BCSL504L

Internet of Things Lab

Experiment 1:

Develop a program to Illustrate the working of LED with Push Button

```
const int buttonPin = 2;     // Push button connected to digital pin 2
const int ledPin = 13;       // LED connected to digital pin 13

bool ledState = false;
bool lastButtonState = LOW;

void setup() {
  pinMode(buttonPin, INPUT);
  pinMode(ledPin, OUTPUT);
  digitalWrite(ledPin, ledState);
}

void loop() {
  bool currentButtonState = digitalRead(buttonPin);

  // Detect rising edge: button goes from LOW to HIGH
  if (currentButtonState == HIGH && lastButtonState == LOW) {
    ledState = !ledState; // Toggle LED state
    digitalWrite(ledPin, ledState);
  }

  lastButtonState = currentButtonState;
}
```

Experiment 2 :

Develop a program to illustrate the working of traffic light for pedestrians

```
#define REDLED 5
#define GREENLED 6

void setup(){
  pinMode(REDLED, OUTPUT);
  pinMode(GREENLED, OUTPUT);
}

void loop() {
  // put your main code here, to run repeatedly:
digitalWrite(REDLED, HIGH); // RED signal
digitalWrite(GREENLED, LOW);
delay(10000);

digitalWrite(REDLED, HIGH); // Yellow Signal
digitalWrite(GREENLED, HIGH);
delay(1000);

digitalWrite(REDLED, LOW); // Green Signal
digitalWrite(GREENLED, HIGH);
delay(5000);

digitalWrite(REDLED, HIGH); // Yellow Signal
digitalWrite(GREENLED, HIGH);
delay(1000);
}
```

EXPERIMENT 3:

Develop a program for fading of LED

```
const int ledPin = 9; // PWM-capable pin on Arduino Uno/Nano/Mega

void setup() {
  pinMode(ledPin, OUTPUT);
}

void loop() {
  // Increase brightness from 0 to 255
  for (int dutyCycle = 0; dutyCycle <= 255; dutyCycle++) {
    analogWrite(ledPin, dutyCycle);
    delay(15);
  }

  // Decrease brightness from 255 to 0
  for (int dutyCycle = 255; dutyCycle >= 0; dutyCycle--) {
    analogWrite(ledPin, dutyCycle);
    delay(15);
  }
}
```

EXPERIMENT 4 :

Develop a program to blink 6 LEDs in ODD and Even Fashion

```
// Define an array containing pin numbers of SIX LEDs
#define LEDCOUNT 6
#define ONTIME 1500  //milliseconds
// ONTIME = PERIOD - OFFTIME
uint8_t leds[LEDCOUNT]={1,2,3,4,5,6}; //GPIO list in order
void setup() {
        for(uint8_t i=0;i<LEDCOUNT;i++){
          pinMode(leds[i],OUTPUT);
          digitalWrite(leds[i],0);
        }
}
void loop() {
        for(uint8_t i=0;i<LEDCOUNT;i++){
          if (i%2){
          digitalWrite(leds[i],HIGH);
          }
          else{digitalWrite(leds[i],LOW);}
        }
delay(ONTIME);
for(uint8_t i=0;i<LEDCOUNT;i++){
  if (!(i%2)){
  digitalWrite(leds[i],HIGH);
  }
  else{digitalWrite(leds[i],LOW);}
}
delay(ONTIME);
}
```

EXPERIMENT 5:

Develop a program to rotate a servo motor both in clockwise and anticlockwise direction

```
#include <Servo.h>
const int servoPin = 9; // Use a PWM-capable pin on Arduino Uno/Nano/Mega
Servo servo1;

void setup() {
  Serial.begin(9600);       // Optional: for debugging
  servo1.attach(servoPin);  // Attach servo to pin
}

void loop() {
  // Sweep from 0 to 180 degrees
  for (int posDegrees = 0; posDegrees <= 180; posDegrees++) {
    servo1.write(posDegrees);
    delay(50); // Smooth movement
  }

  // Sweep back from 180 to 0 degrees
  for (int posDegrees = 180; posDegrees >= 0; posDegrees--) {
    servo1.write(posDegrees);
    delay(10); // Faster return
  }
}
```

EXPERIMENT 6 :

Develop a program to simulate the interfacing of LDR with Arduino ad control the intensity of LED using LDR

```
const int LDRPin = A0;     // Analog pin connected to LDR
const int ledPin = 9;      // PWM-capable pin connected to LED

void setup() {
  Serial.begin(9600);
  pinMode(ledPin, OUTPUT);
}

void loop() {
  // Read LDR value (0 to 1023)
  int LDRValue = analogRead(LDRPin);
  Serial.print("LDR Value: ");
  Serial.println(LDRValue);

  // Map LDR value to PWM range (0 to 255)
  int pwmValue = map(LDRValue, 0, 1023, 0, 255);

  // Set LED brightness
  analogWrite(ledPin, pwmValue);

  delay(20); // Small delay for stability
}
```

EXPERIMENT 7:

Develop a program to simulate the working of potentiometer and LED by varying the intensity of LED using potentiometer

```
//Important : connect potentiometer GND to common GND
const int potPin = A0;     // Analog pin connected to potentiometer
const int ledPin = 9;      // PWM-capable pin connected to LED

void setup() {
  Serial.begin(9600);      // Standard baud rate for Arduino
  pinMode(ledPin, OUTPUT);
}

void loop() {
  // Read potentiometer value (0 - 1023)
  int potValue = analogRead(potPin);
  Serial.print("Potentiometer Value: ");
  Serial.println(potValue);

  // Map to PWM range (0 - 255)
  int pwmValue = map(potValue, 0, 1023, 0, 255);

  // Set LED brightness
  analogWrite(ledPin, pwmValue);

  delay(20); // Small delay for stability
}
```

EXPERIMENT 8:

Develop a program to simulate the working of LCD and print the room temperature value on LCD.

```
#include <Adafruit_GFX.h>
#include <Adafruit_ST7735.h>
#include <SPI.h>
#include "DHT.h"
#define DHT11PIN 4
#define LED 5
#define INTERVAL_MS 100 // 10 seconds
// Use available digital pins on Arduino Uno/Nano
#define TFT_CS    10  // Chip Select
#define TFT_DC     9  // Data/Command
#define TFT_RST    8  // Reset

Adafruit_ST7735 tft = Adafruit_ST7735(TFT_CS, TFT_DC, TFT_RST);
DHT dht(DHT11PIN, DHT11);

bool readDHT = false;
unsigned long lastReadTime = 0;

void setup() {
  Serial.begin(9600);
  dht.begin();
  pinMode(LED, OUTPUT);
  digitalWrite(LED, LOW);

  tft.initR(INITR_BLACKTAB);
  tft.fillScreen(ST7735_BLACK);
```

```
  tft.setTextColor(ST7735_WHITE);
  tft.setTextSize(1);
}

void loop() {
  if (millis() - lastReadTime >= INTERVAL_MS) {
    lastReadTime = millis();
    readDHT = true;
  }
  if (readDHT) {
    readDHT = false;
    digitalWrite(LED, HIGH);
    float humi = dht.readHumidity();
    float temp = dht.readTemperature();

    Serial.println("Temperature: " + String(temp) + " °C");
    Serial.println("Humidity: " + String(humi) + " %");

    tft.fillScreen(ST7735_BLACK);
    tft.setCursor(2, 10);
    tft.print("Temperature: " + String(temp) + " C");
    tft.setCursor(2, 30);
    tft.print("Humidity: " + String(humi) + " %");
    delay(1000);
    digitalWrite(LED, LOW);
  }
}
```

EXPERIMENT 9:

Develop a program for scrolling 5 LEDs back and forth.

```
const int ledPins[] = {2, 3, 4, 5, 6, 7}; // Digital pins connected to LEDs
const int numLEDs = 6;
const int delayTime = 500; // Delay in milliseconds

void setup() {
  for (int i = 0; i < numLEDs; i++) {
    pinMode(ledPins[i], OUTPUT);
    digitalWrite(ledPins[i], LOW);
  }
}

void loop() {
  // Scroll forward
  for (int i = 0; i < numLEDs; i++) {
    digitalWrite(ledPins[i], HIGH);
    delay(delayTime);
    digitalWrite(ledPins[i], LOW);
  }

  // Scroll backward
  for (int i = numLEDs - 1; i >= 0; i--) {
    digitalWrite(ledPins[i], HIGH);
    delay(delayTime);
    digitalWrite(ledPins[i], LOW);
  }
}
```

EXPERIMENT 10:

Develop a program to calculate the distance of an object using ultrasonic sensor.

```
const int trigPin = 4;
const int echoPin = 5;

#define SOUND_SPEED 0.0343
#define CM_TO_INCH 0.393701

long duration;
float distanceCm;
float distanceInch;

void setup() {
  Serial.begin(9600); // Recommended baud rate for Arduino Uno
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
}

void loop() {
  // Clear the trigPin
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);

  // Trigger the sensor
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
```

```
  // Measure the echo time
  duration = pulseIn(echoPin, HIGH);

  // Calculate distance in cm and inches
  distanceCm = duration * SOUND_SPEED / 2;
  distanceInch = distanceCm * CM_TO_INCH;

  // Output to Serial Monitor
  Serial.print("Distance (cm): ");
  Serial.println(distanceCm);
  Serial.print("Distance (inch): ");
  Serial.println(distanceInch);

  delay(1000);
}
```

EXPERIMENT 11:

Develop a program to detect the collision using infrared sensor

```
int irSensor = 2;   // IR sensor output connected to pin D2
int ledPin   = 13;  // LED connected to pin D13
int buzzer   = 8;   // Optional buzzer on pin D8

void setup() {
  pinMode(irSensor, INPUT);    // IR sensor input
  pinMode(ledPin, OUTPUT);     // LED output
  pinMode(buzzer, OUTPUT);     // Buzzer output
  Serial.begin(9600);          // For debugging
}

void loop() {
  int sensorValue = digitalRead(irSensor);

  if (sensorValue == LOW) {
    // IR sensor detects obstacle (active LOW)
    digitalWrite(ledPin, HIGH);    // Turn ON LED
    digitalWrite(buzzer, HIGH);    // Turn ON buzzer
    Serial.println("Collision detected!");
  } else {
    digitalWrite(ledPin, LOW);     // Turn OFF LED
    digitalWrite(buzzer, LOW);     // Turn OFF buzzer
    Serial.println("No collision");
  }

  delay(100); // Small delay for stability
}
```

EXPERIMENT 12:

Develop a program to interface temperature sensor to read the room temperature , humidity, and heat index and print the readings on the serial monitor.

```
#include <DHT.h>
#define DHTPIN 4        // Digital pin connected to the DHT sensor
#define DHTTYPE DHT11   // Change to DHT22 if you're using that model

DHT dht(DHTPIN, DHTTYPE);

void setup() {
  Serial.begin(9600);
  dht.begin();
}

void loop() {
  // Read humidity and temperature
  float humidity = dht.readHumidity();
  float temperature = dht.readTemperature(); // Celsius
  float heatIndex = dht.computeHeatIndex(temperature, humidity, false); // false = Celsius

  // Check if readings are valid
  if (isnan(humidity) || isnan(temperature)) {
    Serial.println("Failed to read from DHT sensor!");
    return;
  }
```

```
  // Display readings
  Serial.print("Temperature (°C): ");
  Serial.println(temperature);
  Serial.print("Humidity (%): ");
  Serial.println(humidity);
  Serial.print("Heat Index (°C): ");
  Serial.println(heatIndex);
  Serial.println("----------------------");

  delay(2000); // Wait 2 seconds before next reading
}
```