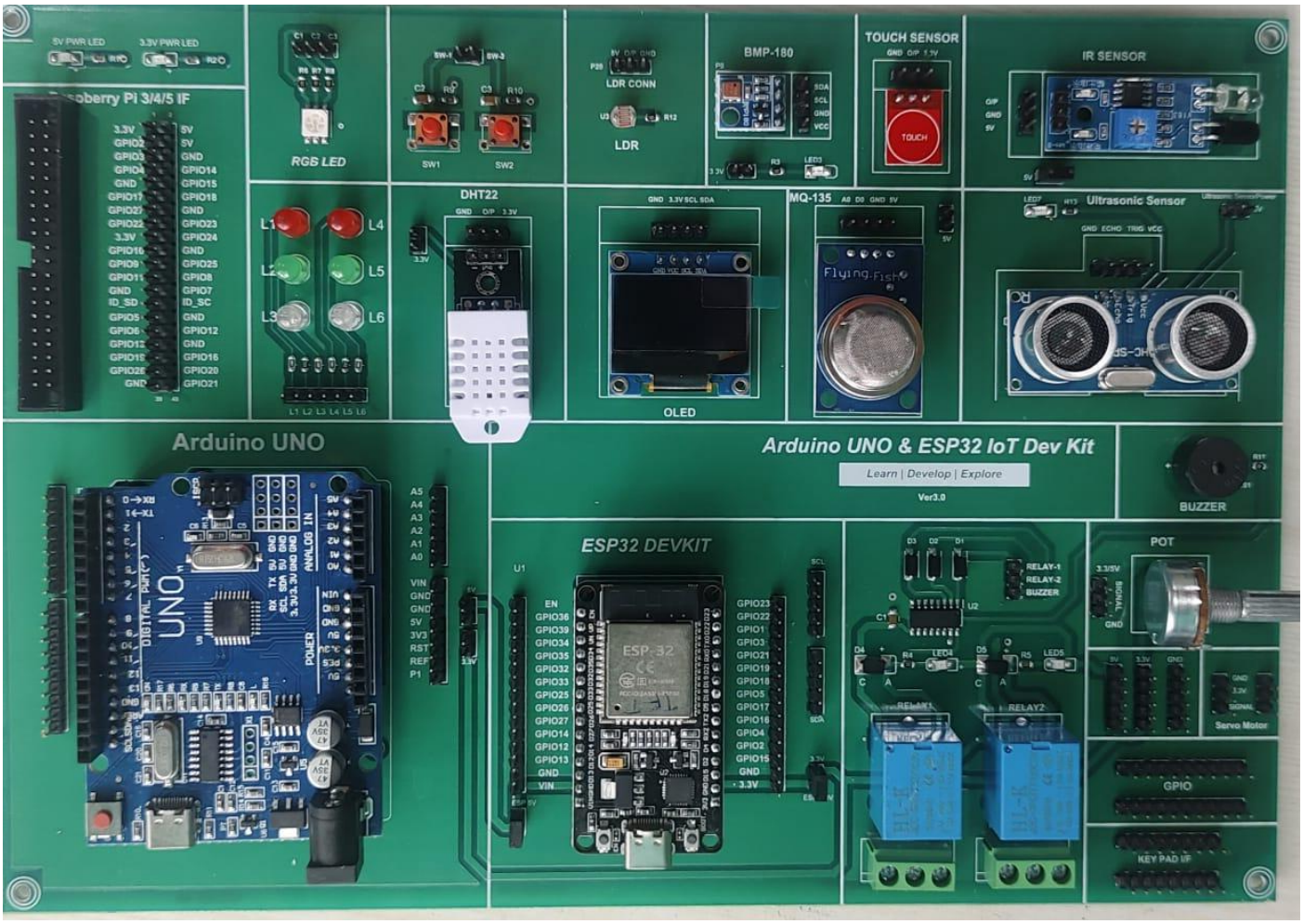
IoT Trainer Kit Manual



List of Experiments:

S.No	Experiment Description
1	Make your own parallel and series circuits using breadboard for any application of your choice.
2	Demonstrate a traffic light circuit using breadboard.
3	Build and demonstrate automatic Street Light using LDR.
4	Simulate the Arduino LED blinking activity in Tinkercad.
5	Build and demonstrate an Arduino LED blinking activity using Arduino IDE.
6	Interfacing IR Sensor and Servo Motor with Arduino.
7	Blink LED using ESP32.
8	LDR Interfacing with ESP32.
9	Control an LED using Mobile App.
10	Design and 3D print a Walking Robot.
11	Design and 3D Print a Rocket.
12	Build a live soil moisture monitoring project, and monitor soil moisture levels of a remote plant in your computer dashboard.
13	Demonstrate all the steps in design thinking to redesign a motor bike.

✅ 1. Install Arduino IDE

1. Open your browser and go to:
   👉 https://www.arduino.cc/en/software
2. Download the Arduino IDE for your operating system (Windows, Mac, or Linux).
3. Run the downloaded file and follow the on-screen installation instructions.
4. After installation, open the Arduino IDE.

✅ 2. Install ESP32 Board Manager

1. Open the Arduino IDE.
2. Go to File → Preferences.
3. In the "Additional Board Manager URLs" field, paste this URL:

https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_index.json

4. Click OK.

✅ 3. Add ESP32 Board in Board Manager

1. Go to Tools → Board → Boards Manager.
2. In the search box, type esp32.
3. Select "esp32 by Espressif Systems", and click Install.
4. Wait for installation to complete.

🖥️ Selecting Your Board for Projects

- For Arduino Uno Projects
  1. Go to Tools → Board → Arduino AVR Boards → Arduino Uno.
- For ESP32 Projects (IoT Experiments)
  1. Go to Tools → Board → ESP32 Arduino → DOIT ESP32 DEVKIT V1.

✅ 4. Select the Right COM Port

1. Connect your board to your PC using a USB cable.
2. Go to Tools → Port, and select the port labeled (Arduino Uno) or (ESP32).
   - Example: COM3 (Arduino Uno) or COM5 (DOIT ESP32 DEVKIT V1)

Experiment 1: Series & Parallel Circuits Using RGB LED

Objective:
Understand series and parallel con

cepts using the RGB and traffic LEDs available in the trainer kit.

Theory:
Series connection: Components share the same current. Parallel: Components share the same voltage. In the trainer kit, common GND and control pins let you emulate parallel and series logic by selectively turning ON combinations of LEDs.

Components Required:

- RGB LED (Common Anode)
- 3 Traffic LEDs

Pin Connections:

LED Color	ESP32 Pin	Arduino UNO Pin
Red	GPIO 16	D5
Green	GPIO 17	D6
Blue	GPIO 5	D4

Code:

```
#define RED_PIN 5     // GPIO 16 ESP32
#define GREEN_PIN 6   // GPIO 17 ESP32
#define BLUE_PIN 4    // GPIO 5 ESP32

void setup() {
  pinMode(RED_PIN, OUTPUT);
  pinMode(GREEN_PIN, OUTPUT);
  pinMode(BLUE_PIN, OUTPUT);
}

void loop() {
  // Emulate Series: One LED at a time
  digitalWrite(RED_PIN, HIGH);
  digitalWrite(GREEN_PIN, LOW);
  digitalWrite(BLUE_PIN, LOW);
  delay(1000);
  digitalWrite(RED_PIN, LOW);
```

```
  digitalWrite(GREEN_PIN, HIGH);
  digitalWrite(BLUE_PIN, LOW);
  delay(1000);
  digitalWrite(RED_PIN, LOW);
  digitalWrite(GREEN_PIN, LOW);
  digitalWrite(BLUE_PIN, HIGH);
  delay(1000);
  // Emulate Parallel: All LEDs together
  digitalWrite(RED_PIN, HIGH);
  digitalWrite(GREEN_PIN, HIGH);
  digitalWrite(BLUE_PIN, HIGH);
  delay(1000);
}
```

Experiment 2: Traffic Light Circuit Using Breadboard

Objective:
Simulate traffic lights using 3 LEDs with time-based switching.

Theory:
Traffic lights change in cycles: RED → GREEN → YELLOW, using delays and digital outputs.

Components Required:

- 3 LEDs (Red, Yellow, Green)

Pin Connections:

LED	ESP32 Pin	Arduino UNO Pin
Red	GPIO 16	D5
Yellow	GPIO 17	D6
Green	GPIO 5	D4

Code:

```
#define RED 5     // GPIO 16
#define YELLOW 6  // GPIO 17
#define GREEN 4   // GPIO 5

void setup() {
  pinMode(RED, OUTPUT);
  pinMode(YELLOW, OUTPUT);
  pinMode(GREEN, OUTPUT);
}

void loop() {
  digitalWrite(RED, HIGH);
  delay(2000);
  digitalWrite(RED, LOW);

  digitalWrite(GREEN, HIGH);
  delay(2000);
  digitalWrite(GREEN, LOW);

  digitalWrite(YELLOW, HIGH);
  delay(1000);
```

```
  digitalWrite(YELLOW, LOW);

}
```

💡 Adjust delays to simulate real-world timing.

Experiment 3: Automatic Street Light Using LDR

Objective:
Automatically turn ON light in darkness using LDR.

Theory:
LDR (Light Dependent Resistor) decreases its resistance in light and increases it in darkness. A microcontroller reads analog voltage and activates a relay or LED based on brightness.

Components Required:

- LDR
- LED or relay
- ESP32 or Arduino UNO

Pin Connections:

Component	ESP32 Pin	Arduino UNO Pin
LDR	GPIO 36	A0
LED	GPIO 2	D2

Code:

```
#define LDR_PIN A0  // GPIO 36 for ESP32, A0 for UNO
#define LED_PIN 2

void setup() {
  pinMode(LED_PIN, OUTPUT);
  Serial.begin(115200);
}
void loop() {
  int ldrValue = analogRead(LDR_PIN);
  Serial.println(ldrValue);

  if (ldrValue < 10) {
    digitalWrite(LED_PIN, HIGH); // Night
  } else {
    digitalWrite(LED_PIN, LOW); // Day
  }
  delay(500);
}
```

💡 Use appropriate analog input pin for your board (GPIO 36 for ESP32, A0 for UNO).

Experiment 4: LED Blinking in Tinkercad

Objective:
Simulate LED blinking using Arduino UNO in Tinkercad.

Theory:
Blinking LEDs is a basic digital output task used to demonstrate microcontroller timing and control using simple digitalWrite() and delay().

Components Required (Tinkercad):

- Arduino UNO (virtual)
- LED
- 220Ω resistor

Steps to Simulate LED Blinking in Tinkercad

1. Open Tinkercad

- Go to 👉 https://www.tinkercad.com
- Click "Join Now" (or "Sign In" if you already have an account).
- Log in using your Google or email ID.

2. Create a New Circuit

- After logging in, click on “Circuits” in the left menu.
- Click “Create new Circuit”.

3. Add Components

From the components panel:

- Drag and drop Arduino Uno R3 onto the canvas.
- Drag and drop an LED.
- Drag and drop a 220Ω resistor.
- Connect the long leg (anode) of the LED to pin 4 of Arduino.
- Connect the short leg (cathode) to one end of the resistor.
- Connect the other end of the resistor to GND on the Arduino.

📝 Tip: Use the “Rotate” option if needed to position components neatly.

4. Add Code (Block or Text)

- Click on the “Code” button (top-right).
- Switch to “Text” mode (optional for C code learners).
- Replace existing code with this:

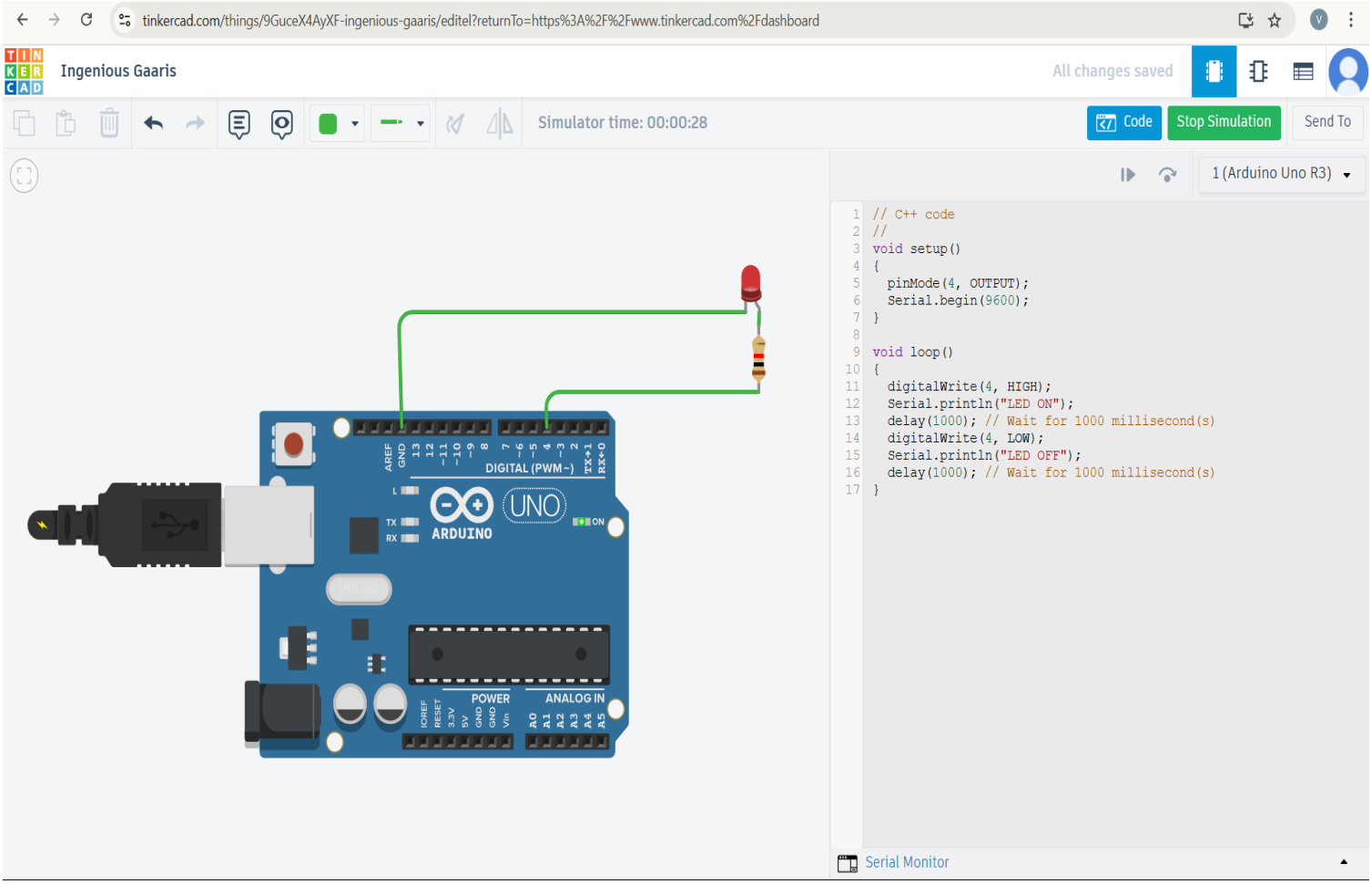
Code :

```
void setup() {
  pinMode(13, OUTPUT);  // Set pin 13 as output
}

void loop() {
  digitalWrite(13, HIGH);  // Turn LED ON
  delay(1000);             // Wait for 1 second
  digitalWrite(13, LOW);   // Turn LED OFF
  delay(1000);             // Wait for 1 second
}
```

5. Start Simulation

- Click “Start Simulation” at the top.
- The LED will now blink ON and OFF every second.



Experiment 5: LED Blinking Using Arduino IDE

Objective:
Demonstrate basic LED blinking using Arduino IDE with ESP32 or Arduino UNO.

Theory:
This is the first step in understanding digital output using microcontrollers. It teaches programming structure and timing.

Components Required:

- LED
- ESP32 or Arduino UNO

Pin Connections:

Component	ESP32 Pin	Arduino UNO Pin
LED	GPIO 2	D2

Code:

```
#define LED_PIN 2

void setup() {

  pinMode(LED_PIN, OUTPUT);

}


void loop() {

  digitalWrite(LED_PIN, HIGH);

  delay(1000);

  digitalWrite(LED_PIN, LOW);

  delay(1000);

}
```

💡 Adjust delay for faster/slower blinking.

Experiment 6: IR Sensor and Servo Motor with Arduino

Objective:
Trigger servo motor movement on IR detection.

Theory:
IR sensor detects object proximity and sends HIGH/LOW signal. Servo rotates based on this signal.

Components Required:

- IR sensor
- Servo motor
- ESP32 or Arduino UNO

Pin Connections:

Component	ESP32 Pin	Arduino UNO Pin
IR Sensor	GPIO 15	D7
Servo Motor	GPIO 14	D9

Code for Arduino UNO:

```
#include <Servo.h>


#define IR_PIN 7
#define SERVO_PIN 9


Servo myServo;


void setup() {
  pinMode(IR_PIN, INPUT);
  myServo.attach(SERVO_PIN);
}


void loop() {
  int val = digitalRead(IR_PIN);
  if (val == LOW) {
    myServo.write(90);
  } else {
    myServo.write(0);
  }
  delay(500);
```

```
}
```

💡 IR output is LOW when object detected.

Code for ESP32 :

```
#include <ESP32Servo.h>


#define IR_PIN 15
#define SERVO_PIN 14


Servo myServo;


void setup() {
  pinMode(IR_PIN, INPUT);
  myServo.attach(SERVO_PIN);
}


void loop() {
  int val = digitalRead(IR_PIN);
  if (val == LOW) {
    myServo.write(90);
  } else {
    myServo.write(0);
  }
  delay(500);
}
```

Experiment 7: Blink LED using ESP32

Objective:
Blink an onboard or external LED using ESP32.

Theory:
Microcontroller GPIO pins can drive LEDs directly using digitalWrite().

Components Required:

- ESP32 Board
- LED

Pin Connections:

Component	ESP32 Pin	Arduino UNO Pin
LED	GPIO 2	D2

Code:

```
#define LED 2
void setup() {
  pinMode(LED, OUTPUT);
}

void loop() {
  digitalWrite(LED, HIGH);
  delay(500);
  digitalWrite(LED, LOW);
  delay(500);
}
```

💡 Use onboard LED if available.

Experiment 8: LDR Interfacing with ESP32

Objective:
Measure light intensity using LDR on ESP32.

Theory:
LDR changes resistance with light. ESP32 reads analog voltage to measure brightness.

Components Required:

- LDR
- ESP32 Board

Pin Connections:

Component	ESP32 Pin	Arduino UNO Pin
LDR	GPIO 15	A0

Code:

```
#define LDR_PIN GPIO15  // GPIO A0 for ESP32


void setup() {
  Serial.begin(115200);
}


void loop() {
  int val = analogRead(LDR_PIN);
  Serial.println(val);
  delay(500);
}
```

💡 Readings decrease in bright light.

Experiment 9: Control LED Using Mobile App (Blynk)

Objective:
Control an LED using Blynk mobile application.

Theory:
Blynk allows controlling devices over internet using virtual pins.

Components Required:

- ESP32
- LED
- Blynk App

Pin Connections:

Component	ESP32 Pin	Arduino UNO Pin
LED	GPIO 2	D2

Code (Blynk Cloud):

```
#define BLYNK_TEMPLATE_ID "TMPL3jf4oigNl"
#define BLYNK_TEMPLATE_NAME "dht11 led"
#define BLYNK_AUTH_TOKEN "RSO8muGlW2LHlTc3z-zp15BS1yA7PjLw"

#define BLYNK_PRINT Serial
#include <WiFi.h>
#include <WiFiClient.h>
#include <BlynkSimpleEsp32.h>

char auth[] = BLYNK_AUTH_TOKEN;  // Use the authentication token from your Blynk project
char ssid[] = "IobiT Solutions";
char pass[] = "IobiT@2025";

#define RELAY_PIN 2  // Digital pin connected to the relay control

void setup() {
  Blynk.begin(auth, ssid, pass);
  pinMode(RELAY_PIN, OUTPUT);
}

void loop() {
  Blynk.run();
}

BLYNK_WRITE(2) {
  int relayState = param.asInt();
  digitalWrite(RELAY_PIN, relayState);
}
```

💡 Use switch widget on Pin 2 in Blynk app.

Steps:

- **Open Blynk Web Dashboard**
  - Go to https://blynk.cloud
  - Login with your account.
- **Create a New Template**
  - Template Name: Switch Control
  - Hardware: ESP32
  - Connection Type: WiFi
- **Create a New DataStream**
  - Name: Device Switch
  - Type: Digital Pin
  - Virtual Pin: 2
  - Data Type: Integer
  - Min: 0, Max: 1
- **Design Dashboard**
  - Add a Switch widget to the canvas.
  - Link it to Devise Switch
- **Get Credentials**
  - Note down the
    - BLYNK_TEMPLATE_ID
    - BLYNK_DEVICE_NAME
    - BLYNK_AUTH_TOKEN and Replace these three lines in code.
- **Upload the Above Code to ESP32**

Experiment 10: 3D Print a Walking Robot

Objective:
Design and print parts of a simple walking robot.

Theory:
Robots Walk by combining servo-controlled legs with balance. 3D printing provides a custom body.

Components Required:

- 3D printer
- Servo motors
- Arduino or ESP32

Steps:

1. Design legs and frame in TinkerCAD/Fusion360
2. Print parts
3. Assemble

Experiment 11: 3D Print a Rocket

Objective:
Design and print a rocket model using CAD tools.

Theory:
Rocket design helps understand aerodynamics and structural modelling.

Steps:

1. Design nose cone, fins, body in CAD
2. Assemble body from segments
3. Print using PLA
4. Optional: Add LED inside for illumination

Experiment 12: Soil Moisture Monitoring and Dashboard

Objective:
Measure soil moisture and monitor it remotely using computer dashboard.

Theory:
Soil moisture sensor gives analog value proportional to water level. ESP32 sends data to Blynk for remote view.

Components Required:

- ESP32
- Soil moisture sensor
- Blynk App

Pin Connections:

Component	ESP32 Pin	Arduino UNO Pin
Soil Sensor (A0)	GPIO 36	A0

Code (Blynk):

```
#define BLYNK_TEMPLATE_ID "TMPL3jf4oigNl"
#define BLYNK_TEMPLATE_NAME "dht11 led"
#define BLYNK_AUTH_TOKEN "RSO8muGlW2LHlTc3z-zp15BS1yA7PjLw"

#define BLYNK_PRINT Serial
#include <WiFi.h>
#include <BlynkSimpleEsp32.h>

char ssid[] = "IobiT Solutions";
char pass[] = "IobiT@2025";

BlynkTimer timer;

#define SOIL_PIN 36  // Better ADC pin

void sendData() {
  int raw = analogRead(SOIL_PIN);
  int moisturePercent = map(raw, 0, 4095, 0, 100);
  moisturePercent = constrain(moisturePercent, 0, 100);

  Serial.print("Raw: ");
```

```
  Serial.print(raw);
  Serial.print(" => Moisture: ");
  Serial.print(moisturePercent);
  Serial.println("%");

  Blynk.virtualWrite(V1, moisturePercent);
}

void setup() {
  Serial.begin(115200);
  Blynk.begin(BLYNK_AUTH_TOKEN, ssid, pass);
  timer.setInterval(2000L, sendData);
}

void loop() {
  Blynk.run();
  timer.run();
}
```

💡 Add a graph or gauge on V1 to monitor changes.

Steps:

1. **Open Blynk Web Dashboard**
   - Go to https://blynk.cloud
   - Login with your account.
2. **Create a New Template**
   - Template Name: Soil Monitoring
   - Hardware: ESP32
   - Connection Type: WiFi
3. **Create a New Datastream**
   - Name: Soil Moisture
   - Type: Virtual Pin
   - Virtual Pin: V1
   - Data Type: Integer

- Min: 0, Max: 100 (or 1023 based on sensor)

4. **Design Dashboard**
    - Add a Gauge or Value Display widget to the canvas
    - Link it to V1
5. **Get Credentials**
    - Note down the
        1. BLYNK_TEMPLATE_ID
        2. BLYNK_DEVICE_NAME
        3. BLYNK_AUTH_TOKEN and Replace these three lines in code.
6. **Upload the Above Code to ESP32**

Experiment 13: Design Thinking Process: Redesigning a Smart Motorbike

Objective: Apply design thinking to identify and solve real-world problems by redesigning a traditional motorbike using IoT features.

1. Empathize – Understand User Needs

- Observe users like students, delivery workers, and commuters.
- Identify challenges: theft, fuel tracking, night visibility, maintenance, and safety.

2. Define – State the Core Problem

- Example Problem Statement:
  "Urban riders face safety, security, and maintenance issues due to lack of smart features in motorbikes."

3. Ideate – Generate Innovative Ideas

- GPS tracking with theft alerts.
- Smart lights using LDR sensors.
- Mobile app integration (fuel, maintenance, location).
- Ultrasonic-based parking assist.
- Smart helmet detection and remote engine cut-off.

4. Prototype – Build a Mini Model

- Use **ESP32/Arduino Uno** as the controller.
- Components: GPS module, LDR, ultrasonic sensor, relay, Blynk app.
- Simulate the redesigned features on a model bike or breadboard.

5. Test – Get Feedback & Improve

- Demonstrate to real users.
- Collect feedback on usability, functionality, and design.
- Iterate to improve the prototype.

Learning Outcome:

- Understand user-centric design.
- Apply IoT tools in real-life applications.
- Develop problem-solving and prototyping skills.