**README.md** 6.35 KB

# IOT Hackathon Project: Steam Guages Driven by a Flight Simualtor


## UNIVERSITY OF WASHINGTON PROFESSIONAL & CONTINUING EDUCATION (UW PCE)

### Internet of Things Certificate Program IOT110

### Leo Salemann leos@uw.edu leo.salemann@me.com


### 12/21/17


(/LeoSalemann/iot110-leos/raw/master/Lab10/FinishedProduct.png)

## Overview

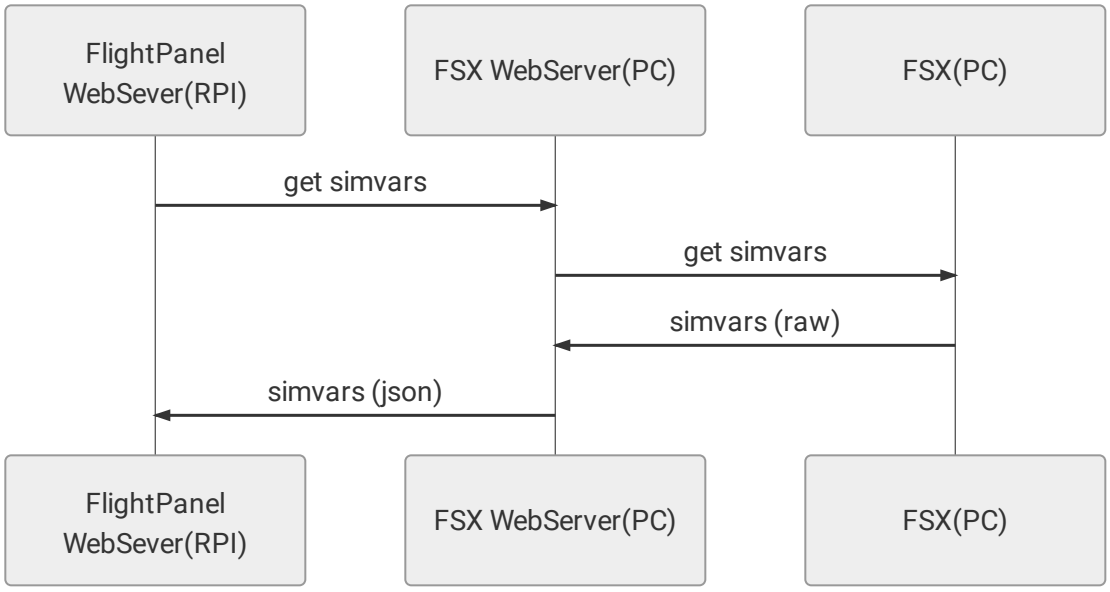
**What's it do?** Read airspeed and other flight varables from a pc-based flight simulator, send parameters to a Rasberry Pi, have the Pi drive a physical guage via stepper motor.

**How's it work?** The PC has MS Flight Simuator (FSX Steam Edition) and a web server that can query simulation variables (SIMVARS) and package them as a JSON document. The Rasberry PI has another webserver that queries the PC one for the simvars.json, parses out the airspeed, translates knots to steps, and turns the stepper motor.

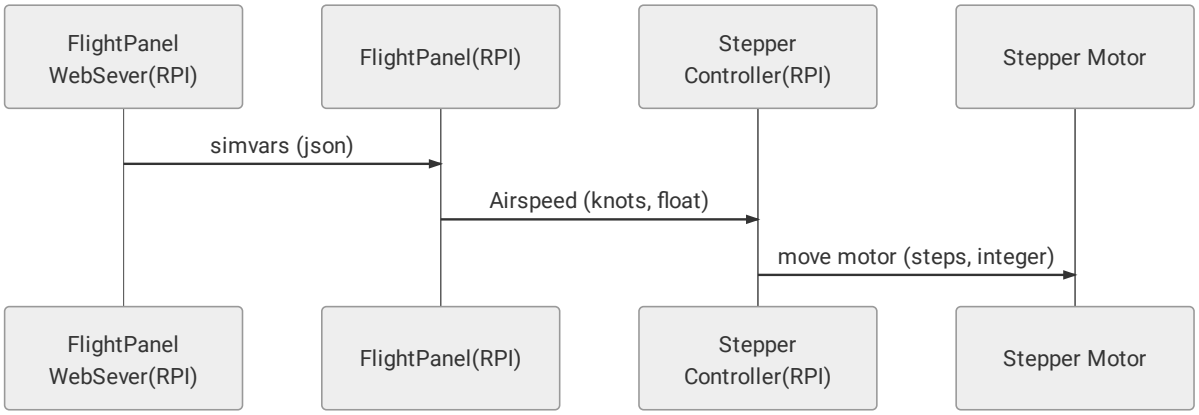
**Why'd you do it this way?** Been a flight-simmer for a few years, wanted to have physical steam guages. Also wanted to learn some IOT, so this let me do both. Sure, a USB-based solution hosted direclty on the PC, talking to an arduino would be faster; but I wanted to learn how to do this over the web.

## Architecture

### Get Simulation Variables (SIMVARS) from the Flight Simualtor (FSX)




### Extract Airspeed (Knots), convert to steps, run the Stepper Motor




## Ingredients, Hardware

- Mid-level Laptop/Desktop capable of runing Win10, PowerShell, Microsoft Flight Simulator Steam Edition
  - I was able to do this on a 2014 MacBook Pro, with Win10 runnin on Parallels

- CantaKit Raspberry Pi 3 Ultimate Starter Kit (https://www.canakit.com/raspberry-pi-3-ultimate-kit.html)
  - Parts I actually used include the Pi itself, the breadboard, the Cobbler (pi-to-breadboard interface), power supply and ribbon cable.
  - For wiring I used a plain wire jumper kit (https://www.circuitspecialists.com/mjw-70b.html?otaid=gpl&gclid=CjwKCAiA4ILSBRA0EiwAsuuBLfF3Cd0fLwPkN3y9zROTXxlRZYytcF6NThUi-8taZgmMHvGGcCrS-xoC-joQAvD_BwE) for on-breadboard wiring and a bonded female/male jumper cable (https://www.adafruit.com/product/1954)to get from the breadboard to the stepper motor.

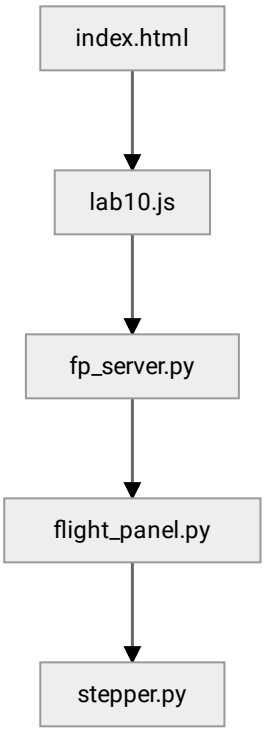
- Adafruit TB6612 1.2A DC/Stepper Motor Driver Breakout Board ID 2448 (https://www.adafruit.com/product/2448)
- Adafruit Automotive Gauge Stepper Motor - x27.168 ID 2424 (https://www.adafruit.com/product/2424)
- Artmind Paper Mache Round Box (http://www.michaels.com/paper-mache-round-box-by-artminds/10329027.html)

## Ingredients, Software

- Microsoft Flight Simulator Steam Edition (FSX SE) (http://store.steampowered.com/app/314160/Microsoft_Flight_Simulator_X_Steam_Edition/)
- FSX RESTful API (PowerShell) (https://github.com/LeoSalemann/fsx), forked from paruljain/fsx (https://github.com/paruljain/fsx)
- UW PCE IOT 110 Lab 7: Actuators (python, javascript, html, css) (https://gitlab.com/iot110/iot110-student/blob/master/Labs/Lab7/setup.md)
- Gauge face from Matton Sébastien's jQuery-Flight-Indicators (https://github.com/sebmatton/jQuery-Flight-Indicators) (printed to hardcopy)

## Details

Basic stepper motor control comes from IOT 110 Lab 7, stepper.py (https://gitlab.com/LeoSalemann/iot110-leos/blob/master/Lab10/stepper.py), which provides functions to move the motor a specific number of steps, read current positon etc. Flight panel modeling comes from flight_panel.py (https://gitlab.com/LeoSalemann/iot110-leos/blob/master/Lab10/flight_panel.py) which provides an interface that lets the caller communicate with flight instruments in terms of feet and knots instead of motor steps. The actual web interface is provided by fp_server.py (https://gitlab.com/LeoSalemann/iot110-leos/blob/master/Lab10/fp_server.py) which uses Flask (http://flask.pocoo.org) to provide HTTPP GET/SET routes to communicate with the flight panel over http. There's a bit of javascript code in lab10.js (https://gitlab.com/LeoSalemann/iot110-leos/blob/master/Lab10/static/js/lab10.js) that "cracks the whip" by querying the fsxWebServer for simulation variables, parsing out the airpseed, and sending that to the flight panel. Finally, index.html (https://gitlab.com/LeoSalemann/iot110-leos/blob/master/Lab10/templates/index.html) provides a web page to check on airpseed and inspect the full simvar json document.




**Note:** Many of the "GET" HTTPP routes actually perform "set-like" functions. It may look confusing, but the idea is to set an airspeed indicator through a simple *curl http://:5000/set-airspeed/100* instead of a tedious *curl --data 'kias=100' http://:5000/set_airpseed*

## Startup Sequence & Operation

### On the PC

1. Start up FSX SE
2. Start up the PowerShell SimConnect WebServer (fsxWebServer.ps1)
3. Connect to the PI (Putty, Ubuntu/ssh, VNC, etc.)

### On the PI

1. Start up the Pi Web Server (python fp_server.py)

### Back on the PC

1. Open the Pi Web Client (pi' IP addr:5000)
2. Begin flying.
3. The Airspeed Indicator needle should move in response to aircraft speed. Dive to increase airspeed; climb to decrease.

## Ideas for Future Enhancements

- [ ] **All python, no javascript** The routine that does the actual FSX query is in Javacript. Better if it was in python like everything else.

- [ ] **Servos instead of steppers** Steepers require four wires; Servos only need two. Could handle twice as many guages (or needles, specifically) by swithing to servos.
- [ ] **Support more guages** Handling more "single needle" guages such as vertical speed indicator and engine rpm would be easy. Multi-needle such as an altimiter would be harder. Attitude indicator would be probably the hardest.
- [ ] **Support a full Cessna 172 panel** Could very well be where ambition transitions into masochism for this architecture. Would provide an intriguing pretence for driving multiple PI's from a single flight simualtor.