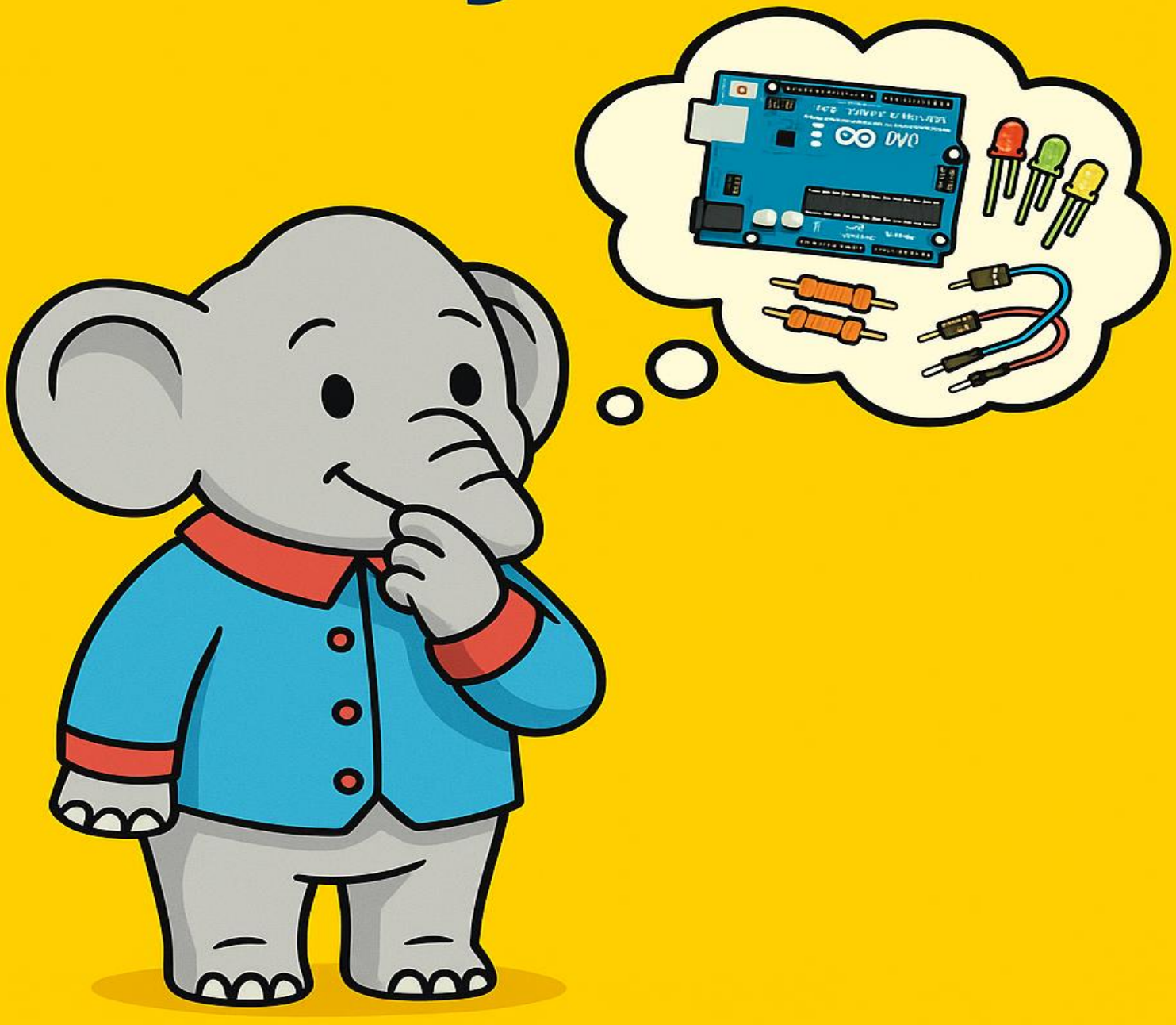
Top Five Arduino Projects!



Top 5 Arduino Projects for Beginners

Contents

HOBITRONICS

Note: This guide is designed for absolute beginners. No prior electronics knowledge needed. Just a passion to explore and create!

Let's begin the fun!

Regards!

Team Hobitronics.

If you have any queried or suggestions, feel free to reach us at ***hobitronicsin@gmail.com***

HOBITRONICS

Project 1: Blinking an LED using Arduino

Introduction

This is the most basic yet iconic project for Arduino beginners. It introduces you to how microcontrollers interact with external devices. By turning an LED on and off repeatedly, you'll learn how digital output pins work and how to control them using code.

Components Required

- Arduino Uno (or any compatible board)
- Breadboard
- LED (any color)
- 220Ω Resistor
- Jumper Wires
- USB Cable (for uploading code)

Wiring Instructions

1. Connect the **longer leg (anode)** of the LED to **digital pin 13** on Arduino.
2. Connect the **shorter leg (cathode)** to one end of the resistor.
3. Connect the other end of the resistor to **GND (Ground)** on Arduino.

It doesn't matter where you put your LED Light between the digital pin 13 and LED or between cathode of the LED and GND

HOBITRONICS

Why?

This can be explained by **KCL and Ohm's Law!**

Current in series is same everywhere. Since there's no splitting or combining of current anywhere along the path, **the same amount of current must flow through every component** (e.g., through both the LED and the resistor).

You can also use Ohm's Law to find the amount of current entering LED with a multimeter and generally the voltage rating of a red LED is 1.8V to 2.2V.

You could easily find the power dissipated by the LED in the form of Light using the formula **P=V.I**

Where,

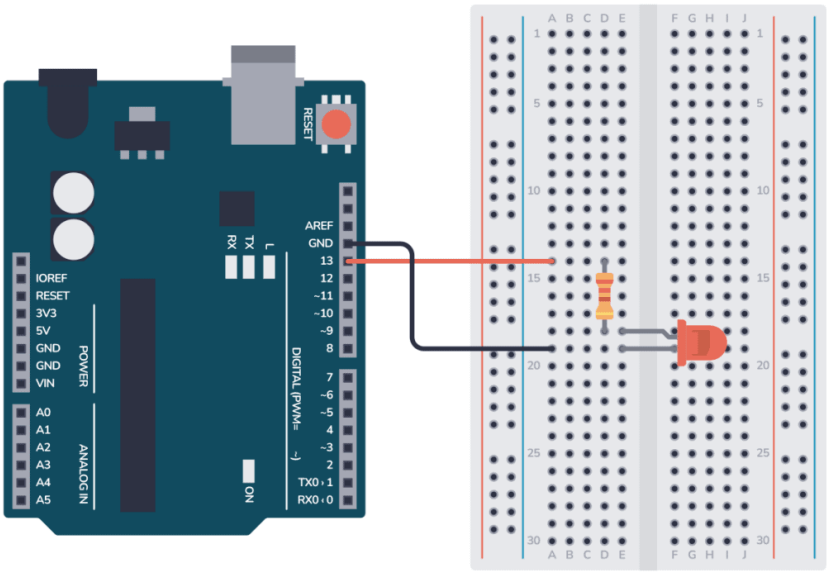
P = Power (in Watts),

V = Potential (In Volts),

I = Current (in Amperes)

Tip: You can also use a breadboard for easier connections.

Circuit Diagram



Arduino Code

```
void setup() {
  pinMode(13, OUTPUT); // Set pin 13 as output
}
void loop() {
  digitalWrite(13, HIGH); // Turn on LED
  delay(1000);            // Wait 1 second
  digitalWrite(13, LOW);  // Turn off LED
  delay(1000);            // Wait 1 second
}
```

HOBITRO

We use milliseconds in **delay()** function , if you want it slightly longer increase the timing in ms in **delay()** function.

What You Learn

- Basics of Arduino IDE
- Uploading code to the board
- How digital pins work
- Controlling an LED using code
- Importance of resistors with LEDs
- Delay and timing concepts in microcontrollers

HOBITRONICS

Project 2: Button Controlled LED using Arduino

Introduction

This project teaches you how to take **user input** with a push button and use it to control an LED. It introduces **digital input**, a critical concept when building interactive electronics projects.

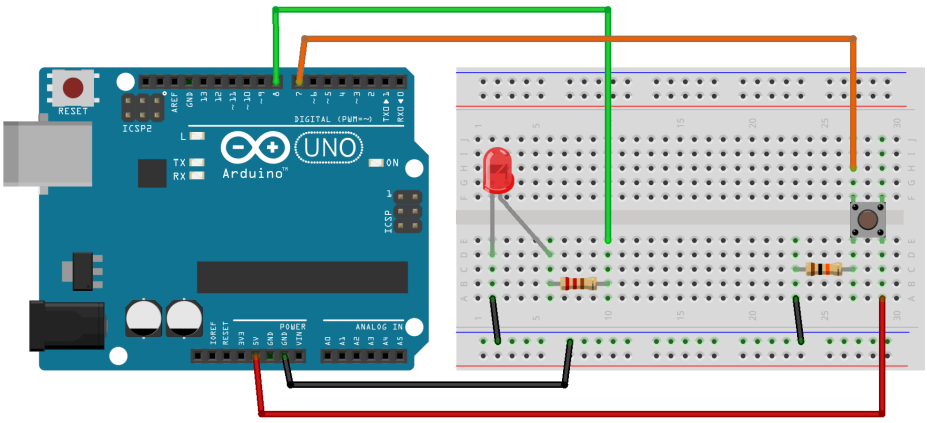
Components Required

- Arduino Uno
- Breadboard
- LED
- 220Ω Resistor (for LED)
- Push Button
- 10kΩ Resistor (for button pull-down)
- Jumper Wires
- USB Cable

Wiring Instructions

1. Connect the **anode of the LED** to **digital pin 13**, and its **cathode** to GND through a 220Ω resistor.
2. Place the **push button** on the breadboard.
3. Connect one leg of the button to **5V**.
4. Connect the opposite leg to **digital pin 2** and also to **GND** via a **10kΩ resistor** (this acts as a pull-down resistor to avoid false triggering).

Circuit Diagram



HOBTRONICS

Arduino Code

```
int buttonPin = 2;

int ledPin = 13;

int buttonState = 0;

void setup() {

  pinMode(ledPin, OUTPUT);

  pinMode(buttonPin, INPUT);

}

void loop() {

  buttonState = digitalRead(buttonPin);

    if (buttonState == HIGH) {

    digitalWrite(ledPin, HIGH); // Turn on LED

  } else {

    digitalWrite(ledPin, LOW);  // Turn off LED

  }

}
```

HOBITRONICS

What You Learn

- How to read digital input using Arduino
- Pull-down resistors and their role
- Using conditional statements in code
- Building user-controlled interactive systems
- Logical high and low states
- Troubleshooting button bouncing issues

Project 3: Traffic Light System using Arduino

Introduction

This project simulates a basic **traffic light system** using three LEDs (red, yellow, green). It helps you understand how **timing**, **sequential control**, and **digital outputs** work — a great stepping stone into the world of automation!

Components Required

- Arduino Uno
- Breadboard
- 3 LEDs (Red, Yellow, Green)
- 3 × 220Ω Resistors
- Jumper Wires
- USB Cable

Wiring Instructions

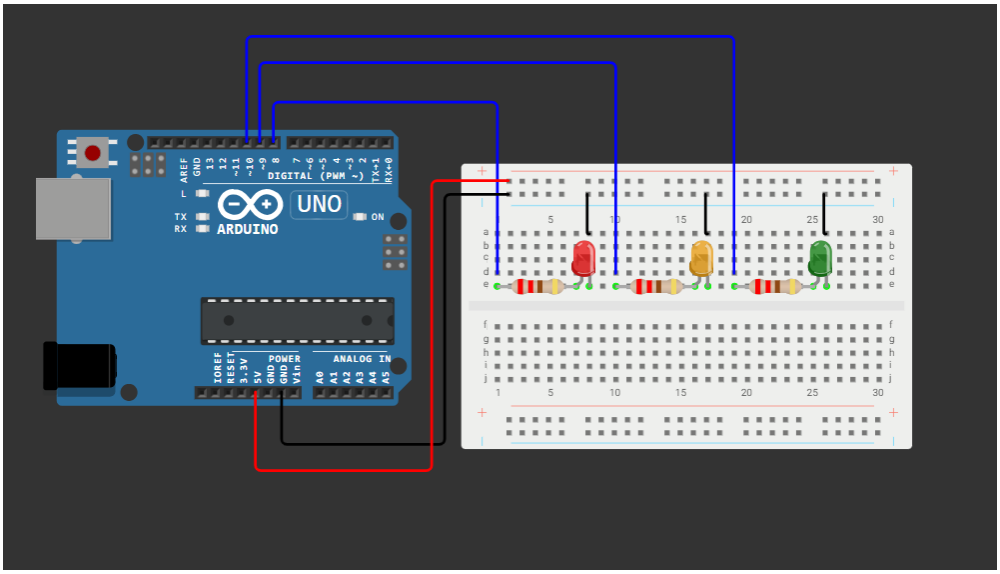
- **Place all 3 LEDs** (red, yellow, green) on the breadboard with their legs not touching.
- **Connect the anode (longer leg)** of the:

    1. **Red LED** to **digital pin 8** on the Arduino.
    2. **Yellow LED** to **digital pin 9**.
    3. **Green LED** to **digital pin 10**.

- **Connect the cathode (shorter leg)** of **each LED to one end of a 220Ω resistor**.
- **Connect the other end** of each resistor to **GND** on the Arduino — this limits current to protect the LEDs.
- Make sure **all GND connections are common** (Arduino GND, breadboard GND rails).

Circuit Diagram

HOBITRONICS



Arduino Code

```
int red = 8;

int yellow = 9;

int green = 10;

void setup() {

  pinMode(red, OUTPUT);

  pinMode(yellow, OUTPUT);

  pinMode(green, OUTPUT);

}

void loop() {

  // RED ON, others OFF

  digitalWrite(red, HIGH);

  digitalWrite(yellow, LOW);

  digitalWrite(green, LOW);

  delay(3000); // Red for 3 seconds
```

HOBITRONICS

```
  // YELLOW ON, others OFF
  digitalWrite(red, LOW);
  digitalWrite(yellow, HIGH);
  digitalWrite(green, LOW);
  delay(1000); // Yellow for 1 second
  // GREEN ON, others OFF
  digitalWrite(red, LOW);
  digitalWrite(yellow, LOW);
  digitalWrite(green, HIGH);
  delay(3000); // Green for 3 seconds
}
```

What You Learn

- How to control multiple LEDs
- Basics of real-time systems and delay handling
- Simulating real-world applications using Arduino
- Creating sequences with timing logic
- How traffic lights work at a basic level

HOBITRONICS

Project 4: Temperature Monitor using Arduino and LM35

Introduction

In this project, we'll build a simple temperature monitor **using the** LM35 sensor **and** Arduino Uno**.** The sensor detects temperature and sends the data to Arduino, which processes and displays the temperature in the **Serial Monitor.** This teaches you how to read analog values, convert sensor data to real-world units (like °C), and display it without any external display — just using your computer!

Components Required

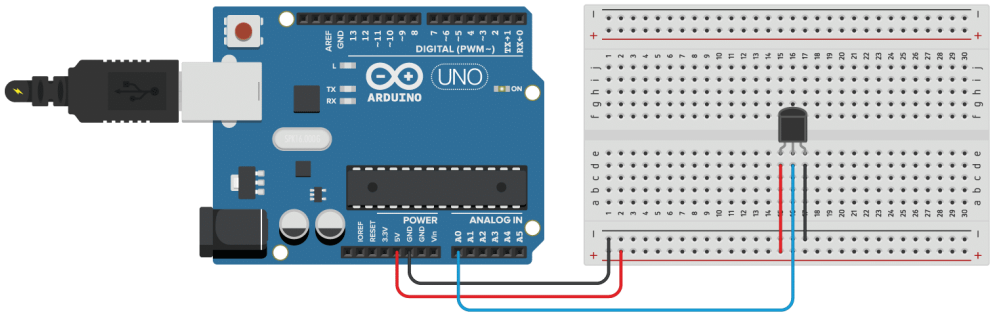
- Arduino Uno
- LM35 Temperature Sensor
- Breadboard
- Jumper Wires
- USB Cable

Wiring Instructions

1. Place the **LM35 sensor** on the breadboard.
2. Connect the **left pin (Vcc)** to **5V on Arduino**.
3. Connect the **middle pin (Vout)** to **A0 (Analog Pin 0)**.
4. Connect the **right pin (GND)** to **GND on Arduino**.

Note: You won't need any LED or display — just use the **Serial Monitor in the Arduino IDE** to view the temperature readings in real time.

Circuit Diagram



HOBITRONICS

Arduino Code

```
int sensorPin = A0;
float temperature;

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

void loop() {
  int reading = analogRead(sensorPin);

  // Convert the analog reading to voltage
  float voltage = reading * (5.0 / 1023.0); // 0–1023 maps to 0–5V
//if you use a 3.3 V board in Arduino or Esp32 use reading * (3.3/1023)
 // LM35 gives 10mV per °C => 0.01V per °C
  temperature = voltage / 0.01;

  Serial.print("Temperature: ");
  Serial.print(temperature);
  Serial.println(" °C");

  delay(1000);
}
```

HOBITRONICS

What You Learn

- How analog sensors work with Arduino
- Understanding voltage-to-temperature conversion
- Basics of real-time temperature monitoring
- How to use the serial monitor for debugging
- A real-world application of Arduino in environmental sensing

Project 5: Light Controlled LED (Using LDR and Arduino)

Introduction

This project uses a **Light Dependent Resistor (LDR)** to automatically turn an LED **ON** in darkness and **OFF** in light. It's a classic beginner project that introduces you to **sensors**, **analog input**, and **decision making with if statements** in Arduino.

Components Required

- 1 x Arduino Uno
- 1 x LDR (Light Dependent Resistor)
- 1 x 10kΩ resistor (for voltage divider)
- 1 x LED
- 1 x 220Ω resistor (for LED)
- Breadboard
- Jumper wires

Wiring Instructions

- **Connect one end of the LDR (Light Dependent Resistor)** to **5V**.
- **Connect the other end of the LDR** to **analog pin A0** and to one side of the **10kΩ resistor** (this forms a **voltage divider** with the LDR).
- **Connect the other side of the 10kΩ resistor to GND**.
- The voltage divider here ensures that the voltage at **A0** varies depending on the amount of light falling on the LDR.
- **Connect the anode (longer leg) of the LED** to **pin 9** on the Arduino **through a 220Ω resistor**.
- **Connect the cathode (shorter leg) of the LED** directly to **GND** on the Arduino.

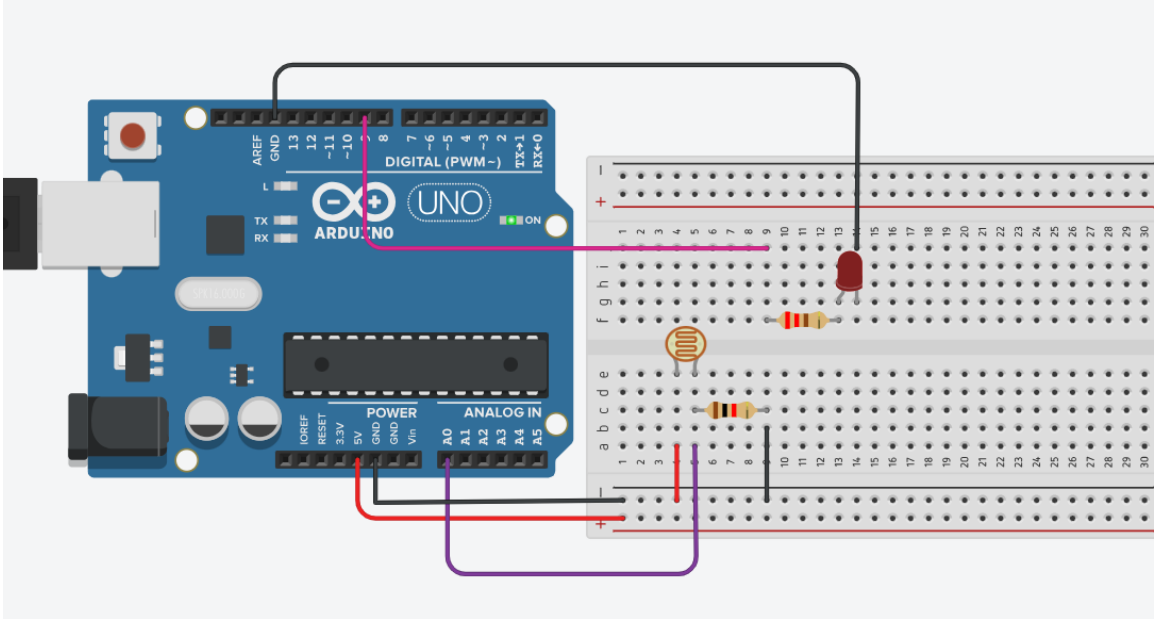
HOBITRONICS

Arduino Code

```
int ldrPin = A0;    // LDR connected to analog pin A0

int ledPin = 9;     // LED connected to digital pin 9

int ldrValue = 0;

void setup() {

 pinMode(ledPin, OUTPUT);   // Set LED pin as output
```

```
  Serial.begin(9600);        // Initialize serial communication for monitoring
}

void loop() {
  ldrValue = analogRead(ldrPin);  // Read the value from the LDR
  Serial.println(ldrValue);       // Print the light level to the serial monitor

  if (ldrValue < 500) {           // Threshold for darkness (can be adjusted)
    digitalWrite(ledPin, HIGH);   // Turn on LED when it's dark
  } else {
    digitalWrite(ledPin, LOW);    // Turn off LED when it's bright
  }
  delay(500);  // Wait for half a second before reading again
}
```

Circuit Diagram



What You Learn

- Use of **analog sensors** with Arduino
- How a **voltage divider** works
- Use of **if-else conditions**
- How to monitor sensor data using the **Serial Monitor**
- Real-world applications: **automatic street lights, smart lighting systems**