while (source->getNextReadPosition() < source->getTotalLength()){

source->getNextAudioBlock(info);

const int numSamples = info.buffer->getNumSamples();

const int numChannels = info.buffer->getNumChannels();

const float \*\*data = info.buffer->getArrayOfReadPointers();

for (int channel = 0; channel < numChannels; channel++){

const float \*channelData = data[channel];

for (int sampleIndex = 0; sampleIndex < numSamples; sampleIndex++) {

if (circularBuffer.process(channelData[sampleIndex])) {

circularBuffer.copyData(buffer);

filter.process(buffer, FFTsize);

dsp::FFT fft{ 10 };

fft.performFrequencyOnlyForwardTransform(buffer);

buffer[0] = 20.0f \* log10f(buffer[0] / (float)FFTsize / 2.0f);

for (int i = 1; i < FFTsize / 2; i++) {

buffer[i] = 20.0f \* log10f((2.0f \* buffer[i]) / (float)FFTsize / 2.0f);}

bool found = false;

for (int i = 0; i < 12; i++) {

Tone t = tones.at(i);

if (t.lookup(buffer, FFTsize / 2)) {

found = true;

if (ripetizioni[12] > 120) { // se c'è un silenzio vado a capo e azzero tutti i contatori dei caratteri

s.push\_back('\n');

for (int j = 0; j < 12; j++)

ripetizioni[j] = 0;}

ripetizioni[12] = 0; // se ho trovato un tono azzero il silenzio trascorso

ripetizioni[i]++;

if (ripetizioni[i] > 12) {

s.push\_back(t.getValore());

ripetizioni[i] = 0;}}}

if (!found) { // se non l'ho trovato conto un silenzio

    ripetizioni[12]++;}}}}}