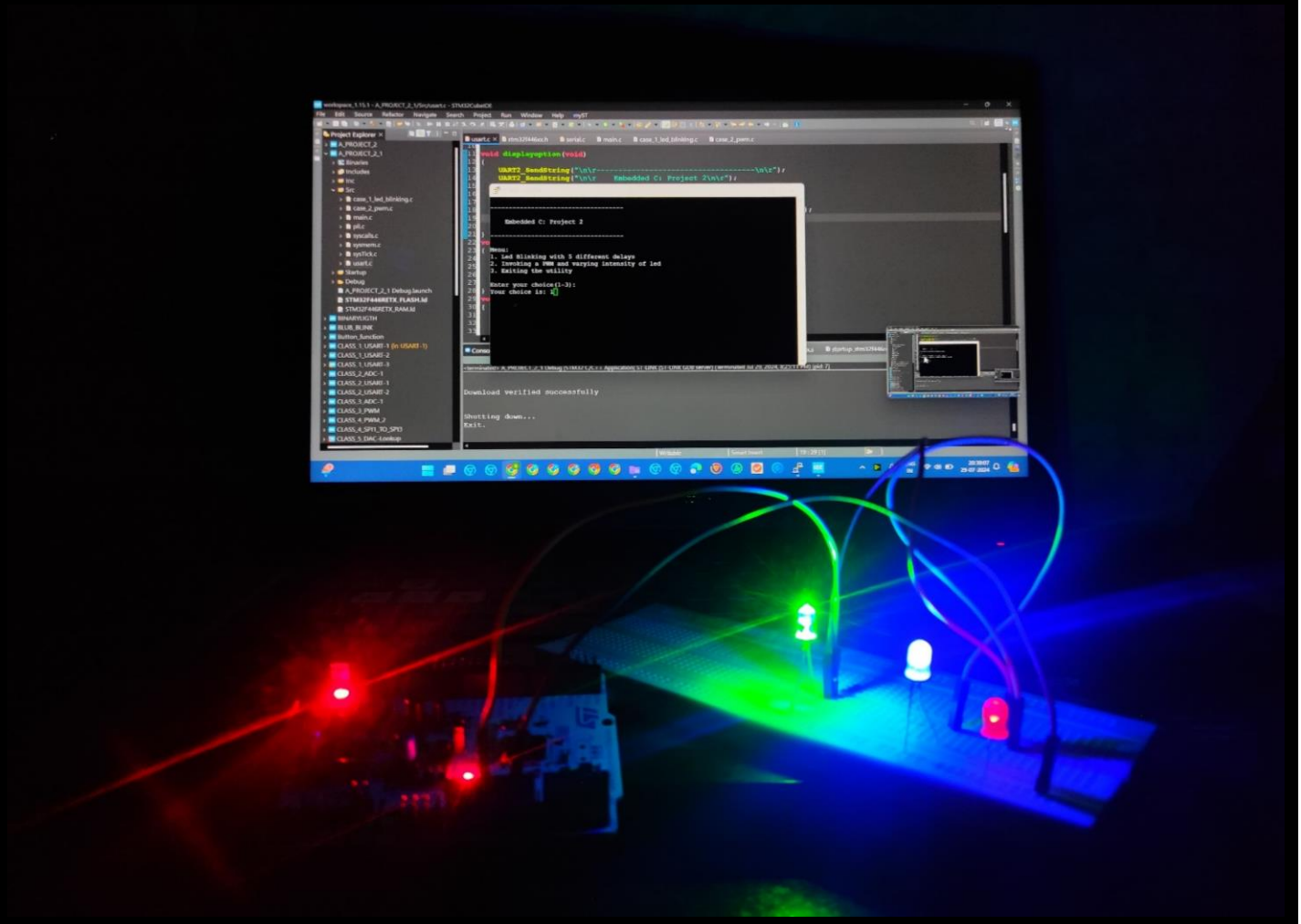
C. V. RAMAN GLOBAL UNIVERSITY

Bhubaneswar, Mahura-752052

PROJECT 2



Made By:

Group – 21

Team Members:

Sofia Akhtar (2201020130)

Sugam Shaw (2201020139)

B. Tech (CSE)

5th Semester

Under the guidance of Md. Salahuddin in CDAC LAB

1. INCLUDE FILES

1.1 case_1_led_blinking.h

```
#pragma once  // Ensures the header file is included only once during compilation
// Function to initialize the LED blinking setup
void led_blinking_init(void);
// Function to start the LED blinking
void led_blinking_start(void);
// Function to stop the LED blinking
void led_blinking_stop(void);
```

1.2 case_2_pwm.h

```
// Ensure this file is included only once in a single compilation
#pragma once
// Function prototypes for initializing, starting, and stopping PWM intensity control
// Function to initialize the PWM intensity control setup
void PWM_Intensity_init(void);
// Function to start the PWM intensity control
void PWM_Intensity_start(void);
// Function to stop the PWM intensity control
void PWM_Intensity_close(void);
```

1.3 pll.h

```
#pragma once  // Ensure this file is included only once in a single compilation
// PLL configuration values for setting the clock speed
#define PLL_M      8ul    // PLLM: Division factor for the main PLL (PLL) and audio PLL (PLLI2S) input clock
#define PLL_N      180ul  // PLLN: Main PLL (PLL) multiplication factor for VCO
#define PLL_P      0ul    // PLLP: Main PLL (PLL) division factor for main system clock
#define PLL_Q      2ul    // PLLQ: Main PLL (PLL) division factor for USB OTG FS, SDIO, and random number generator clocks
// Clock frequencies
#define HCLK_FREQ   180000000uL  // HCLK frequency (System clock frequency)
#define APB1_FREQ   (HCLK_FREQ/4) // APB1 peripheral clock frequency (HCLK divided by 4)
#define APB2_FREQ   (HCLK_FREQ/2) // APB2 peripheral clock frequency (HCLK divided by 2)
// Function prototype for configuring the clock speed using PLL
void clockSpeed_PLL(void);
```

1.4 sysTick.h

```
#pragma once  // Ensure this file is included only once in a single compilation
#include "stm32f446xx.h"  // Include STM32F446xx microcontroller definitions
// Function prototypes for SysTick timer initialization, handler, and delay functions
// Function to initialize the SysTick timer
void SysTick_Init(void);
```

```
// SysTick Handler function to be called when SysTick interrupt occurs
void SysTick_Handler(void);
// Function to create a delay in milliseconds
void delay_ms(uint32_t ms);
// Function to get the current milliseconds since the system started
uint32_t getMillis(void);
```

1.5 Usart.h

```
#pragma once  // Ensure this file is included only once in a single compilation
#include <stdint.h>  // Include standard integer types
// Function prototypes for UART/USART communication
// Function to display available options or settings
void displayoption(void);
// Function to send a single character via UART2
void UART2_SendChar(char ch);
// Function to send a string via UART2
void UART2_SendString(char *string);
// Function to configure USART2 settings
void Usart2_config(void);
// Function to receive a byte of data via UART2
uint8_t receiverdata(void);
```

2. SOURCE FILES

2.1 Main.c

```
#include <stdint.h>        // Include standard integer types
#include <stdio.h>         // Include standard I/O functions
#include <stddef.h>        // Include standard definitions
#include "stm32f446xx.h"    // Include STM32F446xx microcontroller definitions
#include "pll.h"           // Include PLL configuration functions
#include "usart.h"         // Include USART functions
#include "sysTick.h"       // Include SysTick functions
#include "case_1_led_blinking.h" // Include LED blinking functions
#include "case_2_pwm.h"     // Include PWM intensity functions
volatile uint8_t choice;    // Global variable to store user choice
int main()
{
   // Configure USART2 for communication
   Usart2_config();
   // Initialize LED blinking functionality
   led_blinking_init();
   // Initialize PWM intensity control
   PWM_Intensity_init();
   while(1)  // Infinite loop to handle menu options
   {
      // Display available options to the user
      displayoption();
```

```
    // Receive user choice from UART
    choice = receiverdata();
    // Send the received choice back to the user
    UART2_SendString("\n\rYour choice is: ");
    UART2_SendChar(choice);
    // Handle the user's choice
    switch(choice)
    {
      case '1':
        // Start LED blinking and stop it after a certain condition
        led_blinking_start();
        led_blinking_stop();
        UART2_SendString("\n\rCase 1 ends here\n\r");
        break;
      case '2':
        // Start PWM intensity control and stop it after a certain condition
        PWM_Intensity_start();
        PWM_Intensity_close();
        UART2_SendString("\n\rCase 2 ends here\n\r");
        break;

      case '3':
        // Stop LED blinking and PWM intensity control, then exit
        led_blinking_stop();
        PWM_Intensity_close();
        UART2_SendString("\n\rExiting\n\r");
        break;
      default:
        // Handle invalid choices
        UART2_SendString("\n\rInvalid choice!! Enter a correct value\n\r");
        break;
    }
    // Exit the loop if the user chooses option '3'
    if(choice == '3')
    {
      break;
    }
  }
  return 0; // Return 0 to indicate successful execution
}
```

2.2 Usart.c

```
#include "usart.h"
#include <stdint.h>
#include "stm32f446xx.h"
// Define macros for alternate functions
#define PA2_AF          (2 << 4)  // Alternate function mode for PA2
#define PA2_AF_USART2_TX (7 << 8)  // USART2 TX alternate function for PA2
```

```
#define PA3_AF          (2 << 6)  // Alternate function mode for PA3
#define PA3_AF_USART2_RX (7 << 12) // USART2 RX alternate function for PA3
// Function to display menu options over UART
void displayoption(void)
{
        UART2_SendString("\n\r-----------------------------------\n\r");
        UART2_SendString("\n\r    Embedded C: Project 2\n\r");
        UART2_SendString("\n\r-----------------------------------\n\r");
        UART2_SendString("\n\rMenu:\n\r");
        UART2_SendString("1. Led Blinking with 5 different delays \n\r");
        UART2_SendString("2. Invoking a PWM and varying intensity of led\n\r");
        UART2_SendString("3. Exiting the utility\n\r");
        UART2_SendString("\n\rEnter your choice(1-3): ");
}
// Function to send a single character over UART2
void UART2_SendChar(char ch)
{
        USART2->DR = ch; // Load the data register with the character
        while(!(USART2->SR & USART_SR_TXE)); // Wait until the transmit data register is empty
}
// Function to send a string over UART2
void UART2_SendString(char *string)
{
        while(*string != '\0')
        {
                UART2_SendChar(*string); // Send each character of the string
                string++;
        }
}
// Function to configure USART2
void Usart2_config(void)
{
        // Enable the clocks for GPIOA and USART2
        RCC->AHB1ENR |= RCC_AHB1ENR_GPIOAEN; // Enable GPIOA clock
        RCC->APB1ENR |= RCC_APB1ENR_USART2EN; // Enable USART2 clock

        // Configure PA2 and PA3 for alternate function (USART2 TX/RX)
        GPIOA->MODER |= PA2_AF;  // Set PA2 to alternate function mode
        GPIOA->MODER |= PA3_AF;  // Set PA3 to alternate function mode

        GPIOA->OSPEEDR |= (3 << 4); // Set high speed for PA2
        GPIOA->OSPEEDR |= (3 << 6); // Set high speed for PA3

        GPIOA->AFR[0] |= PA2_AF_USART2_TX; // Set alternate function for PA2 as USART2 TX
        GPIOA->AFR[0] |= PA3_AF_USART2_RX; // Set alternate function for PA3 as USART2 RX

        USART2->CR1 = 0x00; // Clear all settings in CR1
        USART2->CR1 |= USART_CR1_UE; // Enable USART2
```

```
	USART2->CR1 &= ~(1 << 12); // Set word length to 8 bits

	USART2->BRR |= (1 << 0); // Set baud rate mantissa
	USART2->BRR |= (24 << 4); // Set baud rate fraction

	USART2->CR1 |= USART_CR1_RE; // Enable receiver
	USART2->CR1 |= USART_CR1_TE; // Enable transmitter
}

// Function to receive data from UART2
uint8_t receiverdata(void)
{
	while(!(USART2->SR & USART_SR_RXNE)); // Wait until data is received
	uint8_t data = USART2->DR; // Read the received data
	return data; // Return the received data
}
```

2.3 Case1_led_blinking.c

```
#include "stm32f446xx.h"  // Include the header file for the STM32F446xx microcontroller
#include "usart.h"        // Include the header file for USART communication
#include <stdio.h>        // Include the standard I/O header file for printf and other functions

#define Led_Output_Pin_PA1     1   // Define PA1 as the output pin for the LED
#define Button_Input_Pin_PA13  13  // Define PA1 as the output pin for the LED

// Declare a volatile integer variable for controlling the LED blinking delay
volatile int led_blinking_delay = 1;

// Function to initialize the LED blinking setup
void led_blinking_init(void)
{
   // Enable the clock for GPIOA and GPIOC
   RCC->AHB1ENR |= RCC_AHB1ENR_GPIOAEN | RCC_AHB1ENR_GPIOCEN;

   // Set PA1 as output
   GPIOA->MODER &= ~(3 << (2*Led_Output_Pin_PA1)); // Reset the MODER bits for PA1
   GPIOA->MODER |= (1 << (2*Led_Output_Pin_PA1));  // Set PA1 as output mode

   // Set PC13 as input
   GPIOC->MODER &= ~(3 << (2*Button_Input_Pin_PA13)); // Reset the MODER bits for PC13
}
// Function to start the LED blinking
void led_blinking_start(void)
{
	UART2_SendString("\n\rLed Blinking Starts\n\r");
	UART2_SendString("Led Blinking at delay speed ");
	UART2_SendChar((char)(led_blinking_delay + '0')); // Print the current delay value
	UART2_SendString("x\n\r");
```

```
  while (led_blinking_delay <= 5)  // Loop while the delay is less than or equal to 5
  {
    GPIOA->ODR ^= (1 << 1);  // Toggle the LED connected to PA1
    for (int i = 0; i < (1000000 * led_blinking_delay); i++)  // Delay loop
    {
      if (!(GPIOC->IDR & (1 << Button_Input_Pin_PA13)))  // Check if the button connected to PC13 is pressed
      {
        led_blinking_delay += 1;  // Increment the delay
        if (led_blinking_delay >= 6)  // If the delay exceeds 6, break the loop
        {
          break;
        }

        while (!(GPIOC->IDR & (1 << Button_Input_Pin_PA13)));  // Wait until the button is released
        UART2_SendString("Led Blinking at delay speed ");
        UART2_SendChar((char)(led_blinking_delay + '0')); // Print the current delay value
        UART2_SendString("x\n\r");
        break;
      }
    }
  }
  led_blinking_delay = 1;  // Reset the delay to 1
}
// Function to stop the LED blinking
void led_blinking_stop(void)
{
  GPIOA->ODR &= ~(1 << 1);  // Turn off the LED connected to PA1
}
```

2.4 Case2_pwm.c

```
#include <stdio.h>          // Include standard I/O functions
#include "stm32f446xx.h"    // Include STM32F446xx microcontroller definitions
#include "pll.h"            // Include PLL configuration functions
#include "sysTick.h"        // Include SysTick functions

volatile int bright_pwm = 10; // Global variable to control PWM brightness

// Pin definitions
#define Led_Output_Pin_PA0 0  // Define PA0 as the output pin for PWM
#define Button_Input_Pin_PC13 13 // Define PC13 as the input pin for the button

// Function to initialize PWM intensity control
void PWM_Intensity_init(void)
{
  // Configure the system clock using PLL settings
  clockSpeed_PLL();
  // Initialize SysTick timer
  SysTick_Init();
  // Enable clock for GPIOA and GPIOC
```

```
  RCC->AHB1ENR |= RCC_AHB1ENR_GPIOAEN | RCC_AHB1ENR_GPIOCEN;
  // Configure PA0 for PWM output
  GPIOA->MODER &= ~(3 << (2 * Led_Output_Pin_PA0)); // Reset mode for PA0
  GPIOA->MODER |= (2 << (2 * Led_Output_Pin_PA0));  // Set PA0 to alternate function mode
  GPIOA->AFR[0] |= (1 << (4 * Led_Output_Pin_PA0)); // Set alternate function for PA0
  // Configure PC13 as input for button
  GPIOC->MODER &= ~(3 << (2 * Button_Input_Pin_PC13));
  // Enable clock for TIM2
  RCC->APB1ENR |=  RCC_APB1ENR_TIM2EN;
  // Configure TIM2 for PWM
  TIM2->ARR = 100;             // Set auto-reload register to 100
  TIM2->CCMR1 |= (1 << 5) | (1 << 6); // Set PWM mode 1 on channel 1
  TIM2->CCER |= (1 << 0);       // Enable capture/compare on channel 1
  TIM2->CR1 |= (1 << 0);        // Enable TIM2 counter
}

// Function to start PWM intensity control
void PWM_Intensity_start(void)
{
          char buffer[10];  // Buffer to hold string representations of numbers
          UART2_SendString("\n\rPWM Starts\n\r");
          UART2_SendString("Led glowing at ");
          sprintf(buffer, "%d", bright_pwm);  // Convert bright_pwm to string
          UART2_SendString(buffer);  // Print the current PWM brightness value
          UART2_SendString("x intensity\n\r");
          TIM2->CCR1 = bright_pwm;
  while (bright_pwm <= 100)  // Loop while PWM brightness is less than or equal to 100
  {
    if (!(GPIOC->IDR & (1 << Button_Input_Pin_PC13)))  // Check if button (PC13) is pressed
    {
      TIM2->CCR1 = bright_pwm; // Set PWM duty cycle
      bright_pwm += 10;      // Increase brightness
      if (bright_pwm > 100)  // If brightness exceeds 100, exit loop
      {
        break;
      }
      while (!(GPIOC->IDR & (1 << Button_Input_Pin_PC13)));  // Wait until button is released
      UART2_SendString("Led glowing at ");
      sprintf(buffer, "%d", bright_pwm);  // Convert bright_pwm to string
      UART2_SendString(buffer);  // Print the current PWM brightness value
      UART2_SendString("x intensity\n\r");
    }
  }
  bright_pwm = 10;  // Reset brightness after loop
}
// Function to stop PWM intensity control
void PWM_Intensity_close(void)
{
  TIM2->CCR1 = 0;  // Set PWM duty cycle to 0 (turn off PWM)
```

```
}
```

2.5 pll.c

```
#include "stm32f446xx.h"
#include "pll.h"
void clockSpeed_PLL(){

   RCC->CR |= RCC_CR_HSION;
   while (!(RCC->CR & RCC_CR_HSIRDY));
   RCC->PLLCFGR = (PLL_M) | (PLL_N << 6) | (PLL_P << 16) | (PLL_Q << 24);
   RCC->PLLCFGR &=~ RCC_PLLCFGR_PLLSRC;

   RCC->CFGR |= RCC_CFGR_HPRE_DIV1 | RCC_CFGR_PPRE2_DIV2 | RCC_CFGR_PPRE1_DIV4;

   RCC->CR |= RCC_CR_PLLON;
   while (!(RCC->CR & RCC_CR_PLLRDY));

   RCC->APB1ENR |= RCC_APB1ENR_PWREN;

   PWR->CR |= PWR_CR_ODEN;
   while (!(PWR->CSR & PWR_CSR_ODRDY)) ;

   PWR->CR |= PWR_CR_ODSWEN;
   while (!(PWR->CSR & PWR_CSR_ODSWRDY)) ;

   FLASH->ACR = FLASH_ACR_PRFTEN | FLASH_ACR_ICEN | FLASH_ACR_DCEN | FLASH_ACR_LATENCY_5WS;

   RCC->CFGR &=~ RCC_CFGR_SW;
   RCC->CFGR |= RCC_CFGR_SW_PLL;
   while ((RCC->CFGR & RCC_CFGR_SWS ) != RCC_CFGR_SWS_PLL);
}
```

2.6 sysTick.c

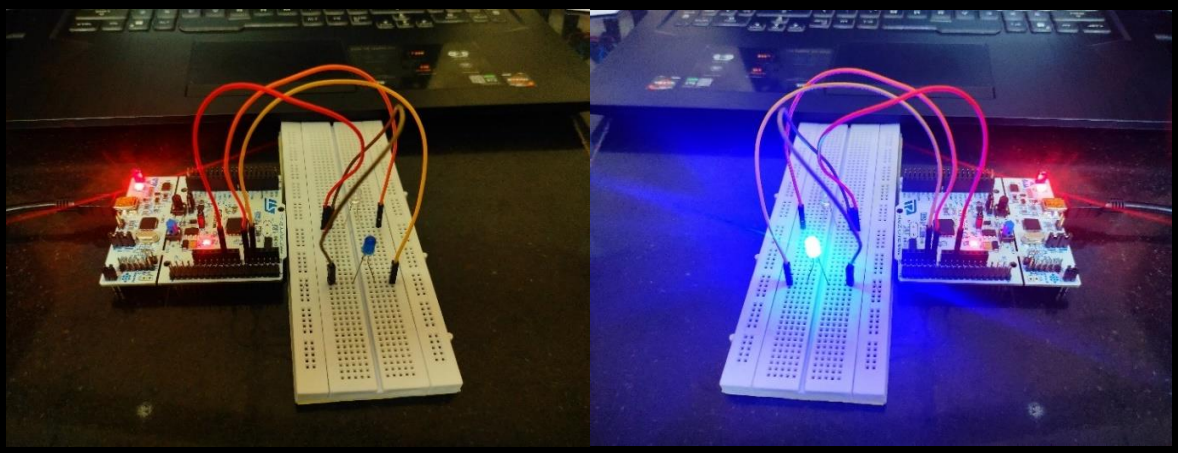
```
#include "stm32f446xx.h"  // Include STM32F446xx microcontroller definitions
#include "pll.h"          // Include PLL configuration functions

volatile uint32_t ms_counter = 0;  // Counter for delay management
volatile uint32_t millis = 0;      // Millisecond counter for timekeeping

// Function to initialize the SysTick timer
void SysTick_Init(){
   // Set SysTick timer value register to 0 (start counting from 0)
   SysTick->VAL = 0;

   // Set SysTick timer load value for 1 millisecond interval
   // HCLK_FREQ / 1000 gives the number of cycles per millisecond
   SysTick->LOAD = (HCLK_FREQ / 1000) - 1;
```

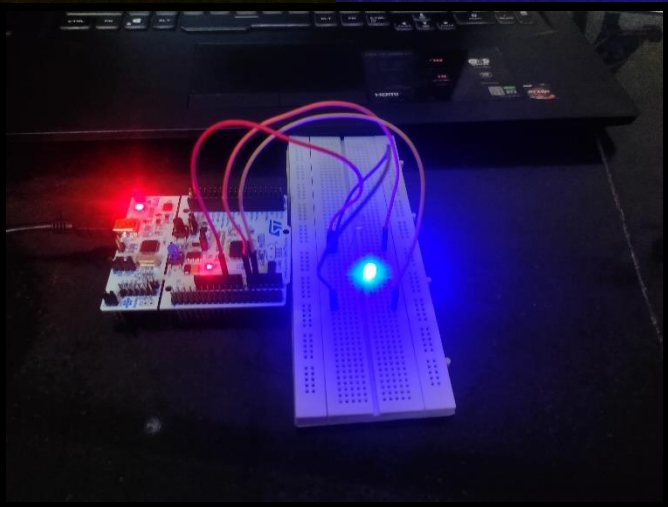
```
  // Configure SysTick:
  // - Use the processor clock as the clock source
  // - Enable SysTick interrupt
  // - Enable SysTick counter
  SysTick->CTRL = SysTick_CTRL_CLKSOURCE_Msk |
                  SysTick_CTRL_TICKINT_Msk |
                  SysTick_CTRL_ENABLE_Msk;
}

// SysTick Handler function called every millisecond
void SysTick_Handler(){
  // Decrement ms_counter if it is greater than zero
  if (ms_counter) ms_counter--;

  // Increment millisecond counter
  millis++;
}

// Function to create a delay in milliseconds
void delay_ms(uint32_t ms){
  ms_counter = ms;     // Set the ms_counter to the requested delay
  while (ms_counter);  // Wait until ms_counter reaches zero
}

// Function to get the number of milliseconds since system startup
uint32_t getMillis(){
  return millis;       // Return the current millisecond count
}
```

OUTPUT

CASE 1

```
-----------------------------------

    Embedded C: Project 2

-----------------------------------

Menu:
1. Led Blinking with 5 different delays
2. Invoking a PWM and varying intensity of led
3. Exiting the utility

Enter your choice(1-3):
Your choice is: 1
Led Blinking Starts
Led Blinking at speed 1x
Led Blinking at speed 2x
Led Blinking at speed 3x
Led Blinking at speed 4x
Led Blinking at speed 5x

Case 1 ends here
```

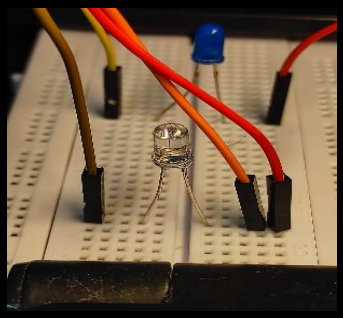
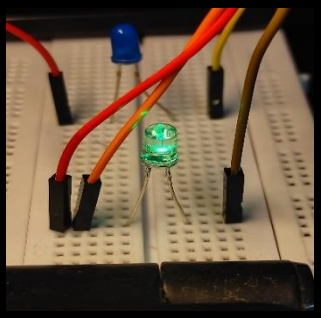
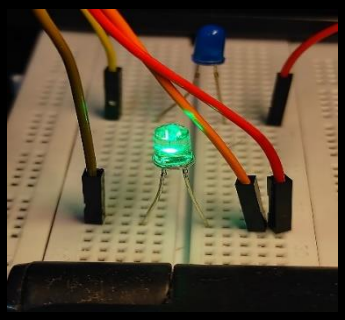
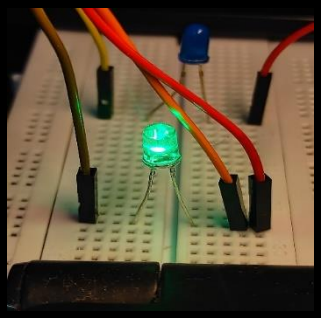
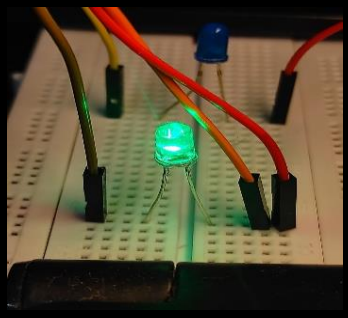
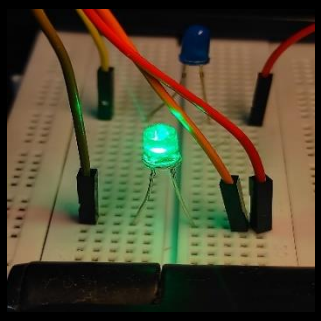
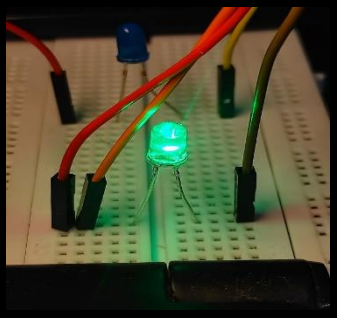
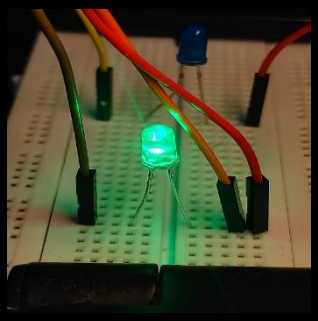
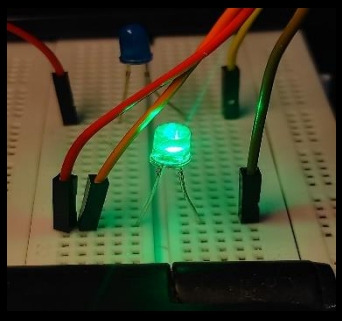
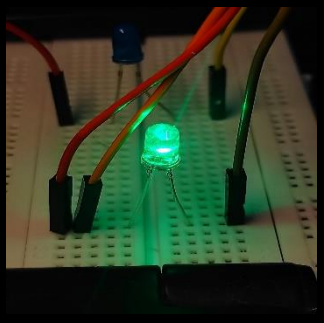




CASE 2

```
-----------------------------------

    Embedded C: Project 2

-----------------------------------

Menu:
1. Led Blinking with 5 different delays
2. Invoking a PWM and varying intensity of led
3. Exiting the utility

Enter your choice(1-3):
Your choice is: 2
PWM Starts
Led glowing at 10x intensity
Led glowing at 20x intensity
Led glowing at 30x intensity
Led glowing at 40x intensity
Led glowing at 50x intensity
Led glowing at 60x intensity
Led glowing at 70x intensity
Led glowing at 80x intensity
Led glowing at 90x intensity
Led glowing at 100x intensity

Case 2 ends here
```

Output of case 2:

CASE 3:

```
-----------------------------------

    Embedded C: Project 2

-----------------------------------

Menu:
1. Led Blinking with 5 different delays
2. Invoking a PWM and varying intensity of led
3. Exiting the utility

Enter your choice(1-3):
Your choice is: 3
Exiting the utility

      Thank You!! Visit Again

-----------------------------------
```

INVALID CASE:

```
-----------------------------------

    Embedded C: Project 2

-----------------------------------

Menu:
1. Led Blinking with 5 different delays
2. Invoking a PWM and varying intensity of led
3. Exiting the utility

Enter your choice(1-3):
Your choice is: 4
Invalid choice!! Enter a correct value

-----------------------------------
```