// IOTL Experiment 4: Basic LED Blink

const int ledPin4 = 12;

void setup() {

pinMode(ledPin4, OUTPUT);

}

void loop() {

digitalWrite(ledPin4, HIGH);

delay(1000);

digitalWrite(ledPin4, LOW);

delay(1000);

}

// IOTL Experiment 5: Ultrasonic Sensor with LED

#define echoPin 2

#define trigPin 3

#define ledPin5 13

long duration;

int distance;

void setup() {

pinMode(trigPin, OUTPUT);

pinMode(echoPin, INPUT);

pinMode(ledPin5, OUTPUT);

Serial.begin(9600);

Serial.println("Distance measurement using Arduino Uno:");

}

void loop() {

digitalWrite(trigPin, LOW);

delayMicroseconds(2);

digitalWrite(trigPin, HIGH);

delayMicroseconds(10);

digitalWrite(trigPin, LOW);

duration = pulseIn(echoPin, HIGH);

distance = duration \* 0.0344 / 2;

Serial.print("Distance: ");

Serial.print(distance);

Serial.println(" cm");

if (distance <= 100) {

digitalWrite(ledPin5, HIGH);

} else {

digitalWrite(ledPin5, LOW);

}

delay(200);

}

// IOTL Experiment 6: LED Traffic Light Based on Counter

const int greenLED6 = 9;

const int yellowLED6 = 12;

const int redLED6 = 11;

int counter6 = 0;

void setup() {

pinMode(greenLED6, OUTPUT);

pinMode(yellowLED6, OUTPUT);

pinMode(redLED6, OUTPUT);

}

void loop() {

counter6++;

if (counter6 > 255) {

counter6 = 0;

}

digitalWrite(greenLED6, LOW);

digitalWrite(yellowLED6, LOW);

digitalWrite(redLED6, LOW);

if (counter6 < 100) {

digitalWrite(greenLED6, HIGH);

} else if (counter6 >= 101 && counter6 <= 200) {

digitalWrite(yellowLED6, HIGH);

} else {

digitalWrite(redLED6, HIGH);

}

delay(20);

}

// IOTL Experiment 7: LED Control via Serial Input

const int greenPin7 = 6;

const int yellowPin7 = 2;

const int redPin7 = 3;

void setup() {

Serial.begin(9600);

pinMode(greenPin7, OUTPUT);

pinMode(yellowPin7, OUTPUT);

pinMode(redPin7, OUTPUT);

digitalWrite(greenPin7, LOW);

digitalWrite(yellowPin7, LOW);

digitalWrite(redPin7, LOW);

}

void loop() {

if (Serial.available() > 0) {

char input = Serial.read();

if (input == 'b') {

blinkGreenLight();

} else if (input == 'g') {

illuminateGreen();

} else if (input == 'r') {

illuminateRed();

} else if (input == 'y') {

illuminateYellow();

}

}

}

void blinkGreenLight() {

for (int i = 0; i < 20; i++) {

digitalWrite(greenPin7, HIGH);

delay(500);

digitalWrite(greenPin7, LOW);

delay(500);

}

}

void illuminateGreen() {

digitalWrite(greenPin7, HIGH);

digitalWrite(yellowPin7, LOW);

digitalWrite(redPin7, LOW);

}

void illuminateRed() {

digitalWrite(greenPin7, LOW);

digitalWrite(yellowPin7, LOW);

digitalWrite(redPin7, HIGH);

}

void illuminateYellow() {

digitalWrite(greenPin7, LOW);

digitalWrite(yellowPin7, HIGH);

digitalWrite(redPin7, LOW);

}

// IOTL Experiment 8: Logic Gates - AND

const int inputPin1A = 2;

const int inputPin2A = 3;

const int outputPinA = 13;

void setup() {

pinMode(inputPin1A, INPUT);

pinMode(inputPin2A, INPUT);

pinMode(outputPinA, OUTPUT);

}

void loop() {

int inputstate1 = digitalRead(inputPin1A);

int inputstate2 = digitalRead(inputPin2A);

if (inputstate1 == HIGH && inputstate2 == HIGH) {

digitalWrite(outputPinA, HIGH);

} else {

digitalWrite(outputPinA, LOW);

}

delay(100);

}

// IOTL Experiment 8: Logic Gates - OR

const int inputPin1O = 2;

const int inputPin2O = 3;

const int outputPinO = 13;

void setup() {

pinMode(inputPin1O, INPUT);

pinMode(inputPin2O, INPUT);

pinMode(outputPinO, OUTPUT);

}

void loop() {

int inputstate1 = digitalRead(inputPin1O);

int inputstate2 = digitalRead(inputPin2O);

if (inputstate1 == HIGH || inputstate2 == HIGH) {

digitalWrite(outputPinO, HIGH);

} else {

digitalWrite(outputPinO, LOW);

}

delay(100);

}

// IOTL Experiment 9: Square a Number

void setup() {

Serial.begin(9600);

while (!Serial);

Serial.println("Please enter a no.=");

}

void loop() {

if (Serial.available() > 0) {

int number = Serial.parseInt();

int squared = number \* number;

Serial.print("The square of ");

Serial.print(number);

Serial.print(" is ");

Serial.println(squared);

delay(1000);

Serial.println("Please enter a new number: ");

}

}

// IOTL Experiment 10: Square Root

void setup() {

Serial.begin(9600);

while (!Serial);

Serial.println("Please enter a no.=");

}

void loop() {

if (Serial.available() > 0) {

float number = Serial.parseFloat();

float result = sqrt(number);

Serial.print("The square root of ");

Serial.print(number);

Serial.print(" is ");

Serial.println(result);

delay(1000);

Serial.println("Please enter a new number: ");

}

}

// IOTL Experiment 11: Cube Root

void setup() {

Serial.begin(9600);

while (!Serial);

Serial.println("Please enter a no.=");

}

void loop() {

if (Serial.available() > 0) {

float number = Serial.parseFloat();

float result = cbrt(number);

Serial.print("The cube root of ");

Serial.print(number);

Serial.print(" is ");

Serial.println(result);

delay(1000);

Serial.println("Please enter a new number: ");

}

}

// IOTL Experiment 12: Cube

void setup() {

Serial.begin(9600);

while (!Serial);

Serial.println("Please enter a no.=");

}

void loop() {

if (Serial.available() > 0) {

int number = Serial.parseInt();

int cube = number \* number \* number;

Serial.print("The cube of ");

Serial.print(number);

Serial.print(" is ");

Serial.println(cube);

delay(1000);

Serial.println("Please enter a new number: ");

}

}

// IOTL Experiment 13: Potentiometer-Controlled Brightness

const int redPin13 = 9;

const int greenPin13 = 10;

const int yellowPin13 = 11;

const int potPin = A0;

void setup() {

pinMode(redPin13, OUTPUT);

pinMode(greenPin13, OUTPUT);

pinMode(yellowPin13, OUTPUT);

Serial.begin(9600);

}

void loop() {

int potValue = analogRead(potPin);

int brightness = map(potValue, 0, 1023, 0, 255);

analogWrite(redPin13, brightness);

analogWrite(greenPin13, brightness);

analogWrite(yellowPin13, brightness);

Serial.print("Potentiometer Value=");

Serial.println(potValue);

delay(100);

}

// IOTL Experiment 14: PIR Motion Sensor

const int pirsensorpin = 7;

const int ledpin14 = 13;

void setup() {

pinMode(pirsensorpin, INPUT);

pinMode(ledpin14, OUTPUT);

Serial.begin(9600);

}

void loop() {

int motionDetected = digitalRead(pirsensorpin);

Serial.print("PIR sensor state: ");

Serial.println(motionDetected);

if (motionDetected == HIGH) {

digitalWrite(ledpin14, HIGH);

Serial.println("Motion Detected!");

} else {

digitalWrite(ledpin14, LOW);

Serial.println("No motion detected.");

}

delay(500);

}

IOTL Arduino Experiments Overview

---

🔹 Experiment 4: LED Blinking

1. Code Structure:

setup(): Sets LED pin as OUTPUT.

loop(): LED ON for 1s, OFF for 1s.

2. Sensor/Module Usage:

Component: LED

Pin: D12 (user-defined)

3. Connections:

D12 → Resistor → Anode of LED

Cathode → GND

4. Expected Inputs/Outputs:

Input: None

Output: LED blinks at 1-second intervals

5. Real-Time Application:

Basic digital output control. Used in indicators, signaling systems.

---

🔹 Experiment 5: Distance-based LED Trigger (Ultrasonic Sensor)

1. Code Structure:

setup(): Initializes trig/echo and LED.

loop(): Measures distance, turns LED ON if distance <= 100cm.

2. Sensor/Module Usage:

HC-SR04 Ultrasonic Sensor (trig, echo)

LED (Pin 13)

3. Connections:

Trig → D3, Echo → D2

LED Anode → D13 (through resistor), Cathode → GND

4. Expected Inputs/Outputs:

Input: Distance (from HC-SR04)

Output: LED turns ON if distance ≤ 100cm

5. Real-Time Application:

Proximity-based triggering (e.g., parking sensors)

---

🔹 Experiment 6: LED Color Based on Counter

1. Code Structure:

setup(): LED pins set as OUTPUT

loop(): Counter drives different LEDs

2. Sensor/Module Usage:

3 LEDs (green, yellow, red)

3. Connections:

D9 → Green LED, D12 → Yellow LED, D11 → Red LED (via resistors)

All cathodes → GND

4. Expected Inputs/Outputs:

Input: Counter variable

Output: LED ON based on range (green <100, yellow 101–200, red >200)

5. Real-Time Application:

Visual range indication systems (e.g., battery level)

---

🔹 Experiment 7: Serial-Controlled LEDs

1. Code Structure:

loop(): Reads character from serial and calls function

Functions control LEDs based on command

2. Sensor/Module Usage:

LEDs: Green, Yellow, Red

Serial Communication

3. Connections:

D6 → Green, D2 → Yellow, D3 → Red

4. Expected Inputs/Outputs:

Input: Serial commands ('g', 'r', 'y', 'b')

Output: LED lights/blink

5. Real-Time Application:

Remote LED control or testing modules via serial

---

🔹 Experiment 8: AND/OR Logic Gate Simulation

1. Code Structure:

Reads two digital inputs, applies logic, outputs LED result

2. Sensor/Module Usage:

Push Buttons (for input)

LED (for output)

3. Connections:

Buttons: D2 & D3 with pull-down resistors

LED: D13

4. Expected Inputs/Outputs:

Input: Button press states

Output: LED ON if logic condition met

5. Real-Time Application:

Digital logic gate simulation, basic input systems

---

🔹 Experiment 9: Square of Serial Input

1. Code Structure:

Takes number via Serial, returns square

2. Sensor/Module Usage:

Serial Monitor

3. Connections:

None (via USB)

4. Expected Inputs/Outputs:

Input: Number from Serial

Output: Square displayed

5. Real-Time Application:

Educational tools/calculators

---

🔹 Experiment 10: Square Root of Serial Input

Similar to Experiment 9 but returns square root.

---

🔹 Experiment 11: Cube Root of Serial Input

Similar structure. Returns cube root of input.

---

🔹 Experiment 12: Cube of Serial Input

Computes cube of number via Serial input.

---

🔹 Experiment 13: LED Brightness with Potentiometer

1. Code Structure:

analogRead() from pot mapped to PWM brightness

2. Sensor/Module Usage:

Potentiometer, 3 LEDs (RGB or separate)

3. Connections:

A0 → Pot center pin

LED pins: D9, D10, D11 (via resistors)

4. Expected Inputs/Outputs:

Input: Potentiometer rotation

Output: Varying LED brightness

5. Real-Time Application:

Dimmers, volume knobs, etc.

---

🔹 Experiment 14: Motion Detection with PIR Sensor

1. Code Structure:

Reads PIR value, controls LED

2. Sensor/Module Usage:

PIR Sensor, LED

3. Connections:

PIR OUT → D7, LED → D13

4. Expected Inputs/Outputs:

Input: Motion sensed by PIR

Output: LED ON when motion detected

5. Real-Time Application:

Intruder alarms, smart lighting