Lab 2

G10 Daniel Dsa EE23DP003
Pradeep Kumar M EE22DP007

Objective:

Learn how to handle digital inputs through GPIO.

PART 1:

- Read the state of one of the switches (either SW1 or SW2) connected to the GPIO pins of the Tiva chip.
- Whenever the button is pressed, the LED should light up RED.
- Whenever the button is not pressed, the LED should be off.

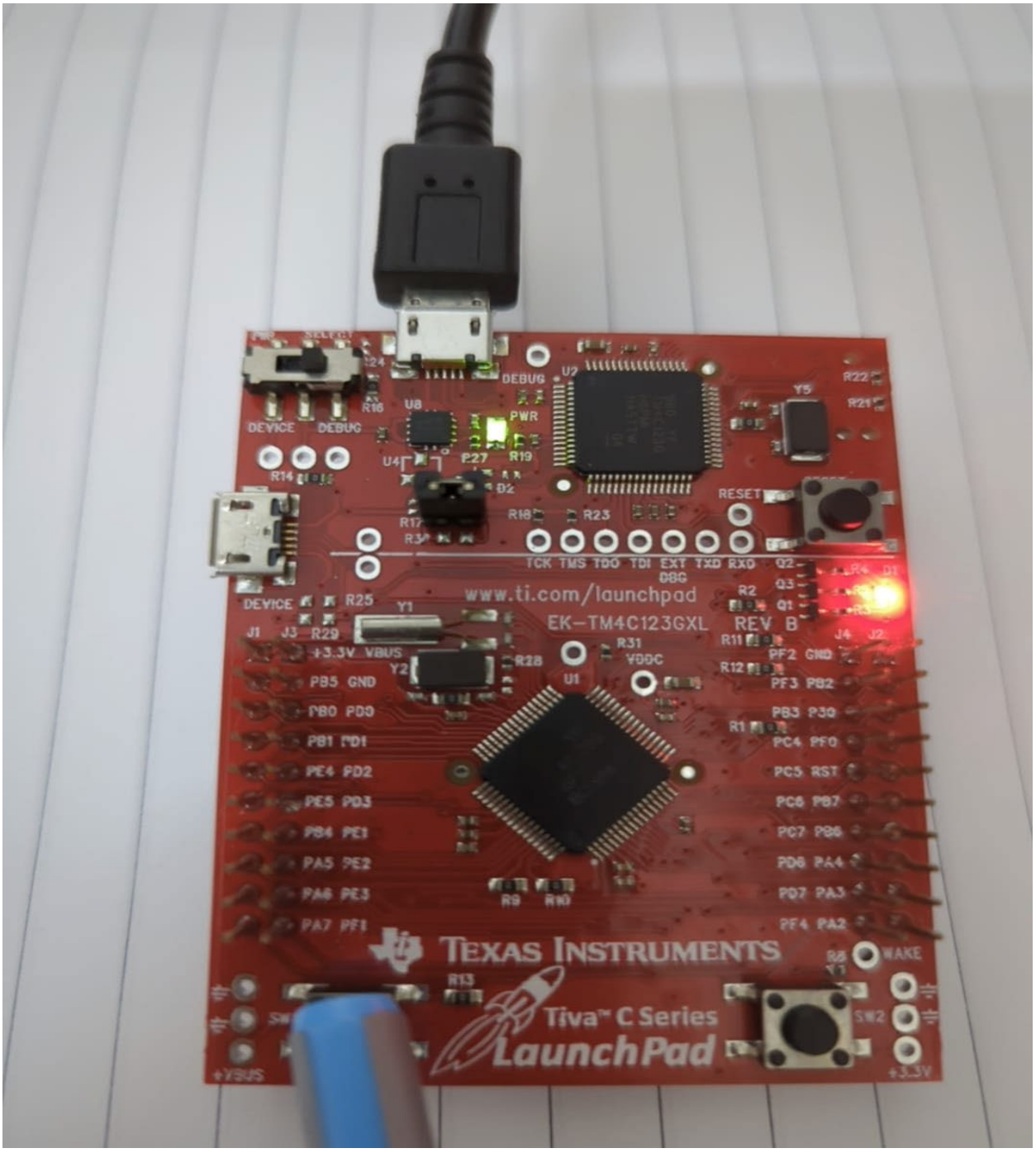
Code:

```
#include "tm4c123gh6pm.h"   // Include the header file for the TM4C123GH6PM board
#include <stdint.h>        // Include the standard integer types header file

#define LED_RED (1U << 1) // macro for red LED, which is connected to pin 1 of port F
#define SW1 (1U<<4)      // macro for switch 1, which is connected to pin 4 of port F

int main(void)
{
    int state;                          // variable to store the state of the switch
    SYSCTL_RCGCGPIO_R |= (1U << 5);   // Enable clock for GPIO port F
    GPIO_PORTF_LOCK_R = 0x4C4F434B;  // Unlock GPIO port F
    GPIO_PORTF_CR_R = 0x01;         // Allow changes to PF0
    GPIO_PORTF_DIR_R |= LED_RED;   // Set the direction of the red LED pin as output
    GPIO_PORTF_DIR_R &= ~SW1;     // Set the direction of switch 1 pin as input
    GPIO_PORTF_DEN_R = 0x1F;     // Enable digital function for pins PF0-PF4
    GPIO_PORTF_PUR_R = 0x11;    // Enable pull-up resistors for switch 1 and switch 2
    while (1)                  // Infinite loop
    {
        state = GPIO_PORTF_DATA_R & 0x10;       // Read the state of switch 1
        if (state == 0x00)                     // If switch 1 is pressed (active low)
        {
            GPIO_PORTF_DATA_R |= LED_RED;     // Turn on the red LED
        }
        else
        {
            GPIO_PORTF_DATA_R &= ~LED_RED;   // Turn off the red LED
        }
    }
}
```

Result:



Red LED turns on when push button SW1 is pressed and turns off when SW1 is released.

PART 2:

- Read the state of one of the switches (either SW1 or SW2) connected to the GPIO pins of the Tiva chip.
- Upon each button is press, the LED color should change sequentially (Red -> Blue -> Green -> Red -> Blue...)
- Make sure to implement some debouncing so that each button press causes exactly one color change.

Code:

```
#include "tm4c123gh6pm.h" // Include the header file for the TM4C123GH6PM board
#include <stdint.h>       // Include the standard integer types header file

void PortF_Init(void);  // Port F initialization function declaration
void Delay(void);      // Debounce delay function declaration

int main(void)
{
    PortF_Init();             // Initialize Port F
    int lastButtonState = 1; // Variable to store the last state of the button
    while (1)               // Infinite loop
     {
        int buttonState = (GPIO_PORTF_DATA_R & 0x10) >> 4; // Read the current state
of the button SW1
        if (buttonState == 0 && lastButtonState == 1) // If the button is pressed and
was not pressed in the last iteration
        {
            Delay();                                    // Debounce the button press
            if ((GPIO_PORTF_DATA_R & 0x02) == 0x02)     // If red LED is on
            {
                GPIO_PORTF_DATA_R = 0x04;              // Turn on blue LED
            }
            else if ((GPIO_PORTF_DATA_R & 0x04) == 0x04) // If blue LED is on
            {
                GPIO_PORTF_DATA_R = 0x08;               // Turn on green LED
            } else
            {
                GPIO_PORTF_DATA_R = 0x02;              // Turn on red LED
            }
        }
        lastButtonState = buttonState; // Update the last state of the button
    }
}

void PortF_Init(void)
{
    SYSCTL_RCGC2_R |= 0x00000020;    // Enable clock for Port F
    GPIO_PORTF_LOCK_R = 0x4C4F434B; // Unlock Port F
    GPIO_PORTF_CR_R = 0x1F;        // Allow changes to PF4-0
    GPIO_PORTF_DIR_R = 0x0E;      // Set PF3-1 as output, PF4 and PF0 as input
    GPIO_PORTF_PUR_R = 0x11;     // Enable pull-up resistors for PF4 and PF0
```

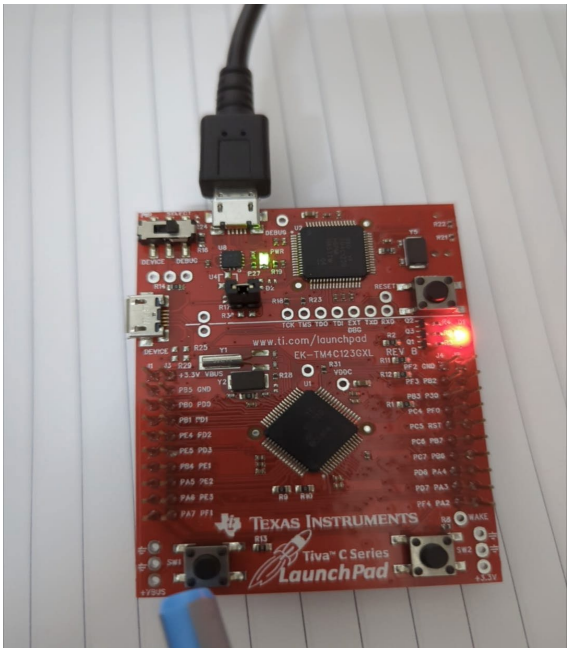
```
    GPIO_PORTF_DEN_R = 0x1F;    // Enable digital function for PF4-0
}

void Delay(void)          // Delay to implement debounce
{
    int time = 1000000; // Variable for delay
    while (time)
    {
        time--;        // Decrement time until it reaches zero
    }
}
```

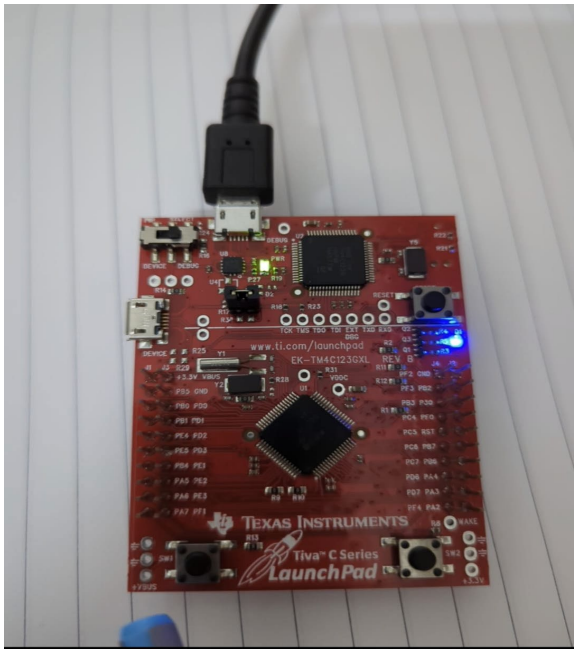
Explanation:

- The code reads the state of the onboard push-button switch SW1 and toggles the LEDs in sequence (Red → Blue → Green → Red→…) if the switch is pressed.
- The PortF_Init function initializes Port F by enabling its clock, unlocking it, allowing changes to its pins, setting its direction, enabling pull-up resistors, and enabling its digital function.
- Switch debounce is also implemented through a Delay function. The Delay function creates a delay by decrementing a variable until it reaches zero.
- In the main function, the current state of the button is read and compared to its previous state.
- If the button is pressed and was not pressed in the previous iteration, the Delay function is called to debounce the button press. Then, depending on which LED is currently on, a different LED is turned on. This process repeats indefinitely.

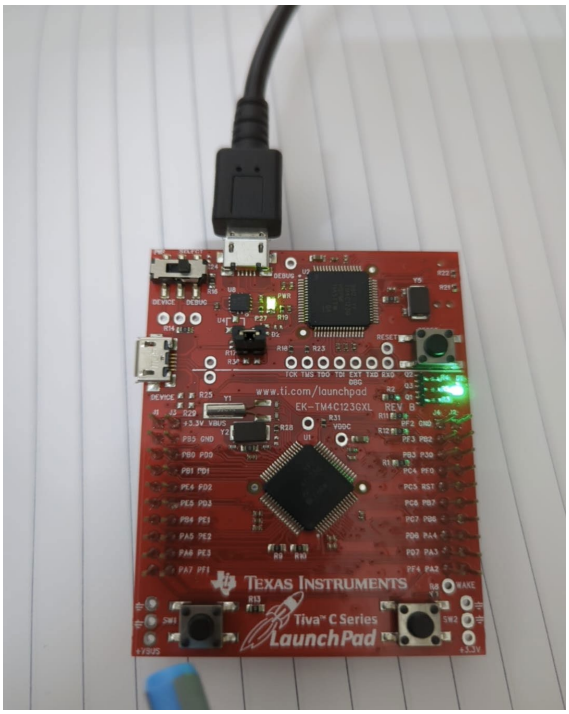
Result:



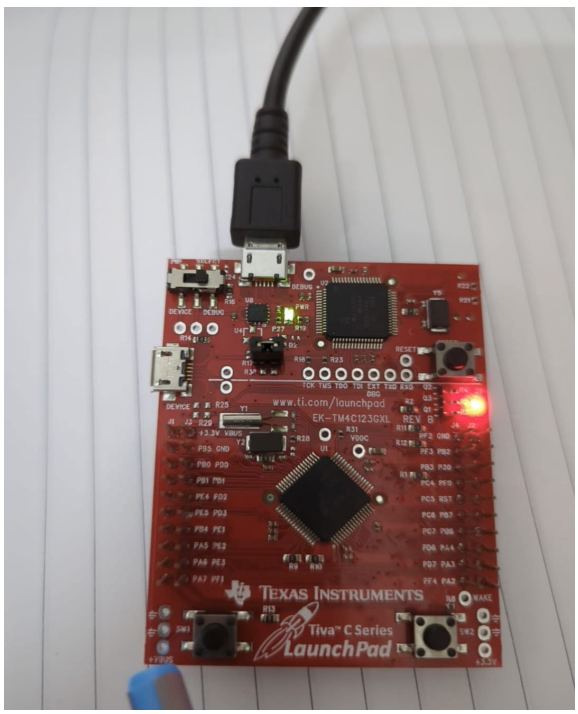
1st press of SW1 turns on Red LED



2nd Press of SW1 turns on Blue LED



3rd press of SW1 turns on Green LED



4th press of SW1 turns on Red LED again