Introduction

Project Title:

LED sequence V3.0

Name:

Nada Abdelazim Mohamed

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. **Two** buttons (**BUTTON0** and **BUTTON1**)

2. Software :

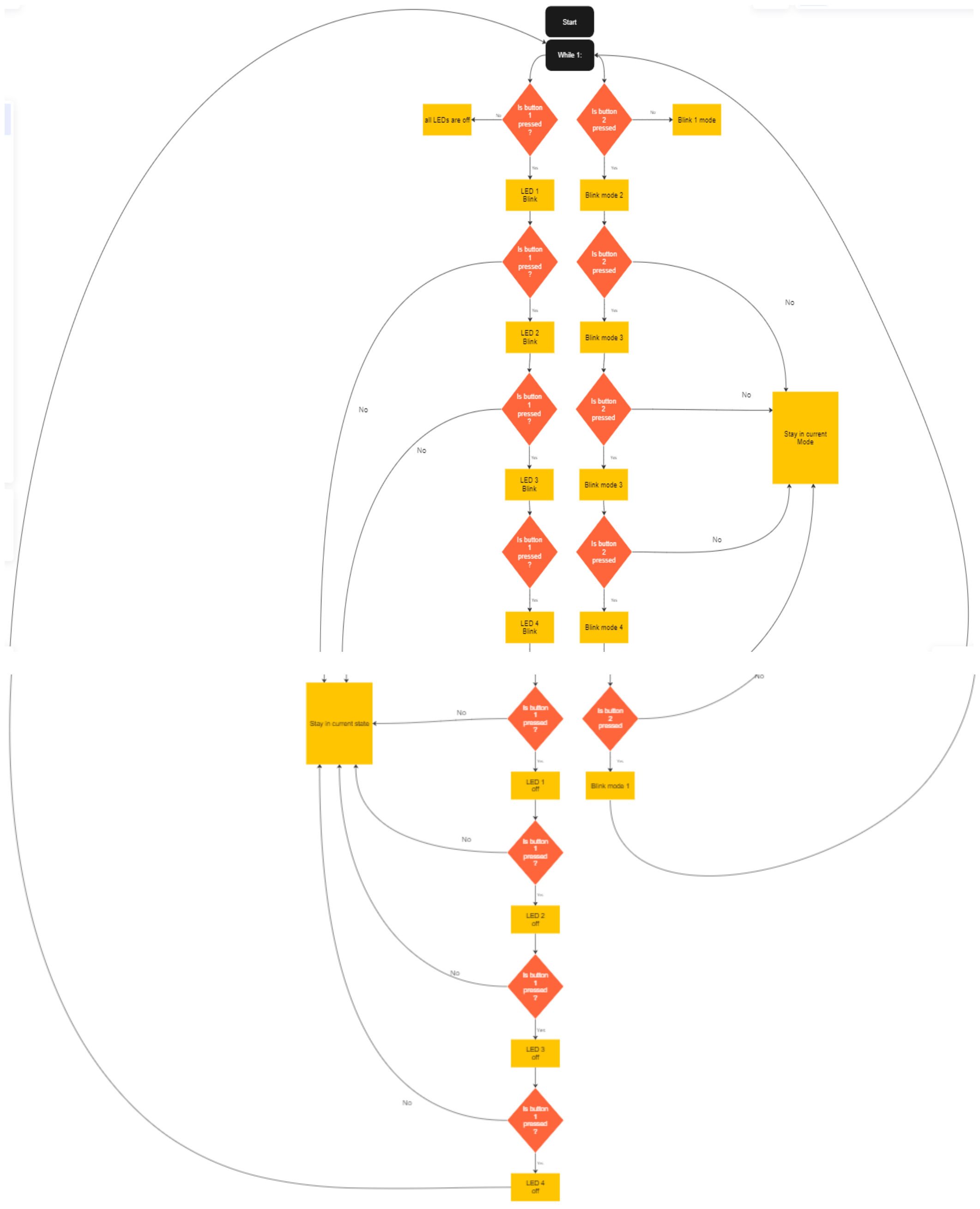
1. Initially, all LEDs are OFF
2. Once **BUTTON0** is pressed, **LED0** will blink with **BLINK_1** mode
3. Each press further will make another LED blinks **BLINK_1** mode
4. At the **fifth press**, **LED0** will changed to be **OFF**
5. Each **press further** will make only one LED is **OFF**
6. This will be repeated forever

7. The sequence is described below:
1. Initially (OFF, OFF, OFF, OFF)
2. Press 1 (BLINK_1, OFF, OFF, OFF)
3. Press 2 (BLINK_1, BLINK_1, OFF, OFF)
4. Press 3 (BLINK_1, BLINK_1, BLINK_1, OFF)
5. Press 4 (BLINK_1, BLINK_1, BLINK_1, BLINK_1)
6. Press 5 (OFF, BLINK_1, BLINK_1, BLINK_1)
7. Press 6 (OFF, OFF, BLINK_1, BLINK_1)
8. Press 7 (OFF, OFF, OFF, BLINK_1)
9. Press 8 (OFF, OFF, OFF, OFF)
10. Press 9 (BLINK_1, OFF, OFF, OFF)

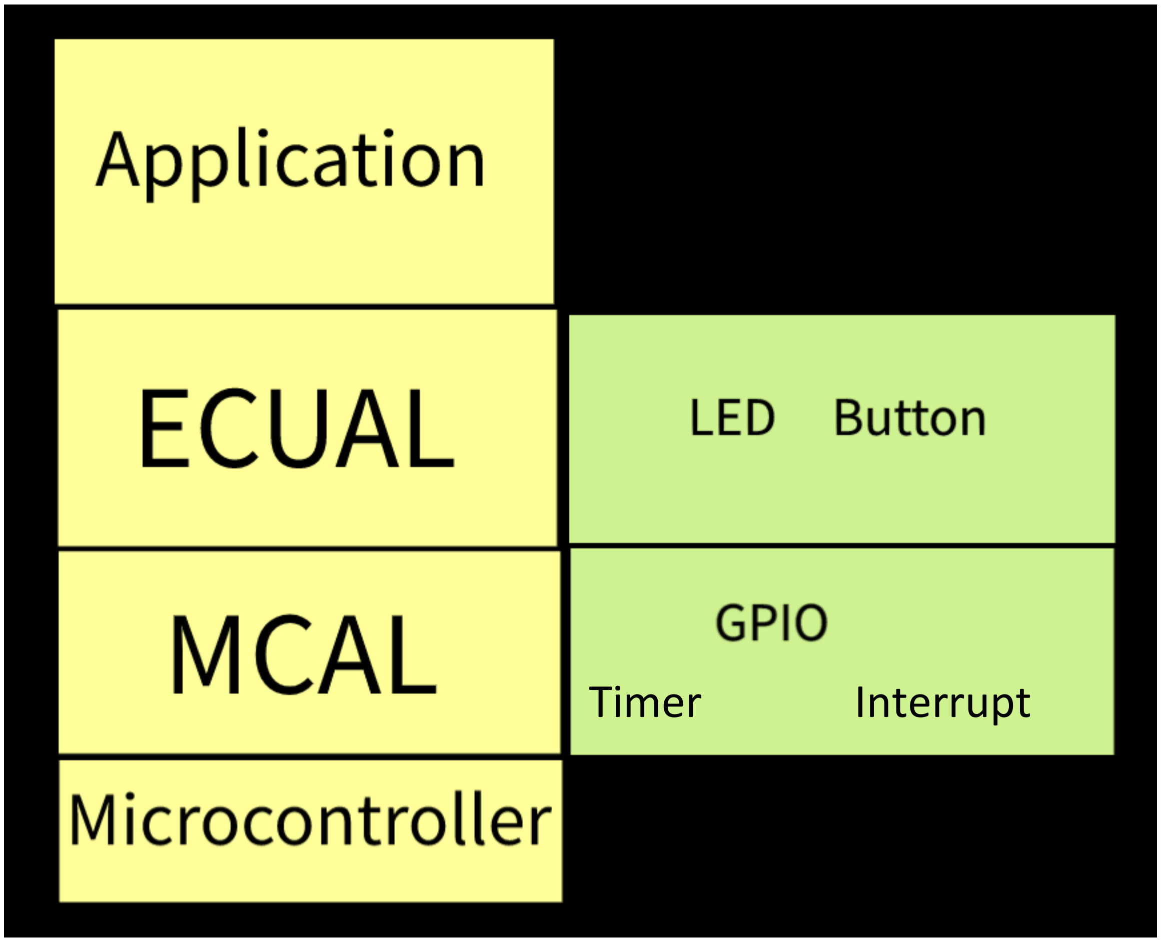
8. When BUTTON1 has pressed the blinking on and off durations will be changed :
1. No press → **BLINK_1** mode (**ON**: 100ms, **OFF**: 900ms)
2. First press → **BLINK_2** mode (**ON**: 200ms, **OFF**: 800ms)
3. Second press → **BLINK_3** mode (**ON**: 300ms, **OFF**: 700ms)
4. Third press → **BLINK_4** mode (**ON**: 500ms, **OFF**: 500ms)
5. Fourth press → **BLINK_5** mode (**ON**: 800ms, **OFF**: 200ms)
6. Fifth press → **BLINK_1** mode

9. USE EXTERNAL INTERRUPTS

Project Flowchart



Layered architecture



Layers description:

(1)- Application layer:

Contains functions calls to implement the main project.

(2)- ECUAL: “Electronics Unit Abstraction Layer”

Contains Drivers of the external electronic devices which will be connected to the microcontroller and the system overall.

(3)-MCAL: “Microcontroller Abstraction Layer”

Contains interfaces of the microcontroller’s peripherals.

(4)-Microcontroller:

The microcontroller type that will be used to implement the project

APIs

1- GPIO API:

Functions prototypes:

```
void DIO_init(uint8_t pinNumber, uint8_t portNumber, uint8_t direction);

void DIO_write(uint8_t pinNumber, uint8_t portNumber, uint8_t value);

void DIO_read(uint8_t pinNumber, uint8_t portNumber, uint8_t *value);
```

2- LED API:

Functions prototypes:

```
void LED_init(uint8_t ledPort, uint8_t ledPin);

void LED_on(uint8_t ledPort, uint8_t ledPin);

void LED_off(uint8_t ledPort, uint8_t ledPin);
```

3- Button API:

Functions prototypes:

```
void Button_init(uint8_t buttonPort, uint8_t buttonPin);

void Button_read(uint8_t buttonPort, uint8_t buttonPin, uint8_t *value);
```

4- External Interrupt:

```
#define cli() __asm__ __volatile__ ("cli" ::: "memory")

#define ISR(INT_VECT)void INT_VECT(void) __attribute__ ((signal,used));\

void INT_VECT(void)

void Exit_enable0 (void);

void Exit_enable1 (void);

void Exit_disable0 (void);

void Global_interrupt_enable (void);

void Global_interrupt_disable (void);

void Exit0_init (void);
void Exit1_init (void);

void External_interrupt0_mode (uint8_t mode);

void External_interrupt1_mode (uint8_t mode);
```

5- Timer API:

```
void Timer0_init(TIMER0_Mode_type mode,TIMER0SCALER_type scaler);

void Timer0_overflowInterrupt_Disable(void);

void Timer0_overflowInterrupt_Enable(void);

void delay_ms(uint8_t delay_time);
```