The Muenster IceCube Display Software documentation

1 What you need

- Hardware:
  - 3× Teensy 3.2 connected to computer via USB, power supply and synchronized with each other
  - 1× LED strip (at least) properly connected to one of the Teensys and power supply
  - Power supply (both the Teensys and the LED strips need 5 V)
- Software
  - IceTray software V17-03-00 (`http://code.icecube.wisc.edu/svn/meta-projects/offline-software/releases/V17-03-00/`)
  - Teensyduino (`https://www.pjrc.com/teensy/td_download.html`) and an Arduino version compatible with it (`https://www.arduino.cc/en/Main/Software`)
  - Teensy Loader (`https://www.pjrc.com/teensy/loader.html`)
  - Python package `pyusb` (get it with `pip` AND `pip3`)
  - All the stuff that is in folders `muenster_icecube_display` and `teensy32_stuff`
  - A nice `.i3` file

2 How to set this up

- Test whether `pyusb` works correctly by running Python interactively (should look something like the following):

```
$ python3
Python 3.4.0 ...
>>> import usb
>>> usb.version_info
(1, 0, 0, 'b2)
```

- Add the following rules from the `teensy32_stuff` folder so you can run all scripts as normal user:

```
$ sudo cp 90-icecube.rules /etc/udev/rules.d/
$ sudo groupadd leddisplay
$ usermod -a -G leddisplay [your username]
```

- Load the `.hex` file in the `teensy32_stuff` folder onto all three Teensys by means of the Teensy loader (explained on the Teensy webpage)

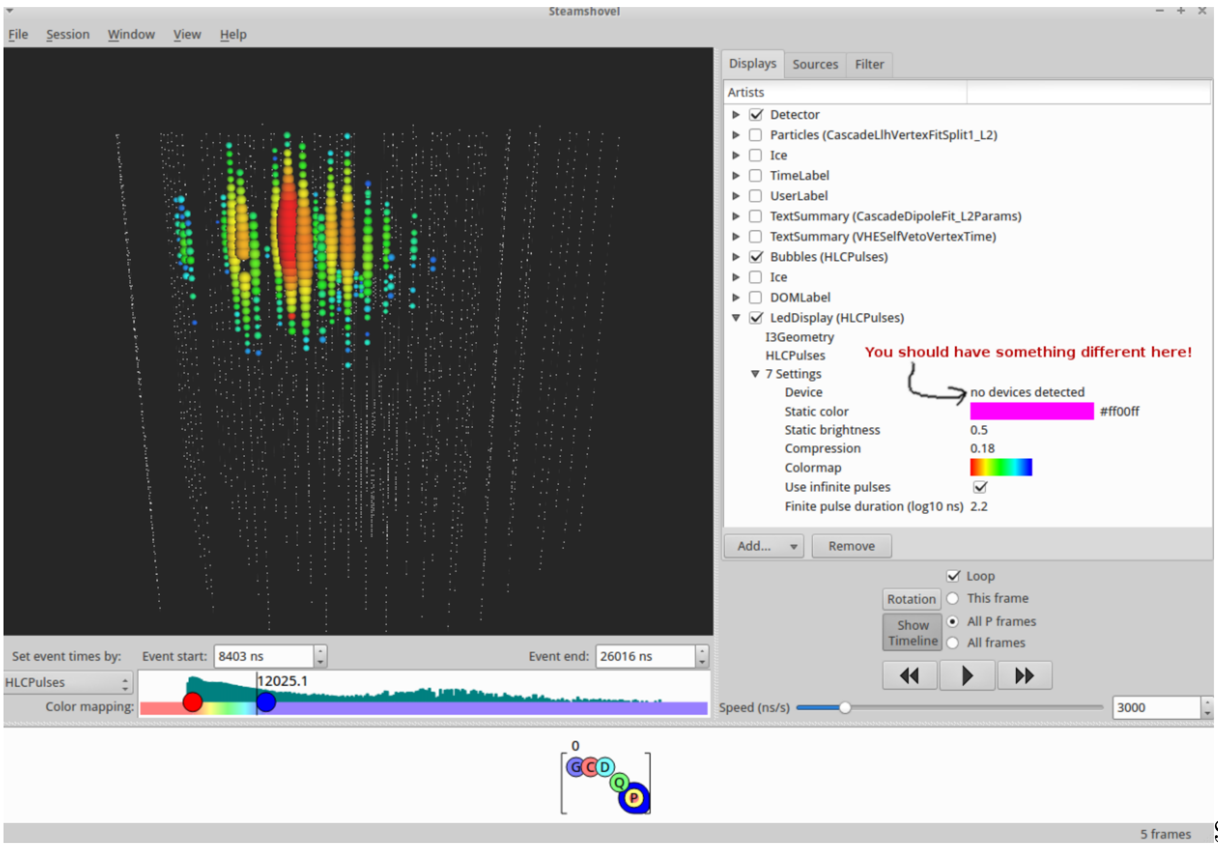
- Run the script `display_info.py` from the `muenster_icecube_display` folder to verify communication with the boards
  ```
  $ python3 display_info.py
  ```
- Now you have to load the display configuration (which strings are located where in the model) onto the Teensys. You do this by means of the `update_eeprom.py` script and the configuration file `display_config_muenster.json`. Should look something like this:
  ```
  $ sudo python3 update_eeprom.py display_config_muenster.json
  Found device
  Please select one of the available device configurations to configure
  the device:
  ```
  (enter `front`, `center` or `back` here)
  ```
  Selected configuration: Configuration uploaded. Please mark the device
  to be able to match it with its serial number or configuration. Please
  reboot the device for the new configuration to take effect.
  ```
- Verify wether writing new configuration worked by running `display_info.py` again; should get a slightly different output.

2.1 Testing and debugging

- The script `test_draw.py` in the `muenster_icecube_display` folder can be used for identifying your channels. The syntax is:
  ```
  $ python3 test_draw.py -m string -c 86 -d 1
  ```
  The above command makes one string light up after the other (assuming 86 strings in total) for one second. Have a look at the script for more options like rgb mode and offset.

2.2 Controlling the display with steamshovel

- Open steamshovel:
  ```
  $ [directory of IceTray software]/build/ ./env-shell.sh
  $ steamshovel
  ```
- It is important at this point that the pyusb package is installed via pip, since steamshovel runs on Python2 and not Python3
- Click the button "Add..." on the right and select `LedDisplay()`
- Select settings `I3Geometry` and e.g. `HLCPulses` (see figure **??**)
- When you run your .i3 file, the onboard LEDs on the Teensys should flicker which means data is transmitted. The LED strips should light up in synch with the steamshovel event.
- HLCPulses if the cleanest service containing the pulses. They are triggered in time therefore only the event is shown. To produce this key (when it is not) and combine several files into only one, please use the script `/preparefiles/processing.py`.



```
☑ Detector
☐ Particles (CascadeLlhVertexFitSplit1_L2)
☐ Ice
☐ TimeLabel
☐ UserLabel
☐ TextSummary (CascadeDipoleFit_L2Params)
☐ TextSummary (VHESelfVetoVertexTime)
☑ Bubbles (HLCPulses)
☐ Ice
☐ DOMLabel
☑ LedDisplay (HLCPulses)
    I3Geometry
    HLCPulses
    7 Settings
        Device                          no devices detected
        Static color                    #ff00ff
        Static brightness               0.5
        Compression                     0.18
        Colormap
        Use infinite pulses             ✓
        Finite pulse duration (log10 ns) 2.2
```

You should have something different here!

$

[directory of IceTray software]/build/ ./env-shell.sh

Figure 2.1: A steamshovel screenshot with a possible configuration.

3 Errors you might see when running it

- After some loops, steamshovel breaks with an error as:
  `$ Could not write frame to controller %s (USB errno 110 (maybe also 32))`
  There is a overlap between the teensy and the computer (probably because your computer is too slow). To solve this, we have to increased the max. time that pyusb allows for comunication between the teensy and the computer. Open the following files with root rights:
  `/usr/local/lib/python2.7/dist-packages/usb/legacy.py`
  Go to the lines 144 and 156, you should be over the member functions `bulkWrite` and `bulkRead` inside class `DeviceHandle`. By default the timeout of both functions is 100 ms. Increase it to 1000 or 10000. This should solve the issue.

- The led sometimes do blink/do something weird and you receive in the shell the following debug alert constantly:
  `$ Rendering too fast, cannot send more than 25FPS`[1]
  Go to:
  `ĨcetrayDirectory7̃src/steamshovel/python/artists/LedDisplay.py`
  You have to modify the parameter `min_delta` of the member function `transmitDisplayBuffer`

[1]The alert is only visible if Icetray was compiled with debug or higher verbosity level (by default it is)

inside the class `DisplayWorker` (line 501 in my version). Increase the render frequency by changing the default value of 1./25 to 1./15 or 1./10. This might not totally solve the problem but it should **improve**. If the problem it is not totally solve, you will still have the debug alert in the shell (still with 25FPS if you didn't change the logger.debug on line 514).

- If you see more errors or are able to solve the last one, please write me to: c_loza01@uni-muenster.de