




1 Load_LPP: a Rust pipeline to Log, Process, and Plot 2 load time series

3 **Luca Peruzzo**  ^{1,2}, **Chunwei Chou**  ², **Tonelato Irene** ³, and **Yuxin Wu**  ²

4 ¹ University of Padova, Department of Geosciences, Italy. ² Environmental and Earth Sciences Area,
5 Lawrence Berkeley National Laboratory, California, USA. ³ Individual contributor

DOI: [10.xxxxxx/draft](https://doi.org/10.xxxxxx/draft)

Software

- [Review](#) 
- [Repository](#) 
- [Archive](#) 

Editor: 

Submitted: 29 July 2025

Published: unpublished

License

Authors of papers retain copyright
and release the work under a
Creative Commons Attribution 4.0
International License ([CC BY 4.0](#))

6 Summary

7 Load_LPP is a data pipeline for Logging, Processing, and Plotting time series from weighing
8 systems. Load_LPP was developed to offer a complete but also flexible and extendable pipeline
9 (see implementation section). It was designed to address the lack of similar libraries and
10 ready-to-use executables in scientific hydrological studies targeting rain and evapotranspiration
11 fluxes through the monitoring of the water-load changes, e.g., using lysimeters and rhizotrons.
12 Such applications face several challenges because of their outdoor location, expected low and
13 difficult maintenance (remote and underground installation), load resolution requirements,
14 and high temporal variability of both signal (water changes associated with rain and ET) and
15 noise (wind and temperature changes in primis). Beyond hydrology, the above challenges are
16 common issues in several other fields where precise weighing monitoring is needed, as described
17 in the following section. Load_LPP is written in Rust and takes advantage of the low-level
18 optimization and general reliability of the compiled software, which become relevant in the
19 expected applications.

20 Mention

21 The project was used during the lysimeter and rhizotron studies by Peruzzo et al. (2024) and
22 Mary et al. (2023) (Figure 1). A third related work is in preparation by the same research
23 group. Considering the number of similar hydrological studies, the software publication could
24 lead to its adoption from other research groups. More in general, the number of published
25 Rust libraries has been increasing, also supported by some public policies and recommendations
26 (Matsakis & Klock, 2014; Perkel, 2020; Schueller et al., 2022). Nonetheless, the language
27 ecosystem for data management is relatively young and there are not many complete data
28 pipelines, from logging to visualization. In this sense, Load_LPP could also serve as a reference
29 for other software focusing data science and pipelines.

*first author

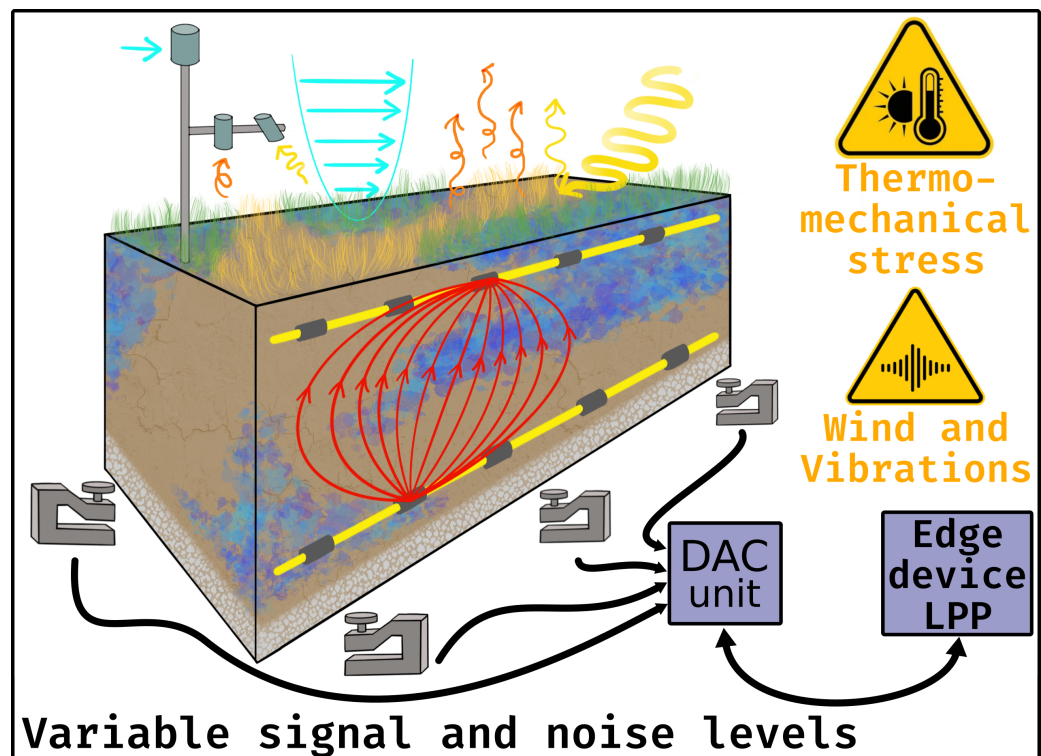


Figure 1: Example of usage in edge device, as from previous scientific studies (Peruzzo et al., 2024).

Statement of need

Weighing systems of load cells have been relevant in many fields, often serving as critical and safety components. For example, weighing systems have been used to track precise dosages in pharmaceutical manufacturing, feed storage and mixing in agriculture and farming, and larger systems are often used in construction and mining. Recent use of weighing systems has been linked to the diffusion of smart technologies, for example in smart cities, healthcare, transportation and logistics. While the variety of applications results in diverse technical challenges, these are typically associated with uneven load distributions, mechanical stresses, temperature effects, and vibrations (Peters et al., 2016; Tiboni et al., 2020). In addition, the temporal dynamics of such predisposing factors result in variable levels of both signal (i.e., load changes) and noise (e.g., vibrations), which further complicates the design of optimal weighing systems and processing procedures. Both temporal resolution and measurement time can vary significantly, with applications that often require the system to reliably function over relatively long periods, from quick weigh-in-motion measurements (Jacob & Feypell-de La Beaumelle, 2010; Lin et al., 2022) to monitoring of years. Such issues have been addressed with a combination of mechanical and processing - filtering solutions, the latter often being preferred because of cost limitations.

In this sense, commercial systems have some possible limitations:

1. Often require specific loggers or laptops (e.g., GUIs).
2. In turn, the reliance on specific loggers and laptops also limits the control over long-term monitorings (e.g., operative system schedules).
3. Provide limited processing and automation functionalities, while sometimes also limiting the flexibility in recording true-raw data at the desired temporal resolution.
4. Can have significant overhead processing steps because of device- or manufacturer-specific formats and functionalities, particularly when multiple systems are used across

- 55 different projects.
- 56 5. Are not, or hardly, extendable.
- 57 6. Are not suited for large data sets, from the simple handling and visualization of large
- 58 files to the numerical optimization.
- 59 7. From a general perspective, they are not open source and thus can conflict with some
- 60 scientific best practices, depending on their complexity level (e.g., point 3).

61 Implementation

62 Load_LPP logging functionality, combined with the rich and optimized processing tools, aims to

63 address the above limitations and introduced challenges. The logging functionality (`load_log`)

64 manages a TCP (Transmission Control Protocol) stream between the logging device and the

65 digital amplifier of the weighing system. This connection is used to receive the data and

66 control the logging settings. The logging functionality only depends on the network primitives

67 provided by the Rust standard library `std::net` and the `Chrono` library. Among the implemented

68 functionalities, `Load_LPP` automatically recovers the connection after possible timeouts and

69 other issues that are not covered by the primitives, and yet frequent in actual applications. This

70 is in-line with more autonomous and remote scientific setups, often also running on limited

71 power supplies. The `Chrono` library is used to manage the time aspects of the connection

72 and logging schedule. Note that data are stored and processed with timezone-aware `datetime`,

73 which also correctly handles clock variations over the year (standard format RFC 3339 - ISO

74 8601). The executable offers several configurations, including connection parameters to adapt

75 to different weighing systems and logging parameters, such as measurement frequency and

76 delays. The configurations are accessible and documented within the command-line help

77 functionality, which also provides convenient default values. The compiled executable requires

78 very limited CPU and memory resources, thanks to the limited dependencies and careful and

79 low-level management of memory and connection. This allows it to run on very minimal and

80 low-cost edge devices. The use of the Rust language offered a good balance of optimization

81 and reliability for such edge applications.

82 The processing functionalities cover the common filtering, gap filling, and smoothing steps,

83 as well as more specific time-series operations. As for the logging part, the processing

84 functionalities are then organized into an executable, which eases and organizes the processing

85 steps (`load_process`). The first processing step checks the continuity and order of the time

86 series, including the automatic handling of time zones and clock variations. The continuity

87 is automatically checked and, if needed, corrected relative to the minimum temporal interval

88 observed within the time series. The missing values are filled with NaN. The second processing

89 step is the removal of periods specified by the user (maintenance, etc.), these are conveniently

90 stored and read from a plain text file. In addition, the filtering phase also provides an automatic

91 detection of anomalous periods, which can be saved into a file for successive user verification

92 merging the two files. The straightforward range-based filter is also implemented, which checks

93 whether the individual values are within an accepted range. The discarded values are replaced

94 with NaN. At this point, flexible smoothing solutions are available to smooth the data and

95 replace the NaN values by interpolation (gap filling), when suitable. A weighted moving

96 average is defined with user-defined width and weight distribution (linear distribution between

97 a central-maximum value and side-minimum weights). A maximum missing weight is also

98 defined so that when too many measurements are missing within the window - accounting by

99 their relative weight - the central value being interpolated is left to NaN. This would warn

100 the user about large data gaps, which are not addressed by simple gap-filling. Because of the

101 expected long time series and possibly large windows (containing many data), these operations

102 are parallelized, using both multi-threading and single-instruction multiple-data optimizations.

103 Finally, an adaptive-window solution is also implemented to address the temporal variability of

104 noise and load dynamics (AWAT). This calculates the relative level of signal and noise based on

105 a polynomial regression of the windowed data (Hannes et al., 2015; Peters et al., 2014). The

106 optimal window width is then calculated based on the Akaike's information criterion (Akaike,

107 1974): strong signal (i.e., significant and fast load changes) favors shorter windows to avoid
 108 excessive smoothing; however, this is balanced by the relative relevance of the noise, which
 109 favors longer windows.

110 A third and last executable is the plotting part, which allows a quick visualization of the
 111 processed data, reading either the raw or processed data. The Rust plotly bindings are used
 112 to handle the smooth and interactive visualization of the expected large time series, up to
 113 some millions of measurements.

114 Several tests are including in the library to verify the correctness of the functionalities. The
 115 testing part uses the cargo test default command.

116 Acknowledgements

117 We acknowledge the support of the Advanced Research Projects Agency - Energy, Rhizosphere
 118 Observations Optimizing Terrestrial Sequestration (ARPA-E ROOTS). We also acknowledge
 119 the Watershed Function Scientific Focus Area funded by the U.S. Department of Energy,
 120 Office of Science, Office of Biological and Environmental Research under award number
 121 DE-AC02-05CH11231.

122 Akaike, H. (1974). A new look at the statistical model identification. *IEEE Transactions on*
 123 *Automatic Control*, 19(6), 716–723. <https://doi.org/10.1109/TAC.1974.1100705>

124 Hannes, M., Wollschläger, U., Schrader, F., Durner, W., Gebler, S., Pütz, T., Fank, J., Von
 125 Unold, G., & Vogel, H.-J. (2015). A comprehensive filtering scheme for high-resolution
 126 estimation of the water balance components from high-precision lysimeters. *Hydrology and*
 127 *Earth System Sciences*, 19(8), 3405–3418. <https://doi.org/10.5194/hess-19-3405-2015>

128 Jacob, B., & Feypell-de La Beaumelle, V. (2010). Improving truck safety: Potential of
 129 weigh-in-motion technology. *IATSS Research*, 34(1), 9–15. <https://doi.org/10.1016/j.iatssr.2010.06.003>

131 Lin, M.-H., Sarwar, M. A., Daraghmi, Y.-A., & Ik, T.-U. (2022). On-Shelf Load Cell Calibration
 132 for Positioning and Weighing Assisted by Activity Detection: Smart Store Scenario. *IEEE*
 133 *Sensors Journal*, 22(4), 3455–3463. <https://doi.org/10.1109/jsen.2022.3140356>

134 Mary, B., Iván, V., Meggio, F., Peruzzo, L., Blanchy, G., Chou, C., Ruperti, B., Wu, Y., &
 135 Cassiani, G. (2023). Imaging of the electrical activity in the root zone under limited-water-
 136 availability stress: A laboratory study for *Vitis Vinifera*. *Biogeosciences*, 20(22), 4625–4650.
 137 <https://doi.org/10.5194/bg-20-4625-2023>

138 Matsakis, N. D., & Klock, F. S. (2014). The rust language. *Proceedings of the 2014 ACM*
 139 *SIGAda Annual Conference on High Integrity Language Technology*, 103–104. <https://doi.org/10.1145/2663171.2663188>

141 Perkel, J. M. (2020). Why scientists are turning to Rust. *Nature*, 588(7836), 185–186.
 142 <https://doi.org/10.1038/d41586-020-03382-2>

143 Peruzzo, L., Chou, C., Hubbard, S. S., Brodie, E., Uhlemann, S., Dafflon, B., Wielandt, S.,
 144 Mary, B., Cassiani, G., Morales, A., & Wu, Y. (2024). Outdoor mesoscale fabricated
 145 ecosystems: Rationale, design, and application to evapotranspiration. *Science of The Total*
 146 *Environment*, 957, 177565. <https://doi.org/10.1016/j.scitotenv.2024.177565>

147 Peters, A., Nehls, T., Schonsky, H., & Wessolek, G. (2014). Separating precipitation and
 148 evapotranspiration from noise – a new filter routine for high-resolution lysimeter data.
 149 *Hydrology and Earth System Sciences*, 18(3), 1189–1198. <https://doi.org/10.5194/hess-18-1189-2014>

151 Peters, A., Nehls, T., & Wessolek, G. (2016). Technical note: Improving the AWAT filter
 152 with interpolation schemes for advanced processing of high resolution data. *Hydrology and*

- 153 *Earth System Sciences*, 20(6), 2309–2315. <https://doi.org/10.5194/hess-20-2309-2016>
- 154 Schueller, W., Wachs, J., Servedio, V. D. P., Thurner, S., & Loreto, V. (2022). Evolving
155 collaboration, dependencies, and use in the Rust Open Source Software ecosystem. *Scientific*
156 *Data*, 9(1), 703. <https://doi.org/10.1038/s41597-022-01819-z>
- 157 Tiboni, M., Bussola, R., Aggogeri, F., & Amici, C. (2020). Experimental and Model-Based
158 Study of the Vibrations in the Load Cell Response of Automatic Weight Fillers. *Electronics*,
159 9(6), 995. <https://doi.org/10.3390/electronics9060995>

DRAFT