13-12-2024-DAILY ASSIGNMENTS

ANSU MARIUM SHIBU

1.Turn LEDs ON/OFF: Write a program to control 4 LEDs connected to GPIO pins. Implement a function void controlLED( ) that turns the specified LED ON (true) or OFF (false).

```
#include "stm32f407xx.h"
#include<stdbool.h>

// Delay function
void delay() {
    for (uint32_t i = 0; i < 500000; i++);
}

// Control LED ON/OFF
void controlLED(uint8_t PinNumber, bool state) {
    if (state) {
        GPIO_WriteToOutputPin(GPIOD, PinNumber, GPIO_PIN_SET);
    } else {
        GPIO_WriteToOutputPin(GPIOD, PinNumber, GPIO_PIN_RESET);
    }
}

int main(void) {
    GPIO_Handle_t GPIOLed;
    GPIOLed.pGPIOx = GPIOD;

    // Initialize all LEDs
    for (uint8_t i = 0; i < 4; i++) {
        GPIOLed.GPIO_PinConfig.GPIO_PinNumber = GPIO_PIN_NO_12 + i;
        GPIOLed.GPIO_PinConfig.GPIO_PinMode = GPIO_MODE_OUT;
```

```
    GPIOLed.GPIO_PinConfig.GPIO_PinSpeed = GPIO_SPEED_FAST;
    GPIOLed.GPIO_PinConfig.GPIO_PinOPType = GPIO_OP_TYPE_PP;
    GPIOLed.GPIO_PinConfig.GPIO_PinPuPdCOntrol = GPIO_NO_PUPD;
    GPIO_Init(&GPIOLed);
  }

  // Turn LED1 ON, LED2 OFF
  controlLED(GPIO_PIN_NO_12, true);
  controlLED(GPIO_PIN_NO_13, false);
  delay();
  // Turn LED1 OFF, LED2 ON
  controlLED(GPIO_PIN_NO_12, false);
  controlLED(GPIO_PIN_NO_13, true);
  delay();

  while(1);
}
```

2. Blink LEDs in Sequence: Write a program that blinks the 4 LEDs in sequence (LED1 -> LED2 -> LED3 -> LED4) with a delay between each. After LED4, the sequence should repeat.

```
#include "stm32f407xx.h"
#include<stdbool.h>

// Delay function
void delay() {
  for (uint32_t i = 0; i < 500000; i++);
}

// Control LED ON/OFF
void controlLED(uint8_t PinNumber, bool state) {
  if (state) {
    GPIO_WriteToOutputPin(GPIOD, PinNumber, GPIO_PIN_SET);
```

```
    } else {
        GPIO_WriteToOutputPin(GPIOD, PinNumber, GPIO_PIN_RESET);
    }
}

void blinkLEDsInSequence() {
    for (uint8_t i = 0; i < 4; i++) {
        controlLED(GPIO_PIN_NO_12 + i, true);  // Turn ON the current LED
        delay();
        controlLED(GPIO_PIN_NO_12 + i, false); // Turn OFF the current LED
        delay();
    }
}

int main(void) {
    GPIO_Handle_t GPIOLed;
    GPIOLed.pGPIOx = GPIOD;

    // Initialize all LEDs
    for (uint8_t i = 0; i < 4; i++) {
        GPIOLed.GPIO_PinConfig.GPIO_PinNumber = GPIO_PIN_NO_12 + i;
        GPIOLed.GPIO_PinConfig.GPIO_PinMode = GPIO_MODE_OUT;
        GPIOLed.GPIO_PinConfig.GPIO_PinSpeed = GPIO_SPEED_FAST;
        GPIOLed.GPIO_PinConfig.GPIO_PinOPType = GPIO_OP_TYPE_PP;
        GPIOLed.GPIO_PinConfig.GPIO_PinPuPdCOntrol = GPIO_NO_PUPD;
        GPIO_Init(&GPIOLed);
    }

    while(1) {
        blinkLEDsInSequence();
    }
```

```
}
```

3. Binary Counter with LEDs: Implement a binary counter using the 4 LEDs. Starting from 0000 (all OFF), increment the count every second, displaying the binary representation of the counter on the LEDs (ON = 1, OFF = 0).

```
#include "stm32f407xx.h"


// Delay function
void delay() {
    for (uint32_t i = 0; i < 500000; i++);
}


// Control LED ON/OFF
void controlLED(uint8_t PinNumber, bool state) {
    if (state) {
        GPIO_WriteToOutputPin(GPIOD, PinNumber, GPIO_PIN_SET);
    } else {
        GPIO_WriteToOutputPin(GPIOD, PinNumber, GPIO_PIN_RESET);
    }
}


void binaryCounter() {
    uint8_t counter = 0;
    while (1) {
        for (uint8_t i = 0; i < 4; i++) {
            if ((counter >> i) & 1) {
                controlLED(GPIO_PIN_NO_12 + i, true); // LED ON for 1
            } else {
                controlLED(GPIO_PIN_NO_12 + i, false); // LED OFF for 0
            }
        }
        delay();
```

```
        counter++; // Increment the counter
    }
}

int main(void) {
    GPIO_Handle_t GPIOLed;
    GPIOLed.pGPIOx = GPIOD;

    // Initialize all LEDs
    for (uint8_t i = 0; i < 4; i++) {
        GPIOLed.GPIO_PinConfig.GPIO_PinNumber = GPIO_PIN_NO_12 + i;
        GPIOLed.GPIO_PinConfig.GPIO_PinMode = GPIO_MODE_OUT;
        GPIOLed.GPIO_PinConfig.GPIO_PinSpeed = GPIO_SPEED_FAST;
        GPIOLed.GPIO_PinConfig.GPIO_PinOPType = GPIO_OP_TYPE_PP;
        GPIOLed.GPIO_PinConfig.GPIO_PinPuPdCOntrol = GPIO_NO_PUPD;
        GPIO_Init(&GPIOLed);
    }

    binaryCounter();
}
```

4. Alternate Blinking: Create a program that makes LED1 and LED3 blink alternately with LED2 and LED4, each group toggling every second.

```
#include "stm32f407xx.h"
#include<stdbool.h>

// Delay function
void delay() {
    for (uint32_t i = 0; i < 500000; i++);
}

// Control LED ON/OFF
```

```
void controlLED(uint8_t PinNumber, bool state) {
    if (state) {
        GPIO_WriteToOutputPin(GPIOD, PinNumber, GPIO_PIN_SET);
    } else {
        GPIO_WriteToOutputPin(GPIOD, PinNumber, GPIO_PIN_RESET);
    }
}

void alternateBlinking() {
    while (1) {
        // Group 1 (LED1 and LED3)
        controlLED(GPIO_PIN_NO_12, true);  // LED1 ON
        controlLED(GPIO_PIN_NO_14, true);  // LED3 ON
        delay();
        controlLED(GPIO_PIN_NO_12, false); // LED1 OFF
        controlLED(GPIO_PIN_NO_14, false); // LED3 OFF

        // Group 2 (LED2 and LED4)
        controlLED(GPIO_PIN_NO_13, true);  // LED2 ON
        controlLED(GPIO_PIN_NO_15, true);  // LED4 ON
        delay();
        controlLED(GPIO_PIN_NO_13, false); // LED2 OFF
        controlLED(GPIO_PIN_NO_15, false); // LED4 OFF
    }
}

int main(void) {
    GPIO_Handle_t GPIOLed;
    GPIOLed.pGPIOx = GPIOD;

    // Initialize all LEDs
```

```
    for (uint8_t i = 0; i < 4; i++) {
        GPIOLed.GPIO_PinConfig.GPIO_PinNumber = GPIO_PIN_NO_12 + i;
        GPIOLed.GPIO_PinConfig.GPIO_PinMode = GPIO_MODE_OUT;
        GPIOLed.GPIO_PinConfig.GPIO_PinSpeed = GPIO_SPEED_FAST;
        GPIOLed.GPIO_PinConfig.GPIO_PinOPType = GPIO_OP_TYPE_PP;
        GPIOLed.GPIO_PinConfig.GPIO_PinPuPdCOntrol = GPIO_NO_PUPD;
        GPIO_Init(&GPIOLed);
    }

    alternateBlinking();
}
```

5.Traffic Light Simulation: Simulate a traffic light system using the 4 LEDs. Assign them as Red, Yellow, Green, and a Pedestrian light. Use appropriate timing sequences to mimic real-world behavior.

```
#include "stm32f407xx.h"

void delay(uint32_t delay_time);

int main(void)
{
    GPIO_Handle_t GPIOLed;
    GPIOLed.pGPIOx = GPIOD;

    // Enable the clock for GPIOD Peripheral
    GPIO_PeriClockControl(GPIOD, ENABLE);

    // Initialize all 4 LEDs (PD12, PD13, PD14, PD15)
    for (uint8_t i = 0; i < 4; i++) {
        GPIOLed.GPIO_PinConfig.GPIO_PinNumber = GPIO_PIN_NO_12 + i;
        GPIOLed.GPIO_PinConfig.GPIO_PinMode = GPIO_MODE_OUT;
        GPIOLed.GPIO_PinConfig.GPIO_PinSpeed = GPIO_SPEED_FAST;
```

```
    GPIOLed.GPIO_PinConfig.GPIO_PinOPType = GPIO_OP_TYPE_PP;
    GPIOLed.GPIO_PinConfig.GPIO_PinPuPdCOntrol = GPIO_NO_PUPD;
    GPIO_Init(&GPIOLed);
}

while (1) {
    // Red light ON, Yellow and Green OFF, Pedestrian light OFF
    GPIO_WriteToOutputPin(GPIOD, GPIO_PIN_NO_12, GPIO_PIN_SET); // Red ON
    GPIO_WriteToOutputPin(GPIOD, GPIO_PIN_NO_13, GPIO_PIN_RESET); // Yellow OFF
    GPIO_WriteToOutputPin(GPIOD, GPIO_PIN_NO_14, GPIO_PIN_RESET); // Green OFF
    GPIO_WriteToOutputPin(GPIOD, GPIO_PIN_NO_15, GPIO_PIN_RESET); // Pedestrian OFF
    delay(5000000); // Red light duration

    // Yellow light ON, Red and Green OFF, Pedestrian light OFF
    GPIO_WriteToOutputPin(GPIOD, GPIO_PIN_NO_12, GPIO_PIN_RESET); // Red OFF
    GPIO_WriteToOutputPin(GPIOD, GPIO_PIN_NO_13, GPIO_PIN_SET); // Yellow ON
    GPIO_WriteToOutputPin(GPIOD, GPIO_PIN_NO_14, GPIO_PIN_RESET); // Green OFF
    GPIO_WriteToOutputPin(GPIOD, GPIO_PIN_NO_15, GPIO_PIN_RESET); // Pedestrian OFF
    delay(2000000); // Yellow light duration

    // Green light ON, Red and Yellow OFF, Pedestrian light OFF
    GPIO_WriteToOutputPin(GPIOD, GPIO_PIN_NO_12, GPIO_PIN_RESET); // Red OFF
    GPIO_WriteToOutputPin(GPIOD, GPIO_PIN_NO_13, GPIO_PIN_RESET); // Yellow OFF
    GPIO_WriteToOutputPin(GPIOD, GPIO_PIN_NO_14, GPIO_PIN_SET); // Green ON
    GPIO_WriteToOutputPin(GPIOD, GPIO_PIN_NO_15, GPIO_PIN_RESET); // Pedestrian OFF
    delay(5000000); // Green light duration

    // Pedestrian light ON, others OFF
    GPIO_WriteToOutputPin(GPIOD, GPIO_PIN_NO_12, GPIO_PIN_RESET); // Red OFF
    GPIO_WriteToOutputPin(GPIOD, GPIO_PIN_NO_13, GPIO_PIN_RESET); // Yellow OFF
    GPIO_WriteToOutputPin(GPIOD, GPIO_PIN_NO_14, GPIO_PIN_RESET); // Green OFF
```

```
        GPIO_WriteToOutputPin(GPIOD, GPIO_PIN_NO_15, GPIO_PIN_SET); // Pedestrian ON
        delay(3000000); // Pedestrian light duration
    }
}

void delay(uint32_t delay_time) {
    for (uint32_t i = 0; i < delay_time; i++);
}
```

6. LED Pattern Generator: Allow the user to define custom ON/OFF patterns for the 4 LEDs via an array. For example, the input [1, 0, 1, 0] should turn LED1 and LED3 ON, and LED2 and LED4 OFF.

```
#include "stm32f407xx.h"

void delay(void);
void controlLED(uint8_t led_num, uint8_t state);

int main(void)
{
    GPIO_Handle_t GPIOLed;
    GPIOLed.pGPIOx = GPIOD;

    // Enable the clock for GPIOD Peripheral
    GPIO_PeriClockControl(GPIOD, ENABLE);

    // Initialize all 4 LEDs (PD12, PD13, PD14, PD15)
    for (uint8_t i = 0; i < 4; i++) {
        GPIOLed.GPIO_PinConfig.GPIO_PinNumber = GPIO_PIN_NO_12 + i;
        GPIOLed.GPIO_PinConfig.GPIO_PinMode = GPIO_MODE_OUT;
        GPIOLed.GPIO_PinConfig.GPIO_PinSpeed = GPIO_SPEED_FAST;
        GPIOLed.GPIO_PinConfig.GPIO_PinOPType = GPIO_OP_TYPE_PP;
```

```
    GPIOLed.GPIO_PinConfig.GPIO_PinPuPdCOntrol = GPIO_NO_PUPD;
    GPIO_Init(&GPIOLed);
  }

  // Define the pattern for LEDs
  uint8_t led_pattern[4] = {1, 0, 1, 0};  // LED1 and LED3 ON, LED2 and LED4 OFF

  while (1) {
    // Apply the pattern
    for (uint8_t i = 0; i < 4; i++) {
      controlLED(i, led_pattern[i]);
    }

    delay();
  }
}

void controlLED(uint8_t led_num, uint8_t state) {
  if (state == 1) {
    GPIO_WriteToOutputPin(GPIOD, GPIO_PIN_NO_12 + led_num, GPIO_PIN_SET); // LED ON
  } else {
    GPIO_WriteToOutputPin(GPIOD, GPIO_PIN_NO_12 + led_num, GPIO_PIN_RESET); // LED OFF
  }
}

void delay(void) {
  for (uint32_t i = 0; i < 500000; i++);
}
```

7. LED Knight Rider Effect:

Write a program to create a "Knight Rider" effect with 4 LEDs. The pattern should move from LED1 to LED4 and then reverse back to LED1, with a delay of between transitions.

```
#include "stm32f407xx.h"
#include <stdio.h>
#include <unistd.h>  // For sleep function

void delay(void);
void KnightRiderEffect(void);

#if !defined(__SOFT_FP__) && defined(__ARM_FP)
  #warning "FPU is not initialized, but the project is compiling for an FPU. Please initialize the FPU before use."
#endif

int main(void)
{
   GPIO_Handle_t GPIOLed;
   GPIOLed.pGPIOx = GPIOD;

   // Task: Toggle green LED connected to PD12.
   GPIOLed.GPIO_PinConfig.GPIO_PinNumber = GPIO_PIN_NO_12;
   GPIOLed.GPIO_PinConfig.GPIO_PinMode = GPIO_MODE_OUT;
   GPIOLed.GPIO_PinConfig.GPIO_PinSpeed = GPIO_SPEED_FAST;
   GPIOLed.GPIO_PinConfig.GPIO_PinOPType = GPIO_OP_TYPE_PP;
   GPIOLed.GPIO_PinConfig.GPIO_PinPuPdCOntrol = GPIO_NO_PUPD;

   // ENABLE THE CLOCK FOR GPIOD PERIPHERAL
   GPIO_PeriClockControl(GPIOD, ENABLE);
   // INITIALIZING THE GPIO PERIPHERAL
   GPIO_Init(&GPIOLed);
```

```
    // Call the Knight Rider effect to control 4 LEDs
    KnightRiderEffect();  // Start the Knight Rider LED effect

    while (1) {
        GPIO_ToggleOutputPin(GPIOD, GPIO_PIN_NO_12);
        delay();  // Delay for visibility of the LED toggle
    }
}

void delay(void)
{
    for (uint32_t i = 0; i < 500000; i++) {
        // Simple delay loop
    }
}

void KnightRiderEffect(void) {
    int leds[4] = {0, 0, 0, 0};  // Array to represent 4 LEDs (0 = OFF, 1 = ON)
    int i;

    // Moving LED from LED1 to LED4 and then back to LED1
    while (1) {
        // LED moves from left to right
        for (i = 0; i < 4; i++) {
            leds[i] = 1;  // Turn ON current LED
            printf("LEDs: ");
            for (int j = 0; j < 4; j++) {
                printf("%d ", leds[j]);
            }
            printf("\n");
```

```
            delay();  // Delay of 1 second
            leds[i] = 0;  // Turn OFF current LED
        }

        // LED moves from right to left
        for (i = 2; i >= 0; i--) {
            leds[i] = 1;  // Turn ON current LED
            printf("LEDs: ");
            for (int j = 0; j < 4; j++) {
                printf("%d ", leds[j]);
            }
            printf("\n");
            delay();  // Delay of 1 second
            leds[i] = 0;  // Turn OFF current LED
        }
    }
}
```

8. SOS Signal with LEDs:

Implement a program where the 4 LEDs blink together to display an SOS signal in Morse code:

3 short blinks (dot)

3 long blinks (dash)

3 short blinks (dot)

Use a delay between each sequence.

```
#include "stm32f407xx.h"
#include <stdio.h>
#include <unistd.h>  // For sleep function

void delayShort(void);
void delayLong(void);
```

```
void SOSSignal(void);


#if !defined(__SOFT_FP__) && defined(__ARM_FP)
  #warning "FPU is not initialized, but the project is compiling for an FPU. Please initialize the FPU before use."
#endif


int main(void)
{
    GPIO_Handle_t GPIOLed;
    GPIOLed.pGPIOx = GPIOD;

    // Task: Toggle green LED connected to PD12.
    GPIOLed.GPIO_PinConfig.GPIO_PinNumber = GPIO_PIN_NO_12;
    GPIOLed.GPIO_PinConfig.GPIO_PinMode = GPIO_MODE_OUT;
    GPIOLed.GPIO_PinConfig.GPIO_PinSpeed = GPIO_SPEED_FAST;
    GPIOLed.GPIO_PinConfig.GPIO_PinOPType = GPIO_OP_TYPE_PP;
    GPIOLed.GPIO_PinConfig.GPIO_PinPuPdCOntrol = GPIO_NO_PUPD;

    // ENABLE THE CLOCK FOR GPIOD PERIPHERAL
    GPIO_PeriClockControl(GPIOD, ENABLE);
    // INITIALIZING THE GPIO PERIPHERAL
    GPIO_Init(&GPIOLed);

    // Call the SOS signal function
    SOSSignal();  // Start the SOS signal

    while (1) {
        // Continuous blinking of LED as part of SOS or testing
        GPIO_ToggleOutputPin(GPIOD, GPIO_PIN_NO_12);
        delayShort();  // Short delay for visible blink
```

```
    }
}

void delayShort(void)
{
    for (uint32_t i = 0; i < 500000; i++) {
        // Simple short delay loop
    }
}

void delayLong(void)
{
    for (uint32_t i = 0; i < 500000; i++) {
        // Simple long delay loop
    }
}

void SOSSignal(void)
{
    int i;
    // Dot is 1 short blink, Dash is 1 long blink

    for (int sequence = 0; sequence < 3; sequence++) {
        // Sending dots: 3 short blinks
        for (i = 0; i < 3; i++) {
            GPIO_ToggleOutputPin(GPIOD, GPIO_PIN_NO_12);  // Turn ON LED
            delayShort();
            GPIO_ToggleOutputPin(GPIOD, GPIO_PIN_NO_12);  // Turn OFF LED
            delayShort();
        }
```

```
        // Pause between parts of SOS (dot/dash)
        delayShort();

        // Sending dashes: 3 long blinks
        for (i = 0; i < 3; i++) {
            GPIO_ToggleOutputPin(GPIOD, GPIO_PIN_NO_12);  // Turn ON LED
            delayLong();
            GPIO_ToggleOutputPin(GPIOD, GPIO_PIN_NO_12);  // Turn OFF LED
            delayShort();
        }

        // Pause between sequences of SOS
        delayLong();
    }
}
```

9.LED Temperature Indicator:

Simulate a temperature indicator using the 4 LEDs:

LED1 ON = Low temperature

LED1 & LED2 ON = Medium-low temperature

LED1, LED2 & LED3 ON = Medium-high temperature

All LEDs ON = High temperature

Update the pattern based on a simulated or actual temperature reading from a sensor.

```
#include "stm32f407xx.h"
#include <stdio.h>

// Simulated temperature sensor reading (You can replace this with actual sensor data)
int getTemperature(void);
```

```
// LED handling functions
void setLEDs(int temperature);


#if !defined(__SOFT_FP__) && defined(__ARM_FP)
  #warning "FPU is not initialized, but the project is compiling for an FPU. Please initialize the FPU before use."
#endif


int main(void)
{
    GPIO_Handle_t GPIOLed;
    GPIOLed.pGPIOx = GPIOD;

    // Configure the pins for the 4 LEDs (PD12, PD13, PD14, PD15)
    GPIOLed.GPIO_PinConfig.GPIO_PinNumber = GPIO_PIN_NO_12;
    GPIOLed.GPIO_PinConfig.GPIO_PinMode = GPIO_MODE_OUT;
    GPIOLed.GPIO_PinConfig.GPIO_PinSpeed = GPIO_SPEED_FAST;
    GPIOLed.GPIO_PinConfig.GPIO_PinOPType = GPIO_OP_TYPE_PP;
    GPIOLed.GPIO_PinConfig.GPIO_PinPuPdCOntrol = GPIO_NO_PUPD;

    // Enable the clock for GPIOD peripheral
    GPIO_PeriClockControl(GPIOD, ENABLE);
    // Initialize the GPIO peripheral
    GPIO_Init(&GPIOLed);

    // Set up the other 3 LEDs (PD13, PD14, PD15)
    for (int i = 13; i <= 15; i++) {
        GPIOLed.GPIO_PinConfig.GPIO_PinNumber = i;
        GPIO_Init(&GPIOLed);
    }
```

```
    while(1) {
        int temp = getTemperature();  // Get the simulated temperature value
        setLEDs(temp);  // Update LED status based on temperature
        delay();  // Simulate delay between updates
    }
}

// Simulate temperature reading (you can replace this function with actual sensor reading)
int getTemperature(void) {
    // Simulate a temperature between 0 and 100
    return rand() % 101;  // Returns a random temperature between 0 and 100
}

// Set the LEDs based on the current temperature
void setLEDs(int temperature) {
    if (temperature <= 25) {
        // Low temperature: Only LED1 ON
        GPIO_WriteOutputPin(GPIOD, GPIO_PIN_NO_12, GPIO_PIN_SET);
        GPIO_WriteOutputPin(GPIOD, GPIO_PIN_NO_13, GPIO_PIN_RESET);
        GPIO_WriteOutputPin(GPIOD, GPIO_PIN_NO_14, GPIO_PIN_RESET);
        GPIO_WriteOutputPin(GPIOD, GPIO_PIN_NO_15, GPIO_PIN_RESET);
    } else if (temperature <= 50) {
        // Medium-low temperature: LED1 and LED2 ON
        GPIO_WriteOutputPin(GPIOD, GPIO_PIN_NO_12, GPIO_PIN_SET);
        GPIO_WriteOutputPin(GPIOD, GPIO_PIN_NO_13, GPIO_PIN_SET);
        GPIO_WriteOutputPin(GPIOD, GPIO_PIN_NO_14, GPIO_PIN_RESET);
        GPIO_WriteOutputPin(GPIOD, GPIO_PIN_NO_15, GPIO_PIN_RESET);
    } else if (temperature <= 75) {
        // Medium-high temperature: LED1, LED2, and LED3 ON
        GPIO_WriteOutputPin(GPIOD, GPIO_PIN_NO_12, GPIO_PIN_SET);
        GPIO_WriteOutputPin(GPIOD, GPIO_PIN_NO_13, GPIO_PIN_SET);
```

```
        GPIO_WriteOutputPin(GPIOD, GPIO_PIN_NO_14, GPIO_PIN_SET);
        GPIO_WriteOutputPin(GPIOD, GPIO_PIN_NO_15, GPIO_PIN_RESET);
    } else {
        // High temperature: All LEDs ON
        GPIO_WriteOutputPin(GPIOD, GPIO_PIN_NO_12, GPIO_PIN_SET);
        GPIO_WriteOutputPin(GPIOD, GPIO_PIN_NO_13, GPIO_PIN_SET);
        GPIO_WriteOutputPin(GPIOD, GPIO_PIN_NO_14, GPIO_PIN_SET);
        GPIO_WriteOutputPin(GPIOD, GPIO_PIN_NO_15, GPIO_PIN_SET);
    }
}

// Simple delay function (can be adjusted based on system clock)
void delay(void) {
    for (uint32_t i = 0; i < 500000; i++) {
        // Simple delay loop
    }
}
```

10. Memory Game with LEDs:

Create a simple memory game:

The program lights up the LEDs in a random sequence (e.g., LED1 -> LED3 -> LED4).

The user must input the sequence using buttons or other input methods.

The program verifies the input and provides feedback with an LED blink pattern for success or failure.

```
#include "stm32f407xx.h"
#include <stdio.h>

// Simulated temperature sensor reading (You can replace this with actual sensor data)
int getTemperature(void);
```

```
// LED handling functions
void setLEDs(int temperature);


#if !defined(__SOFT_FP__) && defined(__ARM_FP)
  #warning "FPU is not initialized, but the project is compiling for an FPU. Please initialize the FPU before use."
#endif


int main(void)
{
    GPIO_Handle_t GPIOLed;
    GPIOLed.pGPIOx = GPIOD;

    // Configure the pins for the 4 LEDs (PD12, PD13, PD14, PD15)
    GPIOLed.GPIO_PinConfig.GPIO_PinNumber = GPIO_PIN_NO_12;
    GPIOLed.GPIO_PinConfig.GPIO_PinMode = GPIO_MODE_OUT;
    GPIOLed.GPIO_PinConfig.GPIO_PinSpeed = GPIO_SPEED_FAST;
    GPIOLed.GPIO_PinConfig.GPIO_PinOPType = GPIO_OP_TYPE_PP;
    GPIOLed.GPIO_PinConfig.GPIO_PinPuPdCOntrol = GPIO_NO_PUPD;

    // Enable the clock for GPIOD peripheral
    GPIO_PeriClockControl(GPIOD, ENABLE);
    // Initialize the GPIO peripheral
    GPIO_Init(&GPIOLed);

    // Set up the other 3 LEDs (PD13, PD14, PD15)
    for (int i = 13; i <= 15; i++) {
        GPIOLed.GPIO_PinConfig.GPIO_PinNumber = i;
        GPIO_Init(&GPIOLed);
    }
```

```
    while(1) {
        int temp = getTemperature();  // Get the simulated temperature value
        setLEDs(temp);  // Update LED status based on temperature
        delay();  // Simulate delay between updates
    }
}

// Simulate temperature reading (you can replace this function with actual sensor reading)
int getTemperature(void) {
    // Simulate a temperature between 0 and 100
    return rand() % 101;  // Returns a random temperature between 0 and 100
}

// Set the LEDs based on the current temperature
void setLEDs(int temperature) {
    if (temperature <= 25) {
        // Low temperature: Only LED1 ON
        GPIO_WriteOutputPin(GPIOD, GPIO_PIN_NO_12, GPIO_PIN_SET);
        GPIO_WriteOutputPin(GPIOD, GPIO_PIN_NO_13, GPIO_PIN_RESET);
        GPIO_WriteOutputPin(GPIOD, GPIO_PIN_NO_14, GPIO_PIN_RESET);
        GPIO_WriteOutputPin(GPIOD, GPIO_PIN_NO_15, GPIO_PIN_RESET);
    } else if (temperature <= 50) {
        // Medium-low temperature: LED1 and LED2 ON
        GPIO_WriteOutputPin(GPIOD, GPIO_PIN_NO_12, GPIO_PIN_SET);
        GPIO_WriteOutputPin(GPIOD, GPIO_PIN_NO_13, GPIO_PIN_SET);
        GPIO_WriteOutputPin(GPIOD, GPIO_PIN_NO_14, GPIO_PIN_RESET);
        GPIO_WriteOutputPin(GPIOD, GPIO_PIN_NO_15, GPIO_PIN_RESET);
    } else if (temperature <= 75) {
        // Medium-high temperature: LED1, LED2, and LED3 ON
        GPIO_WriteOutputPin(GPIOD, GPIO_PIN_NO_12, GPIO_PIN_SET);
        GPIO_WriteOutputPin(GPIOD, GPIO_PIN_NO_13, GPIO_PIN_SET);
```

```
        GPIO_WriteOutputPin(GPIOD, GPIO_PIN_NO_14, GPIO_PIN_SET);
        GPIO_WriteOutputPin(GPIOD, GPIO_PIN_NO_15, GPIO_PIN_RESET);
    } else {
        // High temperature: All LEDs ON
        GPIO_WriteOutputPin(GPIOD, GPIO_PIN_NO_12, GPIO_PIN_SET);
        GPIO_WriteOutputPin(GPIOD, GPIO_PIN_NO_13, GPIO_PIN_SET);
        GPIO_WriteOutputPin(GPIOD, GPIO_PIN_NO_14, GPIO_PIN_SET);
        GPIO_WriteOutputPin(GPIOD, GPIO_PIN_NO_15, GPIO_PIN_SET);
    }
}

// Simple delay function (can be adjusted based on system clock)
void delay(void) {
    for (uint32_t i = 0; i < 500000; i++) {
        // Simple delay loop
    }
}
```