Embedded Systems Advanced Nano Degree

Automotive door control system design

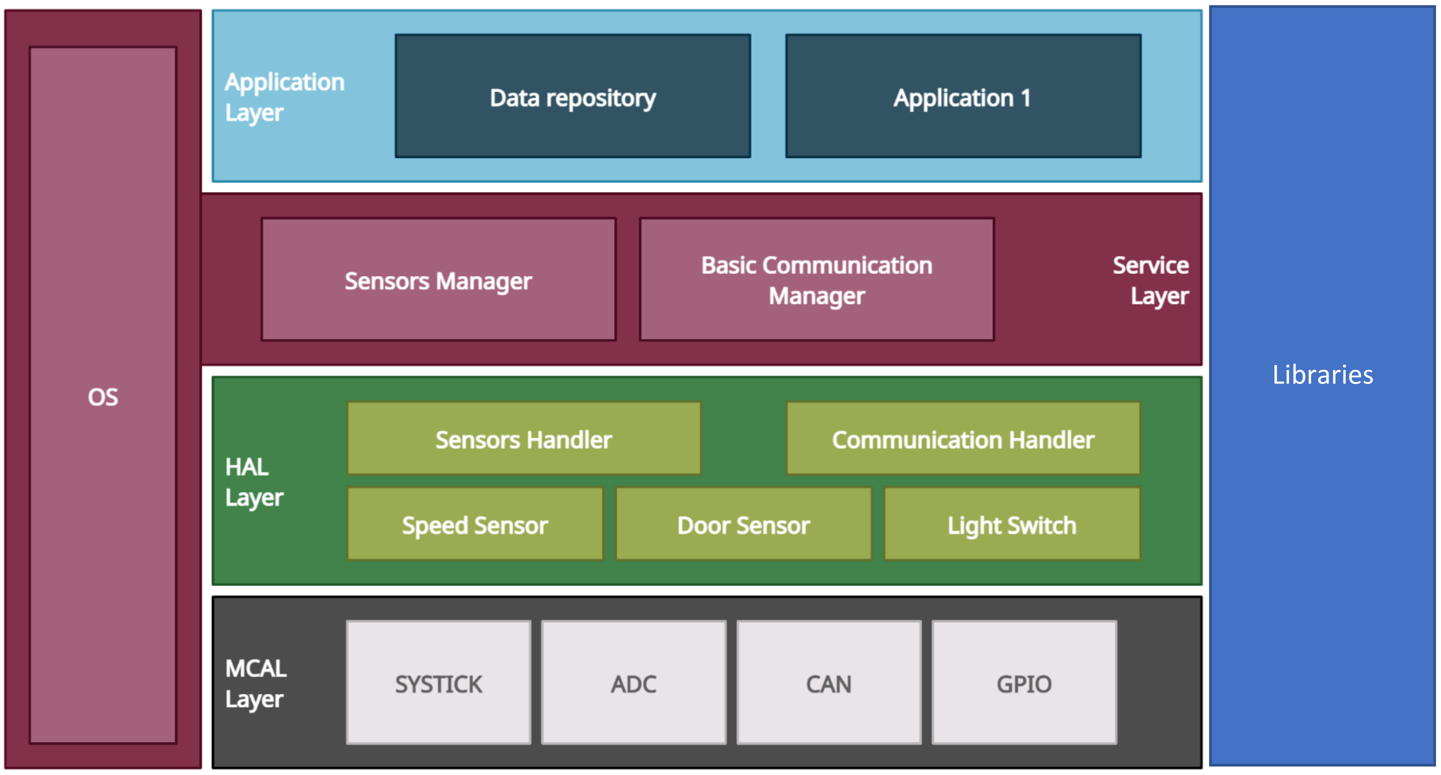
Static Design Analysis

Youssef Hussien Mahmoud

ysfhussien@gmail.com

ECU_1

1- Layered Architecutre



2- APIs

GPIO module:

API	void GPIO_init (struct * config_ptr);		
Description	Initialize the GPIO with the structure configrations		
Sync/Async	Synchronous	Reentrancy	Non-reentrant
Parameters	Pointer to sturcture	Return	void

API	void GPIO_write_Pin (uint32 Port_No, uint32 Pin_No, uint8 value);		
Description	Write the required GPIO port,Pin with the required value		
Sync/Async	Synchronous	Reentrancy	Non-reentrant
Parameters	Port number - Pin number - pin value	Return	None

API	Uint8 GPIO_read_Pin (uint32 Port_No, unint32 Pin_No);		
Description	Read the required Gpio port , pin.		
Sync/Async	Synchronous	Reentrancy	Non-reentrant
Parameters	Port number - Pin number	Return	Uint8

ADC Module:

API	void ADC_init (struct * Config_ptr);		
Description	Initialize the ADC with the structure configrations		
Sync/Async	Synchronous	Reentrancy	Non-reentrant
Parameters	Pointer to structure	Return	None

API	uint32 ADC_readChannel (uint8 Channel_id);		
Description	Read the required channel ID		
Sync/Async	Synchronous	Reentrancy	Non-reentrant
Parameters	channel ID	Return	uint32

CAN Module:

API	void CAN_init (struct * Config_ptr);		
Description	Initialize CAN bus with the structure configrations		
Sync/Async	Synchronous	Reentrancy	Non-reentrant
Parameters	Pointer to structure	Return	void

API	void CAN_transmit (uint8 CanPin_ID, uint64 Message);		
Description	Send a required message via required pin ID		
Sync/Async	Synchronous	Reentrancy	Non-reentrant
Parameters	Can Pin number - Message	Return	None

Speed Sensor Module:

API	void SpeedSensor_init (struct * Config_ptr);		
Description	Initialize the speed sensor pin via ADC		
Sync/Async	Synchronous	Reentrancy	Non-reentrant
Parameters	Pointer to stucture	Return	void

API	Uint16 SpeedSensor_getSpeed (void);		
Description	Get the speed from the speed sensor via ADC		
Sync/Async	Synchronous	Reentrancy	Non-reentrant
Parameters	None	Return	Uint16

Door Sensor Module:

API	void DoorSensor_init (struct * Config_ptr);		
Description	Initialize the door sensor pin via GPIO		
Sync/Async	Synchronous	Reentrancy	Non-reentrant
Parameters	Pointer to structure	Return	void

API	uint8 DoorSensor_getStatus (void);		
Description	Read the door sensor status via GPIO		
Sync/Async	Synchronous	Reentrancy	Non-reentrant
Parameters	None	Return	uint8

Light Switch Module:

API	void LightSwitch_init (struct * Config_ptr);		
Description	Initialize the light switch module with the structure configurations		
Sync/Async	Synchronous	Reentrancy	Non-reentrant
Parameters	Pointer to structure	Return	void

API	uint8 LightSwitch_getStatus (void);		
Description	Read the light swich status		
Sync/Async	Synchronous	Reentrancy	Non-reentrant
Parameters	None	Return	Uint8

Sensor handler Module:

API	uint32 Sensor_handler (uint8 Sensor_ID);		
Description	chooses which sensor to read from hardware layer		
Sync/Async	Synchronous	Reentrancy	Non-reentrant
Parameters	Sensor ID	Return	Uint32

Communication handler module:

API	void BCM_handler (uint64 handler_Message, uint8 bus);		
Description	Choose which bus to send the required message		
Sync/Async	Synchronous	Reentrancy	Non-reentrant
Parameters	Message - bus	Return	None

Sensor manager Module:

API	uint32 Sensor_manager (uint8 sensor_Id);		
Description	Allow the application layer to choose the required sensor		
Sync/Async	Synchronous	Reentrancy	Non-reentrant
Parameters	Sensor ID	Return	Uint32

Basic Communication manager Module:

API	Void BCM_mananger (uint64 Manager_Message, uint8 bus);		
Description	Allow the application layer to choose the required bus		
Sync/Async	Synchronous	Reentrancy	Non-reentrant
Parameters	Message - bus	Return	None

Data Repository Module:

API	void Data_repository (uint64 data);		
Description	Save the required data		
Sync/Async	Synchronous	Reentrancy	Non-reentrant
Parameters	Data to be saved	Return	None

Application1 Module :

API	void SendDoorState (void);		
Description	Send the door sensor state to ECU2 via CAN bus		
Sync/Async	Synchronous	Reentrancy	Non-reentrant
Parameters	None	Return	None

API	void SendSpeed (void);		
Description	Send the speed sensor state to ECU2 via CAN bus		
Sync/Async	Synchronous	Reentrancy	Non-reentrant
Parameters	None	Return	None

API	void SendLightSwitchState (void);		
Description	Send the light switch state to ECU2 via CAN bus		
Sync/Async	Synchronous	Reentrancy	Non-reentrant
Parameters	None	Return	None

3- Folder Structure:

MCAL Folder	HAL Folder	Service Folder
Systick.c	Sensor_Handler.c	OS.c
ADC.c	Comm_Handler.c	Basic_Comm_mngr.c
CAN.c	Light_Switch.c	Sensors_mngr.c
GPIO.c	Door_sensor.c	
	Speed_sensor.c	

Application Folder	Config Folder
Data_repo.c	Systick_PBConfig.c
App.c	ADC_PBConfig.c
	CAN_PBConfig.c
	GPIO_PBConfig.c
	Switch_PBConfig.c
	Door_PBConfig.c
	Speed_PBConfig.c

Common (Header Files) Folder:

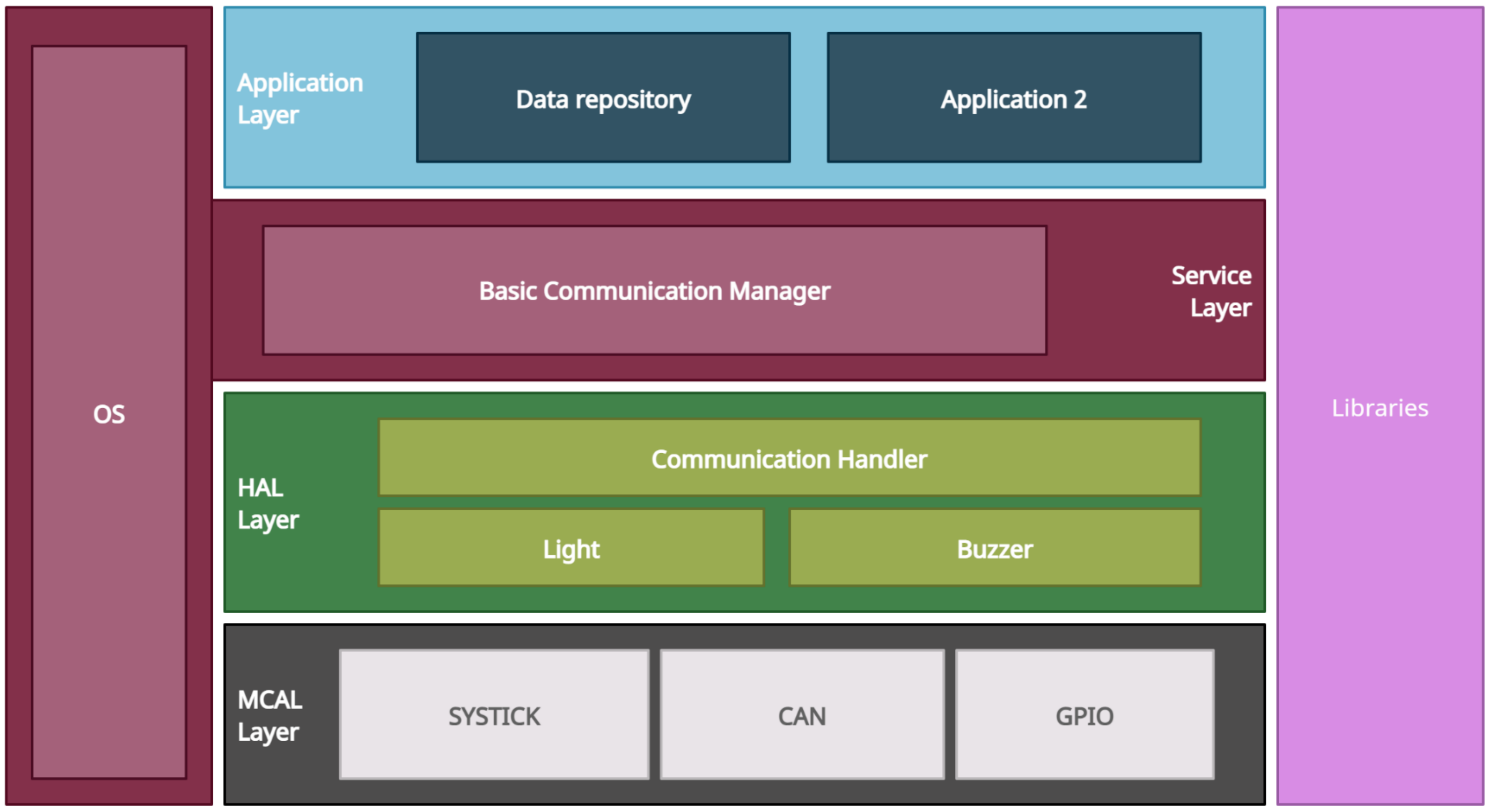
Systick.h	ADC.h	CAN.h	GPIO.h
Sensor_handler.h	Comm_handler.h	Switch.h	Door.h
Speed.h	OS.h	App.h	Data_repo.h
Systick_Config.h	ADC_Config.h	CAN_Config.h	GPIO_Config.h
Switch_Config.h	Door_Config.h	Speed_Config.h	Sensor_mngr.h
Common_Macros.h	Std_lib.h	MCU_Regs.h	Comm_mngr.h

4- **Drivers Structure:**
   1- **Systick Driver:**
      - **Systick.c**
      - **Systick.h**
      - **Systick_PBConfig.c**
      - **Systick_Config.h**
   2- **ADC Driver:**
      - **ADC.c**
      - **ADC.h**
      - **ADC_PBConfig.c**
      - **ADC_Config.h**
   3- **CAN Driver:**
      - **CAN.c**
      - **CAN.h**
      - **CAN_PBConfig.c**
      - **CAN_Config.h**
   4- **GPIO Driver:**
      - **GPIO.c**
      - **GPIO.h**
      - **GPIO_PBConfig.c**
      - **GPIO_Config.h**

And so on for other drivers…

ECU_2

1- Layered Architecture



2- APIs

GPIO module:

API	void GPIO_init (struct * config_ptr);		
Description	Initialize the GPIO with the structure configrations		
Sync/Async	Synchronous	Reentrancy	Non-reentrant
Parameters	Pointer to sturcture	Return	void

API	void GPIO_write_Pin (uint32 Port_No, uint32 Pin_No, uint8 value);		
Description	Write the required GPIO port,Pin with the required value		
Sync/Async	Synchronous	Reentrancy	Non-reentrant

Parameters	Port number - Pin number - pin value	Return	None

API	Uint8 GPIO_read_Pin (uint32 Port_No, unint32 Pin_No);		
Description	Read the required Gpio port , pin.		
Sync/Async	Synchronous	Reentrancy	Non-reentrant
Parameters	Port number - Pin number	Return	Uint8

CAN Module :

API	void CAN_init (struct * Config_ptr);		
Description	Initialize CAN bus with the structure configrations		
Sync/Async	Synchronous	Reentrancy	Non-reentrant
Parameters	Pointer to structure	Return	void

API	Uint64 CAN_Receive (uint8 CAN_Pin_Id);		
Description	Receive the CAN message from the required Pin ID		
Sync/Async	Synchronous	**Reentrancy**	Non-reentrant
Parameters	Can Pin number	**Return**	Uint64

Buzzer Module:

API	void BUZZER_init (struct * Config_ptr);		
Description	Initialize the buzzer with the structure configurations via GPIO		
Sync/Async	Synchronous	**Reentrancy**	Non-reentrant
Parameters	Pointer to structure	**Return**	void

API	void BUZZER_on (void);		
Description	Set the buzzer on		
Sync/Async	Synchronous	**Reentrancy**	Non-reentrant
Parameters	None	**Return**	None

API	`void BUZZER_off (void);`		
Description	Set the buzzer off		
Sync/Async	Synchronous	**Reentrancy**	Non-reentrant
Parameters	None	**Return**	None

Communication handler module:

API	`uint64 BCM_Handler (uint8 bus);`		
Description	Choose which bus to read the message from MCAL layer		
Sync/Async	Synchronous	**Reentrancy**	Non-reentrant
Parameters	the bus	**Return**	Uint64

Basic Communication manager Module:

API	`uint64 BCM_mananger (uint8 bus);`		
Description	Allow the application layer to choose which bus to read the message from		
Sync/Async	Synchronous	**Reentrancy**	Non-reentrant
Parameters	bus	**Return**	Uint64

Light Module:

API	`void Light_init (struct * Config_ptr);`		
Description	Initialize the light via GPIO driver		
Sync/Async	Synchronous	**Reentrancy**	Non-reentrant
Parameters	Pointer to structure	**Return**	void

API	`void light_ON (void);`		
Description	Set the light on		
Sync/Async	Synchronous	**Reentrancy**	Non-reentrant
Parameters	None	**Return**	None

API	void light_OFF (void);		
Description	Set the light off		
Sync/Async	Synchronous	**Reentrancy**	Non-reentrant
Parameters	None	**Return**	None

Data repository Module:

API	void Data_repository (uint64 data);		
Description	Save the required data		
Sync/Async	Synchronous	Sync/Async	Synchronous
Parameters	Data to be saved	Parameters	Data to be saved

Application2 Module:

API	void Receive_Message (void);		
Description	Receive the message from ECU1 periodically to take actions		
Sync/Async	Synchronous	**Reentrancy**	Non-reentrant
Parameters	None	**Return**	None

3- Folder Structure

MCAL Folder	HAL Folder	Service Folder
Systick.c	Light.c	OS.c
GPIO.c	Comm_Handler.c	Basic_Comm_mngr.c
CAN.c	Buzzer.c	

Application Folder	Config Folder
Data_repo.c	Systick_PBConfig.c
App2.c	Light_PBConfig.c
	CAN_PBConfig.c
	GPIO_PBConfig.c
	Buzzer_PBConfig.c

Common (Header files) Folder:

Systick.h	Light.h	CAN.h	GPIO.h
Buzzer.h	Comm_handler.h	OS.h	Comm_mngr.h
Data_repo.h	App2.h	Systick_Config.h	Light_Config.h
CAN_Config.h	GPIO_Config.h	Buzzer_Config.h	MCU_Regs.h
Common_Macros.h	Std_lib.h		

5- Drivers Structure:

1. **Systick Driver:**
   - **-Systick.c**
   - **-Systick.h**
   - **-Systick_PBConfig.c**
   - **-Systick_Config.h**

2. **CAN Driver:**
   - **-CAN.c**
   - **-CAN.h**
   - **-CAN_PBConfig.c**
   - **-CAN_Config.h**
3. **GPIO Driver:**
   - **-GPIO.c**
   - **-GPIO.h**
   - **-GPIO_PBConfig.c**
   - **-GPIO_Config.h**

And so on for other drivers…

Type definitions:

```
- typedef unsigned long uint32   - typedef unsigned short uint16
- typedef unsigned char uint8    - typedef unsigned long long uint64
```