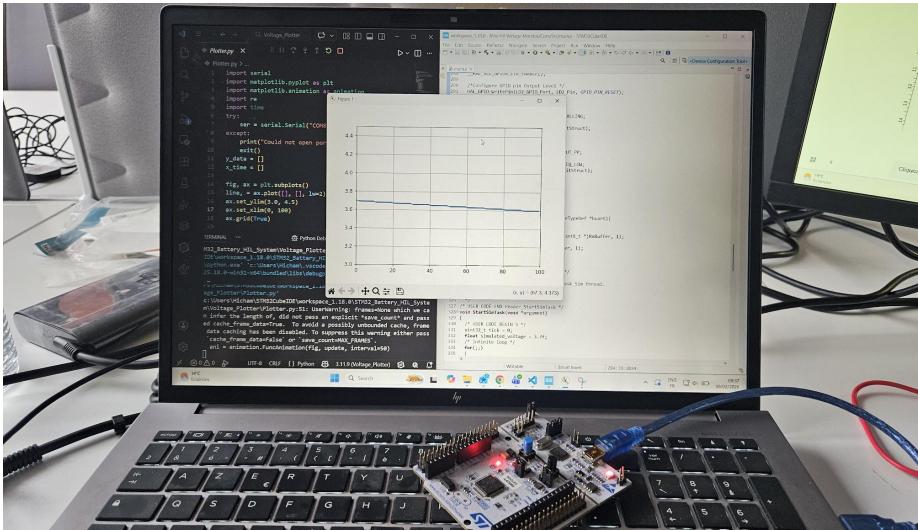


# Hardware-in-the-Loop Battery Simulator

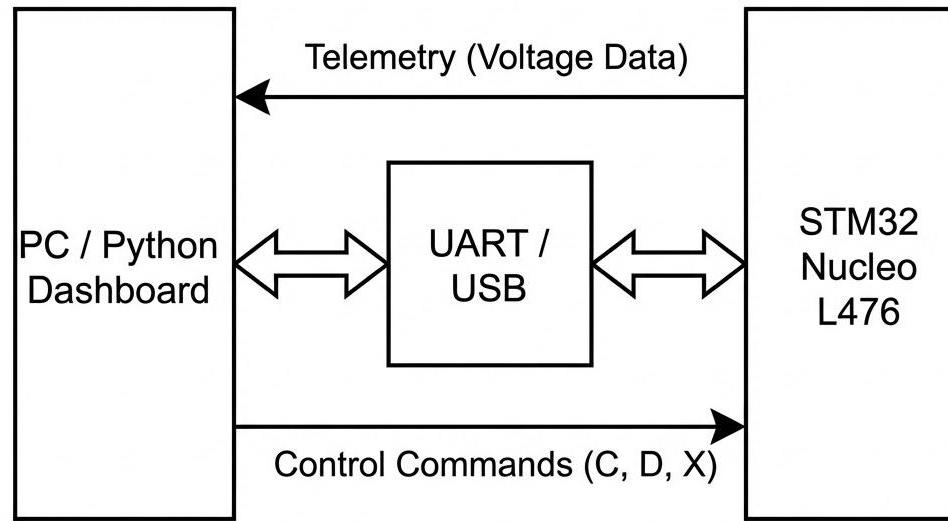
Real-time BMS Testing with STM32 & Python



# Why Simulate a Battery?

- Testing Battery Management Systems (BMS) with real Li-Ion batteries is dangerous (fire risk) and slow (charging takes hours).
- **The Solution:** A HIL Simulator that mimics battery physics in real-time.
- **Benefit:** Allows for safe "Fault Injection" (e.g., short circuits) and instant charging/discharging scenarios.

# High-Level Architecture



# Real-Time Embedded Design (FreeRTOS)

- **Task 1: Simulation (High Priority):** Calculates physics ( $V = 3.7 + \sin(t)$  or  $V_{new} = V_{old} - Load$  ).
- **Task 2: Filter (Normal Priority):** Implements a Moving Average Filter to remove sensor noise and checks safety limits.
- **Task 3: Communication (Low Priority):** Formats and transmits JSON/String data to the PC.

# Dashboard & Control Station

- **Data Visualization:** Uses `matplotlib` for 10Hz real-time plotting.
- **Parsing:** Regex-based engine to extract reliable data from serial streams.
- **Event Handling:** Asynchronous keyboard listener for user control.

# Interactive Control

when pressing 'c' the battery starts charging

The screenshot shows a development environment with two main windows:

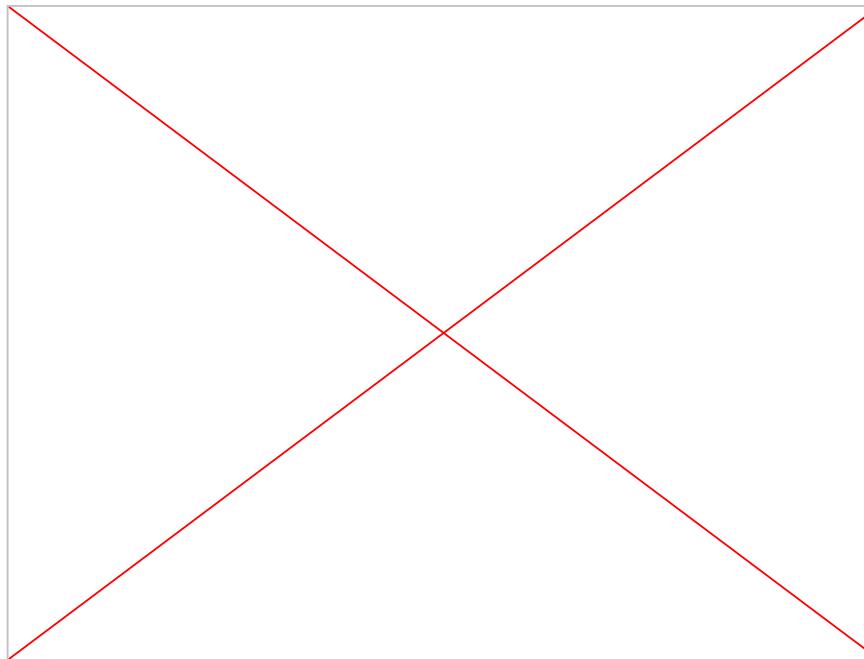
- Plotter.py:** A Python script named `Plotter.py` is open in the left editor. It uses the `serial` module to read from a COM port, `matplotlib.pyplot` to plot the data, and `matplotlib.animation` to create an animated plot. The plot shows a sinusoidal wave oscillating between approximately 3.2 and 4.2. The x-axis ranges from 0 to 100, and the y-axis ranges from 3.0 to 4.4.
- main.c:** An STM32CubeIDE project window titled `workspace_1.18.0 - Mini HIL Voltage Monitor/Core/Src/main.c - STM32CubeIDE` is open in the right editor. It contains C code for initializing GPIO pins and a loop that reads a simulated voltage value from memory. The code includes comments explaining the configuration of GPIO pins and the start of a simulation task.

The terminal window at the bottom shows the command used to run the Python script:

```
r's:\richam\STM32CubeIDE\workspace_1.18.0\STM32_Battery_HIL_System\HIL\Battery_Plotters\Plotter.py'
```

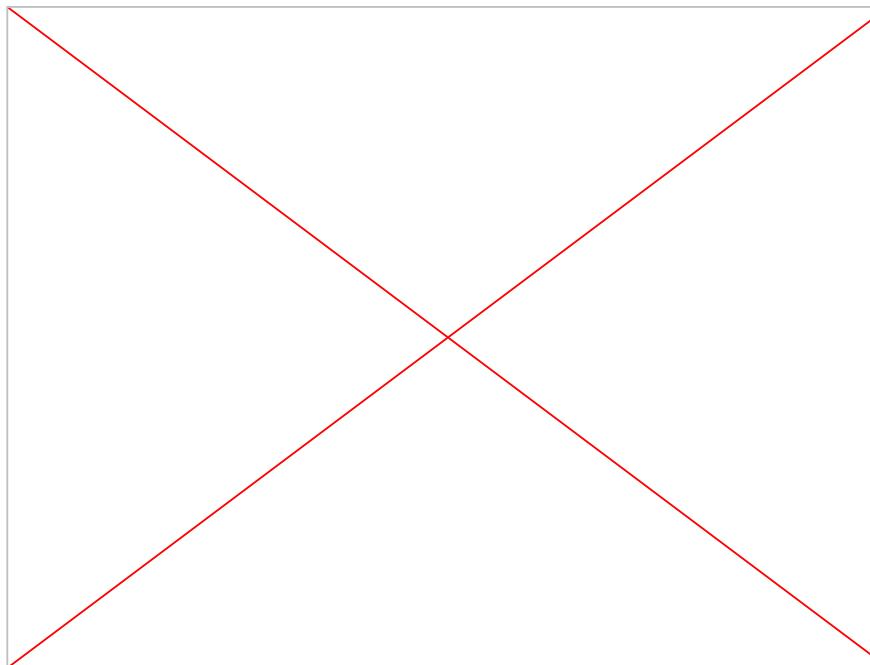
# Safety Critical Testing

when pressing ‘x’ the system simulate a “cut wire” and the system alarm LED turns on in under 10 ms.



# Realistic Discharge Curves

when pressing 'e' Simulating long-term battery drain to test the 'Low Battery' warning logic without waiting hours for a real battery to empty.



# Engineering Challenges

- **Challenge:** The STM32 L476RG clock conflicted with FreeRTOS.
- **Solution:** Migrated the HAL Timebase to TIM6 to prevent SysTick collisions.
- **Challenge:** Serial data parsing was unreliable (partial strings).
- **Solution:** Implemented Regex filtering and a circular buffer.

# Summary

- Successfully built a full-stack HIL test bench.
- Demonstrated RTOS, UART, and Python driver development.
- link:

[https://github.com/HichamBouzalmad/STM32\\_Battery\\_HIL\\_System/tree/master](https://github.com/HichamBouzalmad/STM32_Battery_HIL_System/tree/master)