Electrical Electronics Engineering

Microprocessor Systems

Experiment: 2

Group: 21

Students: Erkan Tiryakioğlu - 150720051

Alperen Yıldız - 150720067

Baran Orman - 150721063

TASKS

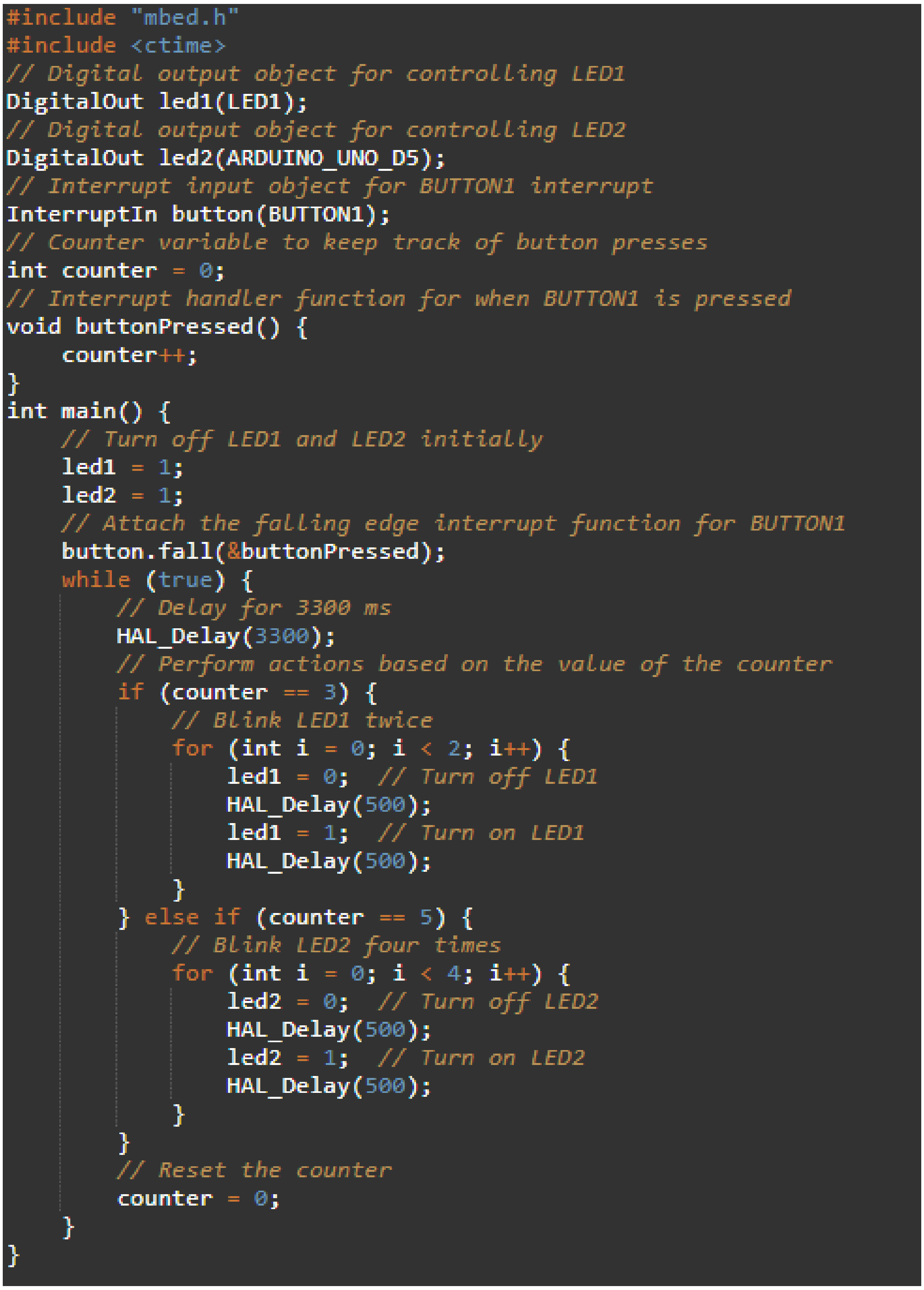
PART A:

- Use a single push button as an interrupt input, and two LEDs as outputs.
- Initially both LEDs are on.
- Count button press in a definite time interval (e.g. 2s).
- If count is 3, LED1 blinks 2 times.
- If count is 5, LED2 blinks4 times.
- This procedure goes indefinitely.

PART B:

- Use 3 LEDs and 2 push buttons (as external interrupts (e.g. pins B13,B14).) will be used.
- Initially both LEDs are off.
- Within a 10 second period;
  - If button 1 is pressed >3 times, led 1 toggles.
  - If button 2 is pressed >5 times, led 2 toggles.
  - Third LED can be used to indicate 20s period.
- This procedure goes indefinitely.

A)



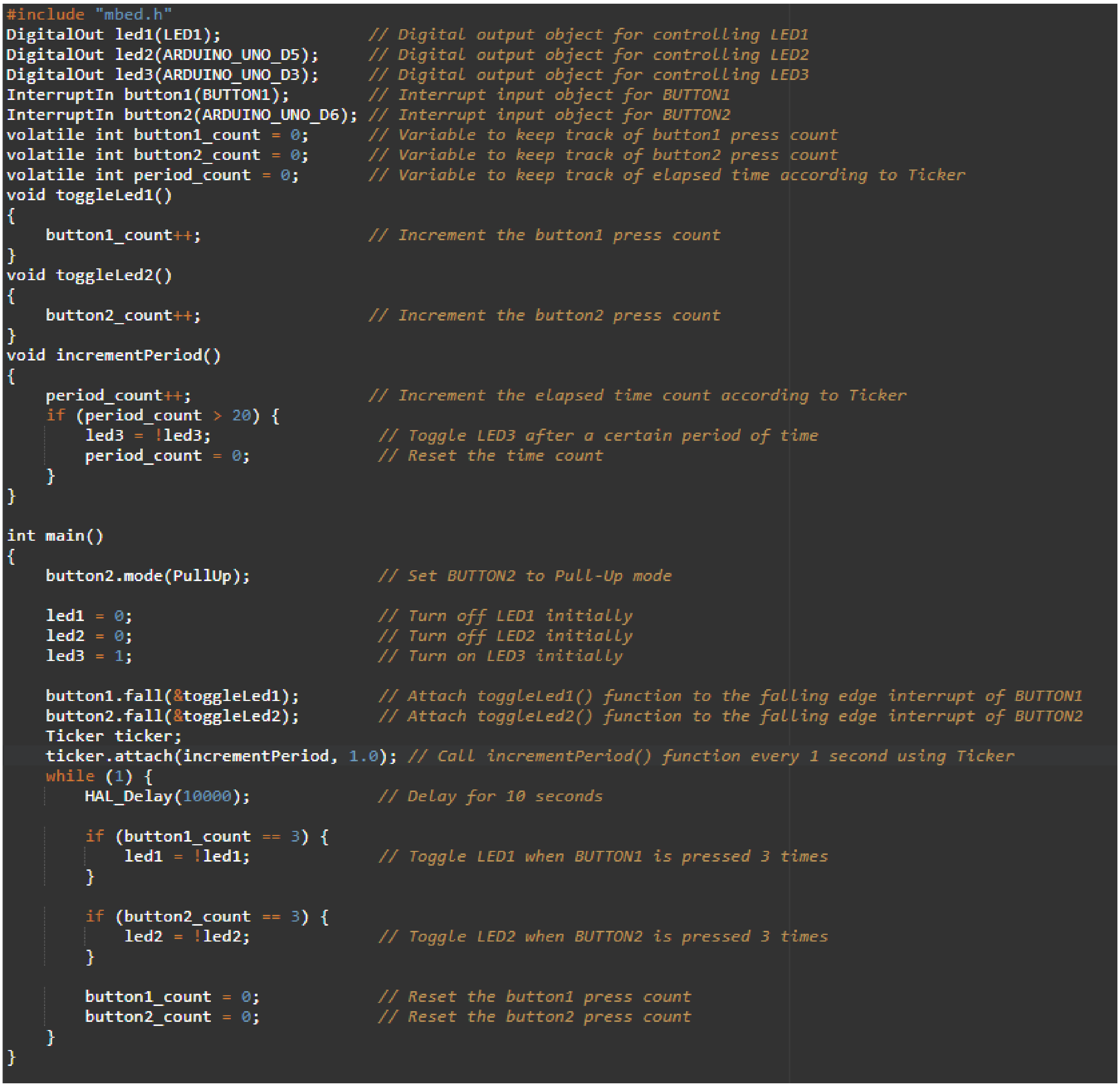
```cpp
#include "mbed.h"
#include <ctime>
// Digital output object for controlling LED1
DigitalOut led1(LED1);
// Digital output object for controlling LED2
DigitalOut led2(ARDUINO_UNO_D5);
// Interrupt input object for BUTTON1 interrupt
InterruptIn button(BUTTON1);
// Counter variable to keep track of button presses
int counter = 0;
// Interrupt handler function for when BUTTON1 is pressed
void buttonPressed() {
    counter++;
}
int main() {
    // Turn off LED1 and LED2 initially
    led1 = 1;
    led2 = 1;
    // Attach the falling edge interrupt function for BUTTON1
    button.fall(&buttonPressed);
    while (true) {
        // Delay for 3300 ms
        HAL_Delay(3300);
        // Perform actions based on the value of the counter
        if (counter == 3) {
            // Blink LED1 twice
            for (int i = 0; i < 2; i++) {
                led1 = 0;  // Turn off LED1
                HAL_Delay(500);
                led1 = 1;  // Turn on LED1
                HAL_Delay(500);
            }
        } else if (counter == 5) {
            // Blink LED2 four times
            for (int i = 0; i < 4; i++) {
                led2 = 0;  // Turn off LED2
                HAL_Delay(500);
                led2 = 1;  // Turn on LED2
                HAL_Delay(500);
            }
        }
        // Reset the counter
        counter = 0;
    }
}
```

This code is designed to control LEDs and a button in a microcontroller-based system running on the ARM mbed platform. Let's explore its functionality and operation:

- Firstly, the code includes the "mbed.h" library and the <ctime> header file.
- Next, it creates digital output objects (**DigitalOut**) for controlling LED1 and LED2, and an interrupt input object (**InterruptIn**) for BUTTON1.
- The **counter** variable is used to keep track of the number of times the button has been pressed.
- The **buttonPressed()** function serves as an interrupt handler triggered by the falling edge interrupt of BUTTON1. Every time the interrupt occurs, the **counter** value is incremented.
- In the **main()** function, LED1 and LED2 are initially turned off (**led1 = 1;** and **led2 = 1;**).
- The line **button.fall(&buttonPressed);** associates the falling edge interrupt of BUTTON1 with the **buttonPressed()** function. This means that whenever BUTTON1 is pressed, the **buttonPressed()** function will execute and increment the **counter** value.
- The program runs inside an infinite loop.
- The line **HAL_Delay(3300);** introduces a delay of 3300 milliseconds.
- Actions are performed based on the value of the **counter**:
    - If the **counter** value is 3, LED1 blinks twice (**led1 = 0;**, **HAL_Delay(500);**, **led1 = 1;**, **HAL_Delay(500);**).
    - If the **counter** value is 5, LED2 blinks four times (**led2 = 0;**, **HAL_Delay(500);**, **led2 = 1;**, **HAL_Delay(500);**).
- After the actions are completed, the **counter** is reset to 0.
- This loop continues as long as BUTTON1 is pressed, incrementing the **counter** value, and providing feedback by blinking the LEDs according to specific patterns.

In this way, the code controls the LEDs and responds to a specific number of button presses, providing visual feedback through LED blinking.

B)



```cpp
#include "mbed.h"
DigitalOut led1(LED1);                 // Digital output object for controlling LED1
DigitalOut led2(ARDUINO_UNO_D5);       // Digital output object for controlling LED2
DigitalOut led3(ARDUINO_UNO_D3);       // Digital output object for controlling LED3
InterruptIn button1(BUTTON1);          // Interrupt input object for BUTTON1
InterruptIn button2(ARDUINO_UNO_D6);   // Interrupt input object for BUTTON2
volatile int button1_count = 0;        // Variable to keep track of button1 press count
volatile int button2_count = 0;        // Variable to keep track of button2 press count
volatile int period_count = 0;         // Variable to keep track of elapsed time according to Ticker
void toggleLed1()
{
    button1_count++;                   // Increment the button1 press count
}
void toggleLed2()
{
    button2_count++;                   // Increment the button2 press count
}
void incrementPeriod()
{
    period_count++;                    // Increment the elapsed time count according to Ticker
    if (period_count > 20) {
        led3 = !led3;                   // Toggle LED3 after a certain period of time
        period_count = 0;               // Reset the time count
    }
}

int main()
{
    button2.mode(PullUp);               // Set BUTTON2 to Pull-Up mode

    led1 = 0;                           // Turn off LED1 initially
    led2 = 0;                           // Turn off LED2 initially
    led3 = 1;                           // Turn on LED3 initially

    button1.fall(&toggleLed1);          // Attach toggleLed1() function to the falling edge interrupt of BUTTON1
    button2.fall(&toggleLed2);          // Attach toggleLed2() function to the falling edge interrupt of BUTTON2
    Ticker ticker;
    ticker.attach(incrementPeriod, 1.0); // Call incrementPeriod() function every 1 second using Ticker
    while (1) {
        HAL_Delay(10000);               // Delay for 10 seconds

        if (button1_count == 3) {
            led1 = !led1;               // Toggle LED1 when BUTTON1 is pressed 3 times
        }

        if (button2_count == 3) {
            led2 = !led2;               // Toggle LED2 when BUTTON2 is pressed 3 times
        }

        button1_count = 0;              // Reset the button1 press count
        button2_count = 0;              // Reset the button2 press count
    }
}
```

This code is designed to control LEDs and buttons in a microcontroller-based system using the ARM mbed platform. Let's discuss its purpose and how it operates:

The code initializes three LEDs, namely LED1, LED2, and LED3, along with two buttons, BUTTON1 and BUTTON2. It also defines variables to keep track of the number of times each button is pressed (**button1_count** and **button2_count**), and a counter for tracking elapsed time (**period_count**).

The code sets up interrupt handlers for BUTTON1 and BUTTON2. When BUTTON1 is pressed, the **toggleLed1()** function is called, incrementing **button1_count**. Similarly, pressing BUTTON2 triggers the **toggleLed2()** function, incrementing **button2_count**. Additionally, a Ticker is used to call the **incrementPeriod()** function every second, which increments **period_count**. After a certain period (20 ticks in this case), LED3 is toggled.

In the **main()** function, BUTTON2 is configured in Pull-Up mode. LED1 and LED2 are initially turned off, while LED3 is turned on. The **toggleLed1()** and **toggleLed2()** functions are attached to the falling edge interrupts of BUTTON1 and BUTTON2, respectively. The **incrementPeriod()** function is scheduled to be called every second using the Ticker.

Inside the infinite **while** loop, there is a 10-second delay. After the delay, the code checks the button press counts. If BUTTON1 has been pressed 3 times, LED1 is toggled. Likewise, if BUTTON2 has been pressed 3 times, LED2 is toggled. Finally, the button press counts are reset.

This process continues indefinitely, with the code responding to button presses and providing visual feedback by toggling the LEDs accordingly. Additionally, LED3 toggles after a certain period, giving a time-based indication. Overall, this code demonstrates basic input and output control using interrupts and timers on the ARM mbed platform.