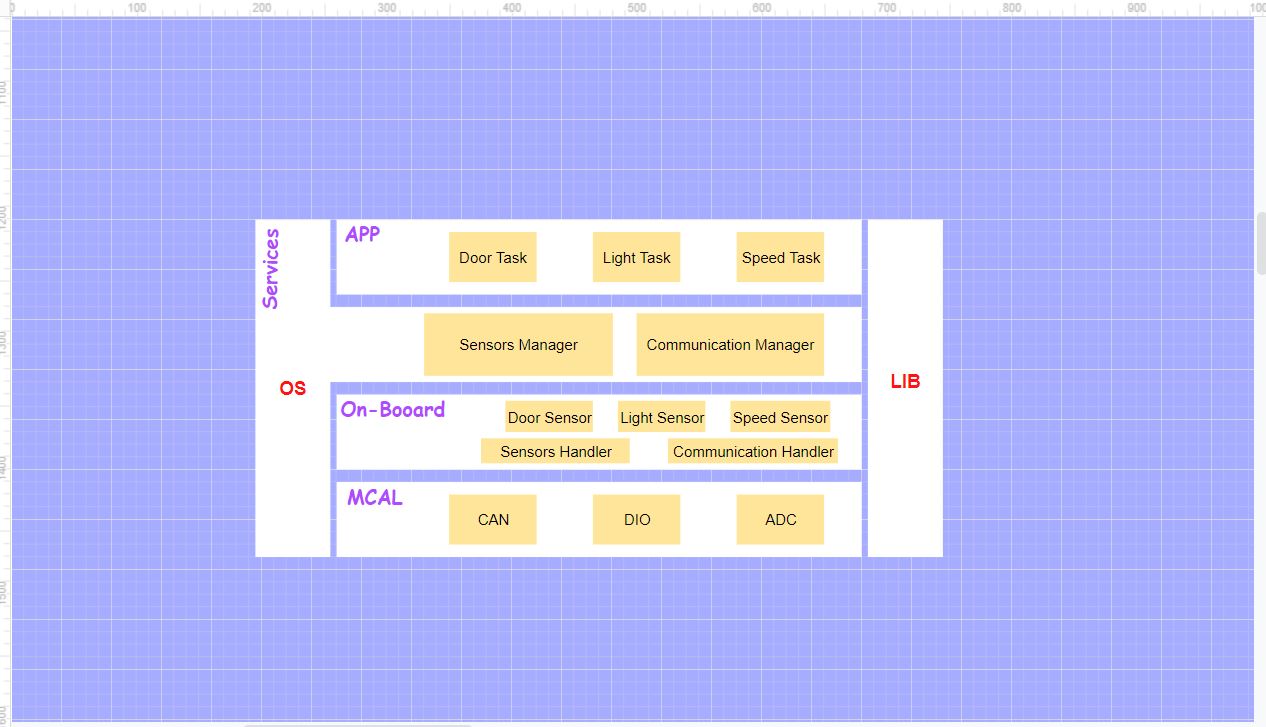
**Static Design**

# ECU\_1:

## Layered Architecture:



## Modules:

### CAN

**APIs:**

|  |  |
| --- | --- |
| Name | CAN\_Init() |
| Syntax | **void CAN\_Init (Can\_ConfigType \*config)** |
| Synchronization | Synchronous |
| Reentrancy | Non Reentrant |
| Parameters (in) | Config |
| Parameters (inout) | None |
| Parameters (out) | None |
| Return | None |
| Description | Initialize CAN module |

|  |  |
| --- | --- |
| Name | CAN\_DeInit() |
| Syntax | **void CAN\_DeInit (void)** |
| Synchronization | Synchronous |
| Reentrancy | Non Reentrant |
| Parameters (in) | None |
| Parameters (inout) | None |
| Parameters (out) | None |
| Return | None |
| Description | De-initialize CAN module |

|  |  |
| --- | --- |
| Name | CAN\_Send() |
| Syntax | **void Can\_Send (u32 message)** |
| Synchronization | Synchronous |
| Reentrancy | Reentrant |
| Parameters (in) | message |
| Parameters (inout) | None |
| Parameters (out) | None |
| Return | None |
| Description | Sends Data through CAN module |

**Typdefs:**

|  |  |
| --- | --- |
| Name | CAN\_ConfigType |
| Type | Struct |
| Description | A struct that holds all of the peripheral initialization |

### DIO

**APIs:**

|  |  |
| --- | --- |
| Name | Dio\_ReadChannel () |
| Syntax | **Dio\_LevelType Dio\_ReadChannel(Dio\_ChannelType channelId)** |
| Synchronization | Synchronous |
| Reentrancy | Reentrant |
| Parameters (in) | channelId |
| Parameters (inout) | None |
| Parameters (out) | None |
| Return | Dio\_LevelType |
| Description | Read a specific pin state |

|  |  |
| --- | --- |
| Name | Dio\_WriteChannel () |
| Syntax | **void Dio\_WriteChannel(Dio\_ChannelType channelId, Dio\_LevelType level)** |
| Synchronization | Synchronous |
| Reentrancy | Reentrant |
| Parameters (in) | channelId, level |
| Parameters (inout) | None |
| Parameters (out) | None |
| Return | Dio\_LevelType |
| Description | Write on a specific pin |

**Typdefs:**

|  |  |
| --- | --- |
| Name | Dio\_ChannelType |
| Type | u8 |
| Description | identifies which pin on the MCU we need to write to or read from |

|  |  |
| --- | --- |
| Name | Dio\_LevelType |
| Type | u8 |
| Description | identifies The state of the pin either high or low |

### ADC

**APIs:**

|  |  |
| --- | --- |
| Name | ADC\_Init () |
| Syntax | **void ADC\_Init(void)** |
| Synchronization | Synchronous |
| Reentrancy | Reentrant |
| Parameters (in) | None |
| Parameters (inout) | None |
| Parameters (out) | None |
| Return | None |
| Description | Initialize ADC Module |

|  |  |
| --- | --- |
| Name | ADC\_DeInit () |
| Syntax | **void ADC\_DeInit(void)** |
| Synchronization | Synchronous |
| Reentrancy | Reentrant |
| Parameters (in) | None |
| Parameters (inout) | None |
| Parameters (out) | None |
| Return | None |
| Description | De-Initialize ADC Module |

|  |  |
| --- | --- |
| Name | ADC\_Read () |
| Syntax | **u16 Adc\_ReadData(u8 channel)** |
| Synchronization | Synchronous |
| Reentrancy | Non-Reentrant |
| Parameters (in) | channel |
| Parameters (inout) | None |
| Parameters (out) | None |
| Return | u16 |
| Description | Get the current reading from the ADC |

### Speed Sensor

**APIs:**

|  |  |
| --- | --- |
| Name | SpeedSensor\_GetVal () |
| Syntax | **u16 SpeedSensor\_GetVal (void)** |
| Synchronization | Synchronous |
| Reentrancy | Reentrant |
| Parameters (in) | None |
| Parameters (inout) | None |
| Parameters (out) | None |
| Return | u16 |
| Description | Gets the current speed value |

|  |  |
| --- | --- |
| Name | DoorSensor\_GetState() |
| Syntax | **DoorSens\_StateType DoorSensor\_GetState (void)** |
| Synchronization | Synchronous |
| Reentrancy | Reentrant |
| Parameters (in) | None |
| Parameters (inout) | None |
| Parameters (out) | None |
| Return | DoorSens\_StateType |
| Description | Gets the current state of the door |

**Typdefs:**

|  |  |
| --- | --- |
| Name | DoorSens\_StateType |
| Type | enum |
| Description | identifies the state of the door either OPEN or CLOSED |

### Light Sensor

**APIs:**

|  |  |
| --- | --- |
| Name | LightSensor\_GetState() |
| Syntax | **LightSens\_StateType LightSensor\_GetState (void)** |
| Synchronization | Synchronous |
| Reentrancy | Reentrant |
| Parameters (in) | None |
| Parameters (inout) | None |
| Parameters (out) | None |
| Return | LightSens\_StateType |
| Description | Gets the current state of the light switch |

**Typdefs:**

|  |  |
| --- | --- |
| Name | LightSens\_StateType |
| Type | enum |
| Description | identifies the state of the light either ON or OFF |

### Sensor Handler

**APIs:**

|  |  |
| --- | --- |
| Name | Sensor\_Handler\_Select |
| Syntax | **u32 Sensor\_Handler\_Select (u8 Id)** |
| Synchronization | Synchronous |
| Reentrancy | Non-Reentrant |
| Parameters (in) | Id |
| Parameters (inout) | None |
| Parameters (out) | None |
| Return | u32 |
| Description | Chooses which sensor to read the data from |

### Communication Handler

**APIs:**

|  |  |
| --- | --- |
| Name | Communication\_Handler\_Send |
| Syntax | **void Communication\_Handler\_Send (u32 msg, u8 Id)** |
| Synchronization | Synchronous |
| Reentrancy | Non-Reentrant |
| Parameters (in) | msg, Id |
| Parameters (inout) | None |
| Parameters (out) | None |
| Return | u32 |
| Description | Chooses which bus or peripheral through which it sends the data |

### Sensors Manager

**APIs:**

|  |  |
| --- | --- |
| Name | Sensor\_Manager\_Select |
| Syntax | **u32 Sensor\_Manager\_Select (u8 Id)** |
| Synchronization | Synchronous |
| Reentrancy | Non-Reentrant |
| Parameters (in) | Id |
| Parameters (inout) | None |
| Parameters (out) | None |
| Return | u32 |
| Description | An interface for the application layer to Chooses which sensor to read the data from |

### Communication Manager

**APIs:**

|  |  |
| --- | --- |
| Name | Communication\_Manager\_Send |
| Syntax | **void Communication\_Manager\_Send (u8 Id, u32 msg)** |
| Synchronization | Synchronous |
| Reentrancy | Non-Reentrant |
| Parameters (in) | Id, msg |
| Parameters (inout) | None |
| Parameters (out) | None |
| Return | None |
| Description | An interface for the application layer to Chooses which bus or peripheral through which it sends the data |

### Light Task

**APIs:**

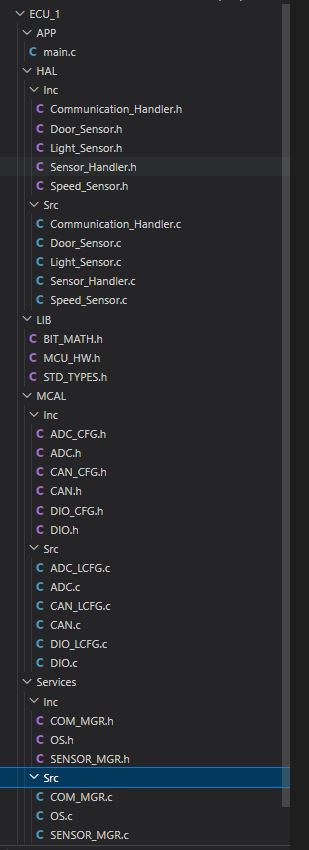
|  |  |
| --- | --- |
| Name | task\_SendLightSwitchState |
| Syntax | **void task\_SendLightSwitchState(void)** |
| Synchronization | Synchronous |
| Reentrancy | Non-Reentrant |
| Parameters (in) | None |
| Parameters (inout) | None |
| Parameters (out) | None |
| Return | None |
| Description | A task that sends the light switch state through the can protocol to another ECU every 20ms |

### Speed Task

**APIs:**

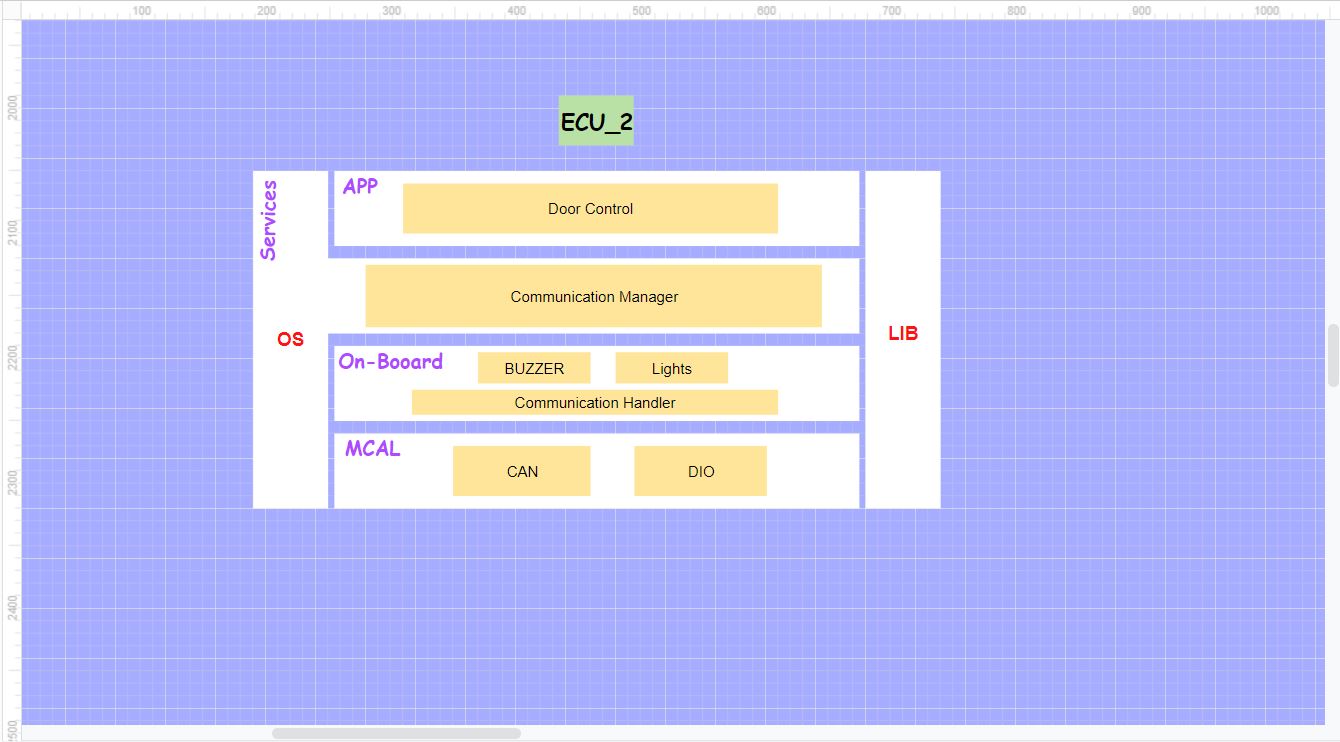
|  |  |
| --- | --- |
| Name | task\_SendSpeedState |
| Syntax | **void task\_SendSpeedState(void)** |
| Synchronization | Synchronous |
| Reentrancy | Non-Reentrant |
| Parameters (in) | None |
| Parameters (inout) | None |
| Parameters (out) | None |
| Return | None |
| Description | A task that sends the speed state through the can protocol to another ECU every 5ms |

## Folder Structure:



# ECU\_2:

## Layered Architecture:



## Modules:

### CAN

**APIs:**

|  |  |
| --- | --- |
| Name | CAN\_Init() |
| Syntax | **void CAN\_Init (Can\_ConfigType \*config)** |
| Synchronization | Synchronous |
| Reentrancy | Non Reentrant |
| Parameters (in) | Config |
| Parameters (inout) | None |
| Parameters (out) | None |
| Return | None |
| Description | Initialize CAN module |

|  |  |
| --- | --- |
| Name | CAN\_DeInit() |
| Syntax | **void CAN\_DeInit (void)** |
| Synchronization | Synchronous |
| Reentrancy | Non Reentrant |
| Parameters (in) | None |
| Parameters (inout) | None |
| Parameters (out) | None |
| Return | None |
| Description | De-initialize CAN module |

|  |  |
| --- | --- |
| Name | CAN\_Send() |
| Syntax | **void Can\_Send (u32 message)** |
| Synchronization | Synchronous |
| Reentrancy | Reentrant |
| Parameters (in) | message |
| Parameters (inout) | None |
| Parameters (out) | None |
| Return | None |
| Description | Sends Data through CAN module |

**Typdefs:**

|  |  |
| --- | --- |
| Name | CAN\_ConfigType |
| Type | Struct |
| Description | A struct that holds all of the peripheral initialization |

### DIO

**APIs:**

|  |  |
| --- | --- |
| Name | Dio\_ReadChannel () |
| Syntax | **Dio\_LevelType Dio\_ReadChannel(Dio\_ChannelType channelId)** |
| Synchronization | Synchronous |
| Reentrancy | Reentrant |
| Parameters (in) | channelId |
| Parameters (inout) | None |
| Parameters (out) | None |
| Return | Dio\_LevelType |
| Description | Read a specific pin state |

|  |  |
| --- | --- |
| Name | Dio\_WriteChannel () |
| Syntax | **void Dio\_WriteChannel(Dio\_ChannelType channelId, Dio\_LevelType level)** |
| Synchronization | Synchronous |
| Reentrancy | Reentrant |
| Parameters (in) | channelId, level |
| Parameters (inout) | None |
| Parameters (out) | None |
| Return | Dio\_LevelType |
| Description | Write on a specific pin |

**Typdefs:**

|  |  |
| --- | --- |
| Name | Dio\_ChannelType |
| Type | u8 |
| Description | identifies which pin on the MCU we need to write to or read from |

|  |  |
| --- | --- |
| Name | Dio\_LevelType |
| Type | u8 |
| Description | identifies The state of the pin either high or low |

### Lights

**APIs:**

|  |  |
| --- | --- |
| Name | Lights\_SetState |
| Syntax | **void lights\_SetState(u8 Id, u8 level)** |
| Synchronization | Synchronous |
| Reentrancy | Reentrant |
| Parameters (in) | Id, level |
| Parameters (inout) | None |
| Parameters (out) | None |
| Return | None |
| Description | It sets the light determined by the provided ID to either on or off |

### Buzzer

**APIs:**

|  |  |
| --- | --- |
| Name | Buzzer\_SetState |
| Syntax | **void Buzzer\_SetState(u8 level)** |
| Synchronization | Synchronous |
| Reentrancy | Reentrant |
| Parameters (in) | level |
| Parameters (inout) | None |
| Parameters (out) | None |
| Return | None |
| Description | It sets the buzzer to either ON or OFF |

### Communication Handler

**APIs:**

|  |  |
| --- | --- |
| Name | Communication\_Handler\_Send |
| Syntax | **void Communication\_Handler\_Send (u32 msg, u8 Id)** |
| Synchronization | Synchronous |
| Reentrancy | Non-Reentrant |
| Parameters (in) | msg, Id |
| Parameters (inout) | None |
| Parameters (out) | None |
| Return | u32 |
| Description | Chooses which bus or peripheral through which it sends the data |

### Communication Manager

**APIs:**

|  |  |
| --- | --- |
| Name | Communication\_Manager\_Send |
| Syntax | **void Communication\_Manager\_Send (u8 Id, u32 msg)** |
| Synchronization | Synchronous |
| Reentrancy | Non-Reentrant |
| Parameters (in) | Id, msg |
| Parameters (inout) | None |
| Parameters (out) | None |
| Return | None |
| Description | An interface for the application layer to Chooses which bus or peripheral through which it sends the data |

### Door Control

**APIs:**

|  |  |
| --- | --- |
| Name | DoorCtrl\_UpdateState |
| Syntax | **void DoorCtrl\_UpdateState (void)** |
| Synchronization | Synchronous |
| Reentrancy | Non-Reentrant |
| Parameters (in) | None |
| Parameters (inout) | None |
| Parameters (out) | None |
| Return | None |
| Description | A function That updates the door control module state with the newly received state from ECU1 and acts on the lights and the buzzer accordingly |

## Folder Structure:

