Python API calls using Flask and WebSockets (Using DroneKit and JSON)

This tutorial will give you a clear idea and a good start for making your customized Drone APIs for calling the particular features and functions from your flight controller, i.e Pixhawk 4. FYI: Pixhawk runs on PX4 firmware.

Steps for writing your first API call using Web sockets and Flask

- Make a directory where you will be performing all the tasks.

```
mkdir drones
```

- And now we need to install dependencies needed for performing the following call via web sockets.
- On Linux you will first need to install **pip** and **python-dev**:

```
sudo apt-get install python-pip python-dev
```

- **pip** is then used to install **dronekit** and **dronekit-sitl**. Mac and Linux may require you to prefix these commands with sudo.

```
sudo pip install dronekit
sudo pip install dronekit-sitl
```

- **Installing Flask: It is a microframework for Python.**

```
pip install Flask
```

- **Installing the Web Framework - Socket.io**

```
pip install flask-socketio
```

Getting started with code!

To run the server, we're going to use a package called Flask. The requirements we are building into this package are the following:

```
Flask>=0.10
Flask-SocketIO>=1.0
dronekit>=2.0.0,<=2.99999
```

"Flask" will be used to serve web requests. "dronekit" allows "Flask" to communicate with our vehicle. "Flask-SocketIO" allows us to use web sockets to talk to a web browser.

In our file **server.py, this creates flask application**

```
app = Flask(__name__)
app.config['SECRET_KEY'] = 'secret!'
socketio = SocketIO(app)
```

And this routes to our .HTML file

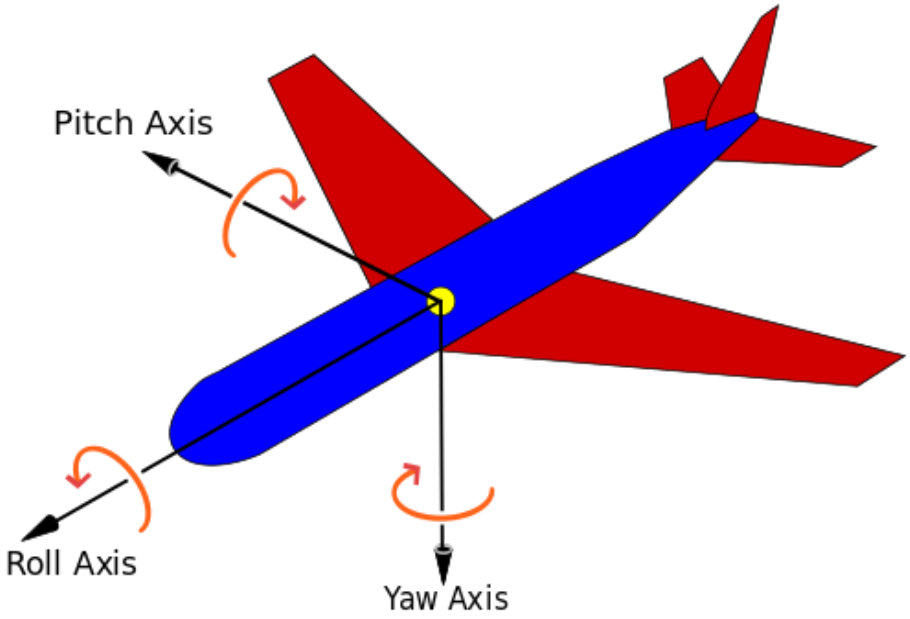
```
@app.route('/')
def index():
    return render_template('index.html')
```

Before we start building the code we need to understand that Vehicle state information is exposed through vehicle attributes. DroneKit-Python currently supports the following "standard" attributes:

Vehicle.version,Vehicle.location.capabilities,
Vehicle.location.global_frame,Vehicle.location.global_relative_frame,
Vehicle.location.local_frame, Vehicle.attitude,Vehicle.velocity, Vehicle.gps_0,
Vehicle.gimbal, Vehicle.battery, Vehicle.rangefinder,
Vehicle.ekf_ok,Vehicle.last_heartbeat, Vehicle.home_location,
Vehicle.system_status, Vehicle.heading,Vehicle.is_armable, Vehicle.airspeed,
Vehicle.groundspeed, Vehicle.armed, Vehicle.mode

You can access one of the attributes and read the documentation which briefly explains about the Dronekit.

Example: Here we're gonna be finding the Pitch, Yaw, and Roll in real time using the `Vehicle.attitude` attribute.



The parameters are pre-defined in radians. The parameters are:

```
pitch - Pitch in radians
yaw - Yaw in radians
roll - Roll in radians
```

So, we're going to make a function which is **pyr (Pitch, Yaw and Roll) having vehicle as an argument.**

```
def pyr(vehicle):
    while True:
        time.sleep(.5)
        atti = vehicle.attitude
        if atti:
            socketio.emit('pyr_status', {
                "Pitch": atti.pitch,
                "Yaw": atti.yaw,
                "Roll": atti.roll,
                })
        else:
            socket.emit('pyr_status', None)
```

This will be called from the main() which creates a vehicle thread and starts it on the server.

What will **if __name == '__main': do?**

- **It will connect the server to the vehicle using a serial port connection.**
- **You can check the serial port using this command**

```
ls /dev/tty*
```

- **It will start the vehicle thread**
- **And finally, it will start the server at a 0.0.0.0 and port 8080**

```
if __name__ == '__main__':
    target = sys.argv[1] if len(sys.argv) >= 2 else '/dev/ttyACM0'
    print 'Connecting to ' + target + '...'
    vehicle = dronekit.connect(target)
    vehiclethread = threading.Thread(target=pyr, args=(vehicle,))
    vehiclethread.start()
    socketio.run(app, host="0.0.0.0", port=8080)
```

Onto the HTML side,

- We are going to create an object Socket which connects to the socketio that's running currently.
- After it connects it checks if the server is emitting the the function or not?
- Then we use JSON Stringify to convert JavaScript object or value to a **JSON** string.
- This helps us get the values in real time in the string format.

```
var socket = io.connect('//' + document.domain + ':' +
location.port);
socket.on('connect', function() {
  socket.on('pyr_status', function (arg) {
    $('h1').text('Current drone location:')
    $('pre').text(JSON.stringify(arg, null, '  '));
  });
});
```

Complete Code: server.py

```
from flask import Flask, render_template
from flask_socketio import SocketIO
import dronekit
import sys
import socket
import threading
import time
import signal

socket.socket._bind = socket.socket.bind
def my_socket_bind(self, *args, **kwargs):
    self.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
    return socket.socket._bind(self, *args, **kwargs)
socket.socket.bind = my_socket_bind
# okay, now that that's done...

app = Flask(__name__)
app.config['SECRET_KEY'] = 'secret!'
socketio = SocketIO(app)

@app.route('/')
def index():
    return render_template('index.html')

def pyr(vehicle):
    while True:
        time.sleep(.5)
        atti = vehicle.attitude
        if atti:
            socketio.emit('pyr_status', {
                "Pitch": atti.pitch,
                "Yaw": atti.yaw,
                "Roll": atti.roll,
                })
        else:
            socket.emit('pyr_status', None)
```

```
if __name__ == '__main__':
    target = sys.argv[1] if len(sys.argv) >= 2 else '/dev/ttyACM0'
    print 'Connecting to ' + target + '...'
    vehiclethread = threading.Thread(target=pyr, args=(vehicle,))
    vehiclethread.start()
    socketio.run(app, host="0.0.0.0", port=8080)
```

HTML Code: **index.html**

```
<!DOCTYPE html>

<html>
  <head>
    <title>Welcome to MAVServer</title>
    <style>

html, body { margin: 0; padding: 0; }
body { padding: 15px 20px; font-family: Helvetica, Arial, sans-serif;
}
pre { font-size: 1.2em; }

    </style>
    <script src="/static/jquery.min.js"></script>
    <script src="/static/socket.io.min.js"></script>
    <script>
var socket = io.connect('//' + document.domain + ':' +
location.port);
socket.on('connect', function() {
  socket.on('pyr_status', function (arg) {
    $('h1').text('Attitude:')
    $('pre').text(JSON.stringify(arg, null, '  '));
  });
});

    </script>
  </head>
  <body>
```

```
    <h1>Waiting for connection...</h1>
    <pre></pre>
  </body>
</html>
```

STEPS TO RUN THE CODE:

- Link to Github - https://github.com/SmitKabrawala/webserver.git
- Download or clone the repo.
- Connect the Pixhawk with the USB into the Raspberry Pi3.
- Open the terminal and type this:

```
cd drones
git clone https://github.com/SmitKabrawala/webserver.git
python server.py
```

This will start the server at http://0.0.0.0:8080