LED_SEQUANCE V1.0

Created By : Sherif Ashraf Ali

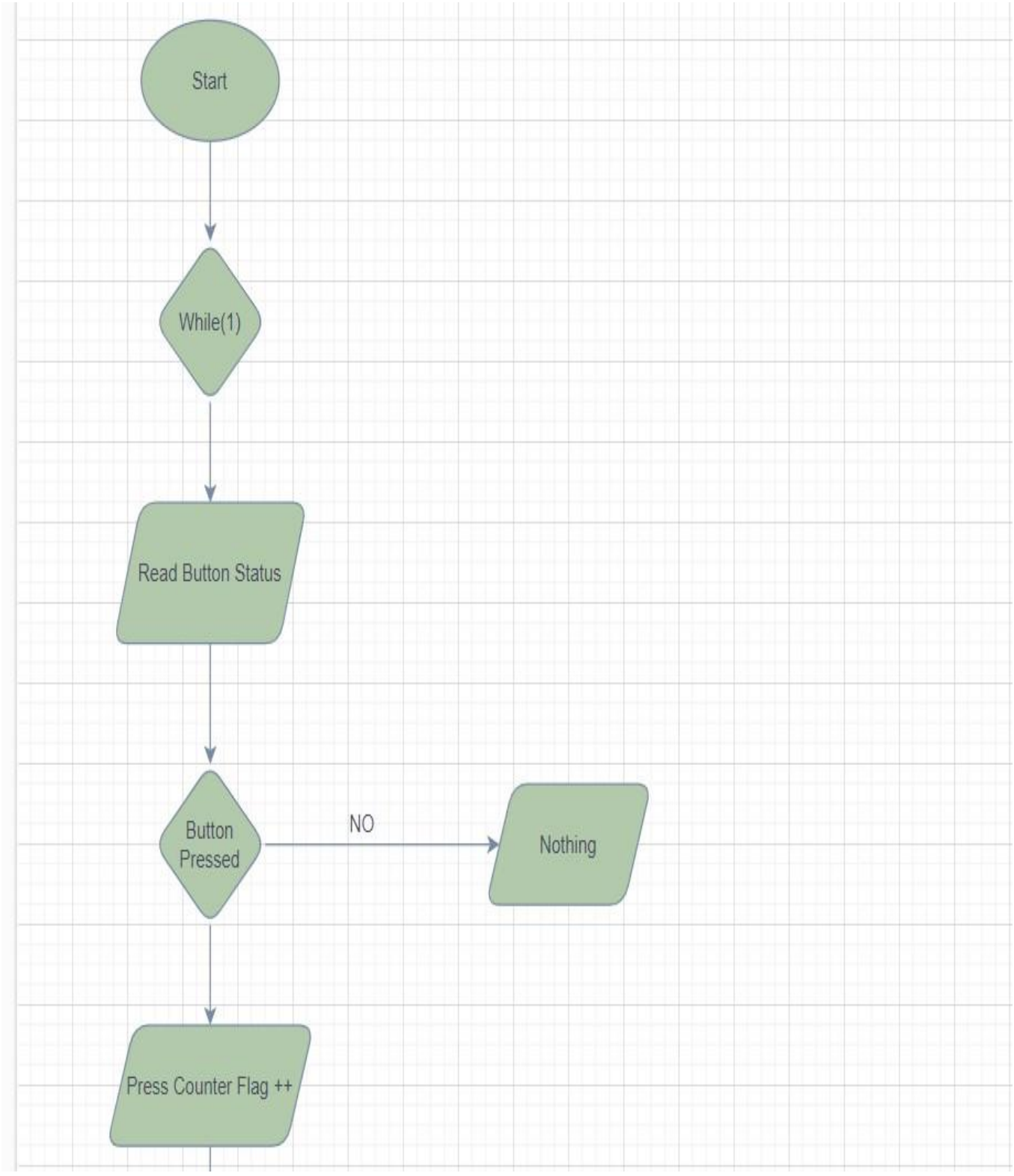
Date : 4/4/2023

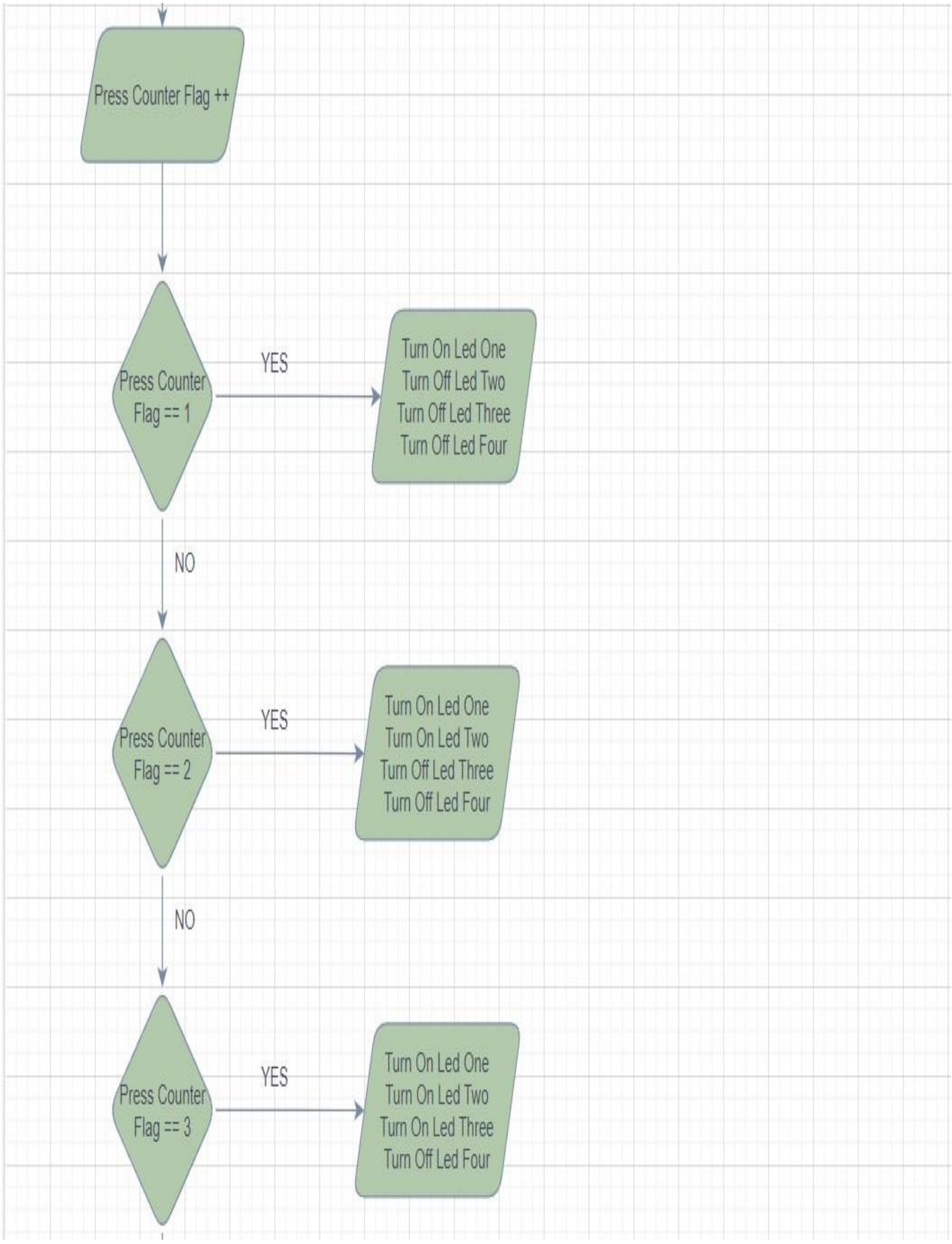
Project Description

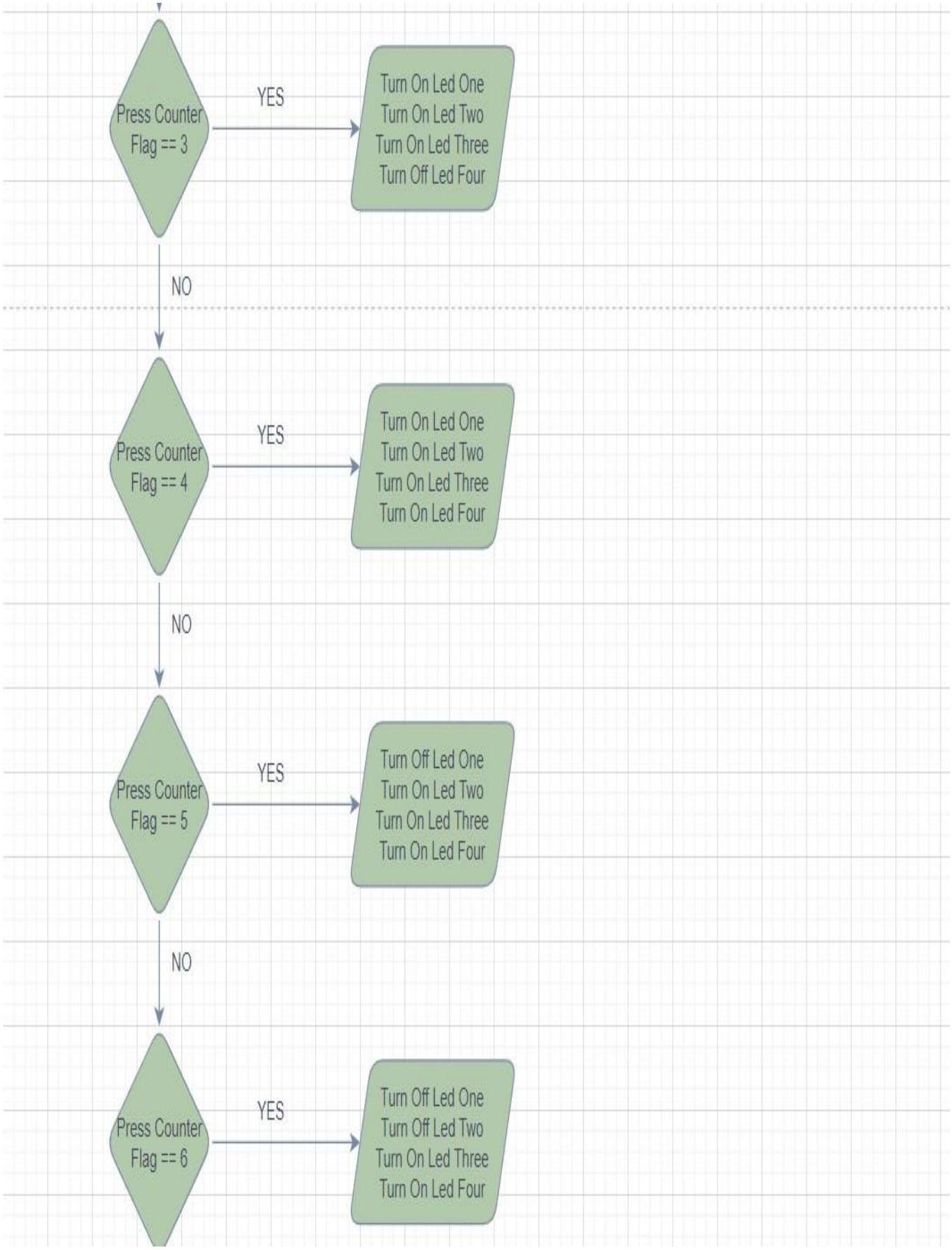
You are supposed to have a system that controls some LEDs lighting sequence according to button pressing.

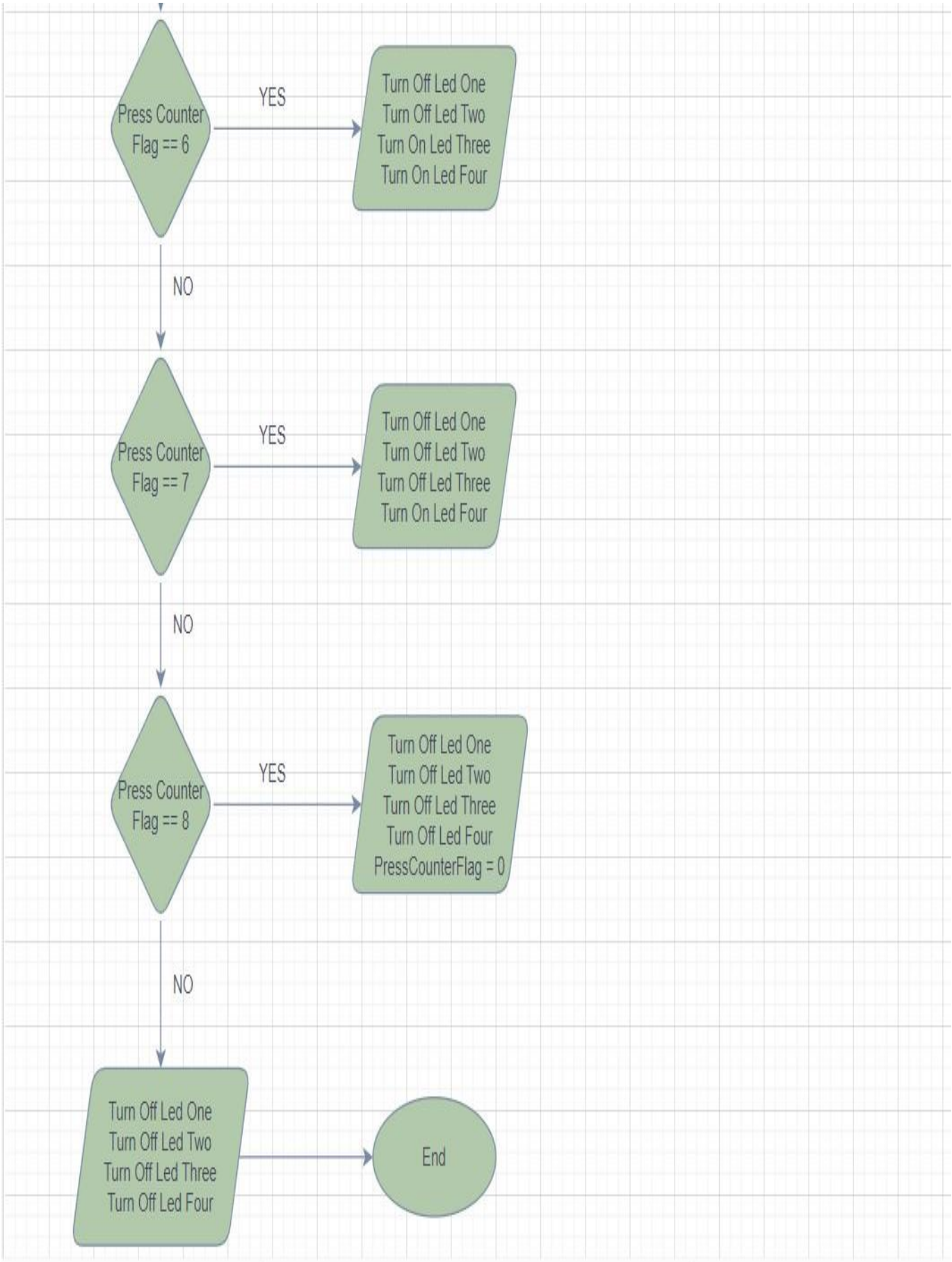
1. ***Hardware Requirements***
    1. *Four LEDs (LED0, LED1, LED2, LED3)*
    2. One button (BUTTON0)
2. ***Software Requirements***
    1. Initially, all LEDs are OFF
    2. Once BUTTON0 is pressed, LED0 will be ON
    3. Each press further will make another LED is ON
    4. At the fifth press, LED0 will changed to be OFF
    5. Each press further will make only one LED is OFF
    6. This will be repeated forever
    7. The sequence is described below
        1. Initially (OFF, OFF, OFF, OFF)
        2. Press 1 (ON, OFF, OFF, OFF)
        3. Press 2 (ON, ON, OFF, OFF)
        4. Press 3 (ON, ON, ON, OFF)
        5. Press 4 (ON, ON, ON, ON)
        6. Press 5 (OFF, ON, ON, ON)
        7. Press 6 (OFF, OFF, ON, ON)
        8. Press 7 (OFF, OFF, OFF, ON)
        9. Press 8 (OFF, OFF, OFF, OFF)
        10. Press 9 (ON, OFF, OFF, OFF)

PROJECT FLOWCHART

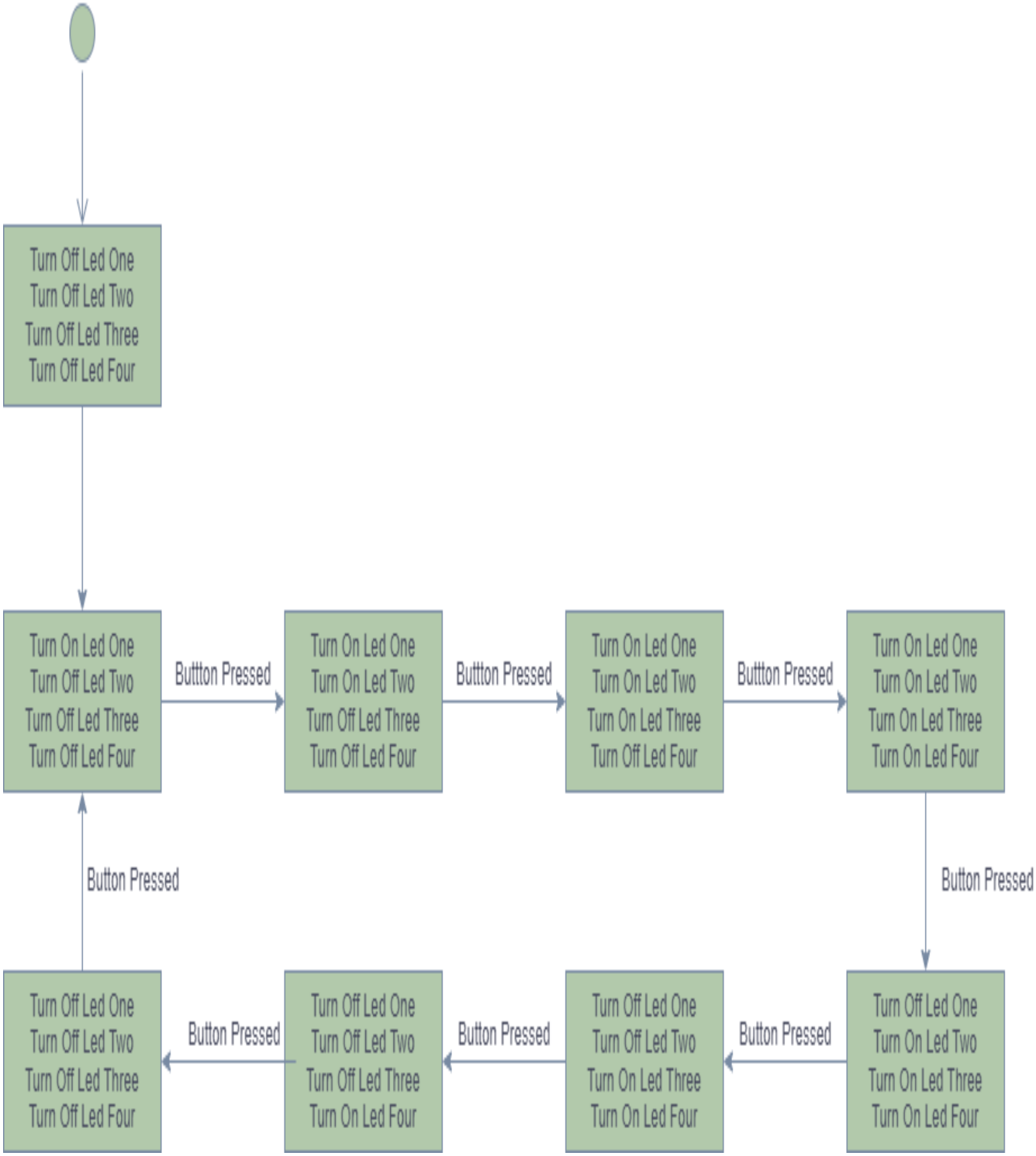



Press Counter Flag ++
Press Counter Flag == 1
YES
Turn On Led One
Turn Off Led Two
Turn Off Led Three
Turn Off Led Four
NO
Press Counter Flag == 2
YES
Turn On Led One
Turn On Led Two
Turn Off Led Three
Turn Off Led Four
NO
Press Counter Flag == 3
YES
Turn On Led One
Turn On Led Two
Turn On Led Three
Turn Off Led Four

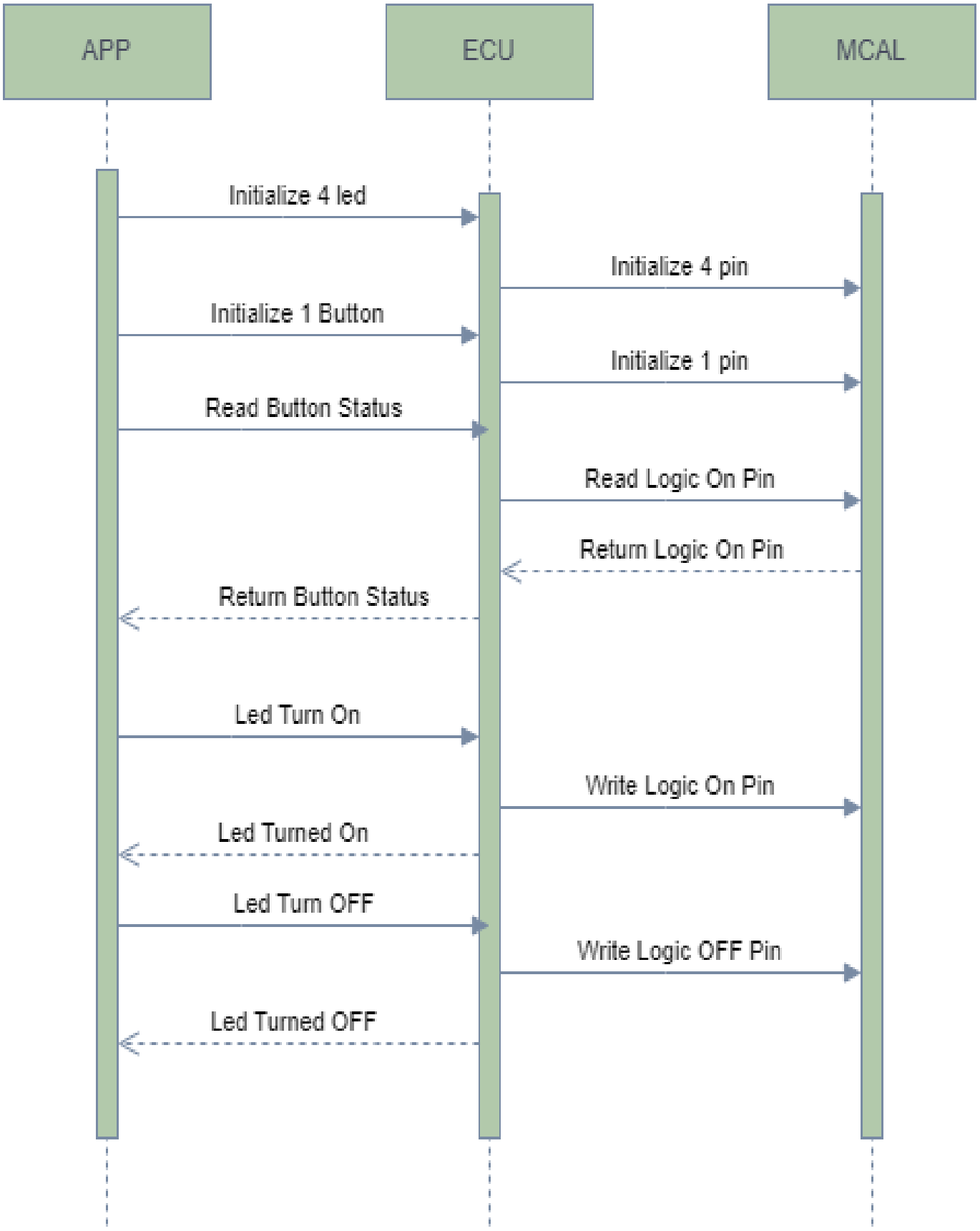

Press Counter
Flag == 3
YES
Turn On Led One
Turn On Led Two
Turn On Led Three
Turn Off Led Four
NO
Press Counter
Flag == 4
YES
Turn On Led One
Turn On Led Two
Turn On Led Three
Turn On Led Four
NO
Press Counter
Flag == 5
YES
Turn Off Led One
Turn On Led Two
Turn On Led Three
Turn On Led Four
NO
Press Counter
Flag == 6
YES
Turn Off Led One
Turn Off Led Two
Turn On Led Three
Turn On Led Four


Press Counter
Flag == 6
YES
Turn Off Led One
Turn Off Led Two
Turn On Led Three
Turn On Led Four
NO
Press Counter
Flag == 7
YES
Turn Off Led One
Turn Off Led Two
Turn Off Led Three
Turn On Led Four
NO
Press Counter
Flag == 8
YES
Turn Off Led One
Turn Off Led Two
Turn Off Led Three
Turn Off Led Four
PressCounterFlag = 0
NO
Turn Off Led One
Turn Off Led Two
Turn Off Led Three
Turn Off Led Four
End

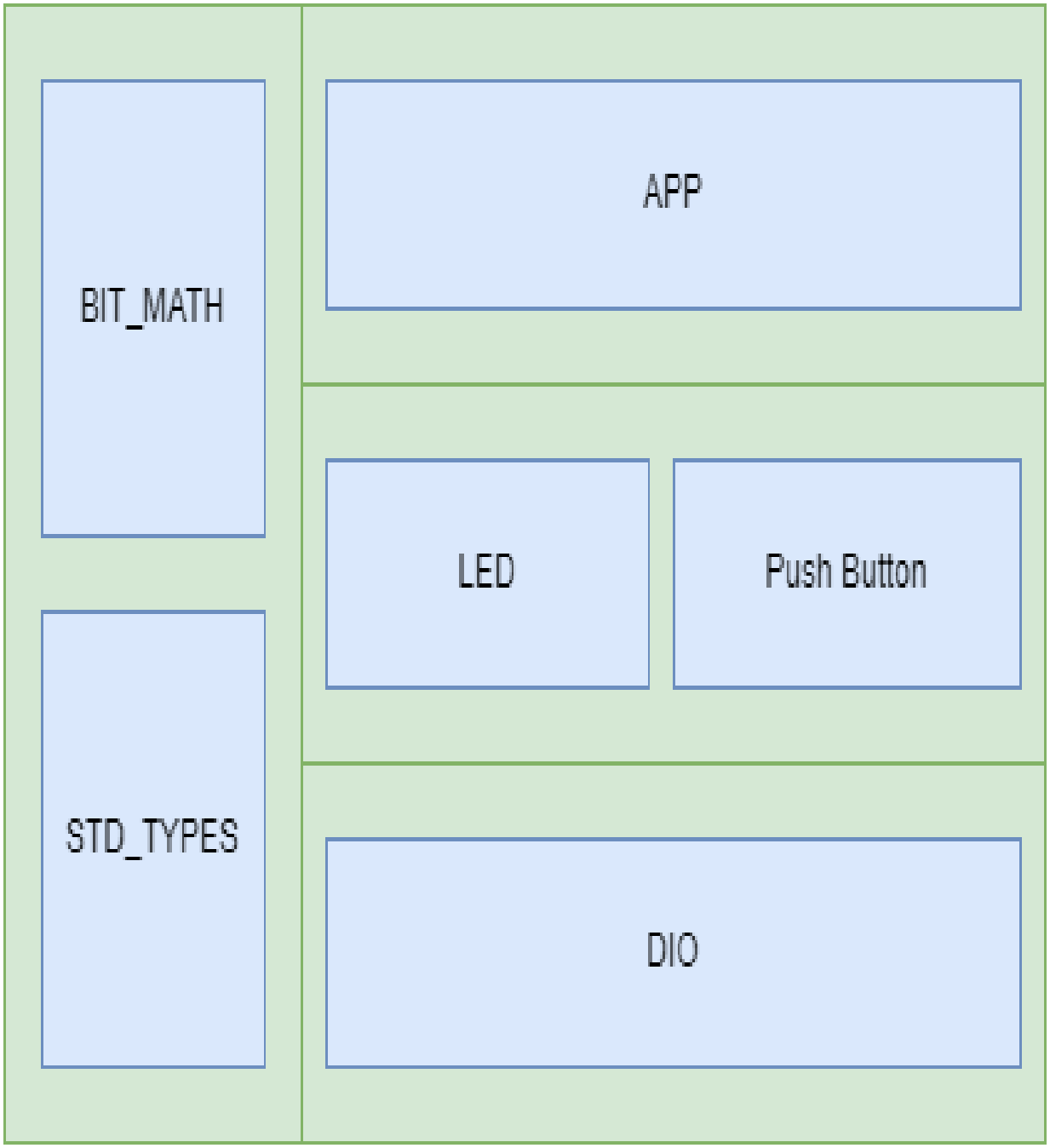
STATE MACHINE



Sequence Diagram



Layered architecture



Project Modules 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);
```

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

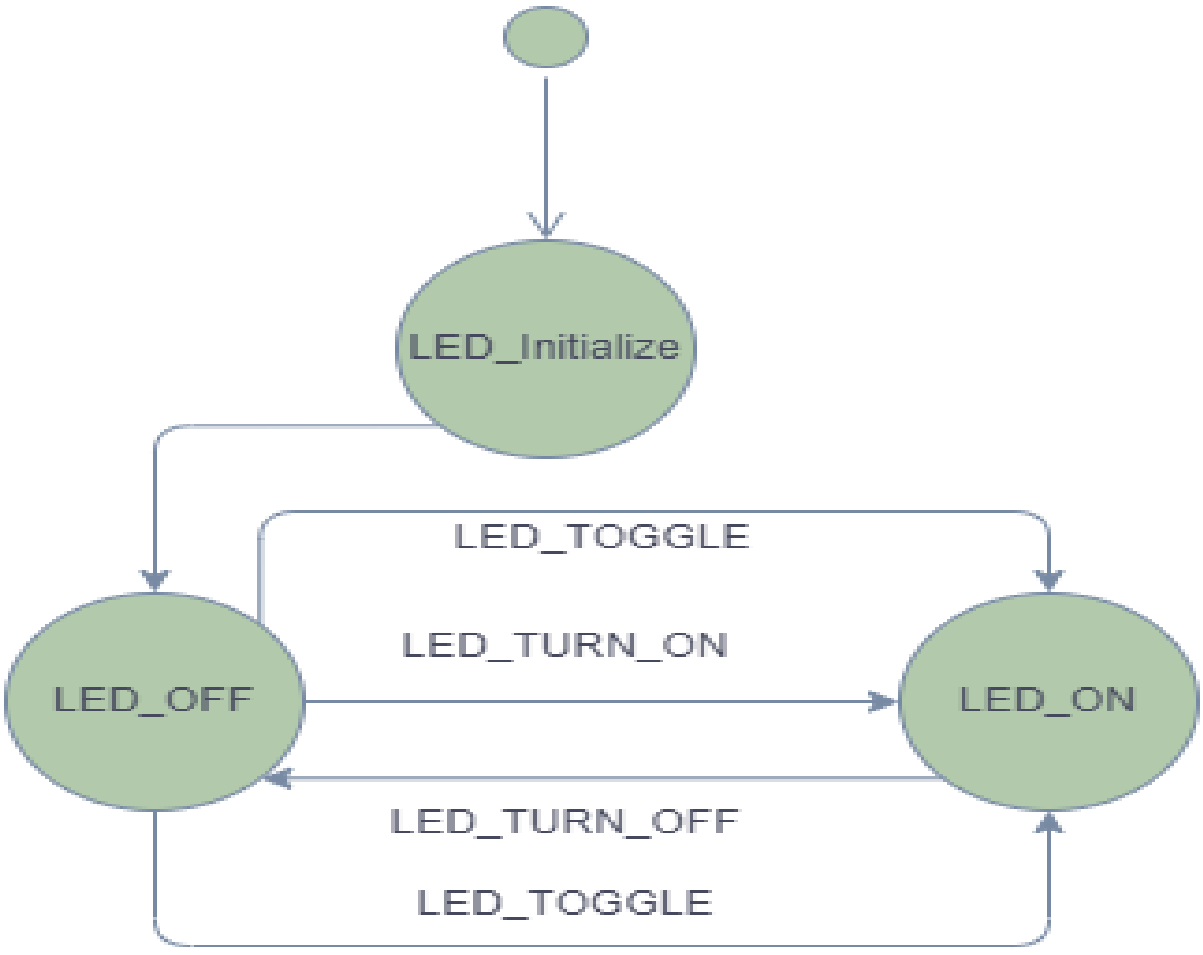
```
typedef enum{
    PUSH_BTN_STATE_PRESSED = 0,
    PUSH_BTN_STATE_RELEASED
}PUSH_BTN_state_t;



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;
```

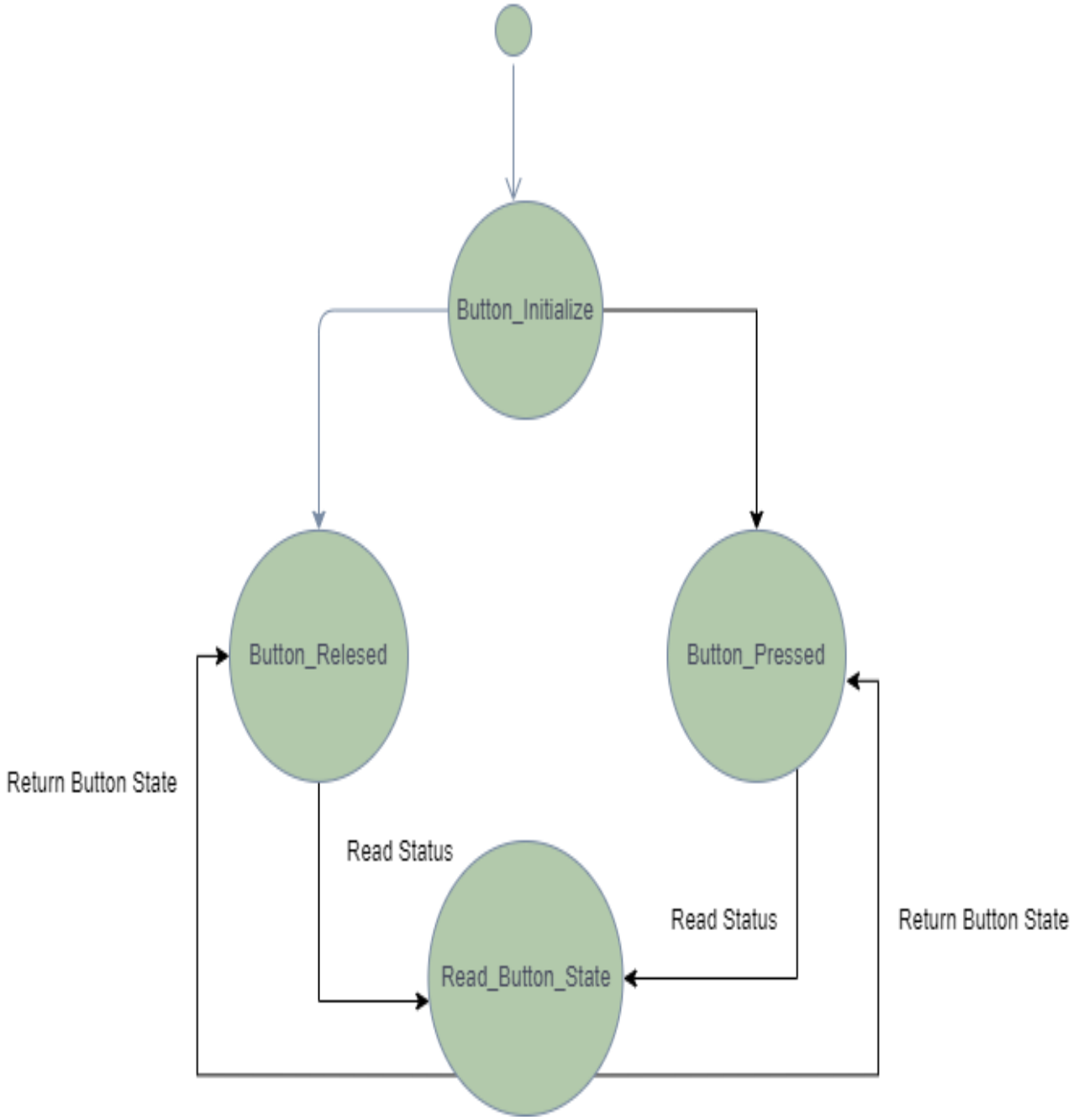
```
Std_ReturnType PUSH_BTN_intialize(const PUSH_BTN_t *btn);
Std_ReturnType PUSH_BTN_read_state(const PUSH_BTN_t *btn , PUSH_BTN_state_t
*btn_state);
```

APIs state machine

LED



PUSH BUTTON


Button_Initialize
Button_Relesed
Button_Pressed
Return Button State
Read Status
Read Status
Return Button State
Read_Button_State