

Materials in Music

Design Workshop

Jianing Zheng, Sophie Skach, Adan L. Benito

Workshop Overview

- Introduction
- Sharing session
- Introduction to Interactive Textiles
- Introducing Bela and Pure Data
- Prototyping and Making - Building sensor and connecting the sensor to Bela
- Sharing and Wrap-up

Textile Materials

- Non-conductive materials:
 - cotton, wool, linen, silk, ...
 - nylon, polyester, acrylic, ...
 -
- Conductive materials:
 - copper, silver, steel, iron, ...
 - carbon, graphene, ...

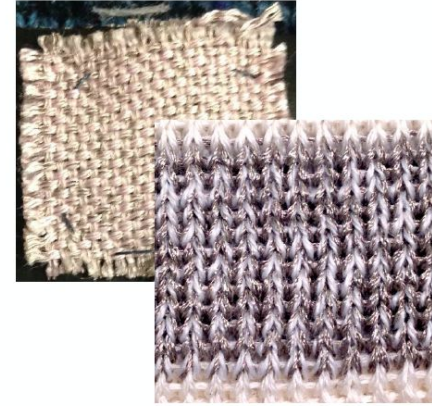
Textile Structure



fibre / filament



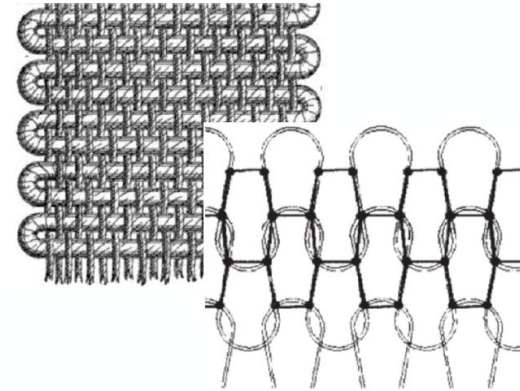
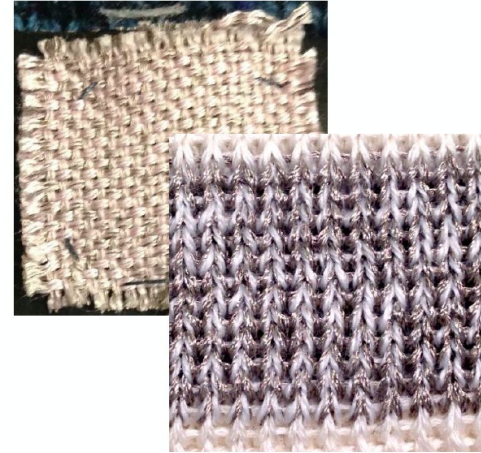
yarn / thread



fabric / textile surface

Textile Structure

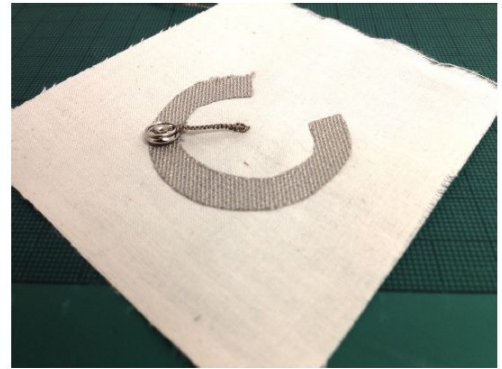
- weaving
- knitting
- felting
- braiding
- crocheting
- embroidering
- ...



Textile Sensors

Examples

- Switches
- Capacitive sensors
- Piezo resistive sensors (pressure, stretch)
- Piezo electric sensors
- Textile speakers
- Antennas



Applications

Examples in music



Mazi, collaborative sound interface
(Antonella Nonnis)



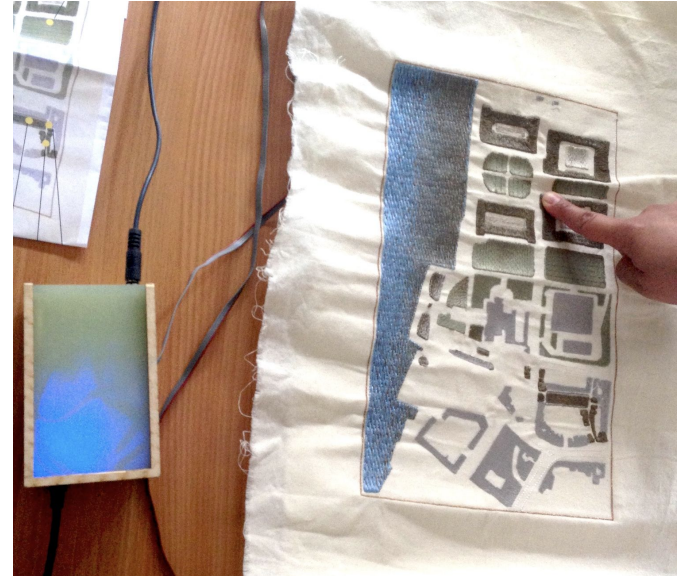
Soft Knit Biosensing for Augmenting Vocal Performances
(Courtney N. Reed, Sophie Skach, et al)

Applications

Examples in music



Text and Iles (Sophie Skach)



Aural Fabric (Alessia Milo)

Applications

Examples in music



KnittedKeyboard II

(Irmandy Wicaksono, MIT Media Lab)

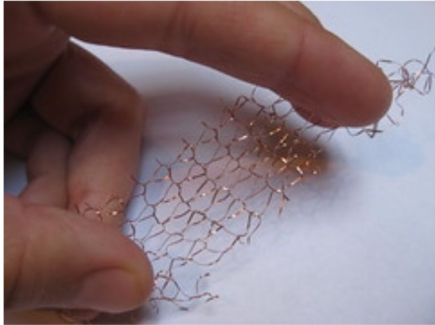
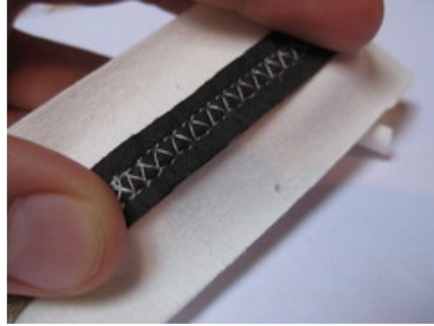


Data Gloves

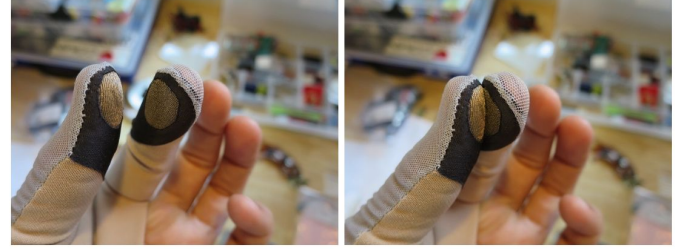
(Hannah Perner Wilson, Mika Satomi, KOBAKANT)

Examples

Stretch Sensor and switch



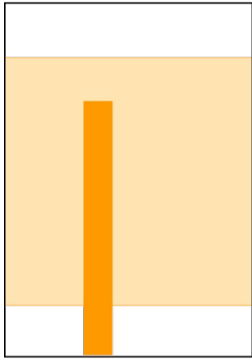
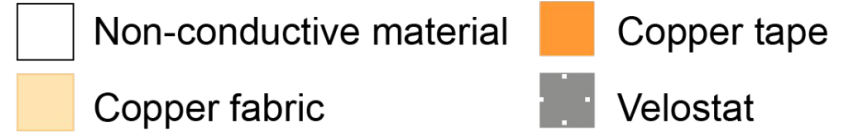
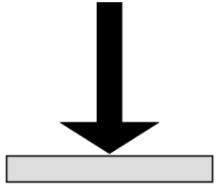
Constructed stretch sensors,
by KOBAKANT DIY Wearable Technology
Documentation



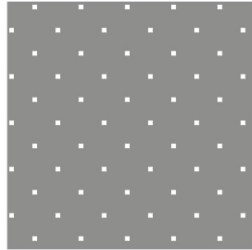
Fingertip contact switch,
by KOBAKANT DIY Wearable Technology
Documentation

Pressure Sensor

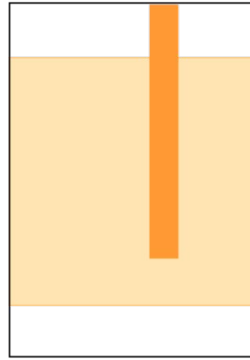
Build a pressure sensor



Top layer



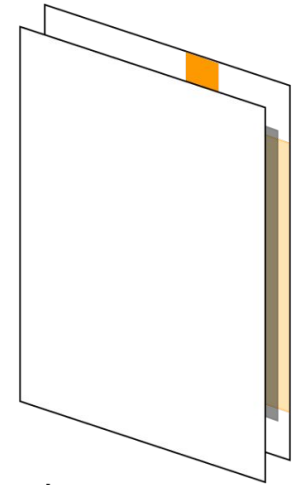
Mid layer



Bottom layer



Overall look



Useful websites

- KOBAKANT. Hannah Perner Wilson and Mika Satomi.
<https://www.kobakant.at/DIY/>
- E-Textile Summercamp Swatch Exchange:
<https://etextile-summercamp.org/swatch-exchange/>
- Data Palette: <https://datapalette.org/>
- Bela Blog:
<https://blog.bela.io/e-textiles-and-embodied-interaction-with-bela/>
- Embelashed Paper Sensors:
<https://embelashed.org/paper/paper-index.html#paper-sensors>

Bela

<https://bela.io/>



Making music with e-textiles
(Nicola Woodham)



Cutting-edge e-textile research with Bela
(Sophie Skach, Rebecca Stewart, et al)

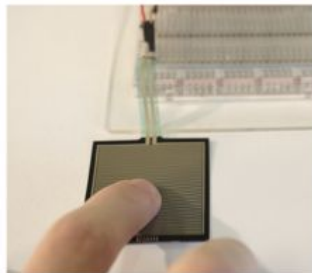
What is Bela?

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



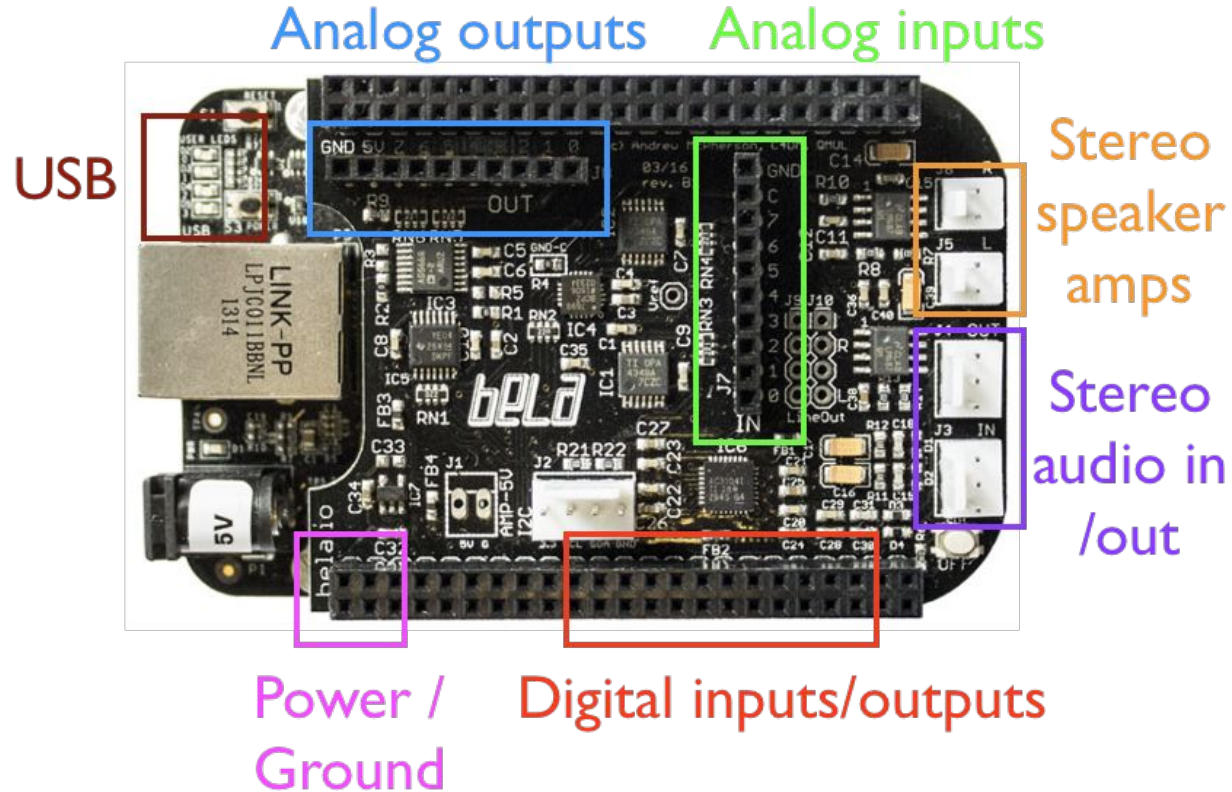
- Analog, digital I/O sampled at audio rate
- Ultra low action-sound latency
- Jitter-free alignment between audio and sensors

Open-source
maker
platform

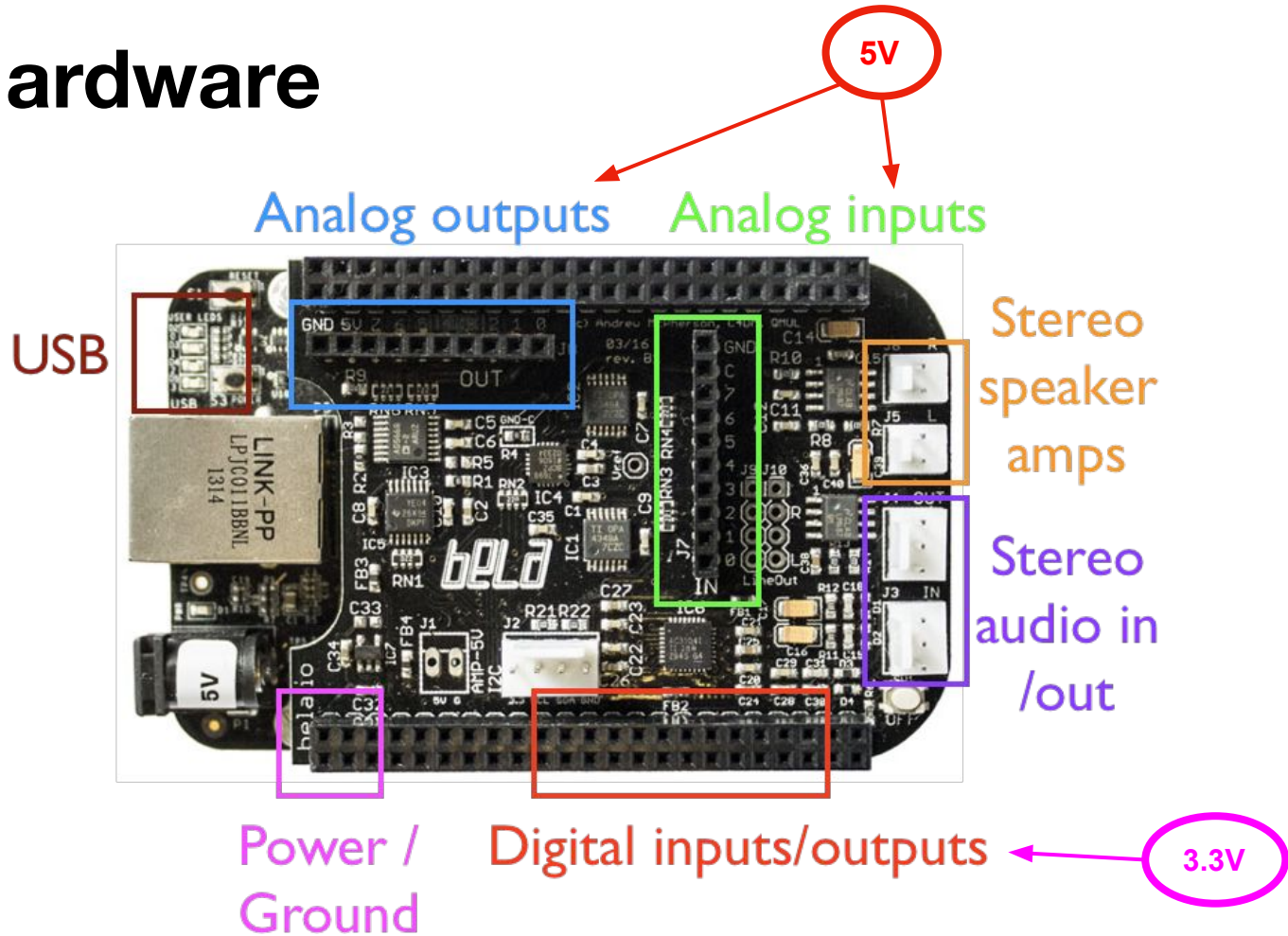


- Open hardware and software
- Targeted at musicians, artists
- Online community resources: forum, wiki, blog.

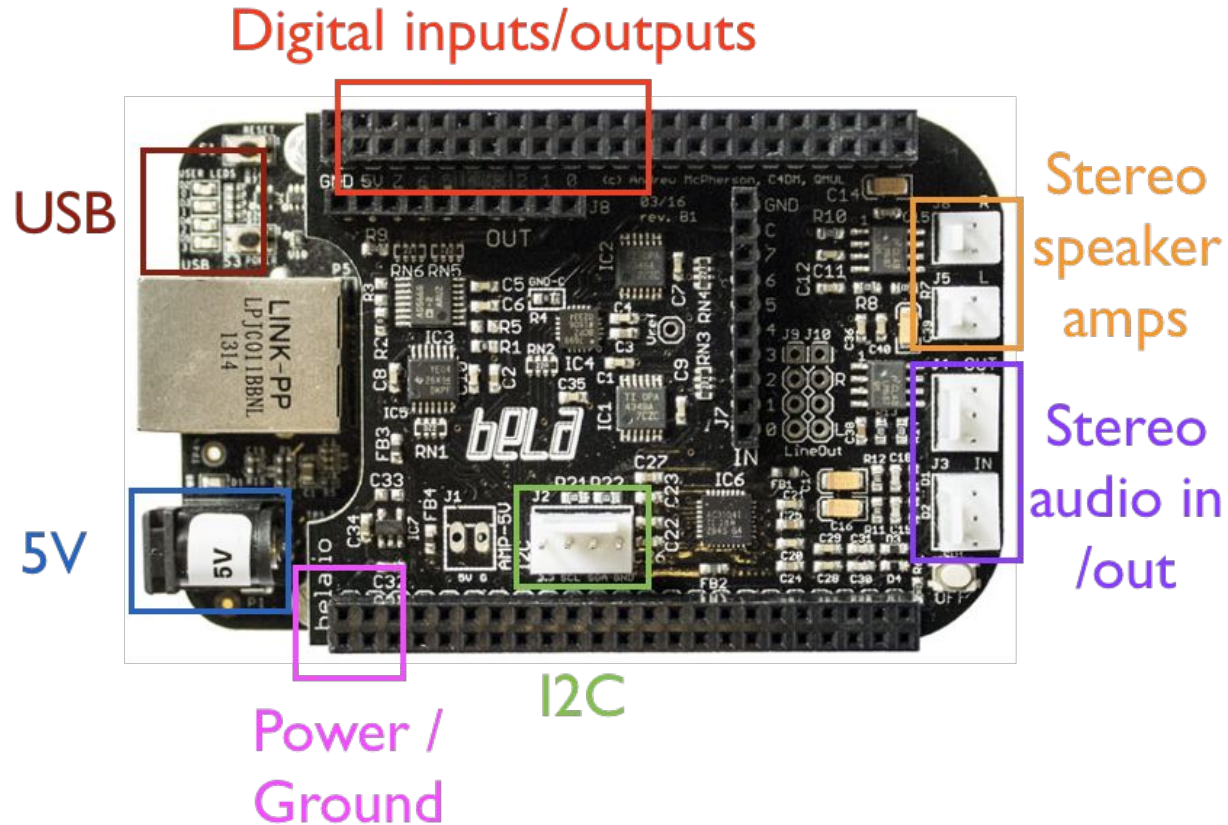
Bela Hardware



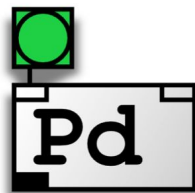
Bela Hardware



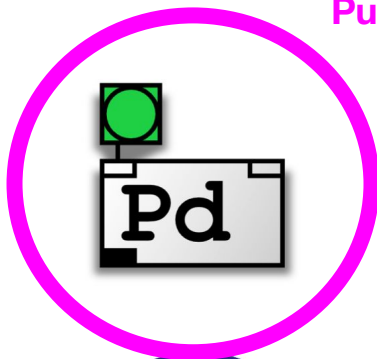
Bela Hardware



Bela is polyglot



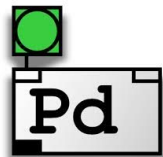
Bela is polyglot



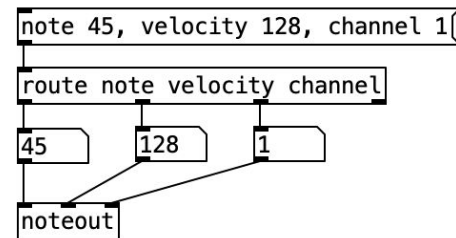
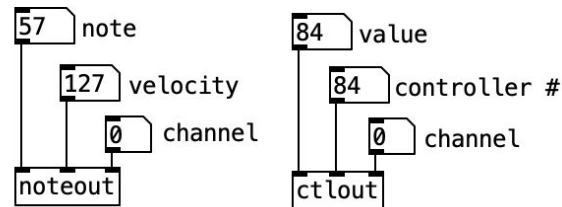
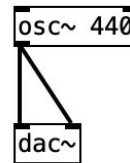
Pure Data



Pure Data

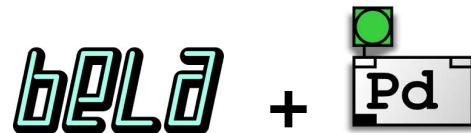


- Free open-source programming language
- Visual environment -> dataflow ('patching')
- Powerful, quick -> good for prototyping
- Several flavours: *Vanilla*, Extended, PurrData
- Objects, messages and signals -> event-driven vs realtime



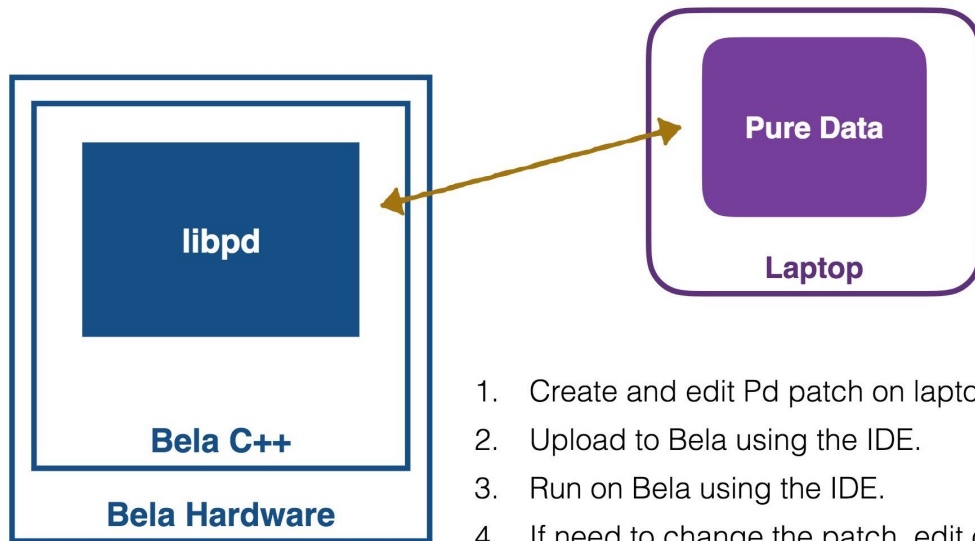
<https://puredata.info/downloads>

Bela & Pure Data



libpd -> library wrapper -> Pd vanilla DSP functionality -> headless!

Workflow:



1. Create and edit Pd patch on laptop.
2. Upload to Bela using the IDE.
3. Run on Bela using the IDE.
4. If need to change the patch, edit on laptop and repeat process.

The main Pd patch of your project must always be named `_main.pd`

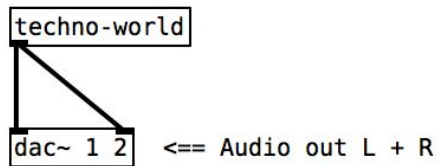
Bela & Pure Data



Audio I/O

[audio rate]

- Stereo audio input: [adc~ 1 2]
- Stereo audio output: [dac~ 1 2]



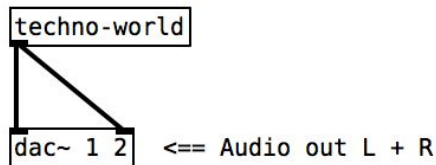
Bela & Pure Data



Audio I/O

[audio rate]

- Stereo audio input: [adc~ 1 2]
- Stereo audio output: [dac~ 1 2]



Analog I/O

- Analog inputs: [adc~ 3 4 5 6 7 8 9 10]
- Analog outputs: [dac~ 3 4 5 6 7 8 9 10]

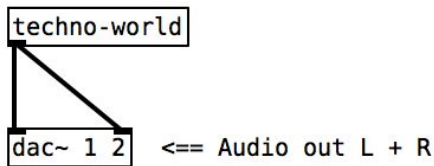
Bela & Pure Data



Audio I/O

[audio rate]

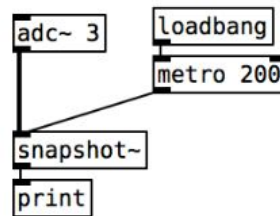
- Stereo audio input: [adc~ 1 2]
- Stereo audio output: [dac~ 1 2]



Analog I/O

[0.5 * audio rate] -> upsampled internally to audio rate

- Analog inputs: [adc~ 3 4 5 6 7 8 9 10]
- Analog outputs: [dac~ 3 4 5 6 7 8 9 10]
- Printing at slower rate:



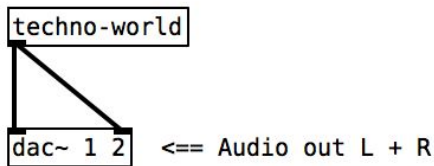
Bela & Pure Data



Audio I/O

[audio rate]

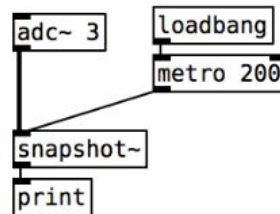
- Stereo audio input: [adc~ 1 2]
- Stereo audio output: [dac~ 1 2]



Analog I/O

[0.5 * audio rate] -> upsampled internally to audio rate

- Analog inputs: [adc~ 3 4 5 6 7 8 9 10]
- Analog outputs: [dac~ 3 4 5 6 7 8 9 10]
- Printing at slower rate:



Scope

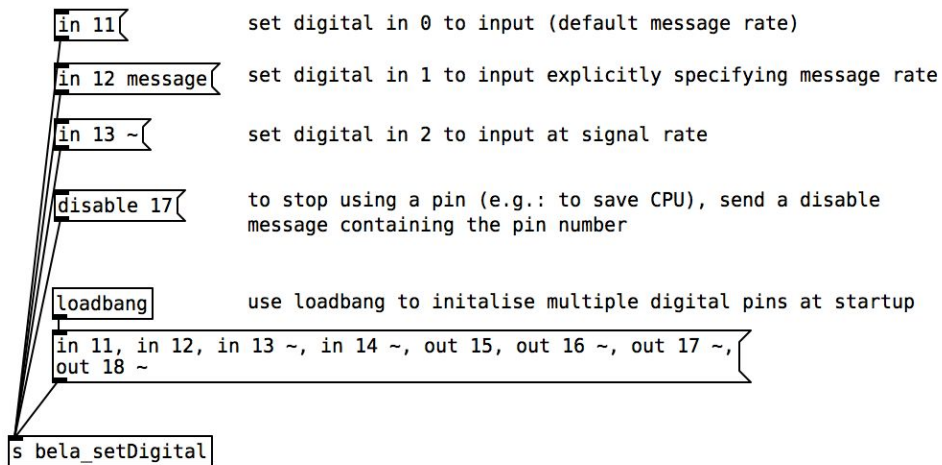
- 4 channels: [dac~ 27 28 29 30] → signals must be audio rate [sig~]

Bela & Pure Data



Digital I/O [audio rate]

- Digital pins 0-16 are numbered **11-26**
- initialised by sending messages to `bela_setDigital`

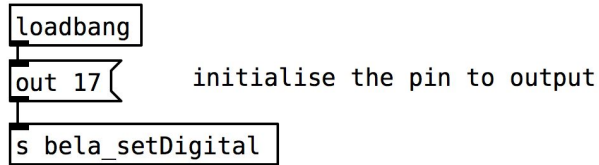


Bela & Pure Data

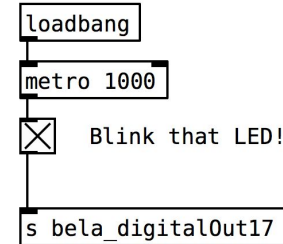


Digital I/O [audio rate]

- *Digital OUT*



Digital out, message rate

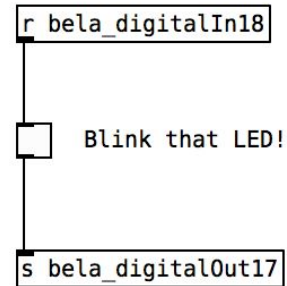
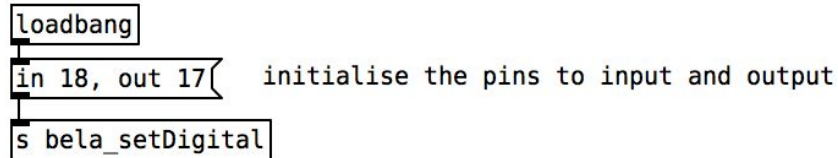


Bela & Pure Data



Digital I/O [audio rate]

- *Digital IN*



Bela & Pure Data



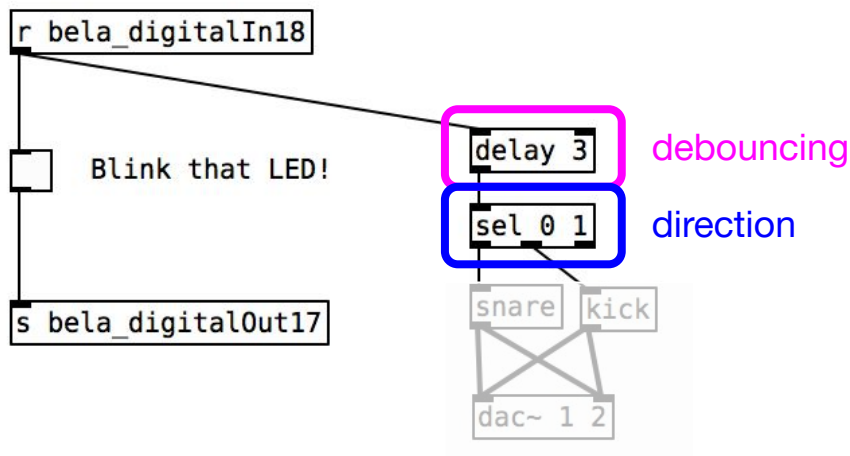
Custom render

- Allows us to write our own C++ code to interact with libpd
- Complex, big file, only part of it can be edited

Bela & Pure Data



Digital input debouncing

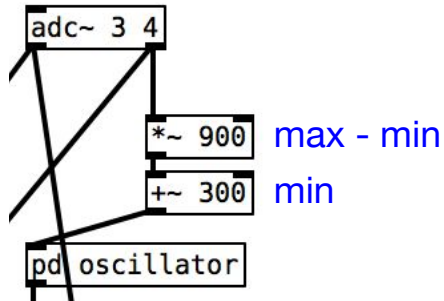


Bela & Pure Data

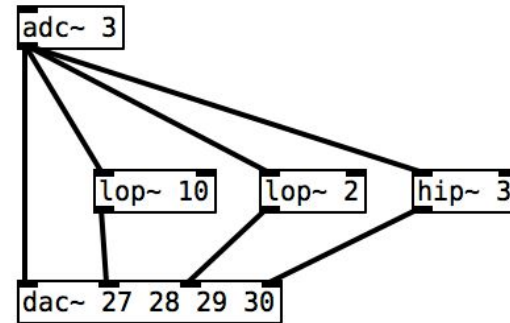


Analog input conditioning

- Scaling to specific range



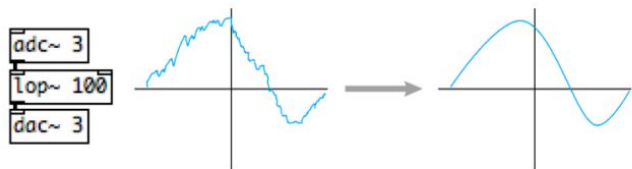
- Filtering



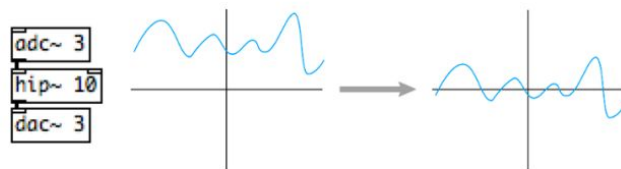
Bela & Pure Data



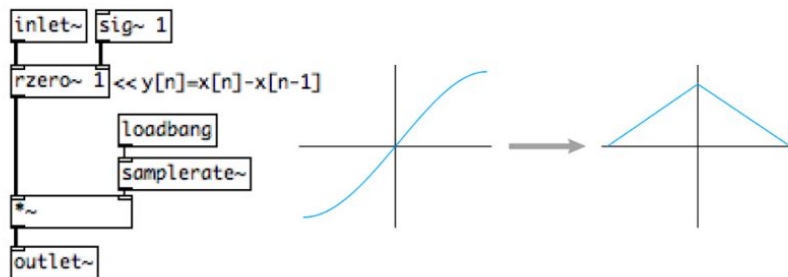
Analog input conditioning



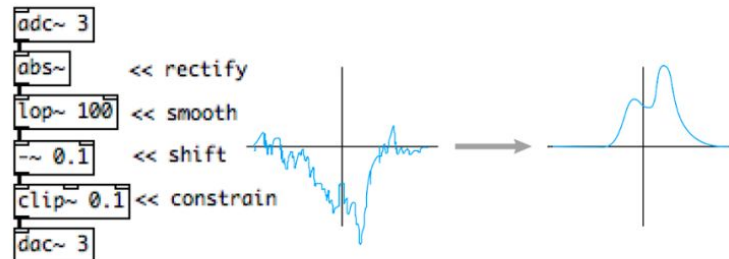
(a) Smoothing



(b) Re-centering



(c) Differentiating



(d) Thresholding

Getting up & running

Required software

Firefox/Chrome browser (*recommended)
The IDE may or may not work in other browsers.



Getting up & running

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>

Bela IDE

bela.local

or

192.168.7.2

or

192.168.6.2

The screenshot displays the Bela IDE interface. At the top left, a yellow box highlights the status bar showing "Project: 00-hello-sound" and "File: _main.pd". The main workspace contains a Pure Data patch with a "techno-world" object connected to a "dac~ 1 2" object. A red box labeled "Your projects" points to the "Your projects" dropdown in the Project Explorer. A blue box labeled "Project files" points to the "Sources" and "Abstractions" sections of the Project Explorer. The Project Explorer on the right shows the current project "00-hello-sound" and lists files like "_main.pd" and "techno-world.pd". The bottom status bar shows CPU and MSW usage, a green checkmark, and the Bela logo. A terminal window at the bottom shows the prompt "root@bela ~/Bela#".

Bela IDE

bela.local

or

192.168.7.2

or

192.168.6.2

The image shows the Bela IDE interface. The main workspace displays a Pure Data patch with objects like 'techno-world' and 'dac~ 1 2'. The Project Explorer sidebar on the right shows the current project '00-hello-sound' and its contents, including sources and abstractions. Annotations include a red box labeled 'Your projects' pointing to the project list, a blue box labeled 'Project files' pointing to the project content, and a pink box labeled 'Download' pointing to the download icon in the project content list.

Project: 00-hello-sound File: _main.pd

Hello Sound
=====

Press the run button in the IDE to hear this example.
You will hear algorithmic techno from the audio output.

<https://learn.bela.io/tutorials/pure-data/fundamentals/hello-sound/>

techno-world techno generator

dac~ 1 2 < Audio out L + R

Your projects

Project files

Download

Project Explorer

Your projects + New Project

Current project: 00-hello-sound

Project content

Sources

> _main.pd 0.79kB

Abstractions

> techno-world.pd 12.4kB

Git manager

CPU: -- MSW: --

root@bela ~/Bela#

Bela IDE

bela.local

or

192.168.7.2

or

192.168.6.2

Project: 00-hello-sound File: _main.pd

Hello Sound
=====

Press the run button in the IDE to hear this example.
You will hear algorithmic techno from the audio output.

<https://learn.bela.io/tutorials/pure-data/fundamentals/hello-sound/>

techno-world techno generator

dac~ 1 2 < Audio out L + R

Your projects

Project files

Download

Project Explorer

Your projects + New Project

Current project: 00-hello-sound

Project content

Sources

> _main.pd 0.79kB

Abstractions

> techno-world.pd 12.4kB

Git manager

CPU: -- MSW: --

bELa

root@bela ~/Bela#

Patch connections can be visualised in the browser,
but cannot be edited here

Bela IDE

bela.local

or

192.168.7.2

or

192.168.6.2

The image shows the Bela IDE interface. The main workspace displays a Pure Data patch titled "Hello Sound" with a "BELA" logo. The patch includes a "techno-world" generator and a "dac~ 1 2" audio output. A green box labeled "Build & run" has an arrow pointing to the play button (a circle with a right-pointing triangle) in the bottom toolbar. An orange box labeled "Stop" has an arrow pointing to the stop button (a square) in the bottom toolbar. The bottom toolbar also includes a refresh button, a volume icon, a mute icon, a checkmark, and a power icon. The Project Explorer on the right shows the current project "00-hello-sound" with a list of sources and abstractions. The bottom status bar shows "root@bela ~/Bela#" and system information like "CPU: --" and "MSW: --".

Bela IDE

bela.local

or

192.168.7.2

or

192.168.6.2

The image shows the Bela IDE interface with several annotations:

- Build & run:** A green box with a green arrow pointing to the 'Build & run' button in the bottom toolbar.
- Oscilloscope:** A purple box with a purple arrow pointing to the oscilloscope icon in the bottom toolbar.
- Stop:** An orange box with an orange arrow pointing to the stop button in the bottom toolbar.

The IDE window displays a Pure Data patch titled "Hello Sound" with the following components:

- text "Hello Sound" and "====="
- text "to hear this example." and "from the audio output."
- text "pure-data/fundamentals/hello-sound/"
- text "techno-world techno generator" with a patch icon
- text "dac~ 1 2 Audio out L + R" with a patch icon
- A "BELA" logo in the background.

The right sidebar shows the "Project Explorer" with the following content:

- Buttons: "Your projects" (dropdown), "+ New Project"
- Current project: **00-hello-sound**
- Project contents: -
- Sources: >_main.pd 0.79kB (edit, download, delete icons)
- Abstractions: >techno-world.pd 12.4kB (edit, download, delete icons)
- Git manager: +

The bottom status bar shows "CPU: --", "MSW: --", a green checkmark, and the "bela" logo with a power button icon.

The terminal window at the bottom shows "root@bela ~/Bela#".

Bela IDE

bela.local

or

192.168.7.2

or

192.168.6.2

The image shows the Bela IDE interface. The main workspace displays a Pure Data patch titled "Hello Sound" with components like "techno-world" and "dac~ 1 2". A "BELA" logo is visible in the background. The Project Explorer on the right shows the current project "00-hello-sound" and lists sources and abstractions. The bottom status bar includes CPU and MSW indicators, a green checkmark, and the "bELd" logo. A terminal window at the bottom shows the prompt "root@bela ~/Bela#".

Callouts in the image:

- Build & run**: A green box with an arrow pointing to the "Build & run" button in the bottom toolbar.
- Oscilloscope**: A purple box with an arrow pointing to the "Oscilloscope" button in the bottom toolbar.
- Stop**: An orange box with an arrow pointing to the "Stop" button in the bottom toolbar.

Bela IDE

bela.local or 192.168.7.2 or 192.168.6.2

The screenshot displays the Bela IDE interface. The main workspace shows a PureData patch with the following components and connections:

- A `techno-world` object connected to a `techno generator` object.
- The `techno generator` object connected to a `dac~ 1 2` object.
- The `dac~ 1 2` object connected to an audio output port labeled `< Audio out L + R`.

Text in the workspace includes:

```
Hello Sound
=====

Press the run button in the IDE to hear this example.
You will hear algorithmic techno from the audio output.

https://learn.bela.io/tutorials/pure-data/fundamentals/hello-sound/
```

A red box labeled "Examples" with an arrow points to the "Examples" icon in the right-hand sidebar. The sidebar contains a list of example patches:

- PureData
- > language-overview
- > hello-sound
- > digital-out
- > digital-in
- > analog-in
- > analog-out
- > am-synthesis
- > fm-synthesis
- > drum-synthesis
- > rubber-duckie
- > delay-effect
- > samples-playback
- > samples-record
- > pitch-tracking

The bottom status bar shows system information: CPU: --, MSW: --, and a green checkmark icon. The Bela logo and a power button icon are also visible in the bottom right corner.

Terminal output at the bottom: `root@bela ~/Bela#`

Bela IDE

bela.local

or

192.168.7.2

or

192.168.6.2

The screenshot shows the Bela IDE interface. The main workspace contains a Pure Data patch with a 'techno-world' message box and a 'dac~ 1 2' audio output object. A blue box labeled 'Settings' with an arrow points to the settings gear icon in the right-hand sidebar. The settings panel is open, showing various configuration options for the project.

Settings

Run project on boot:

Project Settings

Block size (audio frames):

Analog Channels:

Analog Sample Rate (Hz):

Digital Channels:

Use Analog:

Use Digital:

Mute Speaker:

Headphone Left level (dB):

Headphone Right level (dB):

Input Left gain (dB):

IDE, headphone level, I/O settings shutdown, update, etc

root@bela ~/Bela#

Might require rebooting/re-compiling project to take effect

No need to change anything for this workshop

Bela IDE

bela.local

or

192.168.7.2

or

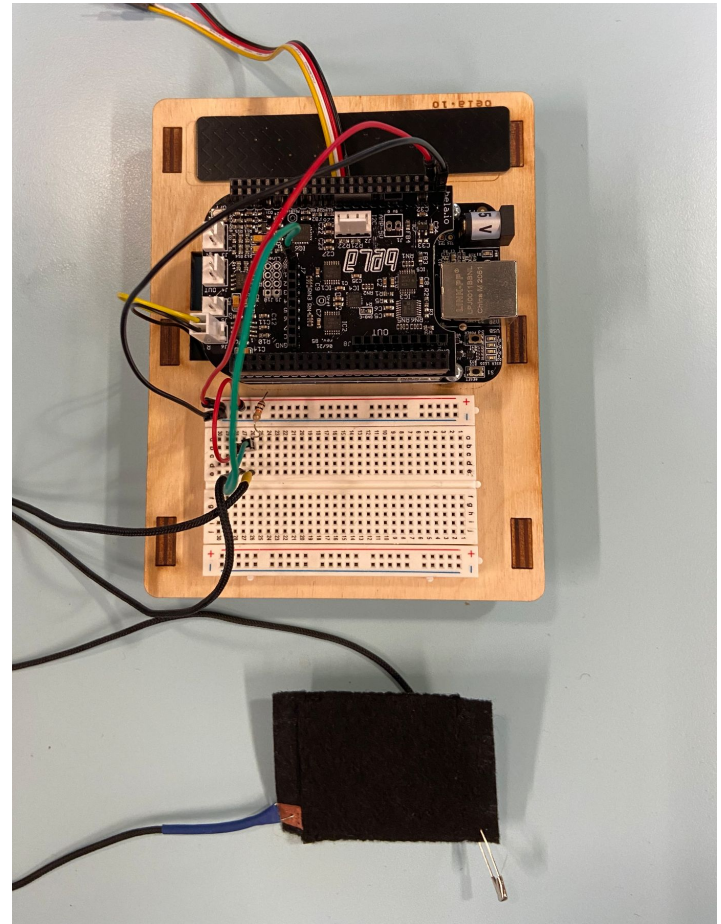
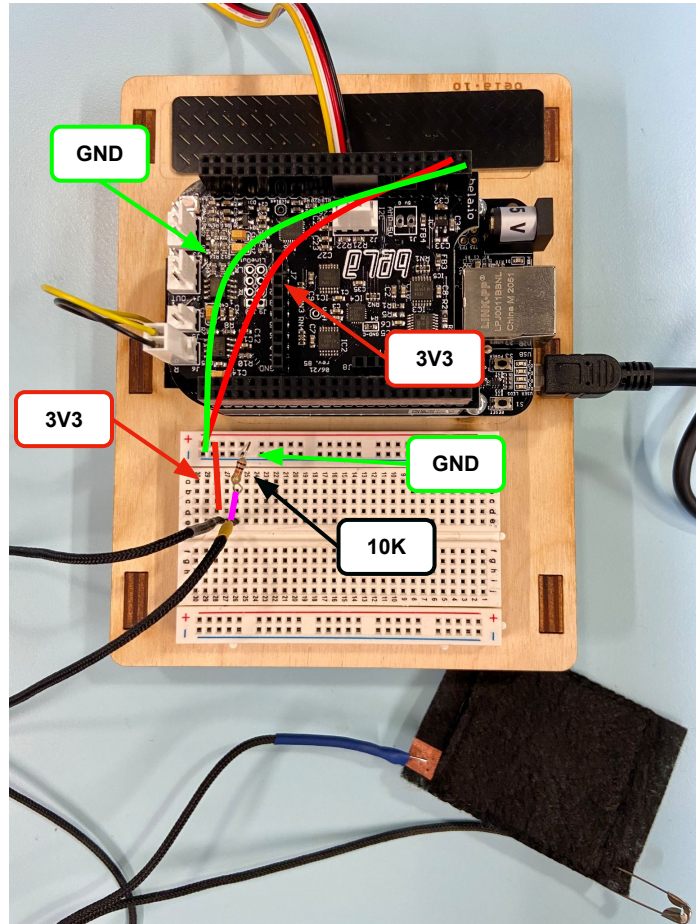
192.168.6.2

The image shows the Bela IDE interface. The main window displays a Pure Data patch titled "Digital Input". The patch includes a "loadbang" object, an "in 18, out 17" object (initialised to input and output), and an "s bela_setDigital" object. Below this, there is a section for "Digital input and out, message rate" with an "r bela_digitalIn18" object, a "Blink that LED!" message box, a "kick" and "snare" object (with a note "< extra sound making abstractions"), and an "s bela_digitalOut17" object. A pink box labeled "Pin diagram" with an arrow points to the pin diagram icon in the right-hand toolbar.

The right-hand panel shows the "Pin Diagram" section. It has a dropdown menu for "Current diagram:" set to "Bela". Below this is a photograph of the Bela board with a callout box pointing to the digital pins, labeled "Digital 7 (Pd: 18)". The Bela logo is visible at the bottom of the board image.

At the bottom of the IDE, there is a status bar with icons for refresh, stop, zoom, and a green checkmark. Below the status bar is a terminal window showing the prompt "root@Bela ~/Bela#".

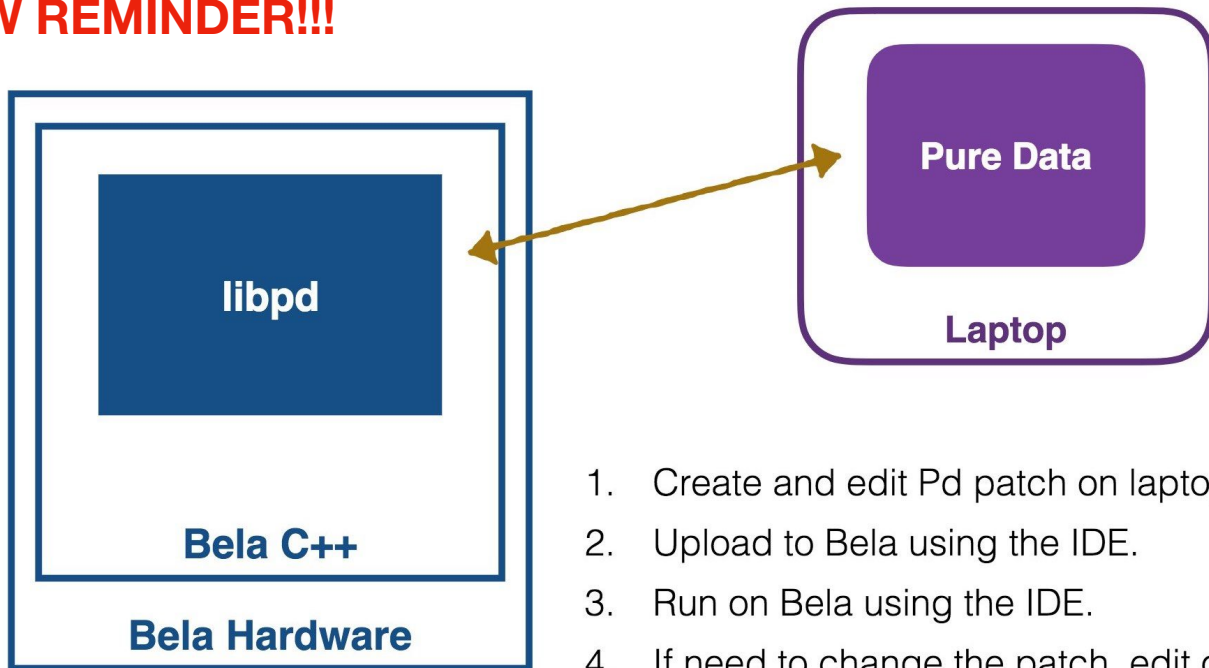
Connect your sensor to Bela!



Bela & Pure Data



WORKFLOW REMINDER!!!



1. Create and edit Pd patch on laptop.
2. Upload to Bela using the IDE.
3. Run on Bela using the IDE.
4. If need to change the patch, edit on laptop and repeat process.

(*)

The main Pd patch of your project must always be named `_main.pd`

Open example [Examples/PureData/sample-scrub](#) In IDE, you can **view** and **run** but **not edit** Pd patches

The screenshot shows the Bela IDE interface. At the top, there are browser tabs for "_main.pd, sample-scrub (exam)" and "Bela Oscilloscope". The address bar shows "Not Secure | bela.local/". Below the browser, the IDE header displays "Project: sample-scrub (example)", "File: _main.pd", and "Running: exampleTempProject".

The main workspace contains a PureData patch with two sections. The first section is titled "Load sample into table and scrub using analog input" and contains a patch with objects: `loadbang`, `read -resize waves.wav exampleTable1`, `soundfiler`, and `s sampleLength1`. The second section contains a patch with objects: `loadbang`, `read -resize rt.wav exampleTable2`, `soundfiler`, and `s sampleLength2`. Between these sections, there is explanatory text: "<<< loading a file: point soundfiler to the samples folder name_of_file.wav" and "<<< If you include your own samples in the project just change the address to the file name".

On the right side, there is a sidebar titled "Examples" with a list of example patches: `> accelerometer`, `> oscilloscope`, `> gui-receive`, `> gui-send`, `> hello-midi`, `> midi-cv-midi`, `> osc-receive`, `> osc-send`, `> custom-render`, `> digital-advanced`, `> envelope-trigger`, `> monophonic-synth`, `> multiplexer-capelet`, and `> sample-scrub` (highlighted in yellow). Below the list are "SuperCollider" and "Csound" with plus signs.

At the bottom, there is a control bar with a green play button, a square stop button, a waveform icon, a volume icon, and a right arrow icon. The status bar shows "CPU: 34.8%", "MSW: 0", and a green checkmark. The "bela" logo is visible in the bottom right corner.

At the very bottom, a log window shows the following output:

```
bela: No MIDI device enabled
bela: bonk version 1.5
bela: fiddle version 1.1 TEST4
bela: pique 0.1 for PD version 23
bela: sigmund~ version 0.07
```

Open example `Examples/PureData/Oscilloscope` In IDE, you can **view** and **run** but **not edit** Pd patches

The screenshot shows the Bela IDE interface. At the top, there are browser tabs for `_main.pd, oscilloscope (example)` and `Bela Oscilloscope`. The address bar shows `Not Secure | bela.local/`. Below the browser tabs, the IDE header displays `Project: oscilloscope (example)`, `File: _main.pd`, and `Running: exampleTempProject`.

The main workspace is divided into two sections. On the left, there is a text area titled "The Bela Oscilloscope" with the following content:
=====
Push scope button in the IDE. This example sends two analog inputs to the scope and filtered white noise.

<https://learn.bela.io/tutorials/pure-data/communication/oscilloscope/>

Below the text is a PureData patch diagram. It features several objects: `adc~ 3 4`, `noise~`, `loadbang`, `*~ 8000`, `metro 50`, `snapshot~`, `+ 0.2`, `* 40`, and `vcf~`. The patch is connected to a scope icon in the bottom toolbar.

On the right side, there is a sidebar titled "Examples" with a list of example patches. The `oscilloscope` example is highlighted in yellow. The list includes:
> delay-effect
> samples-playback
> samples-record
> karplus-strong
> capacitive-sensing
> diy-pressure-sensors
> accelerometer
> oscilloscope
> gui-receive
> gui-send
> hello-midi
> midi-cv-midi
> osc-receive
> osc-send
> custom-render
> digital-advanced
> envelope-trigger
> monophonic-synth

At the bottom of the IDE, there is a status bar with a CPU usage indicator (34.5%), a memory usage indicator (MSW: 0), and a green checkmark. The `bela` logo is visible in the bottom right corner.

Below the IDE interface, there is a terminal window showing the following output:
bela: No MIDI device enabled
bela: bonk version 1.5
bela: fiddle version 1.1 TEST4
bela: pique 0.1 for PD version 23
bela: sigmund~ version 0.07
root@bela ~/Bela#

A red arrow points from the terminal window to the scope icon in the IDE toolbar, with the text "Launch scope to view interactions with sensors" written in red.

Open example [Examples/PureData/rubber-duckie](#) In IDE, you can **view** and **run** but **not edit** Pd patches

The screenshot displays the Bela IDE interface. At the top, a browser window shows the URL `h _main.pd, rubber-duckie (exan x +` and the address bar contains `Not Secure | bela.local/`. Below the browser, the IDE header shows `Project: rubber-duckie (example) File: _main.pd Running: exampleTempProject`.

The main workspace is divided into three sections:

- Left Panel:** A PureData patch diagram with the following objects and comments:
 - `adc- 3` <<< use FSR on analog input 0
 - `sigdelta` <<< calculate differential on input signal
 - `duckie` <<< open this abstraction to see how the model works
 - `*~ 0.1`
 - `dac- 1 2`
- Center Panel:** A large area of ASCII art representing a rubber duck, with a vertical line of dots on the left side.
- Right Panel:** A sidebar titled **Examples** with a list of patch names:
 - > analog-in
 - > analog-out
 - > am-synthesis
 - > fm-synthesis
 - > drum-synthesis
 - > rubber-duckie (highlighted)
 - > delay-effect
 - > samples-playback
 - > samples-record
 - > karplus-strong
 - > capacitive-sensing
 - > diy-pressure-sensors
 - > accelerometer
 - > oscilloscope
 - > gui-receive
 - > gui-send
 - > hello-midi
 - > midi-cv-midi

At the bottom of the IDE, there is a status bar with icons for refresh, stop, and a green checkmark. It also displays system information: `CPU: 21.4%` and `MSW: 0`. The **bela** logo is visible in the bottom right corner.

Below the IDE interface, a terminal window shows the following output:

```
bela: No MIDI device enabled
bela: bonk version 1.5
bela: fiddle version 1.1 TEST4
bela: pique 0.1 for PD version 23
bela: sigmund~ version 0.07

root@bela ~/Bela#
```


Open example `Examples/PureData/analog-in` In IDE, you can **view** and **run** but **not edit** Pd patches

The screenshot shows the Bela IDE interface. At the top, there are browser tabs for `_main.pd, analog-in (example)` and `Bela Oscilloscope`. The address bar shows `Not Secure | bela.local/`. Below the browser, the IDE header displays `Project: analog-in (example)`, `File: _main.pd`, and `Running: exampleTempProject`.

The main workspace is divided into two panes. The left pane contains text and a PureData patch diagram. The text reads: "Analog in (or 'theremin')", "In this example the first two analog inputs are used to control the frequency and amplitude of an oscillator.", "Note that analog inputs and outputs have a range between 0 and 1, unlike regular audio signals (which have a range between -1 and 1).", and a URL: `https://learn.bela.io/tutorials/pure-data/connecting/analog-input/`. The patch diagram shows an `adc- 3 4` object connected to two `*~ 1500` objects, which are connected to a `pd_oscillator` object. The `pd_oscillator` is connected to another `*~` object, which is connected to a `dac- 1 2` object.

The right pane is titled "Examples" and contains a list of example patches:

- > hello-sound
- > digital-out
- > digital-in
- > analog-in
- > analog-out
- > am-synthesis
- > fm-synthesis
- > drum-synthesis
- > rubber-duckie
- > delay-effect
- > samples-playback
- > samples-record
- > karplus-strong
- > capacitive-sensing
- > diy-pressure-sensors
- > accelerometer
- > oscilloscope
- > gui-receive

At the bottom of the IDE, there is a status bar with a CPU usage indicator (35.2%), a memory usage indicator (MSW: 0), a green checkmark, and the `bela` logo. Below the status bar, a terminal window shows the following output:

```
i Building project ...
make: Linking...
make: ...done
i Build finished
i Running project ...
bela: Running Pd 0.48-2
```

Open example [Examples/PureData/diy-pressure-sensors](#)

In IDE, you can **view** and **run** but **not edit** Pd patches

The screenshot displays the Bela IDE interface. The top browser window shows the URL `belocal/` and the project name `diy-pressure-sensors (example)`. The main area contains a PureData patch diagram with the following components:

- `adc- 4` and `adc- 3` (Analog-to-Digital Converters)
- `lop- 1` and `lop- 2` (Low-pass filters)
- `r $0-sampleLength1` and `r $0-sampleLength2` (Read sample length)
- `*~` (Multipliers)
- `tabread4- $0-table1` and `tabread4- $0-table2` (Table readers)
- `dac-` (Digital-to-Analog Converter)

Text in the IDE:

```
Sonifying a diy FSR by scrubbing through a sample
=====
For details on how to build the velostat pressure sensors
see: http://embelashed.org/paper/sensors/pressure.html

https://learn.bela.io/tutorials/pure-data/sensors/diy-pressure-sensor/
```

On the right, a sidebar titled "Examples" lists various patches:

- > am-synthesis
- > fm-synthesis
- > drum-synthesis
- > rubber-duckie
- > delay-effect
- > samples-playback
- > samples-record
- > karplus-strong
- > capacitive-sensing
- > **diy-pressure-sensors**
- > accelerometer
- > oscilloscope
- > gui-receive
- > gui-send
- > hello-midi
- > midi-cv-midi
- > osc-receive
- > osc-send

The bottom status bar shows CPU usage at 23.3%, MSW at 0, and a green checkmark. The "bela" logo is visible in the bottom right corner.

```
make: Linking...
make: ...done
i Build finished
i Running project ...
bela: Running Pd 0.48-2
bela: Audio channels in use: 2
```