Complete Guide to UNO SMD Development Board

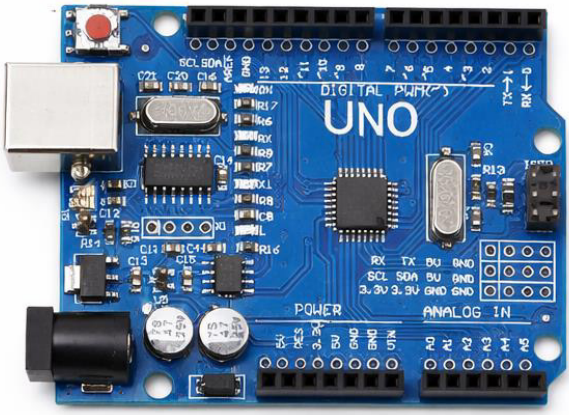
From Setup to 15 Hands-On Projects

Table of Contents

Introduction

Welcome to the comprehensive guide for the UNO SMD Development Board. This book is designed to take you from a complete beginner to a confident maker capable of building complex interactive projects. The UNO SMD board is a robust, affordable, and widely used microcontroller platform based on the ATmega328P chip. Unlike the traditional DIP (Dual Inline Package) version where the main chip can be removed, the SMD (Surface Mount Device) version features a compact, soldered chip, often paired with the CH340G USB-to-Serial converter.

In this guide, you will learn how to set up your development environment, install necessary drivers, and build 15 distinct projects ranging from simple lights to advanced display systems.



Section 1: Getting Started

1.1 What is UNO SMD Development Board

The UNO SMD is functionally identical to the classic Arduino UNO R3 but uses surface-mount technology for its components. This often makes the board more affordable to manufacture.

Key Specifications:

- Microcontroller: ATmega328P
- Operating Voltage: 5V
- Input Voltage (recommended): 7-12V (via DC jack)
- Digital I/O Pins: 14 (of which 6 provide PWM output)
- Analog Input Pins: 6
- Flash Memory: 32 KB
- Clock Speed: 16 MHz

> Difference Note: The primary difference users face is the USB communication chip. Official Arduino boards often use an ATmega16U2, while most affordable SMD clones use the CH340G chip. This requires a specific driver installation.

1.2 Installing Arduino IDE

The Arduino Integrated Development Environment (IDE) is the software where you write code and upload it to the board.
Windows:

1. Visit the official website: `arduino.cc/en/software`
2. Download the "Windows Win 10 and newer, 64 bits" installer.
3. Run the installer and follow the prompts. Allow all USB driver installation requests during the process.

Mac OS:

1. Download the version for your Mac (Intel or Apple Silicon).
2. Drag the Arduino application to your Applications folder.

Linux:

1. Download the AppImage 64 bits.
2. Right-click the file, go to Properties > Permissions, and check "Allow executing file as program".
3. Run the AppImage.

1.3 Connecting Your UNO Board

1. Connect the square end of the USB cable to the UNO board.
2. Connect the flat end (USB-A) to your computer.
3. You should see a green LED labeled ON or PWR light up immediately.
4. Often, an orange LED labeled L (connected to pin 13) may blink, indicating the factory test program is running.

1.4 Installing CH340G USB Driver

If your computer does not recognize the board (no COM port appears), you likely need the CH340 driver.

Windows Installation:

1. Search online for "CH340 driver Windows" (WCH official site is recommended).
2. Download the zip file (usually `CH341SER.ZIP`)
3. Unzip and run `SETUP.EXE`
4. Click INSTALL. A success message should appear.
5. Check Device Manager > Ports (COM & LPT). You should see "USB-SERIAL CH340 (COMX)".

Mac OS Installation:

1. Download the CH340 driver for Mac.
2. Install the PKG file.
3. Restart your Mac. You may need to allow the extension in System Preferences > Security & Privacy.

1.5 Configuring Arduino IDE

1. Open Arduino IDE.
2. Go to Tools > Board > Arduino AVR Boards and select Arduino Uno.
3. Go to Tools > Port.
   - Windows: Select the COM port labeled "COM3", "COM4", etc. (Not COM1).
   - Mac/Linux: Select the port labeled `/dev/cu.usbserial...` or `/dev/ttyUSB...`.

1.6 Testing Your Setup - Blink Example

1. Go to File > Examples > 01.Basics > Blink.
2. Click the Upload button (Right Arrow icon) in the top bar.
3. Watch the status bar at the bottom. It should say "Compiling sketch..." then "Uploading..." and finally "Done uploading."
4. Verify that the onboard orange LED (L) is blinking ON for 1 second and OFF for 1 second.

Section 2: 15 Projects with Sensors and Modules

PROJECT 1: LED Blink - Your First Program

Objective: Control an external LED to understand digital output.

Components: UNO Board, Breadboard, LED (Red), 220fi Resistor, Jumper Wires.

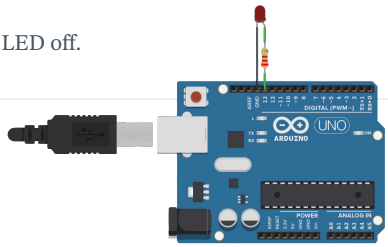
Wiring Diagram Description

1. Place the LED on the breadboard. Note the longer leg is the Anode (+) and the shorter leg is the Cathode (-).
2. Connect a **220Ω resistor** to the Anode (long leg).
3. Connect the other end of the resistor to Digital Pin 13 on the UNO.
4. Connect the Cathode (short leg) directly to a GND (Ground) pin on the UNO.

```
void setup() {
  // Initialize digital pin 13 as an output. pinMode(13, OUTPUT);
}

void loop() {
  digitalWrite(13, HIGH);          // Turn the LED on (HIGH is the voltage level)
  delay(1000);                     // Wait for a second (1000 milliseconds)
  digitalWrite(13, LOW);           // Turn the LED off by making the voltage LOW
  delay(1000);                     // Wait for a second
}
```

Code Explanation:

- `pinMode(13, OUTPUT)` : Configures pin 13 to send voltage out.
- `digitalWrite(13, HIGH)` : Sends 5V to the pin, turning the LED on.
- `delay(1000)` : Pauses the program for 1000 milliseconds (1 second).
- `digitalWrite(1 LOW)` Sends 0V (Ground) to the pin, turning the LED off.
  3,



PROJECT 2: Traffic Light System

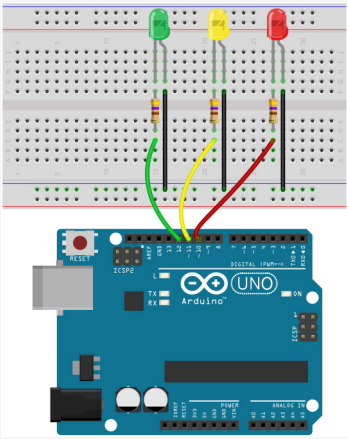
Objective: Control multiple LEDs in a specific timing sequence.

Components: UNO Board, 3 LEDs (Red, Yellow, Green), 3x 220fi Resistors.

Wiring Diagram Description

- Red LED Anode: Connect to Pin 10 (via 220fi resistor).
- Yellow LED Anode: Connect to Pin 11 (via 220fi resistor).
- Green LED Anode: Connect to Pin 12 (via 220fi resistor).
- All Cathodes: Connect to the common rail connected to GND.

```
int redPin = 10;
int yellowPin = 11;
int greenPin = 12;


void setup() { pinMode(redPin, OUTPUT);
    pinMode(yellowPin,OUTPUT);
    pinMode(greenPin, OUTPUT);
}

void loop() {
    // Red Light
    digitalWrite(redPin,HIGH); delay(3000);
    digitalWrite(redPin, LOW);

    // Yellow Light
     digitalWrite(yellowPin,HIGH); delay(1000);
    digitalWrite(yellowPin, LOW);

    // Green Light
     digitalWrite(greenPin,HIGH); delay(3000);
    digitalWrite(greenPin, LOW);
}
```



Working Principle: The code executes sequentially within the `loop()` ıtely proceeds to the next block, simulating a traffic cycle. . It turns one pin HIGH, waits, turns it LOW, and

PROJECT 3: Push Button Control LED

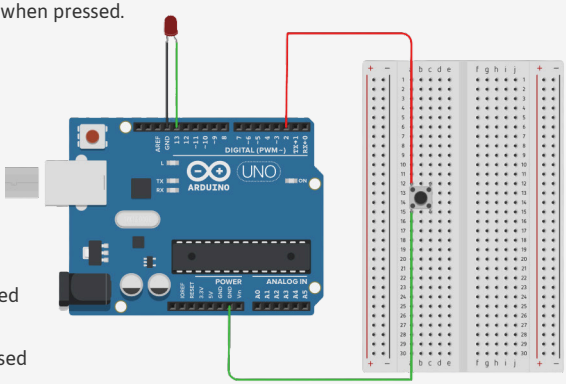
Objective: Read a digital input to control an output.

Components: Push Button, LED, Resistor.

Wiring Diagram Description

- LED: Connected to Pin 13 and GND. (No need Resistors , 13 pin has inbuilt Resistor)
- Button: One leg to Pin 2, diagonal leg to GND. (No external resistor needed due to internal pullup).

```
const int buttonPin = 2; const int ledPin
= 13;

void setup() {
    pinMode(ledPin,OUTPUT);
    // Activate internal resistor. Pin is HIGH when open, LOW when pressed.
    pinMode(buttonPin, INPUT_PULLUP);
}

void loop() {
    int buttonState = digitalRead(buttonPin);

    // Logic is inverted because of Input Pullup
    if (buttonState == LOW) {
        digitalWrite(ledPin, HIGH);                // Button pressed
    } else {
        digitalWrite(ledPin, LOW);                 // Button released
     }
}
```



Explanation: `INPUT_PULLUP` connectsan internal20kfi resistor to 5V. The pin reads HIGH by default. When the buttor connects to GND, the pin LOW.
reads

PROJECT 4: Potentiometer - Analog Reading

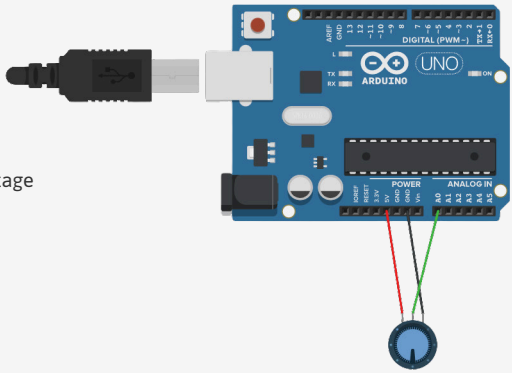
Objective: Read varying voltage from a knob and display values.

Components: 10kfi Potentiometer.

Wiring Diagram Description

- Left Pin: 5V
- Right Pin: GND
- Middle Pin (Wiper): Analog Pin A0

```
void setup() {
    Serial.begin(9600); // Initialize serial communication
}

void loop() {
    int sensorValue = analogRead(A0); // Reads 0-1023
    float voltage = sensorValue * (5.0 / 1023.0); // Convert to voltage

    Serial.print("Analog Value: "); Serial.print(sensorValue);
    Serial.print(" | Voltage: "); Serial.println(voltage);

    delay(500);
}
```



Open Serial Monitor to see the Analog Values

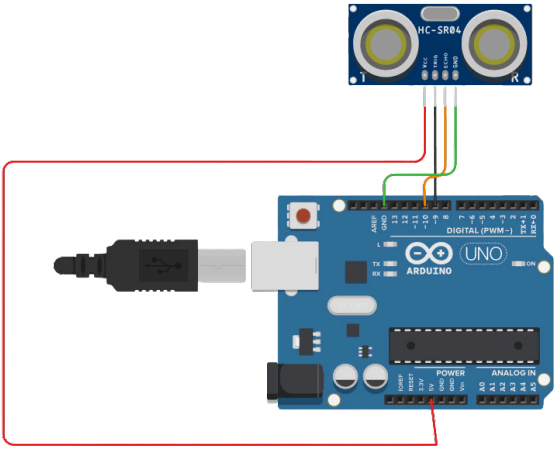
Working Principle: The potentiometer acts as a voltage divider. The Arduino's ADC (Analog to Digital Converter) maps 0-5V to an integer value between 0 and 1023.

PROJECT 5: Ultrasonic Sensor (HC-SR04)

Objective: Measure distance using sound waves.

Wiring Diagram Description

- VCC: 5V
- GND: GND
- Trig: Pin 9
- Echo: Pin 10



```
const int trigPin = 9;
const int echoPin = 10;
long duration;
int distance;

void setup() { pinMode(trigPin,OUTPUT);
    pinMode(echoPin, INPUT);
    Serial.begin(9600);
}

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

    // Send 10 microsecond pulse
    digitalWrite(trigPin,HIGH); delayMicroseconds(10);
    digitalWrite(trigPin, LOW);

    // Read echo pulse duration in microseconds duration =
    pulseIn(echoPin, HIGH);

    // Calculate distance
    (Speed of sound = 0.034 cm/us) distance = duration * 0.034 / 2;

    Serial.print("Distance:");
    Serial.print(distance); Serial.println("
    cm"); delay(100);
}
```

Open Serial Monitor to see the Distances

PROJECT 6: DHT11 Temperature & Humidity Sensor

Objective: Read environmental data. Requires "DHT sensor library" by Adafruit.

Wiring Diagram Description

- VCC: 5V
- GND: GND
- DATA: Pin 2 (Some modules require a 10k resistor between VCC and Data if not built-in).

- Go to Arduino IDE
- Go to Library
- Search for DHT11 Sensor by Adafruit
- Install it

```
#include "DHT.h"

#define DHTPIN 2
#define DHTTYPE DHT11

DHT dht(DHTPIN, DHTTYPE);

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

void loop() {
    delay(2000); // DHT11 is slow, wait 2 seconds

    float h = dht.readHumidity();
    float t = dht.readTemperature(); // Celsius

    if (isnan(h) || isnan(t)) {
        Serial.println("Failed to read from DHT sensor!"); return;
    }

    Serial.print("Humidity:"); Serial.print(h);
    Serial.print("%                Temperature:");
    Serial.print(t); Serial.println("°C");
}
```

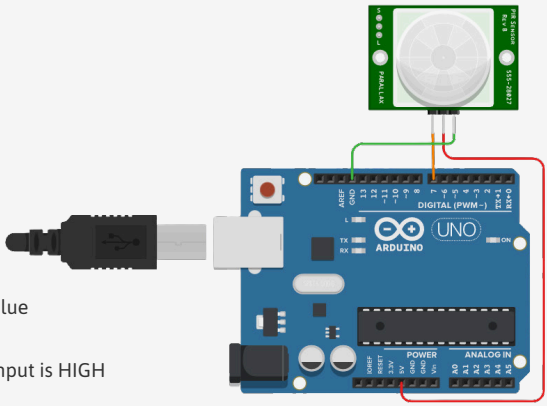
PROJECT 7: PIR Motion Sensor

Objective: Detect human movement.

Wiring Diagram Description

- VCC: 5V
- GND: GND
- OUT: Pin 7
- LED Indicator: Pin 13

```
int ledPin = 13;
int inputPin = 7;
int val = 0;

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

void loop() {
    val = digitalRead(inputPin);                // Read input value

    if (val == HIGH) {                          // Check if the input is HIGH
        digitalWrite(ledPin, HIGH); // Turn LED ON
    } else {
        digitalWrite(ledPin, LOW);              // Turn LED OFF
    }
}
```



Note: Most PIR sensors have two potentiometers to adjust sensitivity and time delay.

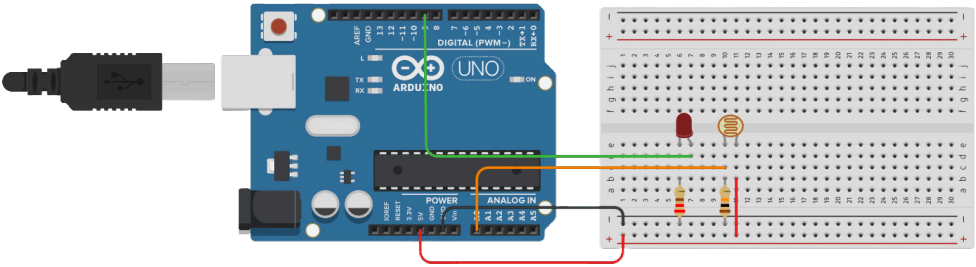
PROJECT 8: LDR Light Sensor

Objective: Create a night-light that turns on in the dark.

Wiring Diagram Description

Voltage Divider Circuit:

- One leg of LDR to 5V.
- Other leg of LDR to Analog Pin A0 AND to one leg of a 10kfi resistor.
- Other leg of 10kfi resistor to GND.
- LED connected to Pin 9.



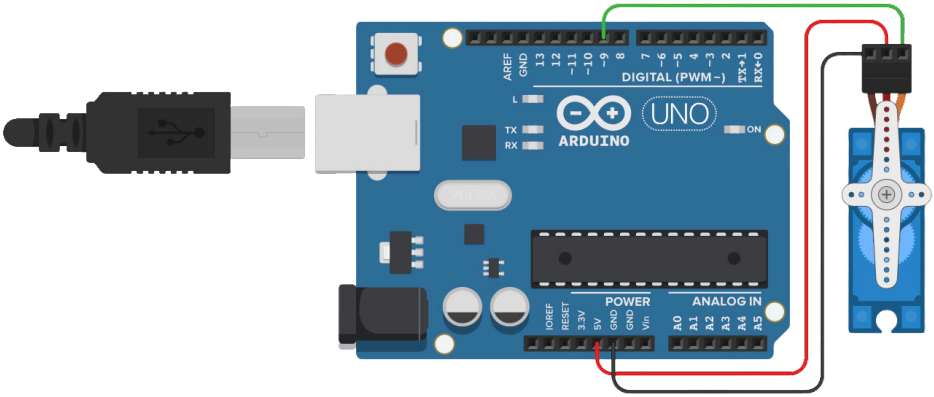
```
const int ldrPin = A0; const int
ledPin = 9; int ldrValue = 0;
int threshold = 500; // Adjust based on ambient light

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

void loop() {
    ldrValue = analogRead(ldrPin);
    Serial.println(ldrValue); // Use to calibrate threshold

    if (ldrValue < threshold) {
    // If it's dark (resistance increases, voltage drops)
    digitalWrite(ledPin, HIGH);
    } else { digitalWrite(ledPin,LOW);
    }
    delay(100);
}
```

PROJECT 9: Servo Motor Control

Objective: Precise motor positioning.

Wiring Diagram Description

- Red Wire: 5V
- Brown/Black Wire: GND
- Orange/Yellow Wire (Signal): Pin 9 (PWM)



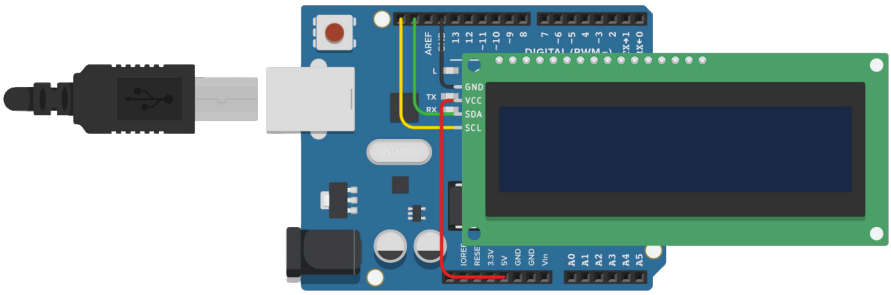
```
#include <Servo.h>

Servo myservo;          // Create servo object
int pos = 0;            // Variable to store position

void setup() {
    myservo.attach(9);          // Attaches servo on pin 9
}

void loop() {
    // Sweep from 0 to 180 degrees
    for (pos = 0; pos <= 180; pos += 1) {
    myservo.write(pos); delay(15);
    }
    // Sweep back
    for (pos = 180; pos >= 0; pos -= 1) { myservo.write(pos);
        delay(15);
    }
}
```

PROJECT 10: LCD Display (16x2 I2C)

Objective: Display text using only 2 data wires. Requires "LiquidCrystal_I2C" library.

Wiring Diagram Description

- VCC: 5V
- GND: GND
- SDA: Pin A4 (Data)
- SCL: Pin A5 (Clock)



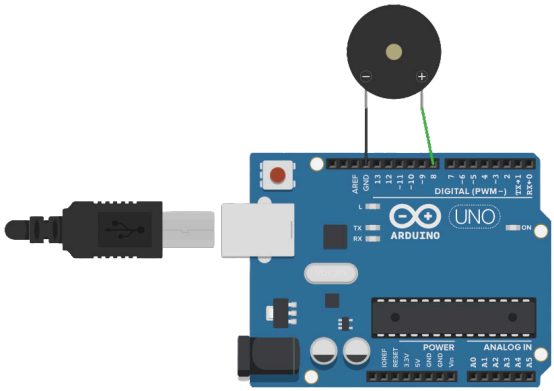
```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>

// Set address to 0x27 or 0x3F usually LiquidCrystal_I2C
lcd(0x27, 16, 2);

void setup() {
    lcd.init();                      // Initialize the LCD
    lcd.backlight(); // Turn on backlight

    lcd.setCursor(0, 0); // Col 0, Row 0
    lcd.print("Hello, World!");

    lcd.setCursor(0, 1); // Col 0, Row 1
     lcd.print("PAGALCODER");
}

void loop() {
    // Static display
}
```

PROJECT 11: Buzzer - Melody Player

Objective: Generate audio tones.

Wiring Diagram Description

- Positive (+): Pin 8
- Negative (-): GND



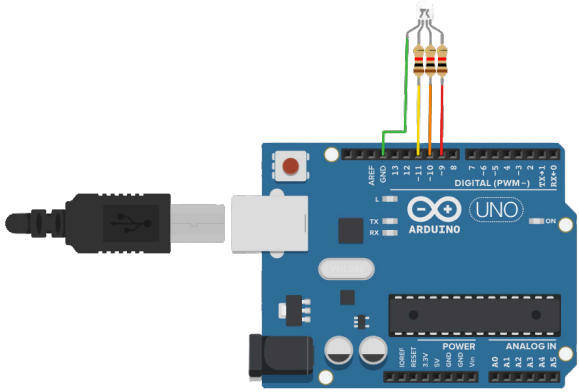
```
int buzzerPin = 8;

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

void loop() {
    // Play A4 (440 Hz) for 500ms
    tone(buzzerPin, 440); delay(500);

    // Stop sound
    noTone(buzzerPin); delay(200);

    // Play C5 (523 Hz)
    tone(buzzerPin,523); delay(500);
    noTone(buzzerPin);

    delay(1000); // Wait before repeating
}
```

PROJECT 12: RGB LED Color Control

Objective: Mix colors using PWM.

Wiring Diagram Description (Common Cathode)

- Red Pin: Pin 9 (PWM) via 220fi resistor.
- Green Pin: Pin 10 (PWM) via 220fi resistor.
- Blue Pin: Pin 11 (PWM) via 220fi resistor.
- Longest Pin (Cathode): GND.



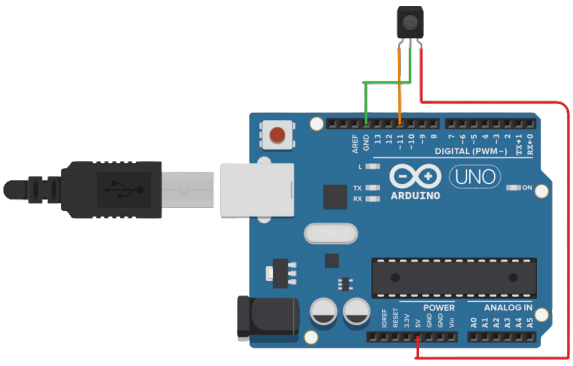
```
int redPin = 9; int greenPin
= 10; int bluePin = 11;

void setup() { pinMode(redPin, OUTPUT);
    pinMode(greenPin,OUTPUT);
    pinMode(bluePin, OUTPUT);
}

void setColor(int red, int green, int blue) { analogWrite(redPin, red);
analogWrite(greenPin, green); analogWrite(bluePin, blue);
}

void loop() {
    setColor(255, 0, 0); // Red delay(1000);
    setColor(0, 255, 0); // Green delay(1000);
    setColor(0, 0, 255); // Blue delay(1000);
    setColor(255, 255, 0); // Yellow delay(1000);
}
```

PROJECT 13: IR Remote Control

Objective: Wireless control. Requires "IRremote" library.

Wiring Diagram Description

- VCC: 5V
- GND: GND
- Signal: Pin 11
- LED: Pin 13



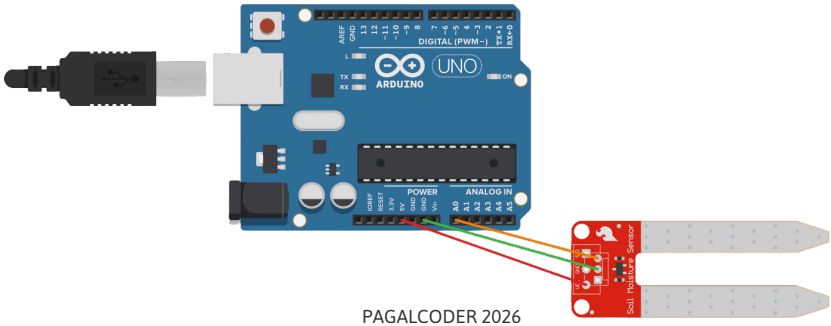
```
#include <IRremote.h>

const int RECV_PIN = 11; const int
ledPin = 13; IRrecvirrecv(RECV_PIN);
decode_results results;

void setup() { Serial.begin(9600);
    irrecv.enableIRIn(); // Start the
    receiver pinMode(ledPin, OUTPUT);
}

void loop() {
    if (irrecv.decode(&results)) {
        Serial.println(results.value, HEX); // Print HEX code to find your button codes

        // Example: Replace 0xFFA25D with your remote's specific code
        if(results.value == 0xFFA25D) {
            digitalWrite(ledPin, !digitalRead(ledPin)); // Toggle LED
        }

        irrecv.resume(); // Receive the next value
    }
}
```

PROJECT 14: Soil Moisture Sensor

Objective: Detect if plants need water.

Wiring Diagram Description

- VCC: 5V
- GND: GND
- AO (Analog Output): Pin A0



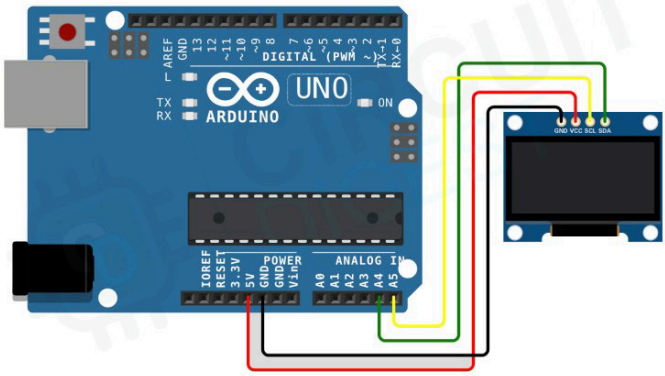
```
const int sensorPin = A0;
const int dryThreshold = 800; // Calibrate via Serial Monitor

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

void loop() {
    int moisture = analogRead(sensorPin); Serial.print("Moisture Value:
    "); Serial.println(moisture);

    if(moisture > dryThreshold)
    { Serial.println("Soil is DRY - Water needed!");
    } else {
        Serial.println("Soil is WET");
    }
    delay(1000);
}
```

PROJECT 15: OLED Display (128x64 I2C)

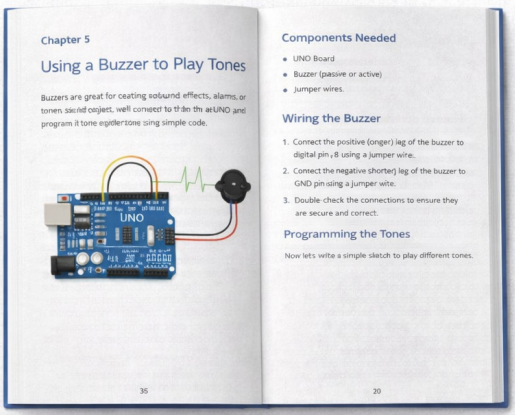
Objective: High-resolution graphics. Requires "Adafruit SSD1306" and "Adafruit GFX" libraries.

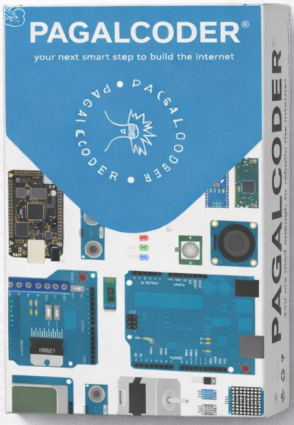
Wiring Diagram Description

- VCC: 5V
- GND: GND
- SDA: A4
- SCL: A5



```
#include <Wire.h> #include
<Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64
#define OLED_RESET -1
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);

void setup() {
    // Address 0x3C for 128x32 usually, check datasheet
    if(!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {
        for(;;); // Loop forever if failed
    }

    display.clearDisplay();

    display.setTextSize(1);
    display.setTextColor(SSD1306_WHITE);
    display.setCursor(0,0); display.println("Hello Maker!");

    display.setTextSize(2); display.setCursor(0,20);
    display.println("OLEDTest");

    display.display(); // Important! Actual command to update screen
}

void loop() {}
```





Conclusion

Congratulations! By completing this guide, you have mastered the essentials of the UNO SMD Development Board. You started by installing the necessary drivers and IDE, and moved on to controlling outputs, reading various inputs, and using communication protocols like I2C and UART.

Skills Acquired:

- Digital I/O control (LEDs, Buttons, PIR).
- Analog sensing (Potentiometers, LDRs, Moisture sensors).
- PWM usage (RGB LEDs, Servos).
- Library integration for complex modules (DHT11, OLED, IR).
- Communication protocols (Serial, I2C).

Your journey doesn't stop here. The sensors and concepts you've learned are the building blocks for robotics, home automation, and IoT devices. We encourage you to combine these projects—try displaying temperature on the OLED, or turning on a fan (motor) based on the temperature sensor.

Keep experimenting, keep breaking things, and keep fixing them. That is the essence of making.

Explore our most advance UNO kits , designed by professionals (visit our store on Amazon/Flipkart

SPECIAL OFFER
999/-

SPECIAL OFFER
1899/-