Com-202: Signal Processing

Python Notebooks and JupyterLab

Python for Signal Processing

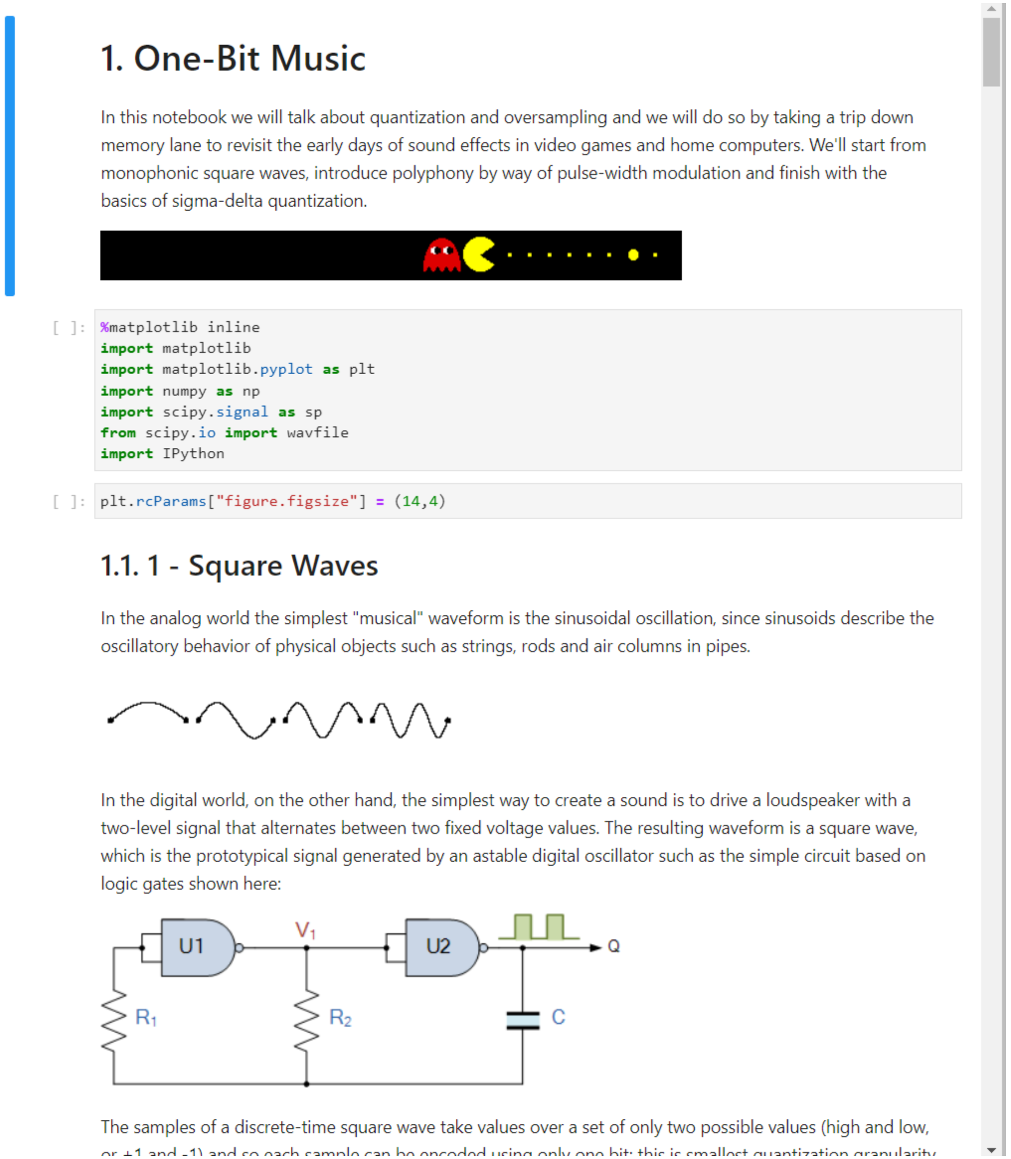
- there’s no better way to understand signal processing than to build signal processing applications

coding = understanding

- we will be using Python:
    - it’s a real programming language (unlike Matlab)
    - it’s free and open-source (unlike Matlab)
    - it has great numerical libraries (NumPy, SciPy, Matplotlib)
    - works great in Jupyter notebooks!

Computational Notebooks

- Literate computing:
  - mix code, comments, rich media
  - plots and charts computed inline
- Coding diary:
  - keep track of progress and attempts
  - share your work, collaborate
- Reuse and repurpose:
  - presentations, slideshows
  - tutorials, interactive books



1. One-Bit Music

In this notebook we will talk about quantization and oversampling and we will do so by taking a trip down memory lane to revisit the early days of sound effects in video games and home computers. We'll start from monophonic square waves, introduce polyphony by way of pulse-width modulation and finish with the basics of sigma-delta quantization.

```
[ ]: %matplotlib inline
     import matplotlib
     import matplotlib.pyplot as plt
     import numpy as np
     import scipy.signal as sp
     from scipy.io import wavfile
     import IPython
```

```
[ ]: plt.rcParams["figure.figsize"] = (14,4)
```

1.1. 1 - Square Waves

In the analog world the simplest "musical" waveform is the sinusoidal oscillation, since sinusoids describe the oscillatory behavior of physical objects such as strings, rods and air columns in pipes.

In the digital world, on the other hand, the simplest way to create a sound is to drive a loudspeaker with a two-level signal that alternates between two fixed voltage values. The resulting waveform is a square wave, which is the prototypical signal generated by an astable digital oscillator such as the simple circuit based on logic gates shown here:

The samples of a discrete-time square wave take values over a set of only two possible values (high and low, or +1 and -1) and so each sample can be encoded using only one bit: this is smallest quantization granularity

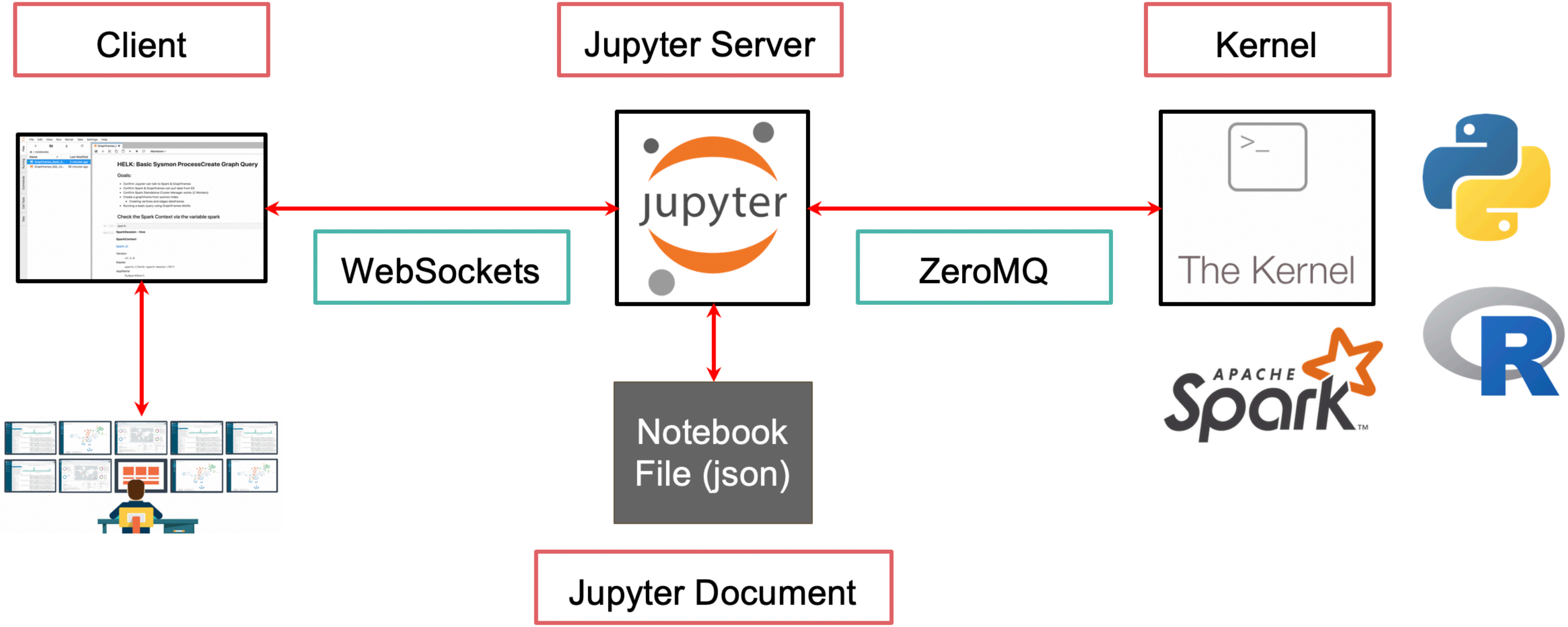
Jupyter

- JupyterLab:
  - notebook authoring application
  - browser-based front-end
  - compatible with many languages

- You can work anywhere:
  - run locally or remotely
  - frictionless integration with code repos

Open source, interactive data science and scientific computing across over 40 programming languages.

Jupyter architecture



Client

Jupyter Server

Kernel

WebSockets

ZeroMQ

Notebook File (json)

Jupyter Document

The Kernel

notebooks

Name	Last Modified
Graphframes_Basic_S...	2 minutes ago
Graphframes_SQL_ini...	36 minutes ago

HELK: Basic Sysmon ProcessCreate Graph Query

Goals:

- Confirm Jupyter can talk to Spark & Graphframes
- Confirm Spark & Graphframes can pull data from ES
- Confirm Spark Standalone Cluster Manager works (2 Workers)
- Create a graphframe from sysmon index
  - Creating vertices and edges dataframes
- Running a basic query using GraphFrames Motifs

Check the Spark Context via the variable spark

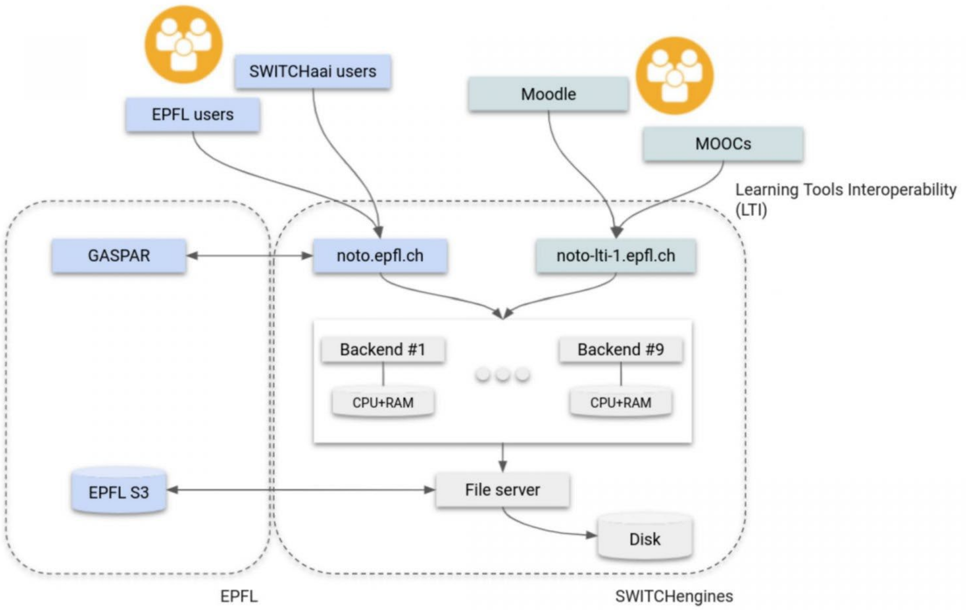
```
In [1]: spark
```

```
Out[1]: SparkSession - hive

SparkContext

Spark UI

Version
    v2.3.0
Master
    spark://helk-spark-master:7077
AppName
    PySparkShell
```

Your options

- Install and run locally on your PC
  - easiest way: install Anaconda
  - roll your own: `pip install jupyterlab`
  - Jupyter Extension for Visual Studio
  - …

- Run remotely:
  - EPFL's NOTO
  - mybinder.org
  - Google's Colab
  - …



Signal Processing Notebooks

- we plan to release 6 notebooks during the semester
- all notebooks will be available on Github
- links in Moodle, pointing to the NOTO framework

- weekly lab sessions to play with signal processing in Python and ask questions
- notebooks contain coding exercises
- there will be Python-based questions in the homework and in the final exam