# 4.Sound-Adaptive RGB Lights

## Project Description: -

The **Sound-Reactive LED System** is designed to dynamically change LED brightness and color based on ambient sound levels. This system creates a **visual representation of audio**, making it ideal for music synchronization, interactive installations, and decorative lighting.

At the core of this project is an **Arduino Pro Mini**, which processes audio signals detected by a **sound sensor**. The sensor captures fluctuations in sound intensity, and the Arduino adjusts the LED brightness accordingly. A **TIP122 transistor** is used to handle the high current required by the LED strip, ensuring smooth and efficient control.

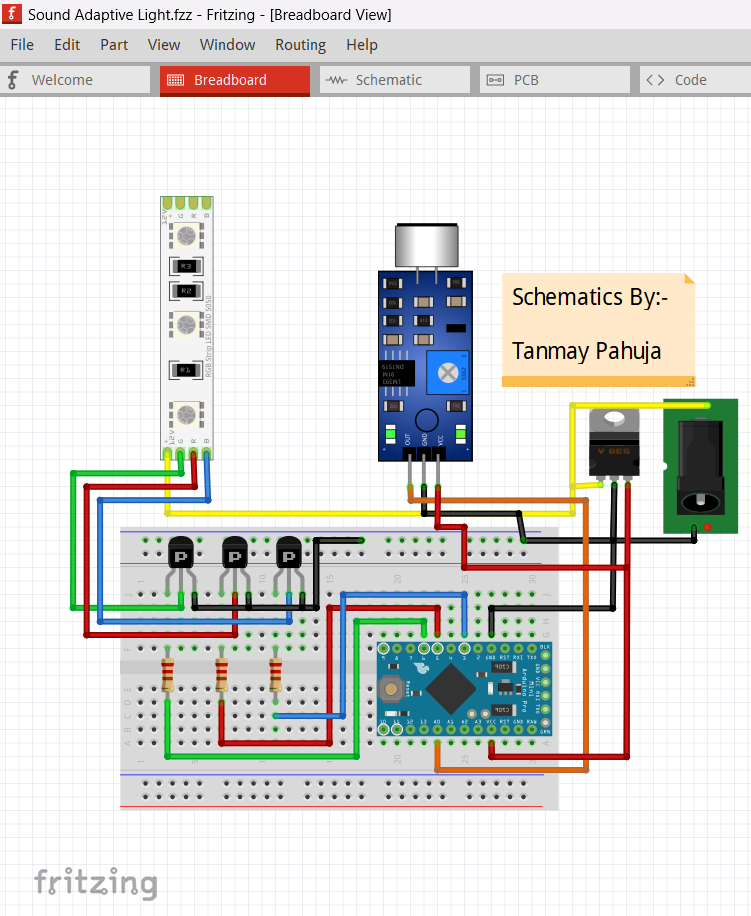
To power the system, a **7805-voltage regulator** is used to step down the supply voltage, providing a stable 5V for the Arduino. This setup allows the system to operate reliably without voltage fluctuations affecting performance.

By reacting to music, voice, or any environmental noise, this project offers an engaging and immersive lighting experience. Whether used in parties, home decor, or artistic displays, the **sound-reactive LED system** enhances the atmosphere by synchronizing light patterns with sound.

## Apparatus: -

| **S. No.** | **Component** | **Quantity** | **Function** |
| --- | --- | --- | --- |
| 1 | **Arduino Pro Mini** | 1 | Microcontroller to process sound input and control LEDs |
| 2 | **Sound Sensor** | 1 | Detects ambient sound levels and converts them into electrical signals |
| 3 | **TIP122 Transistor** | 1 | Acts as a switch to control the LED strip based on sound intensity |
| 4 | **LED Strip** | 1 | Changes brightness and color based on sound input |
| 5 | **7805 Voltage Regulator** | 1 | Converts higher voltage to a stable 5V supply for the Arduino |
| 6 | **Resistors (1kΩ, 10kΩ)** | 3 | Used for circuit stability and signal conditioning |
| 7 | **Power Supply (12V or as required by the LED strip)** | 1 | Provides power to the LED strip and Arduino |
| 8 | **Connecting Wires** | As needed | For circuit connections |
| 9 | **Breadboard or Perf-board** | 1 | For assembling the circuit |

## Connection: -



1. Arduino Pro Mini to Sound Sensor

| **Arduino Pin** | **Sound Sensor Pin** | **Function** |
| --- | --- | --- |
| VCC | VCC | Power supply (5V) |
| GND | GND | Common ground |
| A0 (Analog Pin) | OUT | Reads sound level data |

2. Arduino Pro Mini to TIP122 Transistors (RGB LED Control)

| **Arduino Pin** | **TIP122 Pin** | **Function** |
| --- | --- | --- |
| D3 (PWM) | Base (R) | Controls **Red** LED brightness |
| D5 (PWM) | Base (G) | Controls **Green** LED brightness |
| D6 (PWM) | Base (B) | Controls **Blue** LED brightness |
| GND | Emitter (All 3) | Common ground |
| LED Strip (-) | Collector (R, G, B) | Switches RGB channels ON/OFF based on sound |

3. Power Supply to LED Strip & Arduino

| **Power Source** | **Component** | **Function** |
| --- | --- | --- |
| 12V Power Supply | LED Strip (+) | Powers the LED strip |
| 12V Power Supply | 7805 Regulator (Input) | Steps down voltage to 5V for Arduino |
| 7805 Regulator (Output) | Arduino VCC | Provides stable 5V to Arduino |
| 7805 Regulator GND | Arduino GND | Common ground |

## Code: -

// Define pin numbers for RGB LEDs

#define Rpin 3

#define Gpin 5

#define Bpin 6

#define delayLEDS 5  // Delay in milliseconds

#define sensorPin A0 // Sound sensor analog input pin

// Variables for sensor reading and signal filtering

float sensorValue = 0;

float filteredSignal = 0;

// Array of threshold values for filtered signal

float filteredSignalValues[] = {2.5, 2.4, 2.3, 2.2, 2.1, 2.0, 1.9,1.8,1.7};

void setup() {

  // Initialize serial communication for debugging

  Serial.begin(9600);

  // Set RGB LED pins as outputs

  pinMode(Rpin, OUTPUT);

  pinMode(Gpin, OUTPUT);

  pinMode(Bpin, OUTPUT);

}

void loop() {

  // Main function to read sensor, filter signal, and set LED color

  MainFunction();

}

void MainFunction() {

  // Read the analog value from the sound sensor and convert to voltage

  sensorValue = (float)analogRead(sensorPin) \* (5.0 / 1023.0);

  // Apply a simple low-pass filter to the sensor signal

  FilterSignal(sensorValue);

  // Print sensor and filtered values to the Serial Monitor for debugging

  Serial.print("Sensor Value: ");

  Serial.print(sensorValue);

  Serial.print(" V, Filtered Signal: ");

  Serial.println(filteredSignal);

  // Compare filtered signal against predefined thresholds and set LED color

  CompareSignalFiltered(filteredSignal);

}

void FilterSignal(float sensorSignal) {

  // Apply a basic exponential moving average filter to the sensor signal

  filteredSignal = (0.945 \* filteredSignal) + (0.0549 \* sensorSignal);

}

void CompareSignalFiltered(float filteredSignal) {

  // Check filtered signal against thresholds and set the appropriate color

  if (filteredSignal > filteredSignalValues[0]) {

    RGBColor(0,0,0);

    Serial.println("Blue");

  } else if (filteredSignal > filteredSignalValues[1]) {

    RGBColor(0, 255, 255); // Azure

    Serial.println("Azure");

  } else if (filteredSignal > filteredSignalValues[2]) {

    RGBColor(0, 127, 255); // Cyan

    Serial.println("Cyan");

  } else if (filteredSignal > filteredSignalValues[3]) {

    RGBColor(128, 0, 128); // Aqua marine

    Serial.println("Aqua marine");

  } else if (filteredSignal > filteredSignalValues[4]) {

    RGBColor(255,0, 0); // Green

    Serial.println("Green");

  } else if (filteredSignal > filteredSignalValues[5]) {

    RGBColor(255, 255, 0); // Yellow

    Serial.println("Yellow");

  } else if (filteredSignal > filteredSignalValues[6]) {

    RGBColor(255, 0, 255); // Magenta

    Serial.println("Magenta");

  } else if (filteredSignal > filteredSignalValues[7]) {

    RGBColor(255, 0, 127); // Rose

    Serial.println("Rose");

  } else if (filteredSignal > filteredSignalValues[8]) {

    RGBColor(255, 127, 0); // Orange

    Serial.println("Orange");

  } else {

    RGBColor(0, 0, 255); // Red

    Serial.println("Red");

  }

}

void RGBColor(int Rcolor, int Gcolor, int Bcolor) {

  // Write PWM values to RGB LED pins

  analogWrite(Rpin, Rcolor);

  analogWrite(Gpin, Gcolor);

  analogWrite(Bpin, Bcolor);

  // Short delay to allow LED color change to be visible

  delay(delayLEDS);

}

## Project Outcome: -

The **Sound-Reactive LED System** successfully creates an interactive lighting experience by dynamically adjusting the RGB LED strip’s brightness and colors in response to ambient sound. The integration of an **Arduino Pro Mini, a sound sensor, and three TIP122 transistors** enables efficient real-time control of the LED strip based on sound intensity.

One of the key outcomes is the system’s ability to **synchronize lighting effects with music or environmental noise**. The LED strip’s **red, green, and blue channels** are independently controlled, allowing for vibrant color changes that visually represent sound variations. This makes it ideal for **parties, home décor, or artistic installations** where dynamic lighting enhances the atmosphere.

Another important result is **efficient power management**. The **7805 voltage regulator ensures a stable power supply**, while the transistor-based switching mechanism allows for smooth LED brightness adjustments without excessive energy loss. The system is designed to operate **continuously without overheating or flickering issues**, ensuring long-term reliability.

Additionally, the modular nature of this project allows for **easy customization and expansion**. Users can modify the **sensitivity of the sound sensor** or adjust the **PWM control logic** to create different lighting effects based on personal preferences. The system can also be integrated with additional sensors or microcontrollers for advanced features like **wireless control or mobile app integration**.

Overall, the **Sound-Reactive LED System provides an engaging and visually appealing response to sound**, making it a **simple yet effective solution** for interactive lighting applications.