Mannicken Audio GUI tool

Runs in the browser:

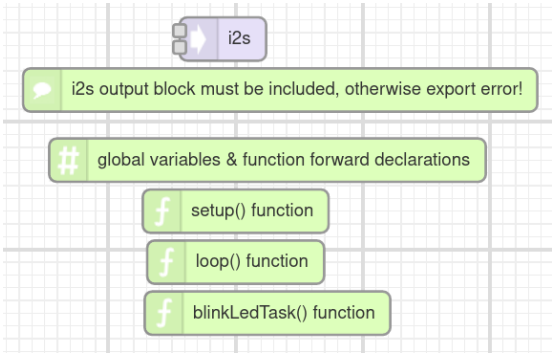
https://manicken.github.io/

Forum:

https://forum.pjrc.com/threads/69109-Audio-Lib-Manicken-design-tool?p=296816#post296816

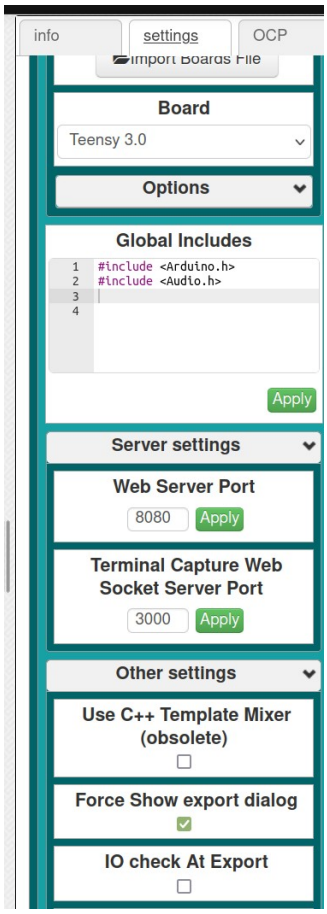
1 Blink example

This non audio example shows how to embeed code into the GUI, so the programming can be done here, before going to the Arduino IDE.



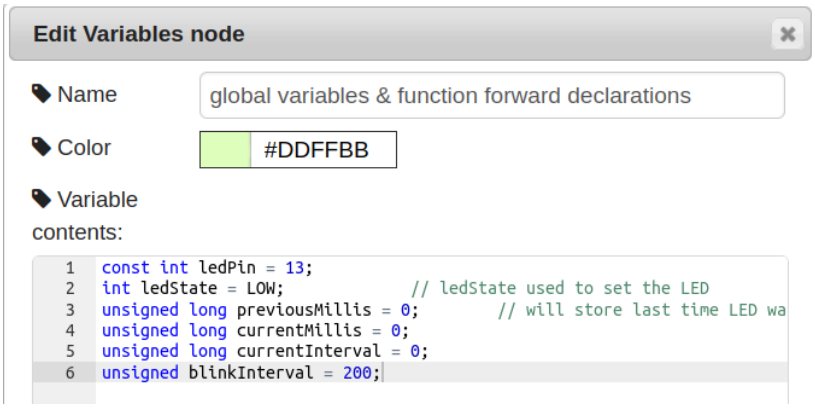
Edit: It is possible to export without the i2s block, see settings – Other settings – IO check at export not checked.

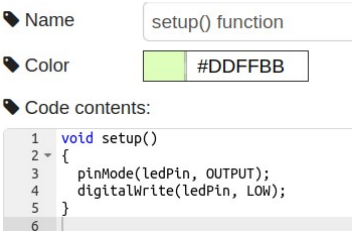
Other interesting setting: Global Includes
If no SD card or other peripherals are used, you can delete the includes that are not necessary.

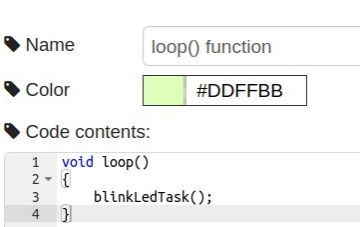


- Place global variables, function and comment tabs (found under “Special”) into the GUI. An output like i2s is also needed, otherwise there will be an export error.

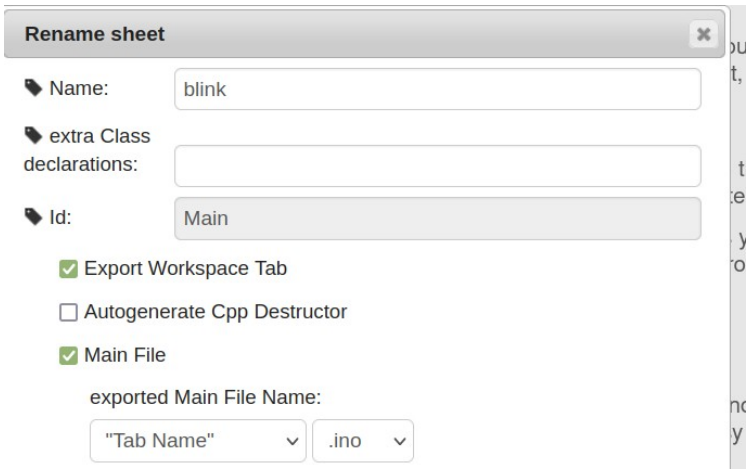
- Double klick on the blocks to name them and to insert code.







By double klicking on the name tab, we can rename it to “blink” and tell the GUI to export it as .ino sketch:



To export the code to the Arduino IDE, we use the “Export – Simple” button.

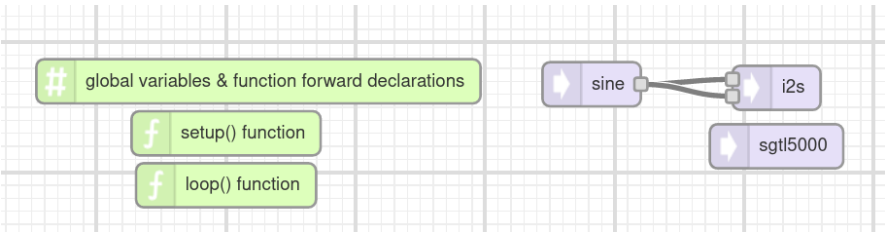
The code can easily be copied and pasted into the Arduino IDE.

There is even a possibility to push it directly to the IDE via webserver.

Another possibility is to use “Export – Class based to zip”, especially for bigger projects.

You get a zip file containing the sketch, together with eventual header files to include, and a JSON file containing a description of the whole sketch. This file can later be used to be imported and so restore the whole project with graphical definitions and code.

2 Hello world blink & audio




```
//global variables + declarations:
const int ledPin = 13;
unsigned blinkInterval = 200;
```

```
void setup()
{
    pinMode(ledPin, OUTPUT);
    digitalWrite(ledPin, LOW);

    AudioMemory(10);
    sgtl5000.enable();
    sgtl5000.volume(0.3);

    sine.frequency(440);
}
```

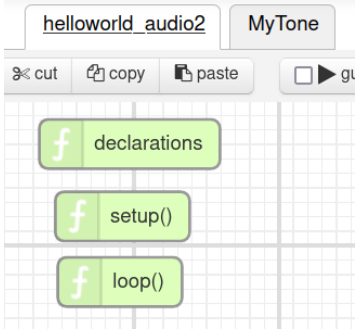
```
void loop()
{
    digitalWrite(ledPin, 1);
    sine.amplitude(0.9);
    delay(250);
    digitalWrite(ledPin, 0);
    sine.amplitude(0);
    delay(1750);
}
```

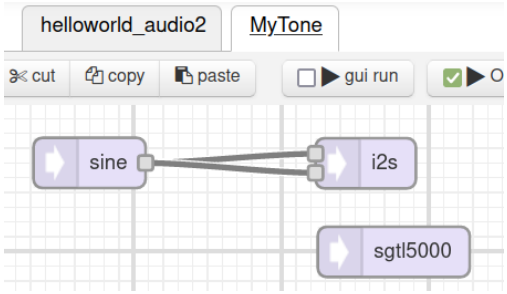
The internal LED blinks and a tone is output every 2 seconds.

3 The same Hello world object oriented

Two Tabs (workspaces) in the GUI: helloworld_audio2 is the main tab generating the .ino file.

MyTone defines a tone generator.





Contents of the function blocks:

`//declarations:`
`#include "MyTone.h"`
`AudioControlSGTL5000 sgtl5000;`
`const int ledPin = 13;`
`MyTone mytone;`
	Always include:
	`AudioControlSGTL5000 sgtl5000;`
	for Audio applications!

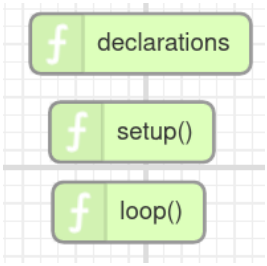
`void setup()`
`{`
`  pinMode(ledPin, OUTPUT);`	

```
    AudioMemory(10);
    sgtl5000.enable();
    sgtl5000.volume(0.3);

    mytone.sine.frequency(440);
}
```

```
void loop()
{
    digitalWrite(ledPin, 1);
    mytone.sine.amplitude(0.9);
    delay(250);
    digitalWrite(ledPin, 0);
    mytone.sine.amplitude(0);
    delay(1750);
}
```

Another way to do it would be to include setfrequency and set_amplitude functions to the MyTone class so that the main tab contains only code:

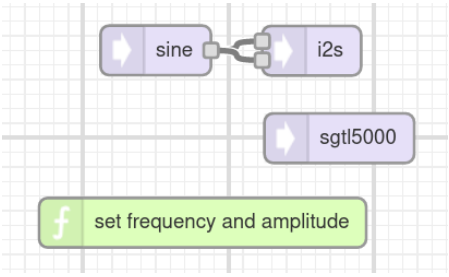



```
#include "MyTone.h"
AudioControlSGTL5000 sgtl5000;
const int ledPin = 13;
MyTone mytone;
```

```
void setup()
{
    pinMode(ledPin, OUTPUT);

    AudioMemory(10);
    sgtl5000.enable();
    sgtl5000.volume(0.3);
    mytone.set_frequency(440);
}
```

```
void loop()
{
    digitalWrite(ledPin, 1);
    mytone.set_amplitude(0.9);
    delay(250);
    digitalWrite(ledPin, 0);
    mytone.set_amplitude(0);
    delay(1750);
}
```




```
//set frequency and amplitude:

void set_frequency(float f){
    sine.frequency(f);
}

void set_amplitude(float a){
    sine.amplitude(a);
}
```

Warning: there should be no audio connections in the Main tab! This is not supported.

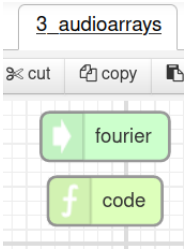
4 Using audio object arrays

When building a polyphonic synthesizer or when using Fourier synthesis it is very practical to have arrays. Many designs you find on the net use a spaghetti schematic in the GUI, as the original Audio GUI does not support arrays. There is also a limitation on the mixer that can only have up to 4 inputs. The Mannicken tool supports very big mixers.

This example also shows how ta block defined in one tab can be used in another tab.

Best look at the tabs from right to left.

Main tab:

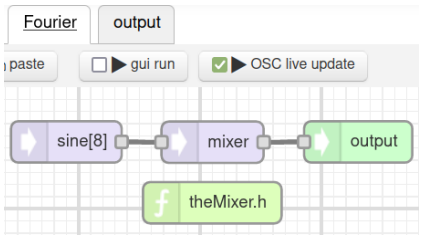



```
const int ledPin = 13;
AudioControlSGTL5000 sgtl5000;
//-----------------------------
void setup()
{
    AudioMemory(10);
    sgtl5000.enable();
    sgtl5000.volume(0.3);

    for (int i=0; i<8, i++){
        fourier.sine[i].frequency((i+1) * 100);
        fourier.sine[i].amplitude(0.2);
    }
}
//-------------------------------
void loop()
{
    digitalWrite(ledPin, 1);
    delay(250);
    digitalWrite(ledPin, 0);
    delay(1750);
}
```

The fourier block is an instance of the Fourier class, found in the palette once the Fourier class is defined.

Fourier tab:

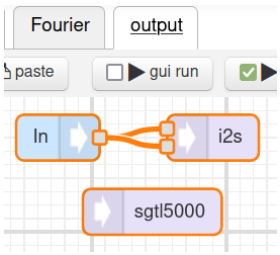


To define an array, simply use “sine[8]” instaed of “sine” for the name.

The mixer is a special object that needs the code “theMixer” found along with the mixer block.

The output block can be found in the palette under “tabs” once it is defined.

Output tab:



The virtual input is found under “special”

The example creates a nice signal: