PROYECTO DE ASPIRANTES II PAO 2021

GRUPO 7

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

|  |  |
| --- | --- |
| Inicio  **Club de desarrollo**  **de Physical Computing**  **“PYCHOM”** | Descripción breve  Desarrollar un dispositivo interactivo que se relaciona con efectos sonoros para la producción de luces a medida que el sonido se reproduzca, lo cual este dispositivo será conectado a una fuente de alimentación para su funcionamiento. |

**Audio Reactive**

AVANCE 1, 2 y FINAL

gRUPO N.º 10

Integrantes

Holguin Sevastian

Veloz Joel

18/07/2021

1. Introducción

Para el presente proyecto de un reactor de audio “Audio reactive” se plantea la propuesta de crear este modelo atractivo y dinámico en donde se puede aplicar física computacional, es decir, permitirá una gran experiencia para las personas que lo deseen adquirir, sin embargo, la aplicación de este proyecto es de crecer en conocimientos acerca de programas de simulación, diseño y codificación.

* 1. Utilidad/Funcionamiento

El funcionamiento de este proyecto parte de la línea de alimentación que se suministra mediante un cable USB de la PC hacia la placa de Arduino NANO, pero si se llegase a conectar una fuente externa que superen los 5V, ejemplo: fuente de alimentación de 12V de AC a CC, se tendrá que usar un regulador de voltaje para reducir la magnitud de corriente que ingresara a la placa, lo cual, permitirá un correcto funcionamiento del Arduino. Luego, se dispondrá del modulo de detector de sonido, lo cual, esto proporcionará una señal analógica al Arduino, que podemos usar para encender inteligentemente las luces RGB.

Durante el proceso de implementación, el módulo detector de sonido se comunicará con el Arduino a través de los pines de entrada analógica. Se tomará el pin de su elección, en este caso escogeremos el A0.

Las tiras de LED necesitan un pulso digital para poder entender a qué LED queremos dirigirnos. Por lo tanto, necesitamos conectar un pin de salida digital al Arduino nano. Usaré el pin número 6.

Para la elección de la superficie se usará el tubo PVC en el área elegida para que el cable no choque entre sí en un espacio reducido más tarde.

* 1. Características
* Dinámico
* Volátil
* Creativo
* Atractivo
* Manipulable
* Adaptable

1. Desarrollo
   1. Materiales

Los materiales para utilizar para la fabricación de este proyecto son:

* Breakout board for electret microphone
* Circuito integrado ML2596
* Arduino NANO V3.0
* Cautin
* Fuente de alimentación 5V ( o 12 voltios con un regulador de voltaje)
* Tiras Leds direccionables
* Alguna superficie que se desee colocar las tiras leds, por ejemplo, un cilindro de policloruro de vinilo (tubos de PVC).
* Silicona
  1. Diagrama, Esquemático

     Descripción generada automáticamenteDiagrama de Bloques
  2. Implementación del diagrama de bloques en Proteus/Tinkercad
  3. Implementación y simulación del código

#include <FastLED.h>

/\*\* BASIC CONFIGURATION \*\*/

//The amount of LEDs in the setup

#define NUM\_LEDS 60

//The pin that controls the LEDs

#define LED\_PIN 6

//The pin that we read sensor values form

#define ANALOG\_READ 0

//Confirmed microphone low value, and max value

#define MIC\_LOW 0.0

#define MIC\_HIGH 200.0

/\*\* Other macros \*/

//How many previous sensor values effects the operating average?

#define AVGLEN 5

//How many previous sensor values decides if we are on a peak/HIGH (e.g. in a song)

#define LONG\_SECTOR 20

//Mneumonics

#define HIGH 3

#define NORMAL 2

//How long do we keep the "current average" sound, before restarting the measuring

#define MSECS 30 \* 1000

#define CYCLES MSECS / DELAY

/\*Sometimes readings are wrong or strange. How much is a reading allowed

to deviate from the average to not be discarded? \*\*/

#define DEV\_THRESH 0.8

//Arduino loop delay

#define DELAY 1

float fscale( float originalMin, float originalMax, float newBegin, float newEnd, float inputValue, float curve);

void insert(int val, int \*avgs, int len);

int compute\_average(int \*avgs, int len);

void visualize\_music();

//How many LEDs to we display

int curshow = NUM\_LEDS;

/\*Not really used yet. Thought to be able to switch between sound reactive

mode, and general gradient pulsing/static color\*/

int mode = 0;

//Showing different colors based on the mode.

int songmode = NORMAL;

//Average sound measurement the last CYCLES

unsigned long song\_avg;

//The amount of iterations since the song\_avg was reset

int iter = 0;

//The speed the LEDs fade to black if not relit

float fade\_scale = 1.2;

//Led array

CRGB leds[NUM\_LEDS];

/\*Short sound avg used to "normalize" the input values.

We use the short average instead of using the sensor input directly \*/

int avgs[AVGLEN] = {-1};

//Longer sound avg

int long\_avg[LONG\_SECTOR] = {-1};

//Keeping track how often, and how long times we hit a certain mode

struct time\_keeping {

unsigned long times\_start;

short times;

};

//How much to increment or decrement each color every cycle

struct color {

int r;

int g;

int b;

};

struct time\_keeping high;

struct color Color;

void setup() {

Serial.begin(9600);

//Set all lights to make sure all are working as expected

FastLED.addLeds<NEOPIXEL, LED\_PIN>(leds, NUM\_LEDS);

for (int i = 0; i < NUM\_LEDS; i++)

leds[i] = CRGB(0, 0, 255);

FastLED.show();

delay(1000);

//bootstrap average with some low values

for (int i = 0; i < AVGLEN; i++) {

insert(250, avgs, AVGLEN);

}

//Initial values

high.times = 0;

high.times\_start = millis();

Color.r = 0;

Color.g = 0;

Color.b = 1;

}

/\*With this we can change the mode if we want to implement a general

lamp feature, with for instance general pulsing. Maybe if the

sound is low for a while? \*/

void loop() {

switch(mode) {

case 0:

visualize\_music();

break;

default:

break;

}

delay(DELAY); // delay in between reads for stability

}

/\*\*Funtion to check if the lamp should either enter a HIGH mode,

or revert to NORMAL if already in HIGH. If the sensors report values

that are higher than 1.1 times the average values, and this has happened

more than 30 times the last few milliseconds, it will enter HIGH mode.

TODO: Not very well written, remove hardcoded values, and make it more

reusable and configurable. \*/

void check\_high(int avg) {

if (avg > (song\_avg/iter \* 1.1)) {

if (high.times != 0) {

if (millis() - high.times\_start > 200.0) {

high.times = 0;

songmode = NORMAL;

} else {

high.times\_start = millis();

high.times++;

}

} else {

high.times++;

high.times\_start = millis();

}

}

if (high.times > 30 && millis() - high.times\_start < 50.0)

songmode = HIGH;

else if (millis() - high.times\_start > 200) {

high.times = 0;

songmode = NORMAL;

}

}

//Main function for visualizing the sounds in the lamp

void visualize\_music() {

int sensor\_value, mapped, avg, longavg;

//Actual sensor value

sensor\_value = analogRead(ANALOG\_READ);

//If 0, discard immediately. Probably not right and save CPU.

if (sensor\_value == 0)

return;

//Discard readings that deviates too much from the past avg.

mapped = (float)fscale(MIC\_LOW, MIC\_HIGH, MIC\_LOW, (float)MIC\_HIGH, (float)sensor\_value, 2.0);

avg = compute\_average(avgs, AVGLEN);

if (((avg - mapped) > avg\*DEV\_THRESH)) //|| ((avg - mapped) < -avg\*DEV\_THRESH))

return;

//Insert new avg. values

insert(mapped, avgs, AVGLEN);

insert(avg, long\_avg, LONG\_SECTOR);

//Compute the "song average" sensor value

song\_avg += avg;

iter++;

if (iter > CYCLES) {

song\_avg = song\_avg / iter;

iter = 1;

}

longavg = compute\_average(long\_avg, LONG\_SECTOR);

//Check if we enter HIGH mode

check\_high(longavg);

if (songmode == HIGH) {

fade\_scale = 3;

Color.r = 5;

Color.g = 3;

Color.b = -1;

}

else if (songmode == NORMAL) {

fade\_scale = 2;

Color.r = -1;

Color.b = 2;

Color.g = 1;

}

//Decides how many of the LEDs will be lit

curshow = fscale(MIC\_LOW, MIC\_HIGH, 0.0, (float)NUM\_LEDS, (float)avg, -1);

/\*Set the different leds. Control for too high and too low values.

Fun thing to try: Dont account for overflow in one direction,

some interesting light effects appear! \*/

for (int i = 0; i < NUM\_LEDS; i++)

//The leds we want to show

if (i < curshow) {

if (leds[i].r + Color.r > 255)

leds[i].r = 255;

else if (leds[i].r + Color.r < 0)

leds[i].r = 0;

else

leds[i].r = leds[i].r + Color.r;

if (leds[i].g + Color.g > 255)

leds[i].g = 255;

else if (leds[i].g + Color.g < 0)

leds[i].g = 0;

else

leds[i].g = leds[i].g + Color.g;

if (leds[i].b + Color.b > 255)

leds[i].b = 255;

else if (leds[i].b + Color.b < 0)

leds[i].b = 0;

else

leds[i].b = leds[i].b + Color.b;

//All the other LEDs begin their fading journey to eventual total darkness

} else {

leds[i] = CRGB(leds[i].r/fade\_scale, leds[i].g/fade\_scale, leds[i].b/fade\_scale);

}

FastLED.show();

}

//Compute average of a int array, given the starting pointer and the length

int compute\_average(int \*avgs, int len) {

int sum = 0;

for (int i = 0; i < len; i++)

sum += avgs[i];

return (int)(sum / len);

}

//Insert a value into an array, and shift it down removing

//the first value if array already full

void insert(int val, int \*avgs, int len) {

for (int i = 0; i < len; i++) {

if (avgs[i] == -1) {

avgs[i] = val;

return;

}

}

for (int i = 1; i < len; i++) {

avgs[i - 1] = avgs[i];

}

avgs[len - 1] = val;

}

//Function imported from the arduino website.

//Basically map, but with a curve on the scale (can be non-uniform).

float fscale( float originalMin, float originalMax, float newBegin, float

newEnd, float inputValue, float curve){

float OriginalRange = 0;

float NewRange = 0;

float zeroRefCurVal = 0;

float normalizedCurVal = 0;

float rangedValue = 0;

boolean invFlag = 0;

// condition curve parameter

// limit range

if (curve > 10) curve = 10;

if (curve < -10) curve = -10;

curve = (curve \* -.1) ; // - invert and scale - this seems more intuitive - postive numbers give more weight to high end on output

curve = pow(10, curve); // convert linear scale into lograthimic exponent for other pow function

// Check for out of range inputValues

if (inputValue < originalMin) {

inputValue = originalMin;

}

if (inputValue > originalMax) {

inputValue = originalMax;

}

// Zero Refference the values

OriginalRange = originalMax - originalMin;

if (newEnd > newBegin){

NewRange = newEnd - newBegin;

}

else

{

NewRange = newBegin - newEnd;

invFlag = 1;

}

zeroRefCurVal = inputValue - originalMin;

normalizedCurVal = zeroRefCurVal / OriginalRange; // normalize to 0 - 1 float

// Check for originalMin > originalMax - the math for all other cases i.e. negative numbers seems to work out fine

if (originalMin > originalMax ) {

return 0;

}

if (invFlag == 0){

rangedValue = (pow(normalizedCurVal, curve) \* NewRange) + newBegin;

}

else // invert the ranges

{

rangedValue = newBegin - (pow(normalizedCurVal, curve) \* NewRange);

}

return rangedValue;

}

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

1. Simulación

Diagrama

Descripción generada automáticamente con confianza media

Para la implementación del circuito en proteus pues se usaron los siguientes componentes como el “sound detector”, el Arduino NANO, el led RGB y adicionalmente el circuito integrado ML2596, lo cual no se encontraba en librería, pero se lo pudo crear a partir del datasheet de dicho circuito. En primera lugar, se uso el Arduino Nano, por su tamaño y la cantidad de pines necesarios que se iba a usar en el proyecto, también se escogió el detector de sonido que permitirá amplificar el volumen y entregara una señal analógica que posteriormente estará guiado hacia el Arduino, luego el Arduino procesa esa información y tendrá como salida la señal del led para que se encienda, este funcionamiento se dará de manera eficiente cuando el circuito integrado ML2596 permita regular el voltaje para que los componentes del detector de sonido y del Arduino trabajen de la forma más optima.

1. Conclusiones

Se diseñó un circuito capaz de captar el sonido de cualquier otro dispositivo, con ello a traves del arduino se emite una variación de voltaje dentro de la luces RGB.

Se cumplió con el objetivo del análisis y la simulación del proyecto completo con todos sus componentes

Se usó Proteus 8.11 para la simulación del circuito.

1. Recomendaciones

Se deberían amplificar los conocimientos de Arduino para mejorar el proyecto para futuras versiones.

Se podría añadir más modos de luces al circuito.

Más colaboración por parte de los otros compañeros del grupo