ipymidicontrols Documentation

Release 0.1.3

Project Jupyter Contributors

Nov 18, 2019

INSTALLATION AND USAGE

Version: 0.1.3

A Jupyter widget for interfacing with MIDI controllers.

Because Chrome is the only browser that implements the Web MIDI API, this package only works in Chrome. Firefox has recent discussion on how to move forward with implementing this standard. The webmidi JavaScript package mentions there is a Firefox plugin that possibly makes this work in Firefox (see https://www.npmjs.com/package/webmidi#browser-support).

Each midi controller needs a custom implementation exposing the interface for that specific midi controller as buttons, knobs, faders, etc. Currently we support the Behringer X-Touch Mini controller, which is currently available for around $60.

CHAPTER

ONE

INSTALLATION

The simplest way to install ipymidicontrols is via pip:

```
pip install ipymidicontrols
```

or via conda:

```
conda install ipymidicontrols
```

If you installed via pip, and notebook version < 5.3, you will also have to install / configure the front-end extension as well. If you are using classic notebook (as opposed to Jupyterlab), run:

```
jupyter nbextension install [--sys-prefix / --user / --system] --py ipymidicontrols

jupyter nbextension enable [--sys-prefix / --user / --system] --py ipymidicontrols
```

with the appropriate flag. If you are using Jupyterlab, install the extension with:

```
jupyter labextension install @jupyter-widgets/midicontrols
```

In JupyterLab, you will also need to install the ipywidgets extension:

```
jupyter labextension install @jupyter-widgets/jupyterlab-manager
```

CHAPTER

TWO

EXAMPLES

2.1 Behringer X-Touch Mini

This notebook can be downloaded here.

Because Chrome is the only browser that implements the Web MIDI API, this package only works in Chrome. Firefox has recent discussion on how to move forward with implementing this standard.

Each midi controller needs a custom implementation exposing the interface for that specific midi controller as buttons, knobs, faders, etc. Currently we support the Behringer X-Touch Mini controller, which is currently available for around $60.

```
In [ ]: from ipymidicontrols import XTouchMini, xtouchmini_ui
        x = XTouchMini()
```

We can work directly with the controls to assign values, listen for value changes, etc., just like a normal widget. Run the cell below, then turn the first knob or press the upper left button. You should see the values below update. Note that the button value toggles when held down, and the light on the physical button reflects this state, where true means light on, false means light off.

```
In [ ]: left_knob = x.rotary_encoders[0]
        upper_left_button = x.buttons[0]
        display(left_knob)
        display(upper_left_button)
```

You can also adjust the values from Python and the changes are reflected in the kernel.

```
In [ ]: left_knob.value = 50
```

2.1.1 Rotary encoders (knobs)

Rotary encoders (i.e., knobs) have a min and max that can be set.

```
In [ ]: left_knob.min=0
        left_knob.max=10
```

Knobs have a variety of ways to display the value in the lights around the knob. If your value represents a deviation from some reference, you might use the `'trim'` light mode. If your value represents the width of a symmetric range around some reference, you might use the `'spread'` light mode.

```
In [ ]: # light_mode can be 'single', 'wrap', 'trim', 'spread'
        left_knob.light_mode = 'spread'
```

We'll set the min/max back to the default (0, 100) range for the rest of the example for consistency with other knobs.

```
In [ ]: left_knob.min = 0
        left_knob.max = 100
```

2.1.2 Buttons

Since the button has a True/False state, and holding down the button momentarily toggles the state, if we set the button to True when it is not held down, we reverse the toggling (i.e., it is now True by default, and pressing it toggles it to False).

```
In [ ]: upper_left_button.value = True
        # Now press the button to see it toggle to false.
```

We can change this toggling behavior in the button by setting the button `mode`. It defaults to `'momentary'`, which means the button state toggles only when the button is held down. Setting `mode` to `'toggle'` makes the button toggle its value each time it is pressed. Run the following cell and press the button several times. Notice how the toggle behavior is different.

```
In [ ]: upper_left_button.mode = 'toggle'
```

Each rotary encoder can also be pressed as a button and the toggle mode can be set as well. Run the cell below and press the left knob.

```
In [ ]: left_knob_button = x.rotary_buttons[0]
        left_knob_button.mode = 'toggle'
        display(left_knob_button)
```

2.1.3 Faders

The fader can send its value to Python and has `min`, `max`, and `value` properties.

```
In [ ]: fader = x.faders[0]
        display(fader)
```

Because the X-Touch Mini does not have motorized faders, the fader cannot be moved to represent a value set from Python. Any value set from Python is overridden by the next fader movement.

2.1.4 Listening to changes

As with any widget, we can observe changes from any control to run a function.

```
In [ ]: from ipywidgets import Output

        out = Output()

        @out.capture()
        def f(change):
            print('upper left button is %s'%(change.new))

        upper_left_button.observe(f, 'value')
        display(out)
```

2.1.5 Linking to other widgets

You can synchronize these widgets up to other widgets using `link()` to give a nicer GUI. Run the cell below and then try turning the left knob or pressing the upper left button. Also try adjusting the slider and checkbox below to see that the values are synchronized both ways.

```
In [ ]: from ipywidgets import link, IntSlider, Checkbox, VBox
        slider = IntSlider(description="Left knob", min=left_knob.min, max=left_knob.max)
        checkbox = Checkbox(description="Upper left button")
```

```
        link((left_knob, 'value'), (slider, 'value'))
        link((upper_left_button, 'value'), (checkbox, 'value'))

        display(VBox([slider, checkbox]))
```

This package includes a convenience function, `xtouchmini_ux()`, to link each control up to a slider or checkbox widget in a GUI that roughly approximates the physical layout.

```
In [ ]: xtouchmini_ui(x)
```

2.1.6 Experimenting with options

Let's set various controls to explore the available button and knob light modes, as well as some random values to see what they look like on the controller.

```
In [ ]: for b in x.buttons:
            b.mode='toggle'
        for b in x.rotary_buttons[:4]:
            b.mode='toggle'
        for b in x.rotary_buttons[4:]:
            b.mode='momentary'
        for b in x.side_buttons:
            b.mode='momentary'
        for b, mode in zip(x.rotary_encoders, ['single', 'single', 'trim', 'trim', 'wrap', 'wrap', 's
            b.light_mode = mode
```

```
In [ ]: # Set some random values
        import secrets
        for b in x.buttons:
            b.value=secrets.choice([False, True])
        for b in x.rotary_encoders:
            b.value = secrets.randbelow(101)
```

2.1.7 Clearing values

Finally, let's clear all of the values.

```
In [ ]: # Clear all values
        for b in x.buttons:
            b.value = False
        for b in x.rotary_buttons:
            b.value = False
        for b in x.rotary_encoders:
            b.value = 0
```

```
In [ ]:
```

CHAPTER

THREE

DEVELOPER INSTALL

To install a developer version of ipymidicontrols, you will first need to clone the repository:

```
git clone https://github.com/jupyter-widgets/midicontrols
cd midicontrols
```

Next, install it with a develop install using pip:

```
pip install -e .
```

If you are planning on working on the JS/frontend code, you should also do a link installation of the extension:

```
jupyter nbextension install [--sys-prefix / --user / --system] --symlink --py␣
↪ipymidicontrols

jupyter nbextension enable [--sys-prefix / --user / --system] --py ipymidicontrols
```

with the appropriate flag. Or, if you are using Jupyterlab:

```
jupyter labextension install .
```