



**GITAM**  
DEEMED TO BE UNIVERSITY

## SUMMER INTERNSHIP EMBEDDED C

AYYAPU REDDY DANDE

BU21EECE0100105

### TASK 12

#### Hands-on Activity-1

- ☐ Write a program to count no. of bits which are set in given binary pattern
- ☐ Write a program to set 5<sup>th</sup> and 12<sup>th</sup> bits in a 16-bit unsigned integer
- ☐ Write a program to clear 6<sup>th</sup> and 19<sup>th</sup> bits in a 32-bit unsigned integer
- ☐ Write a program to flip even positioned bits in a 16-bit unsigned integer
- ☐ An IP Address will be in the form of "a.b.c.d" format, where a,b,c,d will be in the range of 0-255. Given a,b,c,d values (or string format) pack them into 32-bit unsigned integer.
- ☐ Given an unsigned 32-bit integer holding packed IPv4 address, convert it into "a.b.c.d" format.
- ☐ Convert MAC address into 48-bit binary pattern
- ☐ Convert 48-bit binary pattern as MAC address
- ☐ Arduino examples using Bare metal code (Register level Bit Manipulations)
  - Blinky
  - LED controlling using PushButton

**Q1) Write a program to count no. of bits which are set in given binary pattern.**

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

```
int countSetBits(int n) {
```

```

int count = 0;
while (n) {
    count += n & 1;
    n >>= 1;
}
return count;
}

int main() {
    int num;
    printf("Enter an integer: ");
    scanf("%d", &num);
    int setBits = countSetBits(num);
    printf("Number of set bits in %d is %d\n", num, setBits);
    return 0;
}

```

**Q2) Write a program to set 5<sup>th</sup> and 12<sup>th</sup> bit in a 16-bit unsigned integer.**

```

#include <stdio.h>

int main()
{
    unsigned short int value = 0;
    unsigned short int mask = (1 << 4) | (1 << 11);
    value |= mask;
    printf("The value after setting the 5th and 12th bits is: %u\n", value);
    return 0;
}

```

**Q3) Write a program to clear 6<sup>th</sup> and 19<sup>th</sup> bit in a 32-bit unsigned integer.**

```

#include <stdio.h>

unsigned int clearBits(unsigned int num) {
    unsigned int mask = ~((1 << 5) | (1 << 18));

```

```

        return num & mask;
    }

int main() {
    unsigned int num;

    printf("Enter a 32-bit unsigned integer: ");
    scanf("%u", &num);

    unsigned int result = clearBits(num);

    printf("Result after clearing the 6th and 19th bits: %u\n", result);

    return 0;
}

```

**Q4) Write a program to flip even positioned bits in a 16-bit unsigned integer**

```

#include <stdio.h>

unsigned short flipEvenBits(unsigned short num) {
    unsigned short mask = 0x5555;

    return num ^ mask;
}

int main() {
    unsigned short num;

    printf("Enter a 16-bit unsigned integer: ");
    scanf("%hu", &num);

    unsigned short result = flipEvenBits(num);

    printf("Result after flipping the even-positioned bits: %hu\n", result);

    return 0;
}

```

**Q5) An IP Address will be in the form of "a.b.c.d" format, where a,b,c,d will be in the range of 0-255. Given a,b,c,d values (or string format) pack them into 32-bit unsigned integer.**

```

#include <stdio.h>

unsigned int packIP(unsigned char a, unsigned char b, unsigned char c, unsigned char d) {
    return (a << 24) | (b << 16) | (c << 8) | d;
}

```

```
}
```

```
int main() {  
    unsigned char a = 192;  
    unsigned char b = 168;  
    unsigned char c = 1;  
    unsigned char d = 100;  
    unsigned int packedIP = packIP(a, b, c, d);  
    printf("Packed IP address: 0x%X\n", packedIP);  
  
    return 0;  
}
```

**Q6) Given an unsigned 32-bit integer holding packed IPv4 address, convert it into "a.b.c.d" format.**

```
#include <stdio.h>  
  
int main() {  
    unsigned int packed_ip = 0xC0A80164;  
    unsigned char a = (packed_ip >> 24) & 0xFF;  
    unsigned char b = (packed_ip >> 16) & 0xFF;  
    unsigned char c = (packed_ip >> 8) & 0xFF;  
    unsigned char d = packed_ip & 0xFF;  
    printf("The unpacked IP address is: %u.%u.%u.%u\n", a, b, c, d);  
    return 0;  
}
```

**Q7) Convert MAC address into 48-bit binary pattern**

```
#include <stdio.h>  
  
#include <stdlib.h>  
  
unsigned long long convertMACAddress(const char *mac) {  
    unsigned int bytes[6];
```

```

    if (sscanf(mac, "%x:%x:%x:%x:%x:%x", &bytes[0], &bytes[1], &bytes[2], &bytes[3], &bytes[4],
&bytes[5]) != 6) {
        fprintf(stderr, "Invalid MAC address format.\n");
        exit(EXIT_FAILURE);
    }
    unsigned long long macBinary = 0;
    for (int i = 0; i < 6; ++i) {
        macBinary = (macBinary << 8) | (bytes[i] & 0xFF);
    }
    return macBinary;
}

int main() {
    char macString[18];
    printf("Enter MAC address in the format XX:XX:XX:XX:XX:XX: ");
    if (scanf("%17s", macString) != 1) {
        fprintf(stderr, "Failed to read MAC address.\n");
        return EXIT_FAILURE;
    }
    unsigned long long macBinary = convertMACAddress(macString);
    printf("MAC address in 48-bit binary pattern: %012llx\n", macBinary);
    return 0;
}

```

#### Q8) Convert 48-bit binary pattern as MAC address

```

#include <stdio.h>

#include <stdlib.h>

void binaryToMac(const char* binary) {
    unsigned int bytes[6] = {0};
    for (int i = 0; i < 48; ++i) {
        bytes[i / 8] = (bytes[i / 8] << 1) | (binary[i] - '0');
    }
}

```

```

    }

    printf("MAC Address: %02X:%02X:%02X:%02X:%02X:%02X\n",
           bytes[0], bytes[1], bytes[2], bytes[3], bytes[4], bytes[5]);
}

int main() {
    const char* binary_pattern = "1010101010111011110011001101110111101111111111111";
    binaryToMac(binary_pattern);
    return 0;
}

```

## Task 14

### 1)bare metal blinky using arduino1

```

#define F_CPU 16000000UL
#include <avr/io.h>
#include <util/delay.h>

int main(void)
{
    // Set pin 7 (PD7) as an output
    DDRD |= (1 << PD7);

    while (1)
    {
        PORTD |= (1 << PD7);
        _delay_ms(1000);
        PORTD &= ~(1 << PD7);
        _delay_ms(1000);
    }

    return 0;
}

```

### 2)bare metal push button1

```

#define F_CPU 16000000UL

#include <avr/io.h>

#include <util/delay.h>

const uint8_t buttonPin = PD2;

const uint8_t ledPin = PB5;

uint8_t buttonState = 0;

void setup() {
    DDRD &= ~(1 << buttonPin);

    PORTD |= (1 << buttonPin);

    DDRB |= (1 << ledPin);
}

int main(void) {
    setup();

    while (1) {
        buttonState = PIND & (1 << buttonPin);

        if (buttonState) {
            PORTB |= (1 << ledPin);
        } else {
            PORTB &= ~(1 << ledPin);
        }

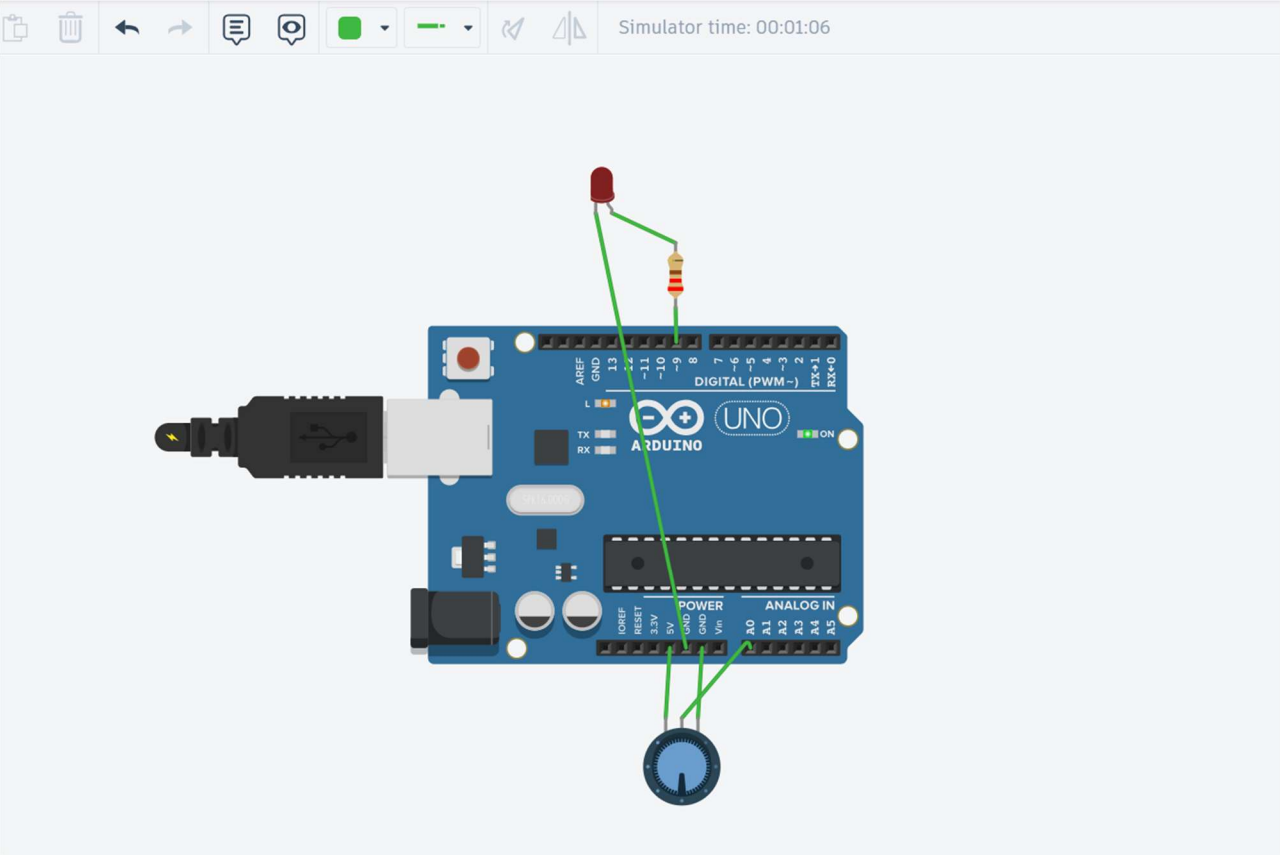
        _delay_ms(10);
    }

    return 0;
}

```

## Task 15

### Analog Read (Potentiometer)

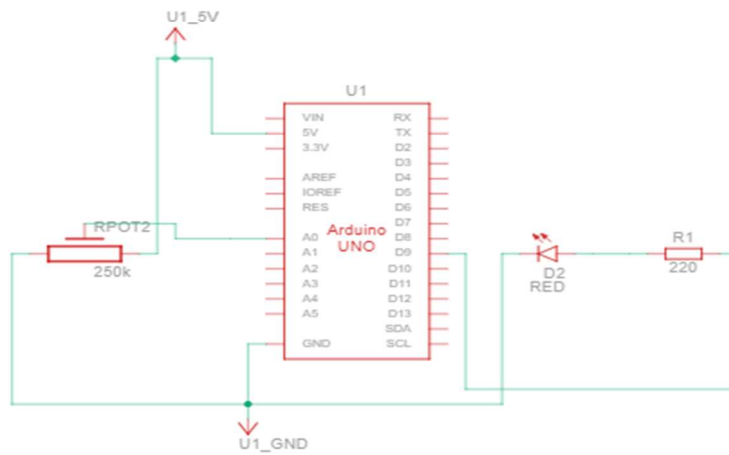
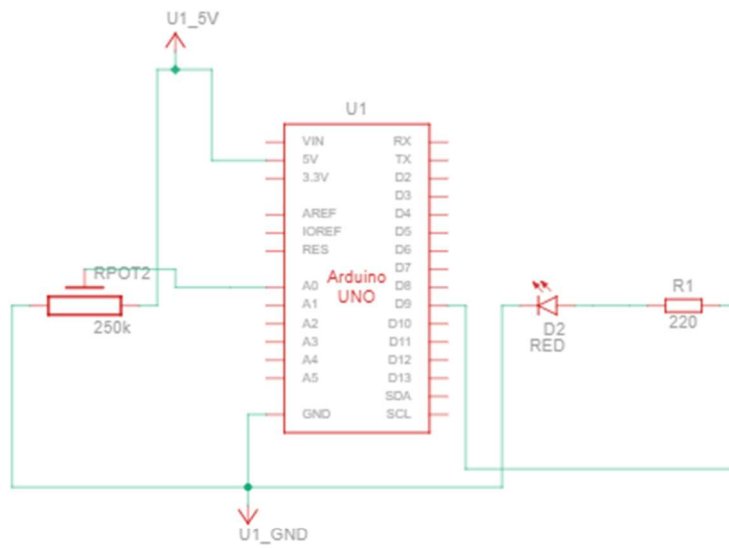


Component List

 Download CSV

Name	Quantity	Component
Rpot2	1	250 kΩ Potentiometer
U1	1	Arduino Uno R3
R1	1	220 Ω Resistor
D2	1	Red LED





```
// Constants for pin assignments
```

```
const int potPin = A0; // Pin where the potentiometer is connected
```

```
const int ledPin = 9; // Pin where the LED is connected
```

```
// Variable to store the potentiometer value
```

```
int potValue = 0;
```

```
void setup() {
```

```
    // Initialize the LED pin as an output
```

```
    pinMode(ledPin, OUTPUT);
```

```
}
```

```
void loop() {
```

```
    // Read the value from the potentiometer
```

```
    potValue = analogRead(potPin);
```

```
// Map the potentiometer value to the PWM range (0-255)
int ledValue = map(potValue, 0, 1023, 0, 255);

// Set the brightness of the LED
analogWrite(ledPin, ledValue);

// Small delay to smooth out the reading
delay(10);
}
```

## **Analout Output(fading)**

```
const int ledPin = 9;    // Pin where the LED is connected

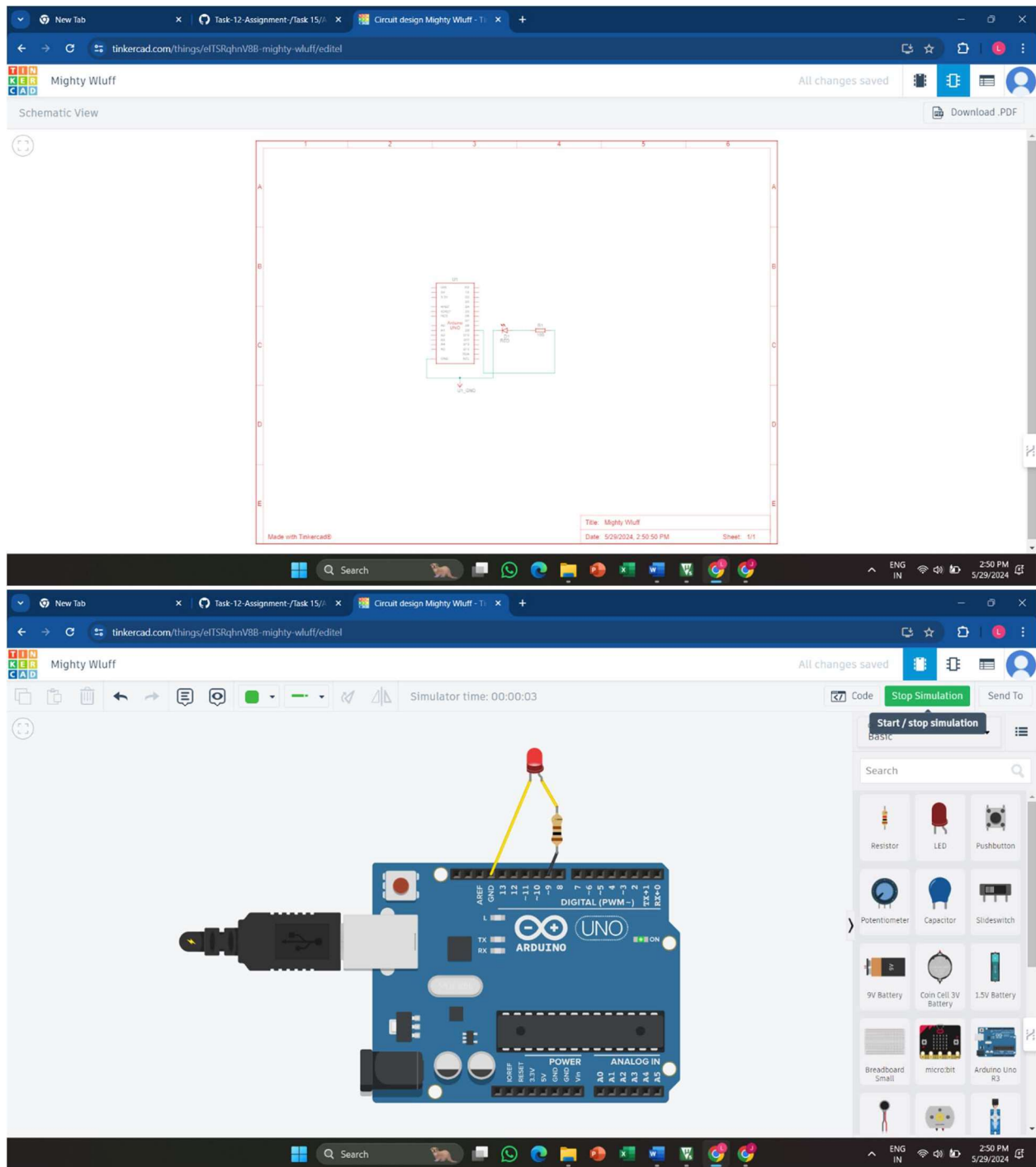
void setup() {
    // Initialize the LED pin as an output
    pinMode(ledPin, OUTPUT);
}

void loop() {
    // Fade in from 0 to 100^6
    for (int brightness = 0; brightness <= 100^6; brightness++) {
        analogWrite(ledPin, brightness); // Set the brightness
        delay(10); // Wait for 10 milliseconds
    }

    // Fade out from 100^6 to 0
    for (int brightness = 100^6; brightness >= 0; brightness--) {
        analogWrite(ledPin, brightness); // Set the brightness
        delay(10); // Wait for 10 milliseconds
    }
}
```

}

 $\frac{1}{2}$



## Digital Input using Interrupt

`const int buttonPin = 2; // Pin where the push button is connected`

`volatile bool buttonPressed = false; // Flag to indicate button press`

```
void setup() {  
    pinMode(buttonPin, INPUT);    // Set the button pin as input  
    attachInterrupt(digitalPinToInterrupt(buttonPin), buttonPressISR, RISING); // Attach interrupt on rising  
    edge  
    Serial.begin(9600);           // Initialize serial communication  
}  
  
void loop() {  
    if (buttonPressed) {  
        Serial.println("Button Pressed!"); // Print message when button is pressed  
        buttonPressed = false;           // Reset the flag  
    }  
}  
  
void buttonPressISR() {  
    buttonPressed = true; // Set the flag to indicate button press  
}
```

Browser tabs: New Tab, Task-12-Assignment-/task 15/, Circuit design Terrific Amur - T...

Address bar: tinkercad.com/things/jr3UsZsL9Od/edit?returnTo=%2Fclassrooms%2Fb7yCq4I566%2Factivities%2F9KjJSRsLOqI

Page title: Terrific Amur

Buttons: All changes saved, Download CSV

Component List

Name	Quantity	Component
U1	1	Arduino Uno R3
R1	1	10 kΩ Resistor
S1	1	Pushbutton

Windows taskbar: Search, ENG IN, 2:45 PM 5/29/2024

Browser tabs: New Tab, Task-12-Assignment-/task 15/, Circuit design Terrific Amur - T...

Address bar: tinkercad.com/things/jr3UsZsL9Od/edit?returnTo=%2Fclassrooms%2Fb7yCq4I566%2Factivities%2F9KjJSRsLOqI

Page title: Terrific Amur

Buttons: All changes saved, Download .PDF

Schematic View

Windows taskbar: Search, ENG IN, 2:44 PM 5/29/2024

