Using the Arduino Audio Tools in Xeus Jupyterlab

In my AudioTools I have quite a few sound effects and it is quite a challange to test them all. In order to make my life a little bit easier I decided to make my framework usable in **Jupyterlab**.

xeus-cling is a Jupyter kernel for C++ based on the C++ interpreter cling and the native implementation of the Jupyter protocol xeus. So we can use the AudioTools directly in Jupyterlab with Xeus/Cling!

As a precondition I expect that you have Xeus/Cling already installed!

So, first we need to provide the path to the source code. We can use the one that we have installed for Arduino

In [1]:
```
#pragma cling add_include_path("/Users/pschatzmann/Dev/Arduino/libraries/
```

Next we can include the application. We also add AudioLibs/Jupyter.h which provides the API for Jupyter

In [2]:
```
#include "AudioTools.h"
#include "AudioLibs/Jupyter.h"
```

Audio API

Now we are ready to define the audio. Nothing special here:

In [3]:
```
int channels = 2;
int sample_rate = 44100;
int frequency = 800;
SineWaveGenerator<int16_t> sineWave(32000);
sineWave.begin(channels, sample_rate, frequency);
GeneratedSoundStream<int16_t> sound(sineWave);              // Stream gene
```

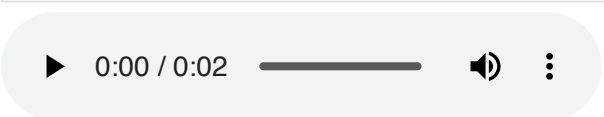
In order to output sound in Jupyterlab we create a JupyterAudio object which defines the generated wav file name, the audio source and the number of buffers and buffer size to limit/specify then length of the generated audio.

In [5]:
```
JupyterAudio audio("test1.wav", sound, 600, 1024);
```

Outputting the audio object is generating a Web Audio Player

In [6]:
```
audio
```

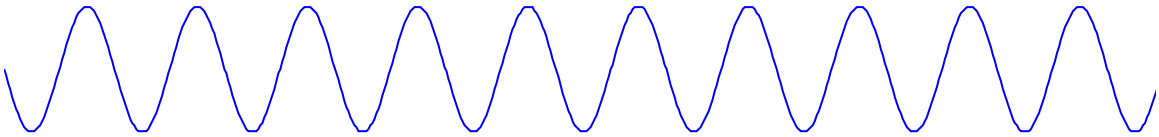
Out[6]:



We can also display the audio as a chart

In [7]:
```
audio.chart(0)
```

Out[7]:



Files

I am also supporting the Arduino SD File API

In [4]:
```
auto file = SD.open("test1.wav", FILE_READ);
file.size()
```

Out[4]:
```
360044
```

In [5]:
```
file.close();
SD.remove("test1.wav");
```

Using C++

We can use standard C++ to process or output data:

In [8]:
```
for (int j=0;j<10;j++){
    std::cout << sineWave.readSample() << endl;
}
```

```
-26351
-28245
-29772
-30913
-31653
-31982
-31896
-31396
-30488
-29185
```