Pipeline for .JSON File Analysis in Manufacturing Tests

This notebook will outline the coding steps to possibly automate the visualizations and analysis of the manufacturing tests performed with the FLUKE ProSim 8.

In [70]:
```
import os
import sys

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import json
```

In [71]:
```
# Step 1: select a folder containing the data files
data_folder = "/Users/alessiotamborini/Documents/Avicena/LeakageTests/Data/S

# visualize the name of the files in the folder
data_files = sorted([file for file in os.listdir(data_folder) if file.endswi
print(f"Number of data files in the folder: {len(data_files)}")
print("Data files in the folder:")
for i, file in enumerate(data_files):
    print(f"    {i}: {file}")
```

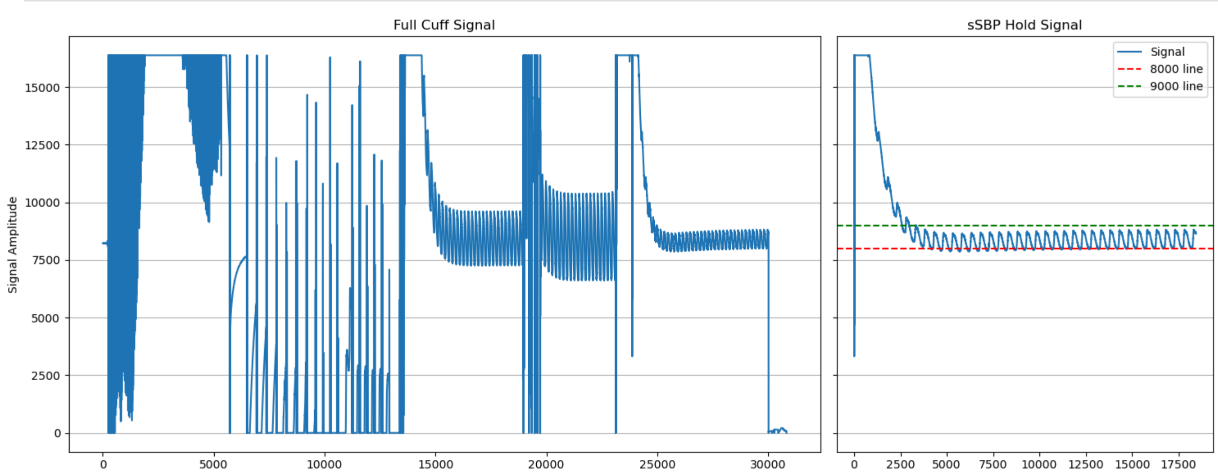
```
Number of data files in the folder: 12
Data files in the folder:
    0: CAA001_VenCuff1 PAA023 RUN 1 60bpm 1234567890.json
    1: CAA001_VenCuff1 PAA023 RUN 2 60bpm 1234567890.json
    2: CAA001_VenCuff1 PAA023 RUN 3 60bpm 1234567890.json
    3: CAA001_VenCuff1 PAA023 RUN 4 60bpm 1234567890.json
    4: CAA001_VenCuff1 PAA023 RUN 5 60bpm 1234567890.json
    5: CAA001_VenCuff1 PAA023 RUN 6 60bpm 1234567890.json
    6: CAA001_VenCuff2 PAA023 RUN 1 60bpm 1234567890.json
    7: CAA001_VenCuff2 PAA023 RUN 2 60bpm 1234567890.json
    8: CAA001_VenCuff2 PAA023 RUN 3 60bpm 1234567890.json
    9: CAA001_VenCuff2 PAA023 RUN 4 60bpm 1234567890.json
    10: CAA001_VenCuff2 PAA023 RUN 5 60bpm 1234567890.json
    11: CAA001_VenCuff2 PAA023 RUN 6 60bpm 1234567890.json
```

In [72]:
```
# Step 2: load the data files
data = {}
for file in data_files:
    with open(os.path.join(data_folder, file), 'r') as f:
        data[file] = json.load(f)
sample_data = data[data_files[0]]
print(f"Sample data keys:\n{list(sample_data.keys())}")
```

```
Sample data keys:
['val_mbp', 'inflation', 'hold_ssbp_cuff', 'hold_mbp_bpkit', 'recording_id',
'batt_lvl_cuff', 'hold_ssbp_ekg', 'hold_mbp_cuff', 'hold_ssbp_bpkit', 'hold_
mbp_ekg', 'hold_dbp_ekg', 'tester_info', 'val_sbp', 'ekg_id', 'val_hr', 'sub
ject_id', 'cuff_id', 'hold_dbp_cuff', 'timestamp_record', 'hold_dbp_bpkit',
'val_dbp', 'site_id', 'batt_lvl_ekg']
```

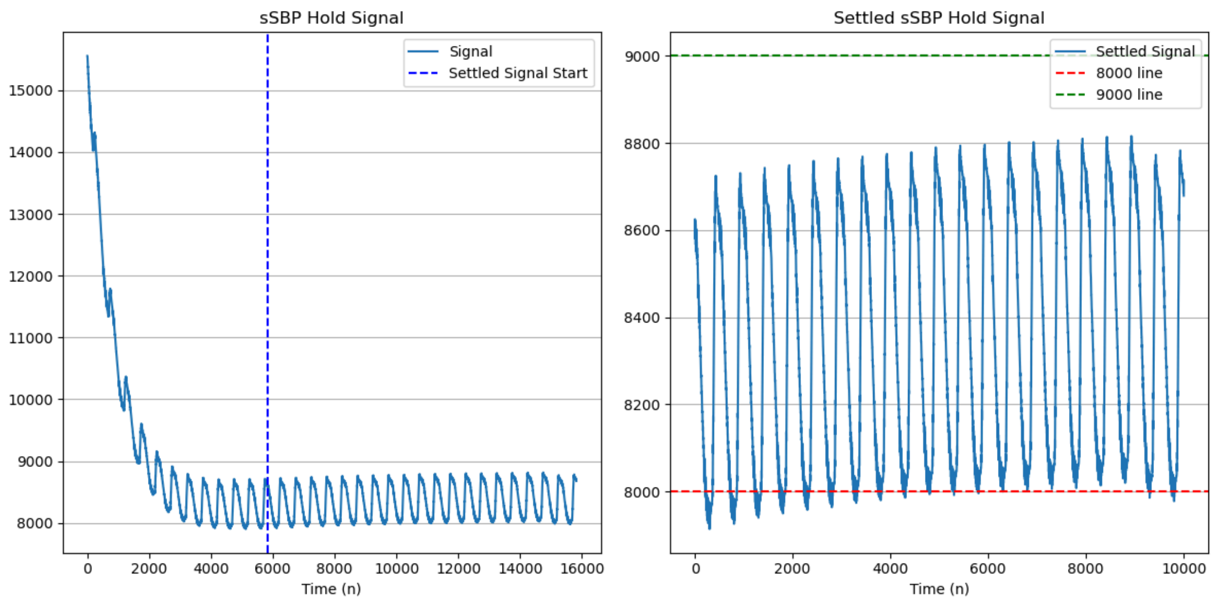
In [73]:

```python
# Step 3: visualize the data run and sSBP hold to assess if in boundaries

# select the full signal data
tester_info = sample_data['tester_info']
cuff_data = tester_info['cuff_data']
cuff_values = np.array(cuff_data['cuff_values'])
time_values =np.array(cuff_data['time'])

# select the sSBP hold data
ssbp_hold = sample_data['hold_ssbp_cuff']
length = len(ssbp_hold)
ssbp_hold_data = cuff_values[-length:]

# compute last drop
gap = len(ssbp_hold_data) - np.argmax(np.abs(np.diff(ssbp_hold_data)))
ssbp_hold_data = cuff_values[-(length+gap):-gap]

# generate a display figure
fig, ax = plt.subplots(1, 2, figsize=(15, 6), gridspec_kw={'width_ratios': [
ax[0].plot(time_values, cuff_values)
ax[0].grid(axis='y')
ax[0].set_title('Full Cuff Signal')
ax[0].set_xlabel('Time (n)')
ax[0].set_ylabel('Signal Amplitude')
ax[1].plot(ssbp_hold_data, label='Signal')
ax[1].axhline(y=8000, color='r', linestyle='--', label='8000 line')
ax[1].axhline(y=9000, color='g', linestyle='--', label='9000 line')
ax[1].grid(axis='y')
ax[1].set_title('sSBP Hold Signal')
ax[1].set_xlabel('Time (n)')
ax[1].legend()
plt.tight_layout()
plt.show()
```



In [75]:
```
# Step 5: summarize results across all files in a table

results = []
for i, (key, item) in enumerate(data.items()):

    # select the full signal data
    tester_info = item['tester_info']
    cuff_data = tester_info['cuff_data']
    cuff_values = np.array(cuff_data['cuff_values'])
    time_values =np.array(cuff_data['time'])

    # select the sSBP hold data
    ssbp_hold = sample_data['hold_ssbp_cuff']
    length = len(ssbp_hold)
    ssbp_hold_data = cuff_values[-length:]

    # compute last drop to cutoff signal
    gap = len(ssbp_hold_data) - np.argmax(np.abs(np.diff(ssbp_hold_data)))
    ssbp_hold_data = cuff_values[-(length+gap):-gap]

    # cutoff the signal after the if drop from the max value ~95% volts
    max_value = np.max(ssbp_hold_data)
    threshold = 0.95 * max_value
    cutoff_index = np.where(ssbp_hold_data > threshold)[0][-1] + 1
    ssbp_hold_data = ssbp_hold_data[cutoff_index:]

    # look only at the signal that has settled
    # for now assume its the last 10000 samples
    if len(ssbp_hold_data) > 10000:
        ind = len(ssbp_hold_data) - 10000
        arr = ssbp_hold_data[ind:]

    # determine if the signals drop to within the boundaries
    thres_8000 = np.any(ssbp_hold_data < 8000)
    thres_9000 = np.any(ssbp_hold_data < 9000)
    cond = thres_8000 and thres_9000

    # generate a plot to visualize the settled signal
    if i == 1:
        fig, ax = plt.subplots(1, 2, figsize=(12, 6))
        ax[0].plot(ssbp_hold_data, label='Signal')
        ax[0].axvline(x=ind, color='b', linestyle='--', label='Settled Signa
        ax[1].plot(arr, label='Settled Signal')
        ax[1].axhline(y=8000, color='r', linestyle='--', label='8000 line')
        ax[1].axhline(y=9000, color='g', linestyle='--', label='9000 line')
        ax[0].grid(axis='y')
        ax[1].grid(axis='y')
        ax[0].set_title('sSBP Hold Signal')
        ax[0].set_xlabel('Time (n)')
        ax[1].set_title('Settled sSBP Hold Signal')
        ax[1].set_xlabel('Time (n)')
        ax[0].legend()
        ax[1].legend()
        plt.tight_layout()
        plt.show()
```

```python
    # store the results
    result = {
        'file_name': key,
        'thres_8000': thres_8000,
        'thres_9000': thres_9000,
        'cond': cond
    }
    results.append(result)
results_df = pd.DataFrame(results)
results_df
```



Out[75]:

	file_name	thres_8000	thres_9000	cond
0	CAA001_VenCuff1 PAA023 RUN 1 60bpm 1234567890....	True	True	True
1	CAA001_VenCuff1 PAA023 RUN 2 60bpm 1234567890....	True	True	True
2	CAA001_VenCuff1 PAA023 RUN 3 60bpm 1234567890....	True	True	True
3	CAA001_VenCuff1 PAA023 RUN 4 60bpm 1234567890....	True	True	True
4	CAA001_VenCuff1 PAA023 RUN 5 60bpm 1234567890....	True	True	True
5	CAA001_VenCuff1 PAA023 RUN 6 60bpm 1234567890....	True	True	True
6	CAA001_VenCuff2 PAA023 RUN 1 60bpm 1234567890....	True	True	True
7	CAA001_VenCuff2 PAA023 RUN 2 60bpm 1234567890....	True	True	True
8	CAA001_VenCuff2 PAA023 RUN 3 60bpm 1234567890....	True	True	True
9	CAA001_VenCuff2 PAA023 RUN 4 60bpm 1234567890....	True	True	True
10	CAA001_VenCuff2 PAA023 RUN 5 60bpm 1234567890....	True	True	True
11	CAA001_VenCuff2 PAA023 RUN 6 60bpm 1234567890....	True	True	True

In [ ]: