"Heaven's Light is Our Guide"

Rajshahi University of Engineering & Technology

Department of Computer Science & Engineering

Lab Final Report

Course No. : CSE 3106

Course Title : Computer Interfacing & Embedded Systems

Submitted by -	***Submitted To -***
Group formed By 2003121- Khandoker Sefayet Alam , 2003131-Nahid Hasan , 2003141-Md. Wakilul Arifin Akash	Md. Farhan Shakib Lecturer Department of CSE, RUET

Title: Whack the LEDs

Description:

A microcontroller is used in the project to build the game "Whack the LEDs". Three LEDs are part of the system, and each has a button that goes with it. The player has to push the corresponding button before an LED that blinks randomly shuts off. An LED success indication illuminates if the player hits the right button at the right moment. A failure alert LED illuminates in the event that the player malfunctions. To ensure responsive and effective operation, the game uses SysTick to generate time delays and EXTI (External Interrupt) to record button pushes.

Components:

Hardware Components:

1. **Microcontroller:** STM32 or equivalent.
2. **LEDs:** 3 main LEDs (task LEDs) and 2 indicator LEDs (success and failure).
3. **Buttons:** 3 corresponding buttons for LED control.
4. **Power Supply:** 5V regulated power supply.
5. **Resistors:** Current-limiting resistors for LEDs.

Software Tools:

1. **IDE:** STM32CubeIDE or Keil uVision.
2. **Language:** Embedded C.
3. **Debugger:** ST-Link or inbuilt debugger.

Methodology:

The "Whack the LEDs" system is designed to implement an interactive game using LEDs and buttons. The process involves configuring GPIO pins for LEDs and buttons, setting up interrupts and timers, and implementing the game logic. Here’s a detailed description of the process:

1. **Hardware Configuration:**
    - The LEDs are connected to GPIO pins PA0, PA5, and PA6.
    - The buttons are mapped to PB0, PB1, and PB10, corresponding to the LEDs.

- Indicator LEDs for success and failure are connected to PA3 and PA4, respectively.

2. **Initialization:**
   - **GPIO Pins:** LEDs are initialized as output, while buttons are configured as input with pull-up resistors for stable readings.
   - **SysTick Timer:** A timer is initialized for generating delays in milliseconds to control LED blink durations and debounce the buttons.
   - **EXTI Interrupts:** External interrupts are configured for the button GPIO pins (PB0, PB1, and PB10). Each interrupt triggers when a button is pressed.

3. **Random LED Selection:**
   - A pseudo-random number generator is used to select one of the three LEDs (PA0, PA5, or PA6) to blink. This ensures unpredictability in the game.

4. **Game Logic Execution:**
   - A loop continuously controls the game. For each iteration, a random LED blinks for a fixed duration (e.g., 1 second).
   - During the LED's ON state, the system monitors the buttons for player input using external interrupts and a debounce mechanism.

5. **Input Validation:**
   - If the player presses the correct button corresponding to the active LED within the allotted time, the system considers it a success.
   - If no button is pressed or the wrong button is pressed, it is considered a failure.

6. **Feedback Indication:**
   - Success is indicated by turning on the success LED (PA3) for a short duration.
   - Failure is indicated by turning on the failure LED (PA4) for the same duration.
   - After each feedback, the LEDs are turned off to reset the system for the next iteration.

7. **Debouncing Buttons:**
   - To avoid false triggers due to mechanical noise, the button press is validated only if the signal remains stable for a short period (50ms).

This process ensures real-time interaction between the LEDs and buttons, providing a responsive and engaging gaming experience. The use of interrupts and SysTick enhances the efficiency of the system, while the debouncing mechanism ensures accurate input recognition.

Circuit Picture:

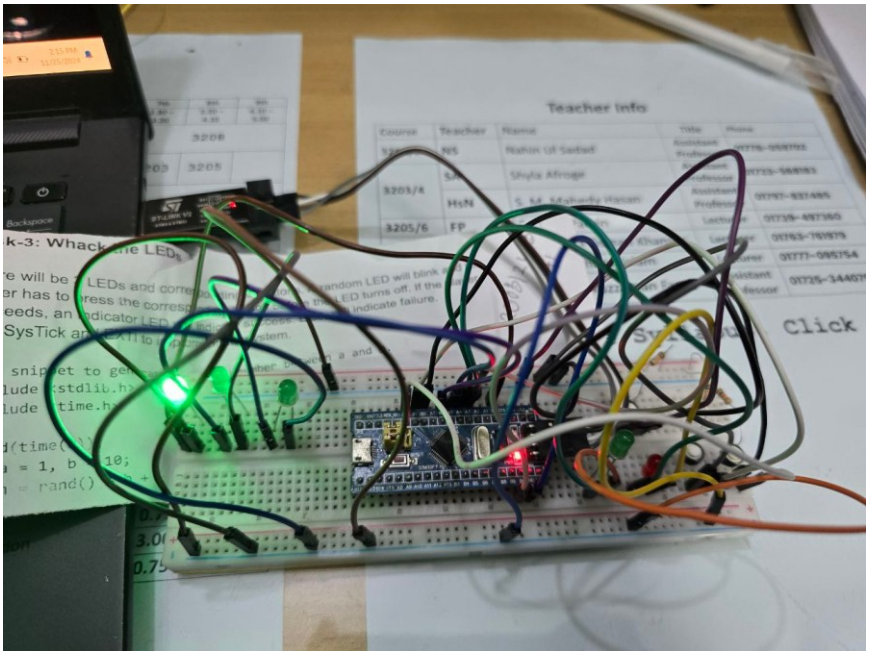


Fig: 1st LED light up

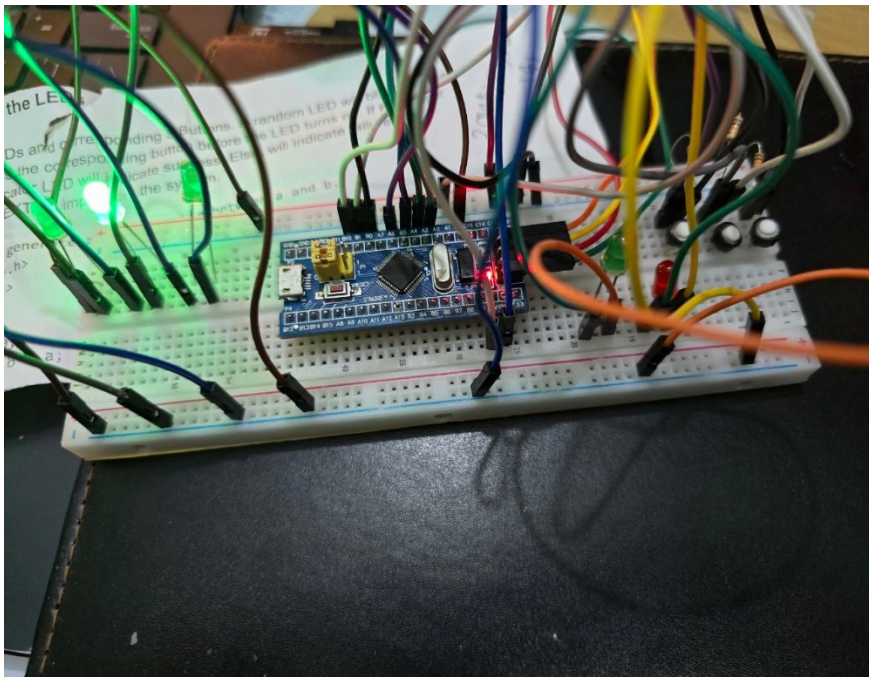
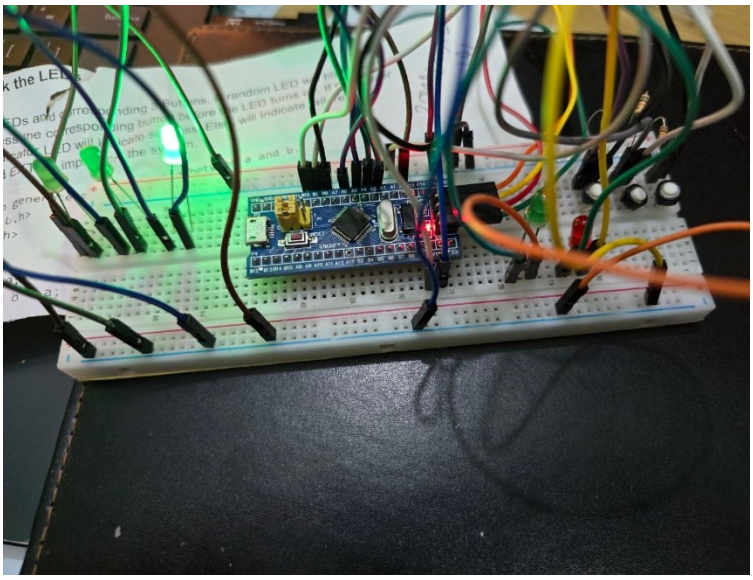
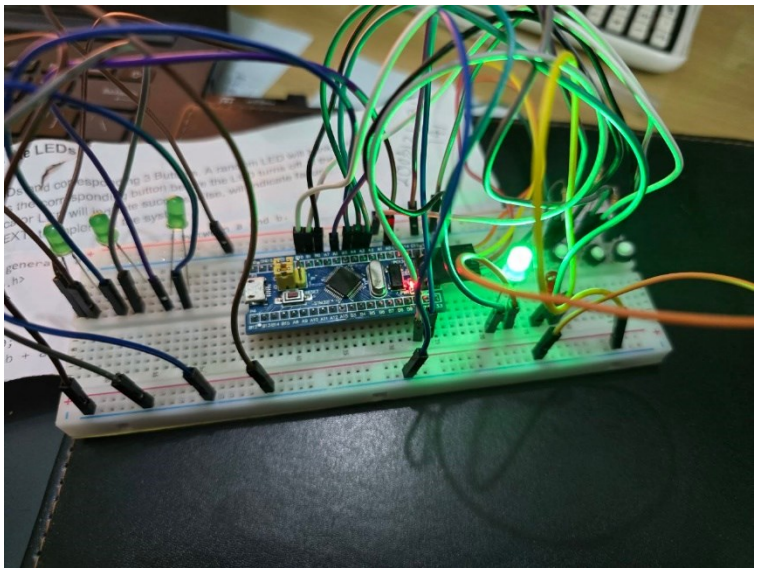


Fig: 2nd LED light up





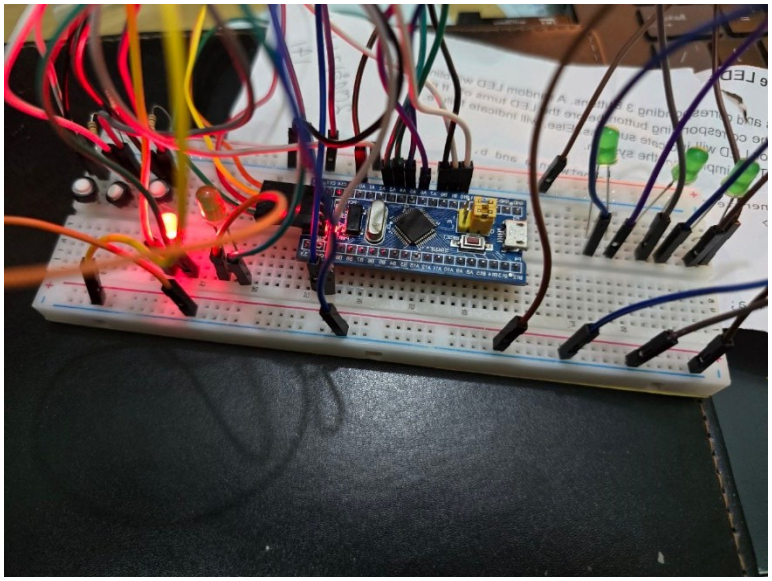


Fig: 3rd LED light up, Success LED light up,Failure LED light up

Code:

```
#include "stm32f10x.h"
#include <stdlib.h>
#include <time.h>

void SysTick_Handler(void);
void EXTI0_IRQHandler(void);
void EXTI1_IRQHandler(void);
void EXTI15_10_IRQHandler(void);
void delay_ms(uint32_t ms);
void init_LEDs(void);
void init_buttons(void);
void init_SysTick(void);
void init_EXTI(void);
uint8_t debounce_button(GPIO_TypeDef *GPIOx, uint16_t GPIO_Pin);

static volatile uint32_t msTicks = 0;
static volatile uint8_t buttonPressed = 0;
static volatile uint8_t currentLED = 0;

int main(void) {
    SystemInit();
    init_LEDs();
    init_buttons();
    init_SysTick();
    init_EXTI();

    srand(1294890499);

    while (1) {
        currentLED = rand() % 3;

        GPIOA->BSRR = GPIO_BSRR_BR0 | GPIO_BSRR_BR5 | GPIO_BSRR_BR6;

        if (currentLED == 0) {
            GPIOA->BSRR = GPIO_BSRR_BS0;
        } else if (currentLED == 1) {
            GPIOA->BSRR = GPIO_BSRR_BS5;
        } else if (currentLED == 2) {
            GPIOA->BSRR = GPIO_BSRR_BS6;
        }
        buttonPressed = 0;

        for (int i = 0; i < 3; ++i) {
            delay_ms(1000);

            if ((currentLED == 0 && debounce_button(GPIOB, 0x0001))
                (currentLED == 1 && debounce_button(GPIOB, 0x0002))
                (currentLED == 2 && debounce_button(GPIOB, 0x0400))) {
                buttonPressed = 1;
                break;
            }
        }

        if (currentLED == 0) {
            GPIOA->BSRR = GPIO_BSRR_BR0;
        } else if (currentLED == 1) {
            GPIOA->BSRR = GPIO_BSRR_BR5;
        } else if (currentLED == 2) {
            GPIOA->BSRR = GPIO_BSRR_BR6;
        }

        if (buttonPressed) {
            GPIOA->BSRR = GPIO_BSRR_BS3;
            delay_ms(2000);
```

```
            GPIOA->BSRR = GPIO_BSRR_BR3;
        } else {
            GPIOA->BSRR = GPIO_BSRR_BS4;
            delay_ms(2000);
            GPIOA->BSRR = GPIO_BSRR_BR4;
        }
    }
}

void SysTick_Handler(void) {
    msTicks++;
}

void EXTI0_IRQHandler(void) {
    if (EXTI->PR & EXTI_PR_PR0) {
        if (currentLED == 0) {
            buttonPressed = 1;
        }
        EXTI->PR = EXTI_PR_PR0;
    }
}
void EXTI1_IRQHandler(void) {
    if (EXTI->PR & EXTI_PR_PR1) {
        if (currentLED == 1) {
            buttonPressed = 1;
        }
        EXTI->PR = EXTI_PR_PR1;
    }
}

void EXTI15_10_IRQHandler(void) {
    if (EXTI->PR & EXTI_PR_PR10) {
        if (currentLED == 2) {
            buttonPressed = 1;
        }
        EXTI->PR = EXTI_PR_PR10;
    }
}

void delay_ms(uint32_t ms) {
    uint32_t target = msTicks + ms;
    while (msTicks < target);
}

void init_LEDs(void) {
    RCC->APB2ENR |= RCC_APB2ENR_IOPAEN;

    GPIOA->CRL &= ~((GPIO_CRL_MODE0 | GPIO_CRL_CNF0) |
                    (GPIO_CRL_MODE5 | GPIO_CRL_CNF5) |
                    (GPIO_CRL_MODE6 | GPIO_CRL_CNF6) |
                    (GPIO_CRL_MODE3 | GPIO_CRL_CNF3) |
                    (GPIO_CRL_MODE4 | GPIO_CRL_CNF4));

    GPIOA->CRL |= (GPIO_CRL_MODE0_1 | GPIO_CRL_MODE5_1 | GPIO_CRL_MODE6_1 |
                   GPIO_CRL_MODE3_1 | GPIO_CRL_MODE4_1);
}

void init_buttons(void) {
    RCC->APB2ENR |= RCC_APB2ENR_IOPBEN;

    GPIOB->CRL &= ~(GPIO_CRL_MODE0 | GPIO_CRL_CNF0);
    GPIOB->CRL |= GPIO_CRL_CNF0_1;
    GPIOB->ODR |= GPIO_ODR_ODR0;

    GPIOB->CRL &= ~(GPIO_CRL_MODE1 | GPIO_CRL_CNF1);
    GPIOB->CRL |= GPIO_CRL_CNF1_1;
    GPIOB->ODR |= GPIO_ODR_ODR1;

    GPIOB->CRH &= ~(GPIO_CRH_MODE10 | GPIO_CRH_CNF10);
    GPIOB->CRH |= GPIO_CRH_CNF10_1;
    GPIOB->ODR |= GPIO_ODR_ODR10;
}

void init_SysTick(void) {
```

```
    SysTick_Config(SystemCoreClock / 1000);
}

void init_EXTI(void) {
    RCC->APB2ENR |= RCC_APB2ENR_AFIOEN;

    AFIO->EXTICR[0] &= ~(AFIO_EXTICR1_EXTI0 | AFIO_EXTICR1_EXTI1);
    AFIO->EXTICR[0] |= (0x01 << 0) | (0x01 << 4);
    AFIO->EXTICR[2] &= ~(AFIO_EXTICR3_EXTI10);
    AFIO->EXTICR[2] |= (0x01 << 8);

    EXTI->IMR |= (EXTI_IMR_MR0 | EXTI_IMR_MR1 | EXTI_IMR_MR10);
    EXTI->FTSR |= (EXTI_FTSR_TR0 | EXTI_FTSR_TR1 | EXTI_FTSR_TR10);

    NVIC_EnableIRQ(EXTI0_IRQn);
    NVIC_SetPriority(EXTI0_IRQn, 0);

    NVIC_EnableIRQ(EXTI1_IRQn);
    NVIC_SetPriority(EXTI1_IRQn, 0);

    NVIC_EnableIRQ(EXTI15_10_IRQn);
    NVIC_SetPriority(EXTI15_10_IRQn, 0);
}

uint8_t debounce_button(GPIO_TypeDef *GPIOx, uint16_t GPIO_Pin) {
    if (!(GPIOx->IDR & GPIO_Pin)) {
        delay_ms(50);
        if (!(GPIOx->IDR & GPIO_Pin)) {
            return 1;
        }
    }
    return 0;
}
```

Conclusion:

The "Whack the LEDs" project shows how to use SysTick and EXTI peripherals to integrate buttons and LEDs with a microcontroller. LEDs show if the system was successful or unsuccessful when it successfully creates random LED patterns and assesses button presses. This project improves knowledge of timer-based job scheduling and interrupt-based I/O management in embedded systems.
Additional enhancements can include more LEDs and buttons for a more complex gaming experience, scoring systems, or difficulty levels.

Area of contribution of each member:

Logic Build Up & Coding Implementation:

Sefayet Alam(2003121) & Nahid Hasan(2003131)

Circuit Configuration:

Md. Wakilul Arifin Akash(2003141)