hedrot
Version 1.2 - User Manual

Table of Contents

0) Preliminary notes

The present documentation is not completely up-to-date regarding calibration! As a matter-of-fact, the calibration routines for the magnetometer and the accelerometer were completely rewritten (and work much better) in version 1.2 of hedrot. This version introduces also the ability to calibrate the magnetometer in real-time (avoiding thus the need for an explicit calibration before each use). This is still experimental and not completely bug-free, but very promising!

1) Quick start

- Please read part **2) Introduction**, and especially the software and hardware requirements, before starting
- Instructions to prepare the head tracker => parts **3) Assembling the Head tracker** and **4) Install/Update the hedrot Firmware in the Teensy 3**
- Positioning the head tracker on the headphone => **part 5) Positioning the head tracker on the headphone**
- First start of a new hedrot head tracker:
  - please calibrate first (see part **7) Calibrating the Head tracker**) !!
  - Start the application *hedrotReceiver*
  - click on "Headtracker is OFF"

  => after a few seconds the head tracker should be detected and start to communicate
- **Tip: If the head tracker works, but the angles seem to be not coherent**: double-check if the magnetometer is properly calibrated. If it seems impossible to calibrate it, it probably means that the magnetic and/or ferromagnetic interferences are too strong => in this case use hedrot without the magnetometer (set the Max Beta parameter to 0). More information in the **Technical Note 3: Madgwick's algorithm and the Beta parameter**.
- To see examples of application with different renderers (Mybino, Ambix/Reaper, Max+Spat...), go to part **9)**.

2) Introduction

a) What is hedrot?

Hedrot (for "head rotation tracker") is a low-cost (around 25 euros with a teensy LC) and efficient open-source hardware/software solution for head tracking. *Hedrot* is especially suitable for binaural rendering (3D-Audio on headphones), and has been initially designed for use with the binaural renderer *Bipan* as part of the Bili Project http://www.bili-project.org/.

Hedrot provides an estimation of the rotation of the sensor (thus of the head if the sensor is attached to headphones) for the most usual x-y-z coordinate systems, either as a quaternion, or as a set of 3 orientation angles yaw, pitch and roll with two different orders (yaw-pitch-roll or roll-pitch-yaw). The main application provided with the distribution, *hedrotReceiver*, sends this information as OSC streams, with the extra possibility to scale each stream independently.

Contrary to several generic open-source head tracking solutions, *hedrot* relies on and has been optimized for specific widely spread and efficient hardware parts, i.e. a Teensy 3 or Teensy LC board (optimized Arduino-like board) combined to a IMU/MARG daughter board with 3 common sensors (Analog Devices ADXL345 accelerometer, Honeywell HMC5883L magnetometer and Invensense ITG-3200 gyroscope).

The estimation algorithm is based on a modified version of the precise and efficient open-source gradient descent algorithm from Sebastian Madgwick. The technology was dramatically optimized for speed: the head tracker can deliver data at rates up to 2 kHz. The hardware latency of the Teensy board and USB communication relies below 2 ms. The overall latency (including sensor latency and time constant of the algorithm) relies between 25 and 45 ms.

b) Further Developments

The next developments and research include among others:

- Measurements of the overall (software+hardware) latency
- A 3D-printable enclosure
- An enhanced automatic magnetometer calibration algorithm
- A precise calibration algorithm for the gyroscope

Contributions to help developing this collaborative project further are warmly welcome. Please contact Alexis Baskind (a@alexisbaskind.net) for proposals and questions.

c) Licensing and Credits

The first development phase of Hedrot has achieved in collaboration with the Conservatoire National Supérieur de Musique et de Danse de Paris (http://www.conservatoiredeparis.fr/) as part of the "Bili" project (http://www.bili-project.org/).

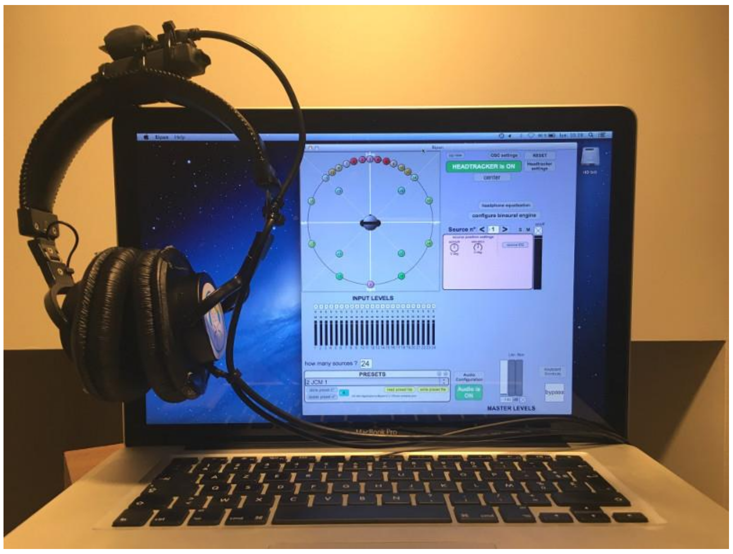


Figure 1: The Bipan binaural renderer with the Head Tracker Hedrot on top of the headphone

Hedrot is licensed under the terms of the GNU General Public License (version 3) as published by the Free Software Foundation.

Part of code is derived from Sebastian Madgwick's open-source gradient descent angle estimation algorithm (http://x-io.co.uk/open-source-imu-and-ahrs-algorithms/)

Part of code is derived from "comport", (c) 1998-2005 Winfried Ritsch, Institute for Electronic Music - Graz.

Part of code is derived from Yuri Petrov "ellipsoid fit" algorithm (initially written for Matlab).

Developers and Contributors:

- Alexis Baskind (sound engineer, main developer)
- Jean-Christophe Messonnier (sound engineer)
- Jean-Marc Lyzwa (sound engineer)
- Matthieu Aussal, CMAP - Ecole Polytechnique / CNRS (calibration routines)

d) Hardware Requirements

Head tracker parts:

- 1 Teensy 3 board (tested with versions 3.1, 3.2 and LC)
- 1 USB to Micro-USB 2.0 cable, minimum length 1.5 m
- 1 gy85 IMU daughter board with the following sensors:
    - 1 Analog Devices ADXL345 accelerometer
    - 1 Honeywell HMC5883L compass (magnetometer)
    - 1 Invensense ITG3200 gyroscope

e) Software Requirements

- Mac: Mac OS 10.9.5 or later
- Windows: Windows 7 or later
- the teensy.app firmware flash loader
- Windows: Microsoft Visual C++ 2012 Redistributable package (x86 or x64 version, depending on if a 32 or 64 bit version of the hedrot is being used)
- Windows: Teensy serial driver (called "Windows Serial Installer" on https://www.pjrc.com)
- Windows: the following DLL files are required: libblas.dll, libgcc_s_seh-1.dll, libgfortran-3.dll, liblapack.dll, liblapacke.dll, libquadmath-0.dll, libwinpthread-1.dll

  Those files are provided with the distribution in the subfolder “dll”. **IMPORTANT: add this folder in one of the Windows path folders (either user or system). For this, open the “Environment Variables” configuration from Windows**

Technical Note 1: Important Note for running "hedrotReceiver" on Mac

When running the application "hedrotReceiver", the following error message can pop-up:

"hedrotReceiver" is damaged and can’t be opened. You should move it to the Trash

This message is wrong: the application is not damaged. This message comes from the fact that the application has not been signed by Apple for the Mac App Store.

To get rid of this message and being able to run the application, the following can be done: Go to "System Preferences", and then to "Security & Privacy", then "General, and in "Allow apps downloaded from", select "Anywhere". **Caution**: this will reduce the security level of your computer and has to be done at your own risk. As an alternative, it's still possible to run the Max Patch from the source code using Max 6 or Max 7.

For Sierra (10.12) users: the option "Anywhere" is hidden by default. To unhide it: open a Terminal a type:

```
sudo spctl --master-disable
```

You'll have to type the administrator password.

Extra Requirements for building from sources:

- Xcode version 6.2 or later (for mac), or Visual Studio 2012 (for Windows)
- Windows: a windows version of awk (like gawk) and the program 7-zip to make the package
- Max version 6 at least (to rebuild the "hedrotReceiver" application. Not needed otherwise)
- Arduino IDE 1.8.3
- Windows: a distribution of BLAS, Lapack and LAPACKE. See README-BUILD.txt for more information about how to build it
- Teensyduino 1.37 (teensy support for the Arduino IDE and the teensy USB serial driver on Windows), with at least the i2c_t3 library. **Note:** Teensyduino already includes the Teensy serial driver on Windows.

f) How to use Hedrot?

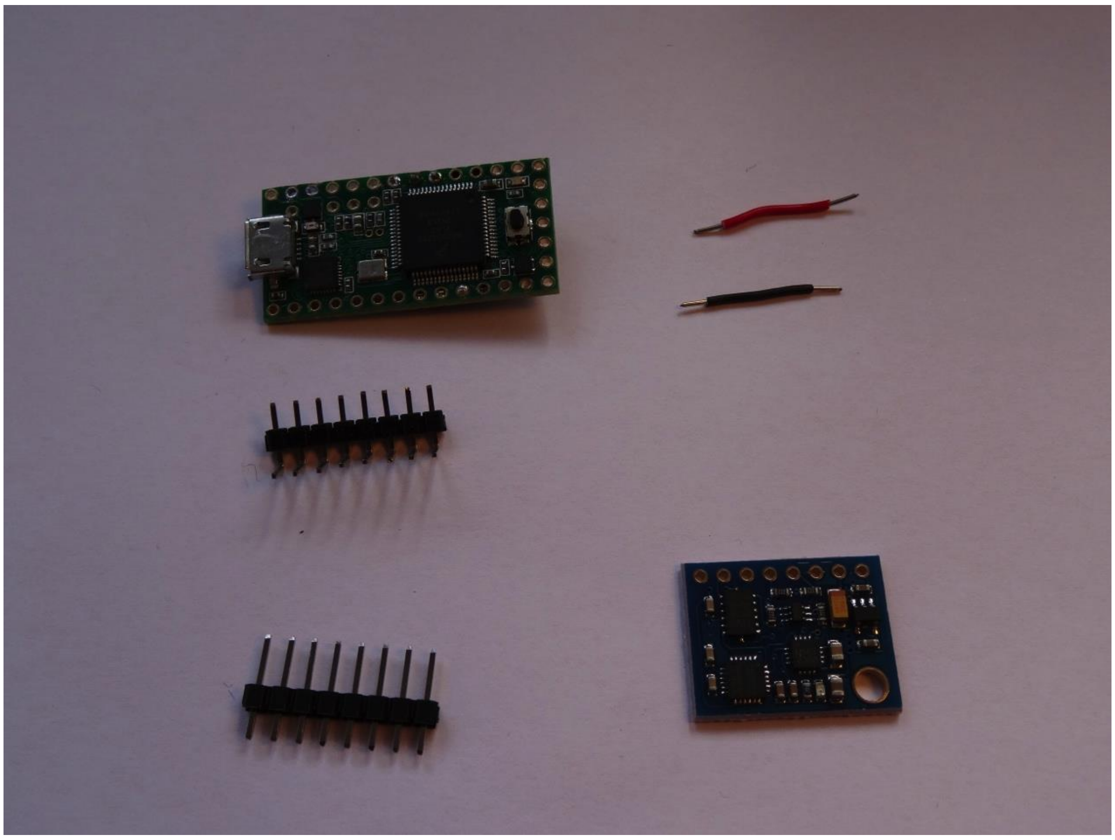
Hedrot can be used in two different ways:

- either through the standalone application *hedrotReceiver*, provided with this distribution, that receives the data from the head tracker, allow the user to calibrate it, and send the calculate angles to a renderer through OSC. Please refer to part "**8) Using the hedrotReceiver Application**" for more details about how to use *hedrotReceiver*
- or by imbedding the source code of the hedrot library in the source code of the renderer. The source code is available on github (https://github.com/abaskind/hedrot) and is free of use in the context of a private or public GPLv3 project.

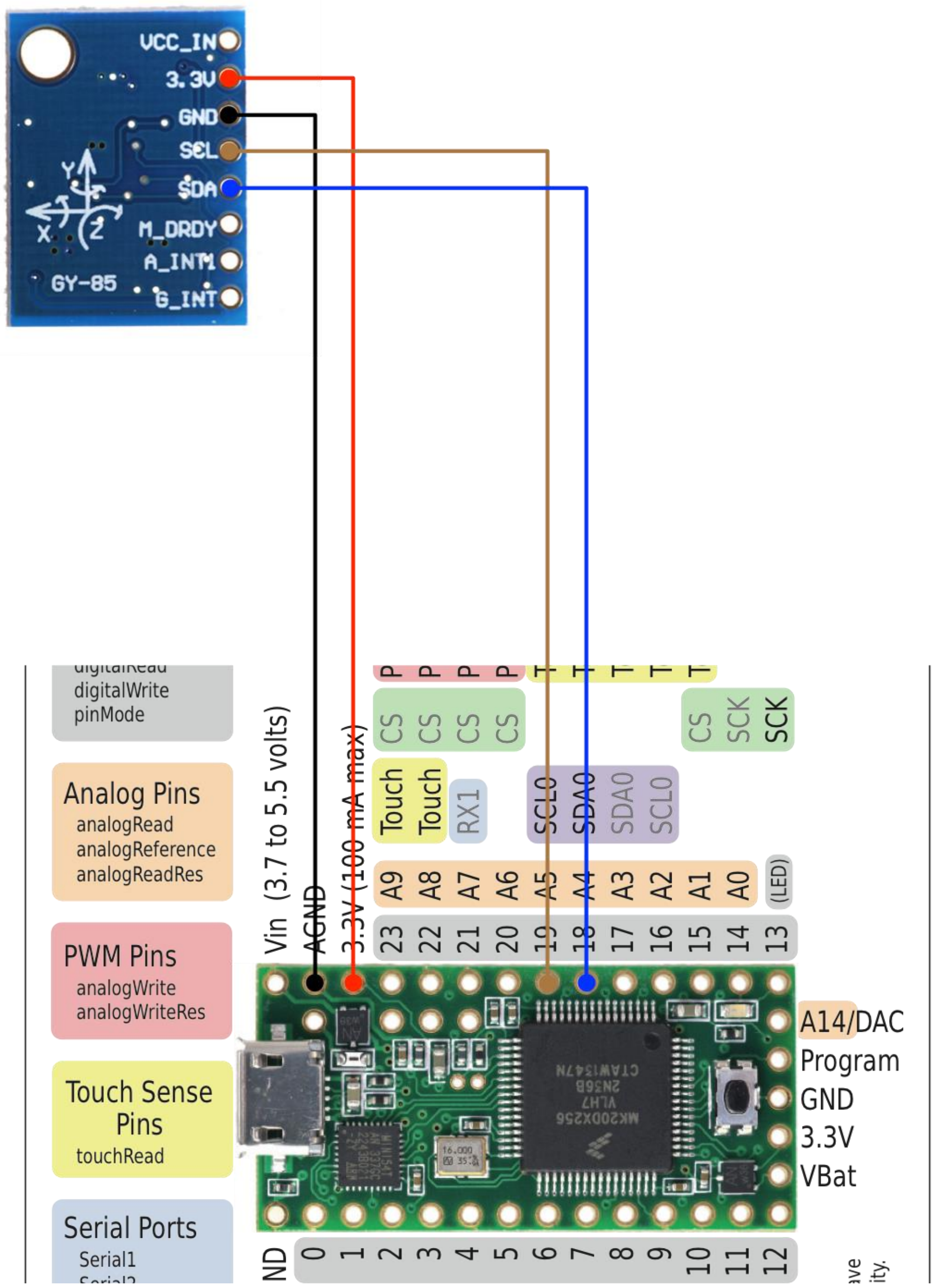
3) Assembling the Head tracker

a) Required parts

- 1 Teensy 3 board (tested with versions 3.1, 3.2 and LC)
- 1 USB to Micro-USB 2.0 cable, minimum length 1.5 m
- 1 gy85 IMU daughter board with the following sensors:
    - 1 Analog Devices ADXL345 accelerometer
    - 1 Honeywell HMC5883L compass (magnetometer)
    - 1 Invensense ITG3200 gyroscope
- 1 straight multipin connector, 4 or 5 pins
- 1 angled multipin connector, 4 or 5 pins
- 2 short electronic cables



b) Schematics



c) Setting the multipin connectors on the Teensy board

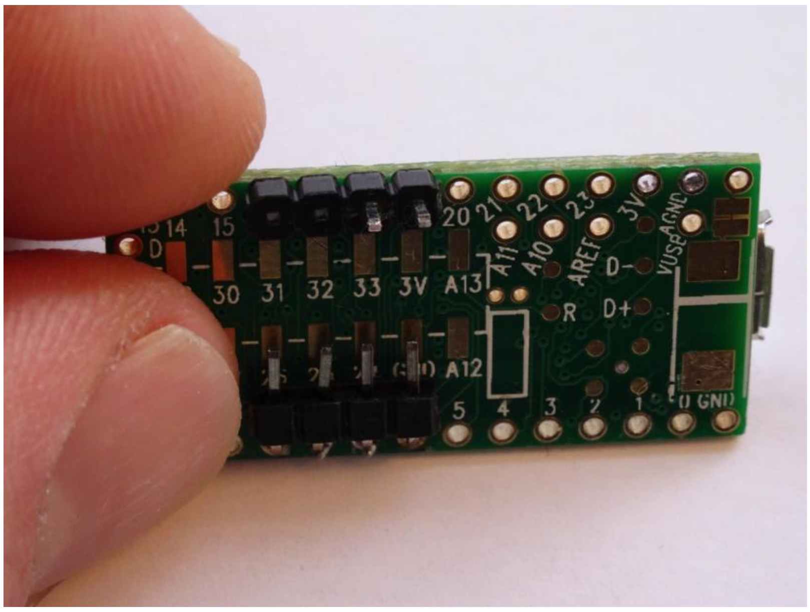
The straight multipin connector has two functions:

- connect pins 18 and 19 from Teensy to respectively pins SDA and SDL of the GY-85 daughter board.
- stabilize the daughter board on the main board mechanically

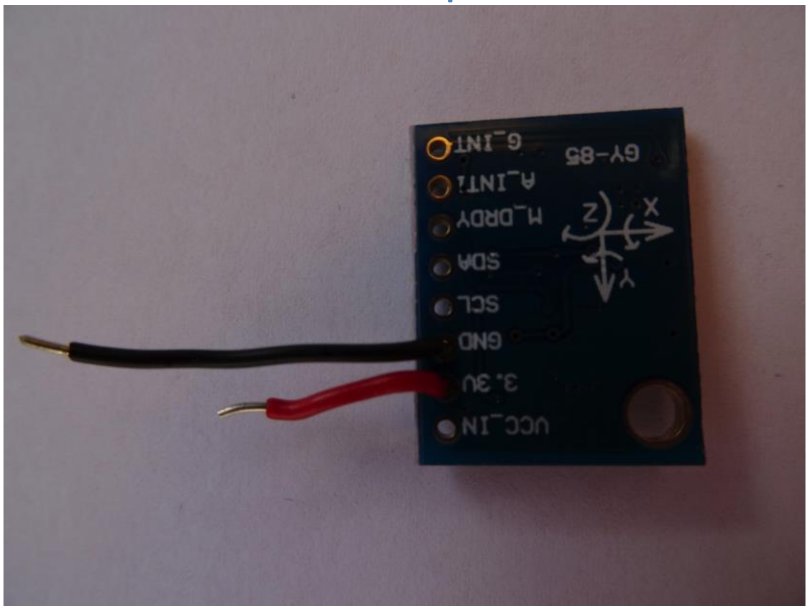
Only both pins 18 and 19 are to be connected! All other metallic connections should be removed. However the plastic connection should be

larger as two pins, in order to ensure a mechanical stability of the daughter board (see photo below).

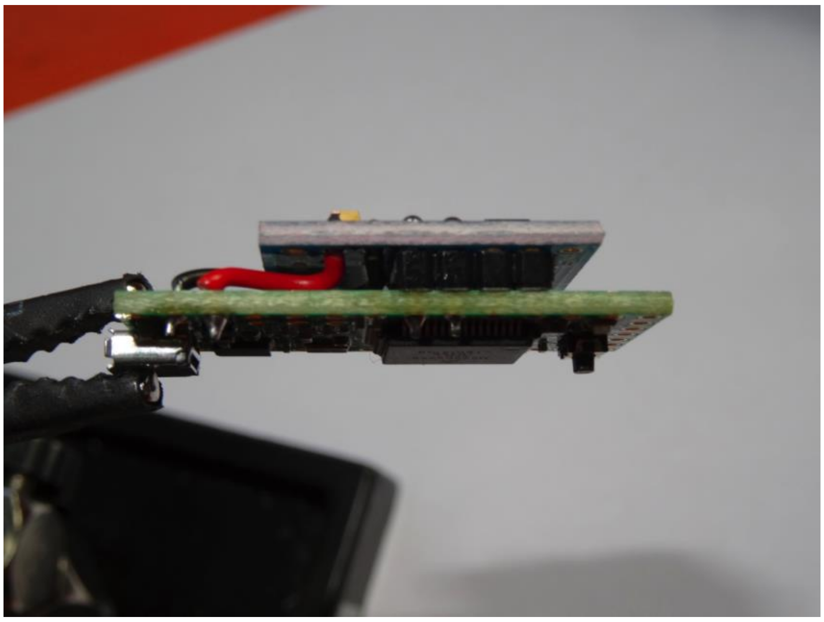
The angled multipin connector has no connection function, **no pin of the daughter board should be connected to it**. Its only function is the mechanical stabilization of the other side of the daughter board.



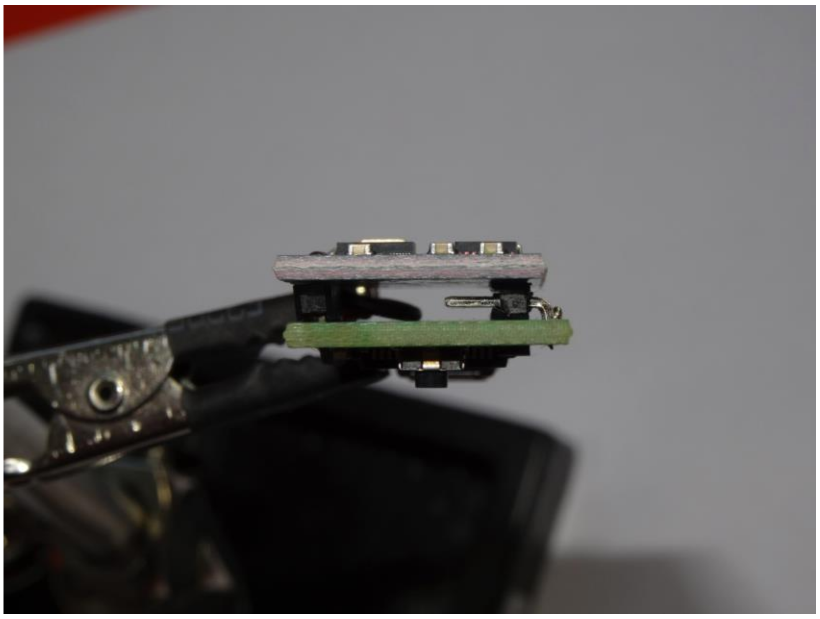
d) Set both cables on GND and +3.3V pins of the GY-85 board



e) Connect both cables on the main board, install the daughter board on the main board and solder



Side view



Rear view

4) Install/Update the hedrot Firmware in the Teensy 3

a) Download the latest version of the Teensy loader

...on Teensy's website https://www.pjrc.com/teensy/

b) Load the head tracker firmware

Important Notice

Before any firmware update (not for a first use), it is highly recommended to save the head tracker calibration settings:

- Start hedrotReceiver
- Start the head tracking (click on "Headtracker is off"). If "Autodiscover" is off (in "Headtracker settings"), select the right serial port (typically "/dev/cu.usbmodemXXXXXX" on mac). If "Autodiscover" is on, nothing has to be done
- When the head tracker is connected to hedrotReceiver, click "export calibration settings". Save the calibration settings in a text file

Loading/Updating the Firmware

- Start the program "teensy.app" (Mac) or “teensy.exe” (Win), set it in “Automatic” mode
- Drag & drop the current version of the firmware provided with the distribution of hedrot (file "hedrot-firmware-teensy31-32_version_XX.hex" for Teensy 3.1 or 3.2, and file "hedrot-firmware-teensyLC_version_XX.hex" for Teensy LC, where "XX" is the version number of the firmware)
- Click on the reset button on the teensy board
- If calibration settings were saved before updating the firmware, import them in the head tracker: when the head tracker is connected to hedrotReceiver, click on "import calibration settings". Select the text file containing the calibration settings

First use of the head tracker

After the firmware has been uploaded for the first time in the head tracker, settings have to be initialized!!!

- Start hedrotReceiver
- Start the head tracking (click on "Headtracker is off"). If "Autodiscover" is off (in "Headtracker settings"), select the right serial port (typically "/dev/cu.usbmodemXXXXXX" on mac). If "Autodiscover" is on, nothing has to be done
- Click on "Headtracker Settings". In the new window, click on "reset all headtracker settings"

5) Positioning the head tracker on the headphone

IMPORTANT

- **The head tracker must be positioned on the top of the headphone**

- **The GY-85 daughter board should be on the top, the USB connector below and on the left side, so that the USB cable runs on the left side of the headphone**

6) Starting the Head tracker

- Start *hedrotReceiver*
- Start the head tracking (click on "Headtracker is off"). If "Autodiscover" is off (in "Headtracker settings"), select the right serial port (typically "/dev/cu.usbmodemXXXXXX" on mac). If "Autodiscover" is on, nothing has to be done
- When the connection has been established and if the calibration is valid, the data should be transmitted regularly
- If the connection has been established but if the calibration has not being done yet or if it's not valid, hedrotReceiver will show an error message

7) Calibrating the Head tracker

a) To show the head tracker raw and calibrated data

- Start *hedrotReceiver*
- Click on "Calibration". On the new window that appears, the raw data is shown above, the calibrated data below
- For the accelerometer, the calibrated data should ideally always remain within -1 and 1 when the head tracker is still, and the norm should always stay close to 1. If this is not the case, it means that the calibration has not been done properly and should be redone
- For the magnetometer, the calibrated data should ideally always remain within -1 and 1, and the norm should always stay close to 1. If this is not the case, it means that the magnetic environment changed and that calibration has to be redone. Remember that the compass is very sensitive to the presence of any ferromagnetic interference (metallic cases, magnetic fields etc.)
- For the gyroscope, the calibrated data should stay around 0 when the head tracker is still.

b) Calibrating the accelerometer

The accelerometer should normally be calibrated only once, and the head tracker does not need to be attached to the headphone for this.

In order to calibrate the accelerometer:

- Click on "Headtracker Settings"
- Click on "calibrate accelerometer"
- Click on "start calibrating"
- Turn the head tracker in as many direction as possible, at least around x, y and z-axes, **AND AS SLOW AS POSSIBLE**
- At the end, click on "stop calibrating", and confirm the data export to the head tracker

c) Calibrating the magnetometer

The magnetometer is very sensitive to the presence of ferromagnetic material and to magnetic fields, and needs to be calibrated quickly **on the headphone and at its definitive position** each time:

- its position relative to the headphone changed
- a new headphone is being used
- the magnetic environment changed. This can happen for example with a laptop if one sits closer or farther to it.

In order to double-check if a new calibration is required or not, check the calibrated data as explained above.

In order to calibrate the magnetometer:

- Click on "Headtracker Settings"
- Click on "calibrate magnetometer"
- Click on "start calibrating"
- Turn the head tracker in as many direction as possible, at least around x, y and z-axes. It does not need to be done slowly as for the accelerometer
- At the end, click on "stop calibrating", and confirm the data export to the head tracker

Note: a procedure for real-time automatic calibration of the magnetometer has been included in hedrot version 1.2+. This procedure is still experimental and not 100% reliable

8) Using the *hedrotReceiver* Application

Note 1: the head tracker needs to be calibrated before the first use. Refer to part "**7) Calibrating the Head tracker**" for more details.

Note 2: the magnetometer should ideally be shortly calibrated before each use, as explained in part "**7)c) Calibrating the magnetometer**"

Technical Note 2: Magnetometer or not Magnetometer

The IMU sensor used in *hedrot* relies on 3 sensors, among others a magnetometer. The magnetometer allows the estimation to use the magnetic north as an absolute reference for orientation. However the data provided by the magnetometer is sensitive to magnetic perturbations, mainly external magnetic fields and/or a too strong ferromagnetic environment (too much iron-based metal around). This is a well-known limitation of all those sensors that can hardly be overcame.

There is a possibility to disconnect the magnetometer in *hedrot* by setting the MaxBeta parameter to 0 (see **Technical Note 3: Madgwick's algorithm and the Beta parameter**). This means however, that the absolute orientation is not any more known and that estimation drifts may occur during the use. The "center" function should be used then more often, if the user notices that the front drifted from its initial position.

a) Starting the head tracker

Click on the red button "Headtracker is OFF" (top left of the window). If the head tracker is connected, the software should find it and connect to it automatically. as soon as the connection is being established, the calculated angles show up.

Click on "center" allow to set the current position as the reference position (0° for yaw/pitch/roll)

Just below, a pop-up menu with two options ("transmit yaw only" and "transmit yaw/pitch/roll") allows to determine if only the yaw (rotation in the horizontal plane) or if all 3 angles should be transmitted.

b) Sending the calculated angles via OSC

Click on "OSC Settings". In the new window:

- edit the IP address and port of the OSC receiver (typically the audio renderer)
- edit the OSC patterns so that the receiver can interpret them
- Select which streams have to be sent (column "transmit?")
- Scale and/or invert the parameters if needed

Those settings can be saved in a preset with all the others (see part "**8)e) Store/recall presets**").

c) Head tracker (hardware) Settings

In the window "Headtracker Settings" are shown all the hardware parameters, i.e. the **hardware refresh rate** ("sample rate", default 1 kHz) and the low-level settings of all 3 sensors.

Among those parameters, only the sample rate is likely to be modified for your needs, the other ones were already set to optimal values. The hardware sample rate can be set up to 2300 Hz without any problems, although it should not make a big difference with the default value of 1000 Hz, the 3 sensors are anyway not quick enough to take benefit of it. **Important**: **The sample rate has a strong influence on the spatial precision. Don't set it too low.**

d) Receiver (software) Settings

In the window "Receiver Settings" are shown all the parameters for the software estimation of the angles, i.e.:

- the **poll period** (software sample period) in ms (default value 5 ms) determines the output refresh rate of the angles. It should be small enough to ensure the head tracker to respond quickly, but should not be smaller than the hardware sample period (as defined by the sample rate) in the hardware settings). For instance, if the hardware sample rate is set to 500 Hz, there is no point setting the poll period higher than 2 ms. Also, for audio applications there is no need to set the poll period smaller than the audio buffer size, since the audio renderer cannot take all values into considerations in this case.
- The "**accelerometer low-pass filter time constant**" (default value 0.01 s) reduces the background noise of the output data of the accelerometer. It should remain small, otherwise latency will be introduced
- "**Gradient Descent: max Beta**" (default value 2.5) and "**Gradient Descent: Beta variation according to movement**" (default value 1) determine how the main parameter "Beta" of Sebastian Magdwick's algorithm (see Technical Note below).

The 3 next settings aim at adapting the coordinate system, output angles and quaternions to the most current cases that can be found in the usual audio renderers (see part **9) Examples of Application**):

- The "**axes references**" parameter allow to change the x-y-z coordinate system among three possibilities:
  - X->right, Y->back, Z->down
  - X->right, Y->front, Z->up
  - X->front, Y->left, Z->up

  => this parameter has an effect on both the quaternion and the angles.
- The "**rotation order**" parameter determines in which order the yaw, pitch and roll angles are derived from the quaternion. This setting has no effect on the quaternion, but modifies the angle
- The "**invert rotation**" parameter allow to change the perspective: if "don't invert rotation" is checked, the angles and quaternion correspond to the rotation from the absolute x-y-z axes to the mobile x-y-z axes related to the head tracker. if "invert rotation" is checked, it's the opposite: the angles and quaternion correspond to the rotation from the

mobile x-y-z axes until the absolute x-y-z axes. Depending on the perspective adopted by the renderer (see part **9) Examples of Application**), both options may be needed. Note: inverting the rotation is not exactly equivalent to reverting the sign of the yaw, pitch and roll angles.

e) Store/recall presets

Software presets

Presets can be stored and recalled in the bottom right zone. **Note:** presets refer only to software settings, i.e.:

- OSC settings
- receiver settings

Hardware settings are not saved with the presets, but in the head tracker itself! The reason for this is simple: the same software can be used with different head trackers, each of them having its own hardware settings.

The preset 1 will be automatically reloaded next time the program is opened.

Saving/recalling hardware settings

If needed (for example during a firmware update), hardware settings (as well as receiver settings) can be store as text files. See part "**4)b) Load the head tracker firmware**" for more information.

Technical Note 3: Madgwick's algorithm and the *Beta* parameter

Sebastian Madgwick's angle estimation algorithm (see http://x-io.co.uk/open-source-imu-and-ahrs-algorithms/), on which *hedrot* is based, relies on one parameter, the so-called *Beta* coefficient.

If Beta is small, the head tracker depends only on the gyroscope data (quick and reliable if movements, but entails drifts). **Setting Beta to 0 entails drifts, but has however the major interest of providing orientation that do not depend on the magnetometer, if the data from the magnetometer is being biased by ferromagnetic materials and/or magnetic fields.**

If Beta is too high, the head tracker depends only on the accelerometer and the magnetometer (biases in cases of movement, but gives a reliable estimate if no movement). According to Sebastian Madgwick, optimal values of *Beta* are around 0.1, and Beta should remain between 0 and 0.5.

The main modification to Sebastian Madgwick's algorithm in *hedrot* is the introduction of a *variable* beta, which depends on the movement. The purpose of this is to get the best of both situations: beta is high if no movements but falls down as soon as the head tracker is in movement.

Then two parameters are to be set: the "Max Beta" is the static value of Beta (if no movement), and the "Beta variation" determines the influence of the movement on the fall down of Beta. A specific study on the optimal values for both is to be done, but the default values (Max Beta= 2.5, Beta Variation = 1) based on informal hearing tests should work well in most standard cases.

9) Examples of Application

a) Mybino

Mybino (http://www.cmap.polytechnique.fr/xaudio/mybino/) is a free VST binaural monitoring plugin developed by the X-Audio Team from the Ecole Polytechnique's Center for Applied Mathematics (CMAP). It relies on a powerful and optimized engine and a very simple interface.

To use *mybino* with *hedrot*, the following settings are required in *hedrotReceiver*:

- receiver settings:
    - axes references "0: X->right, Y->back, Z->down"
    - rotation order "Yaw-Pitch-Roll (ZYX)"

    - don't invert rotation
- OSC settings
    - don't scale/invert any values
    - send only yaw/pitch/roll (quaternion not needed)

b) IRCAM Spat (Max version)

Spat (http://forumnet.ircam.fr/product/spat-en/) is a huge library developed for years by the Acoustic and Cognitive Spaces Team at IRCAM, Paris, to analyze, manipulate and render sound scenes in a very large number of rendering systems, including binaural.

A very simple demonstration patch (examples/Max+Spat/hedrot+Spat.maxpat") is provided with the distribution of *hedrot* as an example using the Max distribution of the Spat library (version 4). **Note**: this tutorial does not cover the plugin version of the Spat developed by Flux.

To connect *hedrot* to *Spat* version 4, the following settings are required in *hedrotReceiver*:

- receiver settings:
    - axes references "1: X->right, Y->front, Z->up"
    - rotation order "Yaw-Pitch-Roll (ZYX)"
    - **invert rotation**
- OSC settings
    - don't scale/invert any values
    - send only yaw/pitch/roll (quaternion not needed in version 4 of the Spat)

c) Reaper and Ambix

Reaper (http://www.reaper.fm/) is an excellent and very cheap DAW that provides unique routing and channel assignment functions (a track can have up to 64 channels) and can fully controlled thru OSC.

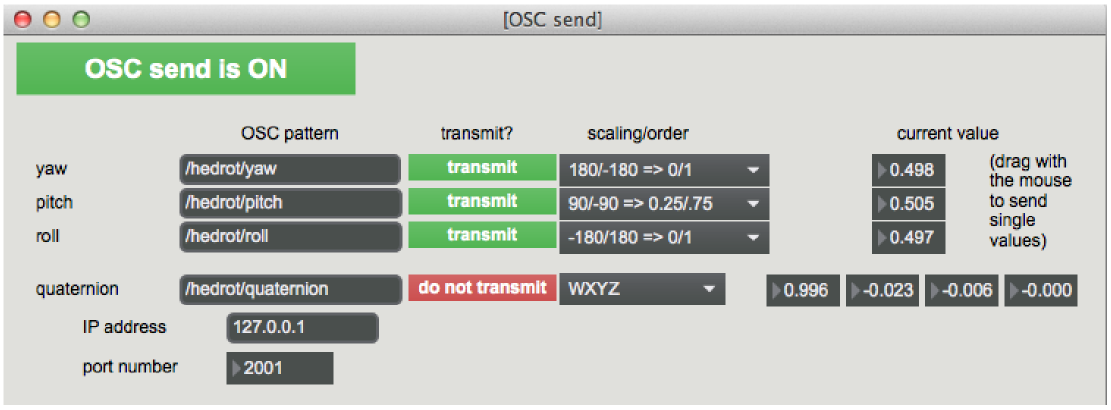
Ambix (http://www.matthiaskronlachner.com/?p=2015) is a free plugin-suite developed by Matthias Kronlachner for creating and manipulating Ambisonics scenes, with the ability to render them in binaural.

This tutorial describes how to connect *hedrotReceiver* with a binaural version of an Ambisonics scene through the plugin ambix_rotator.

Configure *hedrotReceiver*

- receiver settings:
    - axes references "2: X->front, Y->left, Z->up"
    - rotation order "Yaw-Pitch-Roll (ZYX)"
    - **invert rotation**
- OSC settings
    - **invert yaw and scale it to 0/1** (see figure below)

- ○ **invert pitch and scale it to 0.25/.75** (see figure below)
- ○ **don't invert roll and scale it to 0/1** (see figure below)
- ○ send only yaw/pitch/roll (quaternion not needed in version 4 of the Spat)


[OSC send]

OSC send is ON

	OSC pattern	transmit?	scaling/order	current value			
yaw	/hedrot/yaw	transmit	180/-180 => 0/1	0.498			
pitch	/hedrot/pitch	transmit	90/-90 => 0.25/.75	0.505			
roll	/hedrot/roll	transmit	-180/180 => 0/1	0.497			
quaternion	/hedrot/quaternion	do not transmit	WXYZ	0.996	-0.023	-0.006	-0.000

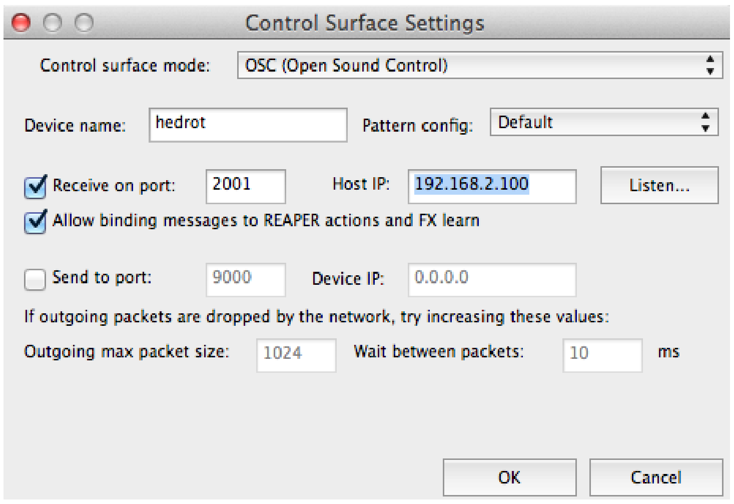
(drag with the mouse to send single values)

IP address: 127.0.0.1

port number: 2001

Configure Reaper

- In "Reaper" menu, click on "Preferences" and then on "Control/OSC/web"
- click on "Add". Select OSC as control surface mode. Give the device a name (for instance "hedrot"), select the same receive port as the send port from hedrot, and click on "allow binding messages to REAPER actions and FX learn", as shown below:


Control Surface Settings

Control surface mode: OSC (Open Sound Control)

Device name: hedrot   Pattern config: Default

[✓] Receive on port: 2001   Host IP: 192.168.2.100   Listen...

[✓] Allow binding messages to REAPER actions and FX learn

[ ] Send to port: 9000   Device IP: 0.0.0.0

If outgoing packets are dropped by the network, try increasing these values:

Outgoing max packet size: 1024   Wait between packets: 10 ms

OK   Cancel

Link *hedrot* with ambix_rotator

The DAW session has to be designed in Ambisonics, ideally in higher order. The purpose of this tutorial is not to explain how to do it. This can be done with the ambix plugins or with other. Important however is to respect the channel ordering (HCN) and normalization standard (SN3D) used by ambix. If the channel ordering and/or normalization does not match, the plugin "ambix_converter" can be used right before the last stage.

The head tracking is used in conjunction with the plugin ambix_rotator that has to be inserted at the very last stage of the session, in the master track, right before the ambisonics to binaural conversion (for example done with the plugin ambix_binaural).

Below (Figure 2) is the screenshot of a very simple session controlled by hedrot. A mono source (track 1) is being encoded in Ambisonics 5th order thanks to the plugin ambix_encoder. In the master track, two plugins are inserted: ambix_rotator (that rotates the scene according to the data provided by *hedrotReceiver*) and ambix_binaural (that converts the scene in binaural format).

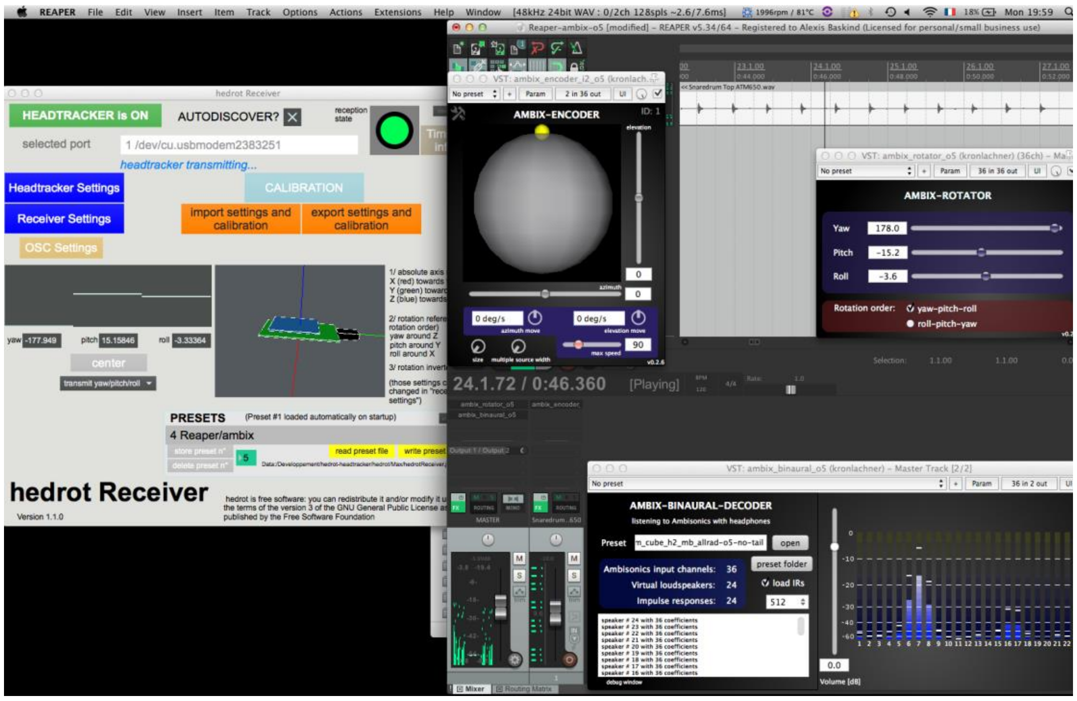


Figure 2: Screenshot of a simple ambisonics in Reaper controlled by *hedrotReceiver*

ambix_rotator is configured with "yaw-pitch-roll" rotation order (same as for *hedrot*).

The next step consists in connecting *hedrot* with ambix_rotator. For this:

- in *hedrotReceiver*, go to "OSC Settings", and stop temporary the transmission for the pitch and roll streams. Only yaw is thus transmitted
- in Reaper/ambix_rotator, click on "Param" (top of the window), then on "FX parameter List => Learn => Yaw". In the field "Command", "/hedrot/yaw" should automatically appear. Click then on "OK"
- Repeat the operation for pitch and roll
- When all three angles are connected to the corresponding parameters of ambix_rotator, enable again the OSC transmission for yaw, pitch and roll in *hedrotReceiver.*