Realtime Arduino Sensor Monitoring with Matplotlib

30 AUGUST 2014 on Python, Arduino

While procrastinating reading datasheets and writing libraries for my various sensors I decided to come up with a solution for monitoring sensor outputs in realtime. I figure this could be useful for future debugging (or maybe just for cool graphics). I think most people choose *processing* for this sort of work, but I would prefer to just stay in *python* for now.

Solution

Let's plot the sensor output using Matplotlib. Using `matplotlib.pyplot` and `matplotlib.animation`, it's possible to create realtime graphs.

Let's start simple by generating fake sensor data with `random.randint()`. Real sensor data can be easily plugged into the resulting code.

The main driver for this is the `matplotlib.animation.FuncAnimation(figure, updateFunc)`.

Some simple housekeeping:

```
import matplotlib.pyplot as plt
import matplotlib.animation as anim
from collections import deque
import random

MAX_X = 100   #width of graph
MAX_Y = 1000  #height of graph
```

Next, we will create a `deque` that will hold the y coordinates for all the points on the graph. As we append new data points to the `deque`, old ones will pop off of the front automatically using `maxlen`.

```
# intialize line to horizontal line on 0
line = deque([0.0]*MAX_X, maxlen=MAX_X)
```

Now, we create the `pyplot` figure and associate the figure with the `FuncAnimation`.

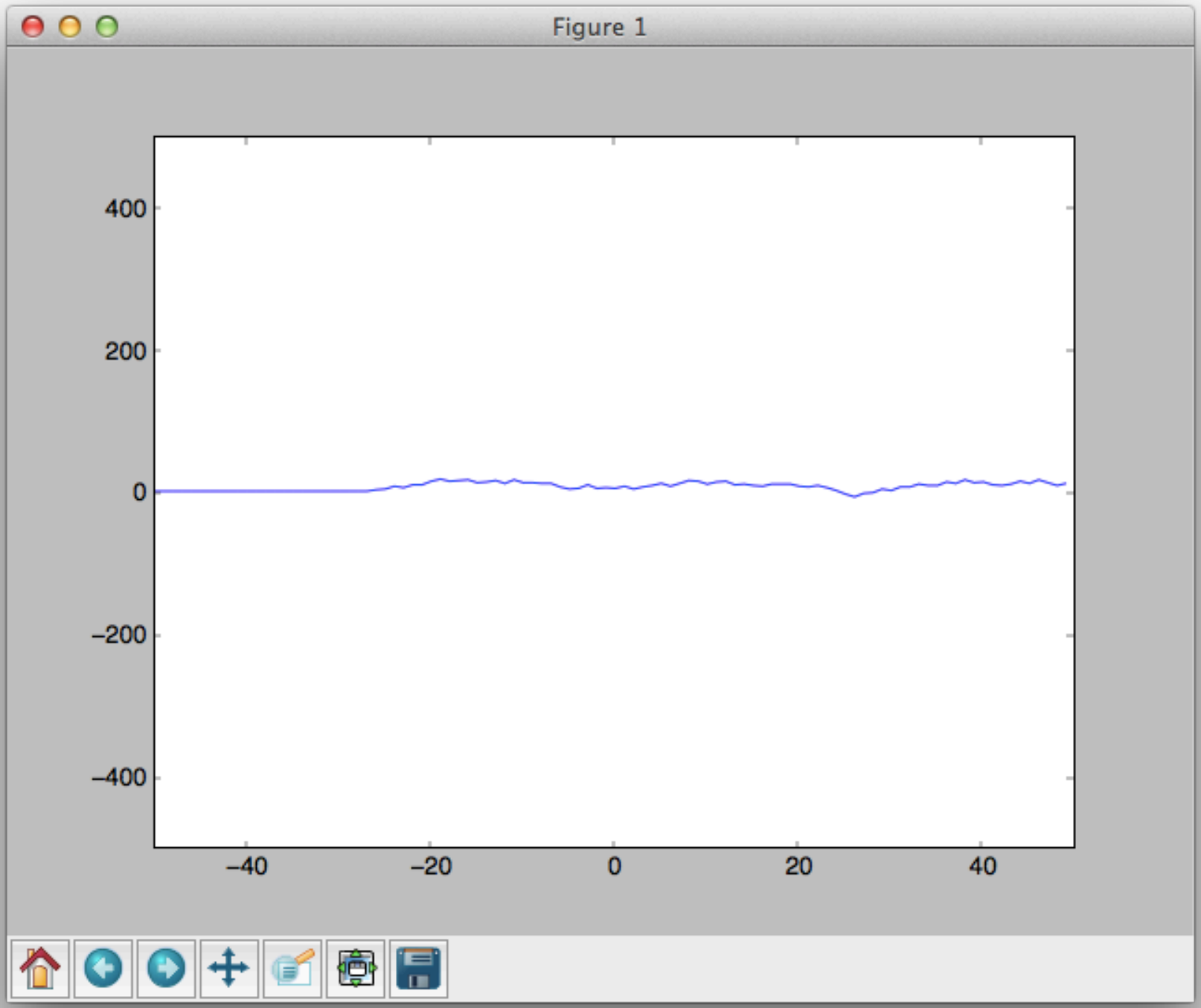
```
fig = plt.figure()
# make the axes revolve around [0,0] at the center
# instead of the x-axis being 0 - +100, make it -50 - +5
# ditto for y-axis -512 - +512
a = plt.axes(xlim=(-(MAX_X/2),MAX_X/2), ylim=(-(MAX_Y/2)
# plot an empty line and keep a reference to the line2d
l1, = a.plot([], [])
ani = anim.FuncAnimation(fig, update, fargs=(l1,), inter
```

Finally, we have to define the `update` function to append new *sensor* measurements to the deque and update the `line2d` of our graph to mirror our `deque`.

```
def update(fn, l2d):
    #simulate data from serial within +-5 of last datapc
    dy = random.randint(-5, 5)
    #add new point to deque
    line.append(line[MAX_X-1]+dy)
    # set the l2d to the new line coords
    # args are ([x-coords], [y-coords])
    l2d.set_data(range(-MAX_X/2, MAX_X/2), line)
```

Oh! Almost forgot... We actually have to show the `pyplot`!

```
plt.show()
```



Here is the full code. Please note that all of the code is included in the snippets above; however, the snippets actually appear out of order due to function and variable declarations needing to be declared before use.

```
import matplotlib.pyplot as plt
import matplotlib.animation as anim
from collections import deque
import random
```

```
MAX_X = 100   #width of graph
MAX_Y = 1000  #height of graph

# intialize line to horizontal line on 0
line = deque([0.0]*MAX_X, maxlen=MAX_X)

def update(fn, l2d):
    #simulate data from serial within +-5 of last datapo
    dy = random.randint(-5, 5)
    #add new point to deque
    line.append(line[MAX_X-1]+dy)
    # set the l2d to the new line coords
    # args are ([x-coords], [y-coords])
    l2d.set_data(range(-MAX_X/2, MAX_X/2), line)

fig = plt.figure()
# make the axes revolve around [0,0] at the center
# instead of the x-axis being 0 - +100, make it -50 - +5
# ditto for y-axis -512 - +512
a = plt.axes(xlim=(-(MAX_X/2),MAX_X/2), ylim=(-(MAX_Y/2)
# plot an empty line and keep a reference to the line2d
l1, = a.plot([], [])
ani = anim.FuncAnimation(fig, update, fargs=(l1,), inter


plt.show()
```

Conclusion

It is easy to create a realtime graph using Matplotlib. Hooking up actual sensor data to this would be as easy as replacing the random generation with serial communication with an arduino or any other data source!

Joseph Rollins

Share this post

Read more posts by this author.

READ THIS NEXT

Talking to Arduino with Python

Programming your Arduino with the provided Arduino IDE is pretty simple. Using the IDE's serial monitor to watch output...

YOU MIGHT ENJOY

React Tetris Clone

I have been really interested in trying out React.js recently. My previous post about matrices in Clojure originated...

.Code() © 2016

Proudly published with **Ghost**