Integrating Vernier Sensor-Based Data in Pyscript

Angel, Eric, Hari
PHY-432
Final Project Proposal

April 15, 2025

I. Project Overview

This proposal outlines the development of a user-friendly, browser-based interface using PyScript to analyze motion data from Vernier Motion Sensors. The system will process CSV files containing position vs. time data, allowing for numerical computation of velocity and acceleration, as well as visualization of kinematic and energy graphs. We want to extend this project later to explore energy dissipation in a damped, and force driven vertical spring-mass system, to predict paramters like spring constant, coefficient of damping etc.

II. Project Objectives

- **Sensor Integration:** Interface Vernier Motion Sensors to record and export position-time data as CSV files.
- **Data Analysis Tool:** Develop a PyScript-based HTML interface to:
  - Upload CSV files
  - Parse and visualize position-time data
  - Compute and plot velocity and acceleration via numerical differentiation
- **Educational Application:**
  Design and implement Kinematic analysis using a motion cart
- **(Stretch-Goal) Open-Access Web Deployment:** Ensure browser-based access requiring no local Python installation, leveraging PyScript and HTML.
  - Use Vernier Motion Sensors to record data directly to PyScript application.
- **(Stretch Goal) Data Collection through Simulations:** Use pyscript simulations with adjustable parameters to generate visuals and data for Data Analysis tool.

III. Technical Implementation

Hardware

- Vernier Motion Detector (Go Direct Motion)
- Tumble-Buggy Car
- Track and cart system
- (Stretch Goal) Spring and mass assembly

Software

- PyScript Application for Python execution in browser
- Python data acquisition/generation tool (Use prebuilt csv file extraction steps from Pyscript to get CSV files)
- HTML for interface
- Python/PyScript libraries: `NumPy`, `Pandas`, `Matplotlib`, `PyDom`

Workflow

In short, the function of our program

1. Motion data is recorded and exported as a CSV file.
2. CSV file is uploaded via the HTML interface.

Implementation Steps

1. Position a Vernier Motion Sensor at one end of the track.
2. Move the car or cart along the track under uniform or non-uniform conditions.
3. Use data acquisition tool to record position vs. time data and export as CSV.
4. Upload the CSV to the created PyScript HTML interface.
5. PyScript executes Python code in-browser to process and visualize the data.
6. Pyscript program computes the velocity using the central difference method and derives acceleration analytically (if applicable).
7. Pyscript creates and displays plots in interface for:
    - position vs. time.
    - velocity vs. time.

- For position and velocity vs. time graphs:
  - Can apply linear regression graphically
  - Can yield equation for linear regression
    - Yields values with units for slope
    - Yields values for with units for y-intercept
- acceleration vs. time. (optional)
- (stretch goal) velocity vs. position. (optional)
  - PyScript provides the option to linearize the graph by squaring the Y-axis.
  - Linearized graph can be analyzed the same way as the first two.

Learning Outcomes:

Students will be able to:

- visualize kinematic motion and graphs.
- apply numerical and graphical analysis and differentiation to data.
- describe and discuss motion using abstract graphical and mathematical physics models.

IV. Road Map

The following describes a projected series of goals to be completed in order to complete the project in no strict order:

- Learn how to use PyDom or other similar libraries to embed Python functions into web applications:
  - Create fields for user to input values for program related parameters.
  - Create buttons for displaying tables, values, and graph elements, such as linear regressions on graph.
  - Create buttons or features to upload .csv files.
  - Buttons or features to run the Python applications.
- Create PyScript analysis tool:
  - Create Python program that analyzes provided data.
  - Create HTML layout and PyScript configuration file.
  - Modify Python program to interface with HTML page.
    - Python application includes PyDom related functions.
- Create PyScript Data Collection tool:

  - Learn how to use GDX library applications.
  - Create a Python script that collects position vs. time data with Vernier motion detector and exports as.csv file.
  - Create a Shell script to install GDX and other required libraries.
  - (Stretch goal) Create a PyScript-based simulation that calculates acceleration analytically from parameters and uses an RK4 integrator that generates position vs. time data and exports as a.csv file.
  - (Stretch goal) Incorporate data collection tool directly in PyScript w/ the data analysis tool.
- Learn how to use MatPlotLib animation libraries to animate graphs in ALL tools.

Appendix

This collaboration is expected to implement sensor data integration, numerical physics (central difference), and browser-based computing to offer a modern, accessible learning platform. By integrating Python-powered tools into web interfaces, students engage more deeply with data analysis, scientific modeling, and core physical principles.

Documentation and Examples

- PyDom (Web Elements in Python):
  - Example:
    - https://pyscript.com/@examples/todo-app/latest?files=README.md,main.py,pyscript.to
  - Documentation:
    - https://docs.pyscript.net/2024.5.1/user-guide/dom/
    - https://pydom.dev/en/stable/
    - https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model
- Vernier Go-Direct (GDX):
  - Go-Direct Python Git: https://github.com/VernierST/godirect-py
  - GDX Application Examples Git: https://github.com/VernierST/godirect-examples
- Matplotlib Animation:
  - Documentation: https://matplotlib.org/stable/api/animation_api.html