Flask Microframework Illustration

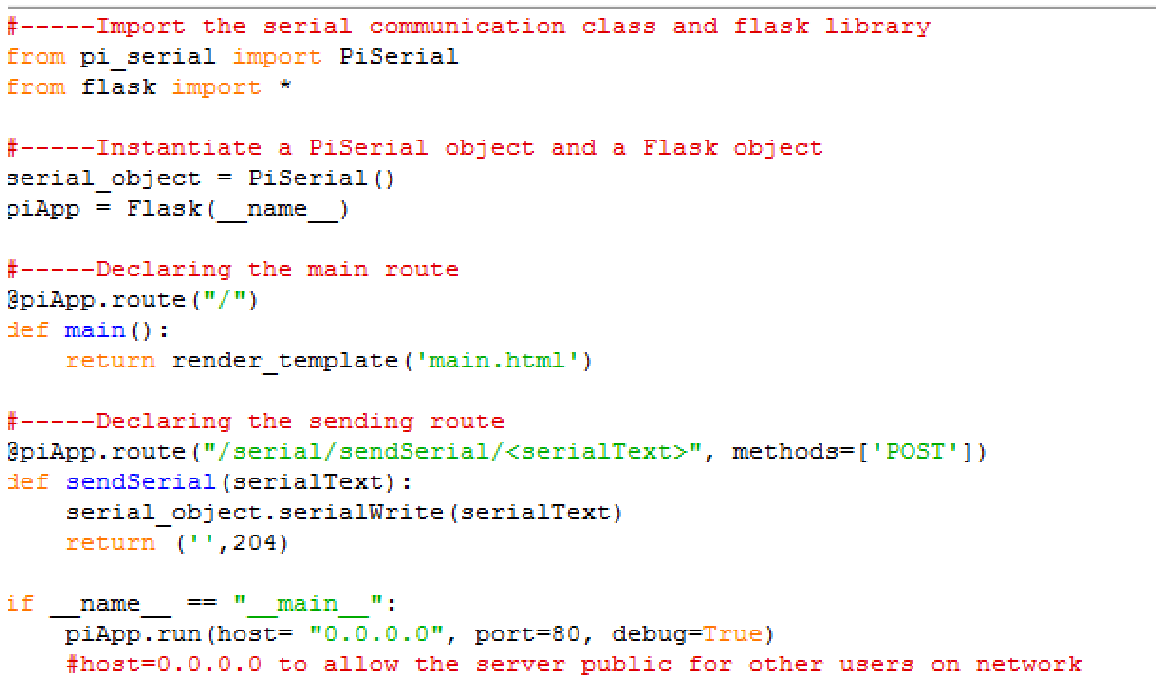
George Saman

FLask is a Python microframework, aimed at small applications with simple requirements. Since it uses Python as the programming language (which is what the RaspberryPi is using) and it's ease of use, it made perfect sense to use it.

The way Flask works is by assigning **functions** to specific URL's, each URL is called a **Route**. Each Route can have any HTTP method, methods tells the server what the client (browser) wants to ***do*** with the requested page. in our case we are interested in the "**GET**" and "**POST**" methods.

- The GET method tells the server to just **get** the information stored on that page and send it.
- The POST method tells the server that it wants to **post** some new information to the URL and that the server must ensure the data is stored and stored only once.

When a browser is accessing the RaspberryPi's ip address (ex, 192.168.0.3/), it is accessing (GET method) the **main route** and in response to that request,the Flask server will return the result of the **function** that is bound to that route. Refer to Figure 1.



```
#-----Import the serial communication class and flask library
from pi_serial import PiSerial
from flask import *

#-----Instantiate a PiSerial object and a Flask object
serial_object = PiSerial()
piApp = Flask(__name__)

#-----Declaring the main route
@piApp.route("/")
def main():
    return render_template('main.html')

#-----Declaring the sending route
@piApp.route("/serial/sendSerial/<serialText>", methods=['POST'])
def sendSerial(serialText):
    serial_object.serialWrite(serialText)
    return ('',204)

if __name__ == "__main__":
    piApp.run(host= "0.0.0.0", port=80, debug=True)
    #host=0.0.0.0 to allow the server public for other users on network
```

Figure 1

@piApp.route("/") is the main route and the function tied to it is called **main** (refer to ***def main()***), main returns the result of **render_template** function which is a flask **built in** function that renders **HTML** files. Flask assumes the templates to be rendered are stored in a "**templates**" folder. This is why main is calling render_template with an argument "main.html" and no directory specified.

As you may noticed, there is another route that is targeting **/serial/sendSerial/<serialText>** URL.
The **<>** declares a **variable** with the name **serialText,** this variable holds whatever data to be sent from the HTML page to Flask. The function bound to that route is called **sendSerial** (refer to ***def sendSerial()***), Flask will execute this function once it makes sure that the requested URL has a method of type "POST" (refer to ***methods=['POST']***), if it does,then the content of sendSerial will be passed to a python function that will send this data serially through the PI (refer to ***serial_object.serialWrite(serialText*)**). Figure 2 describes this flow in a block diagram.

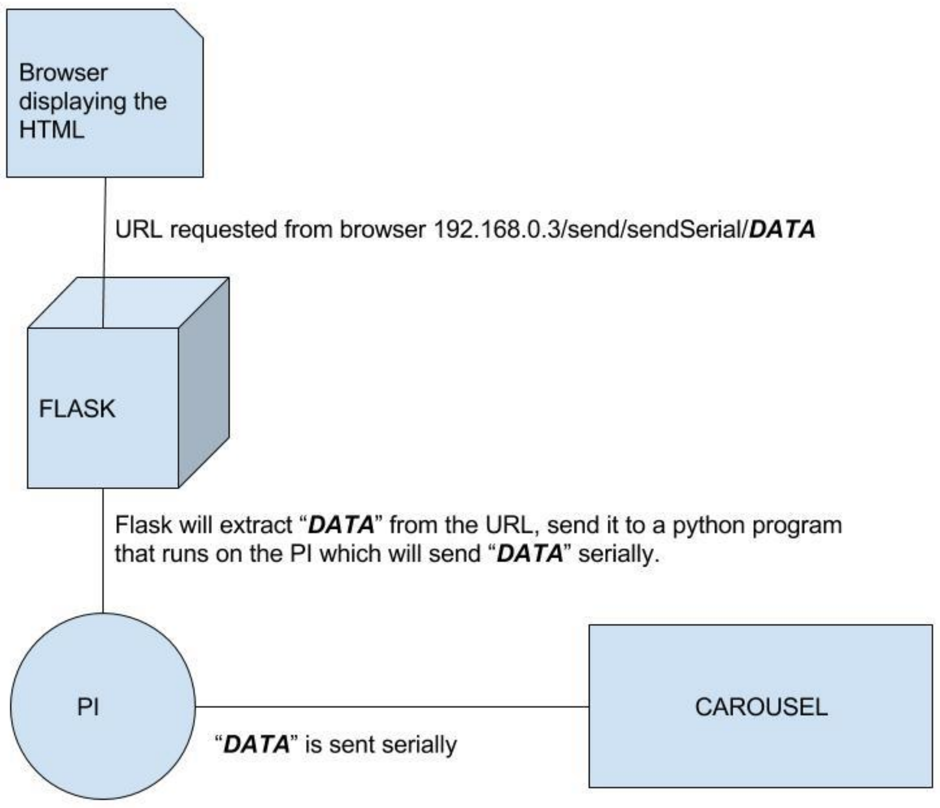

Browser displaying the HTML
URL requested from browser 192.168.0.3/send/sendSerial/***DATA***
FLASK
Flask will extract "***DATA***" from the URL, send it to a python program that runs on the PI which will send "***DATA***" serially.
PI
CAROUSEL
"***DATA***" is sent serially

Figure 2

The last two lines of code, allows the server to be accessed publicly, meaning you can access it outside the PI environment, for ex, Your laptop on the same network.