```
//AURA SETUP B v1.0
//12. February 2020
//written by ALEX REX
//www.alexrex.de/audioreactive/
//This work is licensed under a Creative Commons
Attribution-ShareAlike 4.0 International License.

#include <Audio.h>
#include <Wire.h>
#include <SPI.h>
#include <SD.h>
#include  <SerialFlash.h>
#include  <TeensyStep.h>

//  AUDIO  BOARD  DECLERATION////////////////////////////////

// GUItool: begin automatically generated code
AudioInputI2S            i2s1;           //xy=253,139
AudioAnalyzeFFT1024      fft1024_1;      //xy=388,279
AudioAnalyzePeak         peak1;          //xy=398,232
AudioOutputI2S           i2s2;           //xy=476,149
AudioConnection           patchCord1(i2s1, 0, i2s2, 0);
AudioConnection           patchCord2(i2s1, 0, i2s2, 1);
AudioConnection           patchCord3(i2s1, 0, peak1, 0);
AudioConnection           patchCord4(i2s1, 0, fft1024_1, 0);
AudioControlSGTL5000     sgtl5000_1;     //xy=335,333
// GUItool: end automatically generated code

#define DEBUG 1      //debug anzeige (Serial.print(...);)
aktivieren/deaktivieren

//PINS FÜR POTIS ZUM EINSTELLEN DES GEWÜNSCHEN
FREQUENZBANDES:
#define  PEAK_LEVEL_PIN  A2
#define  BASS_FREQ_PIN  A6
#define  BASS_SPREAD_PIN  A3

int knob_peak = 0;
```

```
int knob_bass_freq = 0;
int knob_bass_spread = 0;

float fft_boi = 0;     //boi = "bin of interest"
float fft_threshold = 0.050; // 0.090
float peak = 0;

int lastFilterValue = 0;     //variable for storing filter
values
int lastFilterPeak = 0;      //variable for storing filter
values
float weight = 0.5;          //this value adjusts how much a
new value affects the filtered result:

elapsedMillis msecs;  //sometimes simpler than using metro()
or ... chrono()



//  STEPPER  DECLERATION//////////////////////////////////

Stepper motor1(31, 32);      // STEP green jumper cable, DIR
yellow jumper cable
Stepper motor2(29, 30);      // STEP, DIR
Stepper motor3(27, 28);      // STEP, DIR
Stepper motor4(25, 26);      // STEP, DIR

Stepper motor5(34, 33);      // STEP green jumper cable, DIR
yellow jumper cable
Stepper motor6(36, 35);      // STEP, DIR
Stepper motor7(38, 37);      // STEP, DIR
Stepper motor8(24, 39);      // STEP, DIR



StepControl controller1;     // Use default settings
```

```
const int microsteppings = 16;
long umdrehung = 100 * microsteppings; // 200*microsteppings

// AUDIO REACTIVE
DECLERATION//////////////////////////////////

//SETUP////////////////////////////////////////////////
void setup()
{
  // AUDIO BOARD SETUP////////////////////

   Serial.begin(115200);
   Serial.println("testsetup");
  AudioMemory(10);
   sgtl5000_1.enable();
  sgtl5000_1.volume(0.6); // 0.4

  // sgtl5000_1.inputSelect(AUDIO_INPUT_MIC);        //if MIC
is solderd directly to the TEENSY
  sgtl5000_1.inputSelect(AUDIO_INPUT_LINEIN);        //if
LINE-IN INPUT is used

  SPI.setMOSI(7);    //ggf. löschen
  SPI.setSCK(14);    //ggf. löschen
  delay(1000);


  // STEPPER SETUP////////////////////

    int t = 5222; // 4000 is super!

    motor1.setMaxSpeed(1* umdrehung); // steps/s
    motor1.setAcceleration(t); // 150000 is max
    // motor1.setInverseRotation(true);

    motor2.setMaxSpeed(1* umdrehung); //steps/s
    motor2.setAcceleration(t); // 150000 is max
```

```
    motor3.setMaxSpeed(1* umdrehung); //steps/s
    motor3.setAcceleration(t); // 150000 is max

    motor4.setMaxSpeed(1* umdrehung); //steps/s
    motor4.setAcceleration(t); // 150000 is max

    motor5.setMaxSpeed(1* umdrehung); // steps/s
    motor5.setAcceleration(t); // 150000 is max

    motor6.setMaxSpeed(1* umdrehung); //steps/s
    motor6.setAcceleration(t); // 150000 is max

    motor7.setMaxSpeed(1* umdrehung); //steps/s
    motor7.setAcceleration(t); // 150000 is max

    motor8.setMaxSpeed(1* umdrehung); //steps/s
    motor8.setAcceleration(t); // 150000 is max



  // AUDIO REACITVE SETUP/////////////////////

   SpinMotorToAudio();


}


//LOOP/////////////////////////////////////////////
void loop()
{
  if (true)
  {

    //FFT AUDIOANALYSE:
    if (peak1.available())
    {
```

```
      //peak = peak1.read() * 1023.0;
      peak = filter(peak1.read() * 1023.0, 0.2,
lastFilterPeak);
      lastFilterPeak = peak;
    }

     if (fft1024_1.available())
    {
      // each time new FFT data is available
      // print to the Arduino Serial Monitor
      if (DEBUG) {
        Serial.print("FFT: ");
        printNumber(fft1024_1.read(0, 49));    // 0, 24
        printNumber(fft1024_1.read(50, 75));   // 25, 49
        printNumber(fft1024_1.read(75, 119));   // 50, 74
        printNumber(fft1024_1.read(100, 200)); // 75, 100
        printNumber(fft1024_1.read(56, 80));
        printNumber(fft1024_1.read(26, 55));
        printNumber(fft1024_1.read(11, 25));
        printNumber(fft1024_1.read(0, 10));
        Serial.println();

      //MOTOR SPINNING

        SpinMotorToAudio();
      }

      }
  }
}
//LOOP ENDET HIER




//EIGENE  FUNKTIONSDEKLARATIONEN:

//weighted average (IIR) filter (also accepts integers)
```

```
float filter(float rawValue, float weight, float lastValue)
{
  float result = weight * rawValue + (1.0 - weight) *
lastValue;
  return result;
}

void printNumber(float n)
{
    Serial.print(" ");
  if (n >= fft_threshold) {
    Serial.print(n, 3);
    Serial.print(" ");
  } else {
    Serial.print("   -   "); // don't print "0.00"
  }
}









void  SpinMotorToAudio()
{

  {

    float TargetSteps = 55; // 100 or 55

    // CHECKE BÄNDER UND MERKE WELCHER MOTOR MÜSSTE DREHEN
    if ( (fft1024_1.read(0, 49)) >= fft_threshold) // 43 Hz
band width
    {
       motor1.setTargetRel(TargetSteps); // Set target
position to 1000 steps from current position
```

```
      }
    else
    {
       motor1.setTargetRel(0);
    }

    if ( (fft1024_1.read(50, 75)) >= fft_threshold) // 43 Hz
band width
    {
       motor2.setTargetRel(TargetSteps); // Set target
position to 1000 steps from current position

    }
    else
    {
       motor2.setTargetRel(0);
    }


     if ( (fft1024_1.read(75, 119)) >= fft_threshold) // 43
Hz band width
    {
       motor3.setTargetRel(TargetSteps); // Set target
position to 1000 steps from current position

    }
    else
    {
       motor3.setTargetRel(0);
    }


    if ( (fft1024_1.read(120, 149)) >= fft_threshold) // 43
Hz band width
    {
       motor4.setTargetRel(TargetSteps); // Set target
position to 1000 steps from current position
```

```
    }
    else
    {
       motor4.setTargetRel(0);
    }


     if ( (fft1024_1.read(56, 80)) >= fft_threshold) // 43 Hz
band width
    {
       motor5.setTargetRel(TargetSteps); // Set target
position to 1000 steps from current position

    }
    else
    {
       motor5.setTargetRel(0);
    }


     if ( (fft1024_1.read(26, 55)) >= fft_threshold) // 43 Hz
band width
    {
       motor6.setTargetRel(TargetSteps); // Set target
position to 1000 steps from current position

    }
    else
    {
       motor6.setTargetRel(0);
    }


      if ( (fft1024_1.read(11, 25)) >= fft_threshold) // 43
Hz band width
    {
       motor7.setTargetRel(TargetSteps); // Set target
position to 1000 steps from current position
```

```
    }
    else
    {
       motor7.setTargetRel(0);
    }


      if ( (fft1024_1.read(0, 10)) >= fft_threshold) // 43 Hz
band width
    {
       motor8.setTargetRel(TargetSteps); // Set target
position to 1000 steps from current position

    }
    else
    {
       motor8.setTargetRel(0);
    }



      controller1.move(motor1, motor2, motor3, motor4,
motor5, motor6, motor7, motor8);


  }
}
```