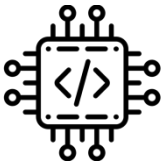
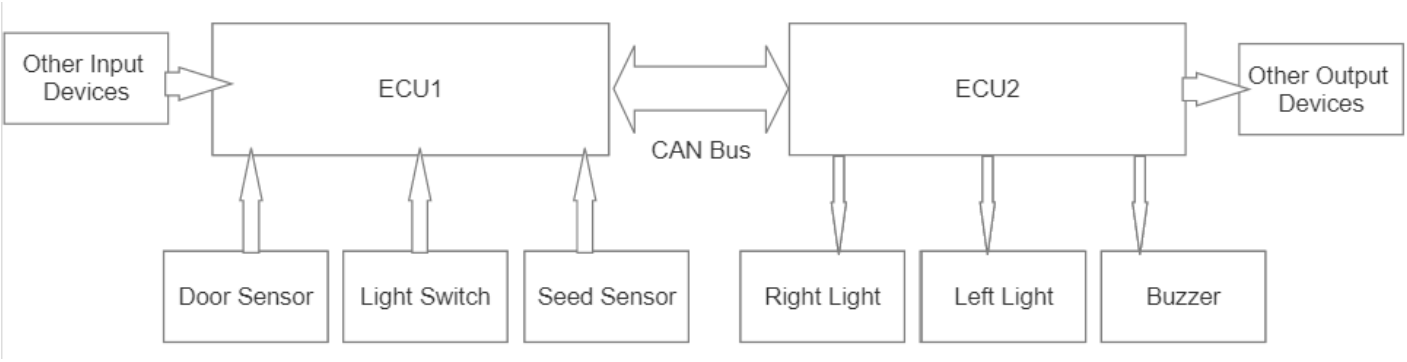


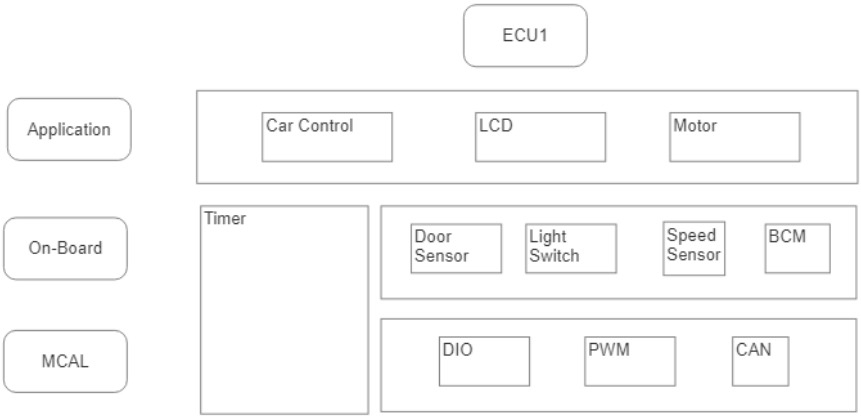
Static Design for Microcontroller

By: Ahmed Sayed

- **Schematic / Block Diagram For System**



- **Layered Architecture For ECU1**



- **Full Detailed APIs For ECU1**

Door Sensor

- **void Door_Sensor_Init(char Port_Name , int Pin_Number) ;** → **Init API**
  Initiates Door Sensor Module by choosing PORT and PIN in GPIO
  Range : all digital PINs in MCU

- **bool Door_Sensor_Get_State(char Port_Name , int Pin_Number) ;** → **Getter API**
  Get State of Door Sensor Module (High or Low)
  Range : all digital PINs in MCU

- **void Door_Sensor_Interval_For_Sending(int Interval) ;** → **Setter API**
  Set Period For Sending State through BCM For Door Sensor Module
  Range : Interval in milliseconds

- **void Door_Sensor_Send_State() ;** → **Periodic API**
  Send State through BCM For Door Sensor Module Every Period

Description : it calls Door_Sensor_Get_State API inside it and update state of it inside Basic_Communcation_Module_Collect API

- **void Door_Sensor_Deinit(char Port_Name , int Pin_Number) ; → Deinit API**
  Deinitiates Door Sensor Module by deleting chosen PORT and PIN in GPIO
  Range : all digital PINs in MCU

Light Switch

- **void Light_Switch_Init(char Port_Name , int Pin_Number) ; → Init API**
  Initiates Light Switch Module by choosing PORT and PIN in GPIO
  Range : all digital PINs in MCU

- **bool Light_Switch _Get_State(char Port_Name , int Pin_Number) ; → Getter API**
  Get State of Light Switch Module (High or Low)
  Range : all digital PINs in MCU

- **void Light_Switch _Interval_For_Sending(int Interval) ; → Setter API**
  Set Period For Sending State through BCM For Light Switch Module
  Range : Interval in milliseconds

- **void Light_Switch _Send_State() ; → Periodic API**
  Send State through BCM For Light Switch Module Every Period
  Description : it calls Light_Switch _Get_State API inside it and update state of it inside Basic_Communcation_Module_Collect API

- **void Light_Switch _Deinit(char Port_Name , int Pin_Number) ; → Deinit API**
  Deinitiates Light Switch Module by deleting chosen PORT and PIN in GPIO
  Range : all digital PINs in MCU

Speed Sensor

- **void Speed_Sensor_Init(char Port_Name , int Pin_Number) ; → Init API**
  Initiates Speed Sensor Module by choosing PORT and PIN in GPIO
  Range : all digital PINs in MCU

- **bool Speed_Sensor _Get_State(char Port_Name , int Pin_Number) ; → Getter API**
  Get State of Speed Sensor Module (High or Low)
  Range : all digital PINs in MCU

- **void Speed_Sensor _Interval_For_Sending(int interval) ; → Setter API**
  Set Period For Sending State through BCM For Speed Sensor Module
  Range : Interval in milliseconds

- **void Speed_Sensor _Send_State() ; → Periodic API**
  Send State through BCM For Speed Sensor Module Every Period
  Description : it calls Speed_Sensor_Get_State API inside it and update state of it inside Basic_Communcation_Module_Collect API

- **void Speed_Sensor _Deinit(char Port_Name , int Pin_Number) ; → Deinit API**
  Deinitiates Speed Sensor Module by deleting chosen PORT and PIN in GPIO
  Range : all digital PINs in MCU

BCM

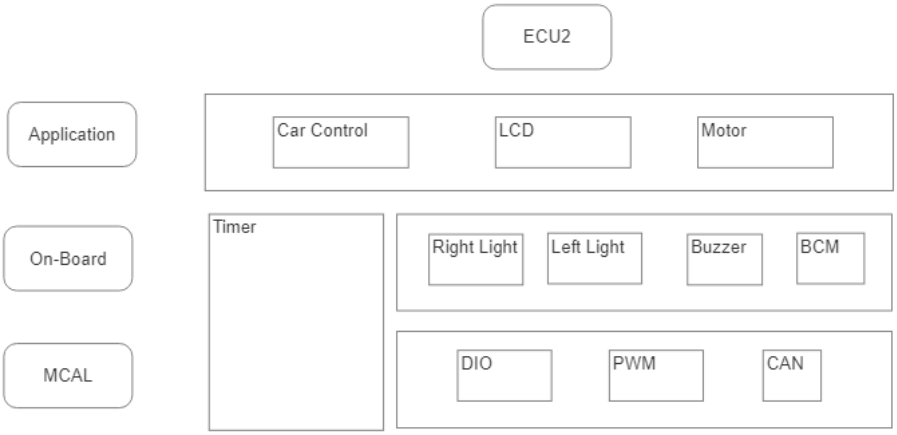
- **void Basic_Communcation_Module_Init() ; → Init API**
  Initates Communication Between ECU1 and ECU2

- **void Basic_Communcation_Module_Collect(bool Light_Switch,bool Speed_Sensor,bool Door_Sensor); → Periodic API**
  Collect States From Light_Switch _Send_State, Speed_Sensor _Send_State,Door_Sensor _Send_State APIs

- **void Basic_Communcation_Module_Send(); → Periodic API**
  Send States of Inputs To ECU2 by using Basic_Communcation_Module_Collect API

- **void Basic_Communcation_Module _Deinit() ; → Deinit API**
  Deinitates Communication Between ECU1 and ECU2

- **Layered Architecture For ECU2**



- **Full Detailed APIs For ECU2**

BCM

- **void Basic_Communcation_Module_Init() ;** → **Init API**
  Initates Communication Between ECU1 and ECU2

- **bool Basic_Communcation_Module_Recieve_Light_Switch();** → **Periodic API**
  Recieve State of Light Switch From ECU1

- **bool Basic_Communcation_Module_Recieve_Speed_Sensor();** → **Periodic API**
  Recieve State of Speed Sensor From ECU1

- **bool Basic_Communcation_Module_Recieve_Door_Sensor();** → **Periodic API**
  Recieve State of Door Sensor From ECU1

- **void Basic_Communcation_Module _Deinit() ;** → **Deinit API**
  Deinitates Communication Between ECU1 and ECU2

Right Light

- **void Right_Light_Init(char Port_Name , int Pin_Number) ;** → **Init API**
  Initiates Right Light Module by choosing PORT and PIN in GPIO
  Range : all digital PINs in MCU

- **void Right_Light _Up_State(char Port_Name , int Pin_Number) ; → Setter API**
  Set State of Right Light to High State
  Range : all digital PINs in MCU

- **void Right_Light _ Down_State(char Port_Name , int Pin_Number); → Setter API**
  Set State of Right Light to Low State
  Range : all digital PINs in MCU

- **void Right_Light _ Toggle_State(char Port_Name , int Pin_Number); → Setter API**
  Toggle State of Right Light State
  Range : all digital PINs in MCU

- **bool Right_Light _ Get_State(char Port_Name , int Pin_Number); → Getter API**
  Get State of Right Light
  Range : all digital PINs in MCU

- **void Right_Light _Deinit(char Port_Name , int Pin_Number) ; → Deinit API**
  Deinitiates Right Light Module by deleting chosen PORT and PIN in GPIO
  Range : all digital PINs in MCU

Left Light

- **void Left_Light_Init(char Port_Name , int Pin_Number) ; → Init API**
  Initiates Left Light Module by choosing PORT and PIN in GPIO
  Range : all digital PINs in MCU

- **void Left_Light _Up_State(char Port_Name , int Pin_Number) ; → Setter API**
  Set State of Left Light to High State
  Range : all digital PINs in MCU

- **void Left_Light _ Down_State(char Port_Name , int Pin_Number); → Setter API**
  Set State of Left Light to Low State
  Range : all digital PINs in MCU

- **void Left_Light _ Toggle_State(char Port_Name , int Pin_Number); → Setter API**
  Toggle State of Left Light State
  Range : all digital PINs in MCU

- **bool Left_Light _ Get_State(char Port_Name , int Pin_Number); → Getter API**
  Get State of Left Light
  Range : all digital PINs in MCU

- **void Left_Light _Deinit(char Port_Name , int Pin_Number) ; → Deinit API**
  Deinitiates Left Light Module by deleting chosen PORT and PIN in GPIO
  Range : all digital PINs in MCU

Buzzer

- **void Buzzer_Init(char Port_Name , int Pin_Number) ; → Init API**
  Initiates Buzzer Module by choosing PORT and PIN in GPIO
  Range : all digital PINs in MCU

- **void Buzzer _Up_State(char Port_Name , int Pin_Number) ; → Setter API**
  Set State of Buzzer to High State
  Range : all digital PINs in MCU

- **void Buzzer _ Down_State(char Port_Name , int Pin_Number); → Setter API**
  Set State of Buzzer to Low State
  Range : all digital PINs in MCU

- **void Buzzer _ Toggle_State(char Port_Name , int Pin_Number); → Setter API**
  Toggle State of Buzzer State
  Range : all digital PINs in MCU

- **bool Buzzer _ Get_State(char Port_Name , int Pin_Number); → Getter API**
  Get State of Buzzer
  Range : all digital PINs in MCU

- **void Buzzer _Deinit(char Port_Name , int Pin_Number) ; → Deinit API**
  Deinitiates Buzzer Module by deleting chosen PORT and PIN in GPIO
  Range : all digital PINs in MCU