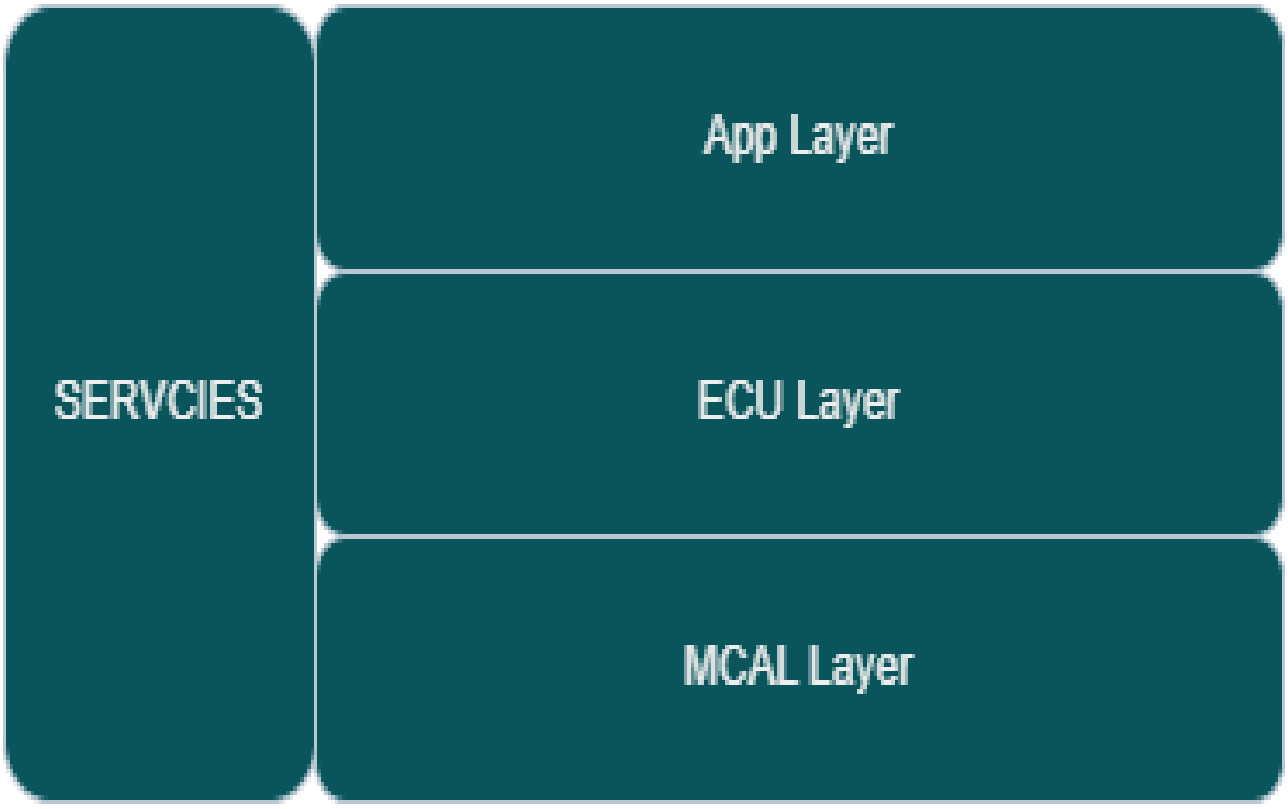
Moving Car System Design

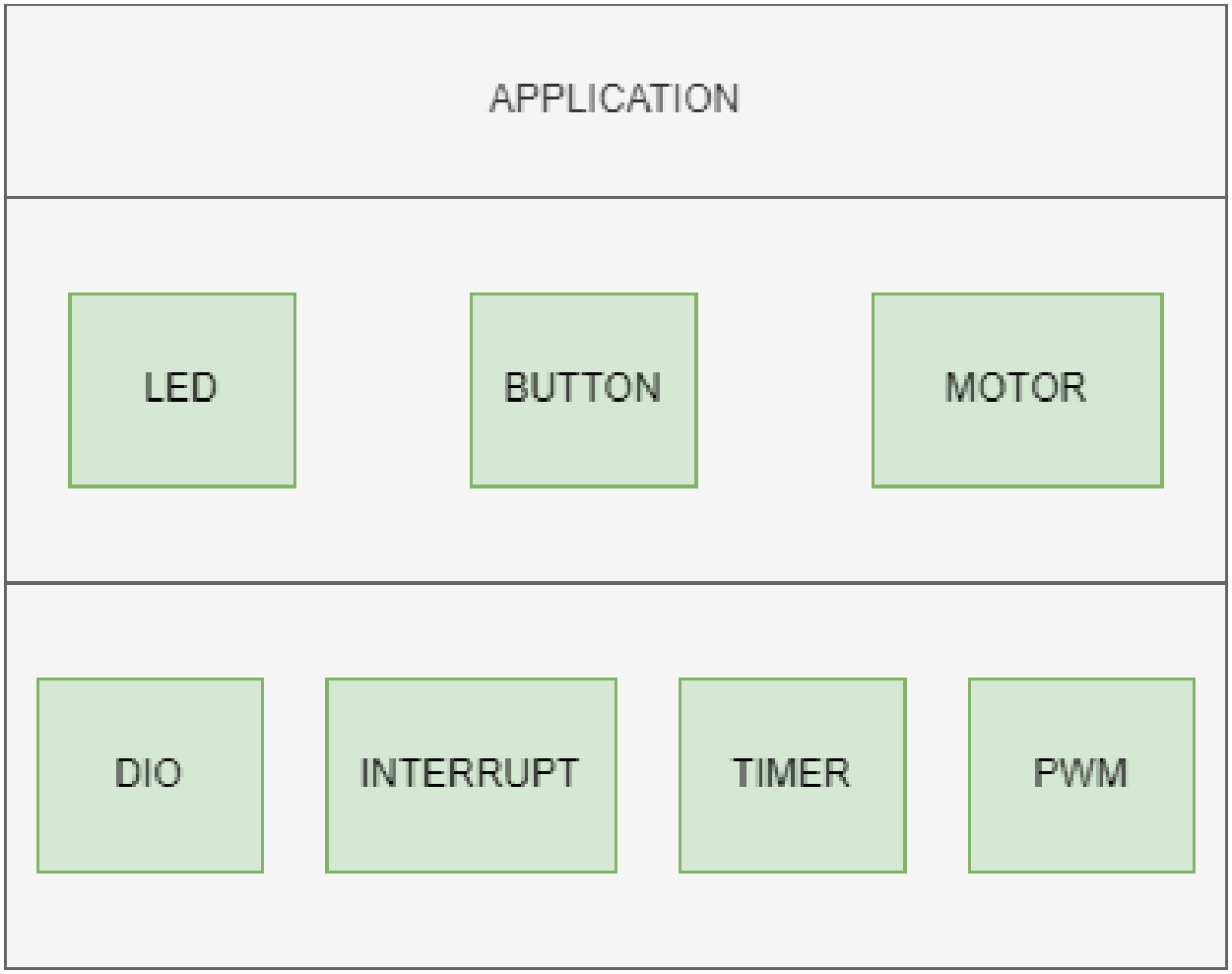
Date : 5/4/2023

Created By : Sherif Ashraf Ali

Layered architecture



System modules



Module Name APIs

1-MCAL

1.1 DIO

typedef enum{

GPIO_LOGIC_LOW = 0,

GPIO_LOGIC_HIGH

}logic_t;

typedef enum{

GPIO_DIRECTION_OUTPUT = 0,

```
GPIO_DIRECTION_INPUT
}direction_t;

typedef enum{
  GPIO_PIN0 = 0,
  GPIO_PIN1,
  GPIO_PIN2,
  GPIO_PIN3,
  GPIO_PIN4,
  GPIO_PIN5,
  GPIO_PIN6,
  GPIO_PIN7
}pin_index_t;

typedef enum{
   GPIO_PORTA_INDEX = 0,
   GPIO_PORTB_INDEX,
   GPIO_PORTC_INDEX,
   GPIO_PORTD_INDEX,
   GPIO_PORTE_INDEX,
}port_index_t;

typedef struct{
  uint8 port : 3;
  uint8 pin : 3;
 uint8 direction : 1;
 uint8 logic : 1;
}pin_config_t;

Std_ReturnType GPIO_pin_direction_intialize(const pin_config_t *_pin_config);
```

Std_ReturnType GPIO_pin_get_direction_status(const pin_config_t *_pin_config , direction_t *direction_status);

Std_ReturnType GPIO_pin_write_logic(const pin_config_t *_pin_config , logic_t logic);

Std_ReturnType GPIO_pin_read_logic(const pin_config_t *_pin_config , logic_t *logic_status);

Std_ReturnType GPIO_pin_toggle_logic(const pin_config_t *_pin_config);

Std_ReturnType GPIO_pin_intialize(const pin_config_t *_pin_config);

Std_ReturnType GPIO_port_direction_intialize(port_index_t port , uint8 direction);

Std_ReturnType GPIO_port_get_direction_status(port_index_t port , uint8 *direction_status);

Std_ReturnType GPIO_port_write_logic(port_index_t port , uint8 logic);

Std_ReturnType GPIO_port_read_logic(port_index_t port , uint8 *logic_status);

Std_ReturnType GPIO_port_toggle_logic(port_index_t port);

1.2 INTERRUPT

typedef enum{

MODE_LOW_LEVEL,

MODE_LOGICAL_CHANGE,

MODE_FALLING_EDGE,

MODE_RISING_EDGE

}enu_EXTI_MODES;

typedef enum{

INT0,

INT1,

INT2

}enu_EXTI;

typedef struct{

enu_EXTI INTx;

enu_EXTI_MODES   INTx_MODE;

}EXTI_cfg;

Std_ReturnType EXTI_vEnableEXTI(const EXTI_cfg *INTx);

Std_ReturnType EXTI_vDisableEXTI(const EXTI_cfg *INTx);

1.3 TIMERS

typedef enum{

TIMER0_NORMAL,

TIMER0_CTC,

TIMER1_NORMAL,

TIMER1_CTC_OCR,

TIMER1_CTC_ICR,

TIMER2_NORMAL,

TIMER2_CTC

}enu_TimerChannel;

typedef enum{

TMR_PRE_NO_CLK,

TMR_PRE_0,

TMR_PRE_8,

TMR_PRE_64,

TMR_PRE_256,

TMR_PRE_1024,

TMR_PRE_EXT_FALLING,

TMR_PRE_EXT_RISING,

TMR_PRE_32,

TMR_PRE_128

```
}enu_prescale_modes;

typedef enum{
TMR_OCMode,
TMR_InterruptMode
}enu_TimerToggleMode;

typedef enum{
OC_Disconnected,
OC_Toggle,
OC_Clear,
OC_Set
}enu_TimerOCMode;

typedef struct{
enu_TimerChannel     TMR_TimerChannel;
enu_prescale_modes   TMR_Prescale;
enu_TimerToggleMode  TMR_ToggleMode;
enu_TimerOCMode   TMR_OCMode;
}TMR_cfg_t;

Std_ReturnType TMR_vInit(const TMR_cfg_t *TMR);
Std_ReturnType TMR_vStop(const TMR_cfg_t *TMR);
Std_ReturnType TMR_vSetICRValue(const TMR_cfg_t *TMR, u16
Copy_u16ICRValue);
Std_ReturnType TMR_vSetOCRValue(const TMR_cfg_t *TMR, u16
Copy_u16OCRValue);
Std_ReturnType TMR_vStartTimer(const TMR_cfg_t *TMR);
```

1.4 PWM

typedef enum{

TIMER0_PhaseCorrect,

TIMER0_FastPWM,

TIMER1_Phase8,

TIMER1_Phase9,

TIMER1_Phase10,

TIMER1_Fast8,

TIMER1_Fast9,

TIMER1_Fast10,

TIMER1_PhaseFreqICR,

TIMER1_PhaseFreqOCR,

TIMER1_PhaseICR,

TIMER1_PhaseOCR,

TIMER1_FastICR,

TIMER1_FastOCR,

TIMER2_PhaseCorrect,

TIMER2_FastPWM

}enu_pwmRunningMode;

typedef enum{

PWM_PRE_NO_CLK,

PWM_PRE_0,

PWM_PRE_8,

PWM_PRE_64,

PWM_PRE_256,

PWM_PRE_1024,

PWM_PRE_EXT_FALLING,

```
PWM_PRE_EXT_RISING,
PWM_PRE_32,
PWM_PRE_128
}enu_pwmPrescale;

typedef enum{
PWM_InvertingMode,
PWM_NonInvertingMode
}enu_pwmInverting;

typedef enum{
PWM_OCmode,
PWM_InterruptMode
}enu_pwmToggleMode;

typedef struct{
enu_pwmRunningMode  PWM_TimerChannel;
enu_pwmPrescale   PWM_TimerPrescale;
enu_pwmToggleMode  PWM_ToggleMode;
enu_pwmInverting   PWM_InvertOrNot;
}PWM_cfg_t;

Std_ReturnType PWM_vInit(const PWM_cfg_t *PWM);
Std_ReturnType PWM_vSetICR(const PWM_cfg_t *PWM, u16 Copy_u16ICRValue);
Std_ReturnType PWM_vSetOCR(const PWM_cfg_t *PWM, u16 Copy_u16OCRValue);
Std_ReturnType PWM_vStart(const PWM_cfg_t *PWM);
Std_ReturnType PWM_vStop(const PWM_cfg_t *PWM);
```

2. ECU

2.1 LED

```
typedef enum{
   LED_STATUS_OFF = 0,
   LED_STATUS_ON,
}led_status_t;

typedef struct{
   uint8 port_name :3;
   uint8 pin : 3;
   uint8 led_status : 1;
   uint8 reserved : 1;
}led_t;

Std_ReturnType LED_initialize(const led_t *led);
Std_ReturnType LED_turn_on(const led_t *led);
Std_ReturnType LED_turn_off(const led_t *led);
Std_ReturnType LED_toggle(const led_t *led);
```

2.2 BUTTON

```
typedef enum{
   PUSH_BTN_STATE_PRESSED = 0,
   PUSH_BTN_STATE_RELEASED
}PUSH_BTN_state_t;



typedef enum{
```

```
    PUSH_BTN_PULL_UP = 0,
    PUSH_BTN_PULL_DOWN
}PUSH_BTN_active_t;

typedef struct{
    pin_config_t PUSH_BTN_pin;
    PUSH_BTN_state_t PUSH_BTN_state;
    PUSH_BTN_active_t PUSH_BTN_connection;
}PUSH_BTN_t;


Std_ReturnType PUSH_BTN_intialize(const PUSH_BTN_t *btn);
Std_ReturnType PUSH_BTN_read_state(const PUSH_BTN_t *btn , PUSH_BTN_state_t
*btn_state);
```

2.3 MOTOR

```
Typedef struct{
Uint8 dc_motor_port :4;
Uint8 dc_motor_pin :4;
Uint8 dc_motor_pin_status :4;
}dc_motor_config_t;

Std_ReturnType DC_motor_initialize(const dc_motor_config_t  *_dc_motor);
Std_ReturnType DC_motor_move_right(const dc_motor_config_t  *_dc_motor);
Std_ReturnType DC_motor_move_left(const dc_motor_config_t  *_dc_motor);
Std_ReturnType DC_motor_stop(const dc_motor_config_t  *_dc_motor);
```

3.APPLICATION

This Layer Will Have The Main Code Of The System