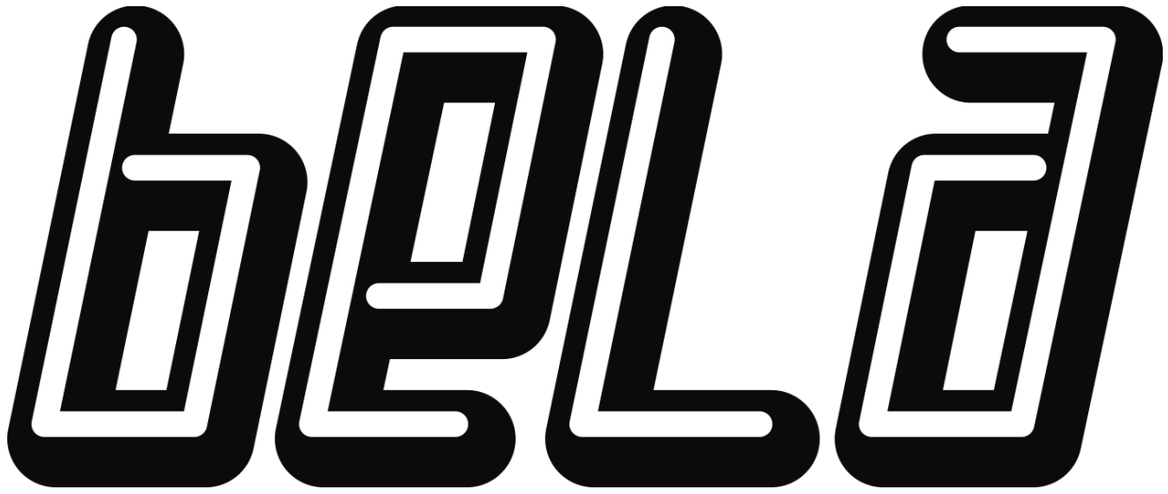
Open-source maker platform for beautifully responsive interactive audio

Adan L. Benito

@BelaPlatform

EPSRC centre for digital music Queen Mary University of London

bela.io



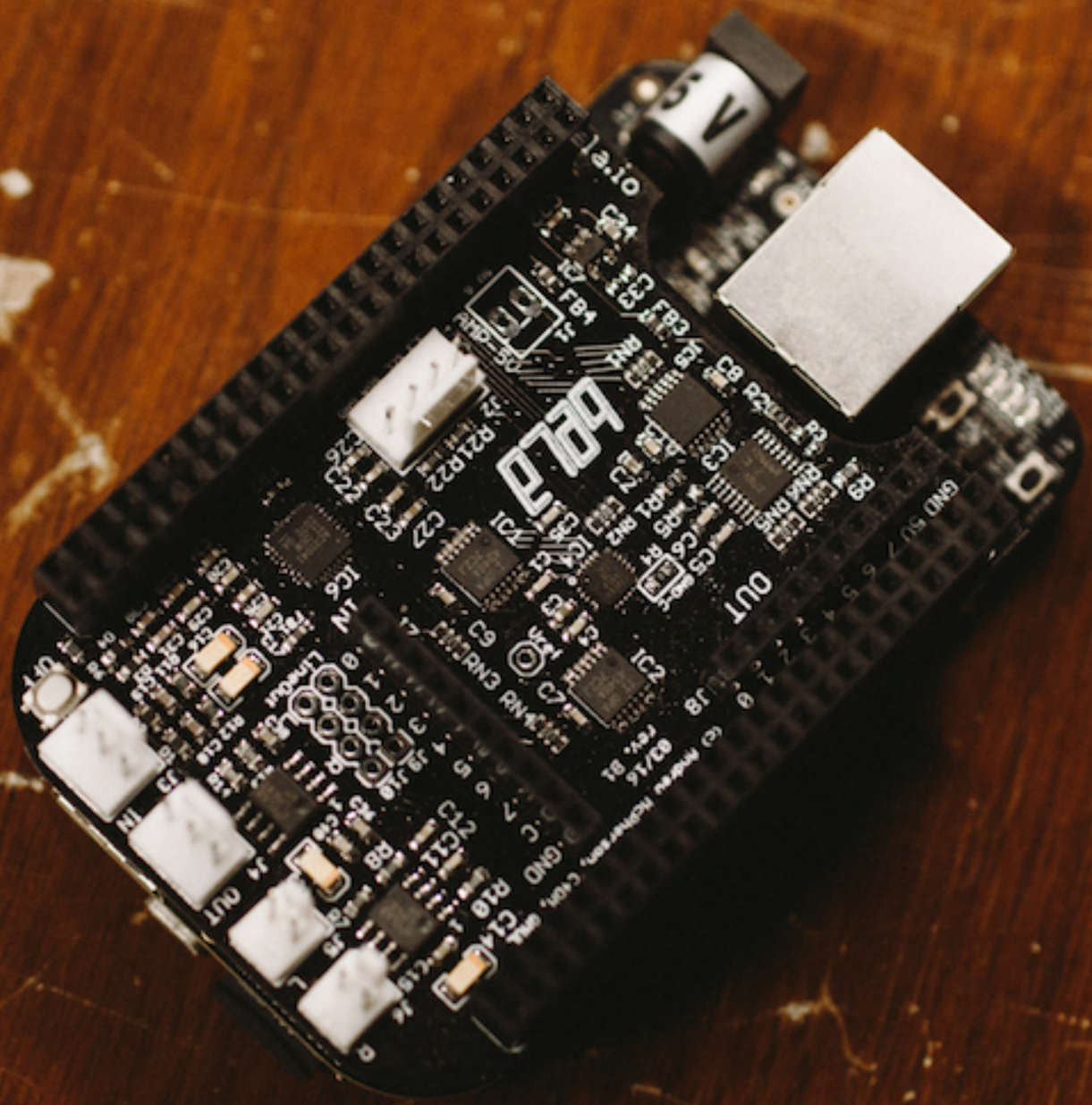
HAID

11th International Workshop on Haptic & Audio Interaction Design

25/08/2022

Queen Mary University of London

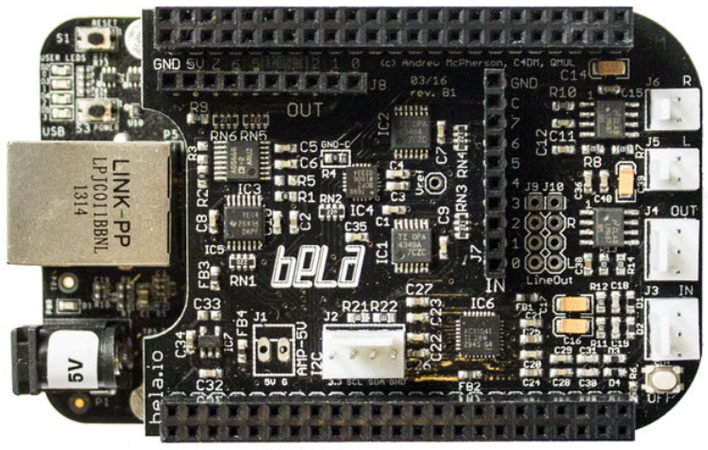
What is Bela?



Augmented Instruments Laboratory

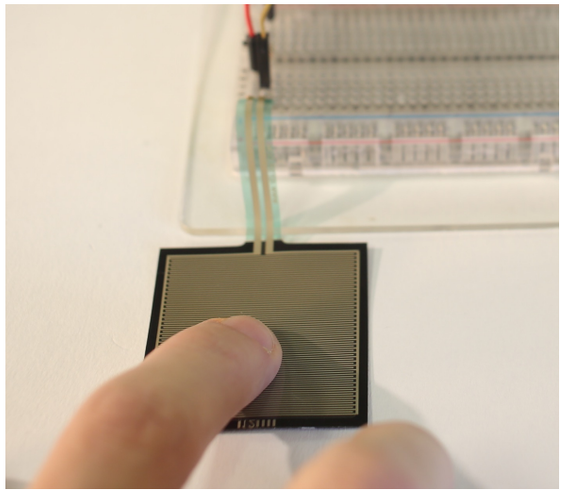
Bela

Embedded computer designed for interactive audio



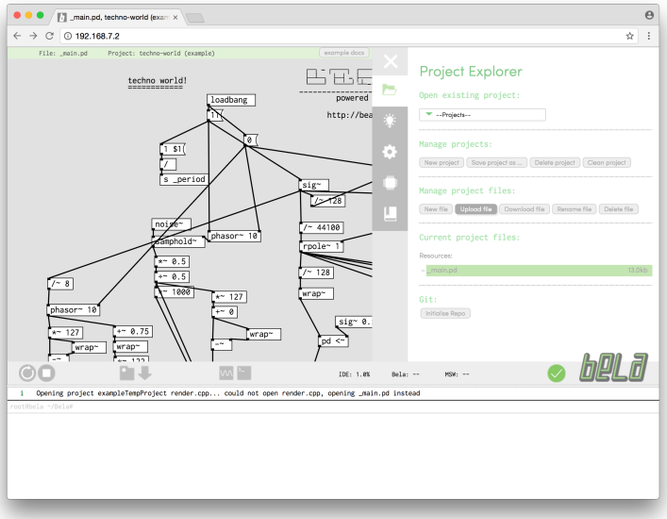
- Power of a single board Linux computer
- Connectivity of a micro-controller
- Combines the benefits of both

New approach to high-bandwidth sensor processing



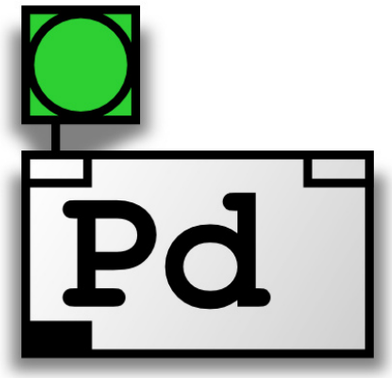
- Analog, digital I/O sampled at audio rate
- Ultra low action-sound latency
- Jitter-free alignment between audio and sensors

Open-source maker platform



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

Bela is a polyglot

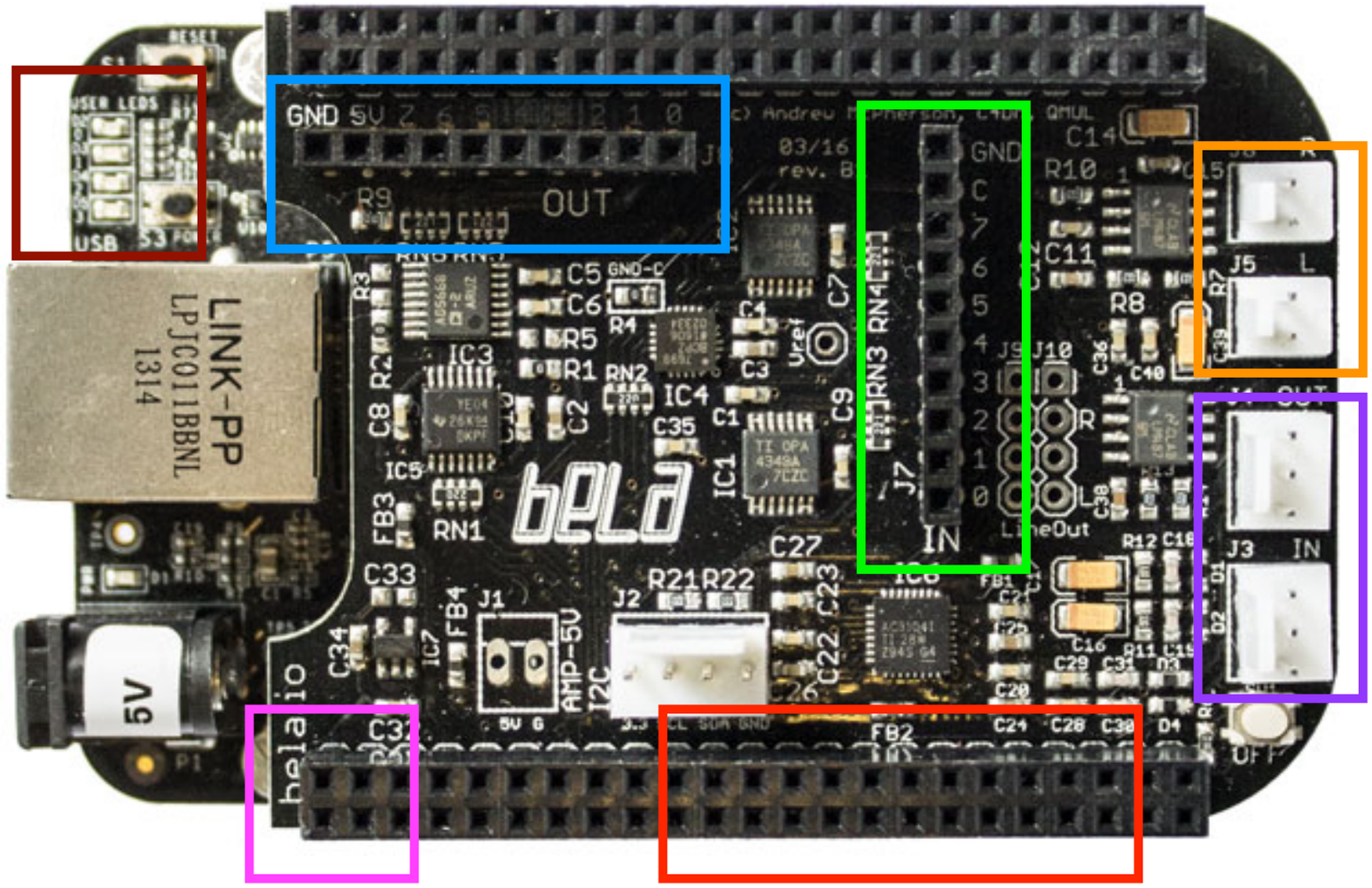


Bela hardware

Analog outputs

Analog inputs

USB



Stereo speaker amps

Stereo audio in /out

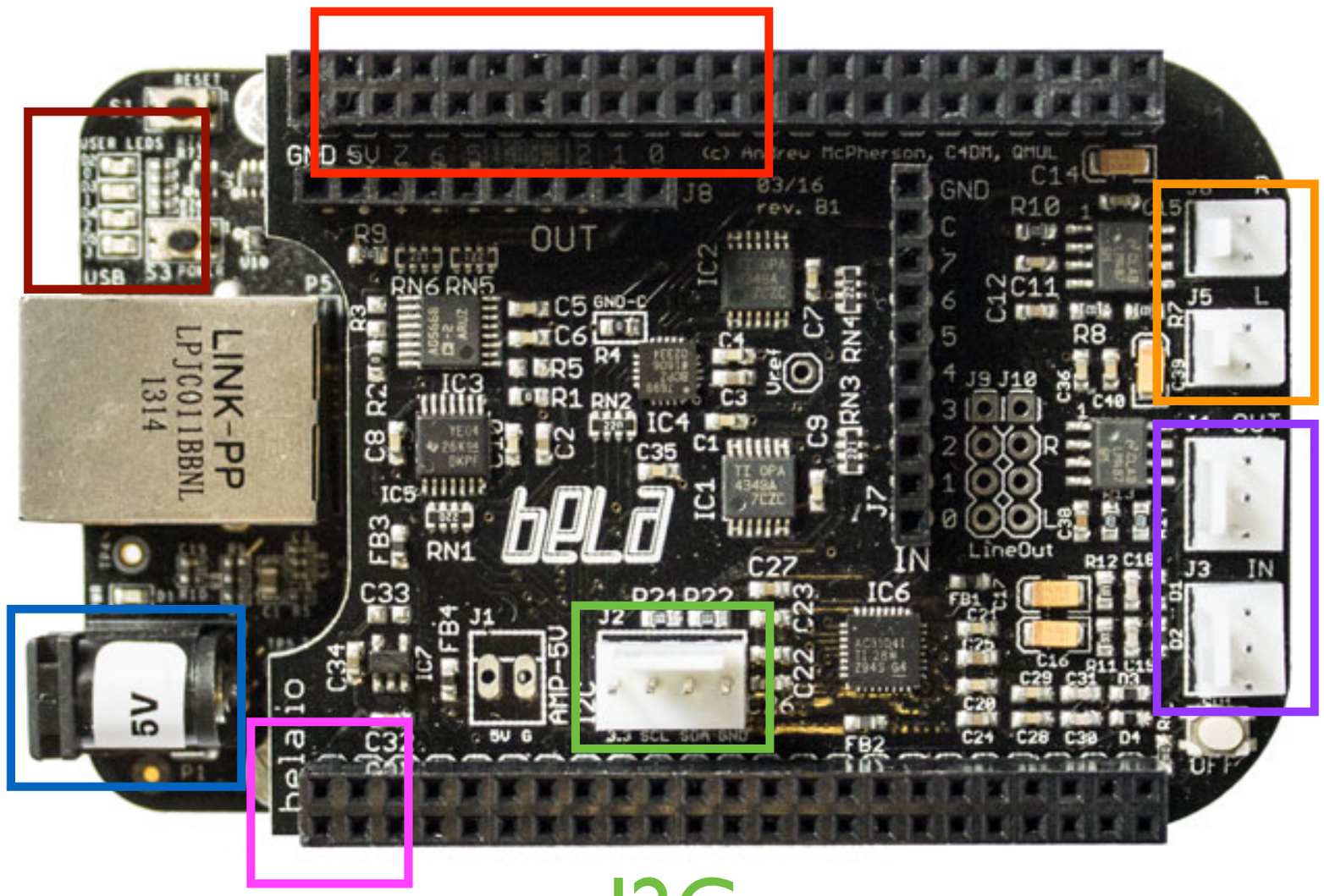
Power / Ground

Digital inputs/outputs

Bela hardware

Digital inputs/outputs

USB



Stereo speaker amps

Stereo audio in /out

5V

Power / Ground

I2C

Required software

Firefox/Chrome browser (*recommended)

The IDE may or may not work in other browsers.

Getting started

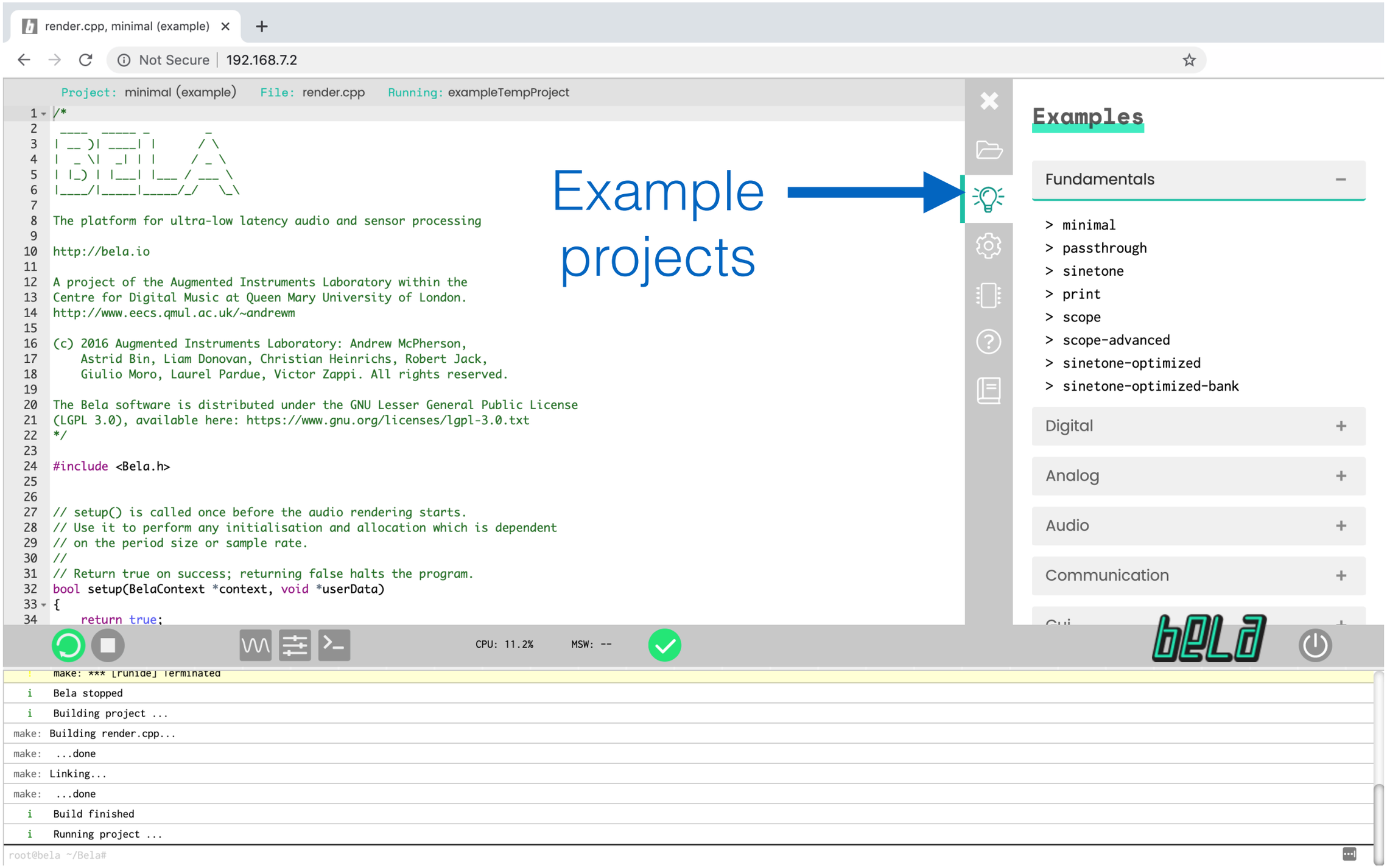
1. Plug in Bela to computer (USB)
2. Launch the browser
3. After around 30s go to the url:

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)

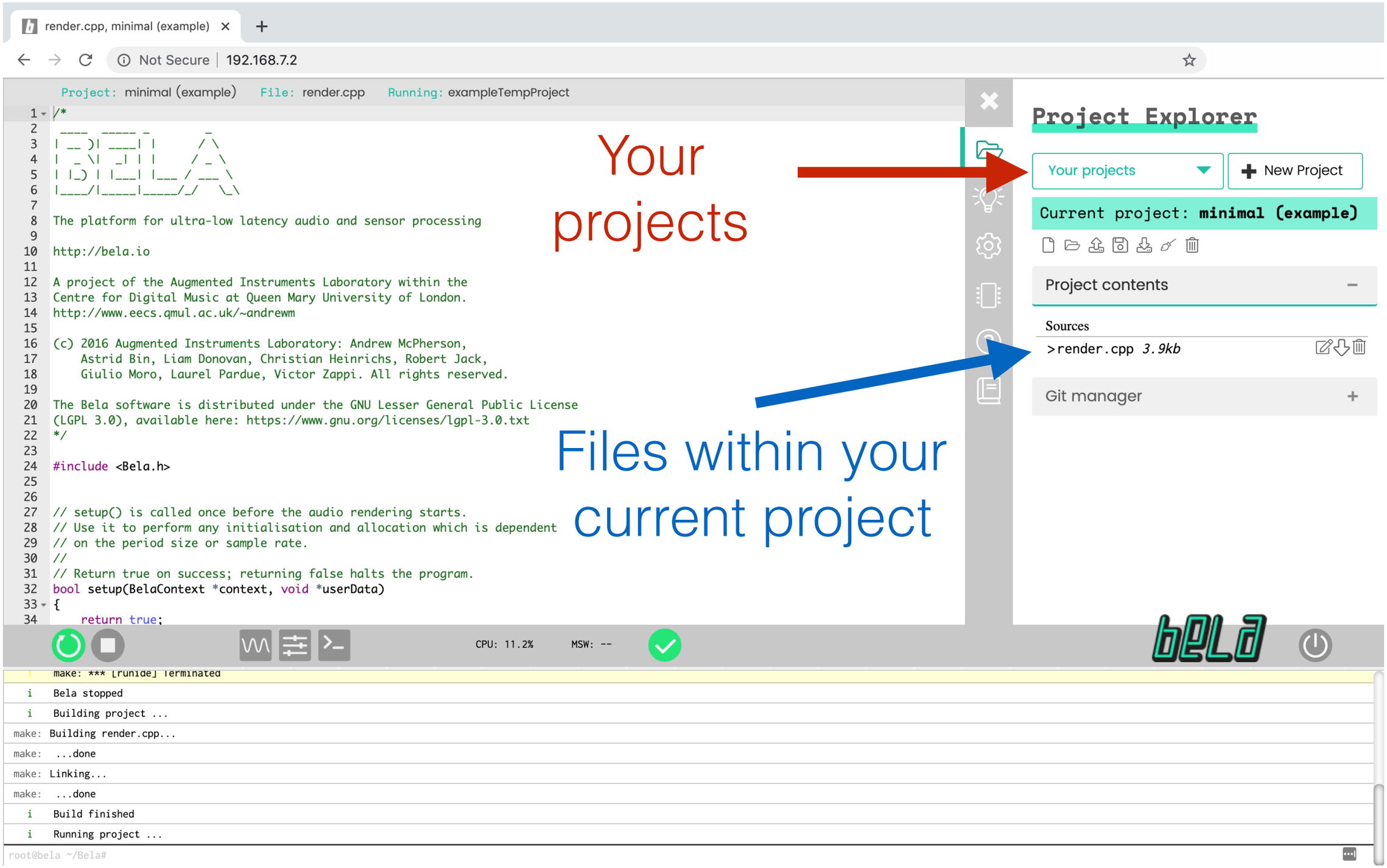
Bring up the Bela IDE:

http://bela.local/



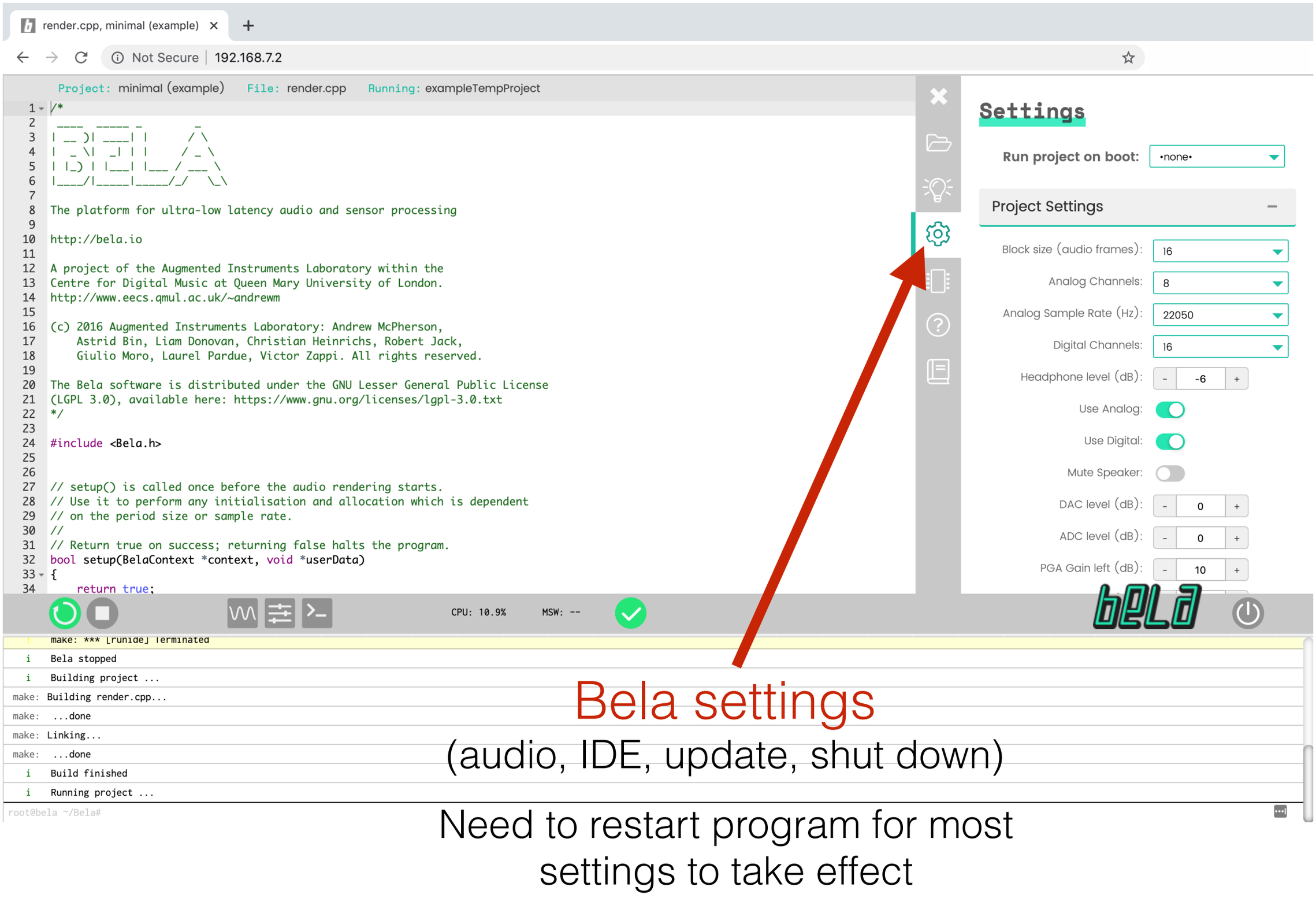
Bring up the Bela IDE:

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/

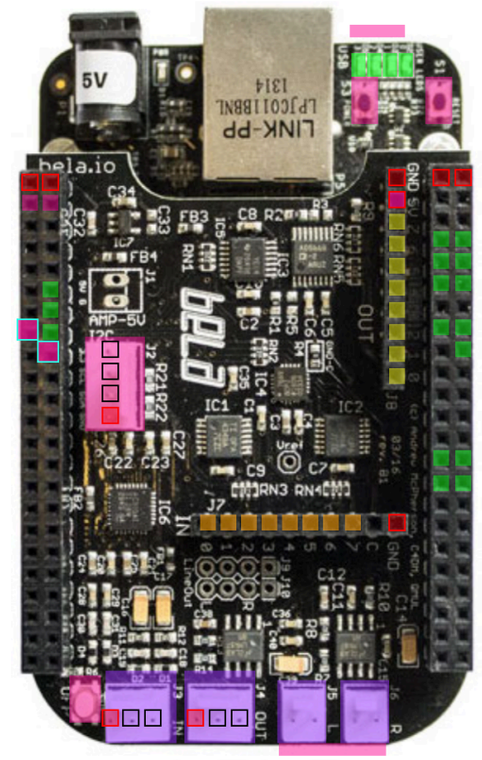
render.cpp, minimal (example)

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

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

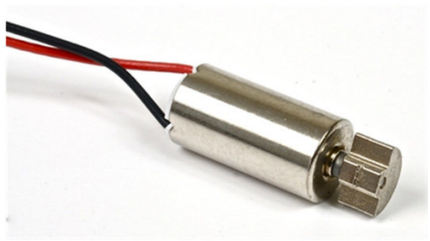
write to buffer of interleaved audio data, specifying a frame, a channel and a value

Keep everything on your breadboard!

We will be using all sensors through the workshop

Haptic Actuators

Eccentric Rotating Mass (ERM)



Linear Resonant Actuators (LRA)

Brushless DC Motors

Piezo Haptic Actuators

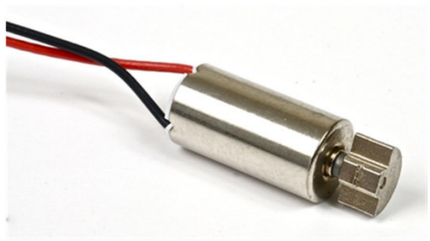
Audio/Tactile Transducers

(drivers)

Solenoids

Haptic Actuators

Eccentric Rotating Mass (ERM)



Linear Resonant Actuators (LRA)

Brushless DC Motors

Piezo Haptic Actuators

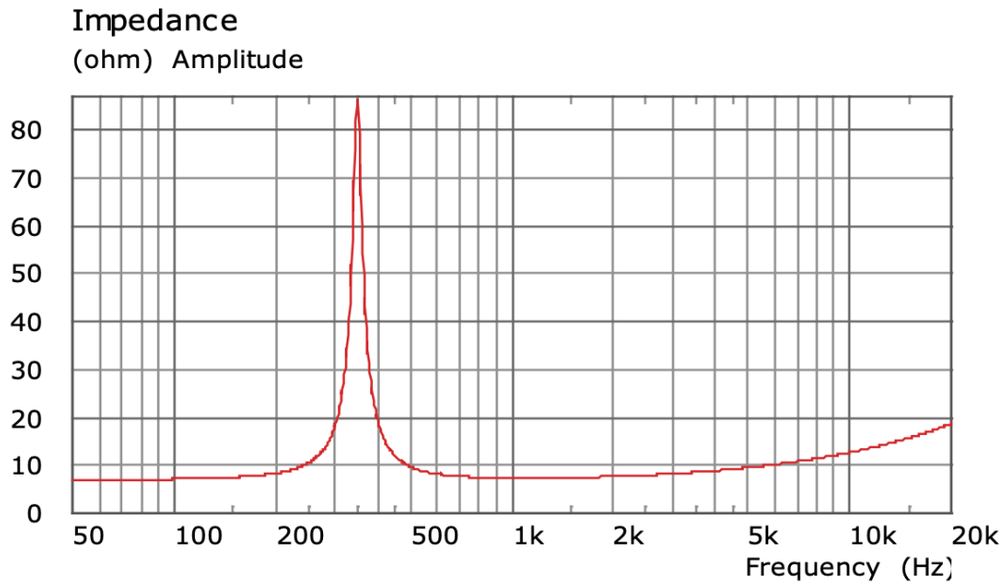
Audio/Tactile Transducers

(drivers)

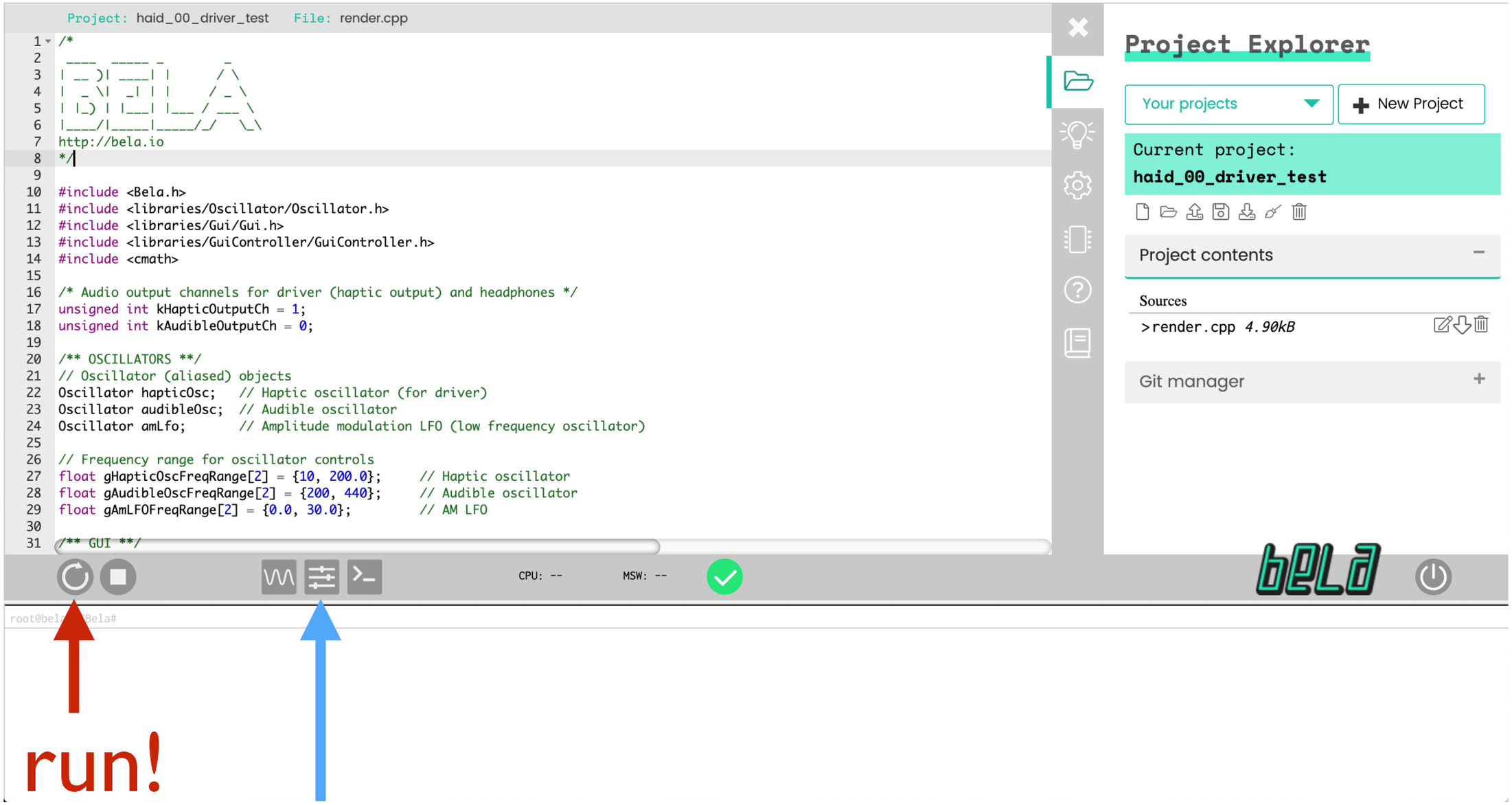
Solenoids

Vibro-tactile actuator

Audible Range 20Hz - 20kHz

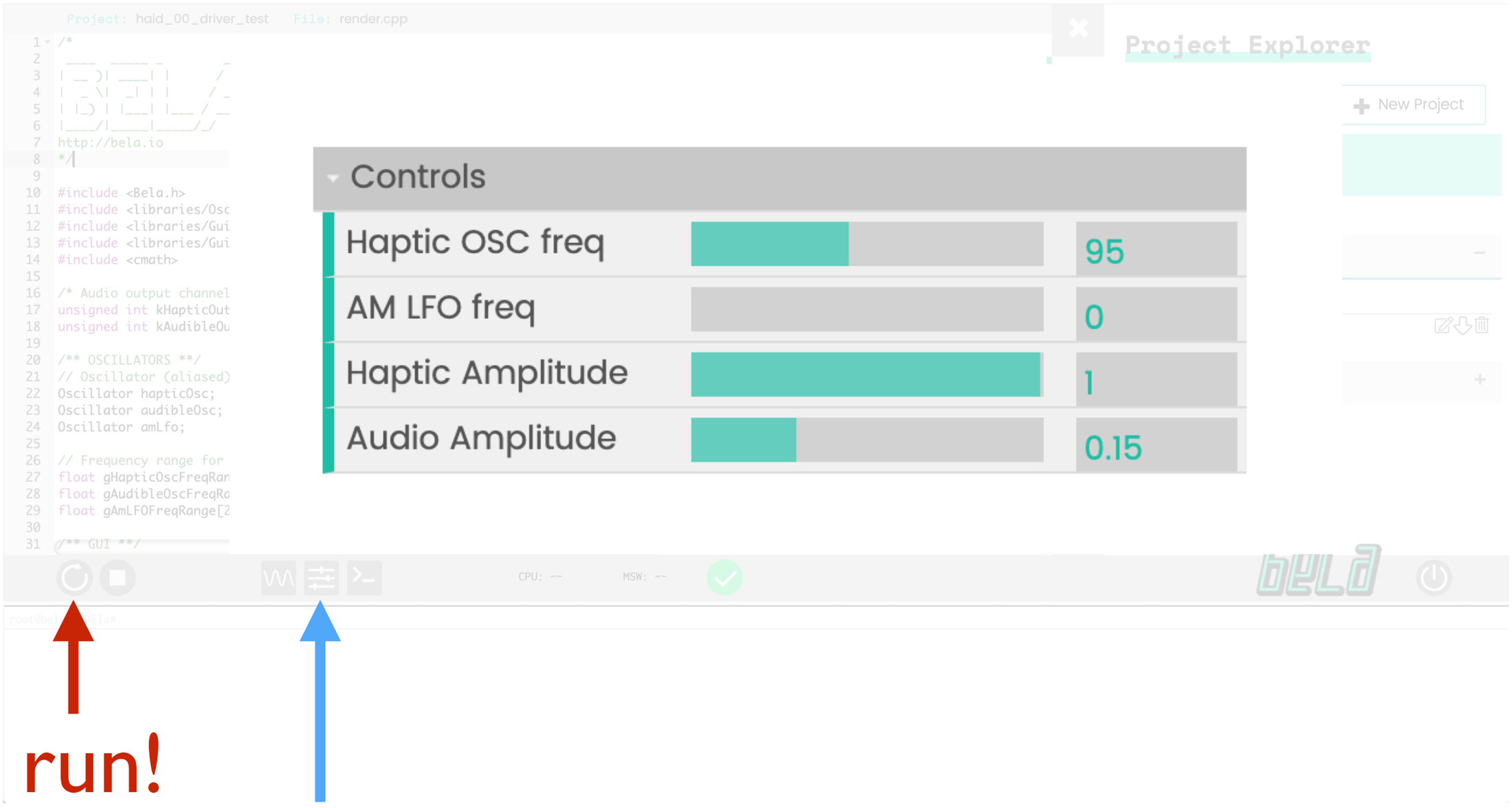


Vibro-tactile actuator



haid_00_driver_test

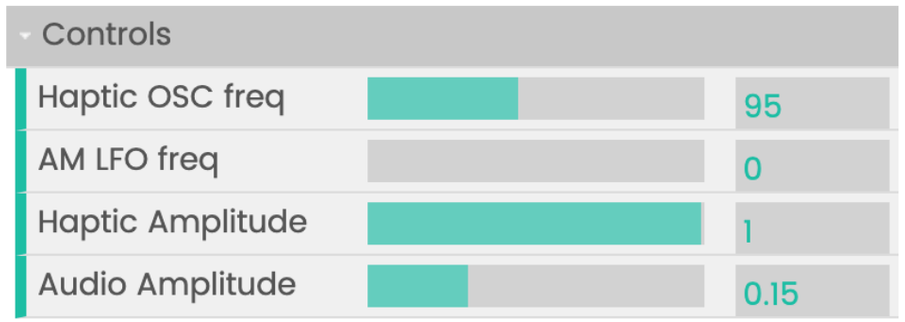
Vibro-tactile actuator



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 gHapticOscFreqSliderId;    // 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
```

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:

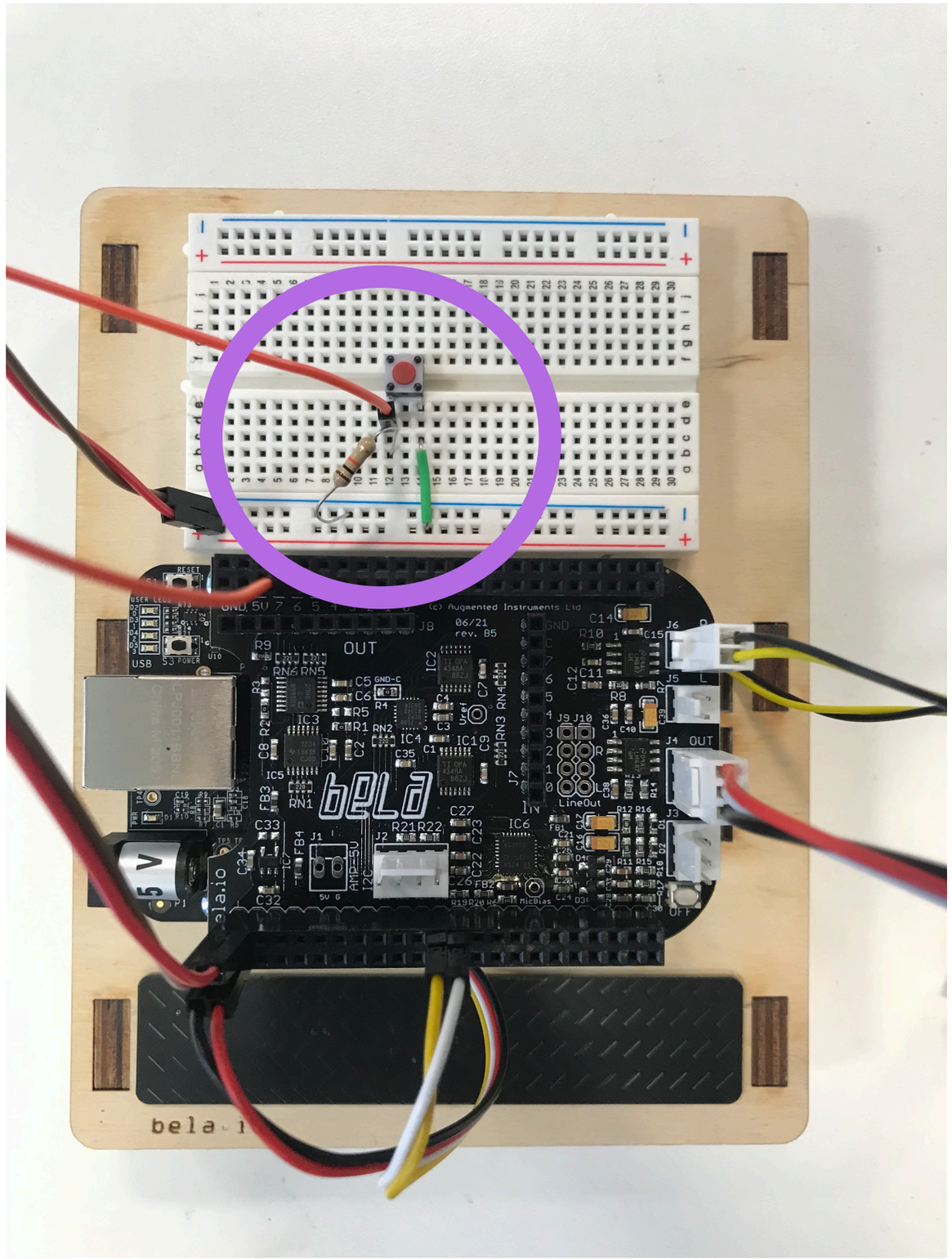
```
76    float hapticAmplitude = controller.getSliderValue(gHapticAmplitudeSliderId);    // Amplitude of Haptic output
77    float audibleAmplitude = controller.getSliderValue(gAudibleAmplitudeSliderId);      // Amplitude of Audible output
```

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);
```

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;
```

setup:

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

render:

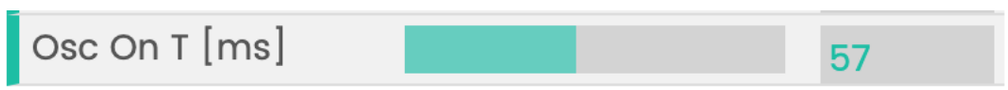
```
110     /* Frame loop (audio rate) */
111     for(unsigned int n = 0; n < context->audioFrames; n++) {
112
113         /** DIGITAL **/
114         // Read the state of the button
115         int buttonState = digitalRead(context, n, kButtonInput);

117         // If button pressed...
118         if(buttonState == 1)
119             gOscToggle = true;
120         // If button released...
121         else
122             gOscToggle = false;
```

haid_01_button

Button

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

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



[Task 1]: A

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

[Task 1]: B

```
118     // If count has elapsed...
119     if(gOscOnCount >= 0)
120     {
121         gOscToggle = true; // Toggle oscillator on
122         gOscOnCount--;  // Decrement counter
123     }
124     // otherwise ...
125     else
126     {
127         gOscToggle = 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(gOscOnTimeRange[0]*0.001, gOscOnTimeRange[0]*0.001, context->audioSampleRate);
```

render:

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

```
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     }
```

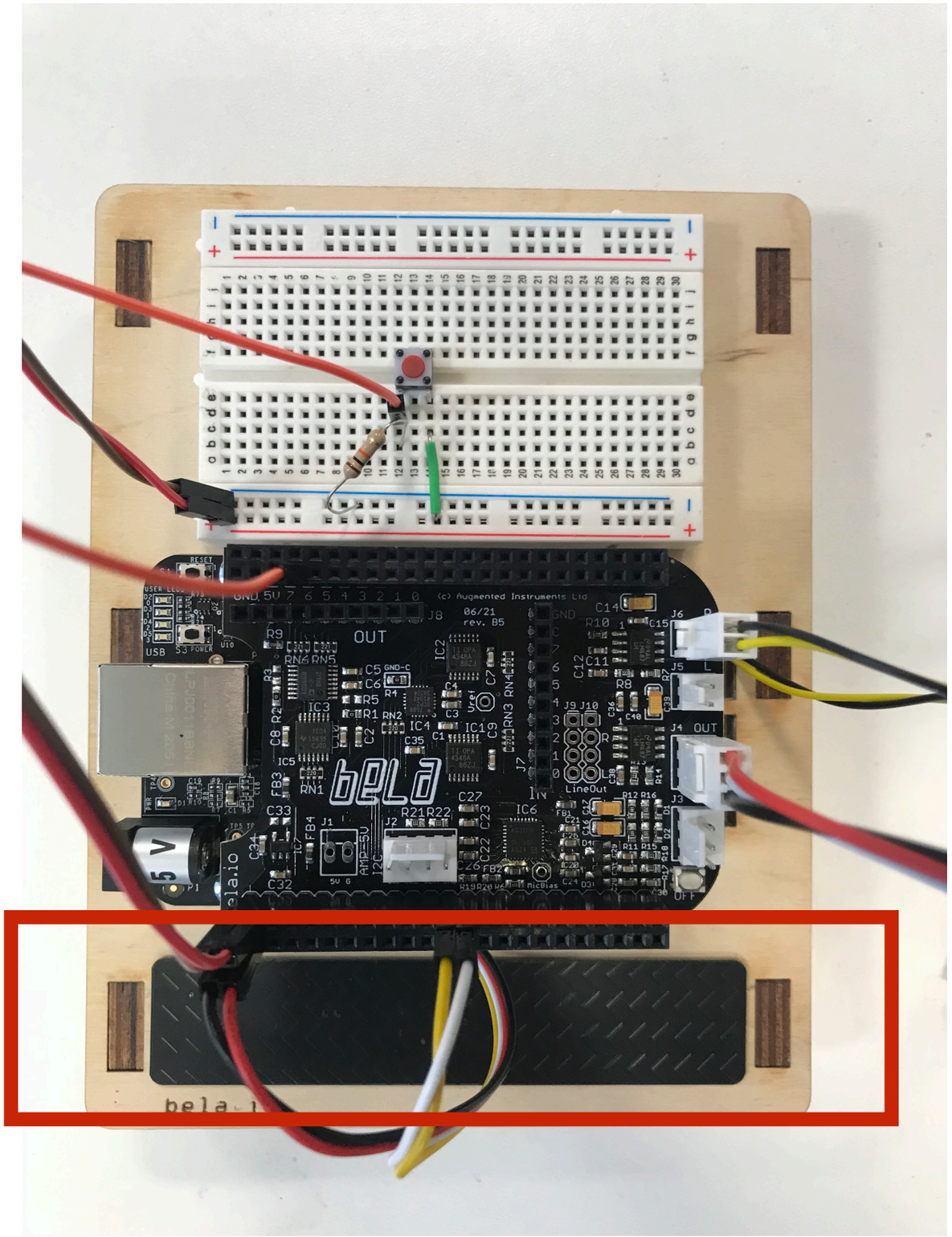
```
138     /** AUDIO **/
139     // Compute output by scaling with the amplitude
140     float envelope = asrEnvelope.process();
141     float hapticOut = hapticOsc.process() * hapticAmplitude * envelope;      // Haptic output
142     float audibleOut = audibleOsc.process() * audibleAmplitude * envelope;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

haid_01_button_C

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 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 (A)

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

[Task 1]

Hint! - *gNumActiveTouches*

```
154         // If there is a new touch
155         if(gNumActiveTouches != gPrevNumActiveTouches)
156         {
157             gOscOnCount = (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             gOscOnCount = (int)(context->audioSampleRate * oscOnTimeMs * 0.001); // Reset counter
155
156             // Scale haptic oscillator frequency based on number of touches
157             float hapticOscFreq = map(gNumActiveTouches, 0, NUM_TOUCH, hapticMinOscFreq, gHapticOscFreqRange[1]
158             hapticOsc.setFrequency(hapticOscFreq);  // Set frequency of haptic oscillator
159
160             // Get frequency for audio oscillator by remapping frequency of haptic oscillator to appropriate ra
161             float audibleOscFreq = map(hapticOscFreq, gHapticOscFreqRange[0], gHapticOscFreqRange[1], gAudibleO
```

haid_01_trill_B

Trill Bar (C)

3. Scale vibration frequency based touch position

[Task 3]

```
156         // Scale haptic oscillator frequency based on touch location of first touch
157         float hapticOscFreq = map(gTouchLocation[0], 0, 1, hapticMinOscFreq, gHapticOscFreqRange[1]);
158         hapticOsc.setFrequency(hapticOscFreq);  // Set frequency of haptic oscillator
159
160         // Get frequency for audio oscillator by remapping frequency of haptic oscillator to appropriate range
161         float audibleOscFreq = map(hapticOscFreq, gHapticOscFreqRange[0], gHapticOscFreqRange[1], gAudibleOscFreqRange[0], gAudibleOscFr
162         audibleOsc.setFrequency(audibleOscFreq); // Set frequency of
```

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         gOscToggle = true;
159
160         // Scale haptic oscillator frequency based on touch location of first touch
161         float hapticOscFreq = map(gTouchLocation[0], 0, 1, minHapticOscFreq, maxHapticOscFreq);
162         hapticOsc.setFrequency(hapticOscFreq);  // Set frequency of haptic oscillator
163
164         // Get frequency for audio oscillator by remapping frequency of haptic oscillator to appropriate range
165         float audibleOscFreq = map(hapticOscFreq, gHapticOscFreqRange[0], gHapticOscFreqRange[1], gAudibleOscFreqRange
166         audibleOsc.setFrequency(audibleOscFreq); // Set frequency of
167
168     }
169     else
170     {
171         gOscToggle = false;
172     }
173
```

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 hapticOscFreq = map(gTouchLocation[0], 0, 1, minHapticOscFreq, maxHapticOscFreq);
164         hapticOsc.setFrequency(hapticOscFreq);  // Set frequency of haptic oscillator
165
166         // Get frequency for audio oscillator by remapping frequency of haptic oscillator to appropriate ra
167         float audibleOscFreq = map(hapticOscFreq, gHapticOscFreqRange[0], gHapticOscFreqRange[1], gAudibleO
168         audibleOsc.setFrequency(audibleOscFreq); // 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         gOscToggle = false;
176     }
177
```

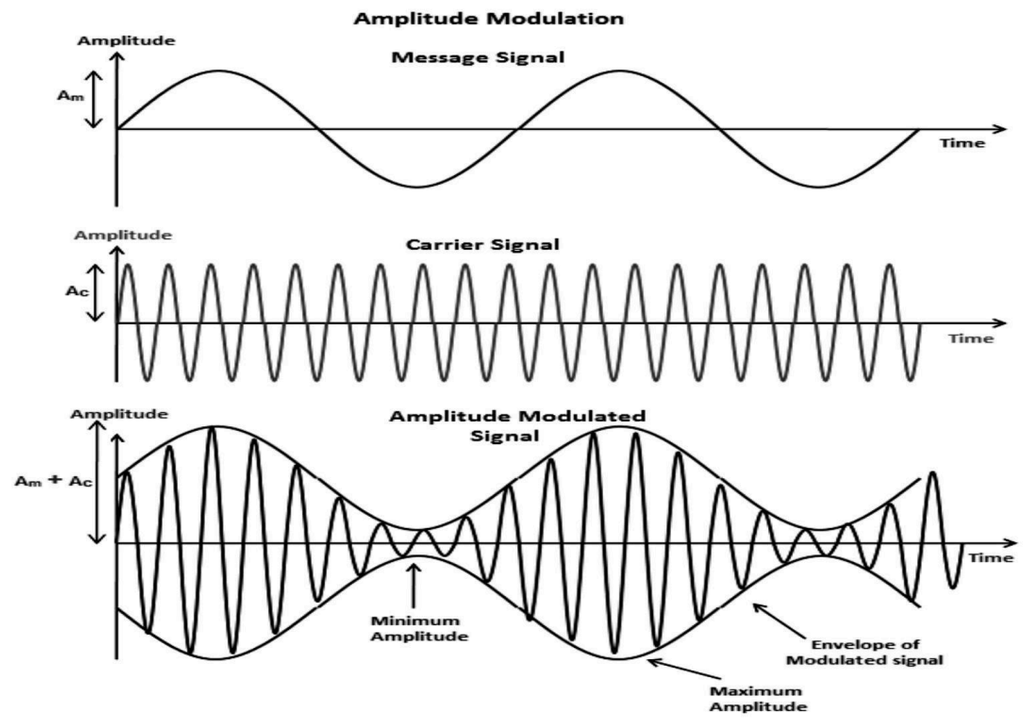
haid_01_trill_D2

Trill Bar (E)

Amplitude Modulation

Second Oscillator (low frequency)

Scale and multiply gain

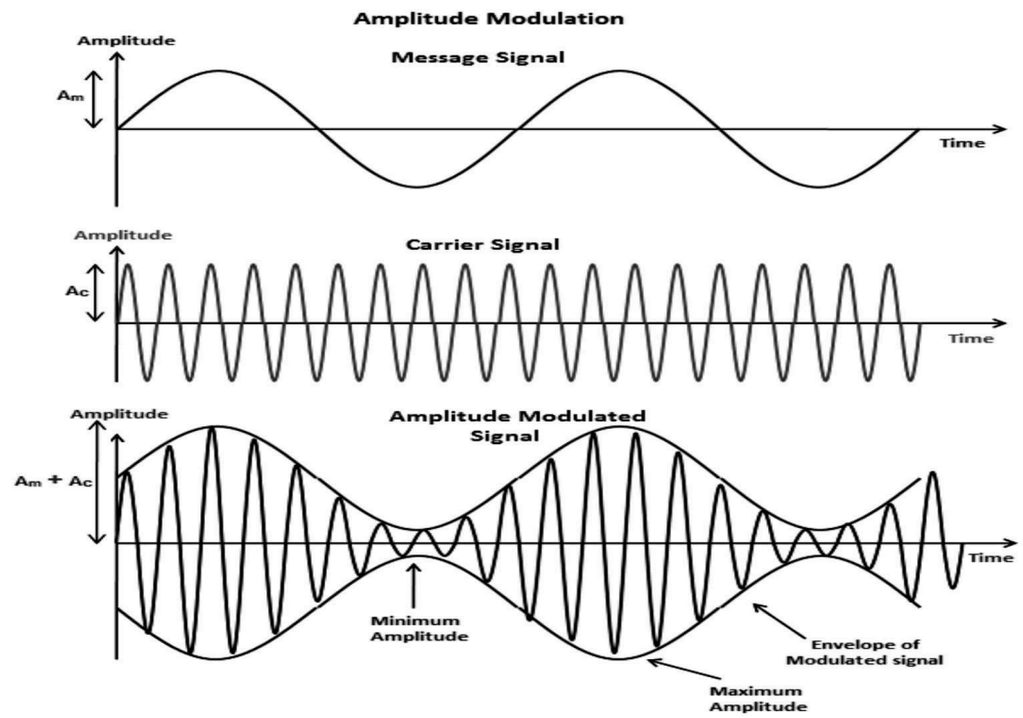



```
175     // Calculate Amplitude Modulation
176     float amLfoVal;
177     if(amLfoFreq == 0.0)    // If frequency is zero...
178         amLfoVal = 1.0;     // .. no amplitude modulation
179     else                    // If frequency is not zero...
180         amLfoVal = 0.5 * (amLfo.process() +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

haid_01_trill_E

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 otput by scaling with the gain and including AM modulation
186     float hapticOut = hapticOsc.process() * hapticAmplitude  * amLfoVal;        // Haptic output
187     float audibleOut = audibleOsc.process() * audibleAmplitude  * amLfoVal; // Audio output
```

haid_01_trill_E

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 audibleOscFreq = map(gTouchLocation[0], 0, 1, gAudibleOscFreqRange[0], gAudibleOscFreqRange[1]);
171             audibleOsc.setFrequency(audibleOscFreq); // Set frequency of audible oscillator
172
173             // Calculate absolute difference to target Frequency
174             float freqDiff = fabs(gTargetAudibleFreq - audibleOscFreq);
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/

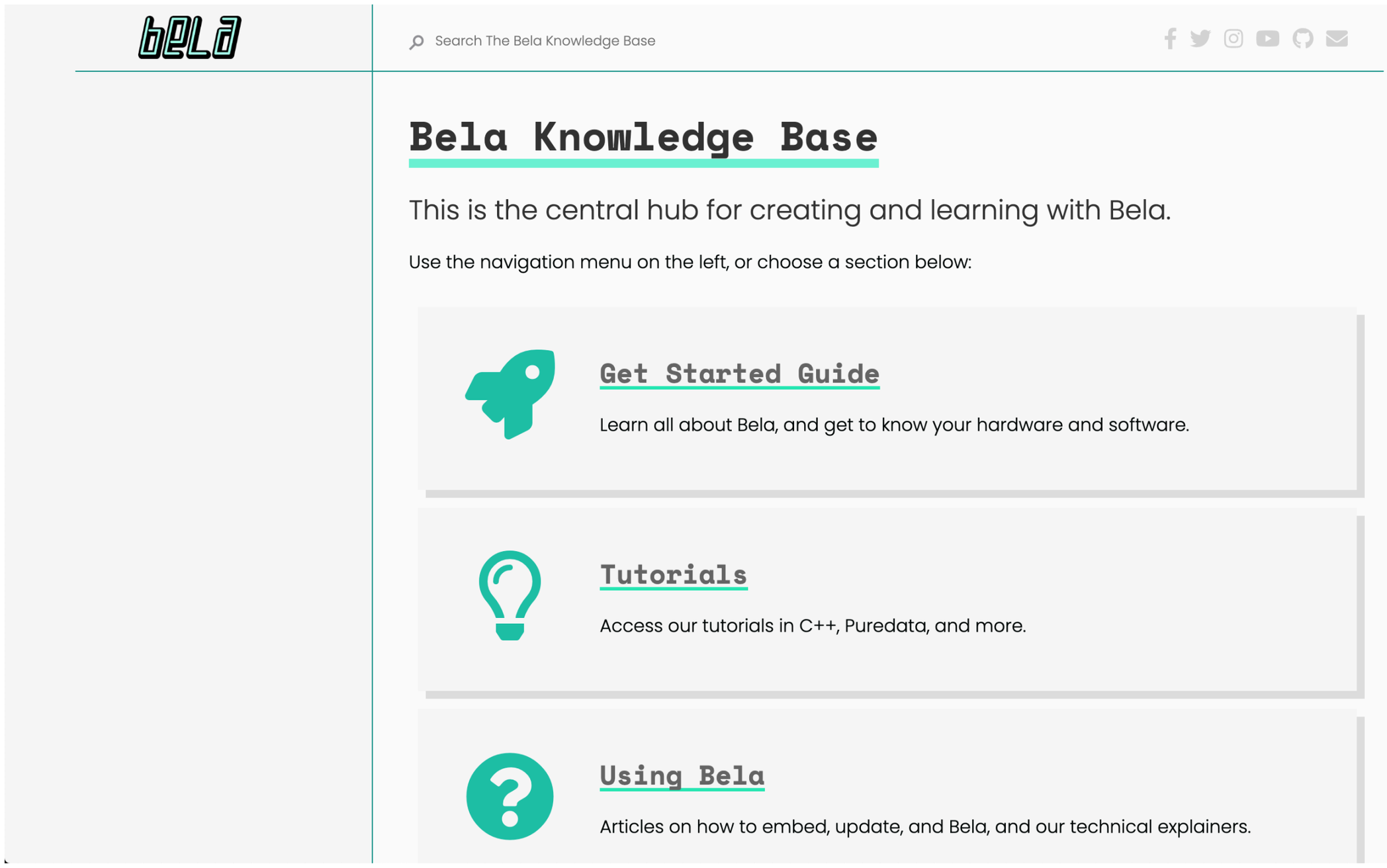
https://blog.bela.io/pandoras-box-ward-slager/

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

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

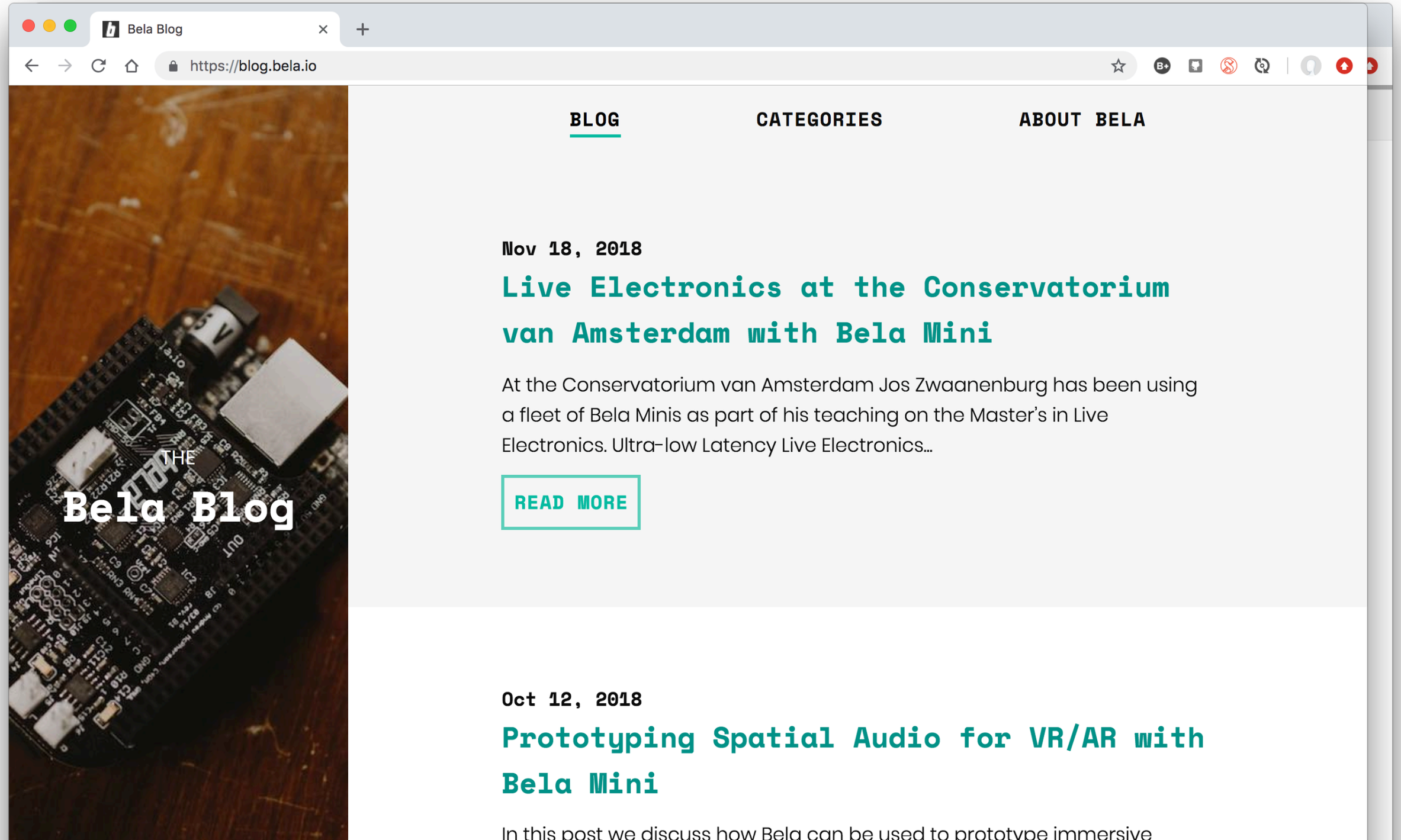
Finding out more

http://learn.bela.io



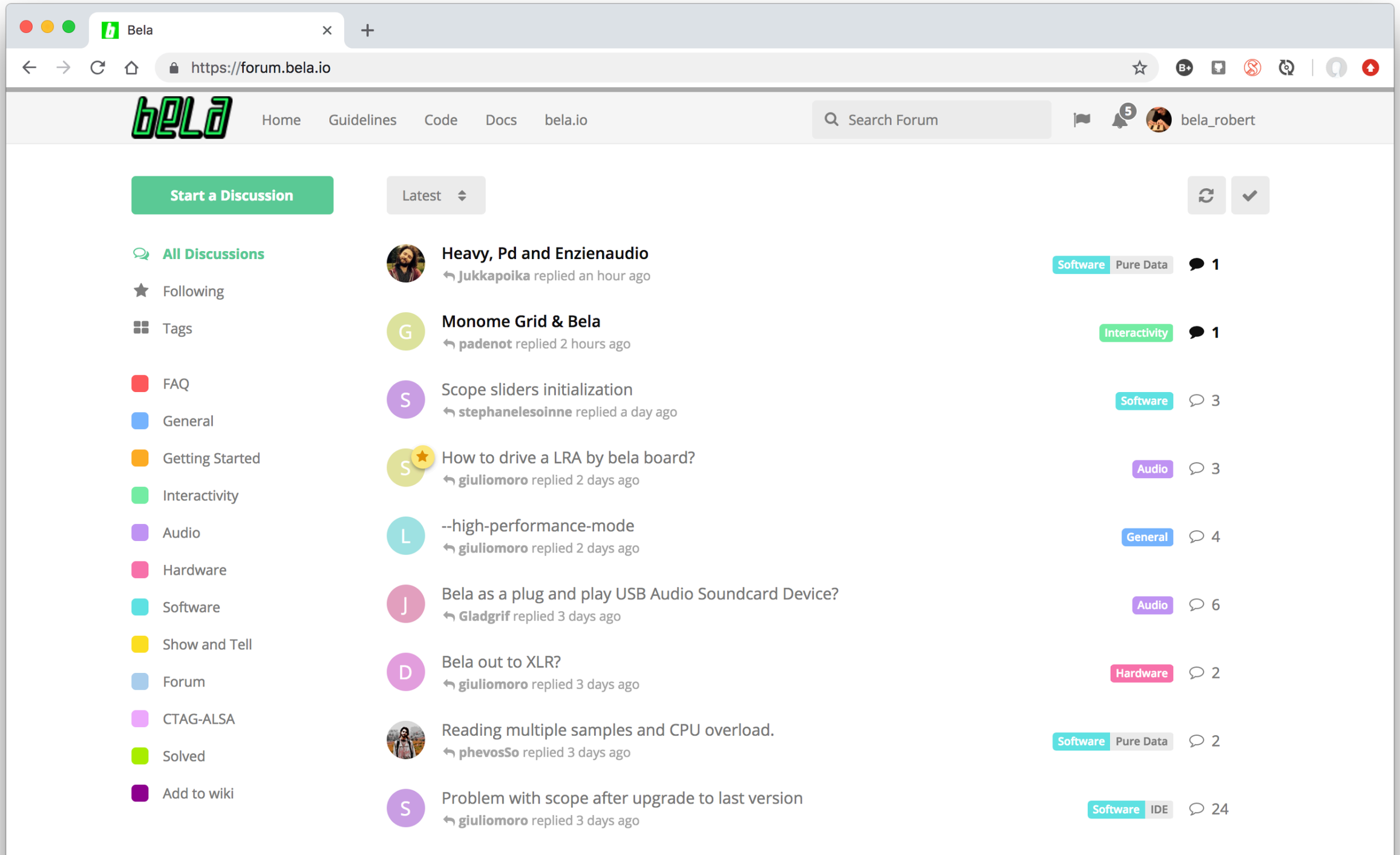
Finding out more

http://blog.bela.io



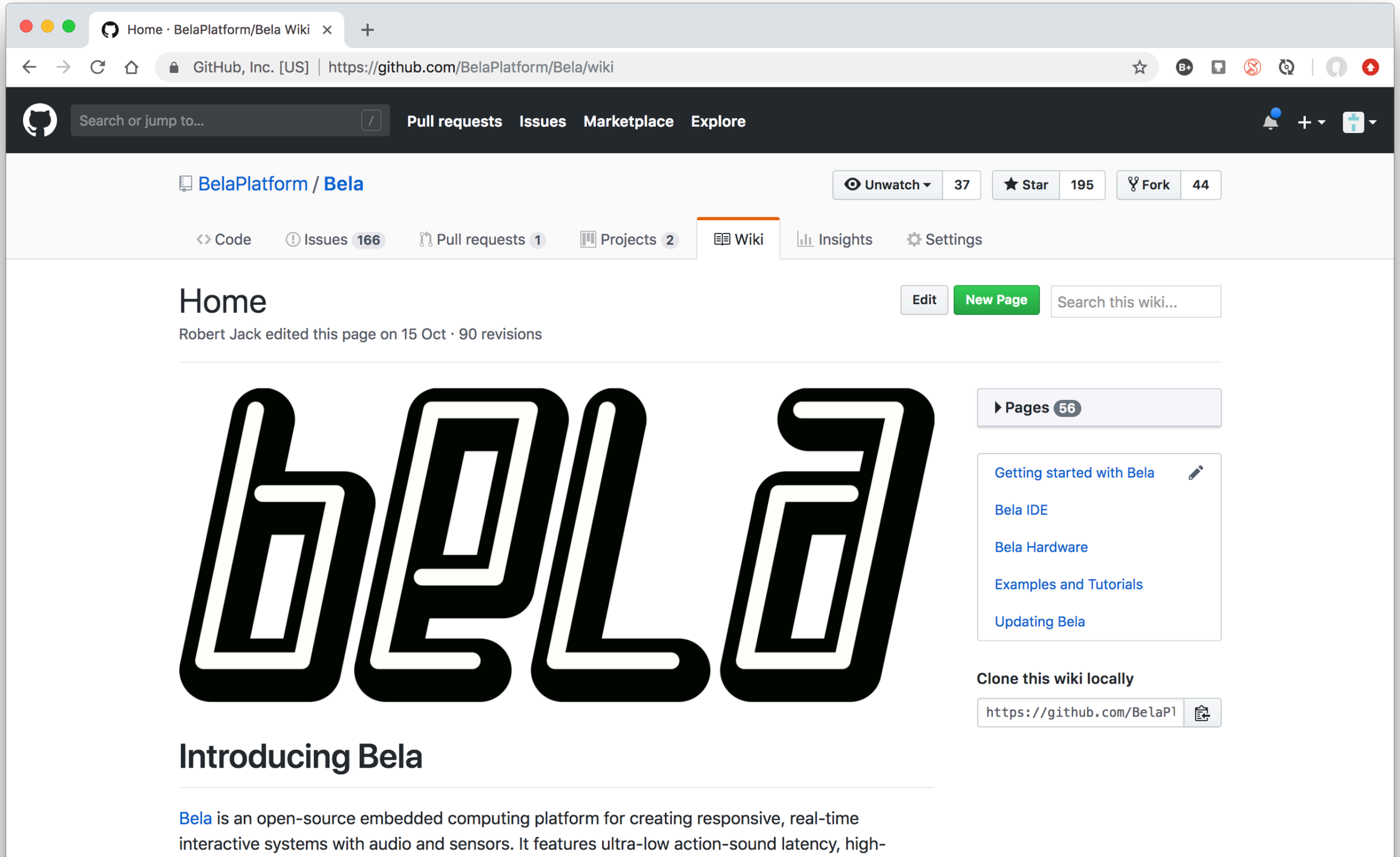
Finding out more

http://forum.bela.io



Finding out more

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



Where can I get one?

https://shop.bela.io/