Getting started with Piano HAT

Introduction

Piano HAT will let you explore your musical prowess, or use those 16 capacitive touch buttons to control any project you might conceive.

This guide will walk you through setting up your Piano HAT and teach you how to detect button presses and light up the LEDs.

Installing

We've created a super-easy installation script that will install all prerequisites and get your Piano HAT up and running in a jiffy. To run it fire up Terminal which you'll find in Menu -> Accessories -> Terminal on your Raspberry Pi desktop like so:

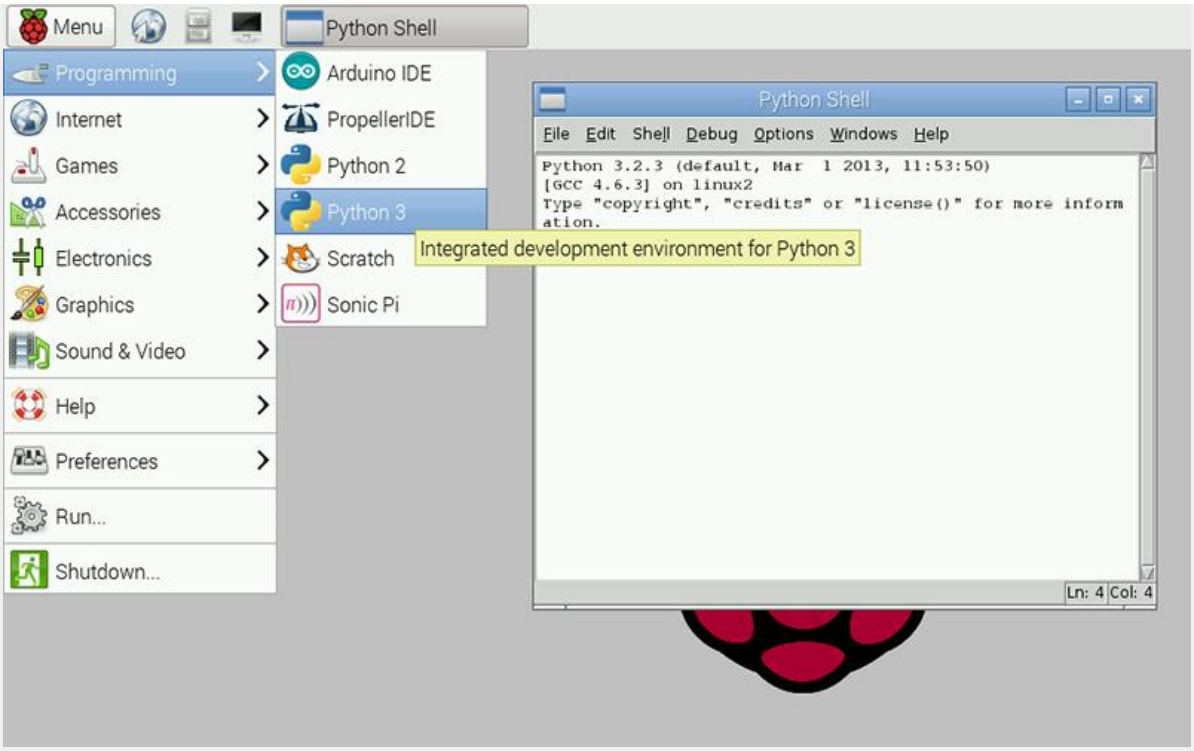
In the new terminal window type the following and follow the instructions:

```
curl -sSL get.pimoroni.com/pianohat | bash
```

If you choose to download examples you'll find them in /home/pi/Pimoroni/pianohat, you can explore these now if you want to see what Piano HAT can do, but let's start from scratch!

Your First Piano HAT

First things first, you need to fire up IDLE, an environment for writing Python scripts. You can find IDLE under the "Programming" category in the Raspberry Pi menu. We'll use "Python 3":



The window that pops up is called a REPL- Read, Eval, Print, Loop. It's a Python command-line indicated by the ">>>" prompt. You can type in individual lines of Python here and they will be evaluated immediately. It's great for testing ideas in Python and exploring modules.

We want to write a script, though, so let's create a new file by going to File -> New File or by pressing CTRL+N.

The new blank window that pops up is our Python script, the very first thing we'll want to do is import the Piano HAT library we installed earlier. Do that by adding this line:

```
import pianohat
```

We'll also need `time`, a fairly common Python module, for later, so import that too:

```
import time
```

Next, we want to create a function to handle key presses. This function will be called every time you tap a piano key on Piano HAT:

```
def handle_note(channel, pressed):
    if pressed:
        print("You pressed key {}".format(channel))
    else:
        print("You released key {}".format(channel))
```

It's a simple function that takes two arguments. The first, `channel`, is a number from 0 to 12 that indicates which of the 13 piano keys is being pressed. The second, `pressed`, is a boolean (true/false) value which indicates if the key has been pressed `True` or released `False`.

By default the Piano HAT library doesn't know it should call this function when a key is pressed, so we'll now register it:

```
pianohat.on_note(handle_note)
```

We're almost there, but not quite. If we ran what we've written so far then our Python script would just exit before we've had the chance to do anything. Piano HAT performs all its magic in the background so we've got to do something to keep Python busy in the meantime!

For now, we'll just settle for an empty loop:

```
while True:

    time.sleep(0.001)
```

Now it's time to fire it up. You can run your new Python script by hitting F5. You'll be prompted to save you script, give it a unique and descriptive name and avoid names like "pianohat.py" because these will confuse Python no end! Try something like "my-first-pianohat.py".

Once saved you'll see the REPL window again, this time your script will be running. When you press a key on the Piano you should see a message.

Making Sounds

Now we've got a basic example running, let's do something useful with it. Since it's a piano, the most obvious thing to do is make some sounds.

Before we write any code, we'll need some sounds to play. We'll be using PyGame which works well with .wav files, so let's grab some. A good place to grab these is freesound.org. I've picked a few out which should help us make an instrument:

- Moo - http://www.freesound.org/people/Bird_man/sounds/275154/
- Woof - http://www.freesound.org/people/delphidebrain/sounds/235890/
- Meow - http://www.freesound.org/people/Npeo/sounds/203121/

Save these files, or some short .wav files of your choosing, alongside your Python script. Give them short, memorable names so you can remember what they're called when we load them into Python.

Setting up the mixer

To play sounds we'll be using PyGame, but only the sound mixer. Let's import it by adding it to the list of imports at the top:

```
from pygame import mixer
```

Now we need to do some basic setting up of the PyGame sound system. We should add this stuff before we set up pianohat, so place it before `pianohat.on_note(handle_note)`:

```
mixer.init(22050, -16, 2, 512)
```

You don't have to worry too much about these values, but I'll explain what they are for the sake of completeness:

- `22050` - This is the frequency, or the number of samples per second that make up your sound
- `-16` - This is the bit depth, the size of each sample. -16 means "16 bit, signed" or "-32,768 to 32,767"
- `2` - This should be 1 to denote mono, and 2 to denote stereo
- `512` - This is the buffer size, it should generally be quite small since a large (the default is 4096) buffer will delay your sounds and a small buffer will distort them. I picked 512 through trial and error.

Next we set up the number of sounds the mixer can play simultaneously, this will default to 8 but sometimes we want more. We've got 13 notes on Piano HAT so let's set up 13 channels:

```
mixer.set_num_channels(13)
```

Loading a sound

Next we need to load one of our sounds. I'll assume you've downloaded Moo and named it `moo.wav` in your home directory (`/home/pi`) but you can change these paths to reflect what you've downloaded.

Add this line, again before `pianohat.on_note(handle_note)`:

```
my_sound = mixer.Sound("/home/pi/moo.wav")
```

That was easy! PyGame handles all the hard stuff for us, and just gives us a sound object which we can call the `play` method of to play our sound.

Now let's zip back and modify our `handle_note` function to play the sound when a key is pressed:

```
def handle_note(channel, pressed):
    if pressed:
        my_sound.play(loops=0)
        print("You pressed key {}".format(channel))
    else:
        print("You released key {}".format(channel))
```

We use `loops=0` because we want our sound to play only once when we hit a key.

Putting it all together

Your final code should look something like this, and play relentless mooing or barking sounds when you mash the keys on Piano HAT:

```
import pianohat
import time
from pygame import mixer


def handle_note(channel, pressed):
    if pressed:
        my_sound.play(loops=0)
        print("You pressed key {}".format(channel))
    else:
        print("You released key {}".format(channel))


mixer.init(22050, -16, 2, 512)
mixer.set_num_channels(13)

my_sound = mixer.Sound("/home/pi/moo.wav")


pianohat.on_note(handle_note)


while True:
    time.sleep(0.001)
```

Next Steps

Hurrah! You've learned the basics of loading and playing sounds with PyGame and Piano HAT. The next step is to find a sensible way to store the 13 sounds you'll need for each key. I recommend using lists and you'll see this technique in the `simple_piano.py` example.

Your list of sounds might end up being:

```
my_sound_files = [
    "/home/pi/moo.wav",
    "/home/pi/bark.wav",
    "/home/pi/meow.wav",
    "/home/pi/squawk.wav",
    "/home/pi/bark.wav",
    ...
]
```

And you could load them like this:

```
my_sounds = [mixer.Sound(sound_file) for sound_file in my_sound_files]
```

And then play them like this:

```
def handle_note(channel, pressed):
    if pressed:
        my_sounds[channel].play(loops=0)
```

Give it a go!