// two ultrasonic distance sensors send D0 the real cm, and D1/2 (range 0 to 30)

// they are sent into dist2led\_sound\_adao2020\_outubro as D0 and D1 respectively

// two leds stripes on pins 3 and 4 of Seeeduino Lotus

#include <FastLED.h>

#define NUM\_LEDS 30 // number of leds n stripe

#define DATA\_PIN\_1 3 // pin for strip 1

#define DATA\_PIN\_2 2 // pin for stripe 2

#define CLOCK\_PIN 13

// Define the array of leds for each stripe

CRGB leds\_0[NUM\_LEDS];

CRGB leds\_1[NUM\_LEDS];

// first led stripe

int H0[NUM\_LEDS]; // array for Hue of each LED

int H0\_[NUM\_LEDS]; // array for desfasamento cores

int V0[NUM\_LEDS]; // array for Value of each LED

int S0[NUM\_LEDS]; // array for Saturation of each LED (unused?)

// second led stripe

int H1[NUM\_LEDS]; // array for Hue of each LED

int H1\_[NUM\_LEDS]; // array for desfasamento cores

int V1[NUM\_LEDS]; // array for Value of each LED

int S1[NUM\_LEDS]; // array for Saturation of each LED (unused?)

// one distance sensor HCSR04 on pin 7

#include "Ultrasonic.h"

Ultrasonic ultrasonic0(6);

Ultrasonic ultrasonic1(7);

float counter;

void setup() {

// sanity check delay - allows reprogramming if accidently blowing power w/leds

delay(2000);

// set up LEDS

// FastLED.addLeds<NEOPIXEL, DATA\_PIN\_1>(leds\_1, NUM\_LEDS); // GRB ordering

// FastLED.addLeds<NEOPIXEL, DATA\_PIN\_2>(leds\_2, NUM\_LEDS);

FastLED.addLeds<WS2811, DATA\_PIN\_1>(leds\_0, NUM\_LEDS); // GRB ordering

FastLED.addLeds<WS2811, DATA\_PIN\_2>(leds\_1, NUM\_LEDS);

// Serial for ultrasonic sensor

Serial.begin(9600);

// H value to desfasamento cores for beat ledstripe

int DESFAS = 16;

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

H0[k]=(30-(k+1))\*8;

Serial.println(H0[k]);

}

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

if ( k < (NUM\_LEDS-DESFAS) ){ H0\_[k] = H0[k+DESFAS]; }

else if ( k >= (NUM\_LEDS-DESFAS) ) {H0\_[k] = H0[k-(NUM\_LEDS-DESFAS)]; }

Serial.println(H0[k]);

}

// // H value to desfasamento cores for sounds ledstripe

//

// for (int k = 0; k < NUM\_LEDS; k++) {

// H1[k]=(30-(k+1))\*8;

// Serial.println(H1[k]);

// }

// for (int k = 0; k < NUM\_LEDS; k++) {

// if ( k < (NUM\_LEDS-DESFAS) ){ H1\_[k] = H1[k+DESFAS]; }

// else if ( k >= (NUM\_LEDS-DESFAS) ) {H1\_[k] = H1[k-(NUM\_LEDS-DESFAS)]; }

// Serial.println(H1[k]);

// }

// set 3 colors for different sounds

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

if ( k < NUM\_LEDS && k > (NUM\_LEDS-10) ){ H1\_[k] = 30; }

else if ( (k < NUM\_LEDS-10) && (k >= NUM\_LEDS-20) ) {H1\_[k] = 115; }

else if ( (k < NUM\_LEDS-20) && (k >= NUM\_LEDS-30) ) {H1\_[k] = 200; }

Serial.println(H1[k]);

}

}

void loop() {

// counter // to engine animations

counter = counter + 1;

if (counter > 2,147,000,000) {counter = 0;}

long D0max = 65; // length of the interaction distance

long D0min = 4; // too close to sensor does not work well

long D1max = 65; // length of the interaction distance

long D1min = 4; // too close to sensor does not work well

// get distance

long D0;

long D1;

D0 = ultrasonic1.MeasureInCentimeters();

D1 = ultrasonic0.MeasureInCentimeters();

// make V luminosity pulsing

//float L = 175 + 75\*abs(sin(((counter)\*6.283)/(D0)));

int L = 255;

// we use the D0 values from 5 to 65!

if (D0>D0max+1 || D0<D0min) {

//Serial.print("D0 ");

Serial.print(0);

// set all stripe to black

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

V0[j] = 0;

}

} else if (D0<=D0max) {

//Serial.print("D0 ");

Serial.print(D0-5);

for( int j = 0; j < (int)(D0-5)/2; j++) {

V0[j] = 0;

}

for( int j = (int)(D0-5)/2; j < NUM\_LEDS; j++) {

V0[j] = L;

}

}

Serial.print(" ");

// we use the D1 values from 5 to 65!

if (D1>D1max+1 || D1<D1min) {

//Serial.print("D1 ");

Serial.println(0);

// set all stripe to black

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

V1[j] = 0;

}

} else if (D1<=D1max) {

//Serial.print("D0 ");

//Serial.println(D1-5);

// light up only 3 leds

int k = int((D1-3)/2);

Serial.println(k);

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

if ( (j >= k-1) & (j <= k+1) ) {

V1[j] = L;

}

else {

V1[j] = 0;

}

}

}

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

leds\_0[k] = CHSV(H0\_[k],255,V0[k]);

leds\_1[k] = CHSV(H1\_[k],255,V1[k]);

}

FastLED.show();

delay(100);

}