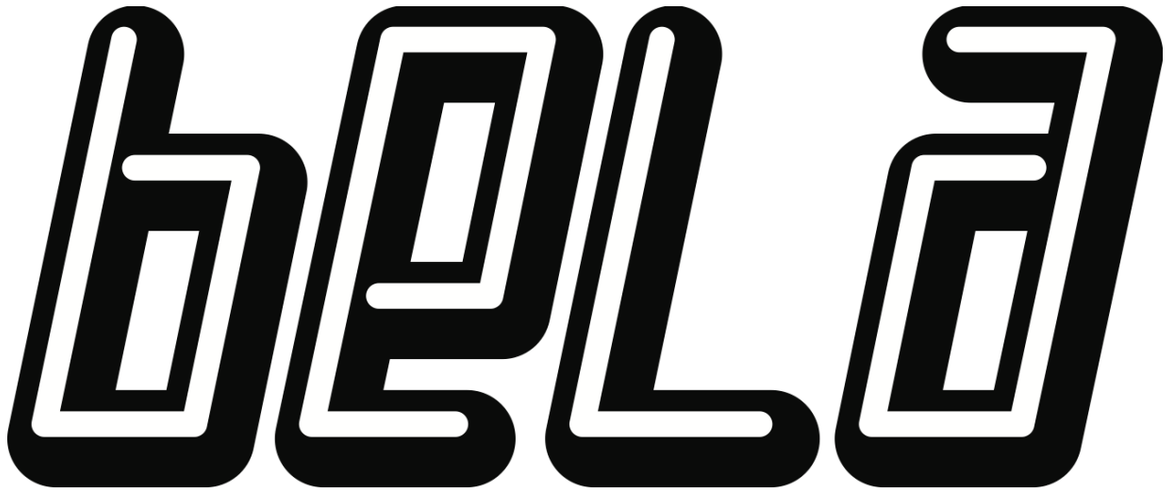
Open-source maker platform for beautifully responsive interactive audio

Adan L. Benito

@BelaPlatform

EPSRC
centre for digital music
Queen Mary University of London

bela.io

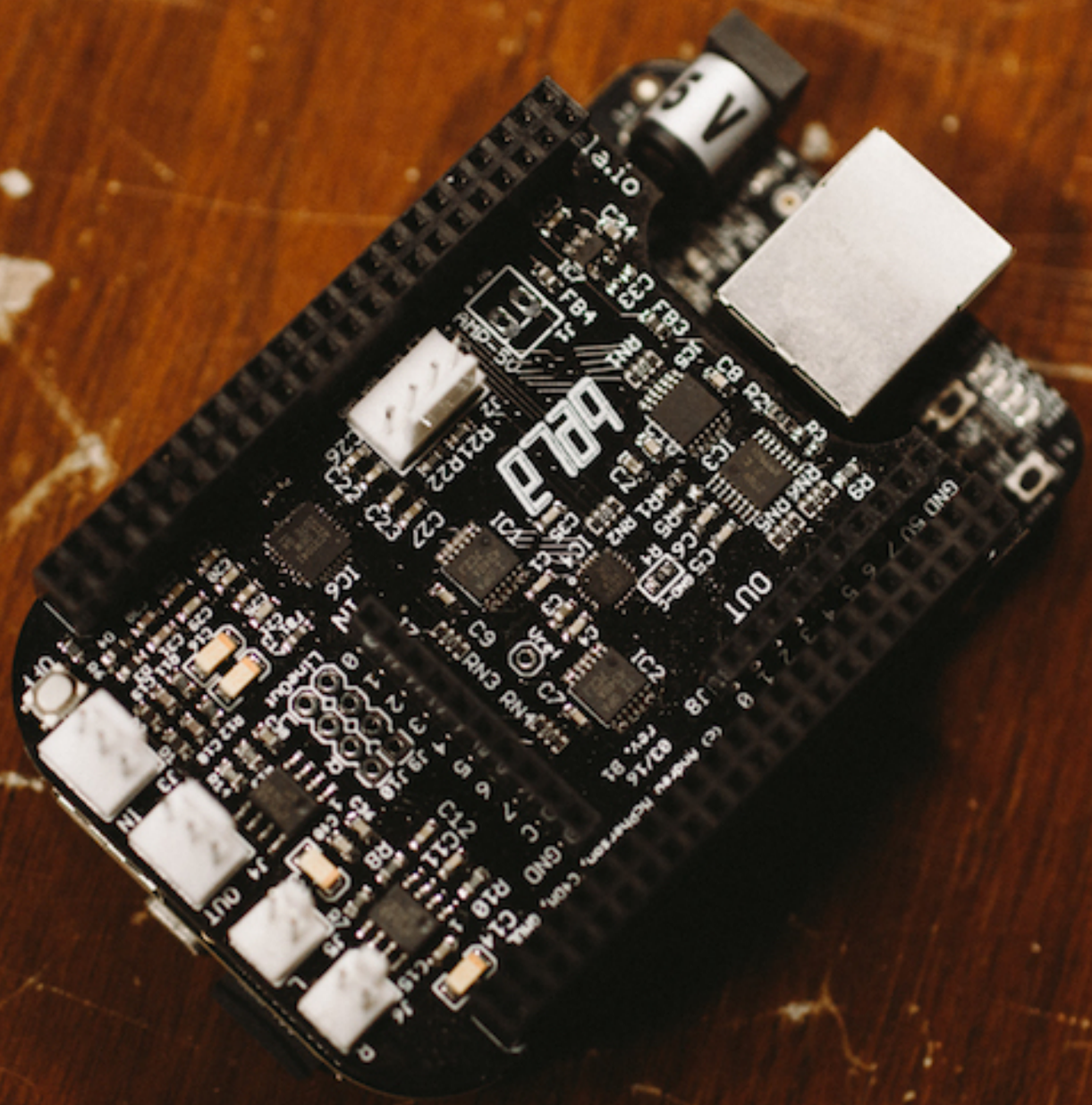


Programmable Audio Workshop

December 14th, 2019

GRAME-CNCM, Lyon

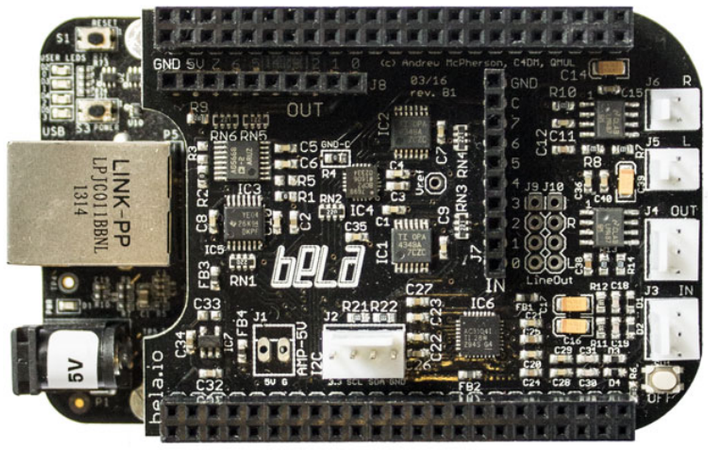
What is Bela?



Augmented Instruments Laboratory

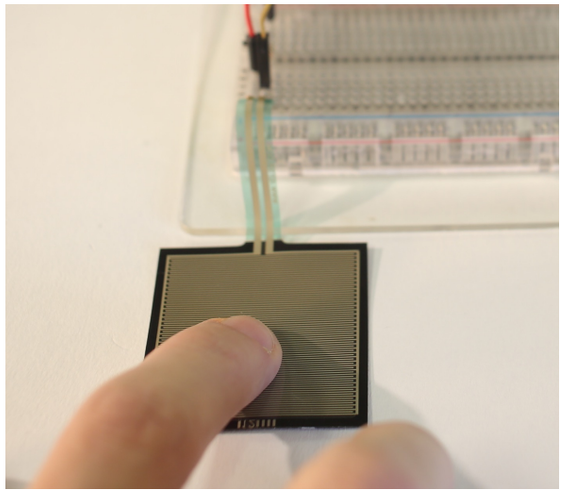
Bela

Embedded computer designed for interactive audio



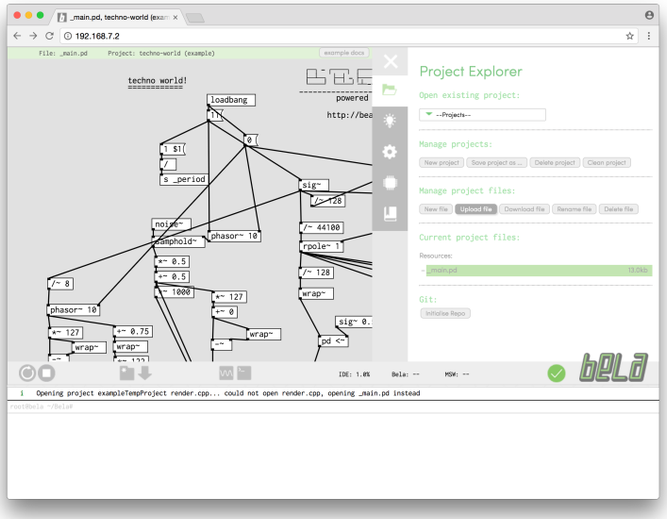
- Power of a single board Linux computer
- Connectivity of a micro-controller
- Combines the benefits of both

New approach to high-bandwidth sensor processing



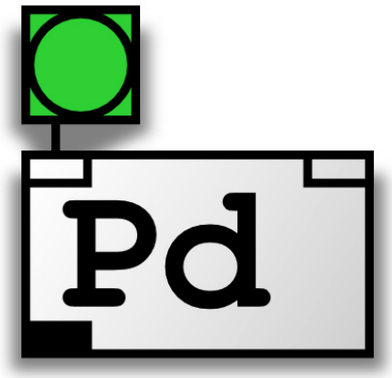
- Analog, digital I/O sampled at audio rate
- Ultra low action-sound latency
- Jitter-free alignment between audio and sensors

Open-source maker platform



- Open hardware and software
- Targeted at musicians, artists
- Online community resources: forum, wiki, blog.

Bela is a polyglot

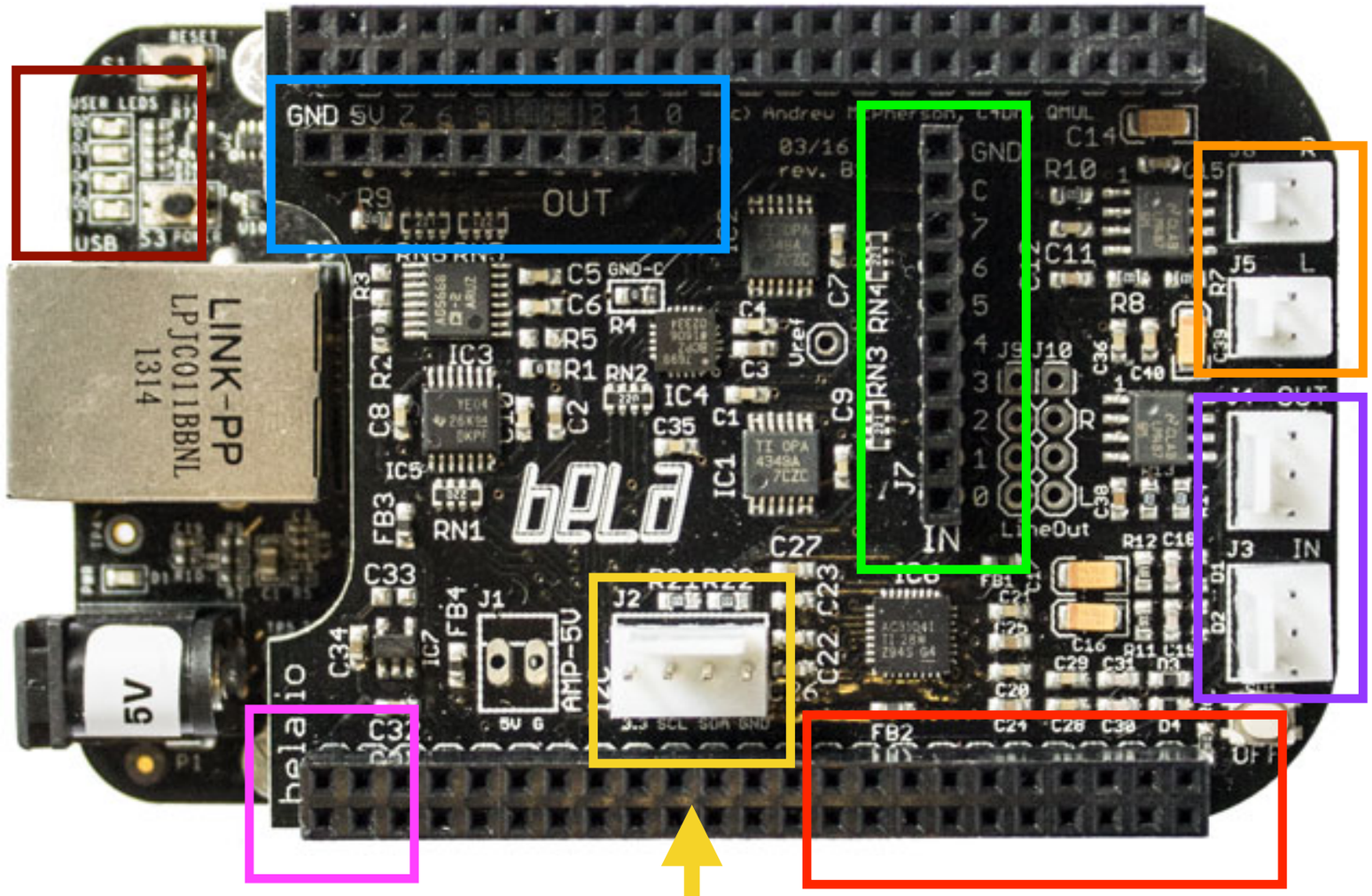


Bela hardware

Analog outputs

Analog inputs

USB



Stereo speaker amps

Stereo audio in /out

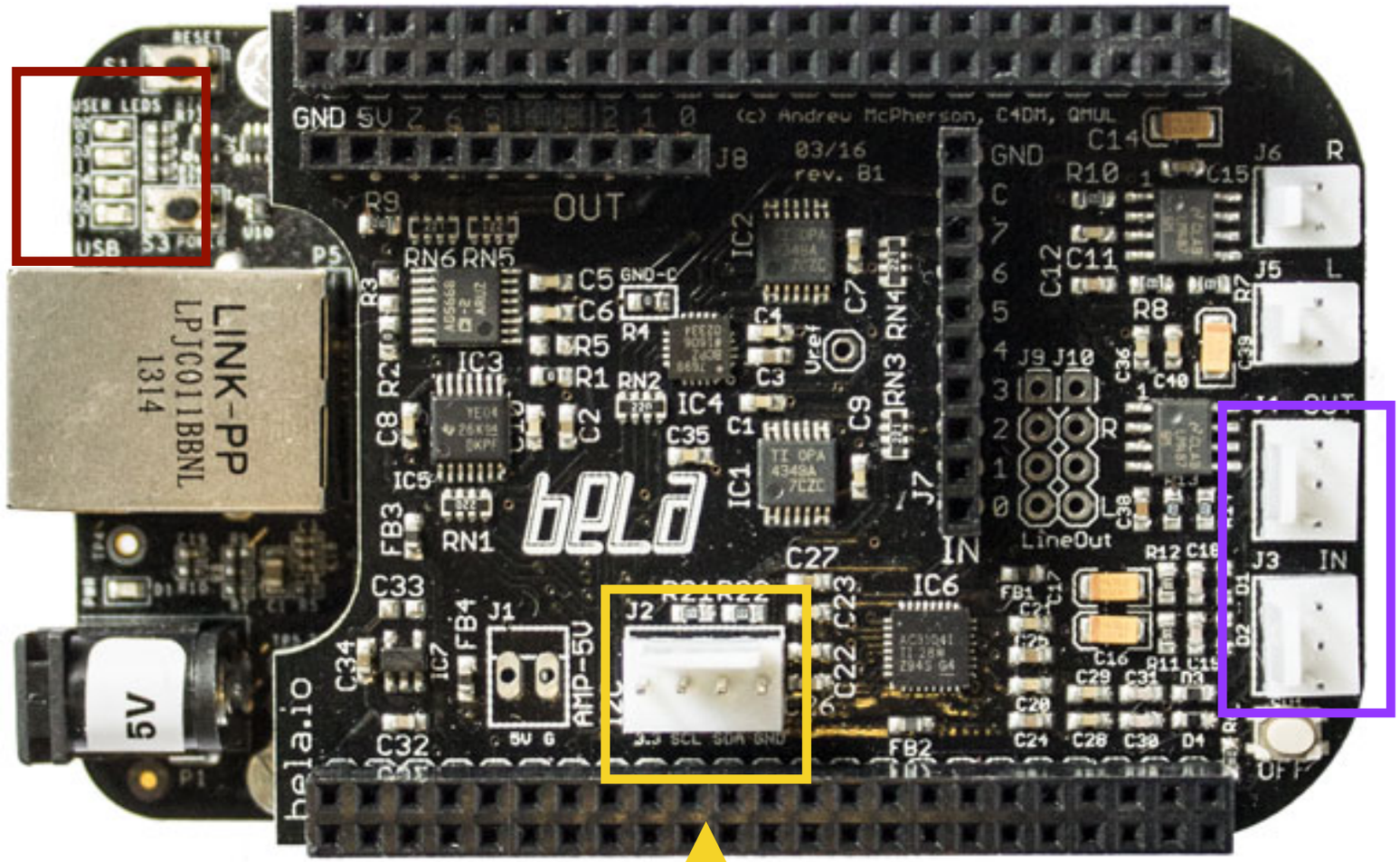
Power / Ground

I2C

Digital inputs/outputs

Bela hardware

USB



Stereo audio in /out

I2C

Required software

Firefox/Chrome browser

The IDE may or may not work in other browsers.

Getting started

1. Plug in Bela to computer (USB)
2. Launch the browser
3. After around 30s go to the url:

MacOS, Linux	http://bela.local/
Windows	http://192.168.6.2

Troubleshooting: drivers

On Mac Yosemite and above (but Catalina), Windows Vista and above, and all Linux no drivers are required.

For older operating systems the drivers can be found on the board

Troubleshooting: drivers

Step 1: Plug in Bela by USB

Wait for it to boot

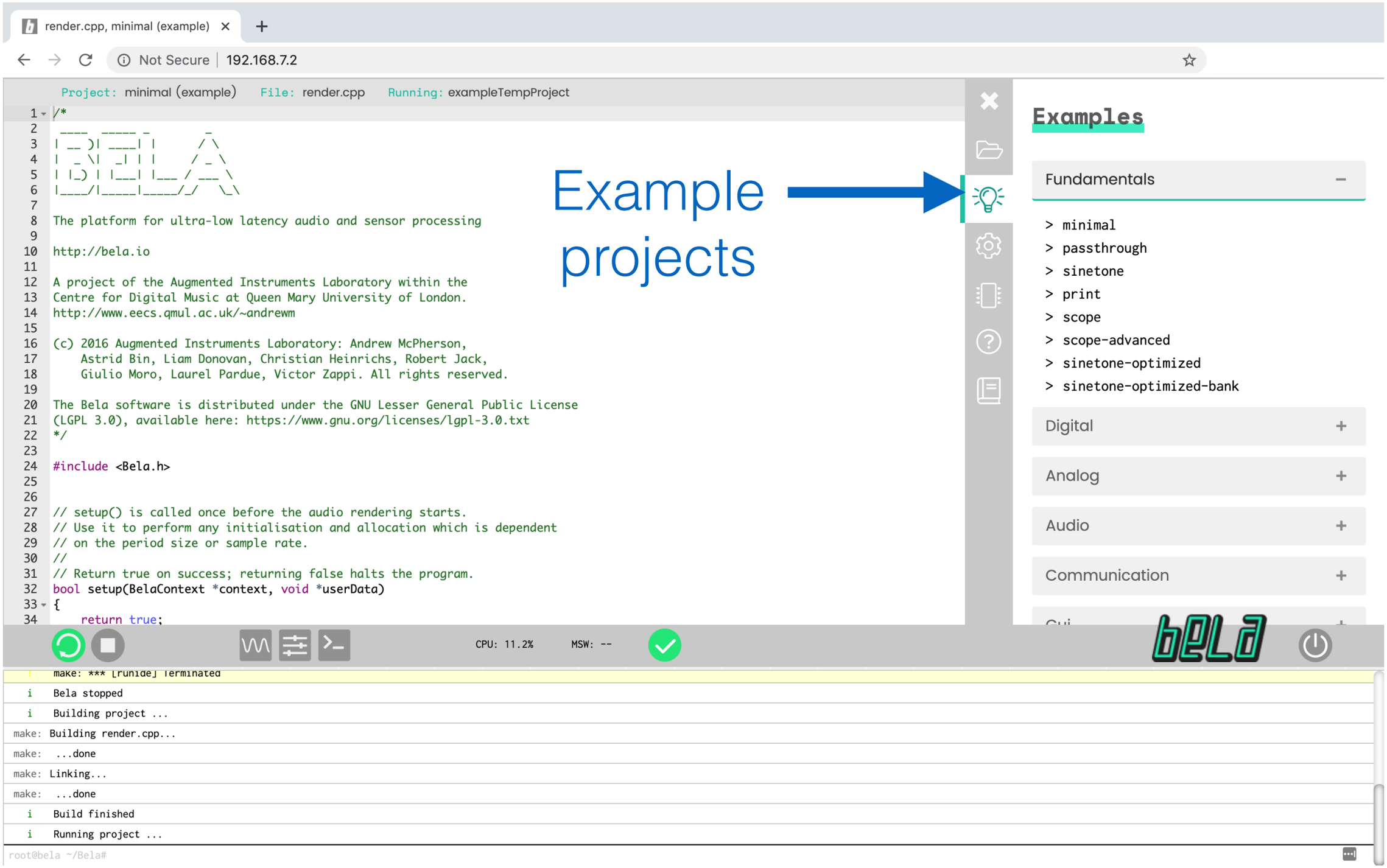
You should see a drive BELABOOT come up

Step 2: Install software from Drivers directory

Step 3: Reboot computer

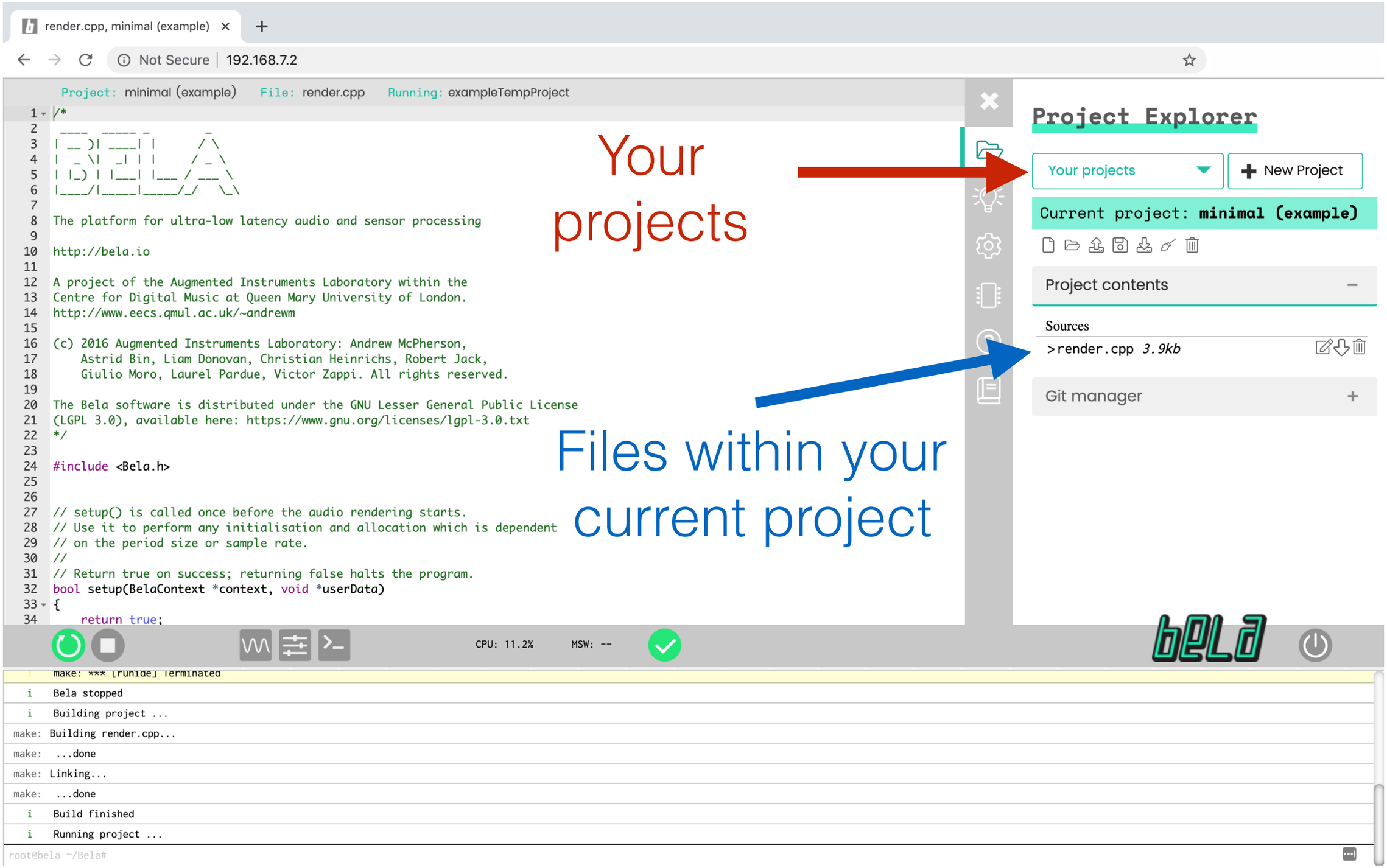
Bring up the Bela IDE:

http://bela.local/



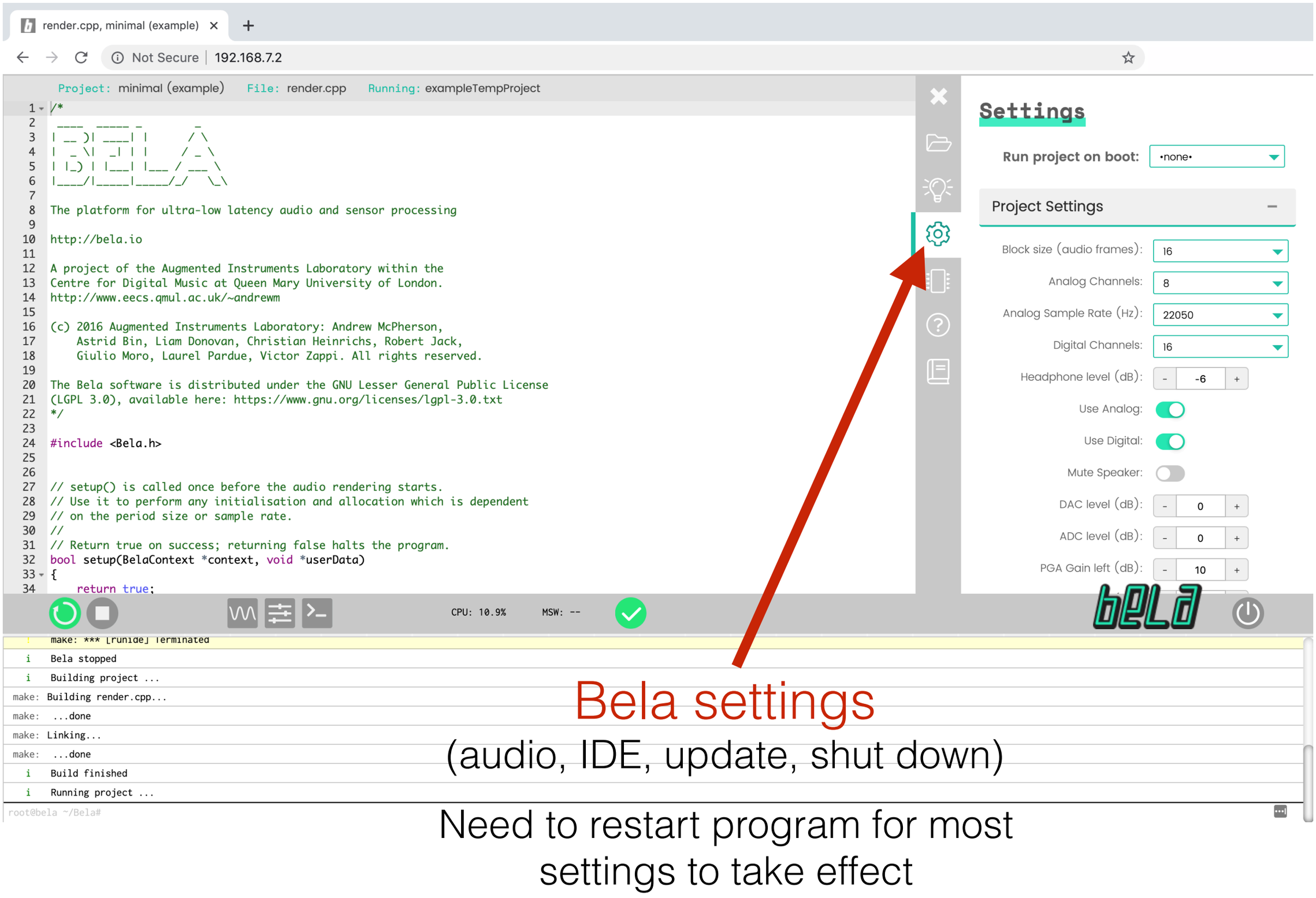
Bring up the Bela IDE:

http://bela.local/



Bring up the Bela IDE:

http://bela.local/



Bela settings

(audio, IDE, update, shut down)

Need to restart program for most settings to take effect

Bring up the Bela IDE:

http://bela.local/

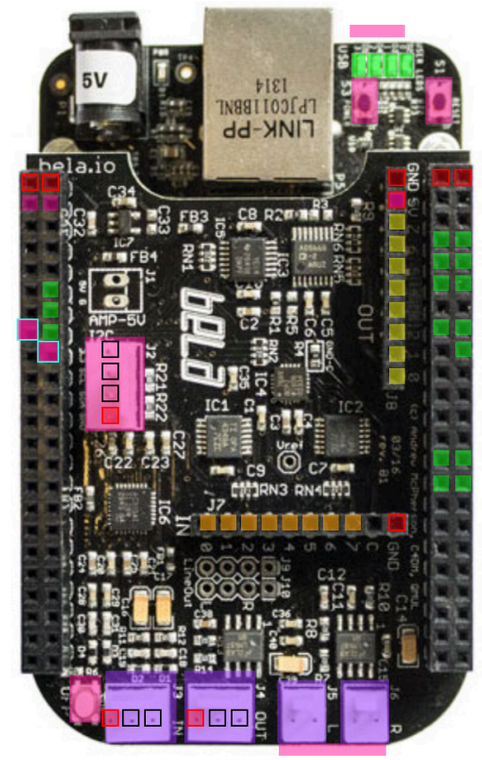
render.cpp, minimal (example)

Not Secure | 192.168.7.2

Project: minimal (example) File: render.cpp Running: exampleTempProject

```
/*
 ____  _____ _        _
| __ )| ____| |      / \
|  _ \|  _| | |     / _ \
| |_) | |___| |___ / ___ \
|____/|_____|_____/_/   \_\

The platform for ultra-low latency audio and sensor processing

http://bela.io

A project of the Augmented Instruments Laboratory within the
Centre for Digital Music at Queen Mary University of London.
http://www.eecs.qmul.ac.uk/~andrewm

(c) 2016 Augmented Instruments Laboratory: Andrew McPherson,
    Astrid Bin, Liam Donovan, Christian Heinrichs, Robert Jack,
    Giulio Moro, Laurel Pardue, Victor Zappi. All rights reserved.

The Bela software is distributed under the GNU Lesser General Public License
(LGPL 3.0), available here: https://www.gnu.org/licenses/lgpl-3.0.txt
*/

#include <Bela.h>


// setup() is called once before the audio rendering starts.
// Use it to perform any initialisation and allocation which is dependent
// on the period size or sample rate.
//
// Return true on success; returning false halts the program.
bool setup(BelaContext *context, void *userData)
{
    return true;
```

Pin Diagram



CPU: 10.8% MSW: --

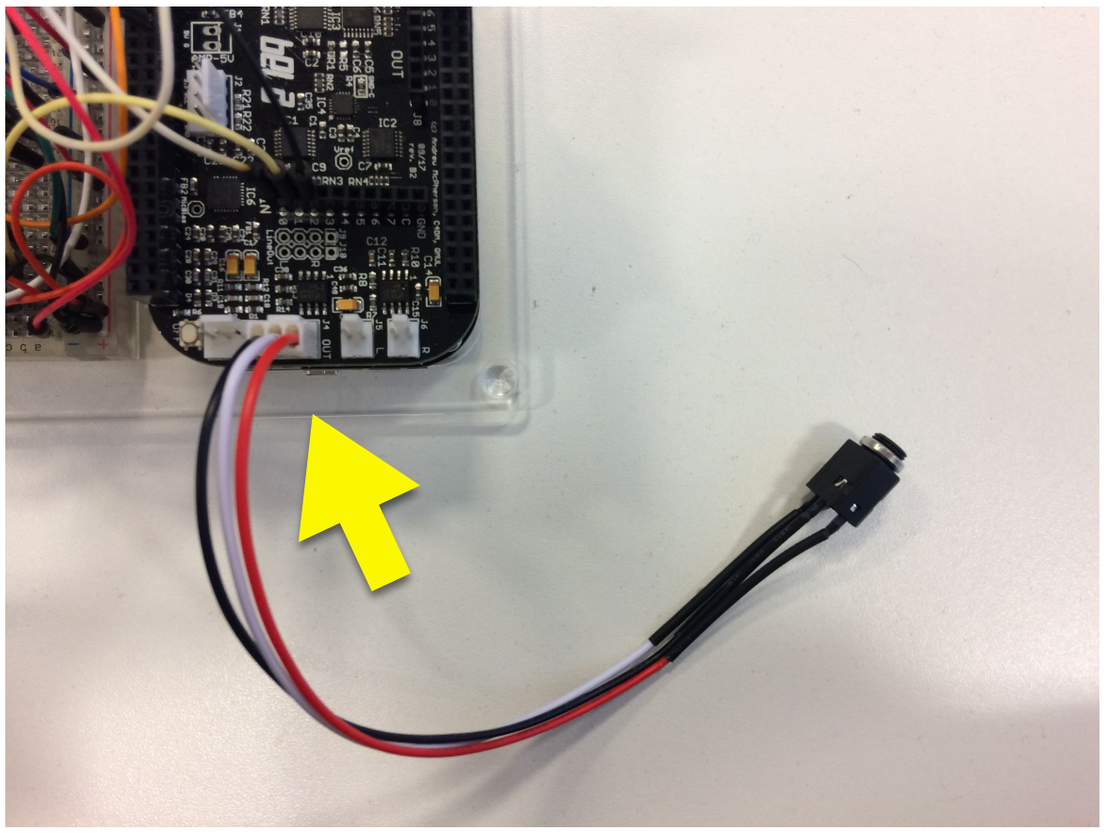
```
make: *** [runIde] Terminated
Bela stopped
Building project ...
make: Building render.cpp...
make:  ...done
make: Linking...
make:  ...done
Build finished
Running project ...
root@bela ~/Bela#
```

Interactive pin diagram

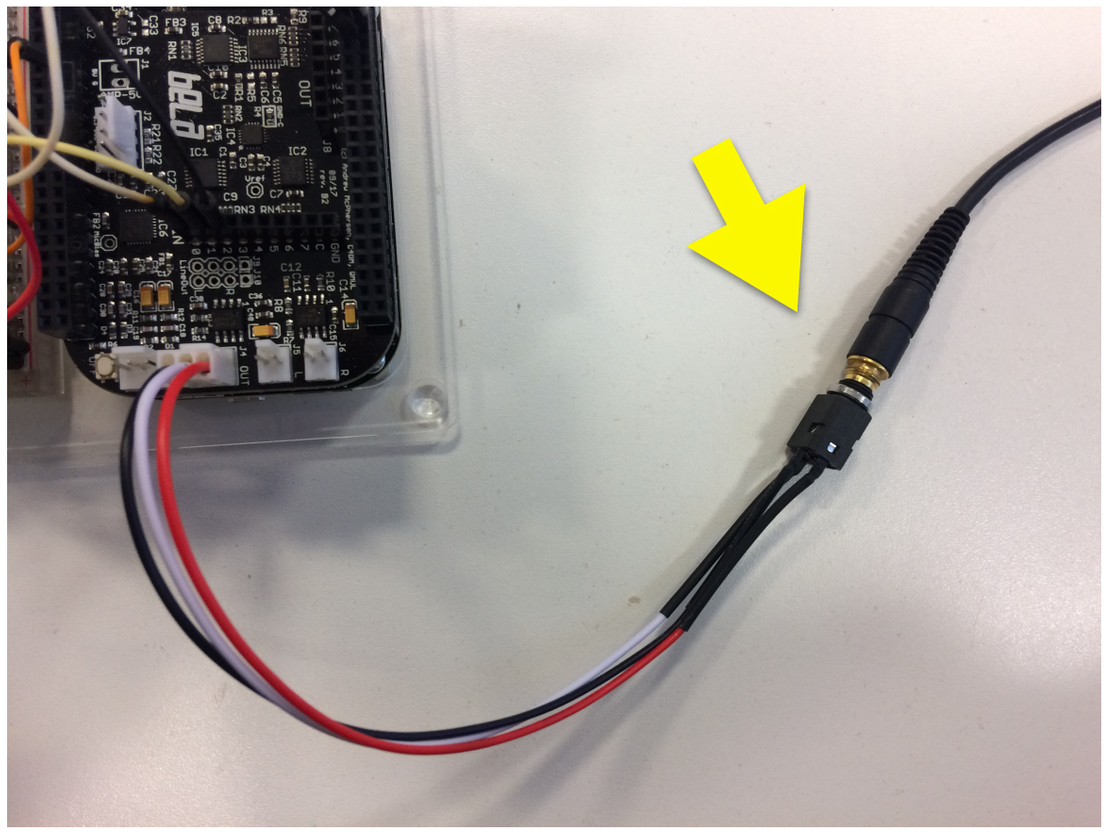
A C++ project is now running **on the board**
not on your personal computer

To listen:

Plug the audio adapter cable into the 2nd audio out molex connector



Plug headphones into the audio adapter cable



render.cpp, minimal (example)

Not Secure | 192.168.7.2

Project: minimal (example) File: render.cpp Running: exampleTempProject

```
/*
 ____  _____ _        _
| __ )| ____| |      / \
|  _ \|  _| | |     / _ \
| |_) | |___| |___ / ___ \
|____/|_____|_____/_/   \_\

The platform for ultra-low latency audio and sensor processing

http://bela.io

A project of the Augmented Instruments Laboratory within the
Centre for Digital Music at Queen Mary University of London.
http://www.eecs.qmul.ac.uk/~andrewm

(c) 2016 Augmented Instruments Laboratory: Andrew McPherson,
    Astrid Bin, Liam Donovan, Christian Heinrichs, Robert Jack,
    Giulio Moro, Laurel Pardue, Victor Zappi. All rights reserved.

The Bela software is distributed under the GNU Lesser General Public License
(LGPL 3.0), available here: https://www.gnu.org/licenses/lgpl-3.0.txt
*/

#include <Bela.h>


// setup() is called once before the audio rendering starts.
// Use it to perform any initialisation and allocation which is dependent
// on the period size or sample rate.
//
// Return true on success; returning false halts the program.
bool setup(BelaContext *context, void *userData)
{
    return true;
```

Settings

Run project on boot: •none•

Project Settings

Block size (audio frames):	16
Analog Channels:	8
Analog Sample Rate (Hz):	22050
Digital Channels:	16
Headphone level (dB):	-6
Use Analog:	on
Use Digital:	on
Mute Speaker:	off
DAC level (dB):	0
ADC level (dB):	0
PGA Gain left (dB):	10

CPU: 10.9% MSW: --

```
!  make: *** [runIde] Terminated
i  Bela stopped
i  Building project ...
make: Building render.cpp...
make:  ...done
make: Linking...
make:  ...done
i  Build finished
i  Running project ...
root@bela ~/Bela#
```

PGA Gain

Changes the level of the microphone

C++ API

- In **`render.cpp`**....
- Three main functions:
- **`setup()`**
  runs once at the beginning, before audio starts
  gives channel and sample rate info
- **`render()`**
  called repeatedly by Bela system ("callback")
  passes input and output buffers for audio and sensors
- **`cleanup()`**
  runs once at end
  release any resources you have used
- Code docs available in sidebar of IDE, or at docs.bela.io

Hello world: sinetone

```
#include <Bela.h>
#include <cmath>

float gPhase = 0.0; /* Phase of the oscillator (global variable) */

void render(BelaContext *context, void *userData)
{
    /* Iterate over the number of audio frames */
    for(unsigned int n = 0; n < context->audioFrames; n++) {
        /* Calculate the output sample based on the phase */
        float out = 0.8 * sinf(gPhase);

        /* Update the phase according to the frequency */
        gPhase += 2.0 * M_PI * gFrequency * gInverseSampleRate;
        if(gPhase > 2.0 * M_PI)
            gPhase -= 2.0 * M_PI;

        for(unsigned int channel = 0;
            channel < context->audioOutChannels; channel++) {
            /* Store the output in every audio channel */
            audioWrite(context, n, channel, out);
        }
    }
}
```

This runs once per block

This runs once per sample in the block (audioFrames gives the number)

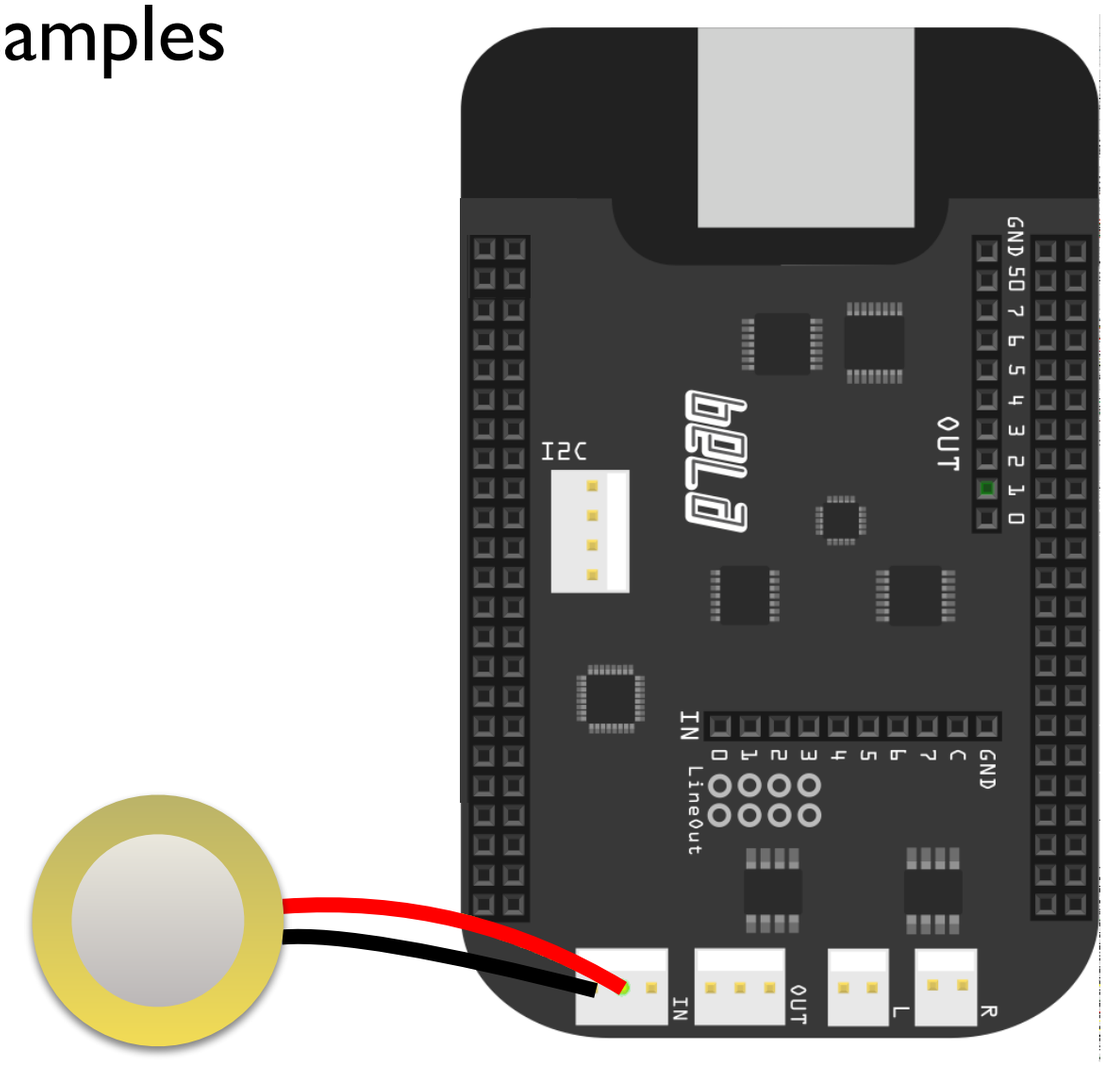
This runs twice per frame, once for each channel

write to buffer of interleaved audio data, specifying a frame, a channel and a value

Piezo Mic

Step 0

- Triggering samples



projects/piezo-trigger-task

Piezo Mic

Step 0

```
32  void render(BelaContext *context, void *userData)
33  {
34
```

CPU: 15.2%

```
!   make: *** [runIde] Terminated
i   Bela stopped
```

Scope

projects/piezo-trigger-task

Piezo Mic

Step 0

- Triggering samples: how?

projects/piezo-trigger-task

Piezo Mic

Step 1

- Triggering samples: how?
  - Rectify piezo signal:

```
piezoRectified = gPiezoInput;

// Full wave rectification
if(piezoRectified < 0)
    piezoRectified *= -1.0f;

// log audio input and rectified input
scope.log(gPiezoInput, piezoRectified);
```

projects/piezo-trigger-task

Piezo Mic

Step II

- Triggering samples: how?
  - Peak detection:

```
// Peak Detection

if(piezoRectified >= peakValue) {
    peakValue = piezoRectified;
    triggered = 0;
} else if(peakValue >= rolloffRate) {
    peakValue -= rolloffRate;
}


scope.log(gPiezoInput, peakValue);
```

projects/piezo-trigger-task

Piezo Mic

Step III

- Triggering samples: how?
  - Primitive onset detection:

```
// Threshold peak to trigger

if(piezoRectified < peakValue - amountBelowPeak
     && peakValue >= thresholdToTrigger && !triggered) {

            triggered = 1;

            // Start sample playback
            gReadPtr = 0;
}
```

projects/piezo-trigger-task

TRILL

touch sensing
for makers

bela.io/trill

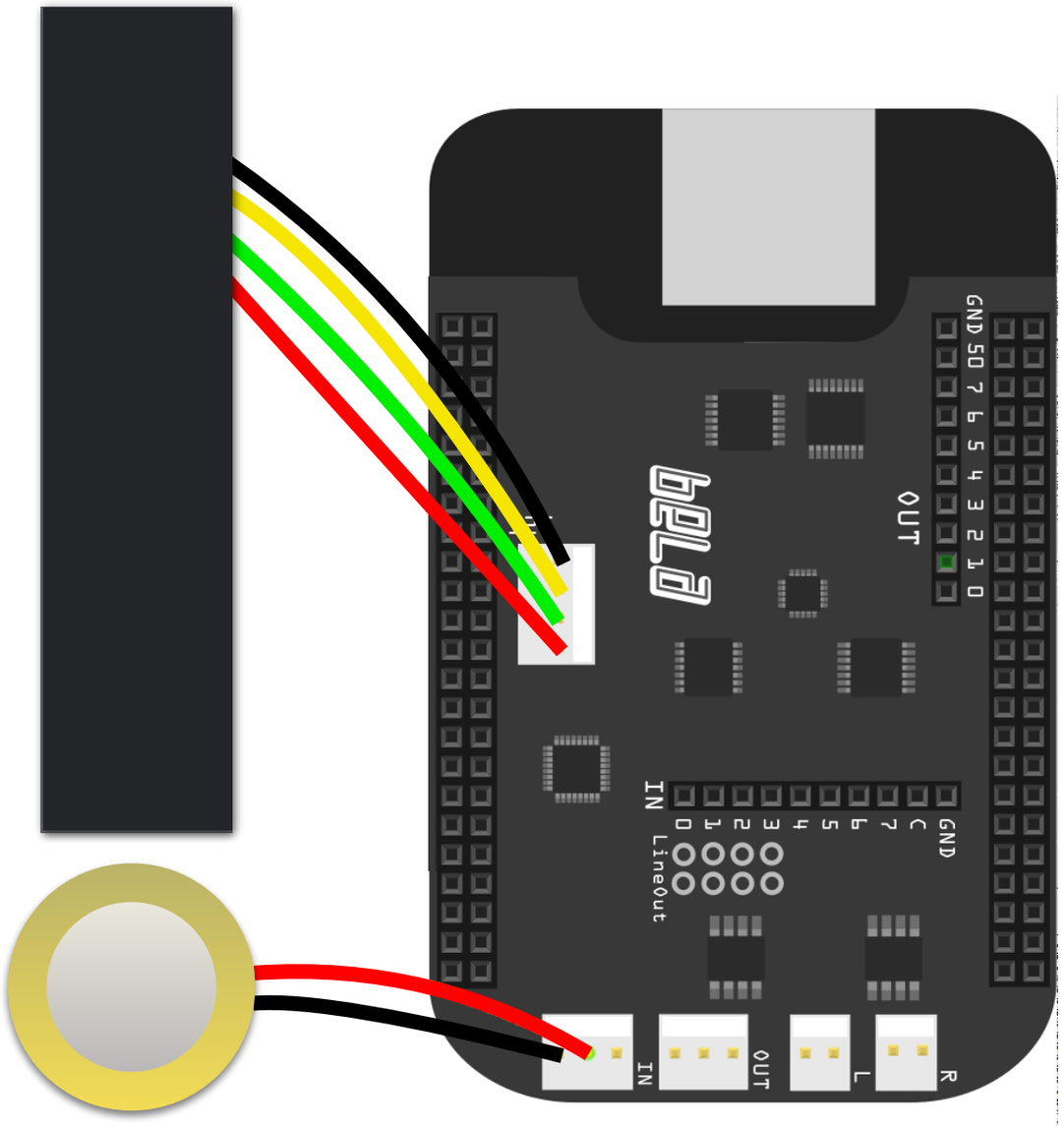
DESIGNED BY BELA

TRILL BAR

- 101 x 22mm
- Resolution of <0.1mm
- 1 axis of sensing
- Multi-touch (up to 5 fingers)
- Touch size sensing
- 5ms reading latency
- Can be cut down to 40 x 14mm

Trill Bar

Step 0



projects/trill-bar-visual

Trill Bar

Step 1

- Library: `#include <libraries/Trill/Trill.h>`
- Read I2C devices
- Auxiliary Task - lower priority than audio thread

```
AuxiliaryTask readI2CTask;

readI2CTask = Bela_createAuxiliaryTask(I2Cloop, 50, "I2C-read", NULL);

Bela_scheduleAuxiliaryTask(readI2CTask);
```

projects/trill-bar-visual

Trill Bar

Step 1

- Library: `#include <libraries/Trill/Trill.h>`
- Read I2C devices
- Auxiliary Task - lower priority than audio thread

```
AuxiliaryTask readI2CTask;

readI2CTask = Bela_createAuxiliaryTask(I2Cloop, 50, "I2C-read", NULL);

Bela_scheduleAuxiliaryTask(readI2CTask);
```

projects/trill-bar-visual

Trill Bar

Step II

- The loop

```
void I2Cloop(void*)
{
    // loop
    while(!gShouldStop)
    {
        // DO STUFF
        usleep(gTaskSleepTime);
    }
}
```

projects/trill-bar-visual

Trill Bar

Step III

- Library: `#include <libraries/Trill/Trill.h>`

```
Trill touchSensor;
```

```
void setup(…) {
  // Touch sensor setup: I2C bus, address, mode, threshold, prescaler
  touchSensor.setup(1, 0x18, Trill::NORMAL, 50, 1)
```

```
void I2Cloop(void*)(…) {
  touchSensor.readLocations();
  for(int i = 0; i <  touchSensor.numberOfTouches(); i++) {
          touchSensor.touchLocation(i); // Location
          touchSensor.touchSize(i);     // Size
  }
```

projects/trill-bar-visual

Trill Bar

Step IV

- The GUI: *p5.js* sketches!
- Library: `#include <libraries/Gui/Gui.h>`

```
void setup(…) {
   // Setup GUI
   gui.setup(context->projectName);
```

```
void render(…) {
     // after some time has elapsed.
     if(count >= gTimePeriod*context->audioSampleRate)
     {
          gui.sendBuffer(0, gNumActiveTouches);
          // […]
          count = 0;
     }
     count++;
```

projects/trill-bar-visual

Trill Bar

Step IV

```
32  void render(BelaContext *context, void *userData)
33  {
34
```

CPU: 15.2%

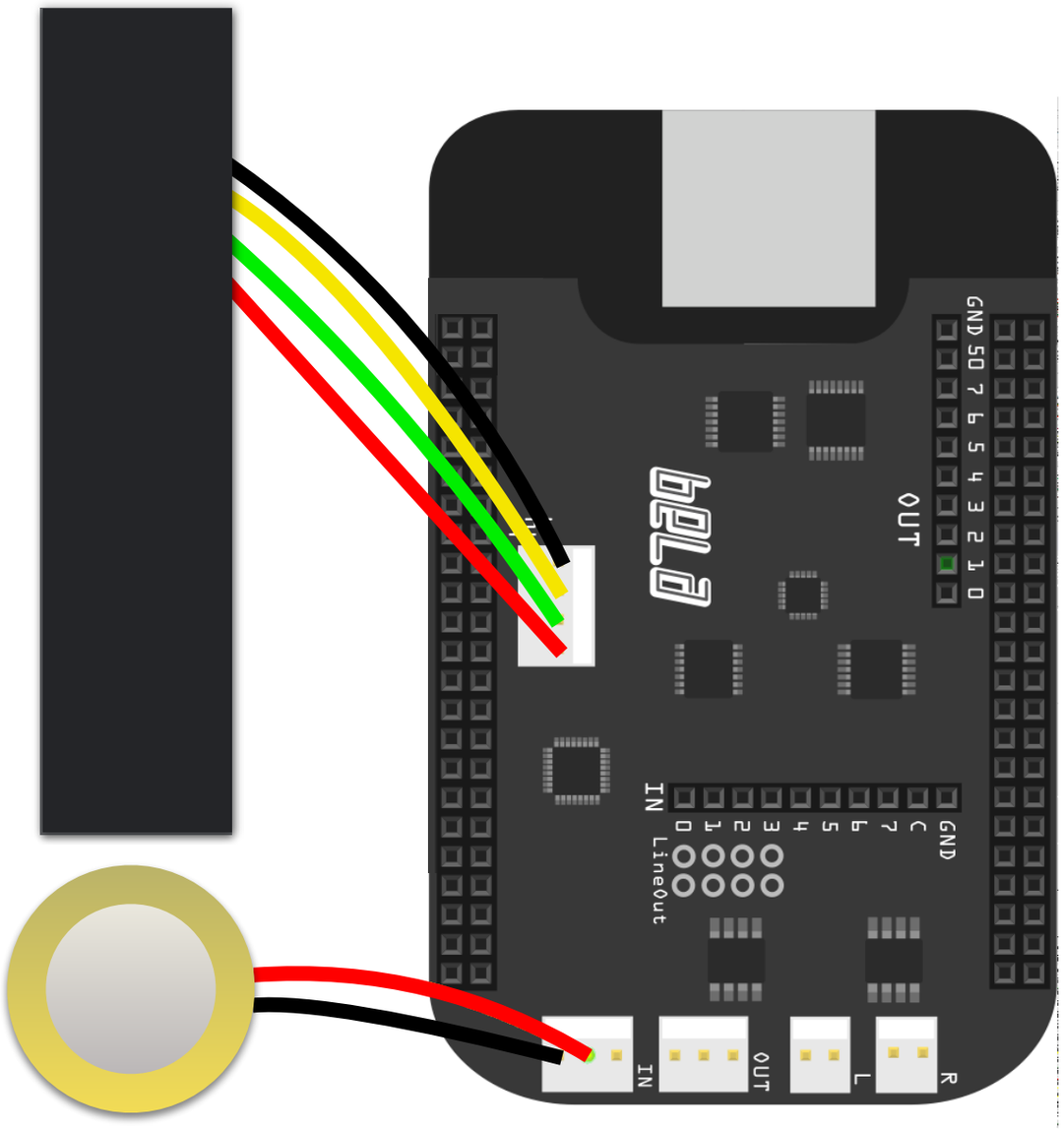
```
!   make: *** [runIde] Terminated
i   Bela stopped
```

GUI

projects/trill-bar-visual

Resonator

- Resonator model: tuned filter bank



projects/resonator-basic

Resonator

- Resonator model `#include "Resonators.h"`

```
Resonator resonator;
ResonatorOptions options; // will initialise to default
```

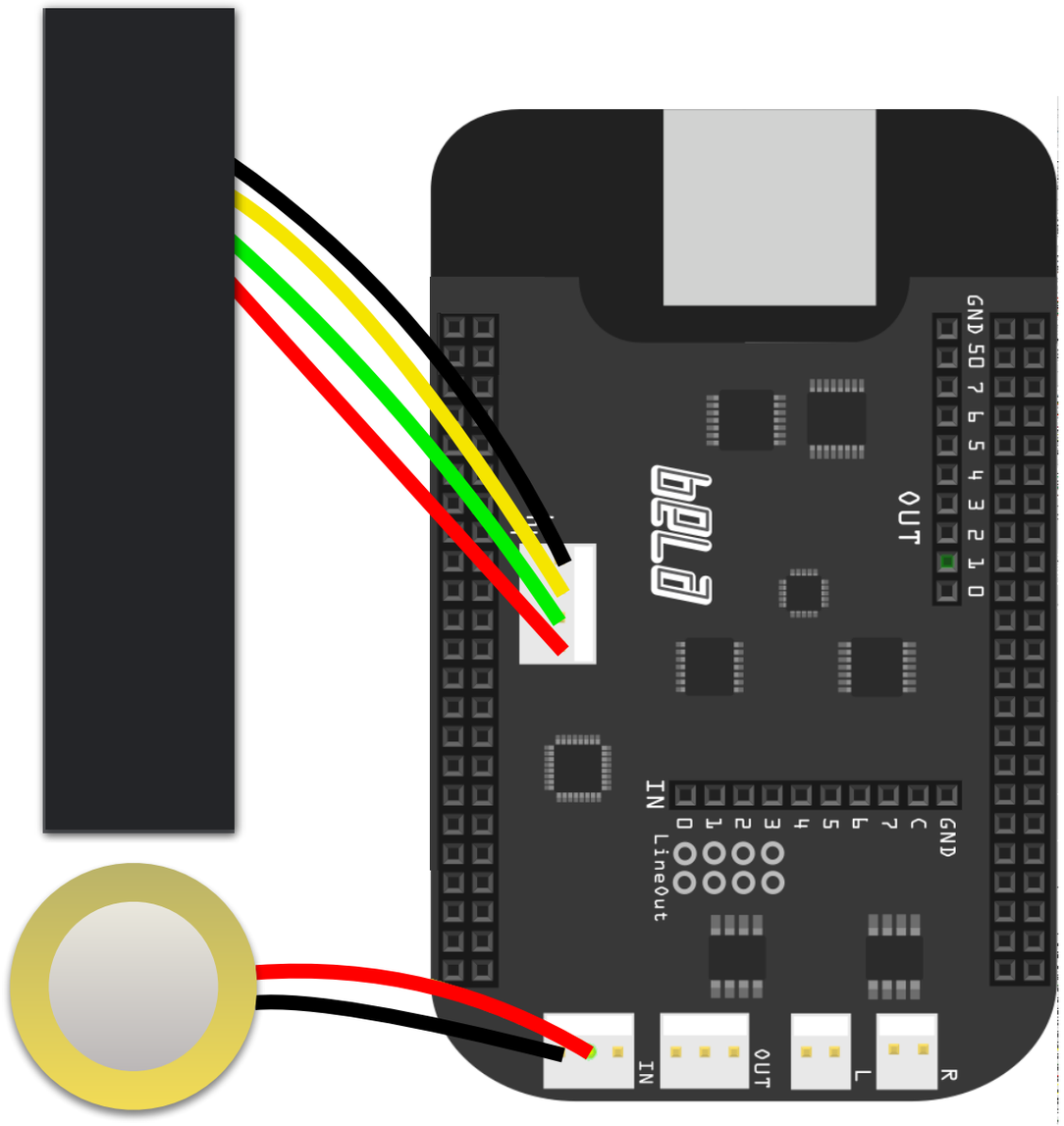
```
void setup(…) {
  resonator.setup(options, context->audioSampleRate, context->audioFrames);
  resonator.setParameters(432, 0.5, 0.05); // freq, gain, decay
```

```
void render(…) {
  float in = audioRead(context, n, 0); // an excitation
  float out = resonator.render(in);

  // write to output
```

projects/resonator-basic

All together

Trill + resonator



projects/trill-resonator

All together

Trill + resonator

```
if(gNumActiveTouches > 0) {
  gResFreq =  map(gTouchLocation[0], 0, 1, gFreqRange[0], gFreqRange[1]);
}

// If enough samples have elapsed ...
if(count > gResonatorUpdateRate * context->audioSampleRate) {
   // Set resonator parameters
   res.setParameter(Resonator::kFreq, gResFreq);   // freq
   res.update();

   count = 0;
}
count++;
```

projects/trill-resonator

All together

Trill + resonator

- Filtering

```
OnePole lowpass;
float gLpCutOff = 5;
```

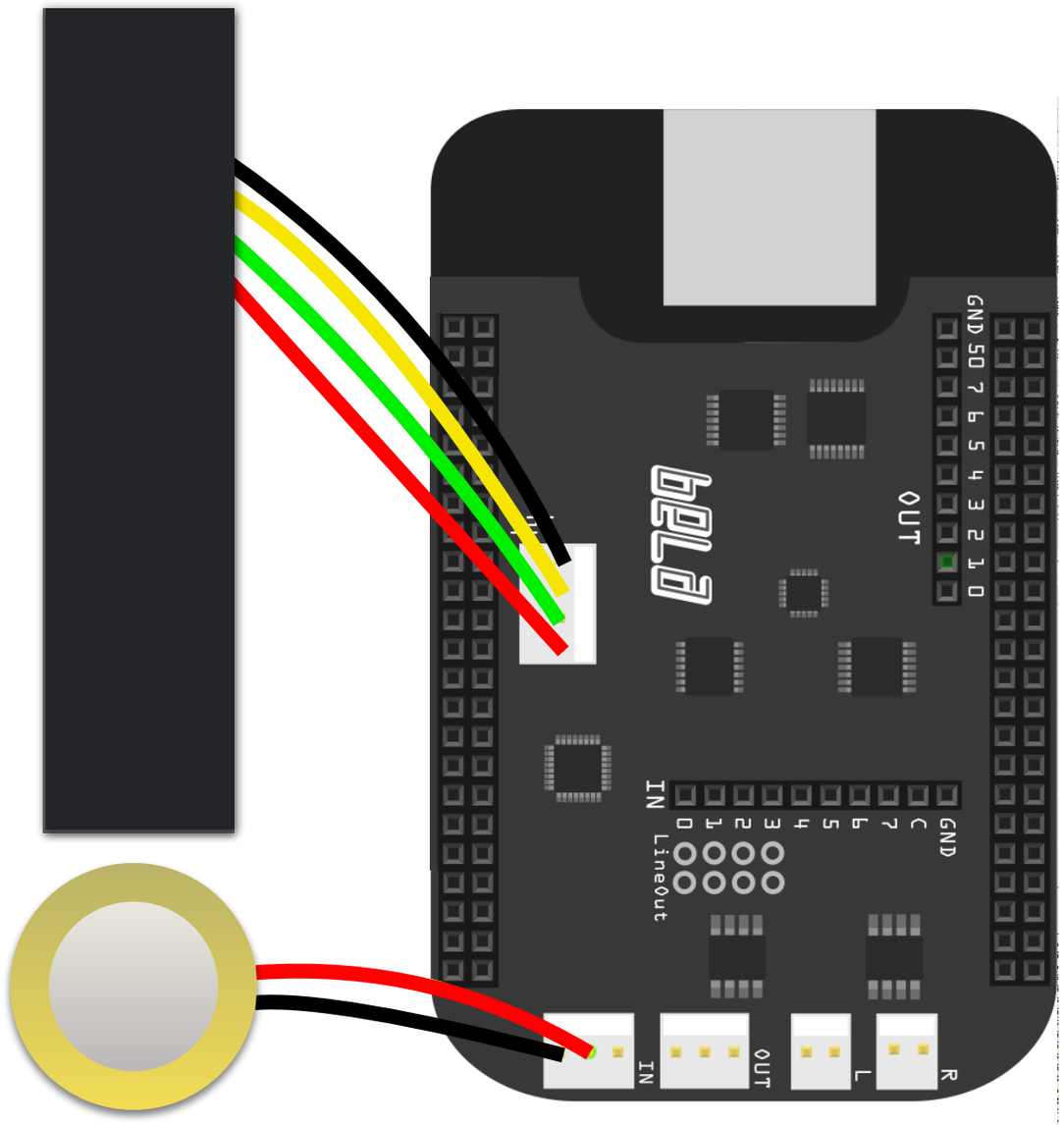
```
void setup(…) {
  lowpass.setup(gLpCutOff, context->audioSampleRate);
```

```
void render(…) {
  float freq = lowpass.process(gResFreq);
```

projects/trill-resonator

Finally…

Trill + resonator-bank



projects/trill-res-bank

Finally…

Trill + resonator-bank

```
void setup(…) {

  model.load("models/marimba.json");

  resBank.setup(resBankOptions, context->audioSampleRate, context->audioFrames);

  resBank.setBank(model.getShiftedToFreq(gResFreq));
```

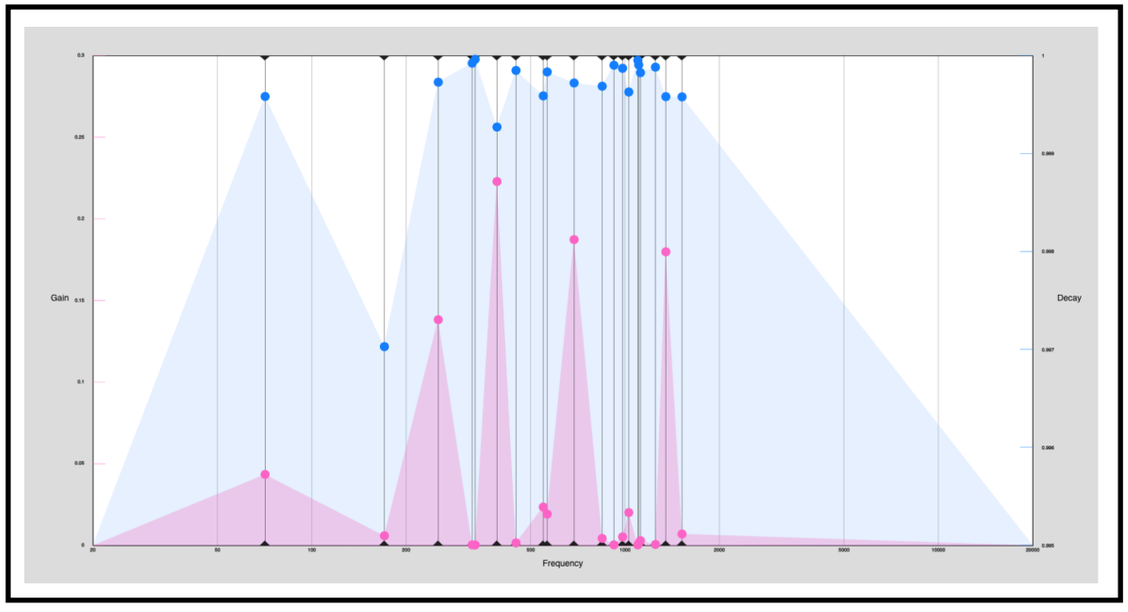
projects/trill-res-bank

Resonators

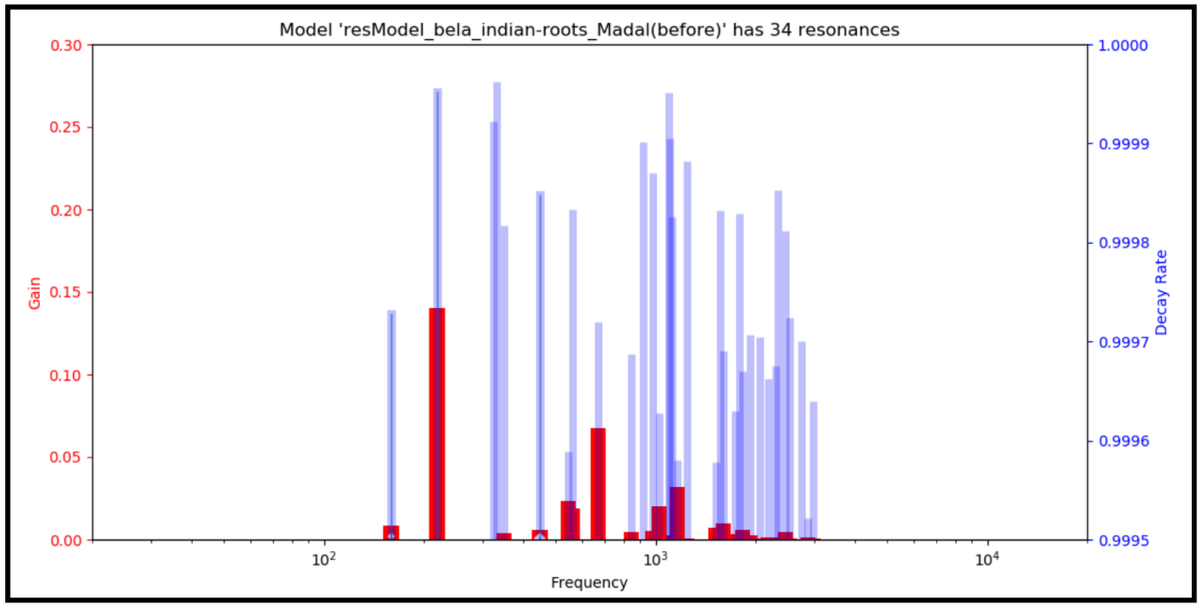
https://github.com/jarmitage/resonators

- based on the [resonators~] Max/Pd object:

p5.js GUI



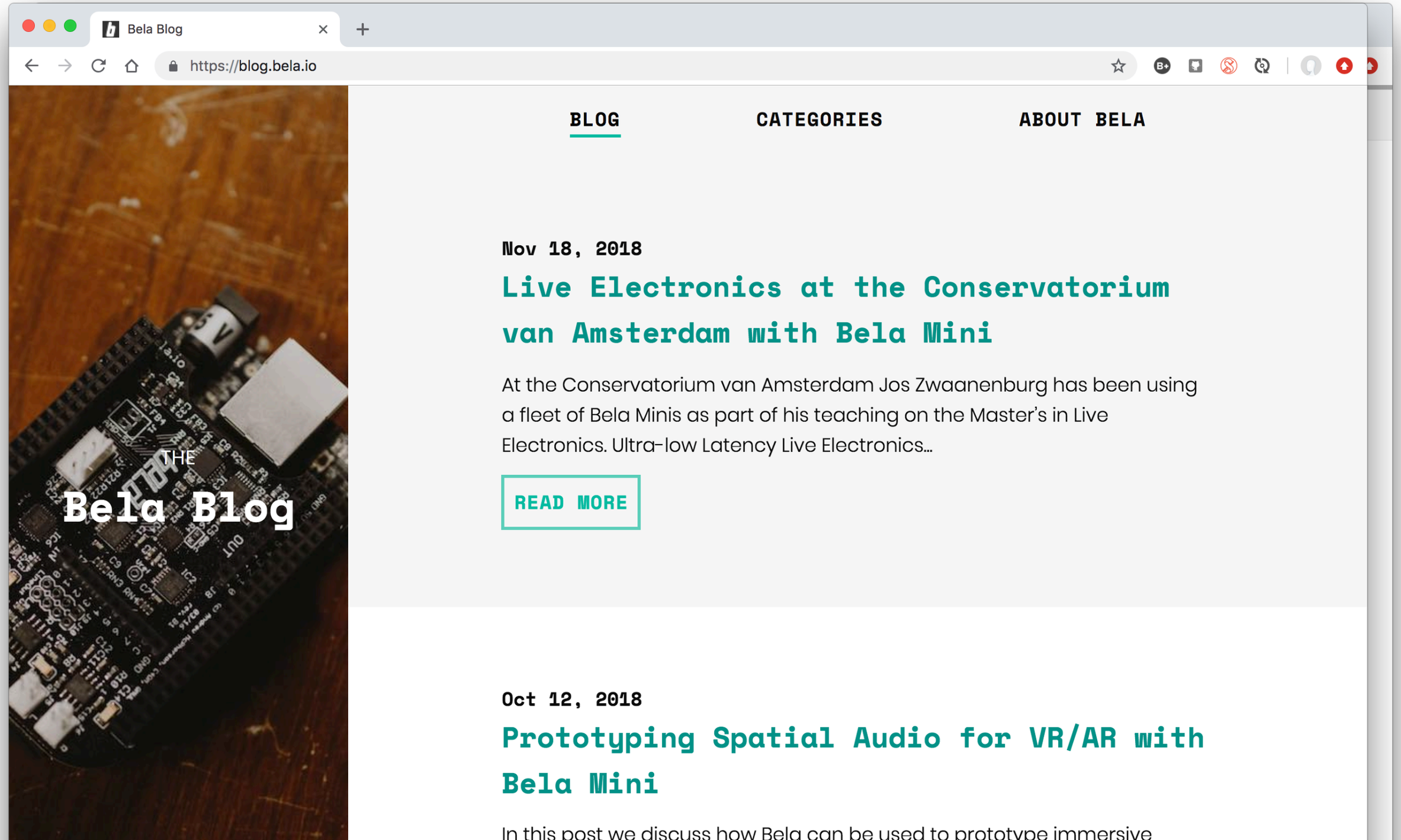
Jupiter & python



https://github.com/BelaPlatform/PAW_2019

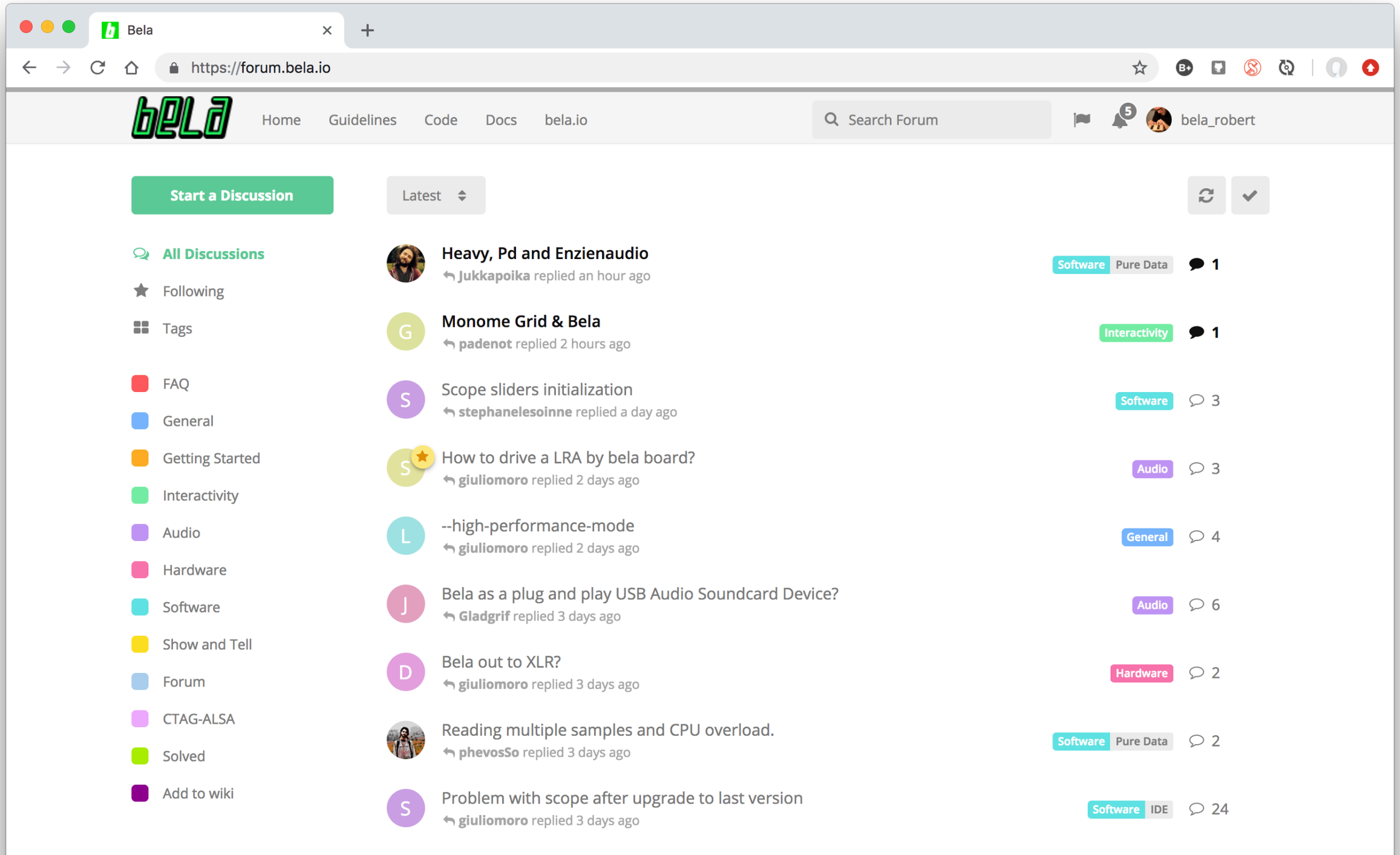
Finding out more

http://blog.bela.io



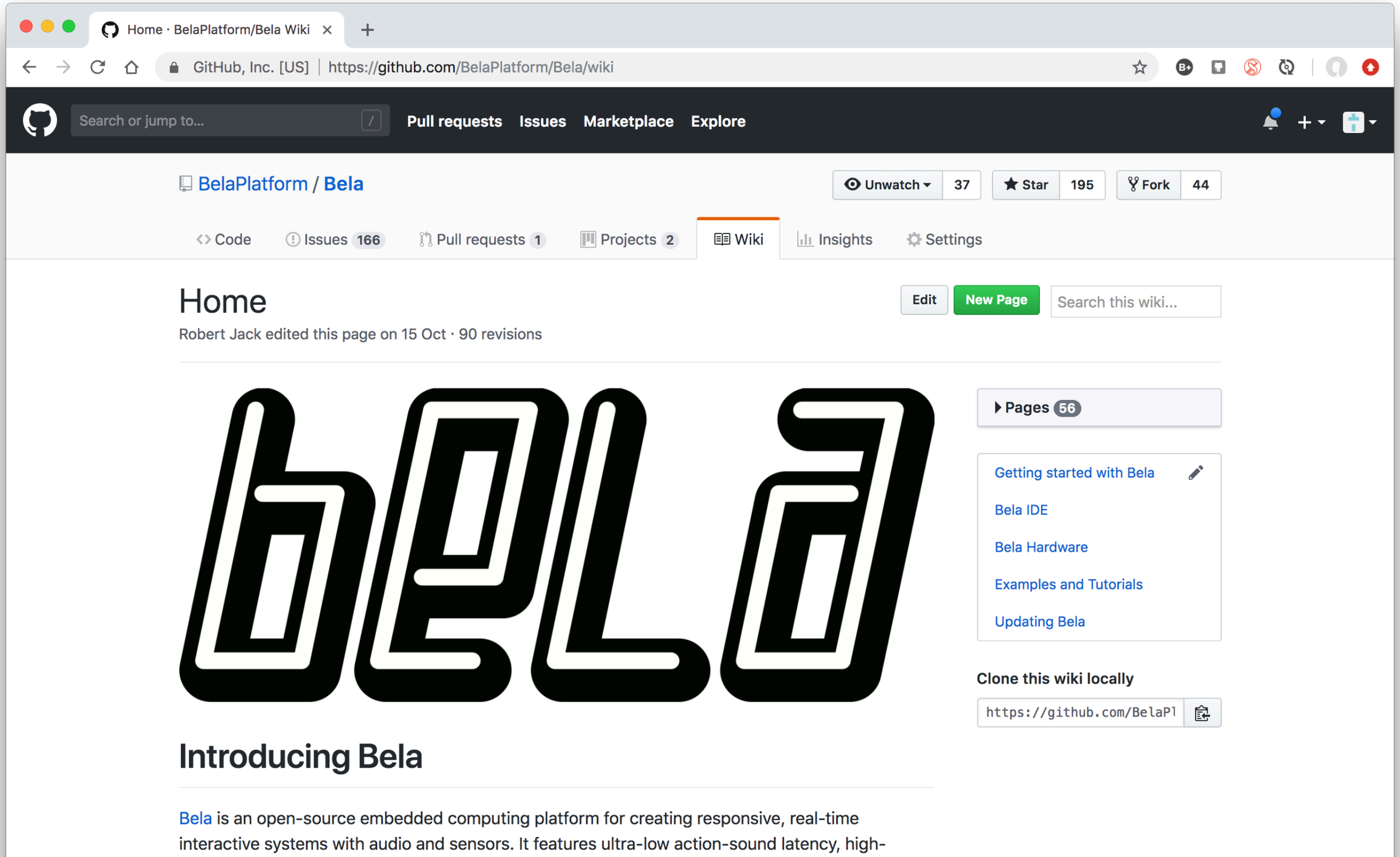
Finding out more

http://forum.bela.io



Finding out more

https://github.com/BelaPlatform/Bela/wiki



Where can I get one?

https://shop.bela.io/