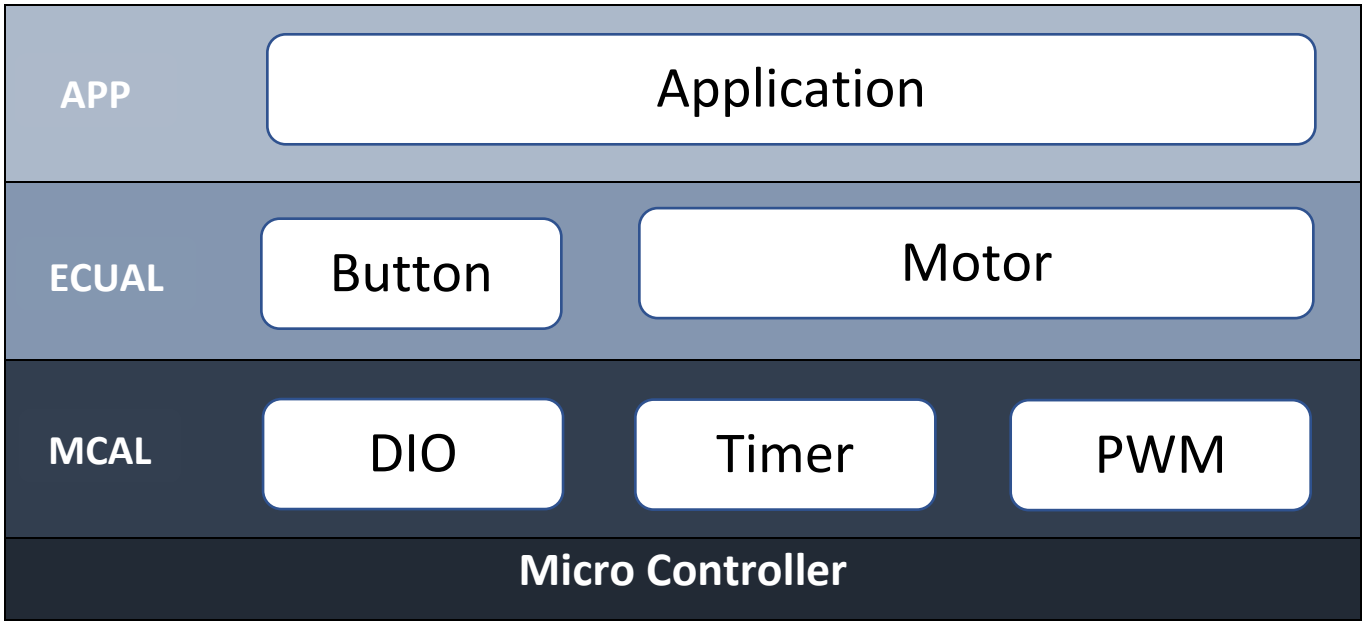
Static Design Task

A) Layered Architecture:



B) Defining APIs:

I) GLOBAL Enum:

```
//return status of ALL functions
typedef enum{
    ERROR,
    SUCCESS
}E_STATUS_t;
```

I) MCAL LAYER:

1- DIO APIs:

```c
//MCU DIO Pins
typedef enum {
    PINA0,
    PINA1,
    ..... ,
    PIND7
}DIO_PinNum_t;

//Pin status of DIO pins
typedef enum {
    OUTPUT,
    INFREE,
    INPLUP
}DIO_PinMode_t;

typedef enum {
    LOW,
    HIGH
}DIO_PinStatus_t;


//Initialization values for all MCU pins
DIO_PinMode_t DIO_InitSetup[MCU_Pin_number]={OUTPUT,INFREE,INPLUP, ..... ,OUTPUT};

//DIO APIs
E_STATUS_t DIO_Init(void); //Initialize all pins accordin to DIO_InitSetup array
E_STATUS_t DIO_InitPin(DIO_PinNum_t Pin_Num,DIO_PinMode_t Mode);
E_STATUS_t DIO_WritePin(DIO_PinNum_t Pin_Num, DIO_PinStatus_t Value);
E_STATUS_t DIO_ReadPin(DIO_PinNum_t Pin_Num,DIO_PinStatus_t *Value);
```

2- Timer APIs:

```c
//timer modes
typedef enum {
    Normal,
    CTC,
    F_PWM,
    PC_PWM
}T_TimerMode_t;

typedef enum {
    Timer_0,
    Timer_1,
    Timer_2
}T_TimerSelect_t;

//Timer prescaler values
typedef enum {T_Prescaler_1, T_Prescaler_8, T_Prescaler_16, .... , T_Prescaler_1024} T_TimerPrescaler_t;

//Timer APIs
E_STATUS_t TimerInit(T_TimerSelect_t Timer,T_TimerMode_t Mode, T_TimerPrescaler_t Prescaler);
E_STATUS_t TimerStart(T_TimerSelect_t Timer,uint32_t Time);
E_STATUS_t TimerCallback(T_TimerSelect_t Timer,uint32_t Time, void(*fptr)(void));
E_STATUS_t TimerStop(T_TimerSelect_t Timer);
E_STATUS_t TimerGetStatus(T_TimerSelect_t Timer);
```

3- PWM APIs:

```c
//PWM channels
typedef enum {
    PWM_CH0,        //Timer 0
    PWM_CH1,        //Timer 1 OCR1A
    PWM_CH2,        //Timer 1 OCR1B
    PWM_CH3,        //Timer 2
}PWM_CH_t;

//PWM Modes
typedef enum {
    PWM_Fast,
    PWM_Phase
}T_PWMMode_t;

//PWM APIs
E_STATUS_t PWM_Init(PWM_CH_t Channel,T_PWMMode_t Mode, uint32_t Frequency, uint8_t DutyCycle);
E_STATUS_t PWM_SetFrequency(PWM_CH_t Channel,uint32_t Frequency);
E_STATUS_t PWM_SetDutyCycle(PWM_CH_t Channel,uint8_t DutyCycle);
E_STATUS_t PWM_Stop(PWM_CH_t Channel);
```

II) ECUAL APIs:

1- Button APIs:

```c
// Button States
typedef enum {
    Not_Pressed,
    Pressed
}B_Status_t;

//Buttons in the system
typedef enum {
    Button_0,
    Button_1,
    Button_2,
    Button_3
}B_BtnNum_t;


E_STATUS_t ButtonInit(void);
E_STATUS_t ButtonRead(B_BtnNum_t Button,B_Status_t* Value);
```

2- Motor APIs:

```c
//Motors selection
typedef enum {
    Motor0,
    Motor1
}B_MotorNum_t;


E_STATUS_t MotorInit (void);
//for DC Motors
E_STATUS_t MotorStart(B_MotorNum_t);
E_STATUS_t MotorStop(B_MotorNum_t);
E_STATUS_t MotorSetSpeed(B_MotorNum_t, uint8_t Speed);
//for Servo Motors
E_STATUS_t MotorSetAngle(B_MotorNum_t, uint8_t Speed);
```

III) Application APIs:

```c
//Application APIs


E_STATUS_t App_Init(void);
E_STATUS_t App_Update(void);
```