Collidoscope software setup instructions

(v0.3)

Collidoscope uses the Teensy++ 2.0 (Teensy) microcontroller and the Raspberry Pi 3 model B (Pi) single-board computer. The Teensy gathers all the sensor information and sends it as MIDI messages to the Pi. The Pi runs the visuals and the sound according to the MIDI messages received from the Teensy.

This document is a step-by-step guide on how to get the respective software running on the Pi and the Teensy. The software that runs on a Pi is called **CollidoscopeApp**, whereas the software running in the Teensy is **CollidoscopeTeensy**. Both are available from the source code repository. CollidoscopeApp binary can also be downloaded from the Downloads section.

Running CollidoscopeApp on Raspberry Pi

Operating System

First of all, you need to get your Pi operating system set up and connected to the Internet. Collidoscope has been tested and run on Raspbian GNU/Linux 8 (Jessie), the official operating system distribution of the Pi. You can download Raspbian from the Pi's website and install it according to the instructions provided.

Running commands on a Terminal

A terminal is another way to run programs in a computer. It is very popular in Linux based systems such as the Pi. For installing CollidoscopeApp software, it will be handy to use a terminal to do some needed tweaks to the system. To open a terminal, go to Menu > Accessories > Terminal.

A black window will appear. You type stuff into that window and press enter, and that's how you execute commands on a terminal. As simple as that.

Some commands need to be executed as root, and need to be preceded by the **sudo** keyword.

CollidoscopeApp uses Jack 2 to deliver low latency audio. In order to install Jack, open a terminal and type the following command (note the Pi needs to be connected to the internet):

- **`sudo apt-get install jackd2`**

When Jack is installed you need to edit the configuration file in your home directory. CollidoscopeApp will run the Jack server automatically and the server will read this file to boot with the right configuration. Type the following command on a command line to set the configuration right (all in one line and including the greater-than ('>') sign):

- **`echo "/usr/bin/jackd -P70 -dalsa -r44100 -p128 -n2 -D -Chw:USB -Phw:USB" > ~/.jackdrc`**

Note that this is the configuration used with the Focusrite USB audio interface (See *Introduction to Collidoscope* for more info on the Focusrite). If you're experienced with Jack, you can try different configurations and see how they sound on your set up.
To make sure you typed the command correctly you can open the file *~/.jackdrc* in a text editor, for instance by typing on a terminal:

- **`nano ~/.jackdrc`**

The file should have one line:
/usr/bin/jackd -P70 -dalsa -r44100 -p128 -n2 -D -Chw:USB -Phw:USB

You can also just use the text editor to add the line directly.

Install and Run Collidoscope

To install CollidoscopeApp just download the executable from the repository downloads page into the folder /home/pi. Right click on the file and select *Properties*. Then go in the *Permissions* tab and change *execute* from *Nobody* to *Only Owner*. Alternatively run the following command from the terminal:

- **chmod u+x ~/CollidoscopeApp**

You're then ready to launch the program by running the following command from the terminal:

- **~/CollidoscopeApp <width> <height>**

<width> and **<height>** should be two numbers equal to respectively the width and the height in pixel of your screen. This is to make sure CollidoscopeApp covers the entire screen.

For example, if your screen resolution is 1024x768, you want to run the following command:

- **~/CollidoscopeApp 1024 768**

Exit the CollidoscopeApp

Collidoscope runs full screen and the mouse is disabled so that the pointer does not stand on your way when you play.

In order to exit the app and return to your desktop, just press the *Esc* button.

Running Collidoscope at startup

The Pi is very small and was easily embedded in the Collidoscope unit. You don't need a keyboard and mouse attached to the Pi to run CollidoscopeApp. So you might want to have CollidoscopeApp to run automatically at start up, so as to avoid having to launch it manually with keyboard or mouse.

You can do so by adding an entry for the CollidoscopeApp executable to the window manager autostart file. Just run the following command from the terminal:

- **echo "@/home/pi/CollidoscopeApp <width> <height>" >> ~/.config/lxsession/LXDE-pi/autostart**

Replacing **<width>** and **<height>** with the actual numbers of your screen; again all in one line and including the greater-than sign ('>>'); this time repeated twice, which means: append at the bottom. You can also just use a text editor and add the following line at the bottom of the file *~/.config/lxsession/LXDE-pi/autostart*; which is what the command above does:

@/home/pi/CollidoscopeApp <width> <height>

If at reboot CollidoscopeApp doesn't start automatically, double check with a text editor that you typed the right command and that the line appears only once in the file. And make sure the sound card is plugged in when you boot the Pi.

In order to get the monitor turned on automatically when you start the Pi, you might need to tweak the settings in /boot/config.txt. In particular you might need to change **disable_overscan, hdmi_force_hotplug, hdmi_group** and **hdmi_mode**.

Installing and Running CollidoscopeTeensy on a Teensy++2.0

Setting up of the CollidoscopeTeensy can be done from any computer and not necessarily from the Raspberry Pi. The steps to install the software on the microcontroller are the following:

- Download and install the Arduino Software (IDE) from the Arduino website;
- Download and install Teensy add-on for Arduino Software from Teensy website;
- Open Arduino Software and make sure the following options are set in the Tools menu
    - Board: "Teensy++ 2.0";
    - CPU Speed: "16 Mhz";
    - USB Type: "MIDI";

  The last setting turns the Teensy in a class-compliant USB MIDI device. So you don't need to install any driver in order for it work with the Pi.

- Create a folder named "CollidoscopeTeensy";
- Download CollidoscopeTeensy_new.ino or CollidoscopeTeensy_original.ino from the code repository, if you're building respectively for the new or original version of Collidoscope;
- Place the downloaded file in the CollidoscopeTeensy folder;
- Rename the file in the folder to CollidoscopeTeensy.ino. The name of your Teensy sketch must be the same as the folder containing it;
- Connect the Teensy board to your computer USB port;
- Open CollidoscopeTeensy.ino in the IDE and press the Upload button;
- If prompted by the IDE, push the button on top the Teensy microcontroller

When the software is loaded, connect the Teensy to the Pi. The Teensy will be reading values from its analog and digital inputs and send MIDI messages to CollidoscopeApp running on the Pi.

Calibrating the Teensy Board

After assembling the Collidoscope, you need to calibrate the strip position sensors in order to have the selection following the knob when you scan it left and right. (See the document *Introduction to Collidoscope* for more information on the strip sensors). On the new version of Collidoscope you need to calibrate the strip sensors for the filters as well (vertical knob).

In order to do that you need to modify the Teensy code and enter new boundary values for the sensors. You can find the bits of code where you need to intervene by looking for the **<calibrate>** tag.

Let's look at the code for the wave 1 selection start (the code for wave two works exactly the same way)

```
// add the following line to calibrate

// Serial.println(Jet1_new);

//send MIDI Wavejet 1 [Position Instrument 1]

// <calibrate>
```

```
if (Jet1_new > Jet1_old+jitter_thresh || Jet1_new < Jet1_old-jitter_thresh) {
    int16_t midiVal = constrain( map(Jet1_new, 988, 121, 0, 149), 0, 149 );
    if( midiVal != Jet1_old_MIDI ){
      Jet1_old_MIDI = midiVal;
      usbMIDI.sendPitchBend( midiVal, midi_chan_inst1 );
    }
    Jet1_old = Jet1_new;
    digitalWrite(Pin_MIDIled, HIGH);
 }
```

In the example the map function maps the values from a range of [988, 121] to [0,149].

The range [0,149] are the values that the Teensy sends to the Pi and must remain untouched.

The range [988, 121] are the voltage values that the sensor sends to the teensy; these values might change with a new Collidoscope setup, so that's where intervention is needed.

In order to calibrate the selection start of wave 1 follow these steps:

- Add a print to the serial port of the wave value (in the example the print is shown in a comment at the beginning);
- Move the wave knob all the way to the left;
- Read the sensor value from the serial port terminal. In Arduino IDE you can access the terminal by clicking on Tools->Serial Monitor;
- The values printed tend to oscillate because of the jitter of the sensor. Just take an average value;
- Place the value you read in the code as second parameter of the map function. So supposed you read 950, then you'd change the code to:

  ```
  int16_t midiVal = constrain( map(Jet1_new, 950, 121, 0, 149), 0, 149 );
  ```

  You might have noticed that, in this example, the second parameter which corresponds to the left of the wave is bigger than the third parameter, which corresponds to the right of the sensor. This depends on how the sensor is mounted. In general you can expect either ways: [big , small] or [small big]. This is not a problem though, as the Arduino *map* function does the job for you in mapping the values respecting the polarity of the arguments;

- Move the wave knob all the way to the right;
- Repeat the same process but put the value in the third parameter of the map function
- Remember to remove the prints from the code or the software will be very sluggish.

Repeat the same process with wave 2. The variable look at in the code is called *Jet2_new.* If you're running the CollidoscopeTeensy_new, repeat the same process also for the filter sensors (in the code they're also tagged with <calibrate>).

Compiling CollidoscopeApp on Raspberry Pi

Build Requirements

- Cinder library for Raspberry Pi *android_linux* branch (commit **1bc47a0e9af1f0a0ca00346a85a1dda356c04b91** was successfully used in the building).
- libjack-jackd2-0
- libjack-jackd2-dev
- libasound2-dev

Build Instructions

- Clone Cinder into your Raspberry Pi (assuming cloning to user's home folder (~) from now on). Cinder depends on the boost libraries, use the **`--recursive`** option when cloning or the boost library won't be cloned and Cinder won't compile;
- Download or clone the CollidoscopeApp source code from the repository;
- Copy the CollidoscopeApp source folder in ~/Cinder/samples;
- Download the precompiled Cinder libraries from the Downloads section of the repository;
- Copy the precompiled libraries for Collidoscope in ~/Cinder/lib/linux/armv7/;
- run the following command in the terminal from the ~/Cinder/samples/CollidoscopeApp folder:
  - **`./cibuild -b Debug|Release`**

Select *Debug* or *Release* according to the type of build you want. The debug build might run significantly slower than the release build.

Build Macros

The following macros are defined in CMakeLists.txt

- **NUM_WAVES=2**: if set to 1 it will build a 1 wave Collidoscope;
- **USE_PARTICLES**: if not defined, no particles will be created when the grain duration is greater than 1. This doesn't affect the sound;
- **__LINUX_ALSA__**: for RtMidi to enable MIDI for Linux

Compile your own version of the Cinder library

If you want to compile the Cinder library yourself instead of using the pre-compiled libraries, have a look at the instructions from the official Cinder repository first.

At the time of writing this document, Jack is not supported by the official Cinder distribution. The official Cinder distribution only supports PulseAudio audio out. So you need to integrate Cinder with the files in the *JackDevice* folder from the Collidoscope repository. These have been created specifically to run Collidoscope with Jack.

- Place DeviceManagerJack.h and ContextJack.h in ~/Cinder/include/cinder/audio/linux
- Place DeviceManagerJack.cpp and ContextJack.cpp in ~/Cinder/src/cinder/audio/linux
- In ~/Cinder/src/cinder/audio/Context.cpp you must include the files that use Jack instead of those using PulseAudio. In Context.cpp, replace the code in the first row with the code in the second row in the tables below;
- Note that Cinder is under continuous development, so Context.cpp might change in future and the following substitutions might no longer be good. In general, every line

where PulseAudio appears must be replaced with the respective Jack counterpart. You can spot such lines easily, because they're always between CINDER_LINUX macros.

```
#elif defined( CINDER_LINUX )
	#include "cinder/audio/linux/ContextPulseAudio.h"
	#include "cinder/audio/linux/DeviceManagerPulseAudio.h"
#else
```

```
#elif defined( CINDER_LINUX )
	#include "cinder/audio/linux/ContextJack.h"
	#include "cinder/audio/linux/DeviceManagerJack.h"
#else
```

```
#elif defined( CINDER_LINUX )
		sMasterContext.reset( new linux::ContextPulseAudio() );
#endif
```

```
#elif defined( CINDER_LINUX )
		sMasterContext.reset( new linux::ContextJack() );
#endif
```

```
#elif defined( CINDER_LINUX )
		sDeviceManager.reset( new linux::DeviceManagerPulseAudio() );
#endif
```

```
#elif defined( CINDER_LINUX )
		sDeviceManager.reset( new linux::DeviceManagerJack() );
#endif
```

Build the Cinder library using the cibuild script as per the instructions. The compiled library will be automatically placed in in ~/Cinder/lib/linux/armv7/, so you should be able to compile Collidoscope straight away.

Build Collidoscope for other platforms

The Collidoscope code can be compiled also for Windows and Mac:

- Setup Cinder for your platform (see Cinder documentation);
- Create a new Cinder project;
- Copy all the files from *include* and *src* directory of the CollidoscopeApp directory into the respective folders of your Cinder project;

- Define the macros, used by Collidoscope, in your IDE. Make sure you use the right RtMidi macro for your platform and that you link to the right libraries. More information on compiling RtMidi can be found on the RtMidi tutorial.

Test CollidoscopeApp with the keyboard

To check that CollidoscopeApp is running properly you don't necessarily need all the sensors and MIDI keyboards in place. You can use the computer keyboard to make a selection and loop it on the red wave. Here are the key commands:

Key	**Action**
a	Move the selection left
d	Move the selection right
w	Increase the selection size
s	Decrease the selection size
r	Record new wave
space bar	Start/stop looping
0	Increase grain duration
9	Decrease grain duration