MUSIC REACTIVE 12V RGB LED STRIPS USING ARDUINO UNO

Introduction:

Music reactive 12v RGB led strip is a fun project which changes its color based on the sound of the music. This project is built using an Arduino Uno board that controls multiple color effects of LED strips based on the frequency that hits the drum of the sound sensor. This RGB LED strip itself has multiple colors such as red, green, yellow, blue, orange, white, and, pink, which allows for a range of lighting effects. The sound sensor detects the beat of the music as input and changes the strip color and intensity based on the trigger created in the LED strip in response to the music or sound. The transistor TIP 122 is used to function the LED strip properly. With this project, one can create a unique dynamic light show and interactive visual display for events or parties.

Components required:

The components used in this project are given below:

- Arduino UNO
- 12v RGB LED strip of less than 1.5 amperes
- 12v power supply with 5 amperes
- Female DC power jack
- Sound sensor
- Breadboard
- Resistor (three 1 kilo ohms)
- Transistor (three TIP 122)
- Jumper wires (Male to male and male-to-female jumper wires.

Principle of working:

When the music is on, the sound waves from the music or audio input reach the sound sensor, vibrate, and hits the diaphragm inside the device as an analog or analog signal. This signal is then converted and sent it to Arduino Uno as a digital signal. This conversion is done using an analog-to-digital converter (ADC) which is a built-in material in Arduino board. The Arduino board analyses and generates appropriate digital signals to control or change the patterns, intensity, and colors of the RGB LED strips. The transistor is used to control the power supply of the led strip as the Arduino output pins do not provide enough current to the led strips. Here, the resistor 1-kilo ohm is used to prevent the damage which is then used to limit the current flowing through each individual color channel (RGB) to work within the specific current range.

Modules description:

First-order-low pass filter:

The first-order low-pass filter is used to reduce the unwanted high-frequency noise which results in stopping the flickering of the led or to remove the fluctuations. The raw signal from the music source contains unwanted noise, makes difficult to analyze the signal accurately. So, the filter will be passed to

smooths out the signal which attenuate the high-frequency noise and allows the low-frequency noise to pass through. The intensity or the color of the led strip is controlled and determined by filtered value or smoothed signal in this project.

Baud rate:

It refers to the data transfer rate and shows how much ability to send data in a second. For this project, the baud rate is taken as 9600 bits per second. It refers to the serial communication between the Arduino board and the computer. When the data is sent via serial communication, it ensures that it is sent and received properly on the receiving side.

Sound sensor:

The sound sensor detects the sound from the music as analog input and converts them into an electrical signal. When the sound waves hit the diaphragm inside the microphone, it vibrates and the waves fluctuate in response to the creation of a small current which creates the output signal by amplifying and filtering the current. The analog signal is then converted and sent it as a digital signal to Arduino uno board. The Arduino board processes the output signal to control the color and brightness of the led strip. There is a potentiometer in the sound sensor and we have to adjust it based on the led strip.

Transistor TIP 122:

The transistor TIP 122 has three pins: base, emitter, and collector. The responsible of the base is to control the current flow between the emitter, the collector allows the current to flow into the transistor and the emitter is responsible to allow the current to flow out of the transistor, Collector is It has the ability to handle high voltage and current. Compared to other transistors, it provides a high current gain in which the higher current loads in led strips are controlled by the high gain, allowing the current to amplify from Arduino's output. In this project, the transistor TIP 122 is particularly used to control the current flow of the led strip based on the music input.

DC power supply:

Sometimes LED strips need more current to power up the lights, especially when multiple LEDs are lit simultaneously. The Arduino board cannot provide sufficient power to the lights. So, a 12v DC power supply is used in this project.

Current and voltage requirements:

The total current rate of Arduino uno and RGB led strips is less than or equal to the current rate of power supply.

The amperage of Arduino uno = 200mA (maximum current drawn)

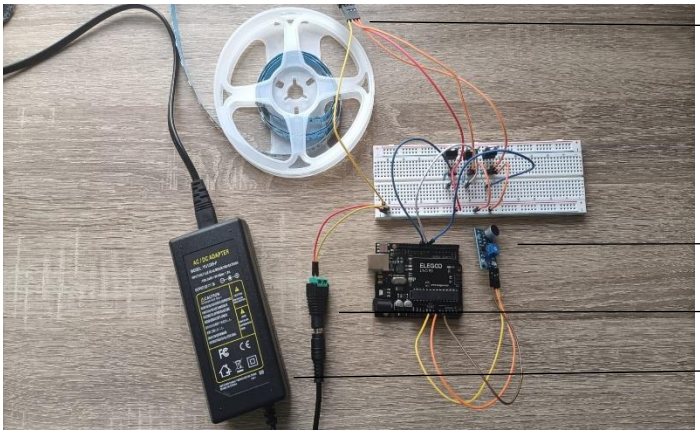
The amperage of RGB led strips = <1.5A (per channel).

Total amperage for three channels (Red, green, and blue) = Number of channels x Amperage per channel = 1.5A x 3 = 4.5A

The amperage of the power supply = 200mA + 4.5 A = 4.52 A.

A power supply with a current rating of 5A or higher is suitable to power the 12v RGB led strips when connected to an Arduino Uno. This external power supply is used because the Arduino board has a maximum current of 200mA which cannot provide the necessary current to power the RGB led strips.

Circuit diagram connections:

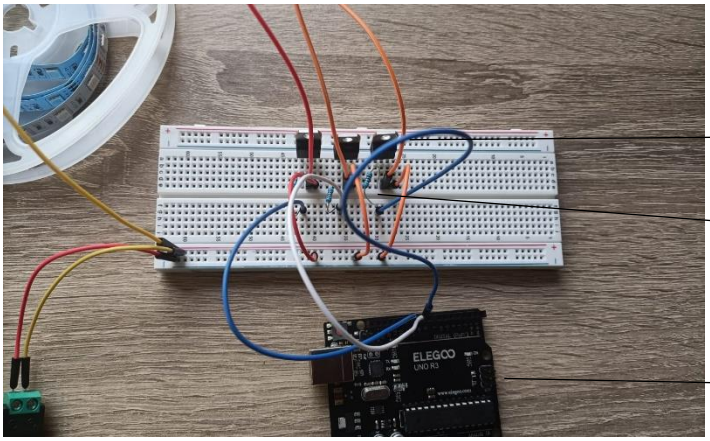


12v RGB LED Strip

Sound sensor

Female DC power jack

12v DC power supply



Transistor TIP 122

Resistor

Arduino UNO board

Base

Collector

Emitter

Circuit connection explanation:

1. Place three transistors TIP 122 on the breadboard. And place the 1-kilo ohm resistor into a hole direct below the base of the three transistors.
2. Three male-male wires are connected to the hole directly below the end of the resistor. These three wires are placed on digital pins 11, 10, and 9 on the Arduino Uno board.
3. The male-male jumper wires are connected to the emitter of the transistors and ground the other end of the wires to the bottom rail of the negative on the breadboard.
4. The sound sensor has three pins: VCC, OUT, and GND. One end of the female-male jumper wires is connected to these three pins. Connect the VCC of the sound sensor to the 5V supply of the Arduino board, GND to the GND (ground) of the Arduino board, and OUT pin to the analog pin A0 of Arduino.
5. The RGB LED strips have four female connectors, where the black wire is the 12v positive power supply of the strip is connected to the bottom rail of the positive of the breadboard, connecting the red, green, and blue of the led strip to the collector of the transistor which is also connected to the pins of 11, 10, 9 of Arduino board.
6. Connect the 12v DC power supply to the positive and negative of the bottom rail of the breadboard.

Explanation of code:

1. First, define the pins for the sound sensor (Sound_pins) and for the RGB LED strip (Red_pin, Green_pin, and Blue_pin).
2. The variable delay_leds is set to 3 which determines the delay between the color changing of LEDs.
3. Filter coefficients:
    - Define the filter parameters: cut-off frequency (variable Cutofffreq) is set to 900Hz which determines the frequency of the first-order low-pass filter and variable alpha is set to 0.2 which determines the influence of current and previous values on the filtered value.
4. Variable initialization:
    - The variables Redvalue, Greenvalue, and Bluevalue are initialized to zero to store the values of RGB colors. The variable Sensorvalue and filteredvalue are also initialized to zero to store the raw and filtered sound sensor value.
5. Setup() function:
    - When the Arduino board starts up, once the setup function is called.
    - RGB pins and serial communication are initialized into the void function, setup().
    - RGB pins are initialized as output using pinMode function and initialize the serial communication with a baud rate of 9600 using the code Serial.begin(9600).
6. The function loop():
    - The main part of the code is the loop function which will be executed repeatedly.
    - The analogRead(Sound_pin) value is called to read the raw sound sensor value and it will be stored in the variable Soundsensor.
    - Then the first-order-low-pass filter is used to attenuate the high-frequency noise and is applied to the Sensorvalue which smooths out the sensor reading using the formula, filteredvalue = Alpha * Sensorvalue + (1 - Alpha) * previous_value, where filtered value refers the current filtered signal value and the Sensorvalue is the value of current input signal from the analog pin. Before applying this formula, initialize the variable

previous_value to be zero. The obtained filtered value is assigned to the previous_value for the next iteration.
    - The different RGB-led strip colors are determined based on the obtained filtered value using the function RGBColor. This function takes three arguments such as Redcolor, Greencolor, and Bluecolor.
    - The filtered value is compared with the threshold value and if the filteredvalue is less than the assigned threshold value, the specific color will light up. The sensor and the filtered values are shown in the serial monitor as output.
    - The function ends with the variable Delay_LEDs in which the loop function waits for the delay (in milliseconds) before starting the next iteration.
7. The function RGBColor():
    - This function sets the values of RGB color on the LED strip and sets corresponding analog output pins.
    - The analog write function writes the pulse width modulation value (analog value) to a corresponding pin for each color based on the given red, green and blue.

Overall, the code reads the sensor value, filters them, and determines the color based on filtered value using a low pass filter, updates the filtered value to change the color, and displays the corresponding color on the led strip. The name of the color, sensor values, and filtered values will be shown in the serial monitor.

Uploading Arduino uno code:

Once the connection is done, upload the code in Arduino IDE. Connect the USB cable from the Arduino board to the computer. Switch on the 12v DC power supply.

When it's all done, select the Arduino port and board. Then compile the sketch. The LEDs will be on and will start to react to the music.

Troubleshooting:

1. We need to be careful when selecting how much voltage and amperage of power supply is suitable for this project. We need to make about three things:
    - The voltage of the RGB strips matches the voltage of the power supply.
    - The amperage requirements of Arduino Uno and RGB lights should be less than or equal to the amperage of the power supply in which the requirements should not exceed the maximum current rate of the power supply.
    - To find the total current needed for the power supply, we have to add the current requirements of Arduino uno and RGB led strip.
2. As 12v RGB led strips are used in this project, it is necessary to use a 12v power supply for it. If the voltage of the power supply is lesser than the voltage of RGB strips, it will overheat the LED and burn out. This will sometime cause an inconsistent color representation of LEDs. At the start of the project, I used a 12v power supply to 5v RGB led strips which leads to damage to lights and burned out.

Conclusion:

Hence, we see how lights react to the music using Arduino Uno. Further development of this project, we can control the music lights using hand gestures. It is important to make the connections properly. This can also be implemented in restaurants, clubs, concerts, and in wedding festivals.

Reference:

1. What is Serial. begin(9600)? - Programming Electronics Academy
2. **https://www.researchgate.net/publication/352568748_Music_Reactive_LED_using_ARDUINO_NANO**
3. (24) Ultimate Guide to Choosing the Right Addressable LED Strip | LinkedIn
4. Make Your Own Sound Activated 12V RGB Party Light - Robotica DIY
5. TIP122 Transistor Pinout, Features, Equivalent & Datasheet (components101.com)
6. Baud Rate Arduino On Serial Port. What Does It Mean? (chippiko.com)
7. [Arduino] Implementation of Basic Filters | TKF's World of Engineering (wordpress.com)
8. Read Analog Voltage | Arduino Documentation