

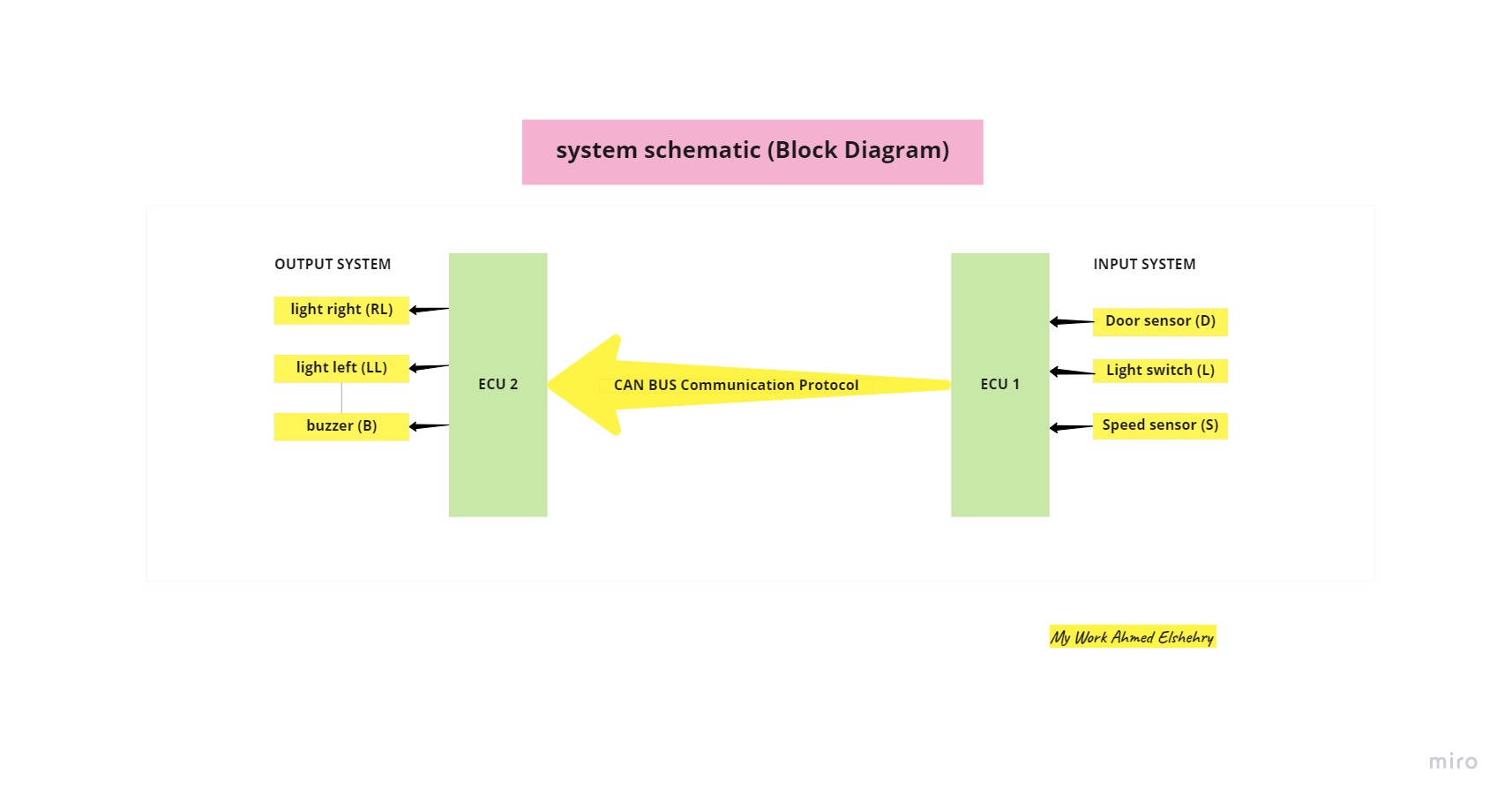
Automotive door control system design

Static design  Report

**Name: Ahmed Mohamed Hussein Elshehry**

**Email :** [elshehry97@gmail.com](mailto:elshehry97@gmail.com)

system schematic (Block Diagram) according to your requirements understanding.



Static Design:

* **For ECU 1:**

# C:\Users\Ahmed Elshehry\Downloads\layered Architecture ECU 1 (1).jpg**the layered architecture:**

# **Specify ECU components and modules**

# **Components connected:**

1. CAN BUS Communication Protocol (for communication between the two ECUs)
2. Light switch
3. Speed Sensor
4. Door Sensor

# **Modules:**

**External hardware:**

1. CAN transiver module
2. Switch module
3. Speed Sensor module
4. Door Sensor module

**Internal hardware:**

1. Port Module (initialize all pins required with modes)
2. DIO Module (switch module, Door Sensor module)
3. TIMER module (timer for application)
4. ADC module (for speed sensor)
5. CAN Module (for can transiver data )

# **Provide full detailed APIs for each module as well as a detailed description**

|  |  |  |  |
| --- | --- | --- | --- |
| Layer | Module | APIs | API Details |
| Application Layer | Main Application | DoorSensorTask | |  |  | | --- | --- | | Syntax: | void DoorSensorTask(void); | | Sync/Async: | Synchronous | | Reentrancy: | Non-Reentrant | | Parameters: | None | | Return: | None | | Description: | Manage Door Sensor Task | |

|  |  |  |  |
| --- | --- | --- | --- |
| Layer | Module | APIs | API Details |
| Application Layer | Main Application | LightSwitchTask  SpeedSensorTask | |  |  | | --- | --- | | Syntax: | void LightSwitchTask(void); | | Sync/Async: | Synchronous | | Reentrancy: | Non-Reentrant | | Parameters: | None | | Return: | None | | Description: | Manage Light Switch Task |  |  |  | | --- | --- | | Syntax: | void SpeedSensorTask(void); | | Sync/Async: | Synchronous | | Reentrancy: | Non-Reentrant | | Parameters: | None | | Return: | None | | Description: | Manage Speed Sensor Task | |
| Servies Layer | Basic Communication ModuleBasic  (BCM Manager) | BCM\_Manager | |  |  | | --- | --- | | Syntax: | void BCM\_Manager (uint8\_t Id\_Bus, uint64\_t Data ); | | Sync/Async: | Synchronous | | Reentrancy: | Non-Reentrant | | Parameters: | Data transmitter , Id Bus selection | | Return: | None | | Description: | Manage request the data Transmitter by CAN Bus W.R.T Id Bus selection | |
| Servies Layer | comm. Manager | Sensor\_Manager  (do Monitoring Sensors) | |  |  | | --- | --- | | Syntax: | uint8\_t Sensor\_Manager (uint8\_t Id\_Sensor); | | Sync/Async: | Synchronous | | Reentrancy: | Non-Reentrant | | Parameters: | Sensor selection want read states | | Return: | Date Read from sensor | | Description: | Manage request read states of data from sensor selection | |
| On Board Layer | Comm. Handler | BCM\_Handler | |  |  | | --- | --- | | Syntax: | void BCM\_Handler (uint8\_t Id\_Bus, uint64\_t Data ); | | Sync/Async: | Synchronous | | Reentrancy: | Non-Reentrant | | Parameters: | Data transmitter , Id Bus selection | | Return: | None | | Description: | Handler request the data Transmitter by CAN BUS but deals with Hardware directly | |
|  |  | Sensor\_Handler | |  |  | | --- | --- | | Syntax: | void Sensor\_Handler (uint8\_t Id\_ Sensor); | | Sync/Async: | Synchronous | | Reentrancy: | Non-Reentrant | | Parameters: | Sensor selection want read states | | Return: | None | | Description: | Handler request read states of data from sensor selection but deals with Hardware directly | |
| On Board Layer | Door Sensor | DoorSensor\_Init  DoorSensor\_ReadStatus | |  |  | | --- | --- | | Syntax: | void DoorSensor\_Init (void); | | Sync/Async: | Synchronous | | Reentrancy: | Non-Reentrant | | Parameters: | None | | Return: | None | | Description: | |  | | --- | | Initialize the used DIO pins for digital input | |  |  |  | | --- | --- | | Syntax: | uint8\_t DoorSensor\_ReadStatus (void); | | Sync/Async: | Synchronous | | Reentrancy: | Non-Reentrant | | Parameters: | None | | Return: | Status of the sensor door | | Description: | |  | | --- | | Get the status of the sensor door (closed or not ) | | |
| On Board Layer | Light Switch | LightSwitch\_Init  LightSwitch\_ReadStatus | |  |  | | --- | --- | | Syntax: | Void LightSwitch\_Init (void); | | Sync/Async: | Synchronous | | Reentrancy: | Non-Reentrant | | Parameters: | None | | Return: | None | | Description: | |  | | --- | | Initialize the used DIO pins for digital input | |  |  |  | | --- | --- | | Syntax: | uint8\_t LightSwitch\_ReadStatus (void); | | Sync/Async: | Synchronous | | Reentrancy: | Non-Reentrant | | Parameters: | None | | Return: | Status of the sensor door | | Description: | |  | | --- | | Get the status of the Light Switch (Pressed or unpressed ) | | |
| On Board Layer | Speed Sensor | SpeedSensor\_Init  SpeedSensor\_ReadStatus | |  |  | | --- | --- | | Syntax: | void SpeedSensor\_Init (void); | | Sync/Async: | Synchronous | | Reentrancy: | Non-Reentrant | | Parameters: | None | | Return: | None | | Description: | |  | | --- | | Initialize the used DIO pins for analog input For (ADC) | |  |  |  | | --- | --- | | Syntax: | uint8\_t SpeedSensor\_ReadStatus (void); | | Sync/Async: | Synchronous | | Reentrancy: | Non-Reentrant | | Parameters: | None | | Return: | Status of the sensor door | | Description: | |  |  | | --- | --- | | Read the value of the speed sensor (moving or stop)   |  | | --- | |  | | | |
| MCAL Layer | DIO | DIO\_Init  DIO\_ReadChannel  DIO\_WriteChannel | |  |  | | --- | --- | | Syntax: | Void DIO\_Init (void); | | Sync/Async: | Synchronous | | Reentrancy: | Non-Reentrant | | Parameters: | None | | Return: | None | | Description: | |  | | --- | | Initialize the used DIO pins with required configuration | |  |  |  | | --- | --- | | Syntax: | uint8\_t DIO\_ReadChannel(uint8\_t Id\_channel); | | Sync/Async: | Synchronous | | Reentrancy: | Non-Reentrant | | Parameters: | Id channel want read | | Return: | Status of pin High or low | | Description: | |  | | --- | | Read the channel required | |  |  |  | | --- | --- | | Syntax: | void DIO\_WriteChannel (uint8\_t Level ); | | Sync/Async: | Synchronous | | Reentrancy: | Non-Reentrant | | Parameters: | Level want to write channel | | Return: | None | | Description: | |  | | --- | | Write the level of the channel required | | |
| MCAL Layer | PORT | PORT\_init | |  |  | | --- | --- | | Syntax: | void RORT\_Init (void); | | Sync/Async: | Synchronous | | Reentrancy: | Non-Reentrant | | Parameters: | None | | Return: | None | | Description: | |  | | --- | | Initialize the used Port with required configuration | | |
| MCAL Layer | Timer | Timer\_Init  Timer\_Start  Timer\_Stop | |  |  | | --- | --- | | Syntax: | void Timer\_Init (void); | | Sync/Async: | Synchronous | | Reentrancy: | Non-Reentrant | | Parameters: | None | | Return: | None | | Description: | Initialize timer required configuration |  |  |  | | --- | --- | | Syntax: | void Timer\_Start (uint8\_t channel\_Id,uint\_32 value count ); | | Sync/Async: | Synchronous | | Reentrancy: | Non-Reentrant | | Parameters: | channel\_Id / value count tick | | Return: | None | | Description: | Initialize timer required configuration |  |  |  | | --- | --- | | Syntax: | Void Timer\_Stop (uint8\_t channel\_Id); | | Sync/Async: | Synchronous | | Reentrancy: | Non-Reentrant | | Parameters: | Channel \_Id of timer | | Return: | None | | Description: | Initialize timer required configuration | |
| MCAL Layer | CAN | CAN\_Init  CAN\_Transmiter | |  |  | | --- | --- | | Syntax: | void CAN\_Init (void); | | Sync/Async: | Synchronous | | Reentrancy: | Non-Reentrant | | Parameters: | None | | Return: | None | | Description: | Initialize CAN bus required configuration and Hardware pin CAN |  |  |  | | --- | --- | | Syntax: | void CAN\_Transmiter (uint8\_t Pin\_Id,uint64\_t Data); | | Sync/Async: | Synchronous | | Reentrancy: | Non-Reentrant | | Parameters: | Data transmitter , Pin\_id | | Return: | None | | Description: | Transmitter data by CAN Bus | |
| MCAL Layer | ADC | ADC\_Init  ADC\_ReadChannel | |  |  | | --- | --- | | Syntax: | void ADC\_Init (void); | | Sync/Async: | **Synchronous** | | Reentrancy: | **Non-Reentrant** | | Parameters: | **None** | | Return: | **None** | | Description: | **Initialize ADC required configuration and Hardware pin ADC connect speed sensor** | |
|  |  |  | |  |  | | --- | --- | | Syntax: | uint16\_tADC\_ReadChannel(uint8\_tPin\_Id); | | Sync/Async: | Synchronous | | Reentrancy: | Non-Reentrant | | Parameters: | Pin\_Id of ADC | | Return: | The value of channel ADC | | Description: | Read the value of channel ADC | |
|  |  |  |  |

# folder structure according to the previous points:

|  |  |  |
| --- | --- | --- |
| Application folder | Servies folder | On Board Layer |
| main.c | Operting\_system.c | BCM\_Handler.c |
|  | BCM\_Manager.c | Sensor\_Handler.c |
|  | Sensor\_Manager.c | Door\_sensor.c |
|  |  | Light\_switch.c |
|  |  | Speed\_sensor.c |

|  |  |
| --- | --- |
| MCAL folder | Configure folder |
| dio.c | Timer\_config.c |
| port.c | Adc\_config.c |
| adc.c | Can\_config.c |
| Timer.c | Port\_config.c |
| can.c | Dio\_config.c |
|  | Door\_sensorconfig.c |
|  | Light\_switchconfig.c |
|  | Speed\_sensorconfig.c |

|  |
| --- |
| Commen folder (all the header (name.h)) |
| Mainapp.h / os.h / servies.h |
| BCS\_manager.h/Sonser\_manager.h |
| Light\_switch.h / speed\_sonser.h / Door\_sensor.h |
| Dio.h / port.h / timer.h /can.h/adc.h |
| dio\_config.h/port\_config.h / timer\_config.h /can\_config.h /adc\_config.h |
| Stdtypes.h /comman\_macro.h /Hw.h |

* **For ECU 2:**

# **the layered architecture:**

# C:\Users\Ahmed Elshehry\Downloads\layered Architecture ECU 2 (2).jpg

# **Specify ECU components and modules**

# **Components connected:**

1. CAN BUS Communication Protocol (for communication between the two ECUs)
2. Light right
3. Light left
4. Buzzer

# **Modules:**

**External hardware:**

1. CAN transiver module
2. Light left module
3. Light right module
4. Buzzer module

**Internal hardware:**

1. Port Module (initialize all pins required with modes)
2. DIO Module (switch module, Door Sensor module)
3. TIMER module (timer for application)
4. CAN Module (for can transiver data )

# **Provide full detailed APIs for each module as well as a detailed description**

|  |  |  |  |
| --- | --- | --- | --- |
| Layer | Module | APIs | API Details |
| Application Layer | Main Application | PeriodicReceive\_Status   |  |  | | --- | --- | | |  | | --- | |  | | | |  |  | | --- | --- | | Syntax: | Void PeriodicReceive\_Status(uint64\_t \* data ,uint8\_t\* id\_CAN); | | Sync/Async: | Synchronous | | Reentrancy: | Non-Reentrant | | Parameters: | Pointer to data act as buffer for data ,pointer of CAN bus id to id cheek it | | Return: | None | | Description: | Manage received data periodicity status of ECU1 | |

|  |  |  |  |
| --- | --- | --- | --- |
| Layer | Module | APIs | API Details |
| Servies Layer | Basic Communication ModuleBasic  (BCM Manager) | BCM\_Manager | |  |  | | --- | --- | | Syntax: | uint64\_t BCM\_Manager (uint8\_t Id\_Bus); | | Sync/Async: | Synchronous | | Reentrancy: | Non-Reentrant | | Parameters: | Id Bus selection want read data from bus | | Return: | Data received from ECU 1 By can bus | | Description: | Manage request the data received by CAN Bus W.R.T Id Bus selection | |
| Servies Layer | comm. Manager | Actuator\_Manager  (do Monitoring Action ) | |  |  | | --- | --- | | Syntax: | Void Actuator\_Manager (uint8\_t actuator\_id ,uint8\_t action ); | | Sync/Async: | Synchronous | | Reentrancy: | Non-Reentrant | | Parameters: | actuator\_id selection want to do action states , action to do(on ,off ) | | Return: | Nona | | Description: | Monitoring action request to do actuator selection | |
| On Board Layer | Comm. Handler | BCM\_Handler | |  |  | | --- | --- | | Syntax: | uint64\_t BCM\_Handler (uint8\_t Id\_Bus); | | Sync/Async: | Synchronous | | Reentrancy: | Non-Reentrant | | Parameters: | Id Bus selection want received data from CAN BUS | | Return: | Data received from can bus | | Description: | Handler request the data Received by CAN BUS but deals with Hardware directly | |
|  |  | Sensor\_Handler | |  |  | | --- | --- | | Syntax: | Void Actuator \_Handler (uint8\_t Id\_ actuator , uint8\_t action ); | | Sync/Async: | Synchronous | | Reentrancy: | Non-Reentrant | | Parameters: | actuator\_id selection want to do action states , action to do(on ,off ) | | Return: | None | | Description: | Handler request to do action actuartor selection but deals with Hardware directly | |
| On Board Layer | Door Sensor | Buzzer\_Init  Buzzer\_on  Buzzer\_off | |  |  | | --- | --- | | Syntax: | Void Buzzer\_Init (void); | | Sync/Async: | Synchronous | | Reentrancy: | Non-Reentrant | | Parameters: | None | | Return: | None | | Description: | |  | | --- | | Initialize the used DIO pins for digital output respect to configuration | |  |  |  | | --- | --- | | Syntax: | void Buzzer\_on(void); | | Sync/Async: | Synchronous | | Reentrancy: | Non-Reentrant | | Parameters: | None | | Return: | None | | Description: | |  | | --- | | Set Buzzer to on states | |  |  |  | | --- | --- | | Syntax: | void Buzzer\_off(void); | | Sync/Async: | Synchronous | | Reentrancy: | Non-Reentrant | | Parameters: | None | | Return: | None | | Description: | |  | | --- | | Set Buzzer to off states | | |
| On Board Layer | Light Switch | Light\_Init  Light\_off  Light\_on | |  |  | | --- | --- | | Syntax: | Void Light\_Init (void); | | Sync/Async: | Synchronous | | Reentrancy: | Non-Reentrant | | Parameters: | None | | Return: | None | | Description: | Initialize the used DIO pins for digital output base the configuration |  |  |  | | --- | --- | | Syntax: | void Light\_off(void); | | Sync/Async: | Synchronous | | Reentrancy: | Non-Reentrant | | Parameters: | None | | Return: | None | | Description: | |  | | --- | | Set Light to off states | |  |  |  | | --- | --- | | Syntax: | Void Light\_on(void); | | Sync/Async: | Synchronous | | Reentrancy: | Non-Reentrant | | Parameters: | None | | Return: | None | | Description: | |  | | --- | | Set light to on states | | |
| MCAL Layer | DIO | DIO\_Init  DIO\_ReadChannel  DIO\_WriteChannel | |  |  | | --- | --- | | Syntax: | void DIO\_Init (void); | | Sync/Async: | Synchronous | | Reentrancy: | Non-Reentrant | | Parameters: | None | | Return: | None | | Description: | |  | | --- | | Initialize the used DIO pins with required configuration | |  |  |  | | --- | --- | | Syntax: | uint8\_t DIO\_ReadChannel(uint8\_t id\_channel ); | | Sync/Async: | Synchronous | | Reentrancy: | Non-Reentrant | | Parameters: | id\_channel | | Return: | Status of pin High or low | | Description: | |  | | --- | | Read the channel required | |  |  |  | | --- | --- | | Syntax: | void DIO\_WriteChannel (uint8\_t Level ); | | Sync/Async: | Synchronous | | Reentrancy: | Non-Reentrant | | Parameters: | Level want to write channel | | Return: | None | | Description: | |  | | --- | | Write the level of the channel required | | |
| MCAL Layer | PORT | PORT\_init | |  |  | | --- | --- | | Syntax: | void RORT\_Init (void); | | Sync/Async: | Synchronous | | Reentrancy: | Non-Reentrant | | Parameters: | None | | Return: | None | | Description: | |  | | --- | | Initialize the used Port with required configuration | | |
| MCAL Layer | Timer | Timer\_Init  Timer\_Start  Timer\_Stop | |  |  | | --- | --- | | Syntax: | void Timer\_Init (void); | | Sync/Async: | Synchronous | | Reentrancy: | Non-Reentrant | | Parameters: | None | | Return: | None | | Description: | Initialize timer required configuration |  |  |  | | --- | --- | | Syntax: | Void Timer\_Start (uint8\_t channel\_Id,uint\_32 value count ); | | Sync/Async: | Synchronous | | Reentrancy: | Non-Reentrant | | Parameters: | channel\_Id / value count | | Return: | None | | Description: | Initialize timer required configuration |  |  |  | | --- | --- | | Syntax: | Void Timer\_Stop (uint8\_t channel\_Id); | | Sync/Async: | Synchronous | | Reentrancy: | Non-Reentrant | | Parameters: | Channel \_Id of timer | | Return: | None | | Description: | Initialize timer required configuration | |
| MCAL Layer | CAN | CAN\_Init  CAN\_ ReceivedData | |  |  | | --- | --- | | Syntax: | Void CAN\_Init (void); | | Sync/Async: | Synchronous | | Reentrancy: | Non-Reentrant | | Parameters: | None | | Return: | None | | Description: | Initialize CAN bus required configuration and Hardware pin CAN |  |  |  | | --- | --- | | Syntax: | Uint64\_t CAN\_ ReceivedData (uint8\_t Pin\_IdCAn); | | Sync/Async: | Synchronous | | Reentrancy: | Non-Reentrant | | Parameters: | Pin\_idcan | | Return: | Data Recivered from Can bus | | Description: | Received data from CAN Bus | |

# **folder structure according to the previous points:**

|  |  |  |
| --- | --- | --- |
| Application folder | Servies folder | On Board Layer |
| main.c | Operting\_system.c | BCM\_Handler.c |
|  | BCM\_Manager.c | Actuator\_Handler.c |
|  | Actuator\_Manager.c | Buzzer\_sensor.c |
|  |  | Light.c |

|  |  |
| --- | --- |
| MCAL folder | Configure folder |
| dio.c | Timer\_config.c |
| port.c | Can\_config.c |
| can.c | Dio\_config.c |
| Timer.c | Port\_config.c |
|  | Light\_config.c |
|  | Buzzer\_config.c |

|  |
| --- |
| Commen folder (all the header (name.h)) |
| Mainapp.h / os.h / servies.h |
| BCS\_manager.h/ Actuator\_manager.h |
| Light\_.h / light.h |
| Dio.h / port.h / timer.h /can.h |
| dio\_config.h/port\_config.h / timer\_config.h /can\_config.h |
| Stdtypes.h /comman\_macro.h /Hw.h |