

LED Sequence V1

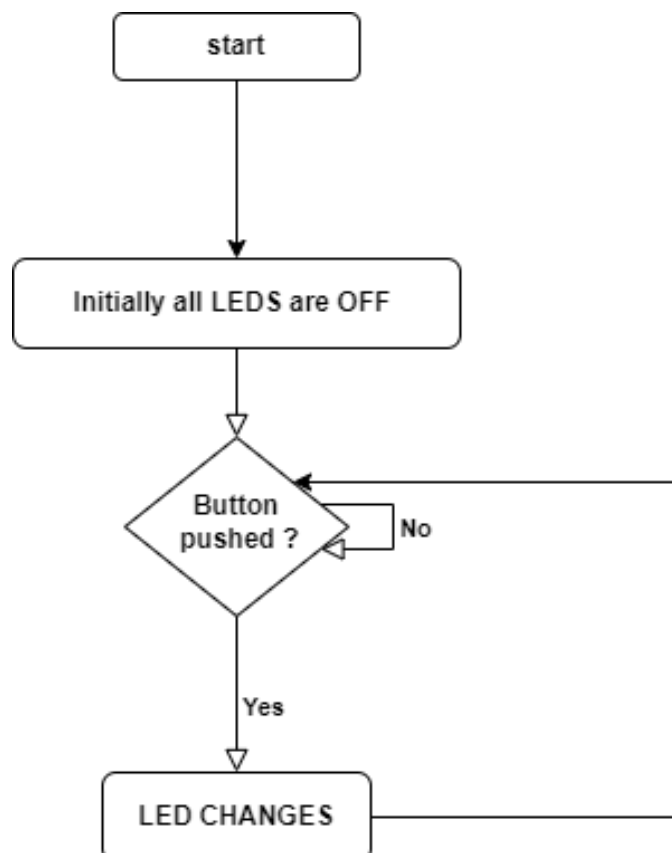
By Sharpel Malak

Project Description :

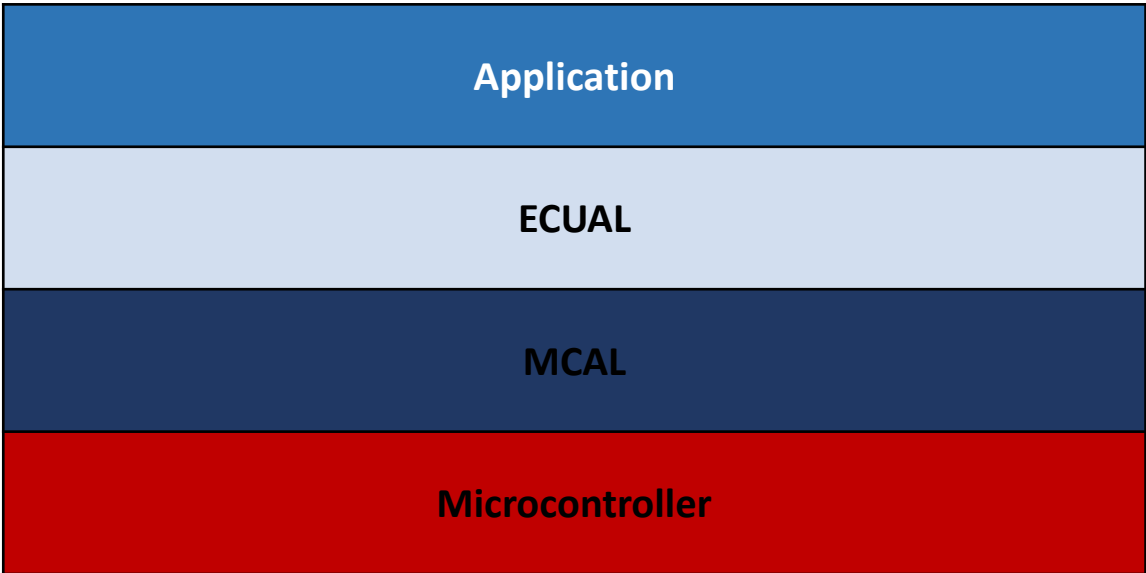
The project contains Four LEDs (LED0, LED1, LED2, LED3) and one button (BUTTON0)

- Initially, all LEDs are OFF
- Once BUTTON0 is pressed, LED0 will be ON
- Each press further will make another LED is ON
- At the fifth press, LED0 will changed to be OFF
- Each press further will make only one LED is OFF
- This will be repeated forever

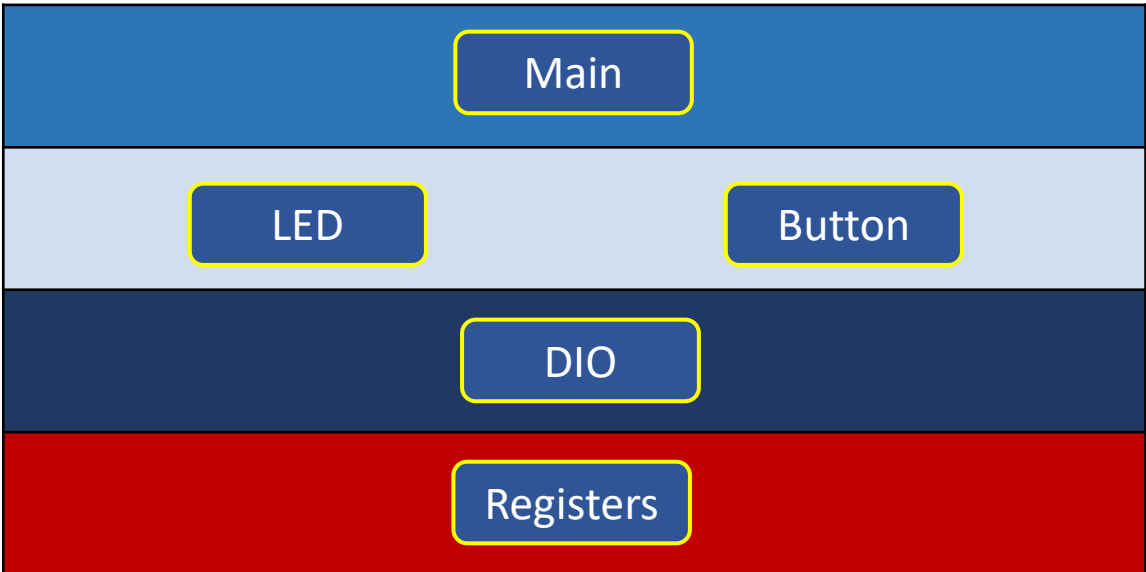
Project Flowchart :



1 : Layered Architecture :



2 : System modules



3 : Project Modules APIs

3.1 - DIO

// DIO TYPEDEFS

```
typedef enum EN_DIO_ERROR{
```

```
    DIO_OK=0,
```

```
    DIO_NOT_OK
```

```
}EN_DIO_ERROR;
```

```
typedef enum EN_DIO_PINS{
```

```
    DIO_PIN0=0,
```

```
    DIO_PIN1,
```

```
    DIO_PIN2,
```

```
    DIO_PIN3,
```

```
    DIO_PIN4,
```

```
    DIO_PIN5,
```

```
    DIO_PIN6,
```

```
    DIO_PIN7,
```

```
}EN_DIO_PINS;
```

```
typedef enum EN_DIO_PORTS{
```

```
    DIO_PORTA=0,
```

```
    DIO_PORTB,
```

```
    DIO_PORTC,
```

```
    DIO_PORTD
```

```
}EN_DIO_PORTS;
```

```
typedef enum EN_DIO_DIRECTION{
```

```
    INPUT=0,
```

```
    OUTPUT
```

```
}EN_DIO_DIRECTION;
```

```
typedef enum EN_DIO_LEVEL{
```

```
    LOW=0,
```

```
    HIGH
```

```
}EN_DIO_LEVEL;
```

// DIO FUNCTIONS PROTOTYPES

// Description : This function initialize PIN and set it's direction

// ARGS : take PIN Number and PORT Number and Direction (INPUT,OUTPUT)

// return : return DIO_OK if the PIN initializes correctly, DIO_NOT_OK otherwise

```
EN_DIO_ERROR DIO_init(EN_DIO_PINS pinNumber,EN_DIO_PORTS portNumber,EN_DIO_DIRECTION direction);
```

// Description : This function write on PIN and set it's level

// ARGS : take PIN Number and PORT Number and level (LOW,HIGH)

// return : return DIO_OK if the PIN level sets correctly, DIO_NOT_OK otherwise

```
EN_DIO_ERROR DIO_write(EN_DIO_PINS pinNumber,EN_DIO_PORTS portNumber,EN_DIO_LEVEL level);
```

// Description : This function toggles PIN level

// ARGS : take PIN Number and PORT Number

// return : return DIO_OK if the PIN toggles correctly, DIO_NOT_OK otherwise

```
EN_DIO_ERROR DIO_toggle(EN_DIO_PINS pinNumber,EN_DIO_PORTS portNumber);
```

// Description : This function reads PIN level and store it in the variable

// ARGS : take PIN Number and PORT Number and pointer to the variable

// return : return DIO_OK if the PIN value stored correctly , DIO_NOT_OK otherwise

```
EN_DIO_ERROR DIO_read(EN_DIO_PINS pinNumber,EN_DIO_PORTS portNumber,uint8_t * value);
```

3.2 - LED

```
typedef enum EN_LED_Error_t
{
    LED_OK = 0,
    LED_NOT_OK
}EN_LED_Error_t;
```

```
EN_LED_Error_t LED_init(EN_DIO_PINS pinNumber,EN_DIO_PORTS portNumber);
EN_LED_Error_t LED_on(EN_DIO_PINS pinNumber,EN_DIO_PORTS portNumber);
EN_LED_Error_t LED_off(EN_DIO_PINS pinNumber,EN_DIO_PORTS portNumber);
EN_LED_Error_t LED_toggle(EN_DIO_PINS pinNumber,EN_DIO_PORTS portNumber);
```

3.3 - Button

// Button typedefs

```
typedef enum EN_BTN_Error_t
{
    BTN_OK = 0,
    BTN_NOT_OK
}EN_BTN_Error_t;
```

```
EN_BTN_Error_t Button_init(EN_DIO_PINS pinNumber,EN_DIO_PORTS portNumber);
EN_BTN_Error_t Button_read(EN_DIO_PINS pinNumber,EN_DIO_PORTS portNumber,uint8_t
*value);
```

4 : APIs State Machine

