

Open-source maker platform
for beautifully responsive
interactive audio



Adan L. Benito

@BelaPlatform

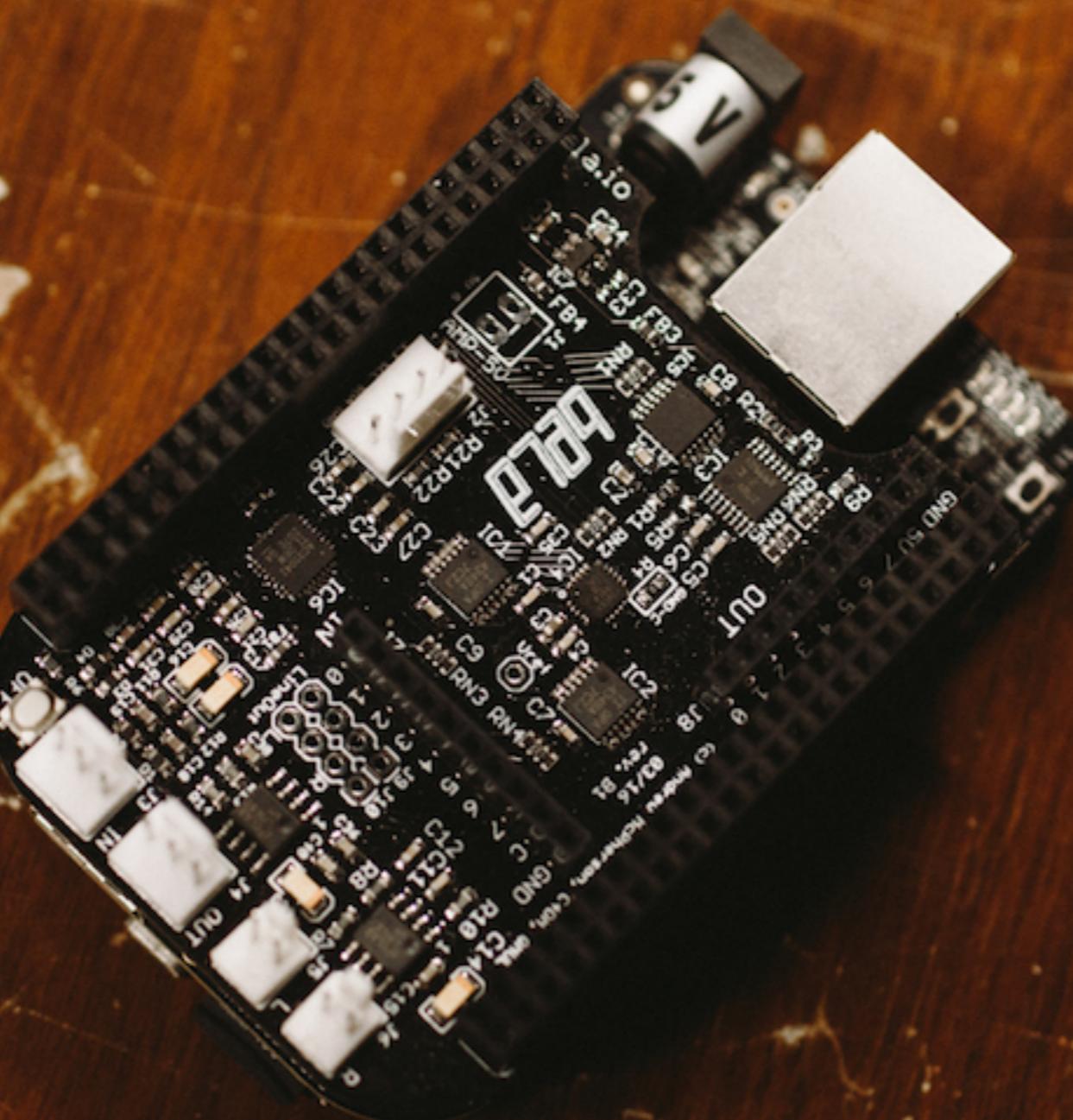


Programmable Audio Workshop

December 14th, 2019

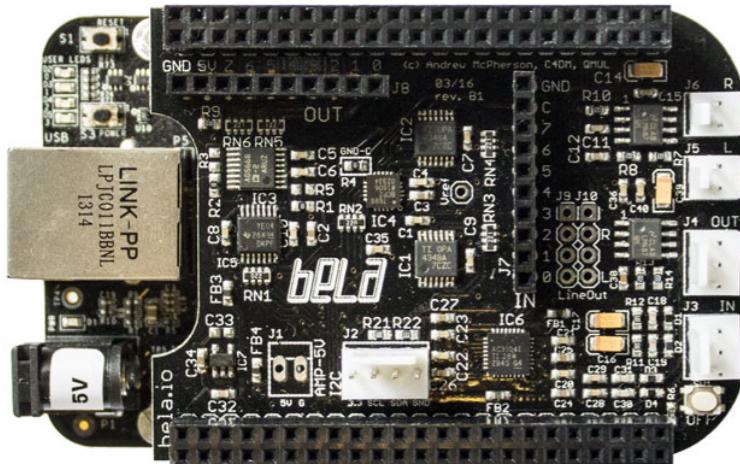
GRAME-CNCM, Lyon

What is Bela?

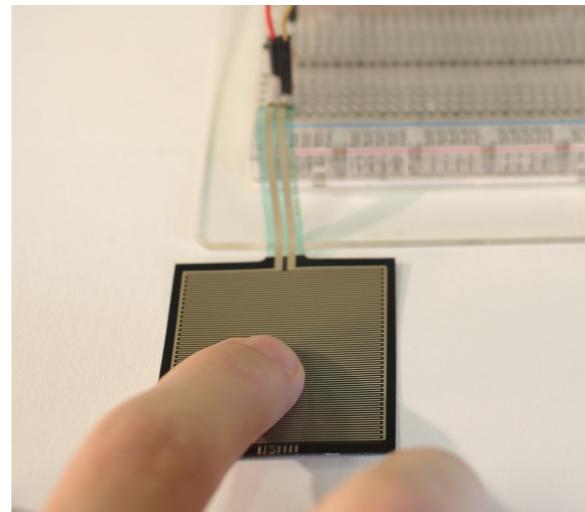


Bela

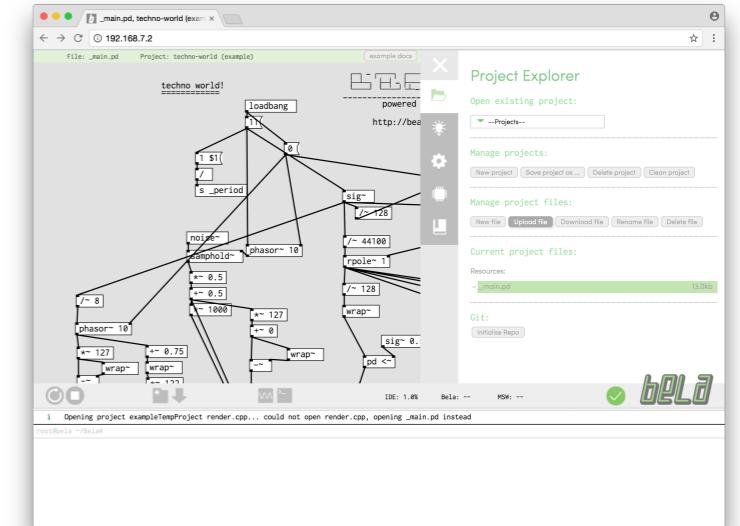
Embedded computer
designed for
interactive audio



New approach
to high-bandwidth
sensor processing

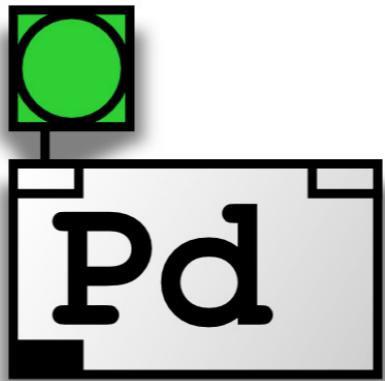


Open-source
maker
platform



- Power of a single board Linux computer
- Connectivity of a micro-controller
- Combines the benefits of both
- Analog, digital I/O sampled at audio rate
- Ultra low action-sound latency
- Jitter-free alignment between audio and sensors
- Open hardware and software
- Targeted at musicians, artists
- Online community resources: forum, wiki, blog.

Bela is a polyglot

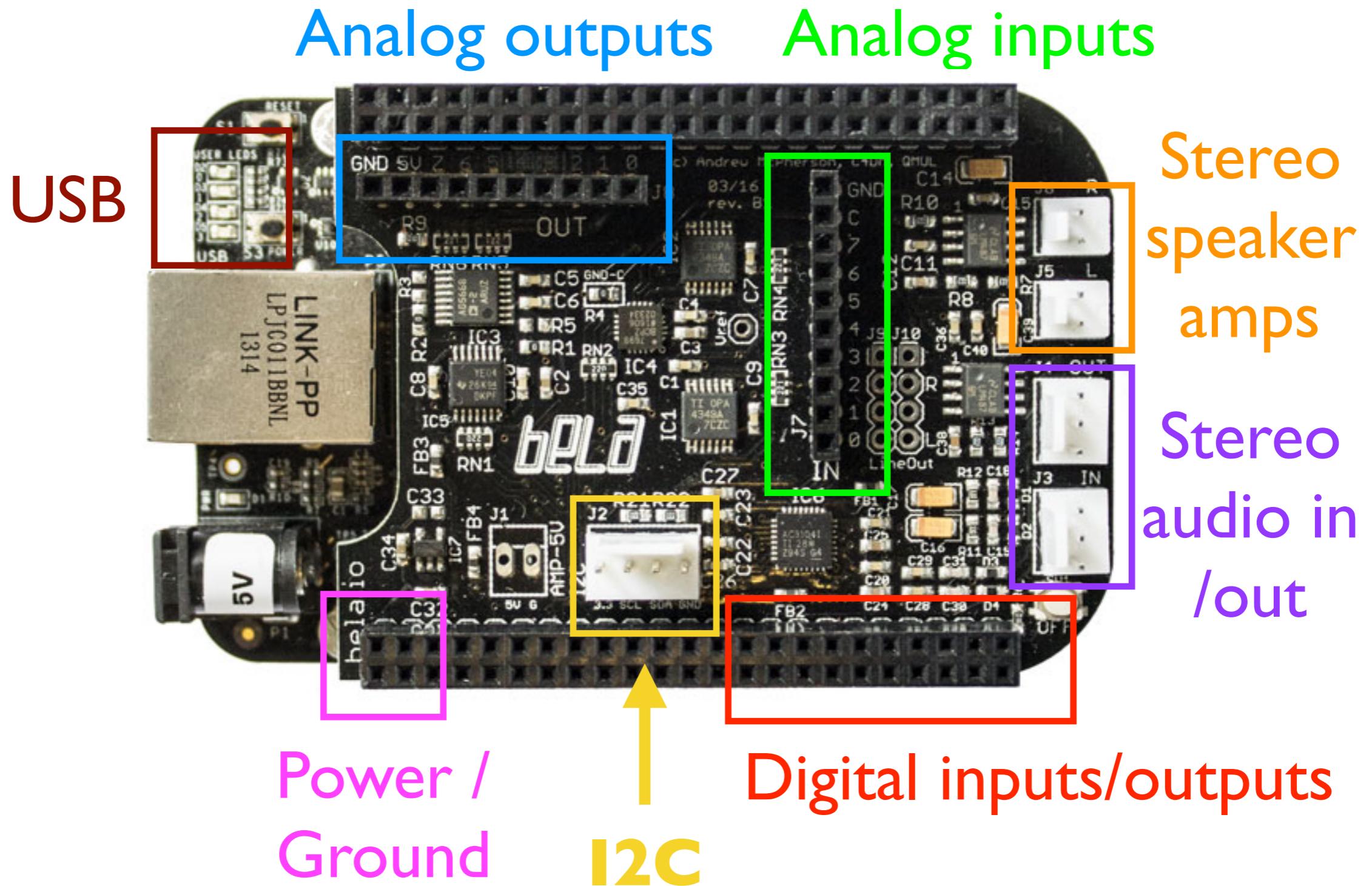


Csound

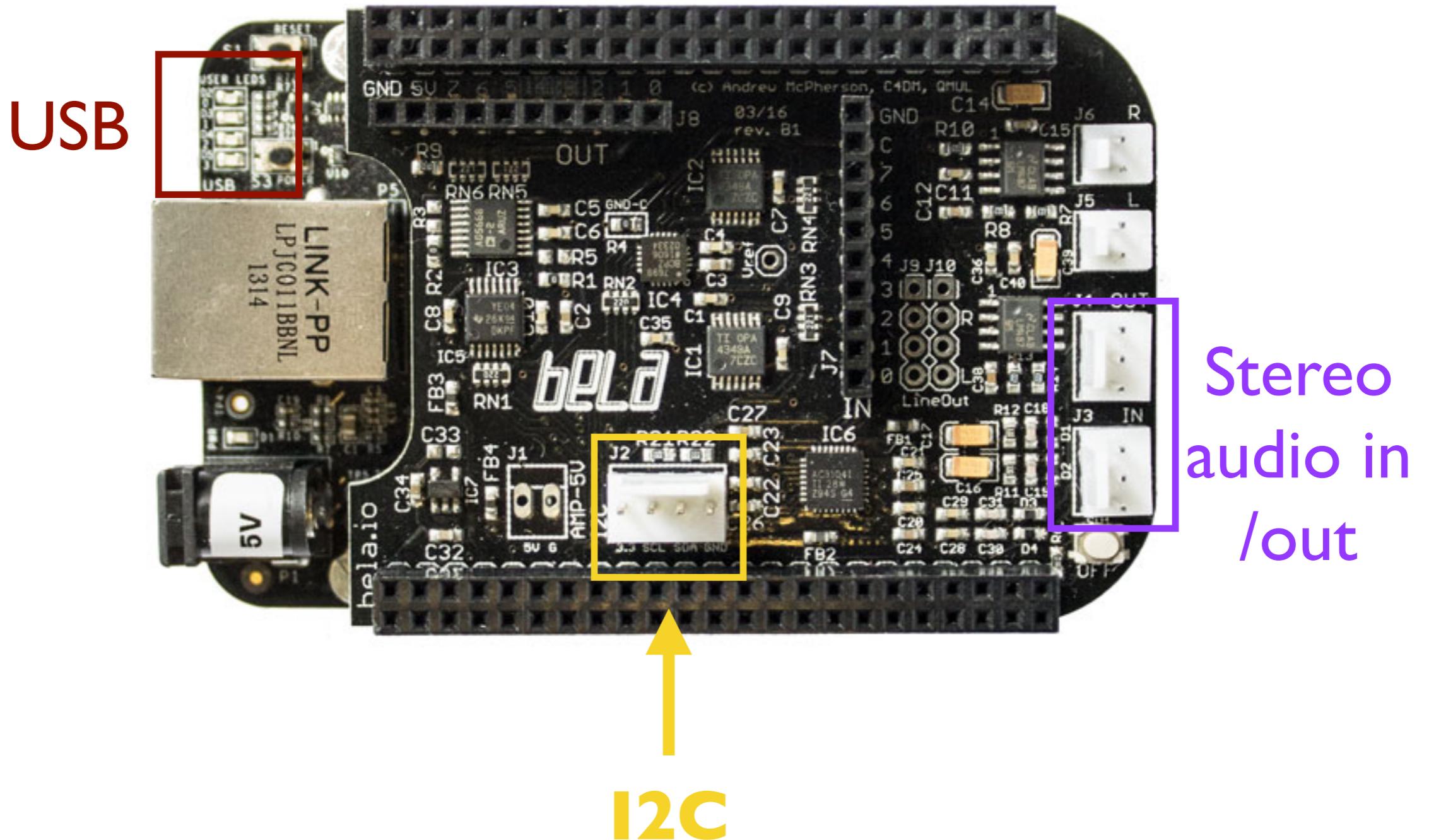


SOUL

Bela hardware



Bela hardware



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

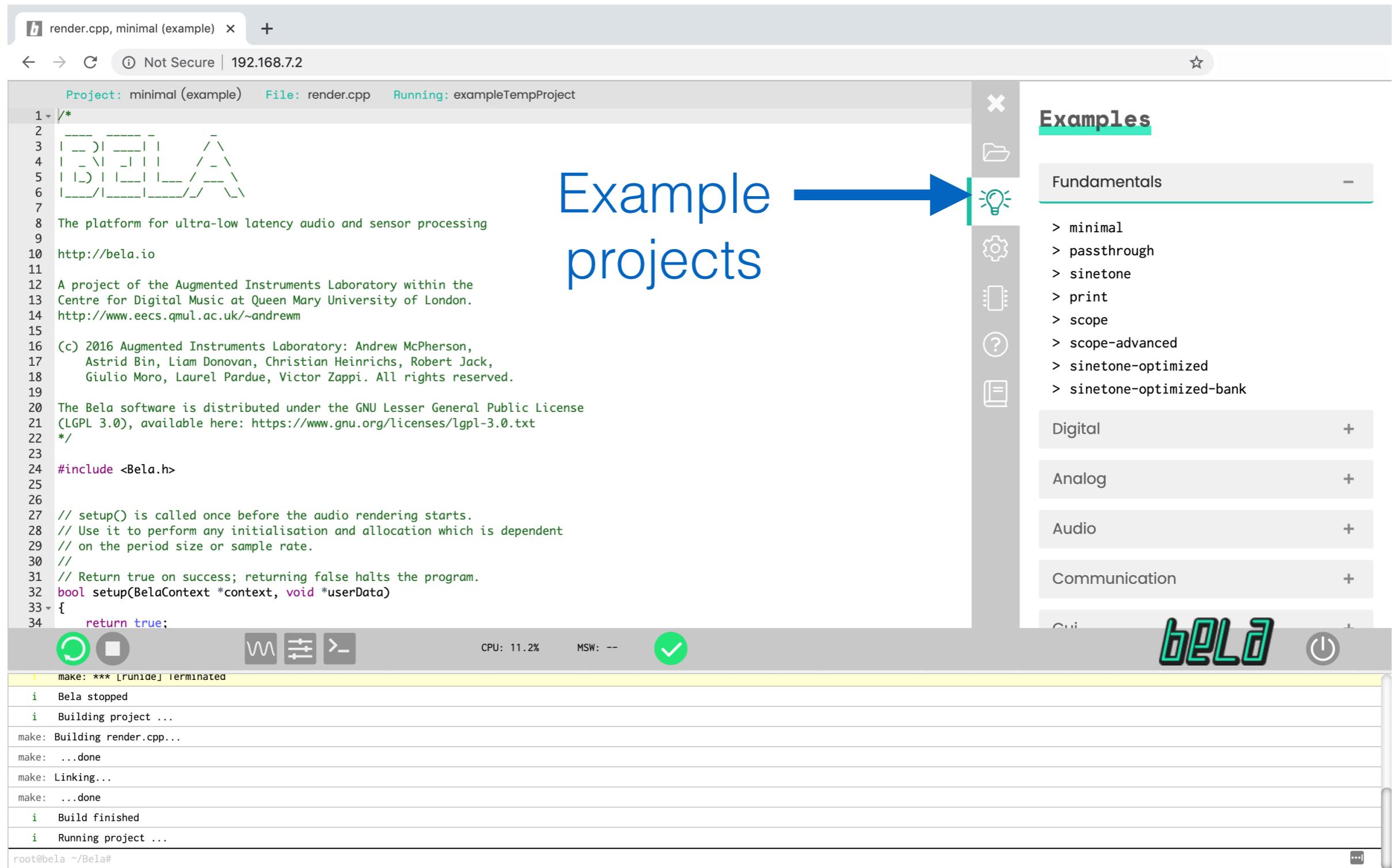
Wait for it to boot

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/>



The screenshot shows the Bela IDE interface. On the left is a code editor window titled "render.cpp, minimal (example)" showing C++ code for a Bela project. The code includes comments about the Bela platform, its license (GNU LGPL 3.0), and copyright information. It defines a `setup()` function that returns `true`. Below the code editor is a terminal window displaying build logs for "render.cpp". The logs show the project being built, linking, and running successfully. At the bottom of the IDE are various toolbars and status indicators. To the right of the code editor is a sidebar titled "Examples" which lists several pre-made projects categorized under "Fundamentals", "Digital", "Analog", "Audio", "Communication", and "GUI". A large blue arrow points from the text "Example projects" towards the "Examples" sidebar.

render.cpp, minimal (example) x +

← → ⌂ ⓘ Not Secure | 192.168.7.2

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

```
1 /*  
2  *-----|  
3  | _ \ | | | |  
4  | _ \ | | | |  
5  | | | | | | | |  
6  | | | | | | | |  
7  | | | | | | | |  
8  The platform for ultra-low latency audio and sensor processing  
9  
10 http://bela.io  
11  
12 A project of the Augmented Instruments Laboratory within the  
13 Centre for Digital Music at Queen Mary University of London.  
14 http://www.eecs.qmul.ac.uk/~andrewm  
15  
16 (c) 2016 Augmented Instruments Laboratory: Andrew McPherson,  
17 Astrid Bin, Liam Donovan, Christian Heinrichs, Robert Jack,  
18 Giulio Moro, Laurel Pardue, Victor Zappi. All rights reserved.  
19  
20 The Bela software is distributed under the GNU Lesser General Public License  
21 (LGPL 3.0), available here: https://www.gnu.org/licenses/lgpl-3.0.txt  
22 */  
23  
24 #include <Bela.h>  
25  
26  
27 // setup() is called once before the audio rendering starts.  
28 // Use it to perform any initialisation and allocation which is dependent  
29 // on the period size or sample rate.  
30 //  
31 // Return true on success; returning false halts the program.  
32 bool setup(BelaContext *context, void *userData)  
33 {  
34     return true;  
}
```

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 ...

CPU: 11.2% MSW: -- ✓

Examples

Fundamentals

- > minimal
- > passthrough
- > sinetone
- > print
- > scope
- > scope-advanced
- > sinetone-optimized
- > sinetone-optimized-bank

Digital

Analog

Audio

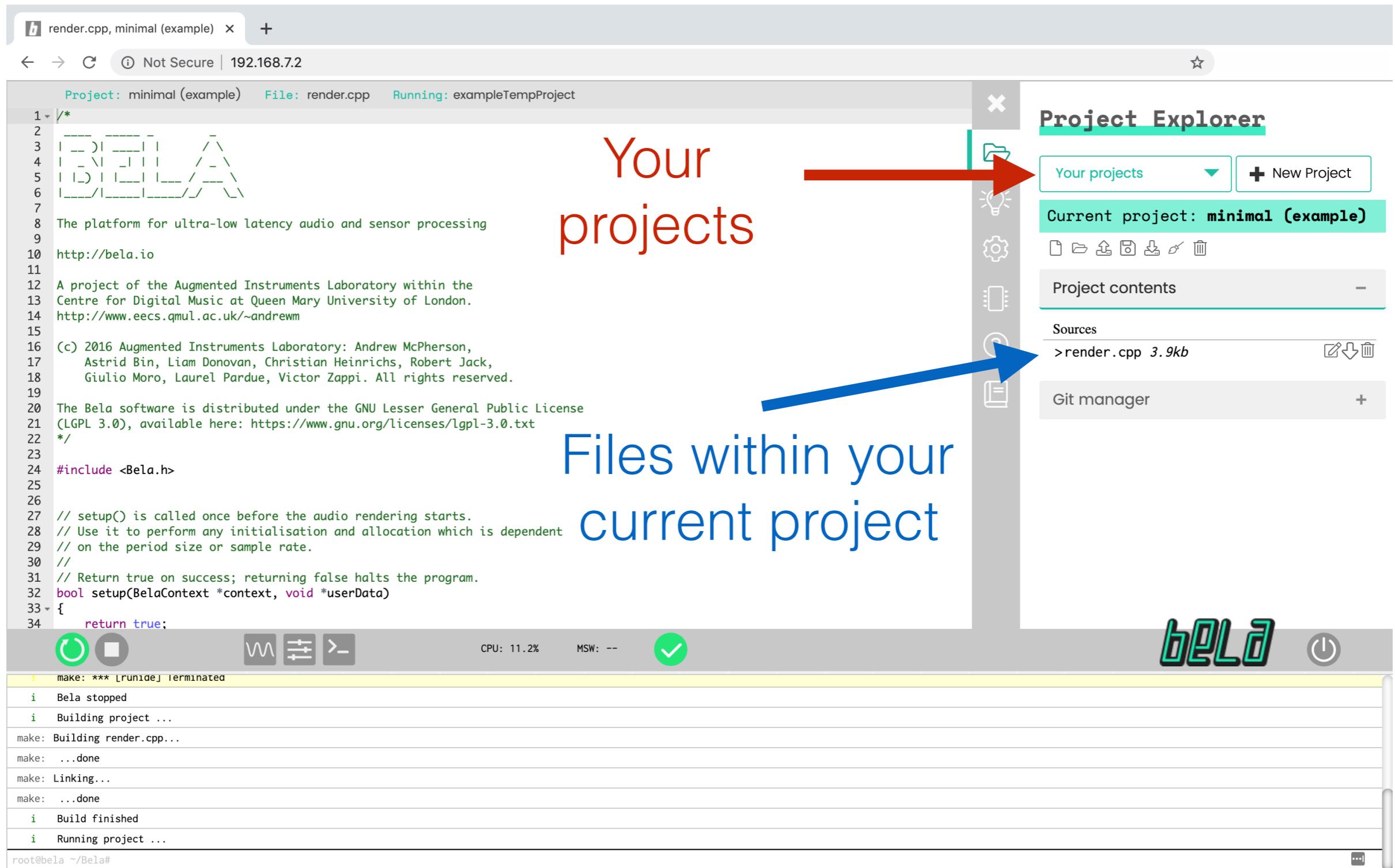
Communication

GUI

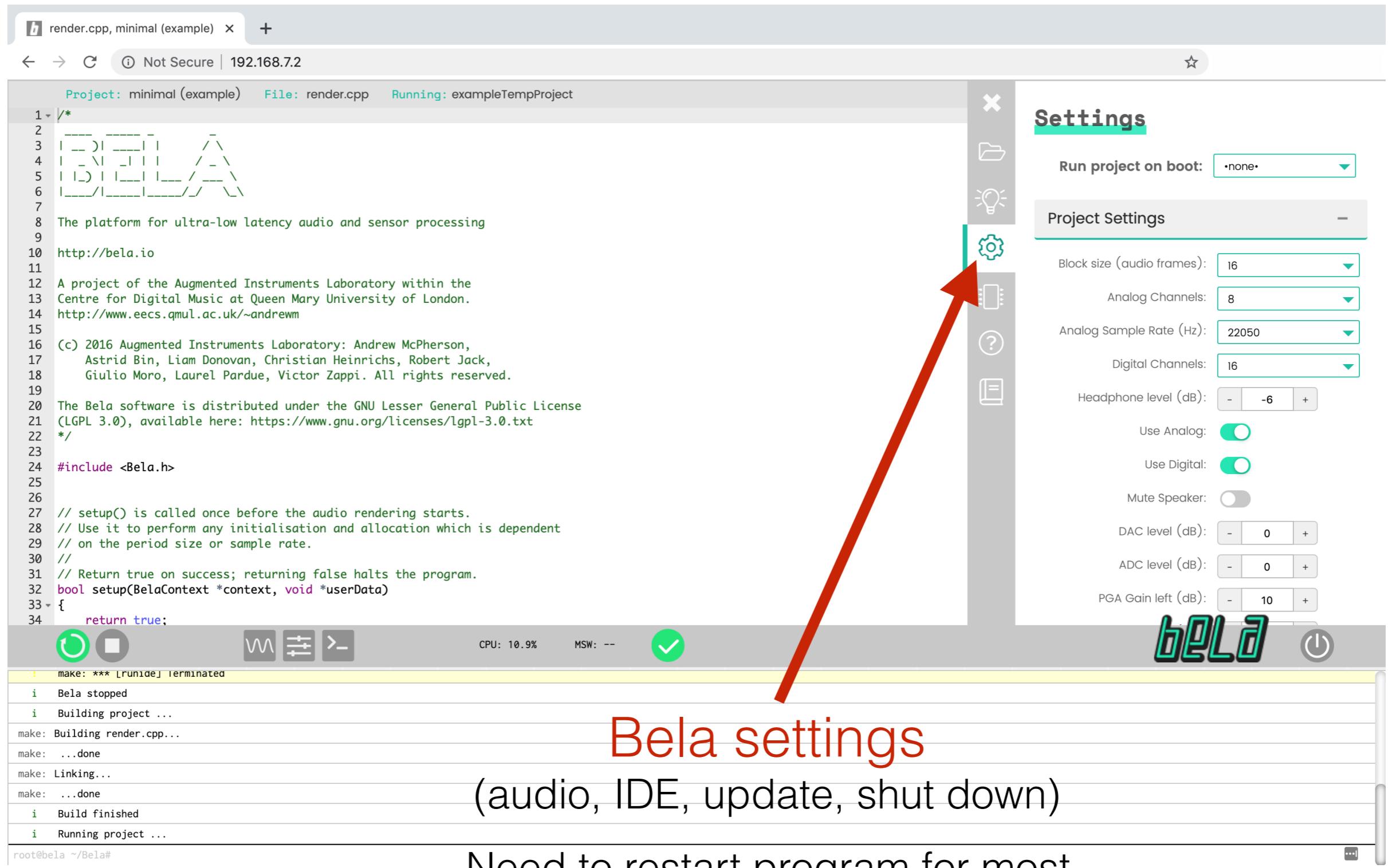
bELA

root@bela ~/Bela#

Bring up the Bela IDE:
http://bela.local/



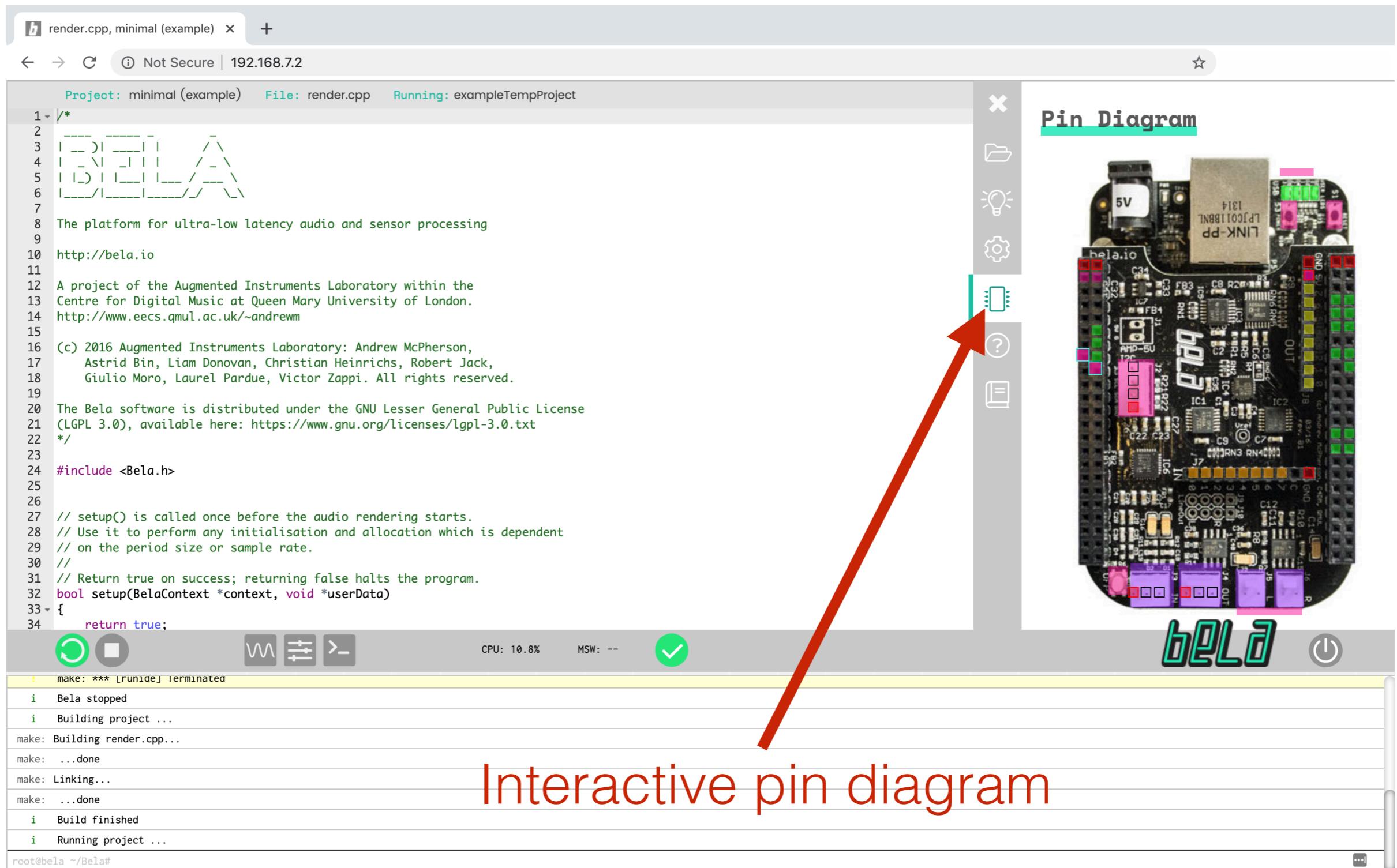
Bring up the Bela IDE: <http://bela.local/>



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/>



The screenshot shows the Bela IDE interface. On the left, a code editor displays the contents of `render.cpp`, which is an example project. The code includes comments about the Bela platform, its license (LGPL 3.0), and copyright information. On the right, there is a detailed **Pin Diagram** for the Bela hardware. A red arrow points from the text "Interactive pin diagram" at the bottom center towards the pin diagram area.

```
1  /*
2  3  _____
3  |   \  |
4  |   \  |
5  |   \  |
6  |   \  |
7  |   \  |
8  The platform for ultra-low latency audio and sensor processing
9
10 http://bela.io
11
12 A project of the Augmented Instruments Laboratory within the
13 Centre for Digital Music at Queen Mary University of London.
14 http://www.eecs.qmul.ac.uk/~andrewm
15
16 (c) 2016 Augmented Instruments Laboratory: Andrew McPherson,
17 Astrid Bin, Liam Donovan, Christian Heinrichs, Robert Jack,
18 Giulio Moro, Laurel Pardue, Victor Zappi. All rights reserved.
19
20 The Bela software is distributed under the GNU Lesser General Public License
21 (LGPL 3.0), available here: https://www.gnu.org/licenses/lgpl-3.0.txt
22 */
23
24 #include <Bela.h>
25
26
27 // setup() is called once before the audio rendering starts.
28 // Use it to perform any initialisation and allocation which is dependent
29 // on the period size or sample rate.
30 //
31 // Return true on success; returning false halts the program.
32 bool setup(BelaContext *context, void *userData)
33 {
34     return true;

```

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#

Pin Diagram

bela

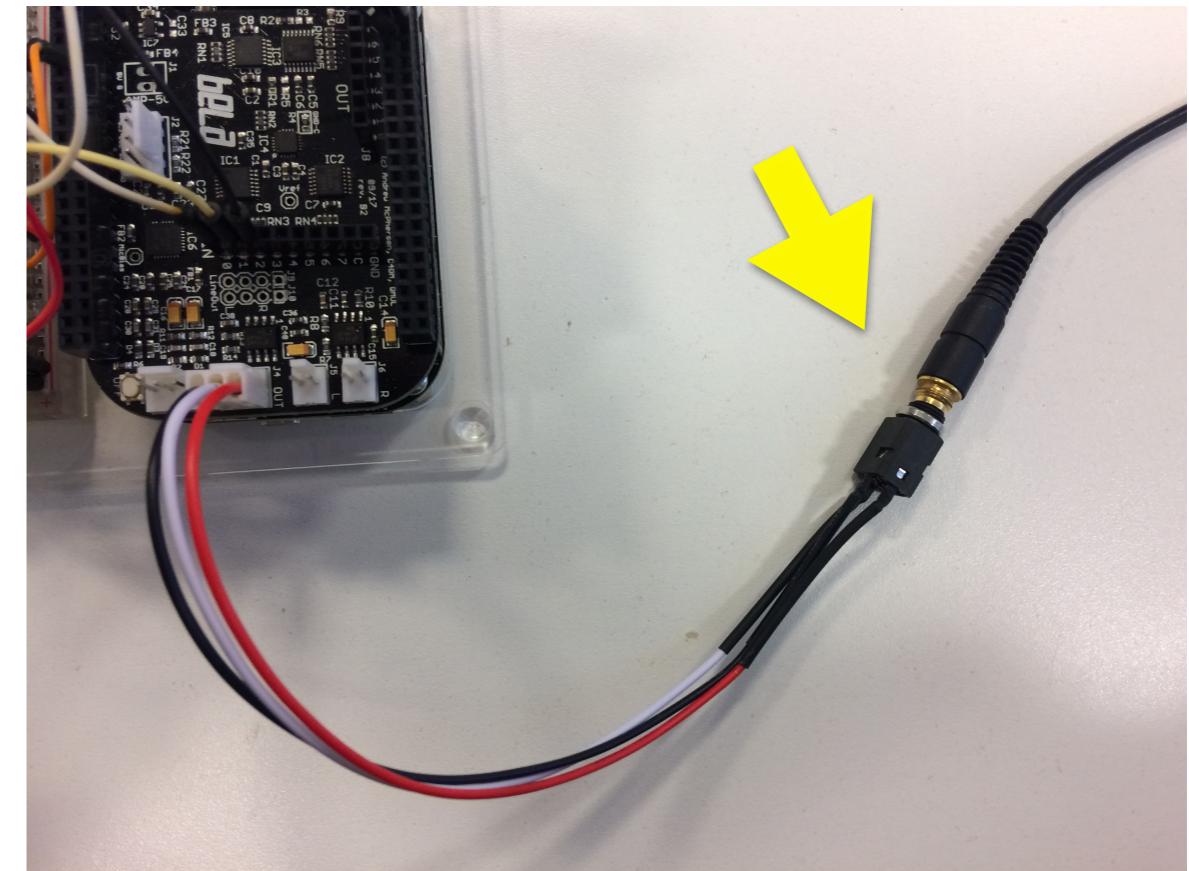
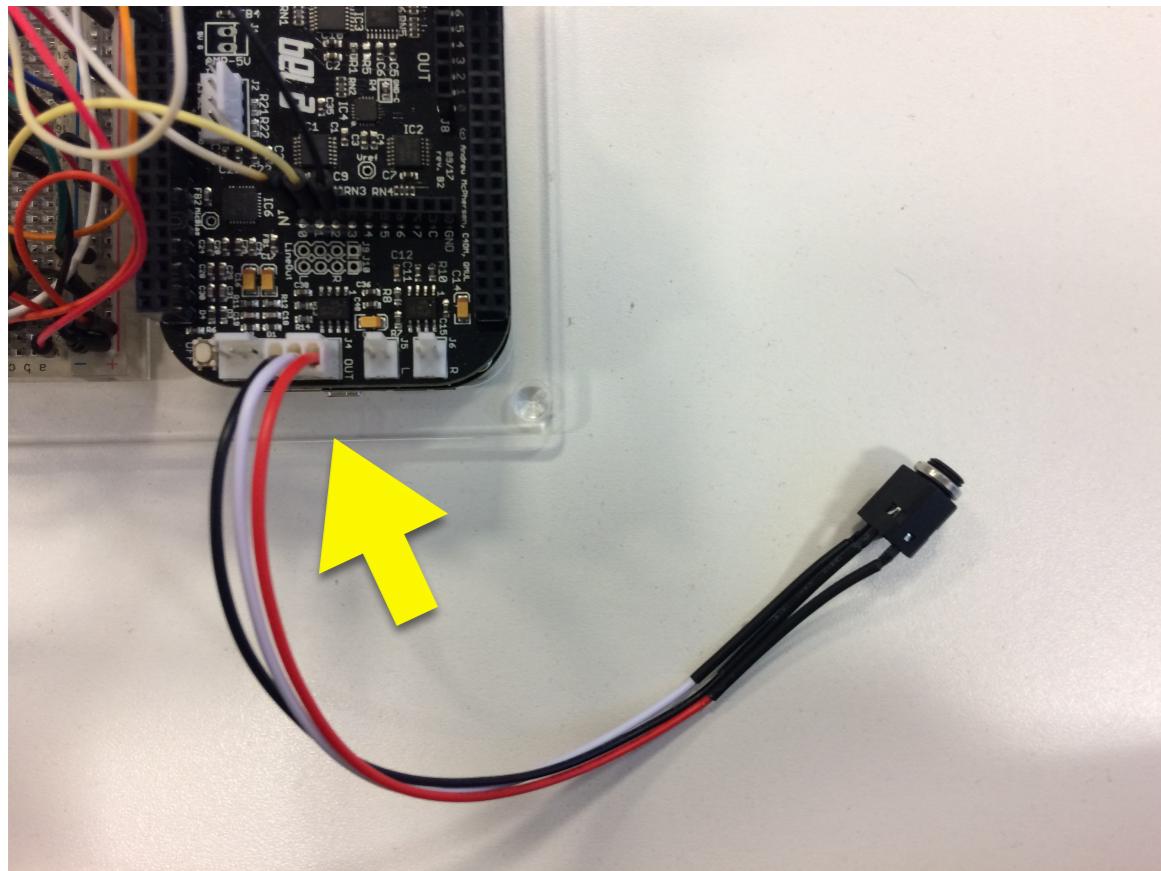
Interactive pin diagram

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



The screenshot shows the Bela IDE interface. On the left is a code editor with the file `render.cpp` open, displaying the Bela license and setup code. Below the code editor is a terminal window showing build logs. On the right is a settings panel titled "Settings" with various audio configuration options. A red arrow points from the text "PGA Gain" to the "PGA Gain left (dB)" slider in the settings panel.

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

```
1  /*
2   _____
3  |  _ \ |
4  | / \ \ |
5  |  _ \ \ |
6  | / \ \ \ |
7  |____|____|____\ \ \ \ 
8 The platform for ultra-low latency audio and sensor processing
9
10 http://bela.io
11
12 A project of the Augmented Instruments Laboratory within the
13 Centre for Digital Music at Queen Mary University of London.
14 http://www.eecs.qmul.ac.uk/~andrewm
15
16 (c) 2016 Augmented Instruments Laboratory: Andrew McPherson,
17 Astrid Bin, Liam Donovan, Christian Heinrichs, Robert Jack,
18 Giulio Moro, Laurel Pardue, Victor Zappi. All rights reserved.
19
20 The Bela software is distributed under the GNU Lesser General Public License
21 (LGPL 3.0), available here: https://www.gnu.org/licenses/lgpl-3.0.txt
22 */
23
24 #include <Bela.h>
25
26
27 // setup() is called once before the audio rendering starts.
28 // Use it to perform any initialisation and allocation which is dependent
29 // on the period size or sample rate.
30 //
31 // Return true on success; returning false halts the program.
32 bool setup(BelaContext *context, void *userData)
33 {
34     return true;

```

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#

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:

Use Digital:

Mute Speaker:

DAC level (dB): 0

ADC level (dB): 0

PGA Gain left (dB): 10

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>

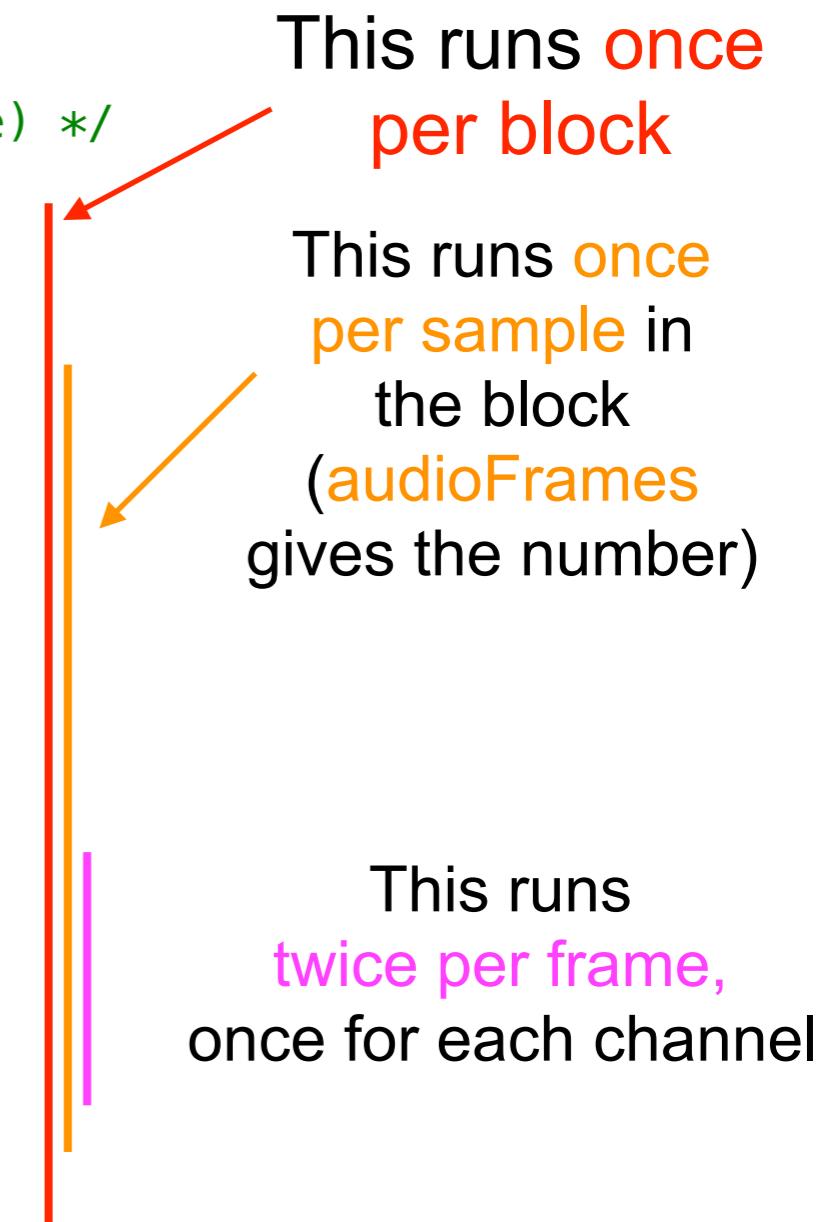
float gPhase = 0.0; /* Phase of the oscillator (global variable) */

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);
        }
    }
}
```

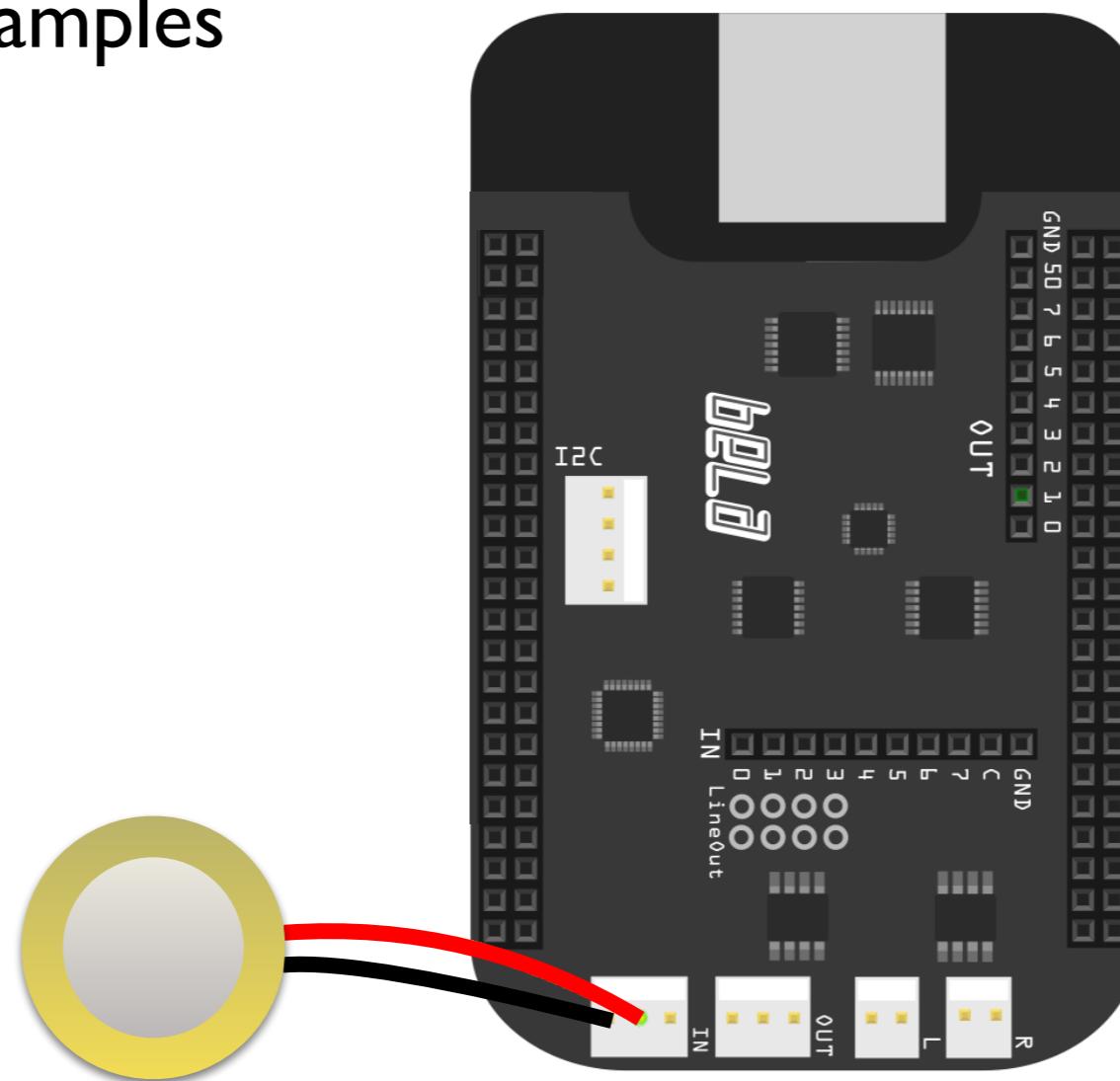
write to buffer of interleaved audio data,
specifying a **frame**, a **channel** and a **value**



Piezo Mic

Step 0

- Triggering samples



[projects/piezo-trigger-task](#)

Piezo Mic

Step 0

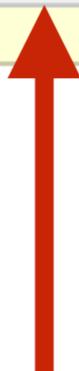
```
32 void render(BelaContext *context, void *userData)
33 {
34
```



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 I

- 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);
```

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);
```

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;
}
```

TRILL



touch sensing
for makers

bela.io/trill

DESIGNED BY BELA



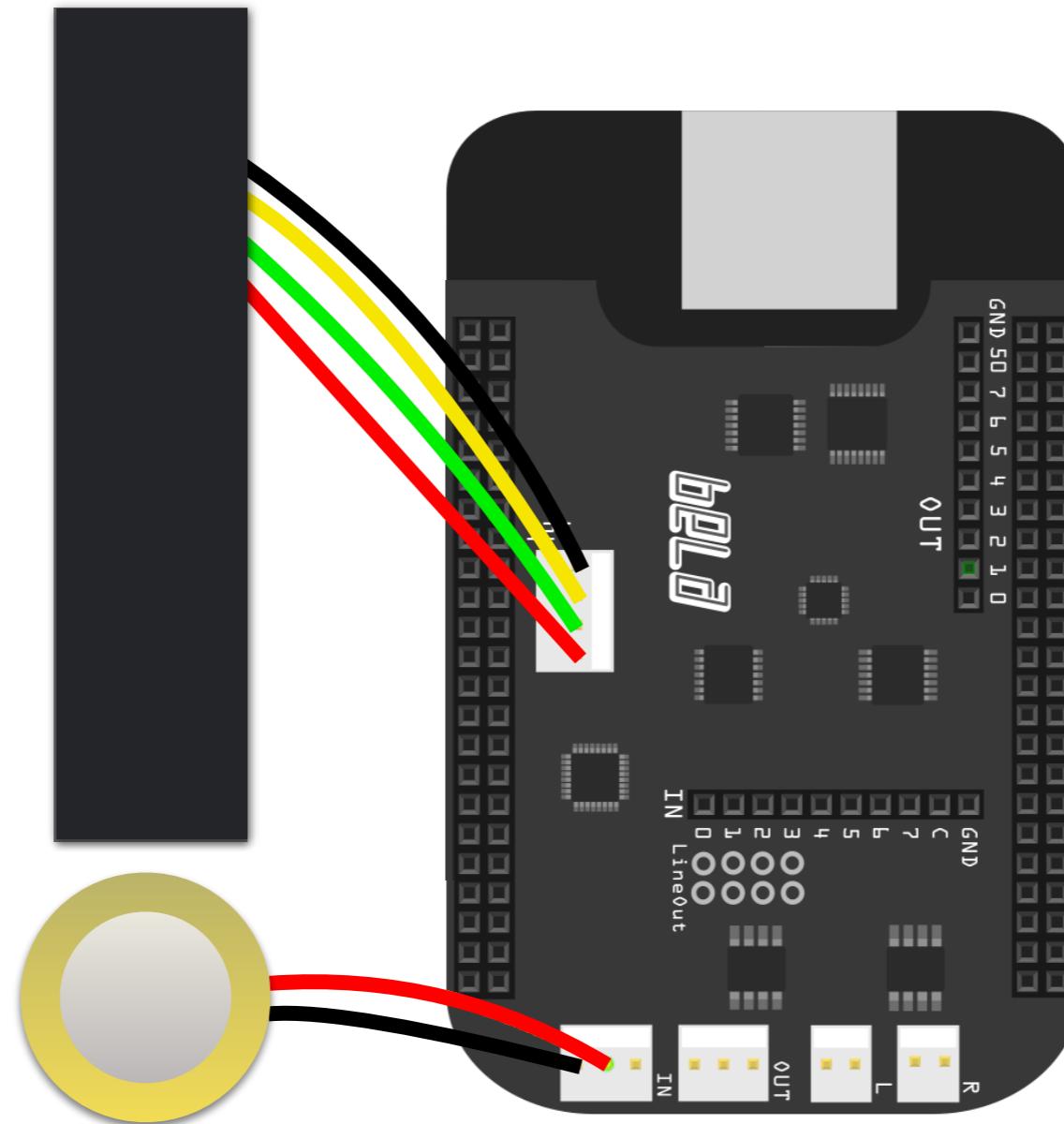
TRILL BAR



- 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 I

- 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);
```

Trill Bar

Step I

- 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);
```



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
    }
}
```

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

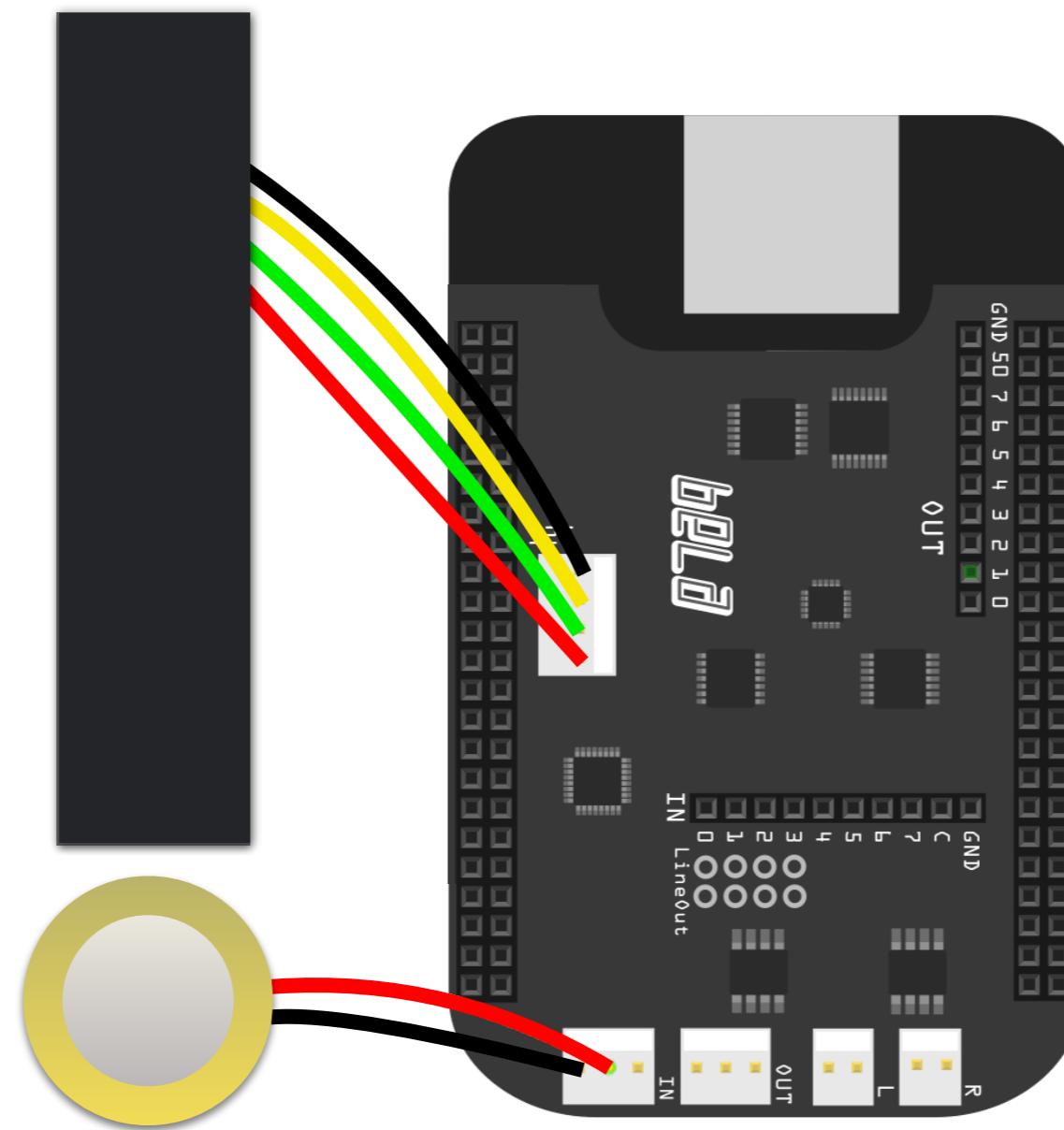


GUI

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
```

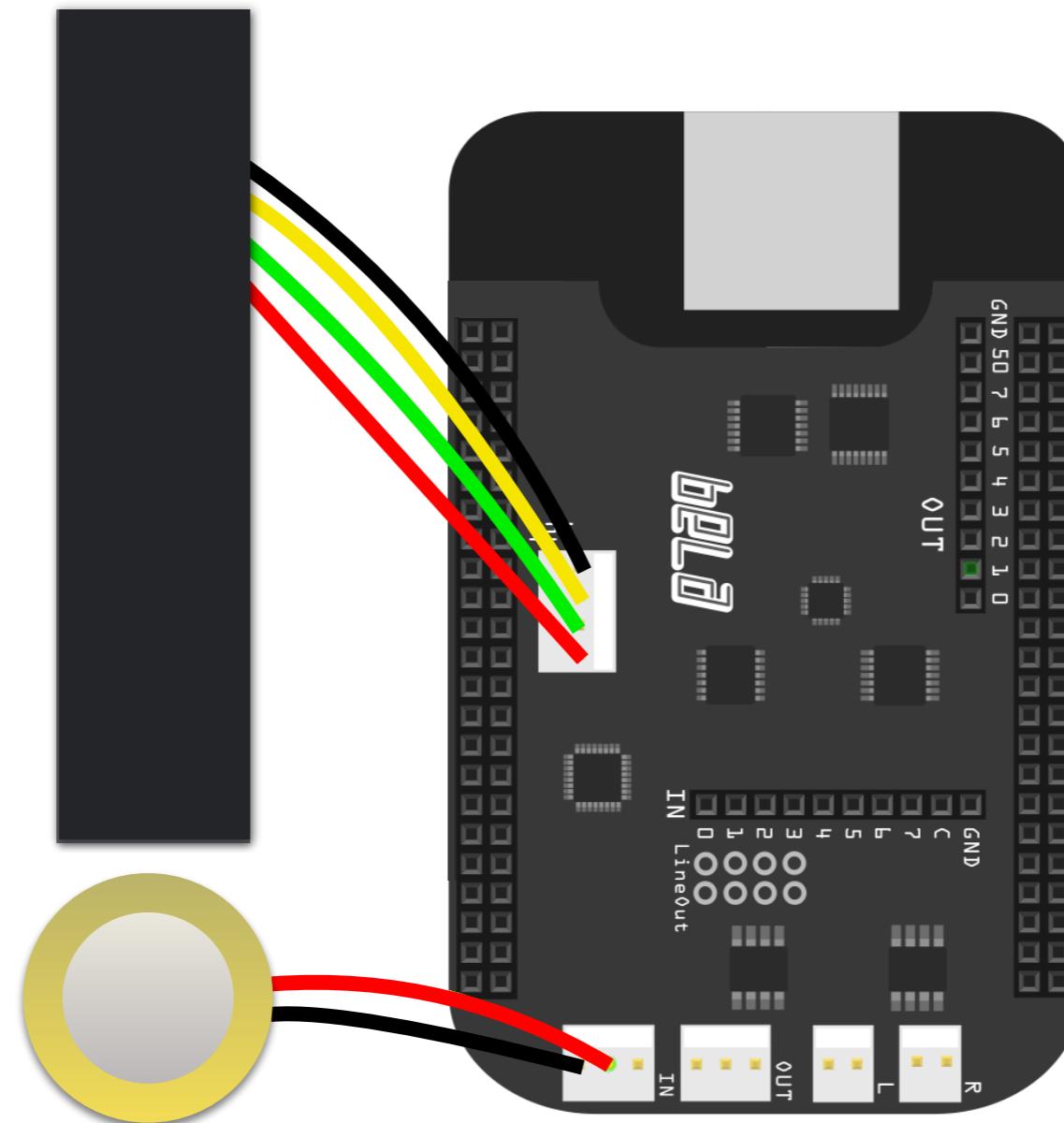
```
void setup(...) {  
    resonator.setup(options, context->audioSampleRate, context->audioFrames);  
    resonator.setParameters(432, 0.5, 0.05); // freq, gain, decay
```

```
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;
```

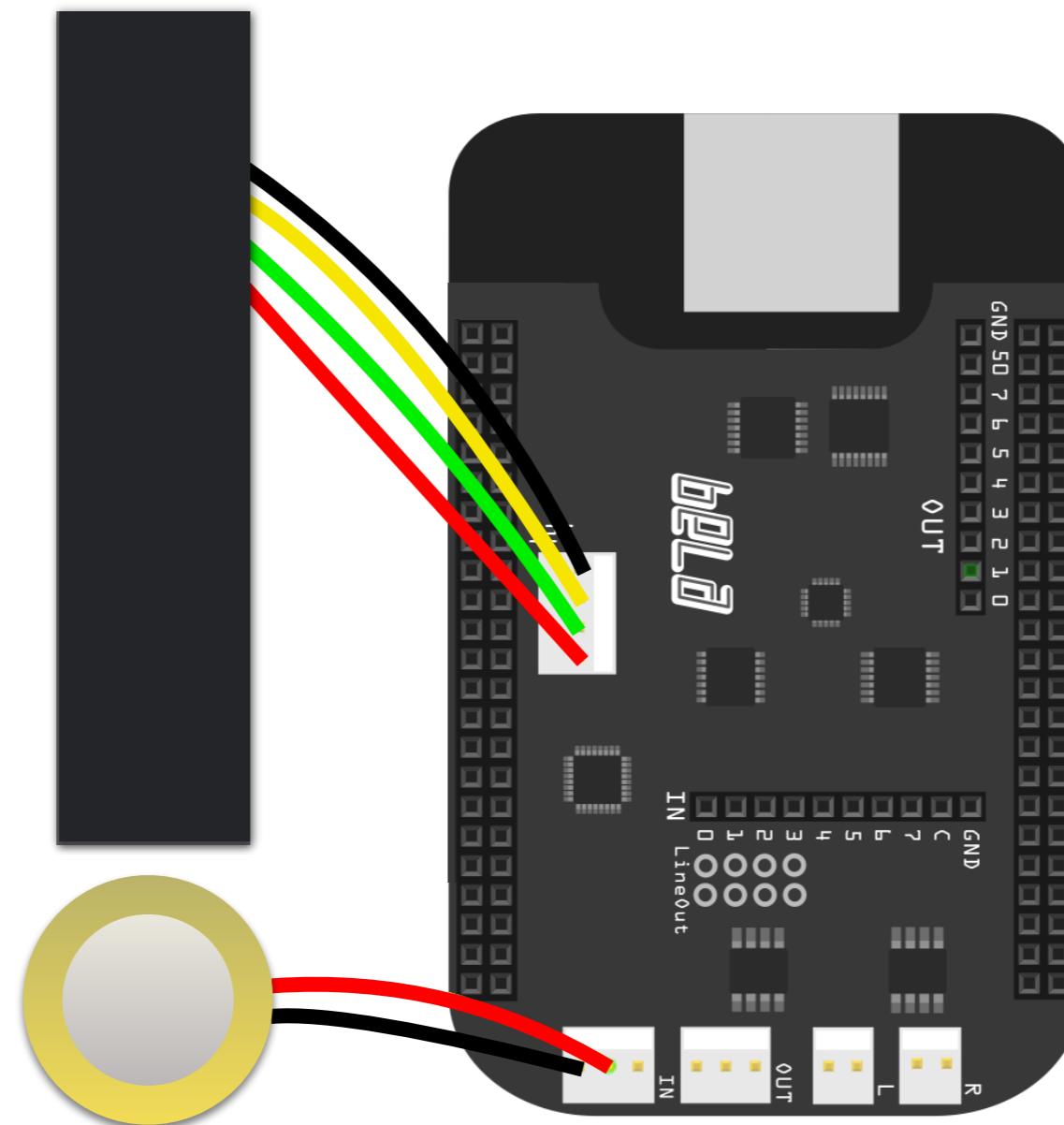
```
void setup(...) {  
    lowpass.setup(gLpCutOff, context->audioSampleRate);
```

```
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));
```

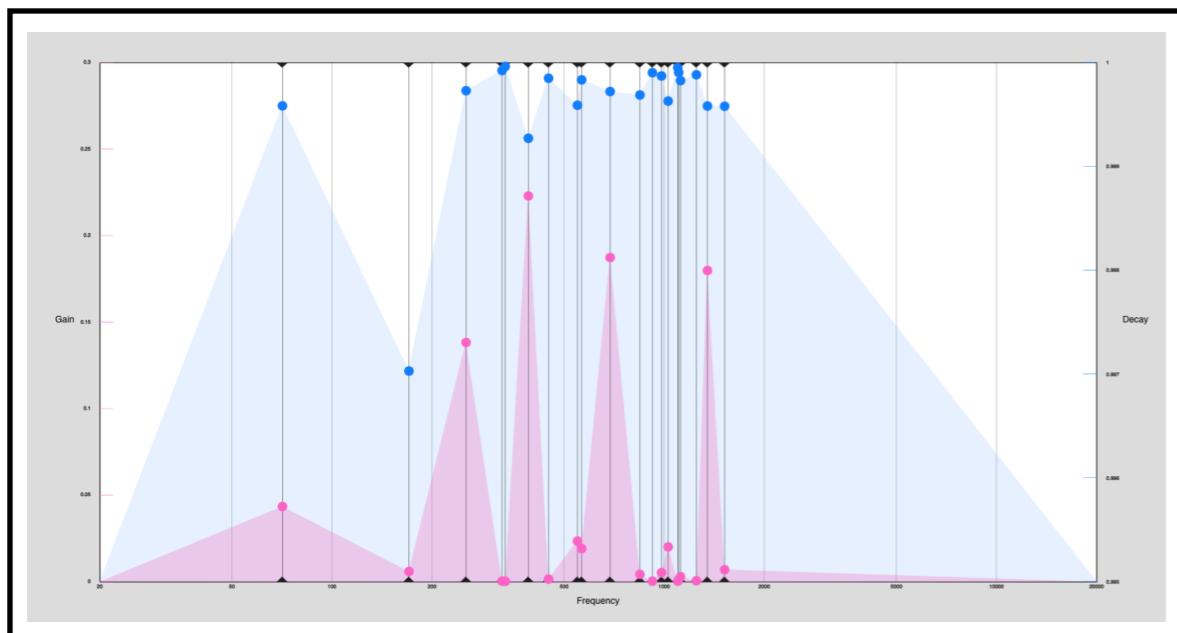
projects/trill-res-bank

Resonators

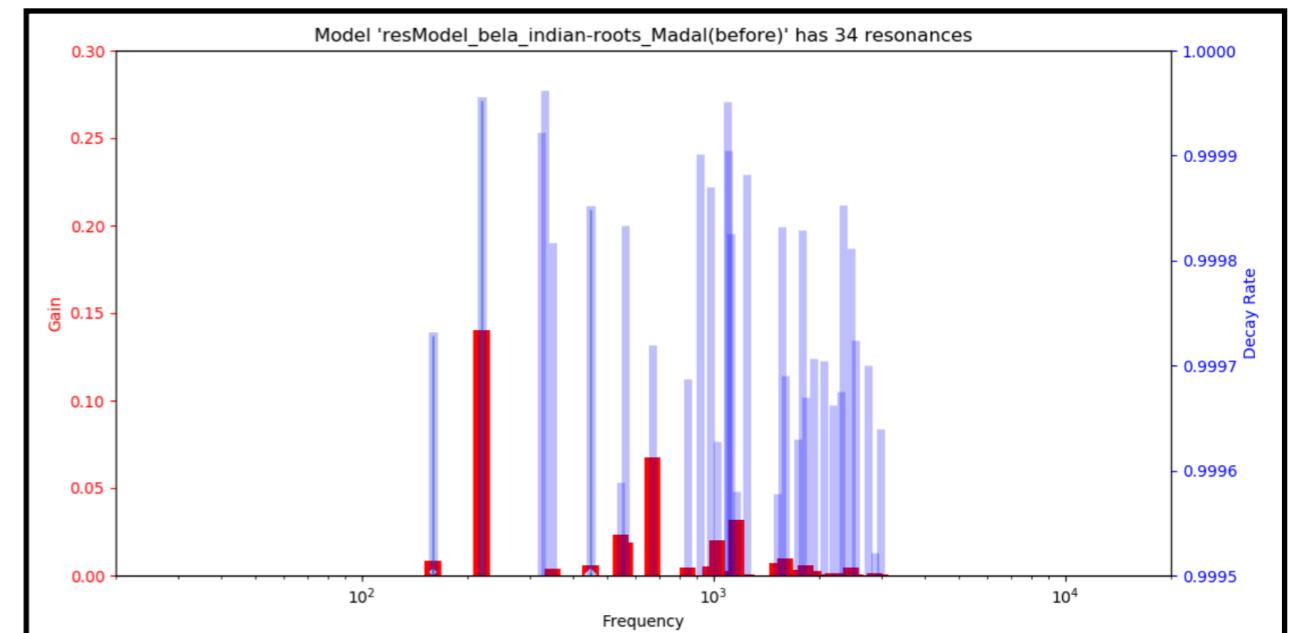
<https://github.com/jarmitage/resonators>

- 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>

BLOG

CATEGORIES

ABOUT BELA

Nov 18, 2018

Live Electronics at the Conservatorium van Amsterdam with Bela Mini

At the Conservatorium van Amsterdam Jos Zwaanenburg has been using a fleet of Bela Minis as part of his teaching on the Master's in Live Electronics. Ultra-low Latency Live Electronics...

READ MORE

Oct 12, 2018

Prototyping Spatial Audio for VR/AR with Bela Mini

In this post we discuss how Bela can be used to prototype immersive

Finding out more

<http://forum.bela.io>

The screenshot shows the Bela forum homepage in a web browser. The URL in the address bar is <https://forum.bela.io>. The page features a navigation bar with links for Home, Guidelines, Code, Docs, and bela.io. On the right side of the nav bar are icons for a search bar, notifications (5), and user profile (bela_robert). Below the nav bar, there's a green button labeled "Start a Discussion" and a dropdown menu set to "Latest". To the right of the dropdown are refresh and checkmark icons.

All Discussions

Following

Tags

FAQ

General

Getting Started

Interactivity

Audio

Hardware

Software

Show and Tell

Forum

CTAG-ALSA

Solved

Add to wiki

Heavy, Pd and Enzienaudio
↳ Jukkapoika replied an hour ago
Software Pure Data 1

Monome Grid & Bela
↳ padenot replied 2 hours ago
Interactivity 1

Scope sliders initialization
↳ stephanelesoinne replied a day ago
Software 3

How to drive a LRA by bela board?
↳ giuliomoro replied 2 days ago
Audio 3

--high-performance-mode
↳ giuliomoro replied 2 days ago
General 4

Bela as a plug and play USB Audio Soundcard Device?
↳ Gladgrif replied 3 days ago
Audio 6

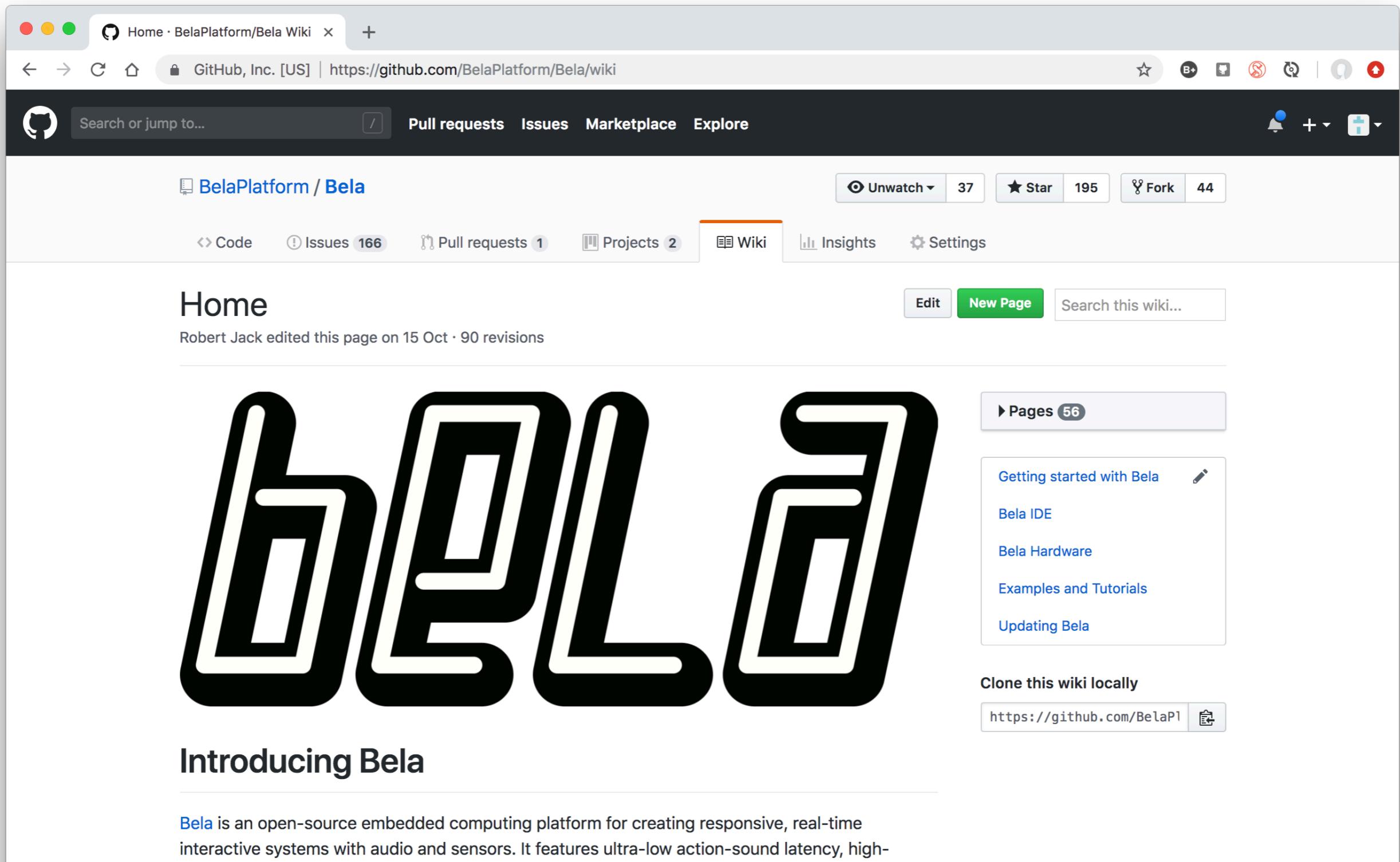
Bela out to XLR?
↳ giuliomoro replied 3 days ago
Hardware 2

Reading multiple samples and CPU overload.
↳ phevoso replied 3 days ago
Software Pure Data 2

Problem with scope after upgrade to last version
↳ giuliomoro replied 3 days ago
Software IDE 24

Finding out more

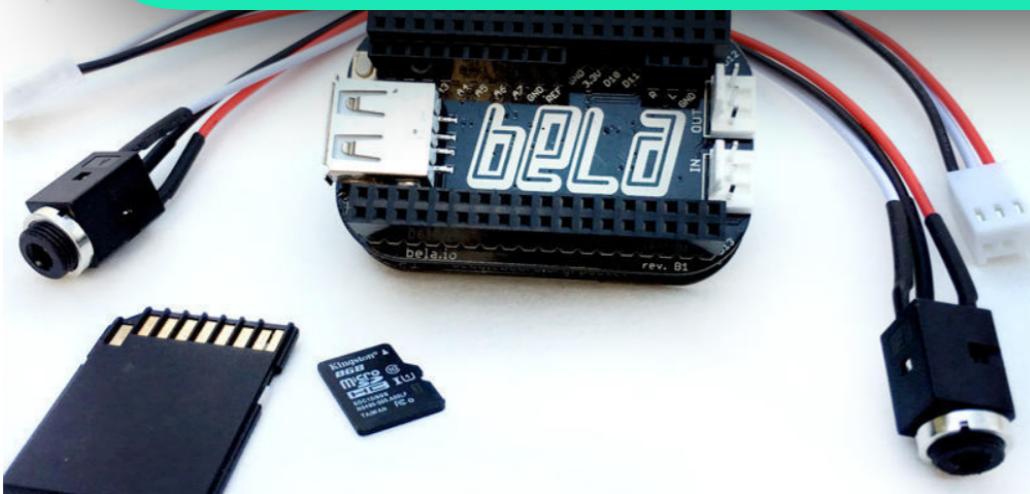
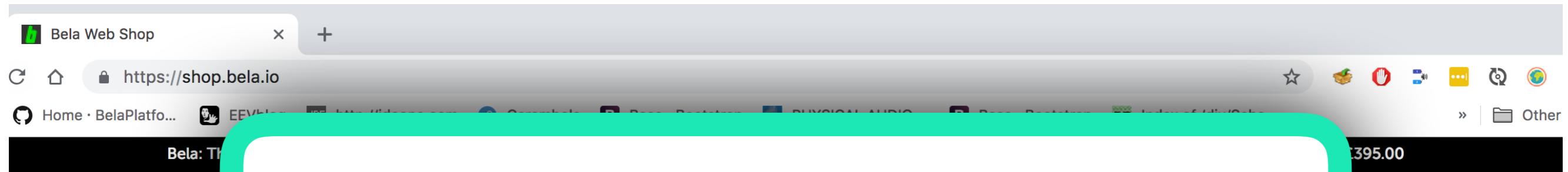
<https://github.com/BelaPlatform/Bela/wiki>



The screenshot shows the GitHub Wiki page for the Bela repository. The page title is "Home · BelaPlatform/Bela Wiki". The header includes a search bar, navigation links for Pull requests, Issues, Marketplace, and Explore, and a notification bell. Below the header, there are buttons for Unwatch (37), Star (195), Fork (44), and a "Wiki" tab which is currently selected. Other tabs include Code, Issues (166), Pull requests (1), Projects (2), Insights, and Settings. The main content area features a large, stylized Bela logo composed of thick black outlines. To the right of the logo is a sidebar titled "Pages 56" containing links to "Getting started with Bela", "Bela IDE", "Bela Hardware", "Examples and Tutorials", and "Updating Bela". At the bottom, there is a link to "Clone this wiki locally" with the URL "https://github.com/BelaPl".

Where can I get one?

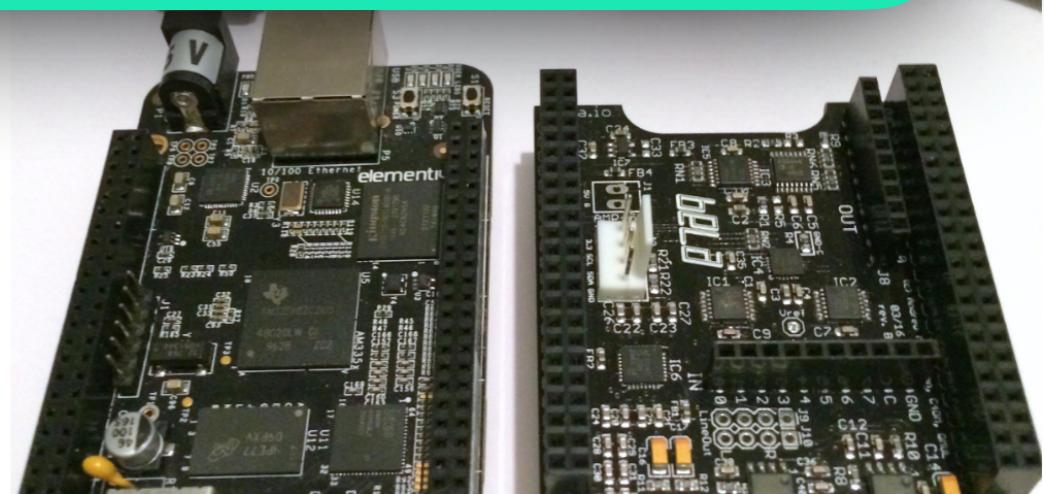
<https://shop.bela.io/>



BELA MINI CAPE AND KITS

The new, smaller Bela Mini cape, based on the PocketBeagle. It has stereo audio input and output, 8 analog inputs, 16 digital I/O.

[SHOP NOW ▶](#)



BELA CAPE AND KITS

The fully-featured Bela cape and kits, based on the BeagleBone Black. It has stereo audio input and output, 8 analog inputs and outputs, 16 digital I/O, two power speaker amplifiers.

[SHOP NOW ▶](#)