

Open-source maker platform  
for beautifully responsive  
interactive audio



Adan L. Benito

@BelaPlatform



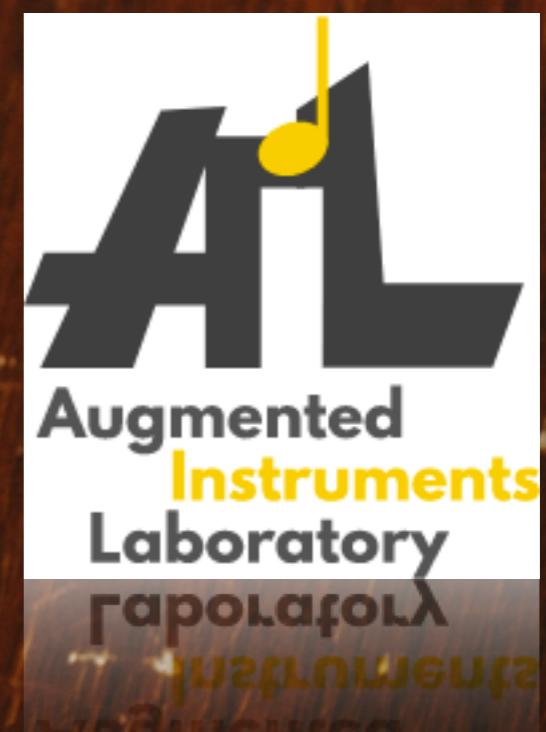
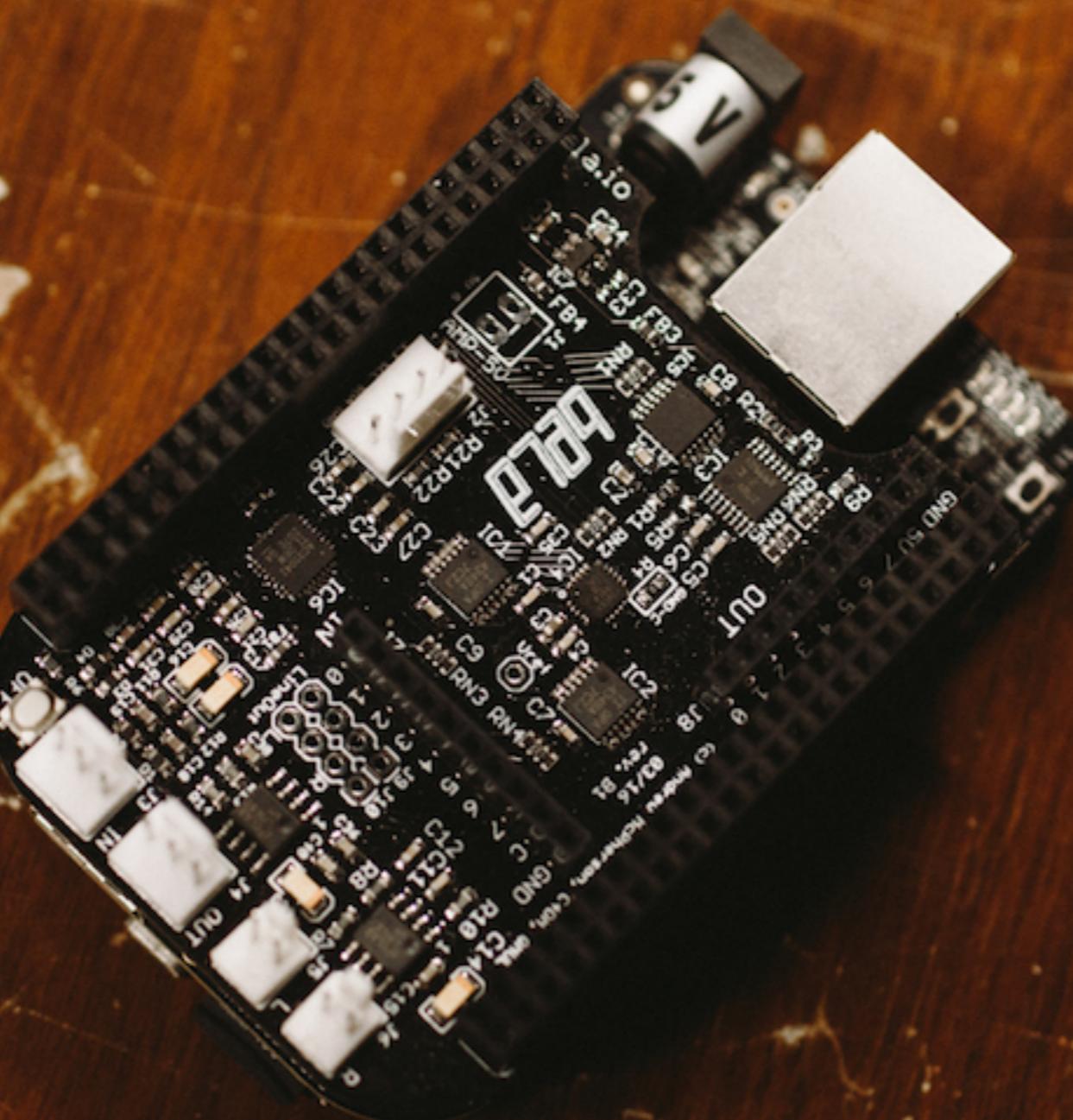
# HAID

11th International Workshop on Haptic & Audio Interaction Design

25/08/2022

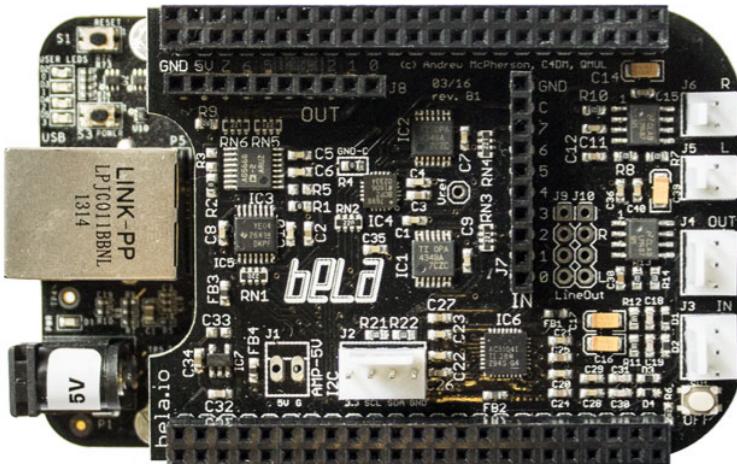
Queen Mary University of London

# What is Bela?

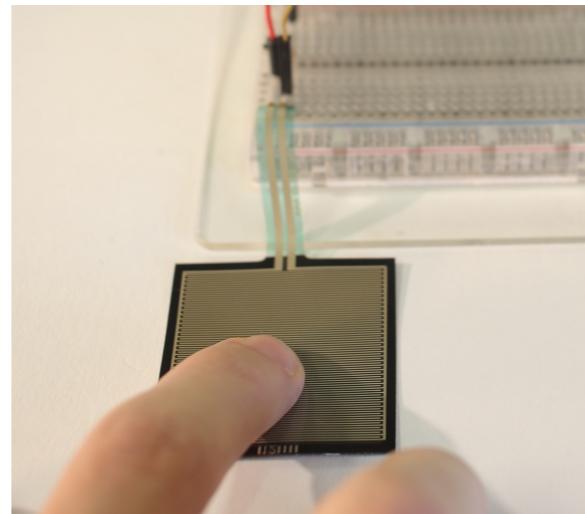


# Bela

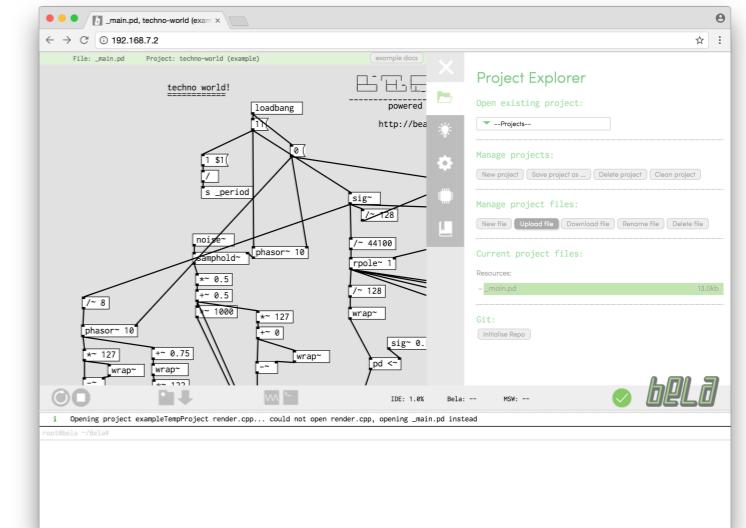
Embedded computer  
designed for  
interactive audio



New approach  
to high-bandwidth  
sensor processing



Open-source  
maker  
platform

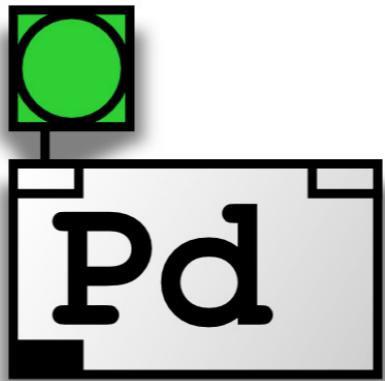


- Power of a single board Linux computer
- Connectivity of a micro-controller
- Combines the benefits of both

- 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

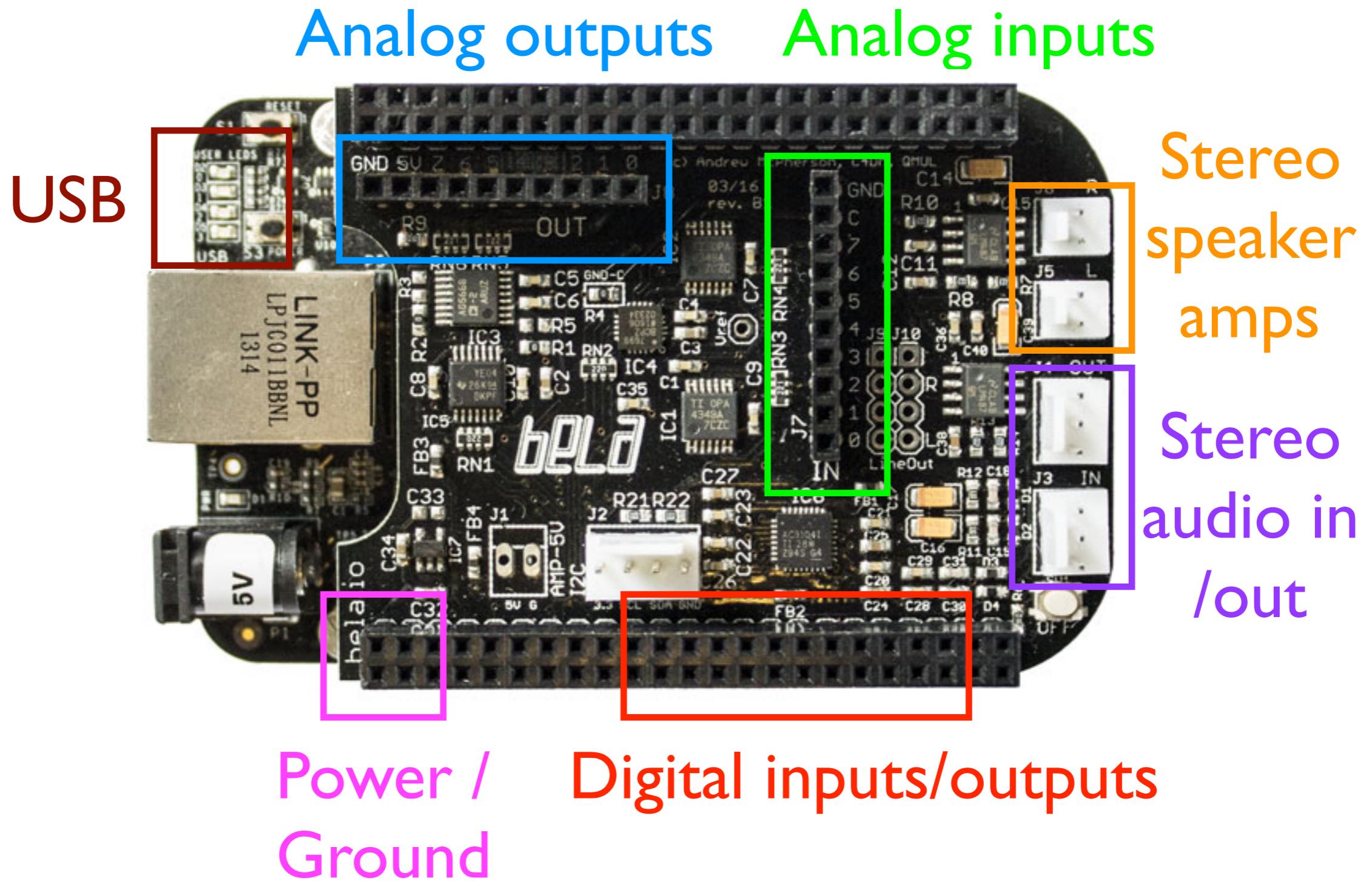


Csound



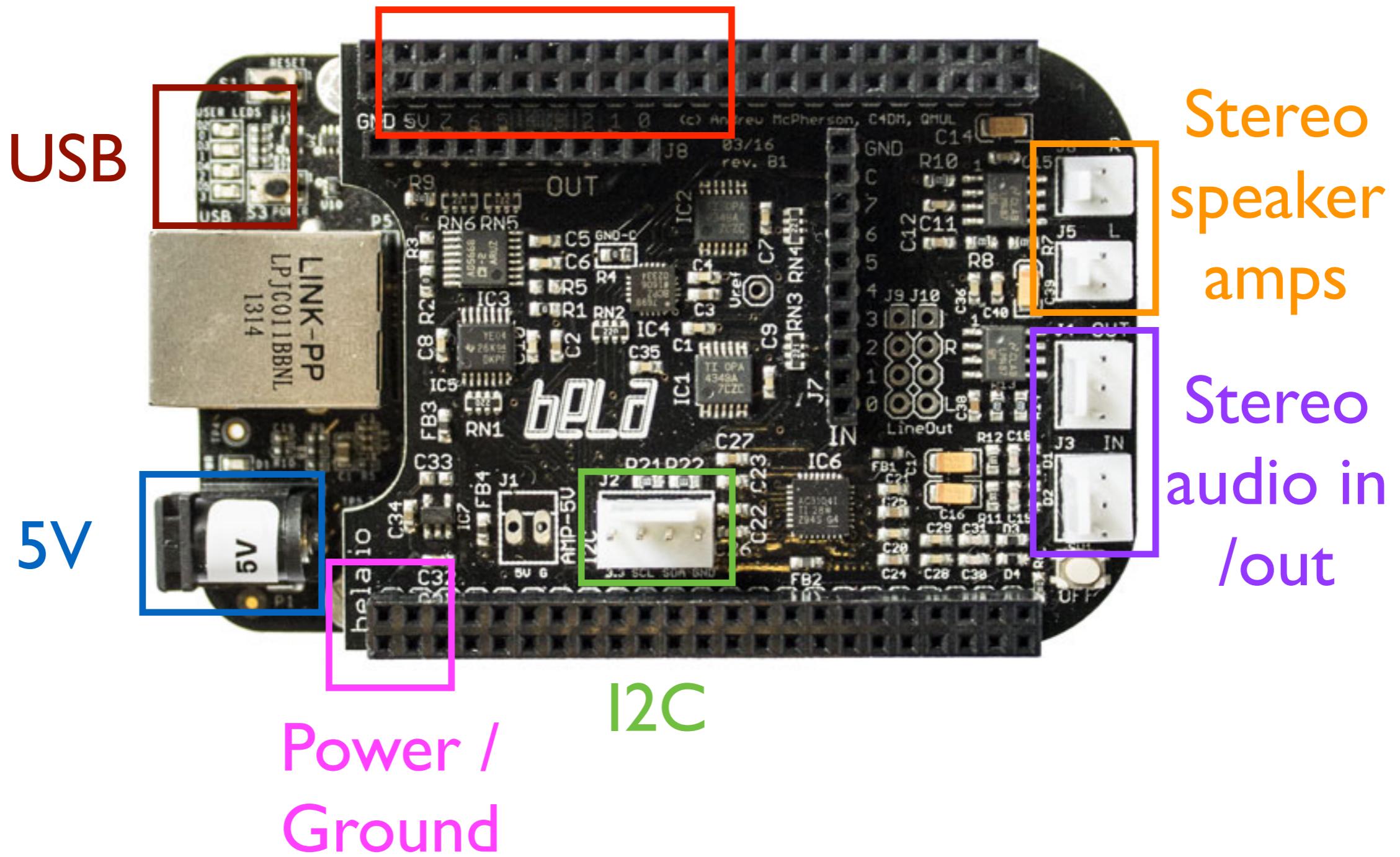
SOUL

# Bela hardware



# Bela hardware

Digital inputs/outputs



# Required software

**Firefox/Chrome browser (\*recommended)**

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 10 (and older)

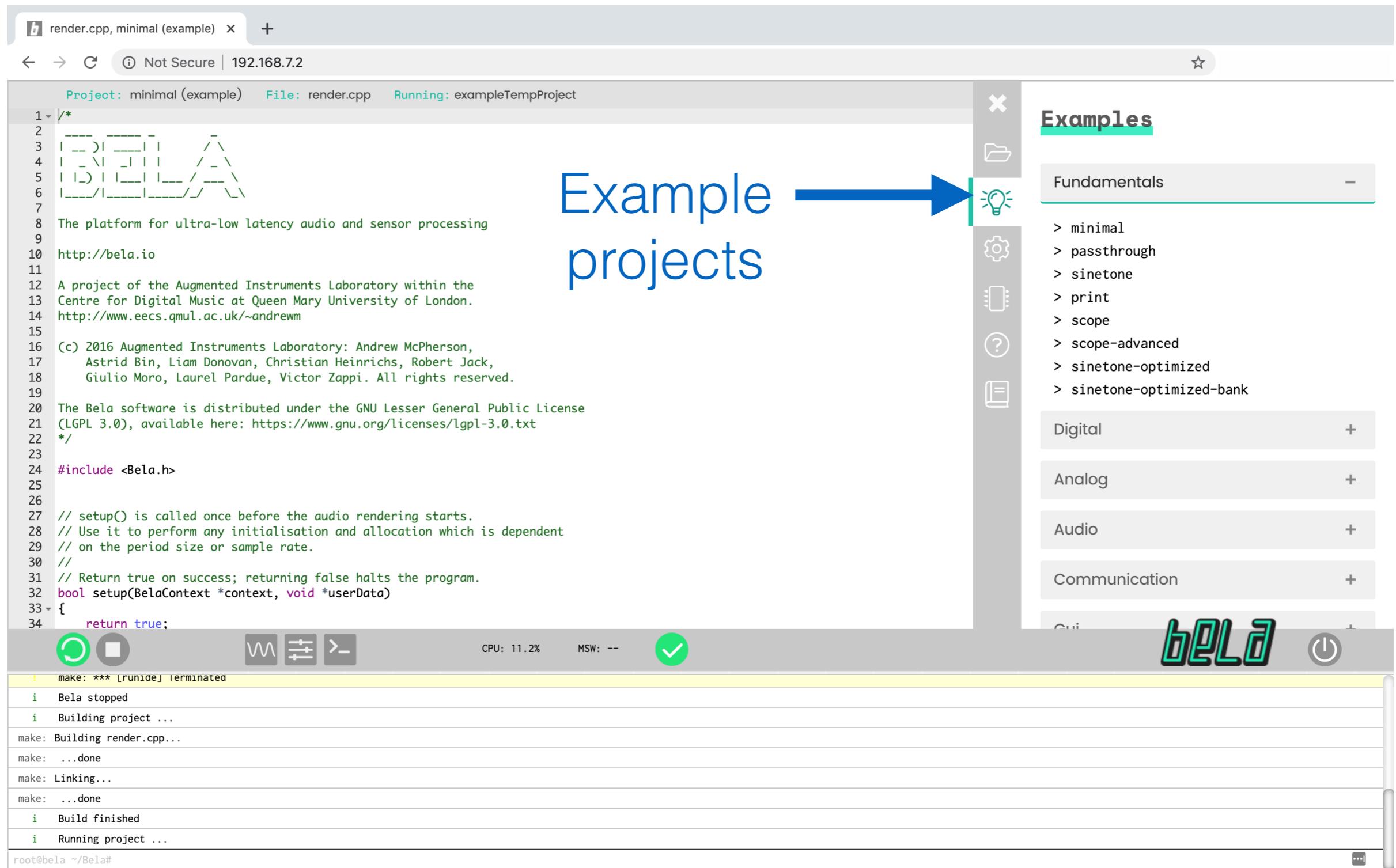
<http://192.168.6.2>

Windows 11

<http://192.168.7.2>

4. Plug in barrel jack cable (because we are using the amps)

# 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 (LGPL 3.0), and copyright information. On the right is a sidebar titled "Examples" with sections for "Fundamentals", "Digital", "Analog", "Audio", "Communication", and "GUI". A large blue arrow points from the text "Example projects" towards the sidebar. At the bottom, there's a terminal window showing build logs and a status bar with CPU usage (11.2%), MSW (--), and a green checkmark icon.

render.cpp, minimal (example) x +

← → C ⓘ 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;  
}
```

Example projects →

Examples

Fundamentals

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

Digital

Analog

Audio

Communication

GUI

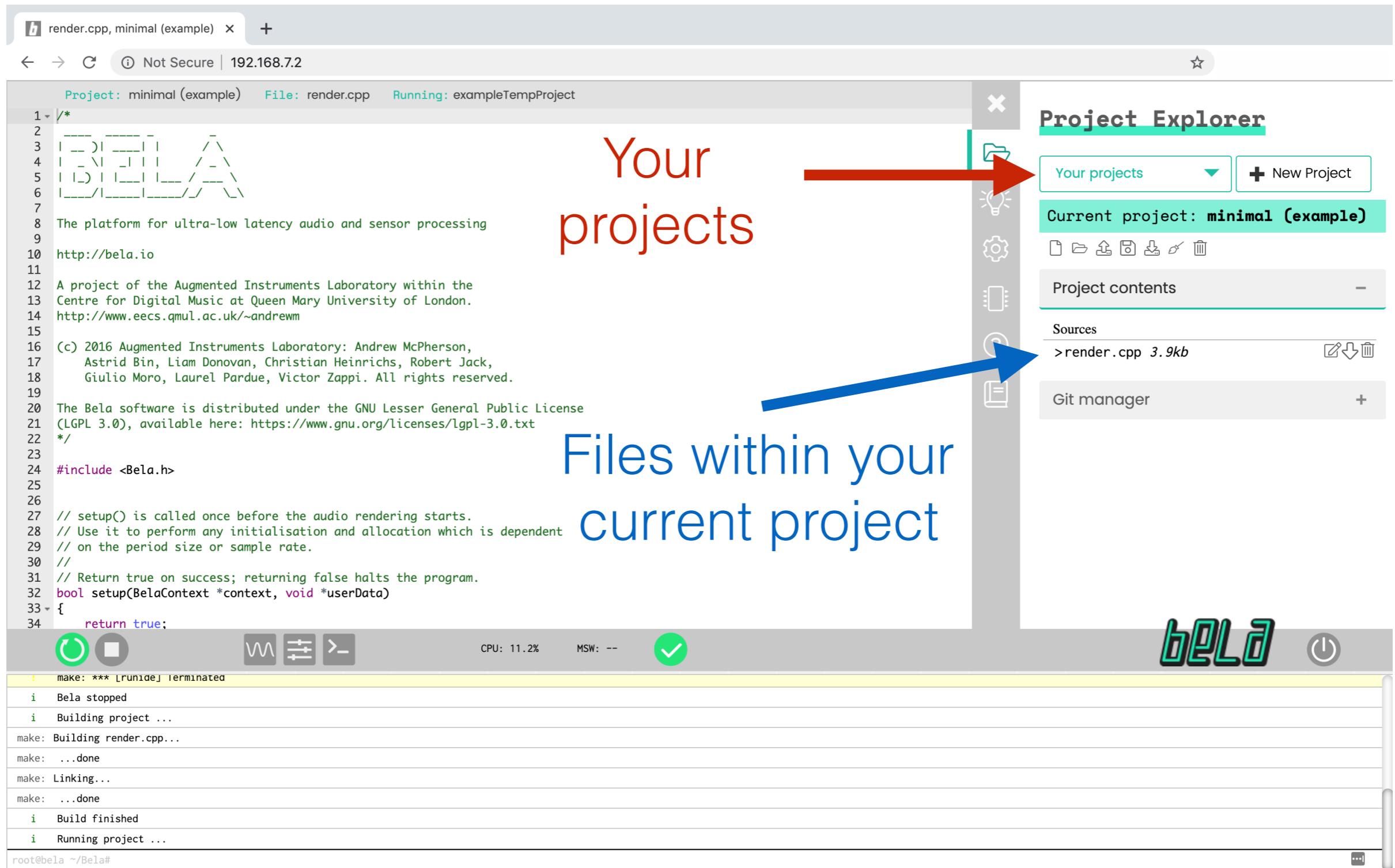
bela

CPU: 11.2% MSW: -- ✓

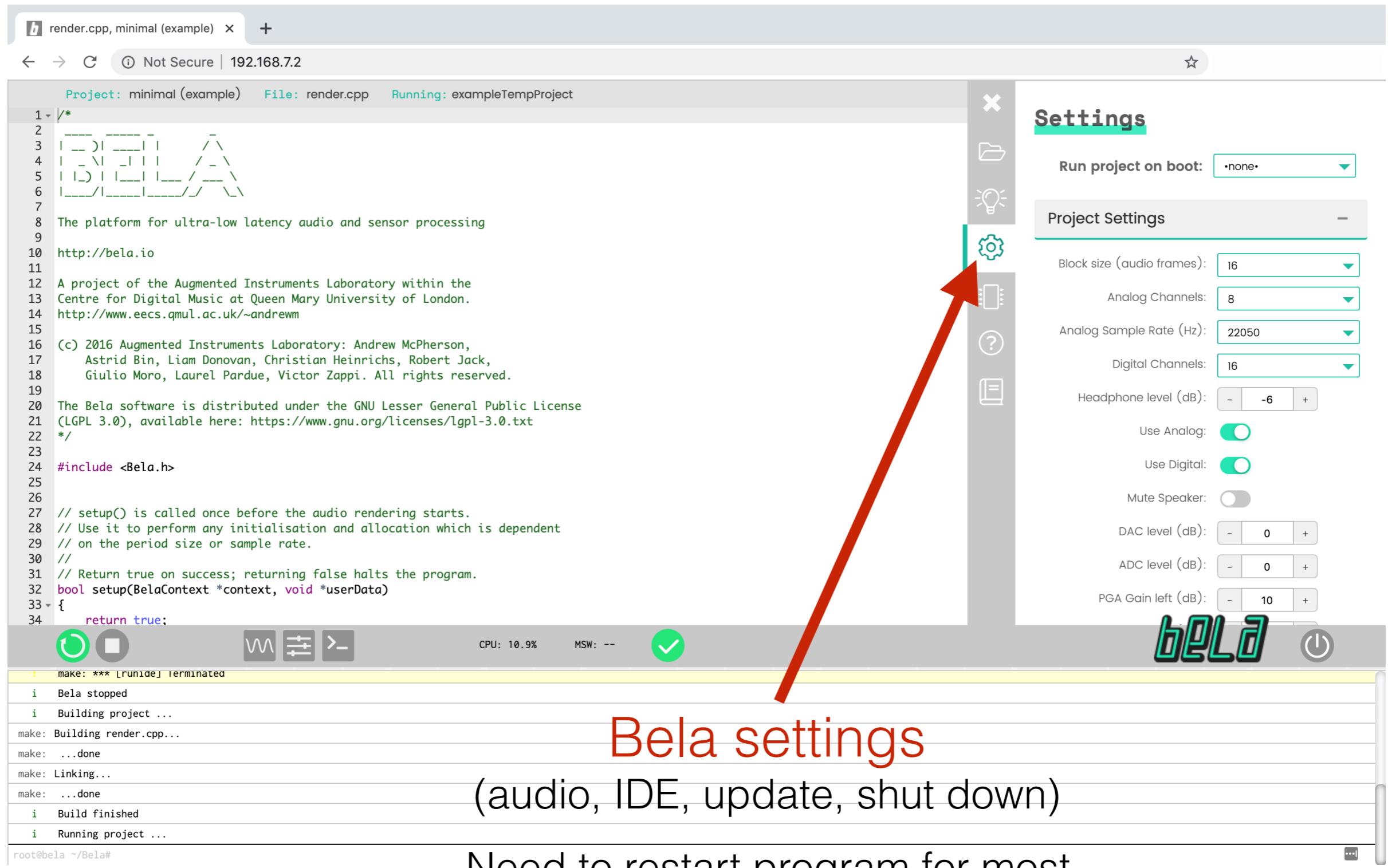
```
! make: *** [runide] terminated  
i Bela stopped  
i Building project ...  
make: Building render.cpp...  
make: ...done  
make: Linking...  
make: ...done  
i Build finished  
i Running project ...
```

root@bela ~/Bela#

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



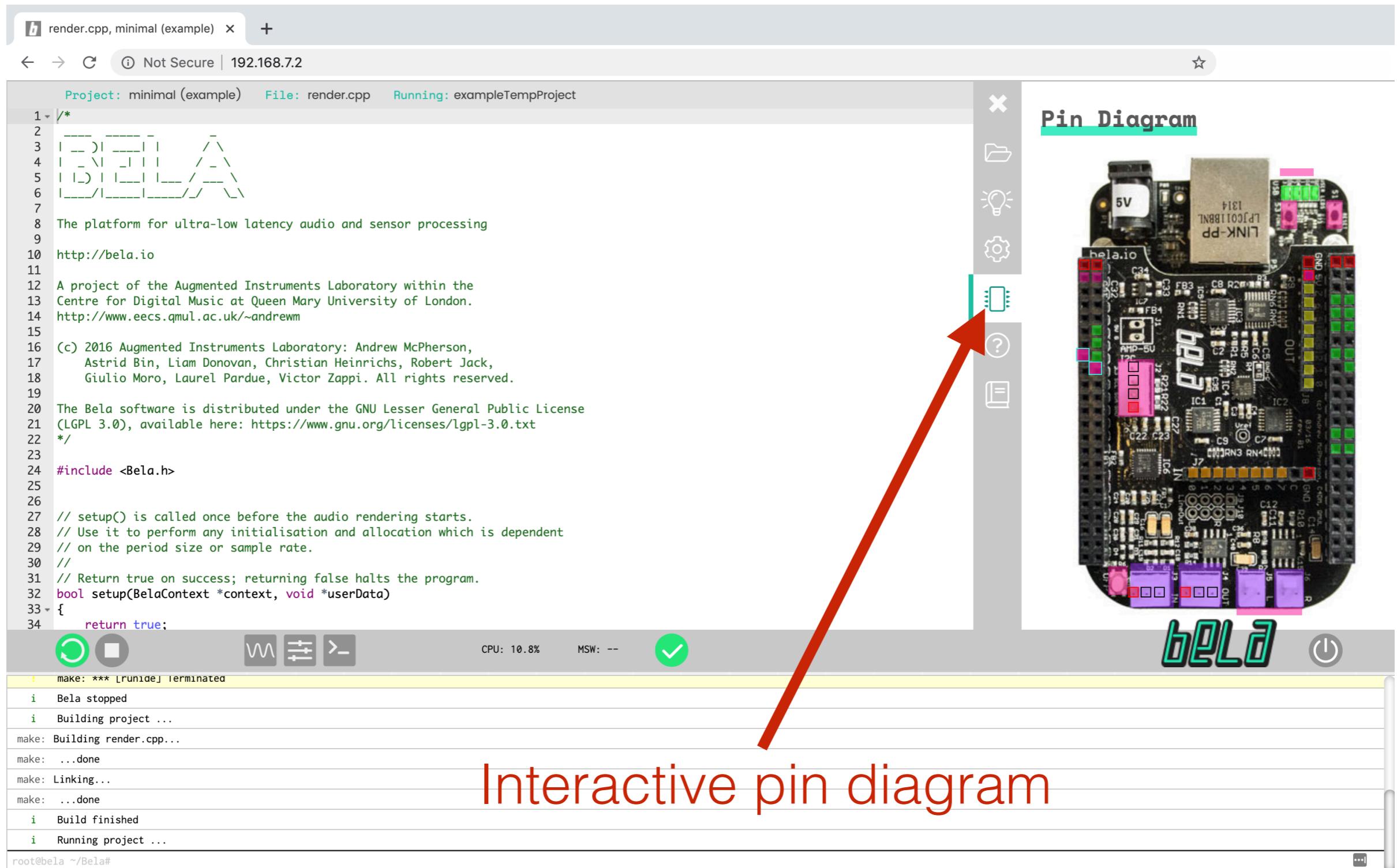
# Bring up the Bela IDE: <http://bela.local/>



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.

```
/*  
 *-----|-----|  
 * | \ | / | | \ |  
 * | -\ | / | | \ |  
 * | | \ | / | | \ |  
 * | / | / | / | / |  
 *  
 * 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;  
}  
  
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

# 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](https://docs.bela.io)

# Hello world: sinetone

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

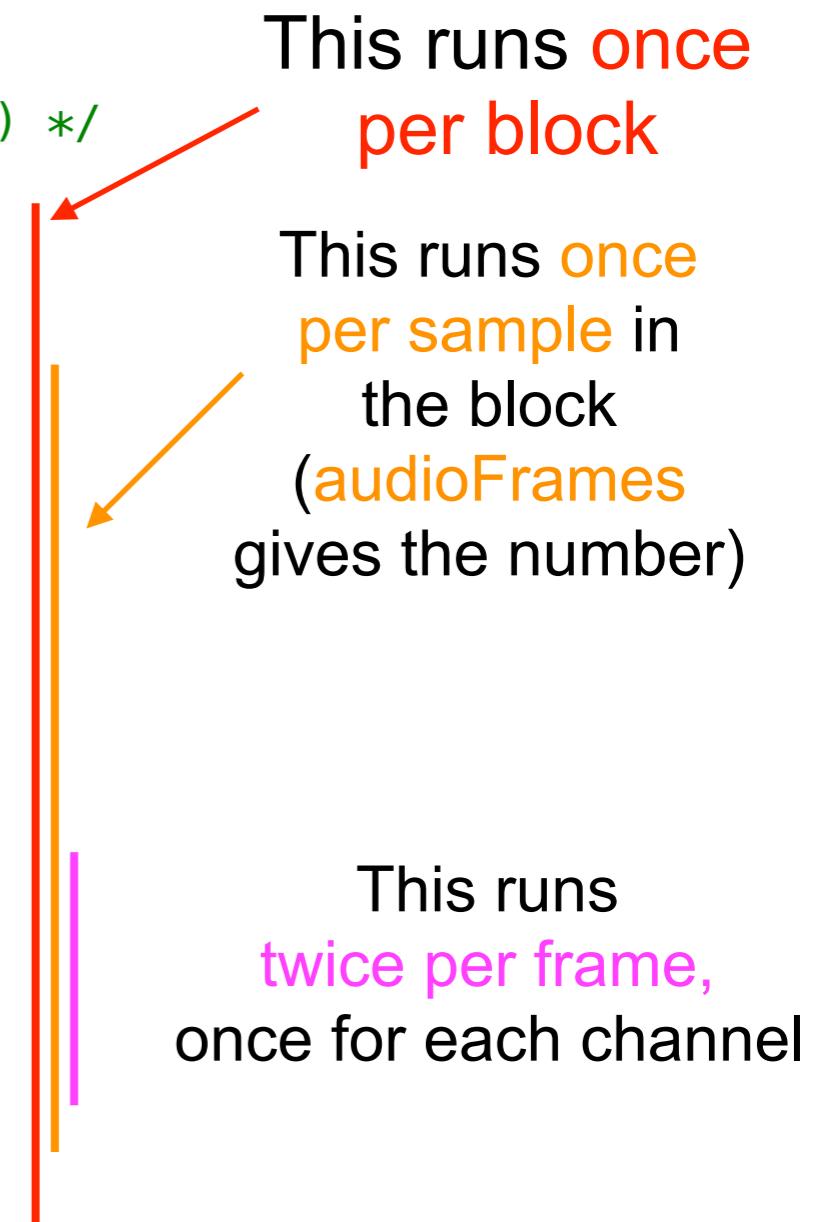
float gPhase = 0.0; /* Phase of the oscillator (global variable) */

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**



Keep everything on  
your breadboard!

We will be using all sensors  
through the workshop

# Haptic Actuators

Eccentric Rotating Mass (ERM)



Linear Resonant Actuators (LRA)



Piezo Haptic Actuators



Brushless DC Motors



Solenoids



Audio/Tactile Transducers  
(drivers)



# Haptic Actuators

Eccentric Rotating Mass (ERM)



Linear Resonant Actuators (LRA)



Piezo Haptic Actuators



Brushless DC Motors



Solenoids



Audio/Tactile Transducers  
(drivers)

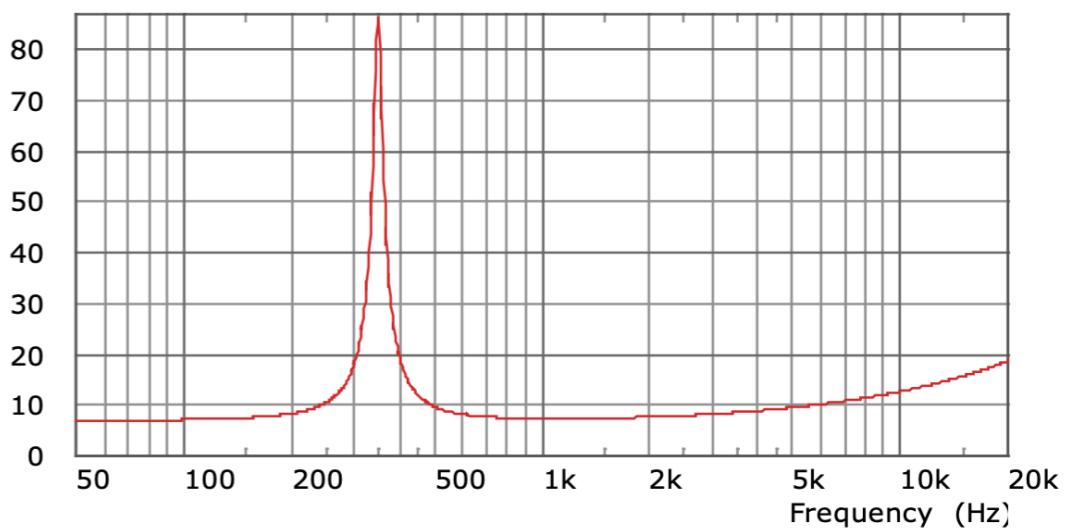


# Vibro-tactile actuator



Audible Range 20Hz - 20kHz

Impedance  
(ohm) Amplitude



# Vibro-tactile actuator

The screenshot shows the Bela IDE interface. On the left is the code editor with the file `render.cpp` open, containing C++ code for a driver test. The code includes includes for `Bela.h`, `Oscillator.h`, `Gui.h`, `GuiController.h`, and `cmath`. It defines audio output channels, oscillators (haptic, audible, AM LFO), and frequency ranges. The `GUI` section is commented out. Below the code editor is a toolbar with icons for refresh, stop, play, and export. In the center is the Project Explorer panel, which shows the current project is `haid_00_driver_test` and lists the source file `render.cpp` (4.90kB). On the right is the terminal window, which shows the command `root@bela:~$ Bela#`. A red arrow points from the terminal up towards the code editor, and a blue arrow points from the terminal up towards the Project Explorer. The word "run!" is written in red at the bottom left, and "GUI" is written in blue at the bottom center.

## haid\_00\_driver\_test

# Vibro-tactile actuator

The screenshot shows the Bela IDE interface. The top left displays the project file 'haid\_00\_driver\_test' and the file 'render.cpp'. The code editor on the left contains C++ code for audio processing, including oscillator definitions and GUI control variables. The top right features the 'Project Explorer' with a 'New Project' button. Below the code editor is the 'Controls' panel, which includes four sliders: 'Haptic OSC freq' at 95, 'AM LFO freq' at 0, 'Haptic Amplitude' at 1, and 'Audio Amplitude' at 0.15. At the bottom, there's a toolbar with various icons and a status bar showing CPU and MSW usage. A red arrow points upwards from the terminal window, and a blue arrow points upwards from the 'run!' text, both pointing towards the controls panel.

```
1 /*
2  *
3  * http://bela.io
4  */
5
6
7 http://bela.io
8 */
9
10 #include <Bela.h>
11 #include <libraries/Osc
12 #include <libraries/Gui
13 #include <libraries/Gui
14 #include <cmath>
15
16 /* Audio output channel
17 unsigned int kHapticOut
18 unsigned int kAudibleOut
19
20 /** OSCILLATORS ***/
21 // Oscillator (aliased)
22 Oscillator hapticOsc;
23 Oscillator audibleOsc;
24 Oscillator amLfo;
25
26 // Frequency range for
27 float gHapticOscFreqRan
28 float gAudibleOscFreqRa
29 float gAmLFOFreqRange[2
30
31 /** GUI **/
```

Project Explorer

New Project

## Controls

Haptic OSC freq	95
AM LFO freq	0
Haptic Amplitude	1
Audio Amplitude	0.15

root@bela: ~ Bela#

run!

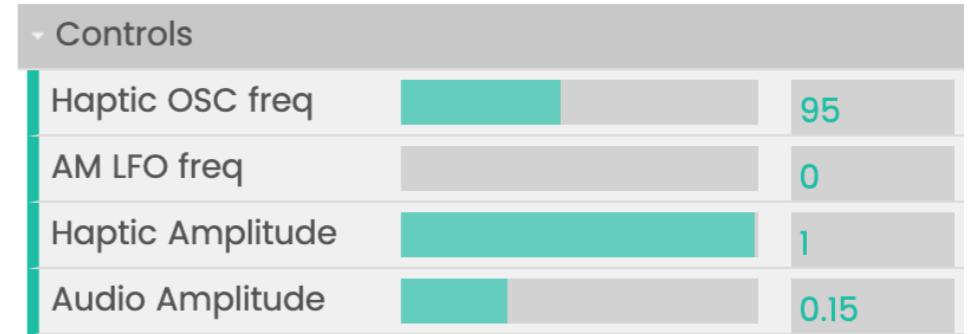
GUI

## haid\_00\_driver\_test

# Vibro-tactile actuator

```
12 #include <libraries/Gui/Gui.h>
13 #include <libraries/GuiController/GuiController.h>

31 /** GUI */
32 // GUI instance
33 Gui gui;
34 // GUI controller instance
35 GuiController controller;
36
37 // Indices for different slider controls
38 unsigned int gHaptic0scFreqSliderId;    // Frequency of haptic oscillator
39 unsigned int gAMLfoFreqSliderId;        // Frequency of AM LFO
40 unsigned int gHapticAmplitudeSliderId;   // Amplitude of Haptic output
41 unsigned int gAudibleAmplitudeSliderId; // Amplitude of Audible output
42
```



## setup:

```
65     gHapticAmplitudeSliderId = controller.addSlider("Haptic Amplitude", 1.0, 0, 1.0, 0.001);
66     gAudibleAmplitudeSliderId = controller.addSlider("Audio Amplitude", 0.15, 0, 0.5, 0.001);
```

## render:

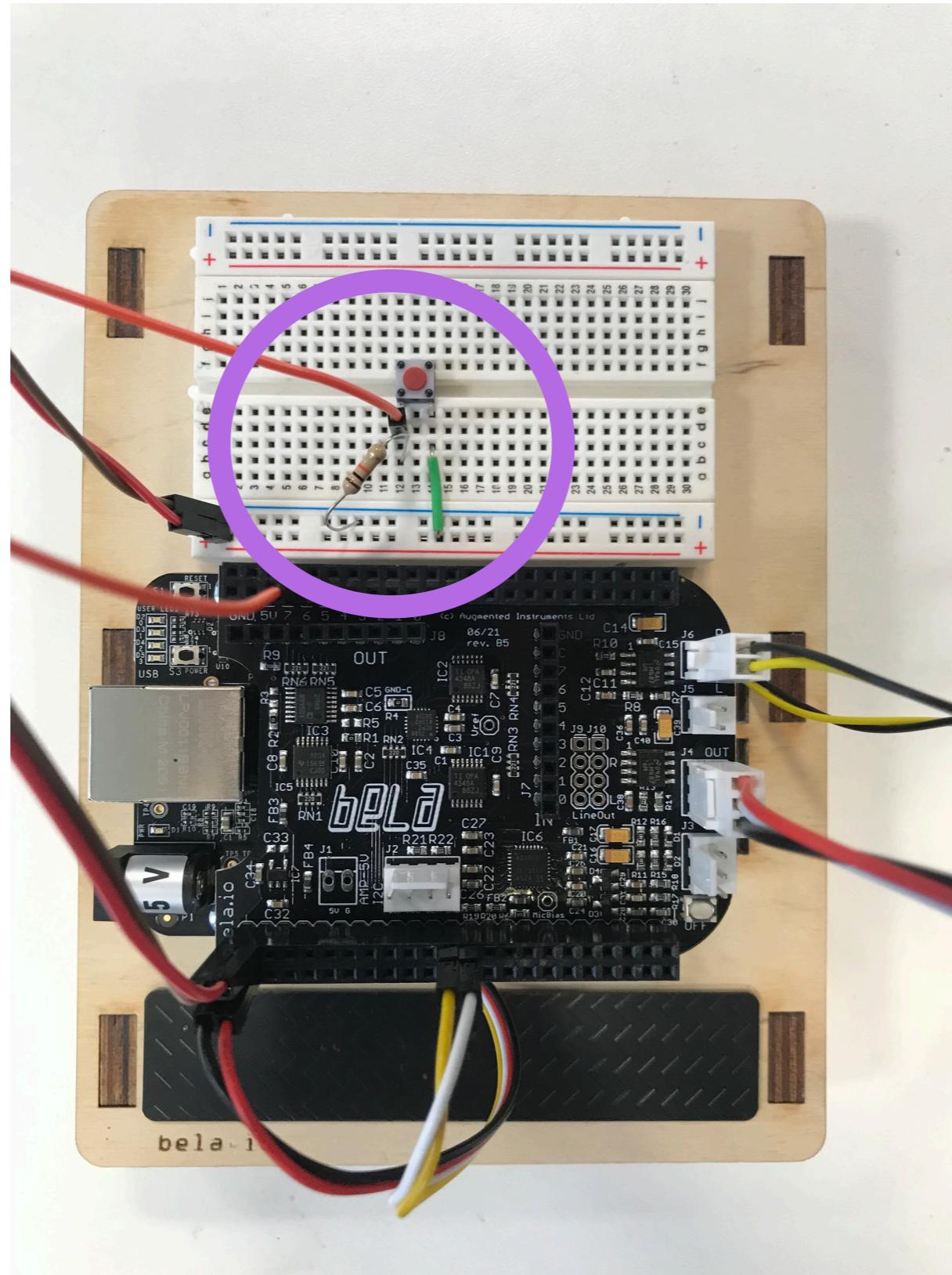
```
76     float hapticAmplitude = controller.getSliderValue(gHapticAmplitudeSliderId);    // Amplitude of Haptic output
77     float audibleAmplitude = controller.getSliderValue(gAudibleAmplitudeSliderId);    // Amplitude of Audible output
```

haid\_00\_driver\_test

# Button

Pull-down resistor

Connected to Digital 0



haid\_01\_button

# Button

```
24  /* Digital input for button */  
25  unsigned int kButtonInput = 0; // Connect button to digital input (see diagram on the IDE)  
26  unsigned int gPreviousButtonState = 0;  
27
```

## setup:

```
79   /** BUTTON **/  
80   pinMode(context, 0, kButtonInput, INPUT); // set button's digital to input  
81
```

## render:

```
110  /* Frame loop (audio rate) */  
111  for(unsigned int n = 0; n < context->audioFrames; n++) {  
112  
113      /** DIGITAL **/  
114      // Read the state of the button  
115      int buttonState = digitalRead(context, n, kButtonInput);  
116
```

# Button

```
24 /* Digital input for button */  
25 unsigned int kButtonInput = 0; // Connect button to digital input (see diagram on the IDE)  
26 unsigned int gPreviousButtonState = 0;  
27
```

## setup:

```
79     /** BUTTON **/  
80     pinMode(context, 0, kButtonInput, INPUT); // set button's digital to input  
81
```

## render:

```
110     /* Frame loop (audio rate) */  
111     for(unsigned int n = 0; n < context->audioFrames; n++) {  
112  
113         /** DIGITAL **/  
114         // Read the state of the button  
115         int buttonState = digitalRead(context, n, kButtonInput);  
116  
117         // If button pressed...  
118         if(buttonState == 1)  
119             g0scToggle = true;  
120         // If button released...  
121         else  
122             g0scToggle = false;  
123
```

haid\_01\_button

# Button

## I. Turn driver only for a few milliseconds when button is pressed

```
38 bool g0scToggle = false; // Toggle oscillator(s) on (true) and off (false)
39
40 float g0scOnTimeRange[2] = {5.0, 120.0}; // Time that the oscillator is on (range for min and max values) (milliseconds)
41 int g0scOnCount = 0; // Counter for toggling oscillator on/off
--
```



### [Task I]:A

```
111     g0scOnCount = (int)(context->audioSampleRate * osc0nTimeMs * 0.001); // Reset counter
--
```

### [Task I]:B

```
118     // If count has elapsed...
119     if(g0scOnCount >= 0)
120     {
121         g0scToggle = true; // Toggle oscillator on
122         g0scOnCount--; // Decrement counter
123     }
124     // otherwise ...
125     else
126     {
127         g0scToggle = false; // Toggle oscillator off
128     }
--
```

haid\_01\_button\_B

# Button

2. Remove clicks

haid\_01\_button\_C

# Button

## 2. Remove clicks - ASR (Attack/Sustain/Release) envelope

```
18 #include "ASR/ASR.h"

56 /** ENVELOPE */
57 ASR asrEnvelope; // Attack-Sustain-Release
```

setup:

```
84 /** ENVELOPE */
85 asrEnvelope.setup(g0scOnTimeRange[0]*0.001, g0scOnTimeRange[0]*0.001, context->audioSampleRate);
```

render:

```
122     asrEnvelope.triggerOn(); // Trigger envelope!!!
123     rt_printf("Trigger envelope!\n");
124

130     // If envelope reaches the SUSTAIN state (ATTACK state finis
131     if(asrEnvelope.getCurrentState() == ASR::SUSTAIN)
132     {
133         asrEnvelope.triggerOff(); // Release envelope!!!
134         rt_printf("Release envelope!\n");
135     }

138     /** AUDIO */
139     // Compute output by scaling with the amplitude
140     float envelope = asrEnvelope.process();
141     float hapticOut = haptic0sc.process() * hapticAmplitude * envelope; // Haptic output
142     float audibleOut = audible0sc.process() * audibleAmplitude * envelope; // Audio output
143
```

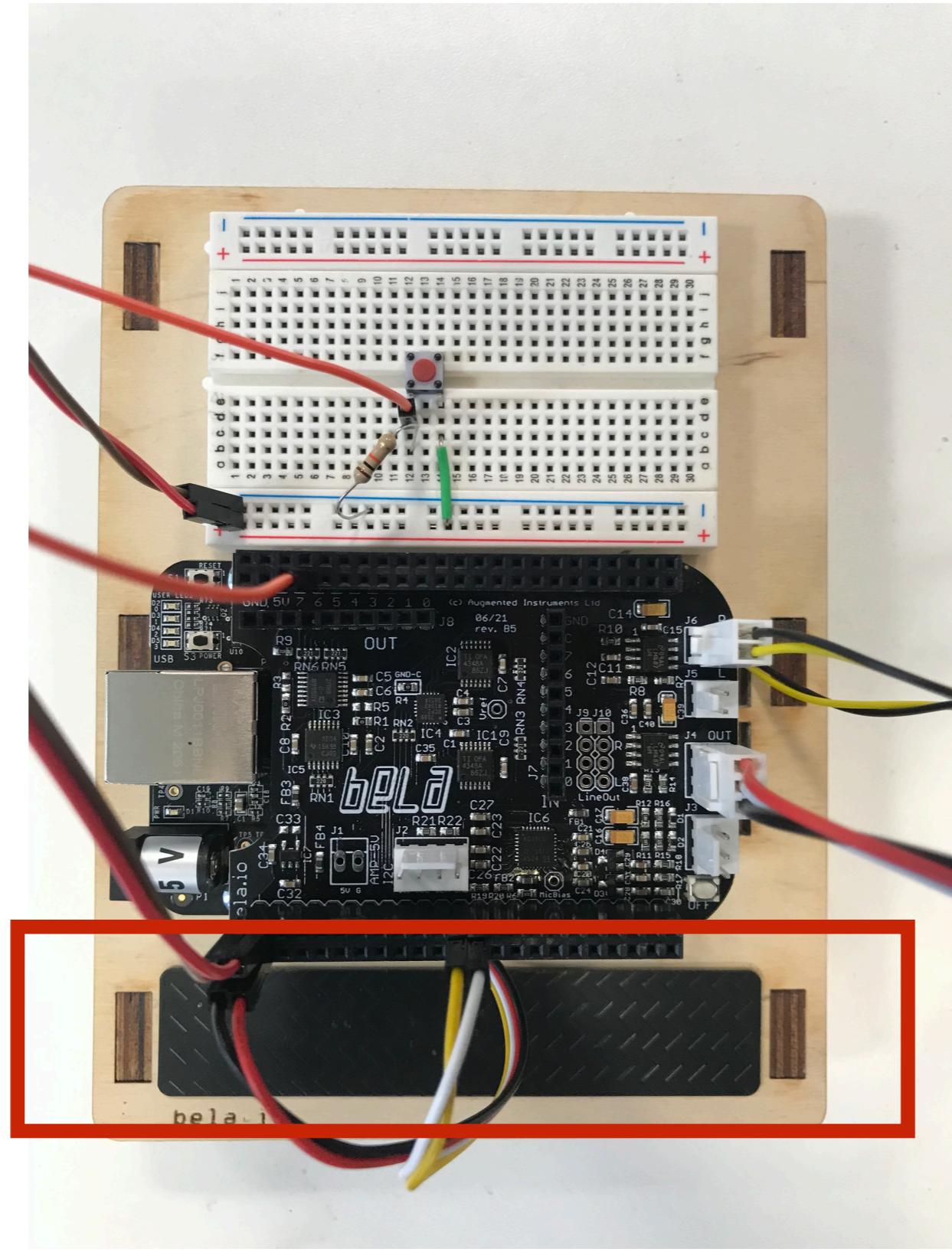
haid\_01\_button\_

# 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



# 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 (A)

I. Turn driver on for a few ms when there is a new touch

[Task I]

Hint! - *gNumActiveTouches*

```
154 // If there is a new touch
155 if(gNumActiveTouches != gPrevNumActiveTouches)
156 {
157     g0scOnCount = (int)(context->audioSampleRate * oscOnTimeMs * 0.001); // Reset counter
158 }
159 // Update previous number to current number
160 gPrevNumActiveTouches = gNumActiveTouches;
161
```

haid\_01\_trill\_A

# Trill Bar (B)

## 2. Scale vibration frequency based on number of touches

### [Task 2]

```
154     g0scOnCount = (int)(context->audioSampleRate * osc0nTimeMs * 0.001); // Reset counter
155
156     // Scale haptic oscillator frequency based on number of touches
157     float haptic0scFreq = map(gNumActiveTouches, 0, NUM_TOUCH, hapticMin0scFreq, gHaptic0scFreqRange[1]);
158     haptic0sc.setFrequency(haptic0scFreq); // Set frequency of haptic oscillator
159
160     // Get frequency for audio oscillator by remapping frequency of haptic oscillator to appropriate range
161     float audible0scFreq = map(haptic0scFreq, gHaptic0scFreqRange[0], gHaptic0scFreqRange[1], gAudible0scFreqRange[0], gAudible0scFreqRange[1]);
```

haid\_01\_trill\_B

# Trill Bar (C)

## 3. Scale vibration frequency based touch position

### [Task 3]

155  
156           // Scale haptic oscillator frequency based on touch location of first touch  
157       float haptic0scFreq = map(gTouchLocation[0], 0, 1, hapticMin0scFreq, gHaptic0scFreqRange[1]);  
158       haptic0sc.setFrequency(haptic0scFreq); // Set frequency of haptic oscillator  
159  
160           // Get frequency for audio oscillator by remapping frequency of haptic oscillator to appropriate range  
161       float audible0scFreq = map(haptic0scFreq, gHaptic0scFreqRange[0], gHaptic0scFreqRange[1], gAudible0scFreqRange[0], gAudible0scFr  
162       audible0sc.setFrequency(audible0scFreq); // Set frequency of  
163

haid\_01\_trill\_c

# Trill Bar (D)

4. Same as before but only use first touch (ignore the rest)

## [Task 4]

```
155 // If there is a new touch
156 if(gNumActiveTouches > 0)
157 {
158     g0scToggle = true;
159
160     // Scale haptic oscillator frequency based on touch location of first touch
161     float haptic0scFreq = map(gTouchLocation[0], 0, 1, minHaptic0scFreq, maxHaptic0scFreq);
162     haptic0sc.setFrequency(haptic0scFreq); // Set frequency of haptic oscillator
163
164     // Get frequency for audio oscillator by remapping frequency of haptic oscillator to appropriate range
165     float audible0scFreq = map(haptic0scFreq, gHaptic0scFreqRange[0], gHaptic0scFreqRange[1], gAudible0scFreqRange
166     audible0sc.setFrequency(audible0scFreq); // Set frequency of
167
168 }
169 else
170 {
171     g0scToggle = false;
172 }
```

haid\_01\_trill\_D

# Trill Bar (D2)

## 5. Map touch size to amplitude

```
162 // Scale haptic oscillator frequency based on touch location of first touch
163 float haptic0scFreq = map(gTouchLocation[0], 0, 1, minHaptic0scFreq, maxHaptic0scFreq);
164 haptic0sc.setFrequency(haptic0scFreq); // Set frequency of haptic oscillator
165
166 // Get frequency for audio oscillator by remapping frequency of haptic oscillator to appropriate ra
167 float audible0scFreq = map(haptic0scFreq, gHaptic0scFreqRange[0], gHaptic0scFreqRange[1], gAudible0
168 audible0sc.setFrequency(audible0scFreq); // Set frequency of
169
170 // Map touch size of first touch to amplitude
171 hapticAmplitude = map(gTouchSize[0], 0, 1, minHapticAmplitude, maxHapticAmplitude);
172 }
173 else
174 {
175     g0scToggle = false;
176 }
177 }
```

haid\_01\_trill\_D2

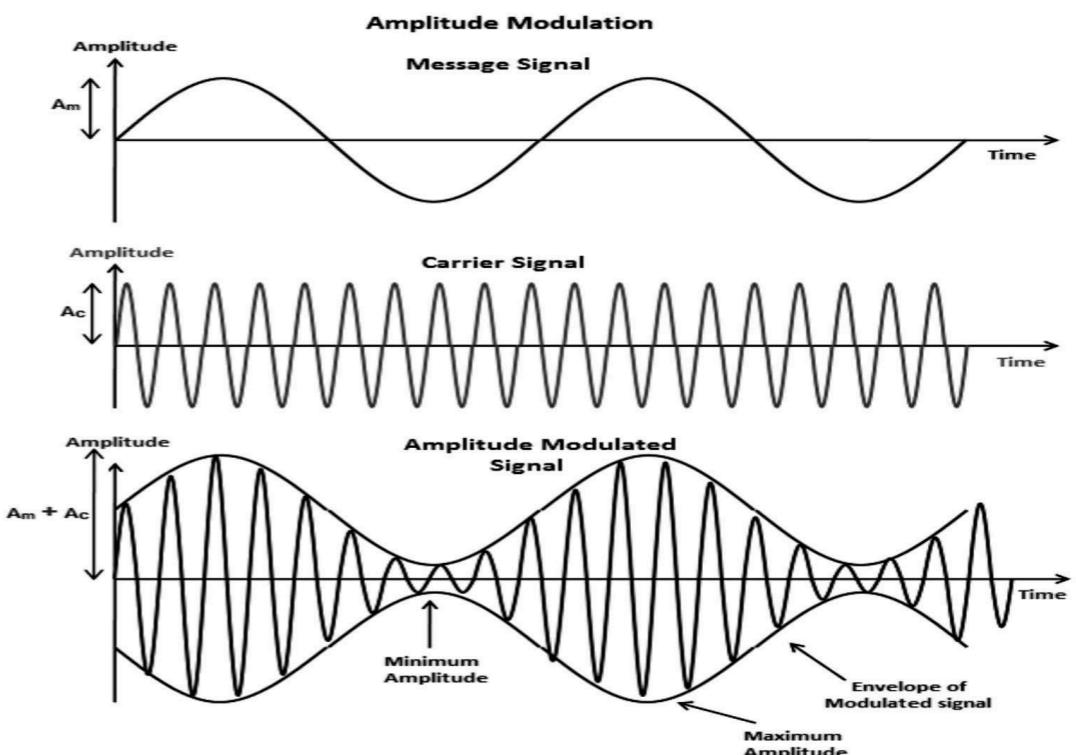
# Trill Bar (E)

## Amplitude Modulation

## Second Oscillator (low frequency)

## Scale and multiply gain

```
175 // Calculate Amplitude Modulation
176 float amLfoVal;
177 if(amLfoFreq == 0.0)    // If frequency is zero...
178     amLfoVal = 1.0;      // .. no amplitude modulation
179 else                      // If frequency is not zero...
180     amLfoVal = 0.5 * (amLfo.process() + 1); // .. re-scale oscillator output from (-1,1) to (0,1)
181
182
183
184
185 // Compute output by scaling with the gain and including AM modulation
186 float hapticOut = haptic0sc.process() * hapticAmplitude * amLfoVal;           // Haptic output
187 float audibleOut = audible0sc.process() * audibleAmplitude * amLfoVal; // Audio output
188
```



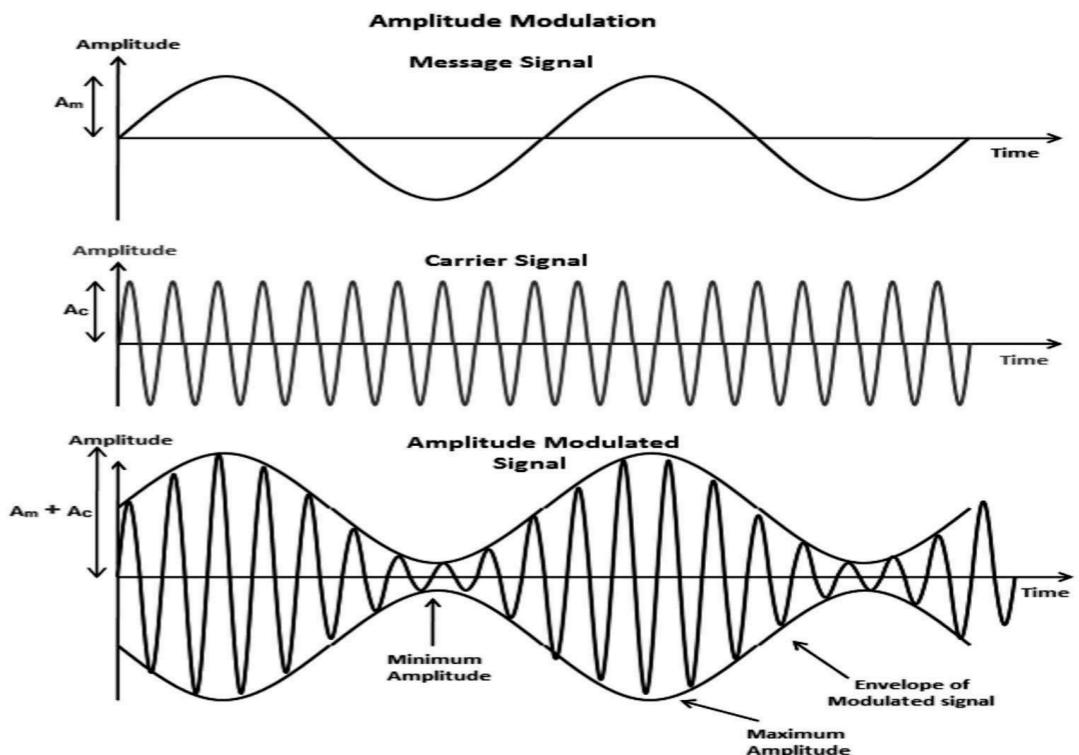
# Trill Bar (E)

## Amplitude Modulation

Second Oscillator (low frequency)

Scale and multiply gain

```
175 // Calculate Amplitude Modulation
176 float amLfoVal;
177 if(amLfoFreq == 0.0)    // If frequency is zero...
178     amLfoVal = 1.0;      // .. no amplitude modulation
179 else                      // If frequency is not zero...
180     amLfoVal = 0.5 * (amLfo.process() + 1); // .. re-scale oscillator output from (-1,1) to (0,1)
181
182
183
184
185 // Compute output by scaling with the gain and including AM modulation
186 float hapticOut = haptic0sc.process() * hapticAmplitude * amLfoVal;          // Haptic output
187 float audibleOut = audible0sc.process() * audibleAmplitude * amLfoVal; // Audio output
188
```



# Trill Bar (E)

6. Map touch location to LFO frequency

haid\_01\_trill\_E

# Trill Bar (F)

Target frequency: 320 Hz

LFO frequency increases when we approach target

```
169     // Calculate audio oscillator frequency based on touch location
170     float audible0scFreq = map(gTouchLocation[0], 0, 1, gAudible0scFreqRange[0], gAudible0scFreqRange[1]);
171     audible0sc.setFrequency(audible0scFreq); // Set frequency of audible oscillator
172
173     // Calculate absolute difference to target Frequency
174     float freqDiff = fabs(gTargetAudibleFreq - audible0scFreq);
175
176     // Remap frequency difference to AM LFO frequency range so that the LFO is slower the closer we get to
177     amLfoFreq = map(freqDiff, 0.0, 100.0, gAmLFOFreqRange[0], gAmLFOFreqRange[1]);
178
179     amLfo.setFrequency(amLfoFreq); // Set frequency of haptic oscillator
180
```

haid\_01\_trill\_F

# Trill Bar (G)

Touch velocity mapped to amplitude

haid\_01\_trill\_G

# Bela + Haptics + Feedback

<https://blog.bela.io/dais-haptic-feedback-instrument/>

<https://blog.bela.io/crescente-electroacoustic-instrument/>

<https://blog.bela.io/pandoras-box-ward-slager/>

<https://blog.bela.io/here-and-now/>

<https://blog.bela.io/pulsewave-connects-wood-and-flesh/>

# Finding out more

<http://learn.bela.io>



Search The Bela Knowledge Base

f t i y m

## Bela Knowledge Base

This is the central hub for creating and learning with Bela.

Use the navigation menu on the left, or choose a section below:



### Get Started Guide

Learn all about Bela, and get to know your hardware and software.



### Tutorials

Access our tutorials in C++, Puredata, and more.



### Using Bela

Articles on how to embed, update, and Bela, and our technical explainers.

# 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

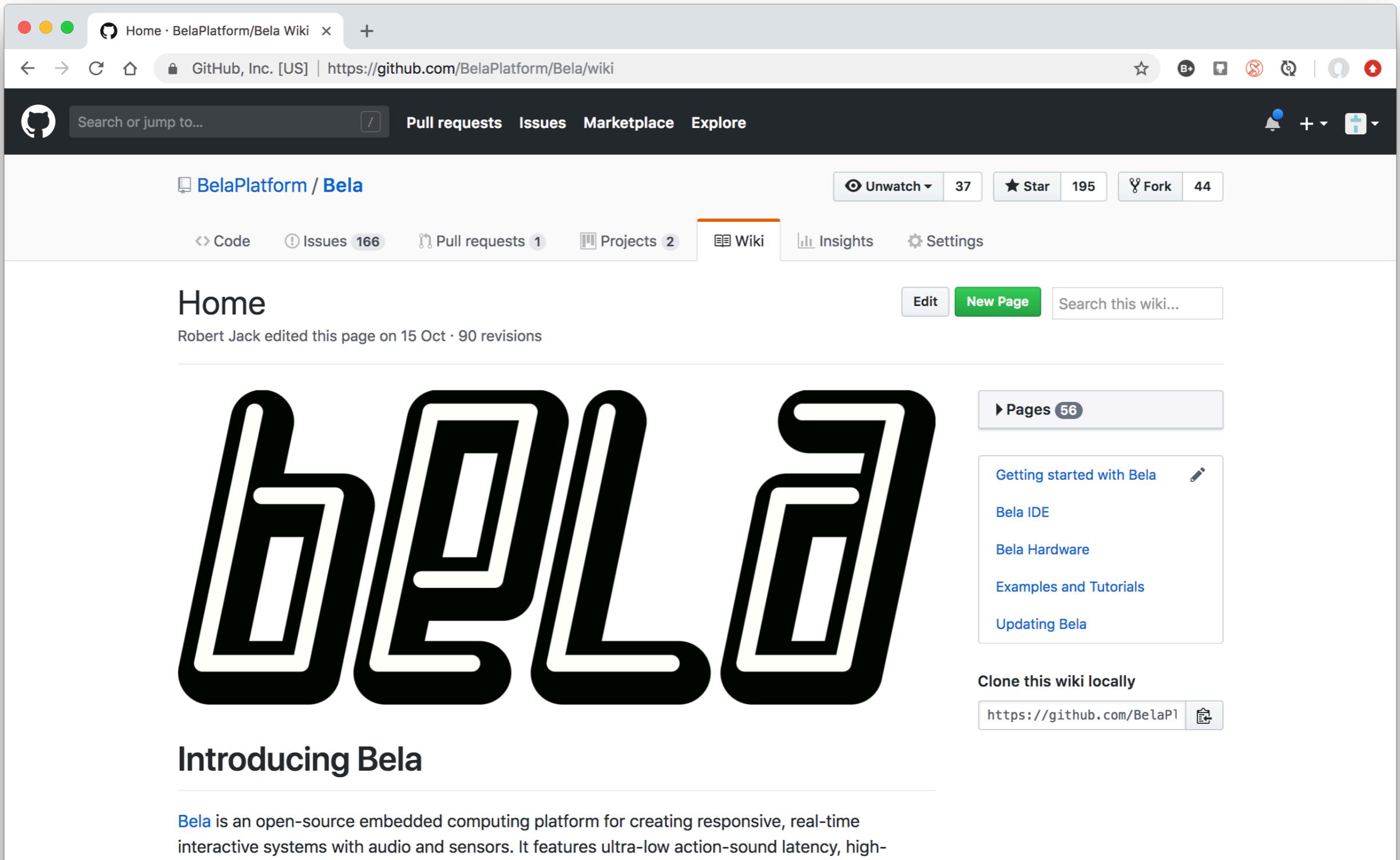
## <http://forum.bela.io>

The screenshot shows the Bela forum homepage with a list of recent discussions. The sidebar on the left includes links for 'Start a Discussion', 'All Discussions', 'Following', 'Tags', 'FAQ', 'General', 'Getting Started', 'Interactivity', 'Audio', 'Hardware', 'Software', 'Show and Tell', 'Forum', 'CTAG-ALSA', 'Solved', and 'Add to wiki'. The main content area displays ten discussions:

- Heavy, Pd and Enzienaudio** by Jukkapoika (replied an hour ago) - Software, Pure Data, 1 reply
- Monome Grid & Bela** by padenot (replied 2 hours ago) - Interactivity, 1 reply
- Scope sliders initialization** by stephanelesoinne (replied a day ago) - Software, 3 replies
- How to drive a LRA by bela board?** by giuliomoro (replied 2 days ago) - Audio, 3 replies
- high-performance-mode** by giuliomoro (replied 2 days ago) - General, 4 replies
- Bela as a plug and play USB Audio Soundcard Device?** by Gladgrif (replied 3 days ago) - Audio, 6 replies
- Bela out to XLR?** by giuliomoro (replied 3 days ago) - Hardware, 2 replies
- Reading multiple samples and CPU overload.** by phevoso (replied 3 days ago) - Software, Pure Data, 2 replies
- Problem with scope after upgrade to last version** by giuliomoro (replied 3 days ago) - Software, IDE, 24 replies

# 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 links for Code, Issues (166), Pull requests (1), Projects (2), Wiki (selected), 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 with a "Pages 56" section containing links to "Getting started with Bela", "Bela IDE", "Bela Hardware", "Examples and Tutorials", and "Updating Bela". At the bottom, there is a "Clone this wiki locally" section with a link to the URL <https://github.com/BelaPl1>.

## Home

Robert Jack edited this page on 15 Oct · 90 revisions

# Bela

## Introducing Bela

Bela is an open-source embedded computing platform for creating responsive, real-time interactive systems with audio and sensors. It features ultra-low action-sound latency, high-

Pages 56

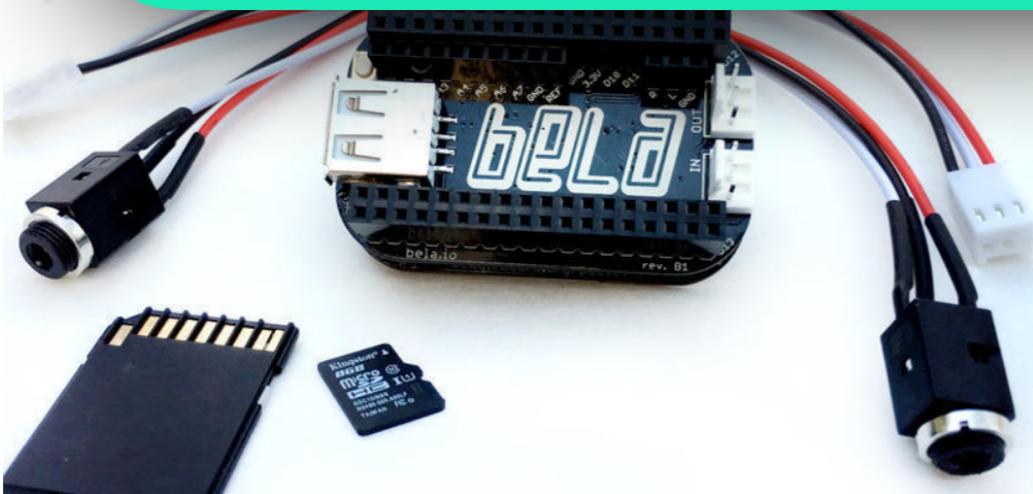
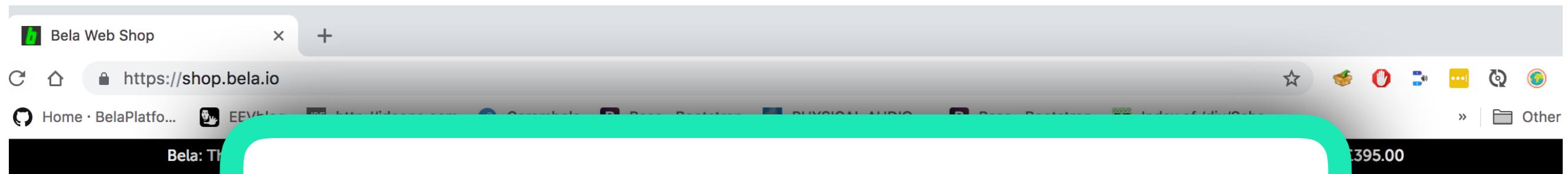
- Getting started with Bela
- Bela IDE
- Bela Hardware
- Examples and Tutorials
- Updating Bela

Clone this wiki locally

<https://github.com/BelaPl1>

# Where can I get one?

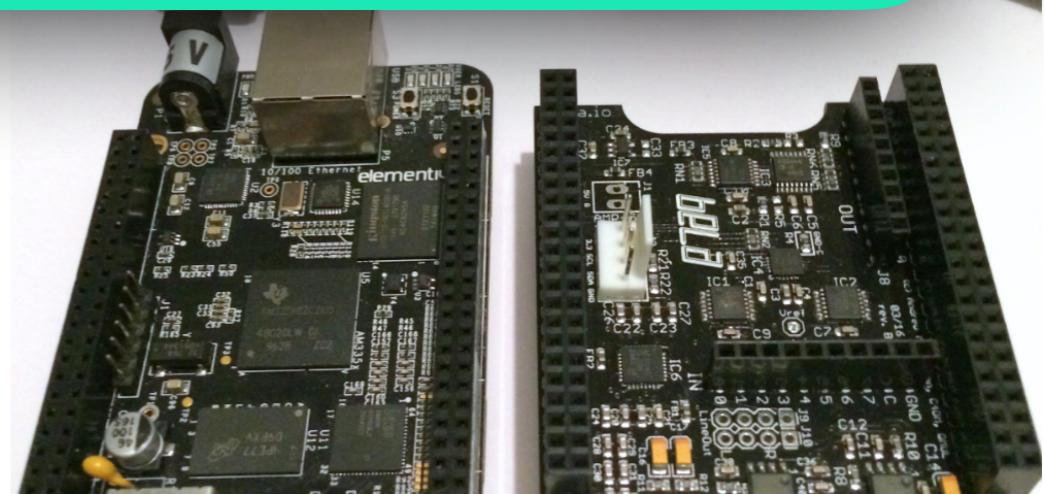
<https://shop.bela.io/>



## 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 ▶](#)