| --- | --- |
| Syntax | void dataLog\_Save (uint64 data) |
| Sync/Async | Synchronous |
| Reentrancy | Non-Reentrant |
| Parameters | Data |
| Return | None |
| Description | Save the logged data. |

|  |  |
| --- | --- |
| Module | APIs |
| (BCM)  Communication manager | BCM\_Manager |

|  |  |
| --- | --- |
| Syntax | void BCM\_Manager (uint64 Data, uint8 Bus\_ID) |
| Sync/Async | Synchronous |
| Reentrancy | Non-Reentrant |
| Parameters | Data, Bus id |
| Return | None |
| Description | Coordinates communication requests |

|  |  |
| --- | --- |
| Module | APIs |
| Sensor Manager | sensorManager |

|  |  |
| --- | --- |
| Syntax | Uint8 sensorManager (uint8 Sensor\_ID) |
| Sync/Async | Synchronous |
| Reentrancy | Non-Reentrant |
| Parameters | Sensor\_ID |
| Return | Sensor status |
| Description | Read the selected sensor. |

|  |  |
| --- | --- |
| Module | APIs |
| Communication Handler | BCM\_Handler |

|  |  |
| --- | --- |
| Syntax | void BCM\_Handler (uint64 data, uint8 Bus\_ID) |
| Sync/Async | Synchronous |
| Reentrancy | Non-Reentrant |
| Parameters | Data, Bus\_ID |
| Return | None |
| Description | Coordinates communication requests (HW layer) |

|  |  |
| --- | --- |
| Module | APIs |
| Sensor Handler | Sensor\_Handler |

|  |  |
| --- | --- |
| Syntax | uint8 Sensor\_Handler (uint8 Sensor\_ID) |
| Sync/Async | Synchronous |
| Reentrancy | Non-Reentrant |
| Parameters | Sensor\_ID |
| Return | Data |
| Description | Read the selected sensor (HW Layer) |

|  |  |
| --- | --- |
| Module | APIs |
| Door Sensor | doorSensor\_Init  doorSensor\_Status |

|  |  |
| --- | --- |
| Syntax | ERROR\_STATE doorSensor\_Init (void) |
| Sync/Async | Synchronous |
| Reentrancy | Non-Reentrant |
| Parameters | None |
| Return | ERROR\_STATE |
| Description | Initialise Door Sensor |

|  |  |
| --- | --- |
| Syntax | uint8 doorSensor\_Status (void) |
| Sync/Async | Synchronous |
| Reentrancy | Non-Reentrant |
| Parameters | None |
| Return | Sensor\_Status |
| Description | Read Door Sensor Status |

|  |  |
| --- | --- |
| Module | APIs |
| speed sensor | speedSensor\_Init  speedSensor\_Status |

|  |  |
| --- | --- |
| Syntax | ERROR\_STATE speedSensor\_Init (void) |
| Sync/Async | Synchronous |
| Reentrancy | Non-Reentrant |
| Parameters | None |
| Return | ERROR\_STATE |
| Description | Initialise Speed Sensor |

|  |  |
| --- | --- |
| Syntax | uint8 speedSensor\_Status (void) |
| Sync/Async | Synchronous |
| Reentrancy | Non-Reentrant |
| Parameters | None |
| Return | Sensor\_Status |
| Description | Read Speed Sensor Status |

|  |  |
| --- | --- |
| Module | APIs |
| Light switch | lightSwitch\_Init  lightSwitch\_Status |

|  |  |
| --- | --- |
| Syntax | ERROR\_STATE lightSwitch\_Init (void) |
| Sync/Async | Synchronous |
| Reentrancy | Non-Reentrant |
| Parameters | None |
| Return | ERROR\_STATE |
| Description | Initialise Light Switch |

|  |  |
| --- | --- |
| Syntax | uint8 lightSwitch\_Status (void) |
| Sync/Async | Synchronous |
| Reentrancy | Non-Reentrant |
| Parameters | None |
| Return | Switch\_Status |
| Description | Read Light Switch Status |

|  |  |
| --- | --- |
| Module | APIs |
| dio | DIO\_Init  DIO\_WriteChannel  DIO\_ReadChannel |

|  |  |
| --- | --- |
| Syntax | ERROR\_STATE DIO\_Init (void) |
| Sync/Async | Synchronous |
| Reentrancy | Non-Reentrant |
| Parameters | None |
| Return | ERROR\_STATE |
| Description | Initialise DIO pins |

|  |  |
| --- | --- |
| Syntax | void DIO\_WriteChannel(DIO\_PinType Pin\_ID, DIO\_PinLevel Value) |
| Sync/Async | Synchronous |
| Reentrancy | Non-Reentrant |
| Parameters | Pin\_ID, Value |
| Return | None |
| Description | Write Value on selected pin. |

|  |  |
| --- | --- |
| Syntax | DIO\_PinLevel DIO\_ReadChannel(DIO\_PinType Pin\_ID) |
| Sync/Async | Synchronous |
| Reentrancy | Non-Reentrant |
| Parameters | Pin\_ID |
| Return | Value |
| Description | Read Value from the selected pin. |

|  |  |
| --- | --- |
| Module | APIs |
| ADC | ADC\_Init  ADC\_ReadChannel |

|  |  |
| --- | --- |
| Syntax | ERROR\_STATE ADC\_Init (uint8 Pin\_ID) |
| Sync/Async | Synchronous |
| Reentrancy | Non-Reentrant |
| Parameters | Pin\_ID |
| Return | ERROR\_STATE |
| Description | Initialise ADC to selected pins |

|  |  |
| --- | --- |
| Syntax | uint8 ADC\_ReadChannel(uint8 Pin\_ID) |
| Sync/Async | Synchronous |
| Reentrancy | Non-Reentrant |
| Parameters | Pin\_ID |
| Return | Value |
| Description | Read Value from the selected pin. |

|  |  |
| --- | --- |
| Module | APIs |
| timer | Timer\_Init  Timer\_Start  Timer\_Stop |

|  |  |
| --- | --- |
| Syntax | ERROR\_STATE Timer\_Init (void) |
| Sync/Async | Synchronous |
| Reentrancy | Non-Reentrant |
| Parameters | None |
| Return | ERROR\_STATE |
| Description | Initialise internal timer |

|  |  |
| --- | --- |
| Syntax | void Timer\_Start (TimerTickType ticks) |
| Sync/Async | Synchronous |
| Reentrancy | Non-Reentrant |
| Parameters | Ticks |
| Return | None |
| Description | Set desired ticks in the timer |

|  |  |
| --- | --- |
| Syntax | void Timer\_Stop (void) |
| Sync/Async | Synchronous |
| Reentrancy | Non-Reentrant |
| Parameters | None |
| Return | None |
| Description | Stop timer |

|  |  |
| --- | --- |
| Module | APIs |
| CAN | CAN\_Init  CAN\_Transmit |

|  |  |
| --- | --- |
| Syntax | ERROR\_STATE CAN\_Init (void) |
| Sync/Async | Synchronous |
| Reentrancy | Non-Reentrant |
| Parameters | None |
| Return | ERROR\_STATE |
| Description | Initialise CAN communication |

|  |  |
| --- | --- |
| Syntax | void CAN\_Transmit(DIO\_PinType Pin\_ID, uint64 Data) |
| Sync/Async | Synchronous |
| Reentrancy | Non-Reentrant |
| Parameters | Pin\_ID, Data |
| Return | None |
| Description | Transmit data via CAN |

|  |  |
| --- | --- |
| Module | APIs |
| Periodic Transmitter | periodicTransmitter\_Init  PeriodicTransmitter\_Send |

|  |  |
| --- | --- |
| Syntax | ERROR\_STATE periodicTransmitter\_Init (void) |
| Sync/Async | Synchronous |
| Reentrancy | Non-Reentrant |
| Parameters | None |
| Return | ERROR\_STATE |
| Description | Initialise periodic transmission |

|  |  |
| --- | --- |
|  |  |
| Syntax | void periodicTransmitter\_Send(DIO\_PinType Pin\_ID, uint64 Data) |
| Sync/Async | Synchronous |
| Reentrancy | Non-Reentrant |
| Parameters | Pin\_ID, Data |
| Return | None |
| Description | Send periodic status to ECU 1 via CAN. |

1. Typedef

|  |  |
| --- | --- |
| typedef unsigned char uint8 |  |
| typedef unsigned long long uint64 | CAN Frame |
| typedef uint8 DIO\_Pin\_Level | HIGH = 1, LOW = 0 |
| typedef uint8 DIO\_Pin\_Type | OUTPUT = 1, LOW = 0 |

## ECU 2: -

1. Layered Architecture

Graphical user interface, website

Description automatically generated

Notes

* We have one task that receives the status message via CAN bus and checks their values versus the initial value and does some changes based on the conditions

1. Modules

Diagram

Description automatically generated

1. ECU 2 APIs

|  |  |
| --- | --- |
| Module | APIs |
| DATA LOGGER | dataLog\_Save |

|  |  |
| --- | --- |
| Syntax | void dataLog\_Save (uint64 data) |
| Sync/Async | Synchronous |
| Reentrancy | Non-Reentrant |
| Parameters | Data |
| Return | None |
| Description | Save the logged data. |

|  |  |
| --- | --- |
| Module | APIs |
| (BCM)  Communication manager | BCM\_Manager |

|  |  |
| --- | --- |
| Syntax | void BCM\_Manager (uint64 Data, uint8 Bus\_ID) |
| Sync/Async | Synchronous |
| Reentrancy | Non-Reentrant |
| Parameters | Data, Bus id |
| Return | None |
| Description | Coordinates communication requests |

|  |  |
| --- | --- |
| Module | APIs |
| Actuator Manager | actuatorManager |

|  |  |
| --- | --- |
| Syntax | void actutatorManager (uint8 Actuator\_ID, uint8 Value) |
| Sync/Async | Synchronous |
| Reentrancy | Non-Reentrant |
| Parameters | Actuator\_ID, Value |
| Return | None |
| Description | Select the actuator and change its value |

|  |  |
| --- | --- |
| Module | APIs |
| Communication Handler | BCM\_Handler |

|  |  |
| --- | --- |
| Syntax | void BCM\_Handler (uint64 data, uint8 Bus\_ID) |
| Sync/Async | Synchronous |
| Reentrancy | Non-Reentrant |
| Parameters | Data, Bus\_ID |
| Return | None |
| Description | Coordinates communication requests(HW layer) |

|  |  |
| --- | --- |
| Module | APIs |
| actuator Handler | Actuator\_Handler |

|  |  |
| --- | --- |
| Syntax | void Actuator\_Handler (uint8 Actuator\_ID, uint8 value) |
| Sync/Async | Synchronous |
| Reentrancy | Non-Reentrant |
| Parameters | Actutator\_ID, Value |
| Return | None |
| Description | Read the selected actuator (HW Layer) |

|  |  |
| --- | --- |
| Module | APIs |
| Buzzer | Buzzer\_Init  Buzzer\_Start  Buzzer\_Stop |

|  |  |
| --- | --- |
| Syntax | ERROR\_STATE Buzzer\_Init (void) |
| Sync/Async | Synchronous |
| Reentrancy | Non-Reentrant |
| Parameters | None |
| Return | ERROR\_STATE |
| Description | Initialise Buzzer |

|  |  |
| --- | --- |
| Syntax | void Buzzer\_Start (void) |
| Sync/Async | Synchronous |
| Reentrancy | Non-Reentrant |
| Parameters | Ticks |
| Return | None |
| Description | Turn on the buzzer |

|  |  |
| --- | --- |
| Syntax | void Buzzer\_Stop (void) |
| Sync/Async | Synchronous |
| Reentrancy | Non-Reentrant |
| Parameters | None |
| Return | None |
| Description | Stop Buzzer |

|  |  |
| --- | --- |
| Module | APIs |
| Lights | Lights\_Init  Lights\_ON  Lights\_OFF |

|  |  |
| --- | --- |
| Syntax | ERROR\_STATE Lights\_Init (void) |
| Sync/Async | Synchronous |
| Reentrancy | Non-Reentrant |
| Parameters | None |
| Return | ERROR\_STATE |
| Description | Initialise the Lights |

|  |  |
| --- | --- |
| Syntax | void Lights\_ON (void) |
| Sync/Async | Synchronous |
| Reentrancy | Non-Reentrant |
| Parameters | Ticks |
| Return | None |
| Description | Turn on the Lights |

|  |  |
| --- | --- |
| Syntax | void Lights\_OFF (void) |
| Sync/Async | Synchronous |
| Reentrancy | Non-Reentrant |
| Parameters | None |
| Return | None |
| Description | Turn off Lights |

|  |  |
| --- | --- |
| Module | APIs |
| dio | DIO\_Init  DIO\_WriteChannel  DIO\_ReadChannel |

|  |  |
| --- | --- |
| Syntax | ERROR\_STATE DIO\_Init (void) |
| Sync/Async | Synchronous |
| Reentrancy | Non-Reentrant |
| Parameters | None |
| Return | ERROR\_STATE |
| Description | Initialise DIO pins |

|  |  |
| --- | --- |
| Syntax | void DIO\_WriteChannel(DIO\_PinType Pin\_ID, DIO\_PinLevel Value) |
| Sync/Async | Synchronous |
| Reentrancy | Non-Reentrant |
| Parameters | Pin\_ID, Value |
| Return | None |
| Description | Write Value on selected pin. |

|  |  |
| --- | --- |
| Syntax | DIO\_PinLevel DIO\_ReadChannel(DIO\_PinType Pin\_ID) |
| Sync/Async | Synchronous |
| Reentrancy | Non-Reentrant |
| Parameters | Pin\_ID |
| Return | Value |
| Description | Read Value from the selected pin. |

|  |  |
| --- | --- |
| Module | APIs |
| timer | Timer\_Init  Timer\_Start  Timer\_Stop |

|  |  |
| --- | --- |
| Syntax | ERROR\_STATE Timer\_Init (void) |
| Sync/Async | Synchronous |
| Reentrancy | Non-Reentrant |
| Parameters | None |
| Return | ERROR\_STATE |
| Description | Initialise internal timer |

|  |  |
| --- | --- |
| Syntax | void Timer\_Start (TimerTickType ticks) |
| Sync/Async | Synchronous |
| Reentrancy | Non-Reentrant |
| Parameters | Ticks |
| Return | None |
| Description | Set desired ticks in the timer |

|  |  |
| --- | --- |
| Syntax | void Timer\_Stop (void) |
| Sync/Async | Synchronous |
| Reentrancy | Non-Reentrant |
| Parameters | None |
| Return | None |
| Description | Stop timer |

|  |  |
| --- | --- |
| Module | APIs |
| CAN | CAN\_Init  CAN\_Receive |

|  |  |
| --- | --- |
| Syntax | ERROR\_STATE CAN\_Init (void) |
| Sync/Async | Synchronous |
| Reentrancy | Non-Reentrant |
| Parameters | None |
| Return | ERROR\_STATE |
| Description | Initialise CAN communication |

|  |  |
| --- | --- |
| Syntax | uint64 CAN\_Transmit(DIO\_PinType Pin\_ID) |
| Sync/Async | Synchronous |
| Reentrancy | Non-Reentrant |
| Parameters | Pin\_ID, Data |
| Return | Data |
| Description | Receive data via CAN |

|  |  |
| --- | --- |
| Module | APIs |
| Periodic Transmitter | periodicTransmitter\_Init  PeriodicTransmitter\_Receive |

|  |  |
| --- | --- |
| Syntax | ERROR\_STATE periodicTransmitter\_Init (void) |
| Sync/Async | Synchronous |
| Reentrancy | Non-Reentrant |
| Parameters | None |
| Return | ERROR\_STATE |
| Description | Initialise periodic transmission |

|  |  |
| --- | --- |
|  |  |
| Syntax | void periodicTransmitter\_Receive(DIO\_PinType Pin\_ID, uint64\* Data) |
| Sync/Async | Synchronous |
| Reentrancy | Non-Reentrant |
| Parameters | Pin\_ID, Data pointer |
| Return | None |
| Description | Receive periodic status to ECU 2 via CAN |

1. Typedef.

|  |  |
| --- | --- |
| typedef unsigned char uint8 |  |
| typedef unsigned long long uint64 | CAN Frame |
| typedef uint8 DIO\_Pin\_Type | OUTPUT = 1, INPUT = 0 |
| typedef uint8 DIO\_Pin\_Level | HIGH = 1, LOW = 0 |
| typedef uint8 CAN\_Pin\_Type | CANH = 1, CANL = 0 |