The Journal of Open Source Software

Plaquette: An Object-Oriented Framework for Embedded Signal Processing in Interactive Media

Sofian Audry [1*] **and Thomas Ouellet Fredericks** [2*]

1 Université du Québec à Montréal, Canada **2** Collège Montmorency, Canada * These authors contributed equally.

DOI: 10.xxxxxx/draft

Software

- Review
- Repository
- Archive

Editor: Daniel S. Katz

Reviewers:

- @WarmCyan
- @timonmerk

Submitted: 30 September 2025
Published: unpublished

Summary

Plaquette is an object-oriented C++ framework for interactive media on embedded systems, supporting a wide range of platforms including AVR, SAMD, STM32, and ESP32. It provides a signal-centric architecture and a suite of modular abstractions (oscillators, filters, units, and scheduling engines) that simplify the design of time-based behaviors. Its expressive syntax allows fast prototyping with multiple sensors, actuators, and real-time processes, enabling researchers and creative practitioners to experiment and design with complex physical computing systems.

Beyond its technical contributions, *Plaquette* serves as a bridge between scientific research and creative practice. Its applications to interdisciplinary projects involving affective biofeedback and robotic behaviors demonstrate the framework's flexible and robust infrastructure in support of creativity and experimentation in interactive media.

Statement of Need

Plaquette is designed for research—particularly practice-based research (Candy & Edmonds, 2018) and research-creation (Loveless, 2019)—in embedded interactive media, with applications ranging from affective computing and digital lutherie to robotic art, interactive installation, connected objects, and performance. The open-source platform for physical computing Arduino (Banzi & Shiloh, 2022) anchors a large and active ecosystem, and remains the go-to environment for artists, makers, and researchers working with embedded interactive media. Its accessibility, community, libraries, and hardware options have made it the most popular microcontroller platform worldwide. Yet, Arduino's core library is not optimized for real-time signal processing: it lacks an object-oriented design, has few abstractions for managing concurrent events, and often requires manipulating raw numerical values, making interaction design unintuitive and hindering expressive experimentation.

In contrast, dataflow software popular within scientific and creative communities working with interactive media such as Pure Data, Max, and TouchDesigner provide powerful models for composing with signals, but are too memory and CPU intensive to run on constrained hardware. Similarly, scientific tools such as Python's NumPy/SciPy, Matlab, or R, while offering rich signal analysis tools, are not designed for real-time processing on embedded devices.

Plaquette addresses these gaps by bringing the expressive power of dataflow signal-based programming into a lightweight object-oriented framework optimized for microcontrollers (i.e., requiring minimal CPU and memory usage). It enables intuitive handling of signals, providing efficient implementations of core data processing functions such as peak detection, normalization, scaling, and smoothing (low-pass). This enables researchers in art and science to focus on experimentation and expressivity while ensuring accurate and reliable real-time performance on resource-constrained platforms.

The framework provides a strong foundation for workshop-based research-creation projects, where participants often have diverse levels of technical skills. Its accessibility ensures that beginners can quickly grasp and apply core concepts, while its efficient, expressive, and extensible architecture support the needs of advanced users. This makes it particularly well-suited for collaborative prototyping in media arts, design, and human-computer interaction, where embodied and situated practices require adaptable tools.

Plaquette has already supported a number of public research projects. It was used to improve real-time physiological signal processing as part of the BioData library for affective biofeedback, supporting creative applications in music and performance (Gee, 2023), and used in studies of electrodermal activity (Hagler et al., 2022). It was integrated at the core of the MisBKit, a robotic kit enabling research on object behaviors (Bianchini et al., 2015). It was also employed for signal processing and robotic expression in *Morphosis*, an installation featuring three spheroid robots that learn in real-time using reinforcement learning Audry (2023). These examples illustrate the framework's role not only as a technical tool but also as a catalyst for interdisciplinary research.

Functionality and Design Overview

The core of *Plaquette* is organized around two complementary abstractions called *units* and *engines* that provide a coherent structure for building complex, real-time interactive systems on microcontrollers. *Units* are modular building blocks that encapsulate behaviors such as sensing, generating, filtering, or actuating. *Engines* operate as conductors, managing initialization and timing of units so that they execute consistently without blocking or interruptions.

All units implement a unified interface consisting of a single input and a single output function. This design makes it possible to chain units together in a dataflow-like manner using a piping operator (>>), where the output of one unit is sent as input to another. This signal-centric approach allows developers to work with flows of information rather than low-level procedural code.

The framework includes a set of core unit types:

- **Base units**: basic analog and binary input/output.
- **Generators**: generative source signals such as square, triangle, and sine waves, as well as ramps.
- **Timing units**: scheduling and temporal control such as timers and metronomes.
- **Filters**: real-time signal transformations such as min-max scaling, normalizing, and detecting peaks.
- **Fields**: spatial functions sampled at fractional positions to plot, shape, or transform signals across space.

Engines and units have a low memory footprint, with static allocation at compile time to avoid dynamic allocation and fragmentation. In particular, signal-processing units such as min-max scaling and normalizing use exponential moving averages rather than circular buffers, ensuring low and predictable memory usage.

Examples

The following program chains an analog input through a min-max scaling filter to bring it to full range, then uses the value to influence the period ofoscillation of an LED using a sine wave.

```
#include <Plaquette.h>

AnalogIn input{A0};        // analog input on pin A0
```

```
AnalogOut led{9};          // PWM-controlled LED on pin 9
MinMaxScaler scaler{};     // min-max scaler
Wave oscillator{SINE};     // sine wave

void begin() {}

void step() {
  input >> scaler;         // rescale to full range [0, 1]
  oscillator.period(scaler.mapTo(1, 10)); // set period from 1 to 10 seconds
  oscillator >> led;       // send oscillator value to LED
}
```

This program reacts to peaks in the incoming signal by triggering a sudden movement (ramp) in a servo motor. The peak detector reacts to outliers after signal normalization. The normalization is calibrated over a sliding time window, adapting to slow changes in the input.

```
#include <Plaquette.h>

AnalogIn input{A0};          // analog input on pin A0
ServoOut servo{9};           // servomotor connected on pin 9
Normalizer normalizer{0, 1}; // re-normalizes to N(0, 1)
PeakDetector peak{1.5};      // detects outliers at 1.5 times stddev
Ramp ramp{2.0};              // ramp with duration of 2 seconds

void begin() {
  normalizer.timeWindow(60); // calibration sliding time window (60 seconds)
}

void step() {
  input >> normalizer >> peak; // chain-process input signal
  if (peak) ramp.start();      // on peak detection: restart ramp
  ramp >> servo;               // send ramp value to servo motor
}
```

Acknowledgements

This work was partially supported by the Natural Sciences and Engineering Research Council of Canada, the Social Sciences and Humanities Research Council of Canada, the Fonds de Recherche du Québec — Société et Culture, and the Canada Council for the Arts. We thank our collaborators and colleagues for supporting the project, in particular Erin Gee, Luana Belinsky, and Chris Salter.

References

Audry, S. (2023). Choreomata. In R. A. Trillo & M. Poliks, *Choreomata* (1st ed., pp. 283–307). Chapman and Hall/CRC. https://doi.org/10.1201/9781003312338-16

Audry, S., Dumont-Gagné, R., & Scurto, H. (2020, December). Behaviour Aesthetics of Reinforcement Learning in a Robotic Art Installation. *4th NeurIPS Workshop on Machine Learning for Creativity and Design*. https://hal.archives-ouvertes.fr/hal-03100907

Banzi, M., & Shiloh, M. (2022). *Getting Started With Arduino: The Open Source Electronics Prototyping Platform*. Make Community, LLC. ISBN: 978-1-68045-693-6

Bianchini, S., Bourganel, R., Quinz, E., Levillain, F., & Zibetti, E. (2015). (Mis)behavioral Objects. In D. Bihanic (Ed.), *Empowering Users through Design: Interdisciplinary Studies*

and Combined Approaches for Technological Products and Services (pp. 129–152). Springer International Publishing. https://doi.org/10.1007/978-3-319-13018-7_8

Candy, L., & Edmonds, E. (2018). Practice-Based Research in the Creative Arts: Foundations and Futures from the Front Line. *Leonardo*, *51*(1, 1), 63–69. https://doi.org/10.1162/LEON_a_01471

Gee, E. M. (2023). *The BioSynth—an affective biofeedback device grounded in feminist thought*. 479–485. https://doi.org/10.5281/zenodo.11189254

Hagler, J., Kim, C., Kateb, P., Yeu, J., Gagnon-Lafrenais, N., Gee, E., Audry, S., & Cicoira, F. (2022). Flexible and stretchable printed conducting polymer devices for electrodermal activity measurements. *Flexible and Printed Electronics*, *7*(1), 014008. https://doi.org/10.1088/2058-8585/ac4d0f

Loveless, N. (2019). *How to make art at the end of the world: A manifesto for research-creation*. Duke University Press. https://doi.org/10.1215/9781478004646