Sound Activated Lights

Introduction

I have always liked to play with lights and i had a led strip laying around my pc setup and i thought it would be fun if it was sound reactive. The lights should react to the sound waves, either because of the rithm or the wavelength of the sound.

General Description

There are 3 parts of the circuit, one for each color(RGB); the tranzistors are connected to the assigned arduino pins (gnd to gnd) and to the leds; the sound sensor is only connected to the arduino board(GND to GND, Vcc to 5V and out to A0). The power supply is conntected to the breadboard. Ultimately there are 5 different modules connected to each other:

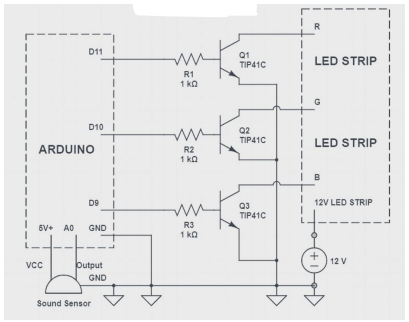
Block scheme:

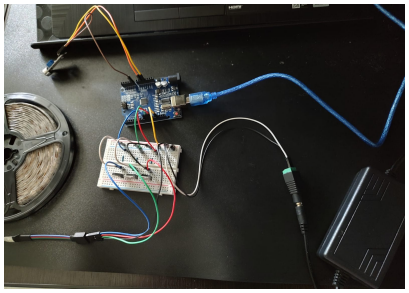
Hardware Design

The following components are required for this project:

- arduino
- Sound sensor with built in microphone and analog output
- 3 transistors tip41c
- 3 1k resistors
- breadboard
- led strip
- power supply 12v

Electric scheme:





Software Design

- ADC (lab4)

-sensorValue = (float)analogRead(sensorPin) * (5.0 / 1024.0);

-performs an analog-to-digital conversion using the analogRead function; it reads the analog value from sensorPin and converts it to a digital value

- Timer (lab3)

-Timer1.initialize(100000);

-sets the timer interval to 100ms (0.1s); this allows for periodic execution of a function based on the timer interval

- Interrupt (lab2)

-Timer1.attachInterrupt(FlashLEDs);

-attaches the FlashLEDs function to the interrupt triggered by Timer1; this means that every time the timer interrupt occurs (every 100ms), the FlashLEDs function will be executed.

- libraries and 3rd party sources:

-<avr/interrupt.h>

-TimerOne.h (It is used in this code to set up a timer interrupt for flashing the LEDs)

- algorithms and structures:

-MainFunction(): Reads the sensor value, filters the signal, and compares it to predefined values to determine the color to display on the RGB LED strip

-FilterSignal(float sensorSignal): Applies a filter to the sensor signal using a weighted moving average algorithm

-CompareSignalFiltered(float filteredSignal): Compares the filtered signal to predefined thresholds to determine the appropriate color to display

-RGBColor(int Rcolor, int Gcolor, int Bcolor): Sets the color of the RGB LED strip by writing appropriate values to the Rpin, Gpin, and Bpin pins

- functions:

-setup(): Sets up the initial configuration of the program, including serial communication initialization and timer setup

-loop(): Contains the main execution loop, calling the MainFunction() repeatedly

-FlashLEDs(): A timer interrupt service routine (ISR) that alternates the LED strip between white and off states

```
#include <avr/interrupt.h>
#include <TimerOne.h>

#define Rpin 11
#define Gpin 10
#define Bpin 9
#define delayLEDS 3
#define sensorPin A0

float sensorValue = 0, filteredSignal = 0,
    filteredSignalValues[] = {3.4, 3.1, 2.7, 2.4, 2.1, 1.7, 1.3, 0.9, 0.4};

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

  Timer1.initialize(100000);  // Set the timer interval to 100ms (0.1s)
  Timer1.attachInterrupt(FlashLEDs);  // Attach the FlashLEDs function to
the timer interrupt
}

void loop() {
  MainFunction();
}

void MainFunction() {
  sensorValue = (float)analogRead(sensorPin) * (5.0 / 1024.0);
  FilterSignal(sensorValue);
  Serial.print(sensorValue);
  Serial.print(" ");
  Serial.println(filteredSignal);
  CompareSignalFiltered(filteredSignal);
}
```

```
void FilterSignal(float sensorSignal) {
  filteredSignal = (0.945 * filteredSignal) + (0.0549 * sensorSignal);
}

void CompareSignalFiltered(float filteredSignal) {
  if (filteredSignal > filteredSignalValues[0]) {
    RGBColor(0, 0, 255);
    Serial.println("Blue");
  } else if (filteredSignal <= filteredSignalValues[0] && filteredSignal >
filteredSignalValues[1]) {
    Serial.println("Azure");
    RGBColor(0, 255, 255);
  } else if (filteredSignal <= filteredSignalValues[1] && filteredSignal >
filteredSignalValues[2]) {
    RGBColor(0, 127, 255);
    Serial.println("Cyan");
  } else if (filteredSignal <= filteredSignalValues[2] && filteredSignal >
filteredSignalValues[3]) {
    RGBColor(0, 255, 127);
    Serial.println("Aqua marine");
  } else if (filteredSignal <= filteredSignalValues[3] && filteredSignal >
filteredSignalValues[4]) {
    RGBColor(0, 255, 0);
    Serial.println("Green");
  } else if (filteredSignal <= filteredSignalValues[4] && filteredSignal >
filteredSignalValues[5]) {
    RGBColor(255, 255, 0);
    Serial.println("Yellow");
  } else if (filteredSignal <= filteredSignalValues[5] && filteredSignal >
filteredSignalValues[6]) {
    RGBColor(255, 0, 255);
    Serial.println("Magenta");
  } else if (filteredSignal <= filteredSignalValues[6] && filteredSignal >
filteredSignalValues[7]) {
    RGBColor(255, 0, 127);
    Serial.println("Rose");
  } else if (filteredSignal <= filteredSignalValues[7] && filteredSignal >
filteredSignalValues[8]) {
    RGBColor(255, 127, 0);
    Serial.println("Orange");
  } else if (filteredSignal <= filteredSignalValues[8]) {
    RGBColor(255, 0, 0);
    Serial.println("Red");
  } else {
    RGBColor(0, 0, 255);
    Serial.println("Default: Blue");
  }
}

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

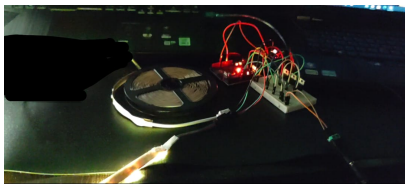
```
  analogWrite(Rpin, Rcolor);
  analogWrite(Gpin, Gcolor);
  analogWrite(Bpin, Bcolor);

  delay(delayLEDS);
}

void FlashLEDs() {
  static boolean flashState = false;

  if (flashState) {
    RGBColor(255, 255, 255);  // White color
    flashState = false;
  } else {
    RGBColor(0, 0, 0);  // Turn off LEDs
    flashState = true;
  }
}
```

Results

The project utilizes an Arduino board along with a sound sensor, three resistors, three transistors, and a LED strip. The goal of the project is to change the color of the LED strip based on the intensity of the sound signal. The code implements a filtering mechanism to smooth out the sensor signal and compares it with predefined threshold values to determine the color.

During testing, the project demonstrated successful functionality. As the sound intensity changes, the LED strip reacts by changing colors accordingly. The code effectively filters the sensor signal to provide more stable readings and compares the filtered signal with a set of predefined values to determine the appropriate color.

Additionally, a timer interrupt has been incorporated into the code to introduce a flashing effect on the LEDs. This feature alternates between turning the LEDs on and off at a regular interval, creating a blinking effect.



Conclusion

Overall, the project achieves the desired outcome of changing the LED strip's colors based on sound intensity, enhancing the visual experience in response to music or audio input. The added flashing effect further enhances the visual appeal of the project.

Download

proj_details.zip

Jurnal

Finished initial doc. (7/5/2023)

Added hardware scheme. (28/5/2023)

Added software program.(28/05/2023)

Bibliography

https://www.geeksforgeeks.org/introduction-of-led/

https://randomnerdtutorials.com/guide-for-ws2812b-addressable-rgb-led-strip-with-arduino/

Export to PDF