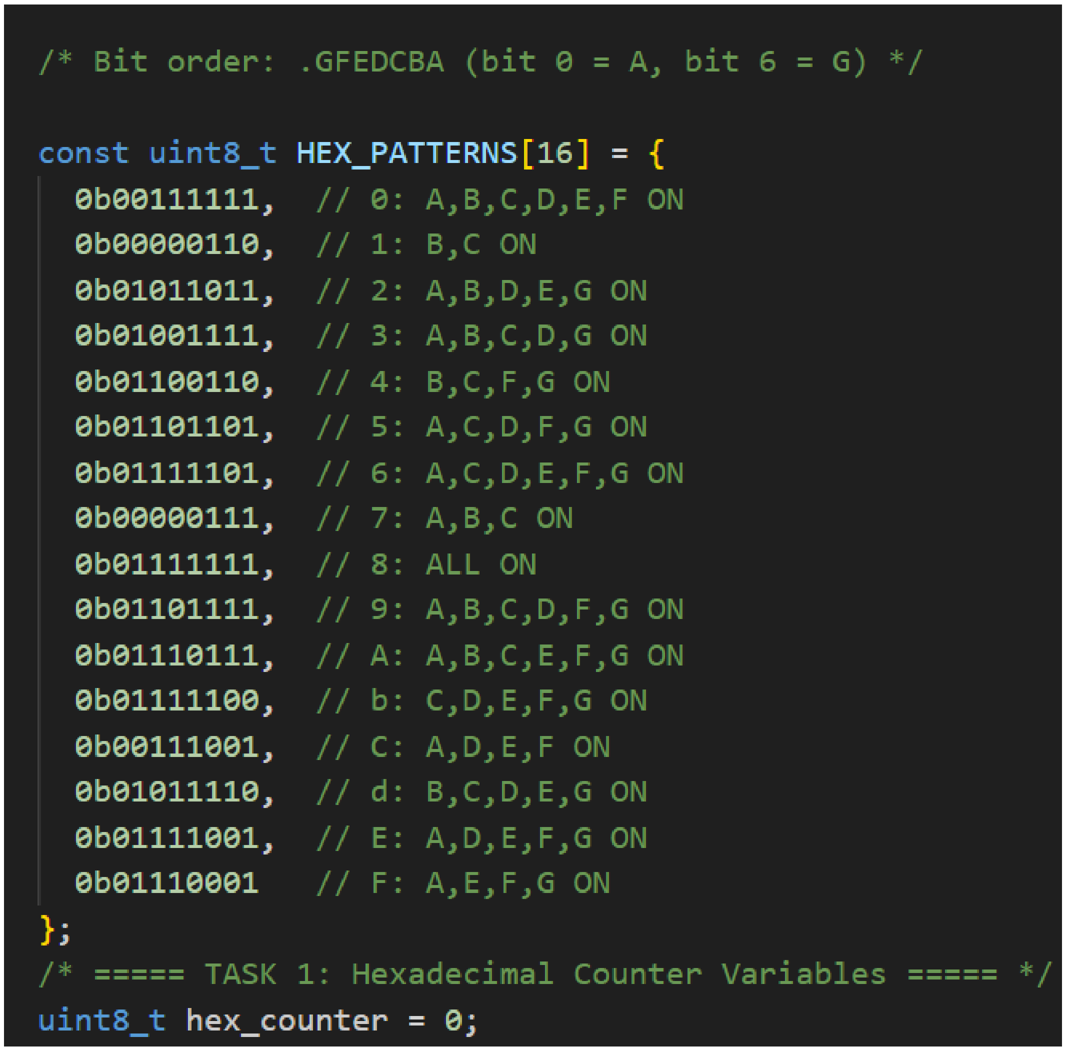
MCI report Lab #3

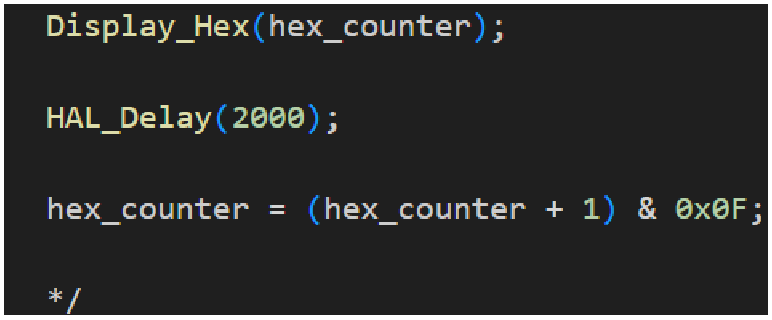
Ali Hussain Zaidi (sz10384)

Muhammad Haider Zaidi (mz10270)

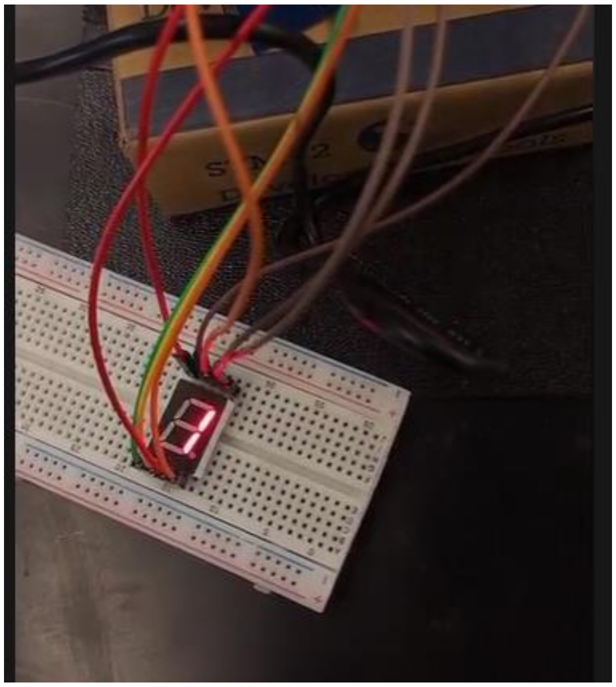
Task # 1:



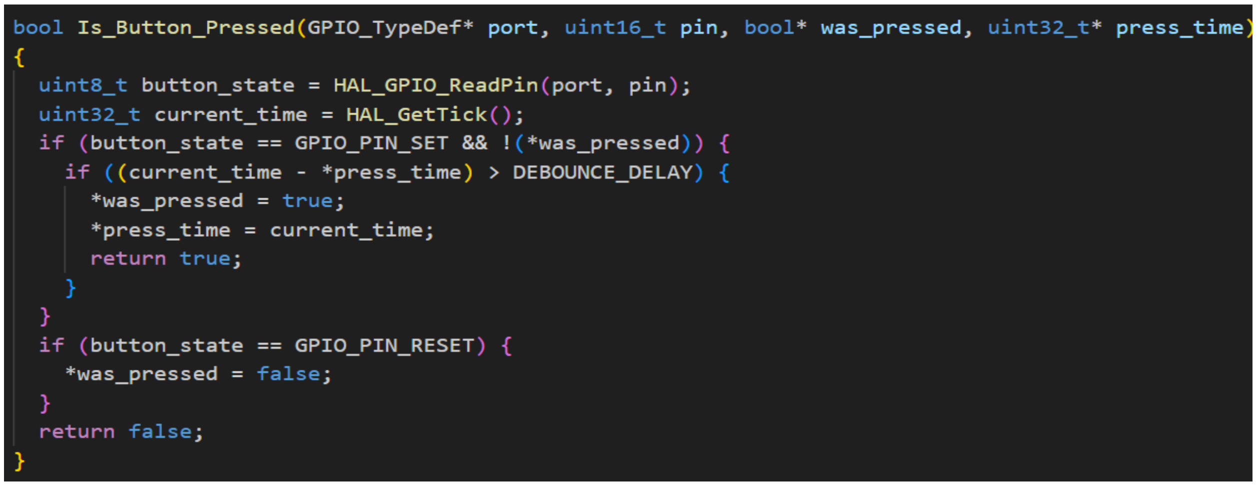
```
/* Bit order: .GFEDCBA (bit 0 = A, bit 6 = G) */

const uint8_t HEX_PATTERNS[16] = {
  0b00111111,  // 0: A,B,C,D,E,F ON
  0b00000110,  // 1: B,C ON
  0b01011011,  // 2: A,B,D,E,G ON
  0b01001111,  // 3: A,B,C,D,G ON
  0b01100110,  // 4: B,C,F,G ON
  0b01101101,  // 5: A,C,D,F,G ON
  0b01111101,  // 6: A,C,D,E,F,G ON
  0b00000111,  // 7: A,B,C ON
  0b01111111,  // 8: ALL ON
  0b01101111,  // 9: A,B,C,D,F,G ON
  0b01110111,  // A: A,B,C,E,F,G ON
  0b01111100,  // b: C,D,E,F,G ON
  0b00111001,  // C: A,D,E,F ON
  0b01011110,  // d: B,C,D,E,G ON
  0b01111001,  // E: A,D,E,F,G ON
  0b01110001   // F: A,E,F,G ON
};
/* ===== TASK 1: Hexadecimal Counter Variables ===== */
uint8_t hex_counter = 0;
```



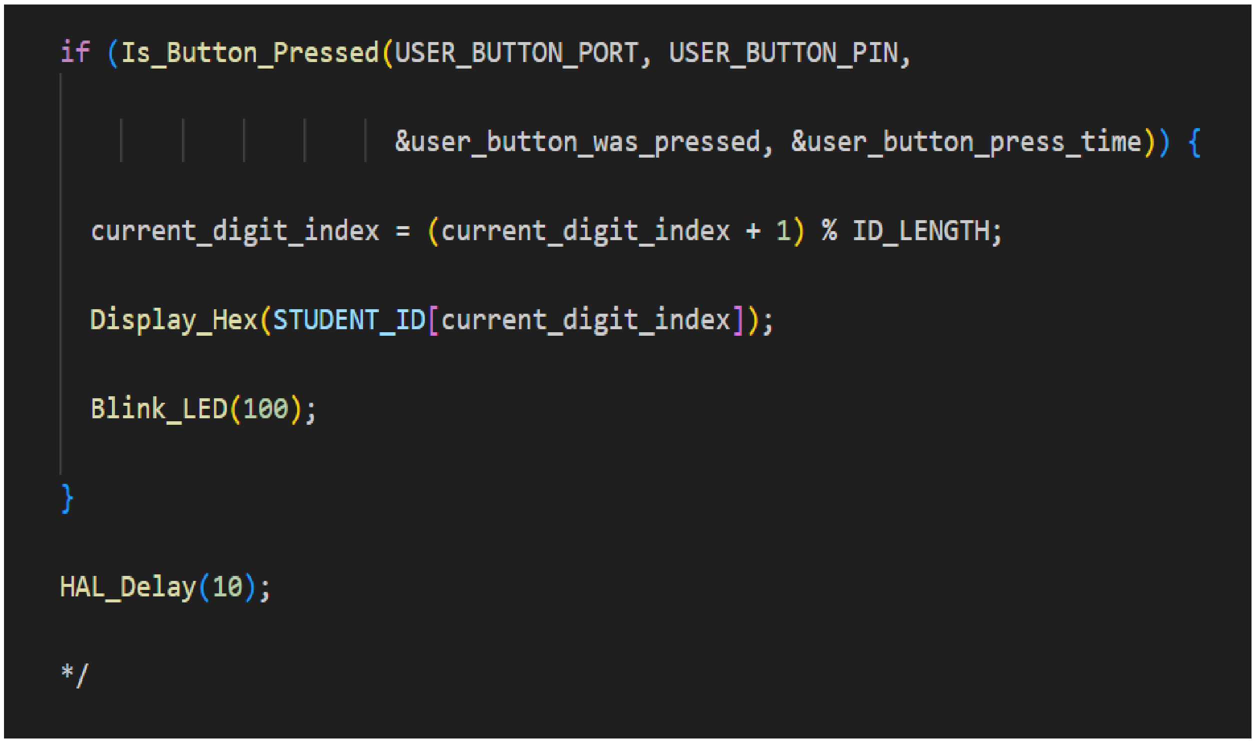
```
Display_Hex(hex_counter);

HAL_Delay(2000);

hex_counter = (hex_counter + 1) & 0x0F;

*/
```



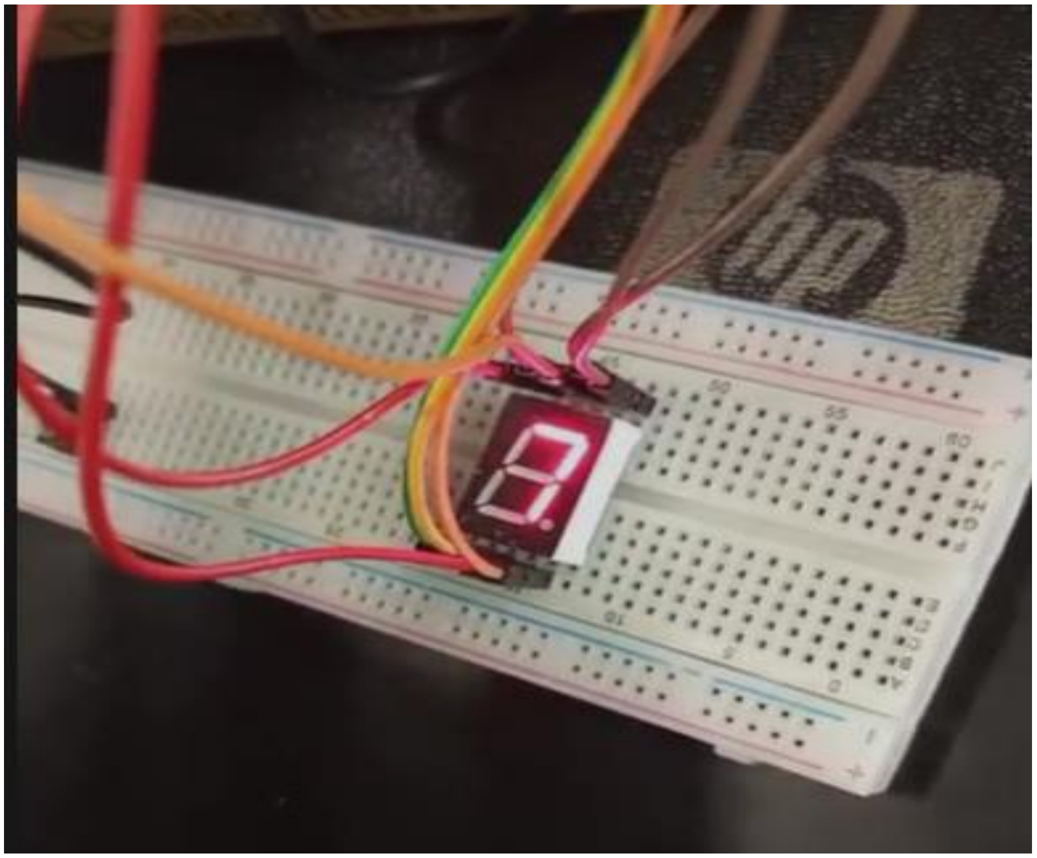
Task # 2:



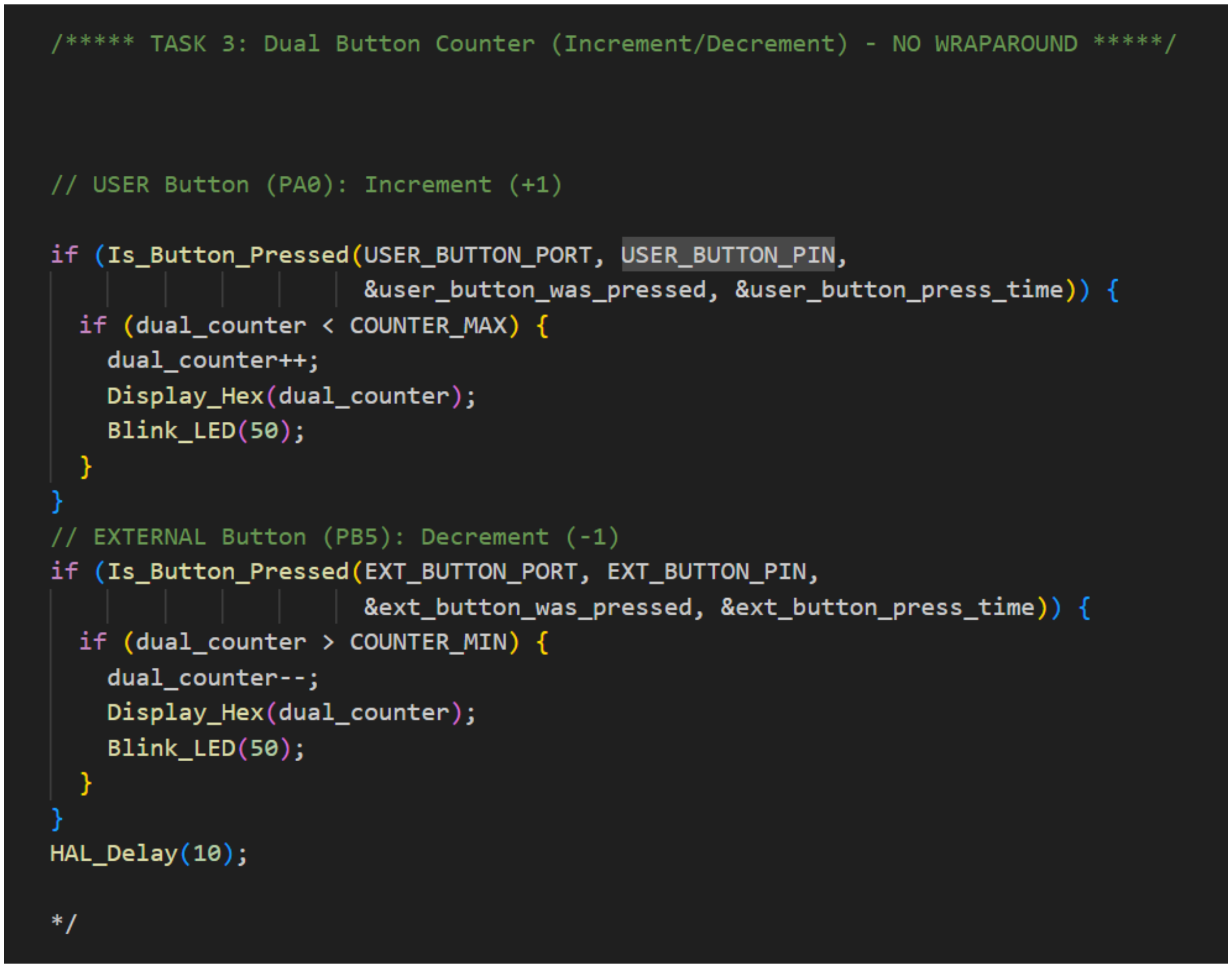
```c
bool Is_Button_Pressed(GPIO_TypeDef* port, uint16_t pin, bool* was_pressed, uint32_t* press_time)
{
  uint8_t button_state = HAL_GPIO_ReadPin(port, pin);
  uint32_t current_time = HAL_GetTick();
  if (button_state == GPIO_PIN_SET && !(*was_pressed)) {
    if ((current_time - *press_time) > DEBOUNCE_DELAY) {
      *was_pressed = true;
      *press_time = current_time;
      return true;
    }
  }
  if (button_state == GPIO_PIN_RESET) {
    *was_pressed = false;
  }
  return false;
}
```



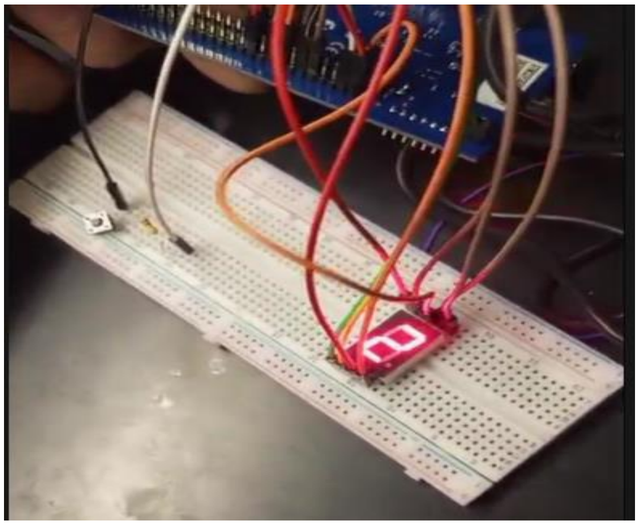
```
if (Is_Button_Pressed(USER_BUTTON_PORT, USER_BUTTON_PIN,
                      &user_button_was_pressed, &user_button_press_time)) {
  current_digit_index = (current_digit_index + 1) % ID_LENGTH;
  Display_Hex(STUDENT_ID[current_digit_index]);
  Blink_LED(100);
}
HAL_Delay(10);
*/
```



Task # 3:

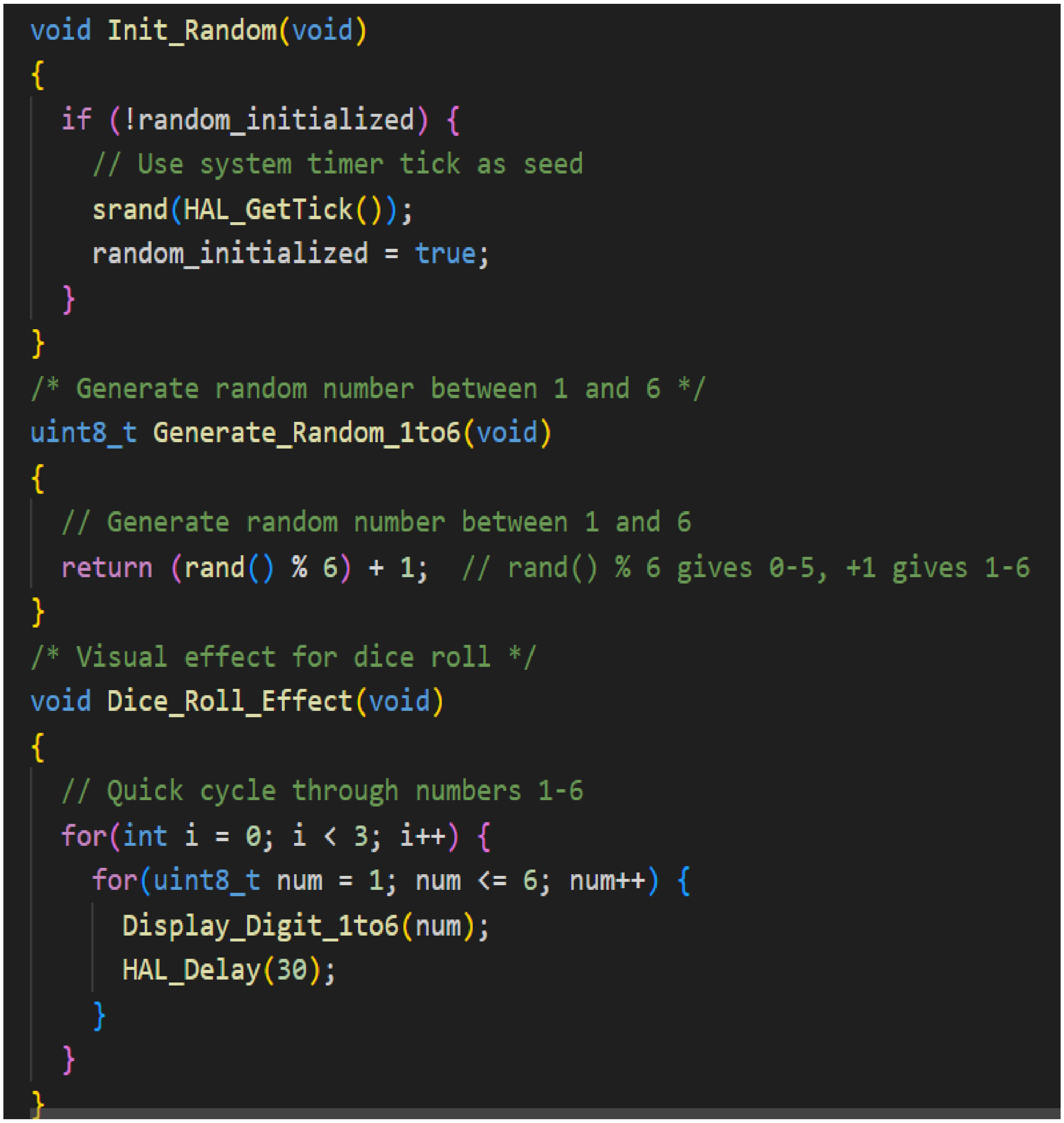


```
/***** TASK 3: Dual Button Counter (Increment/Decrement) - NO WRAPAROUND *****/



// USER Button (PA0): Increment (+1)

if (Is_Button_Pressed(USER_BUTTON_PORT, USER_BUTTON_PIN,
                      &user_button_was_pressed, &user_button_press_time)) {
  if (dual_counter < COUNTER_MAX) {
    dual_counter++;
    Display_Hex(dual_counter);
    Blink_LED(50);
  }
}
// EXTERNAL Button (PB5): Decrement (-1)
if (Is_Button_Pressed(EXT_BUTTON_PORT, EXT_BUTTON_PIN,
                      &ext_button_was_pressed, &ext_button_press_time)) {
  if (dual_counter > COUNTER_MIN) {
    dual_counter--;
    Display_Hex(dual_counter);
    Blink_LED(50);
  }
}
HAL_Delay(10);

*/
```

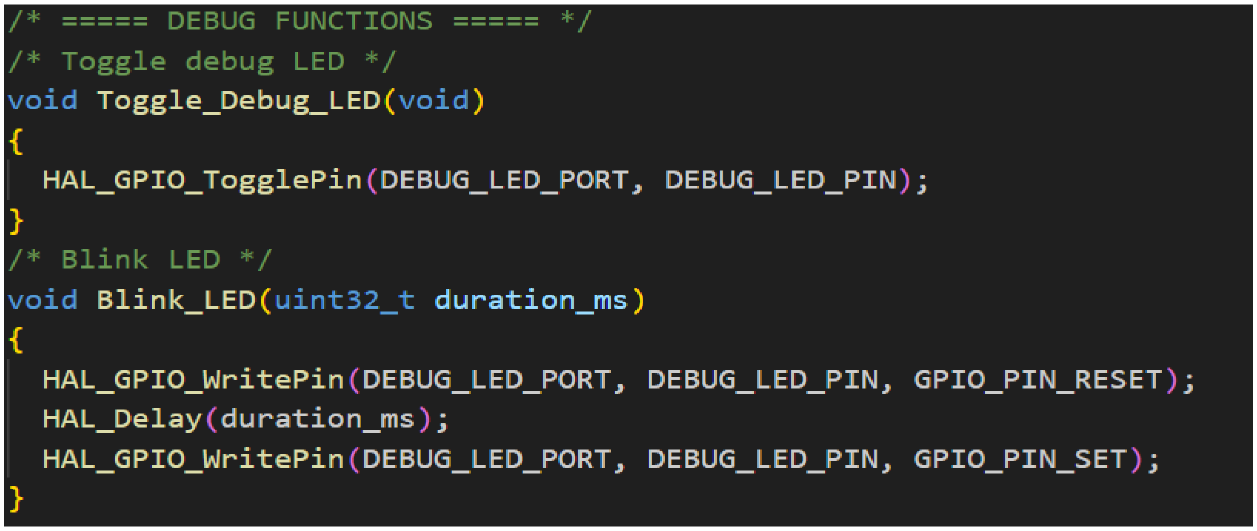


Task # 4

TASK 4 SPECIFIC FUNCTIONS

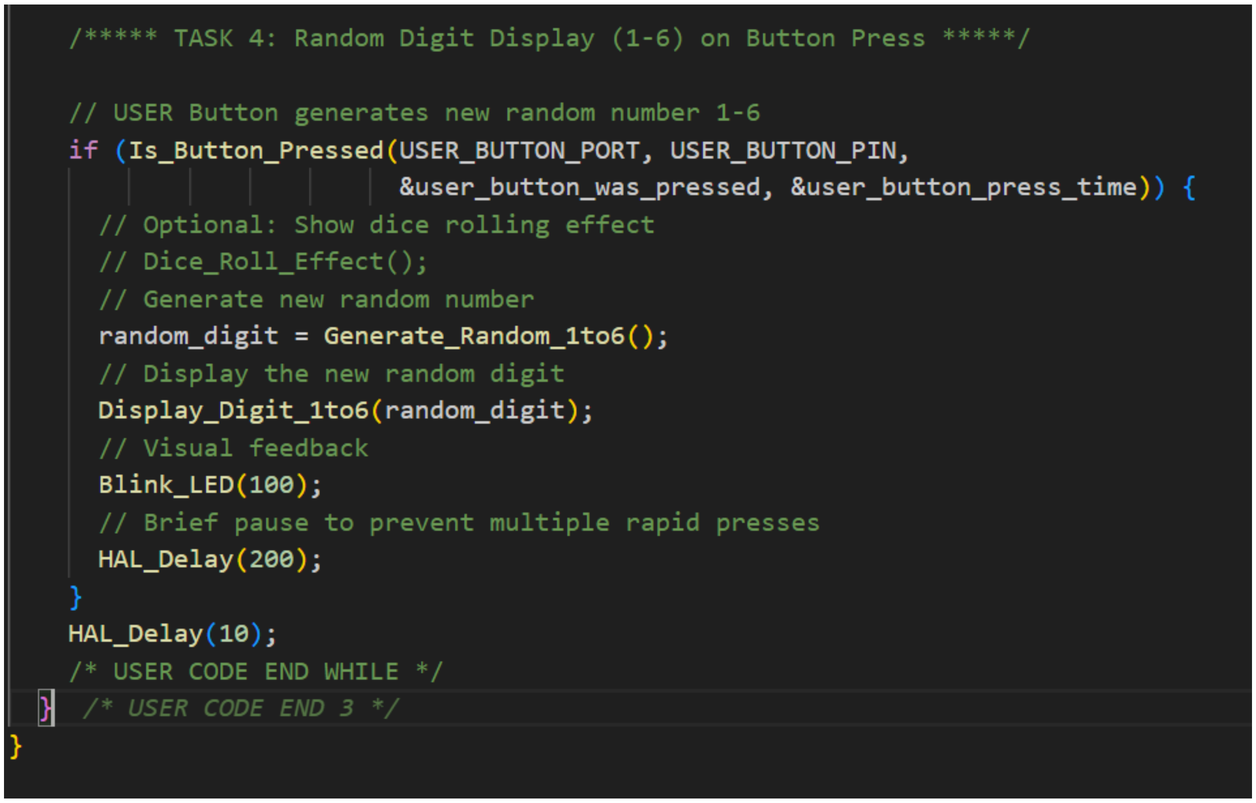


```c
void Init_Random(void)
{
  if (!random_initialized) {
    // Use system timer tick as seed
    srand(HAL_GetTick());
    random_initialized = true;
  }
}
/* Generate random number between 1 and 6 */
uint8_t Generate_Random_1to6(void)
{
  // Generate random number between 1 and 6
  return (rand() % 6) + 1;  // rand() % 6 gives 0-5, +1 gives 1-6
}
/* Visual effect for dice roll */
void Dice_Roll_Effect(void)
{
  // Quick cycle through numbers 1-6
  for(int i = 0; i < 3; i++) {
    for(uint8_t num = 1; num <= 6; num++) {
      Display_Digit_1to6(num);
      HAL_Delay(30);
    }
  }
}
```

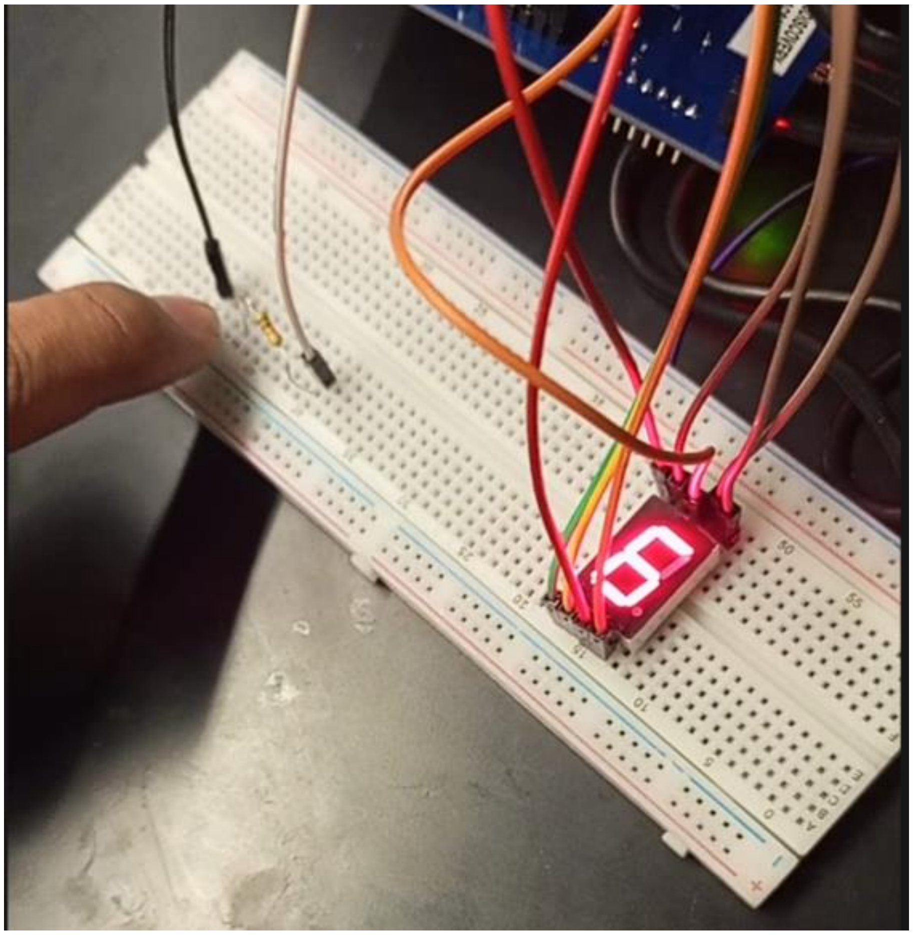


```c
/* ===== DEBUG FUNCTIONS ===== */
/* Toggle debug LED */
void Toggle_Debug_LED(void)
{
  HAL_GPIO_TogglePin(DEBUG_LED_PORT, DEBUG_LED_PIN);
}
/* Blink LED */
void Blink_LED(uint32_t duration_ms)
{
  HAL_GPIO_WritePin(DEBUG_LED_PORT, DEBUG_LED_PIN, GPIO_PIN_RESET);
  HAL_Delay(duration_ms);
  HAL_GPIO_WritePin(DEBUG_LED_PORT, DEBUG_LED_PIN, GPIO_PIN_SET);
}
```

Code:



```c
    /***** TASK 4: Random Digit Display (1-6) on Button Press *****/

    // USER Button generates new random number 1-6
    if (Is_Button_Pressed(USER_BUTTON_PORT, USER_BUTTON_PIN,
                          &user_button_was_pressed, &user_button_press_time)) {
      // Optional: Show dice rolling effect
      // Dice_Roll_Effect();
      // Generate new random number
      random_digit = Generate_Random_1to6();
      // Display the new random digit
      Display_Digit_1to6(random_digit);
      // Visual feedback
      Blink_LED(100);
      // Brief pause to prevent multiple rapid presses
      HAL_Delay(200);
    }
    HAL_Delay(10);
    /* USER CODE END WHILE */
  }  /* USER CODE END 3 */
}
```



What is the key difference between polling and interrupt-based GPIO input handling, and when would you prefer using interrupts?

Polling means the MCU keeps checking the GPIO pin in the main loop, which is easy but wastes CPU time and may miss quick changes. Interrupt-based handling uses EXTI to trigger an ISR only when the pin changes, so it's more efficient and responsive. You'd use interrupts when you want fast response, lower power use, or when the CPU has other work to do.

Suppose you're connecting a push button to a GPIO pin. Why is a pull-down resistor necessary, and what might happen if it's missing?

A pull-down resistor keeps the GPIO pin at a defined LOW level when the button is not pressed, so the input doesn't float. If it's missing, the pin can pick up noise and randomly read HIGH or LOW, causing false button presses or unstable behavior.