hedrot

Installing and Using the Head Tracker

**Table of Contents**

1) Introduction 2

a) What is hedrot? 2

b) Further Developments 2

c) Licensing and Credits 3

d) Hardware Requirements 3

e) Software Requirements 3

2) How to use Hedrot? 4

3) Assembling the Head tracker 4

a) Required parts 4

b) Schematics 6

c) Setting the multipin connectors on the Teensy board 6

d) Set both cables on GND and +3.3V pins of the GY-85 board 8

e) Connect both cables on the main board, install the daughter board on the main board and solder 8

4) Install/Update the hedrot Firmware in the Teensy 3.1 9

a) Download the latest version of the Teensy loader 9

b) Load the head tracker firmware 9

Important Notice 9

Loading/Updating the Firmware 9

First use of the head tracker 10

5) Axes conventions, positioning the head tracker 10

a) Axes convention 10

b) Positioning the Head tracker on the Headphone 10

6) Starting the Head tracker 10

7) Calibrating the Head tracker 11

a) To show the head tracker raw and calibrated data 11

b) Calibrating the accelerometer 11

c) Calibrating the magnetometer 11

8) Using the *hedrotReceiver* Application 12

a) Starting the head tracker 12

b) Sending the calculated angles via OSC 12

c) Head tracker (hardware) Settings 12

d) Receiver (software) Settings 13

e) Store/recall presets 14

Software presets 14

Saving/recalling hardware settings 14

# Introduction

## What is hedrot?

*Hedrot* (for "head rotation tracker") is a low-cost (less than 40 euros) 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/>.

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 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) is being currently measured.



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

## Further Developments

The next developments and research include among others:

* Measurements of the overall (software+hardware) latency
* A Windows Version
* 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.

## 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/>).

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)

## Hardware Requirements

Head tracker parts:

* 1 Teensy 3 board (tested with versions 3.1 and 3.2)
* 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

## 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>)

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 1.8.1
* Teensyduino 1.35 (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

# 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.

# Assembling the Head tracker

## Required parts

* 1 Teensy 3 board (tested with versions 3.1 and 3.2)
* 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 connection cables



## Schematics



## 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.



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



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



Side view



Rear view

# Install/Update the hedrot Firmware in the Teensy 3.1

## Download the latest version of the Teensy loader

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

## Load the head tracker firmware

### Important Notice

**Before any update of the firmware, 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"
* Drag & drop the current version of the firmware provided with the distribution of hedrot (file "hedrot-firmware\_version\_XX.hex", 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"

# Axes conventions, positioning the head tracker

## Axes convention

Looking at the Teensy board with the USB connector below and on the left side (the GY-85 daughter board is above the Teensy):

* the x axis points to the front
* the y axis points to the right
* the y axis points down

## Positioning the Head tracker on the Headphone

* The head tracker must be positioned on the top of the
* The GY-85 daughter board should be on the top, the USB connector below and on the left side, so that the USB cable is connected on the left side of the board

# 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

# Calibrating the Head tracker

## 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.

## 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

## 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

# Using the *hedrotReceiver* Application

**Note 1:** the head tracker is assumed here to be calibrated. 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"

## 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.

## 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

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

## Head tracker (hardware) Settings

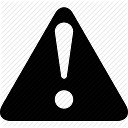
In the window "Headtracker Settings" are shown all the hardware parameters, i.e. the hardware refresh rate ("sample rate", default 1 kHz) and the low-level settings of all 3 sensors.

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.

## 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 1 ms) determines the output refresh rate of the angles. It should be high enough to ensure the head tracker to respond quickly, but should be smaller or equal to the hardware sample period. For instance, if the hardware sample rate is set to 500 Hz, there is no point setting the poll period higher than 2 ms.
* 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" and "Gradient Descent: Beta variation according to movement" determine how the main parameter "Beta" of Sebastian Magdwick's algorithm (see technical explanation below). Keep those values to the standards settings (2.5 and 1) unless you know what you're doing.

**Technical Note: 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* relies among others, relies on one parameter, the so-called *Beta* coefficient.

If Beta is too small, the head tracker depends only on the gyroscope data (quick and reliable if movements, but entails drifts). 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.

## 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.