RTHybrid: A Standardized and Open-Source Real-Time Software Model Library for Experimental Neuroscience

Rodrigo Amaducci*, Manuel Reyes-Sanchez, Irene Elices, Francisco B. Rodriguez and Pablo Varona*

Grupo de Neurocomputación Biológica, Departamento de Ingeniería Informática, Escuela Politécnica Superior, Universidad Autónoma de Madrid, Madrid, Spain

Closed-loop technologies provide novel ways of online observation, control and bidirectional interaction with the nervous system, which help to study complex non-linear and partially observable neural dynamics. These protocols are often difficult to implement due to the temporal precision required when interacting with biological components, which in many cases can only be achieved using real-time technology. In this paper we introduce RTHybrid (www.github.com/GNB-UAM/RTHybrid), a free and open-source software that includes a neuron and synapse model library to build hybrid circuits with living neurons in a wide variety of experimental contexts. In an effort to encourage the standardization of real-time software technology in neuroscience research, we compared different open-source real-time operating system patches, RTAI, Xenomai 3 and Preempt-RT, according to their performance and usability. RTHybrid has been developed to run over Linux operating systems supporting both Xenomai 3 and Preempt-RT real-time patches, and thus allowing an easy implementation in any laboratory. We report a set of validation tests and latency benchmarks for the construction of hybrid circuits using this library. With this work we want to promote the dissemination of standardized, user-friendly and open-source software tools developed for open- and closed-loop experimental neuroscience.

Keywords: closed-loop technologies, hybrid circuits, hard real-time, dynamic-clamp, activity-dependent stimulation

1. INTRODUCTION

The study of neural systems dynamics is hindered by various factors. The first one is their intrinsic non-linearity, since they process information in several interacting spatial and temporal scales and are affected by multiple transient adaptation and learning mechanisms. Also, from all magnitudes involved in these dynamics, just a few can be accessed simultaneously, making the system partially observable. The third factor is related to the use of the traditional stimulus-response paradigm in most experimental neuroscience research, which only allows to record the behavior of the system under different stimuli and then to analyze the collected data offline. Thus, highly complex non-stationary neural activity, which has influence from the context and previous events' feedback, can not be completely assessed. Closed-loop techniques provide an efficient way to overcome such difficulties by interacting online with the system, producing precise stimulus according to

the recorded information and presenting valuable insights on transient neural processes. This paradigm allows more flexibility in the experiment and favors its automation, as well as the control of neural dynamics (Chamorro et al., 2012; Potter et al., 2014; Roth et al., 2014; Varona et al., 2016).

While closed-loop technology allows researchers to conduct online observation, control and goal-driven interaction with neural elements, it also presents some drawbacks in its implementation. Some of these difficulties are related to the complexity of the associated experimental design, but also to the accomplishment of precise temporal restrictions, in the scale of milliseconds or lower, which are often required during data acquisition and stimulation in biological experiments (Christini et al., 1999; Muñiz et al., 2005, 2008, 2009). The capacity of a system to perform periodic tasks and respond to asynchronous external events in an strict time slot (neither sooner nor later) is known as real-time (Furht et al., 1991). Therefore, to ensure the compliance of the previously mentioned temporal margins in an experimental setup, real-time technology is needed. It is also important to mention that the term "real-time" may as well be used in neuroscience literature referred to online recording, feedback or control (Siegle et al., 2017). In this paper we always refer to strict temporal precision in the millisecond range.

Electronic components easily fulfill the speed and precision requirements needed for most real-time scenarios, so hardware-based implementations are one possible solution (Robinson and Kawai, 1993; Le Masson et al., 1995; Broccard et al., 2017). Current development of inexpensive microcontrollers with large computational power allows efficient implementations (Desai et al., 2017). On the other hand, software-based solutions are usually more programmable and manageable, offering maximum flexibility and a wide range of user-friendly frameworks. Moreover, nowadays personal computers have enough hardware capacity to comply with standard real-time restrictions and benefit from large memory for complex closed-loop protocol implementations. However, modern general purpose operating systems (GPOS), such as Windows, MacOS or GNU/Linux, are multitask environments with internal schedulers, which assign computer resources to different running tasks following specific policies (Stallings, 2012). These schedulers can not be controlled by the users, hence it can not be ensured that a given task will run without interruptions and, therefore, real-time can not be assured. In order to run protocols accomplishing a set of established temporal boundaries with software-based real-time, another system framework, known as real-time operating system (RTOS), is needed.

Numerous real-time tools for experimental neuroscience are already available. Some of them are hardware-based (Franke et al., 2012; Tessadori et al., 2012; Müller et al., 2013; Desai et al., 2017). Several software tools have been designed, particularly for dynamic-clamp electrophysiology experiments, both following soft- (Pinto et al., 2001; Nowotny et al., 2006; Linaro et al., 2014; Ciliberti and Kloosterman, 2017; Hazan and Ziv, 2017) and hard real-time (Christini et al., 1999; Dorval et al., 2001; Muñiz et al., 2005, 2009; Biró and Giugliano, 2015; Patel et al., 2017) approaches, using distinct platforms and RTOS, which have

diverse purposes and architectures, hence presenting different advantages and disadvantages.

A highly relevant example of closed-loop interactions can be found in hybrid circuits, which are networks built by connecting model neurons and synapses to living cells. They are a powerful tool to explore and characterize neural system dynamics, as well as a means to assess the role of specific circuit components (e.g., see Yarom, 1991; Pinto et al., 2000; Szücs et al., 2000; Varona et al., 2001; Le Masson et al., 2002; Nowotny et al., 2003; Oprisan et al., 2004; Arsiero et al., 2007; Chamorro et al., 2009; Grashow et al., 2010; Brochini et al., 2011; Kispersky et al., 2011; Wang et al., 2012; Thounaojam et al., 2014; Hooper et al., 2015; Norman et al., 2016; Broccard et al., 2017). The most common paradigm to build hybrid circuits consists in using dynamic-clamp to read the membrane potential of a cell and, after computing a model using this voltage, inject the resulting current into the same or into a different cell (Robinson and Kawai, 1993; Sharp et al., 1993; Prinz et al., 2004; Destexhe and Bal, 2009; Nowotny and Varona, 2014). Neuron models range from simple mathematical approximations to more complex conductance-based implementations, which reproduce biophysical behaviors with high accuracy. The same thing occurs for synapse models, which can cover from simple gap junctions using Ohm's law to chemical connections described by non-linear differential equations (Torres and Varona, 2012).

Beyond dynamical-clamp protocols, other stimulation and detection techniques that can improve their performance by using real-time technology are fMRI (Rana et al., 2016), optogenetics (Krook-Magnuson et al., 2013; Prsa et al., 2017), EEG setups (Arrouët et al., 2005; Sitaram et al., 2016), neuroprostheses (Levi et al., 2018), or any kind of activity-dependent stimulation experiment, such as the ones that use simultaneous electrophysiological and video tracking (Muñiz et al., 2011), acute mechanical stimulation (Muñiz et al., 2008), electric signaling during behavior (Forlim et al., 2015; Lareo et al., 2016) or drug microinjection (Chamorro et al., 2009, 2012).

Due to the lack of flexibility in hardware real-time solutions, as well as RTOS heterogeneity and intrinsic difficulties in their use, many neuroscience researchers overlook this technology when designing closed-loop experiments. In this paper we introduce RTHybrid, a novel, multiplatform, real-time software neuron and synapse model library to build hybrid circuits. With this tool we aim to promote the use of standardized and user-friendly real-time software technology, available in different platforms, to favor the implementation of closed-loop experimentation in neuroscience research. We provide validation examples of this tool in the context of hybrid circuit implementations using dynamic-clamp, including detailed analysis and benchmarking of temporal precision in different RTOS.

2. MATERIALS AND METHODS

2.1. Real-Time Software

Real-time performance is often wrongly considered as a matter of speed, which is of course important, when it actually relies on temporal precision: actions must be delivered within a pre-established interval, neither sooner nor later. Many neurons follow precisely a similar behavioral pattern: slow activity (less

than 1 kHz) but precise subcellular sequential dynamics or spiking coding.

When a computer performs a given task, there is always some latency between the moment when this task is expected to be accomplished and when it is actually done, as well as some jitter of these latency values, due to the performance of the operating system scheduler. Moreover, not all tasks are equally sensitive to high latency values or data loss, hence real-time software can be classified in two types: soft real-time, when some deadlines can be missed without performance degradation as long as some threshold is not exceeded (for example, an online music streaming service can lose some data packages and users will not notice it), and hard real-time, when all deadlines must be met or the system fails critically (computers controlling a nuclear plant or a satellite, for example) (Shin and Ramanathan, 1994). In this paper, the term real-time always refers to hard real-time.

Despite how differently a RTOS can be designed, the functioning of all of them rely on two elements: their scheduling algorithms, which are usually preemptive, meaning that they are able to interrupt a running process without its permission; and how they manage hardware interruptions (Abbott, 2006). Some of them are implemented from scratch, while others are based in existing GPOS' kernels (Hambarde et al., 2014). Among the later, dual-kernel (a real-time microkernel is used along the standard one) and single-kernel solutions (the standard kernel is modified to support real-time) are common (Yaghmour, 2003; Dietrich and Walker, 2005).

There exist several commercial RTOS implementations, mainly designed for embedded systems, such as QNX Neutrino, VxWorks and Windows Compact Embedded, among others. Other proprietary real-time software solutions are MATLAB's toolbox xPC and National Instruments' LabVIEW Real-Time, which can achieve hard real-time precision if external hardware is used for the computation but otherwise are soft real-time, and Simulink Desktop Real-Time, which provides a hard real-time kernel (up to 1kHz sample rate) for executing Simulink models running Windows or Mac OS X. The main drawback of these tools is their high economic cost, which may be unaffordable for many research groups and laboratories. There are also open-source and free solutions to obtain an RTOS from Linux, an open-source GPOS, which avoid the stated inconveniences of commercial options and provide similar usability, or even better, in terms of performance (Aroca and Caurin, 2009; Hambarde et al., 2014). In this paper, three of these patches were studied, and their performance and characteristics were compared in relation to the task of building hybrid circuits with living and model neurons: RTAI, Xenomai and Preempt-RT.

RTAI (Real-Time Application Interface) (Mantegazza et al., 2000) was first developed in 1996, becoming one of the first and most widely used open-source real-time environments. It is based in a dual-kernel approach, using a real-time microkernel with a preemptive scheduler, which controls the interrupt requests (IRQ) and treats the standard Linux kernel as a low priority task. Another dual-kernel solution is Xenomai (Gerum, 2004), a project which was part of the RTAI project until 2005, when it become a fully independent tool. Other RTOS follow a single kernel approach, meaning that no auxiliary microkernel is used

and that the standard one is modified instead to work in real-time. All changes made to Linux standard kernel for this purpose throughout the past decade are included in Preempt-RT (Dietrich and Walker, 2005).

2.2. Experimental Setup
2.2.1. Benchmarking Tests
Performance on the three RTOS described in the previous section, RTAI, Xenomai and Preempt-RT, was measured and compared among them and also to a GPOS with no real-time capabilities. Specifications of the computers used for these tests can be found in **Table 1**. Most modern computers currently have multicore processors, i.e., one component with several independent processing units. Linux also allows to isolate specific cores, so the scheduler will not assign them any task, and to manually bound a specific task to this empty core. We have also analyzed how this core isolation affected the performance in both real-time and non real-time implementations.

The benchmarking procedure consisted in a latency test, measuring the time difference between when an action was expected and when it really happened (see **Figure 1**). The benchmarking program[1] consisted in a periodic loop with a frequency of 20 kHz that sent a digital 0 or 1 to a digital acquisition (DAQ) device alternately on each iteration, thus producing a 100 μs square-wave signal. After sending the corresponding value the program slept until the next interval arrived: the time interval between the real and the expected awaking time was the measured latency. An Agilent MSO7104A oscilloscope was used as an external temporal reference and *stress*[2], a workload generator software, was utilized to create a worst case scenario, with all processor cores and the file Input/Output system running at full capacity. To send the signal to the oscilloscope a National Instruments PCI-6251 board and a BNC-2090A DAQ device were used.

2.2.2. RTHybrid Validation Tests
In order to validate the proper functioning of RTHybrid, it has been tested in a real experimental environment. Hybrid circuits were built by bidirectionally connecting a neuron model simulated by the software and a living neuron, using chemical graded model synapses. For these tests, a computer with a 4-core Intel Core i7-6700 3.40 GHz processor and 16 GB RAM memory, as well as a National Instruments PCI-6251 board with a NI BNC-2090A DAQ device, were used. Measures of the time taken by each task in the interaction cycle were also conducted to evaluate their contribution to minimum and maximum computational cost for each of the real-time platforms analyzed.

The membrane potential of the biological component of the experiment was recorded using *in vitro* electrophysiology in neurons of the pyloric central pattern generator (CPG) of adult *Carcinus maenas*, bought in a local fish store and kept in artificial sea water. Before the dissection, the crab was anesthetized by introducing it in the freezer for 20/30 min.
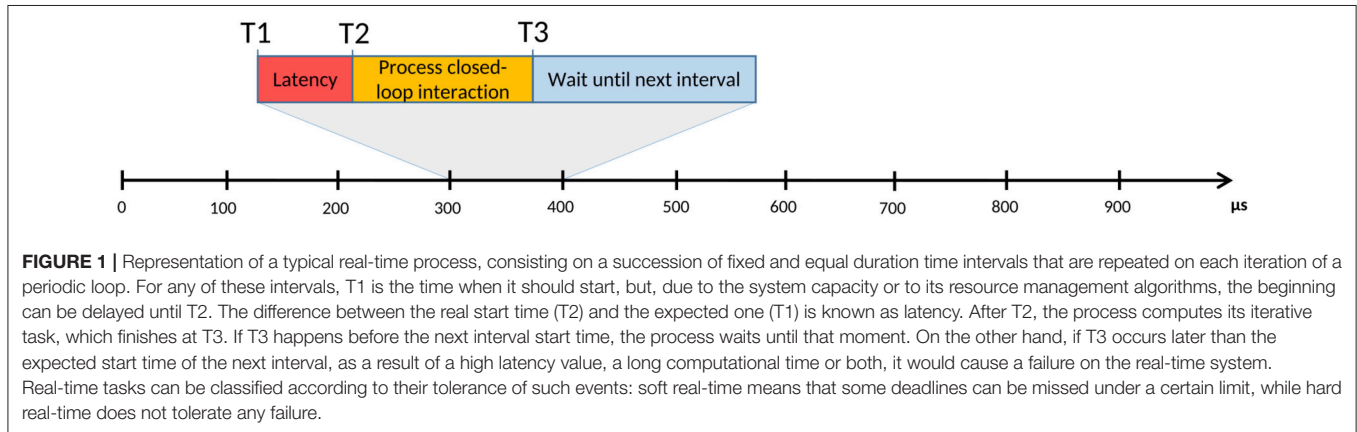
---

[1] Source code of the latency test: www.github.com/RoyVII/Latency_tests.git
[2] stress software website: www.people.seas.harvard.edu/~apw/stress/

TABLE 1 | Software and hardware specifications of the computers used on the benchmarking tests for each operating system and RTOS used in this study.

	Operating system	Kernel version	RAM	Processor	Cores
Non real-time	Debian 9	4.9.0-4	16 GB	Intel Core i7-4790 3.6 GHz	4
Preempt-RT	Debian 9	4.9.0-4	16 GB	Intel Core i7-4790 3.6 GHz	4
Xenomai 3.0.5	Ubuntu 16.04	4.9.90	16 GB	Intel Core i7-4790 3.6 GHz	4
RTAI 3.4	Ubuntu 10.04	2.6.34.5	4 GB	Intel Core i7-2600 3.4 GHz	4

Note that the computer used for non-real-time, Preempt-RT and Xenomai 3 tests was the same (we will call it Computer 1 hereafter) and the one for RTAI was different (Computer 2).

FIGURE 1 | Representation of a typical real-time process, consisting on a succession of fixed and equal duration time intervals that are repeated on each iteration of a periodic loop. For any of these intervals, T1 is the time when it should start, but, due to the system capacity or to its resource management algorithms, the beginning can be delayed until T2. The difference between the real start time (T2) and the expected one (T1) is known as latency. After T2, the process computes its iterative task, which finishes at T3. If T3 happens before the next interval start time, the process waits until that moment. On the other hand, if T3 occurs later than the expected start time of the next interval, as a result of a high latency value, a long computational time or both, it would cause a failure on the real-time system. Real-time tasks can be classified according to their tolerance of such events: soft real-time means that some deadlines can be missed under a certain limit, while hard real-time does not tolerate any failure.

The stomatogastric ganglion, dissected following the standard procedure, was attached to a Petri with Sylgard cold saline dissolution (13-15°C kept by a microcontroller and always perfused) using pins. The saline had the following composition (in mM): 433 *NaCl*, 12 *KCl*, 12 *CaCl$_2$* · 2 *H$_2$O*, 20 *MgCl$_2$* · 6 *H$_2$O*, 10 *HEPES*, adjusted to pH 7.60 with 4M *NaOH*. Neurons were identified after desheathing the ganglion by their membrane potential waveforms and their corresponding spike times in nerves. Intracellular recordings were performed using 3 M KCl filled microelectrodes (50 *MΩ*) and a DC amplifier (ELC-03M, NPI Electronic, Hauptstrasse, Tamm, Germany). For details on the preparation see (Elices et al., 2018).

3. RESULTS

3.1. Real-Time Benchmarking and Comparison

3.1.1. Computers Internal Clock Validation

Tests were conducted to certify that the internal clocks of the systems specified in **Table 1** were capable of working with the required microsecond precision. These consisted in three 10 second tests on each platform, generating a square-wave signal of period 100μs, which was recorded with an external oscilloscope. When compared to the signal period measured by the computers, the oscilloscope registered a $\pm10\mu s$ inaccuracy in the precision of the computers' clocks (see **Table 2**). This is an acceptable margin for our purposes with sampling rates for the hybrid circuit electrophysiology ranging from 10 to 20 kHz.

3.1.2. RTOS Usability Comparison

RTAI, Xenomai and Preempt-RT were studied and compared in terms of installation requirements, usability and user-friendliness. This analysis is summarized in **Table 3**.

3.1.2.1. RTAI

Installation requires to patch a vanilla Linux kernel with the RTAI patch and then compile it, which is a long and tricky operation, even for experimented users. The last version of the official installation guide dates from 2008 (Monteiro, 2008). Utilization of the real-time functions provided by the platform is done through its own API, which is very powerful and complete, but documentation and examples are scarce. The safest way to achieve real-time is implementing the programs as kernel space modules, which carries many impediments. User space real-time can be reached using the LXRT library, although its use is discouraged for non-senior RTAI programmers (Racciu and Mantegazza, 2006). Currently, this tool is still maintained, but not regularly: version 4 last maintenance was in 2013 and in May 2017 version 5 was released, followed by the 5.1 patch in February 2018.

3.1.2.2. Xenomai

Patching and compiling a Linux vanilla kernel is the only way of installing Xenomai and its developers provide an up-to-date guide. Since Xenomai's main purpose is to offer an open-source alternative to proprietary RTOS, it includes different APIs, intended to emulate other environments and libraries, such as VxWorks, pSOS+ and even POSIX. All of them, as well as their own API, are accessible from user space. Complete and updated documentation is available for all APIs. Although

TABLE 2 | Results of the internal clock precision tests for the computers described in **Table 1**, in microseconds.

	Computer 1			Computer 2		
μs	Min	Max	Mean ± Std	Min	Max	Mean ± Std
Measured by computer	99.5	100.5	100.0 ± 0.1	99.9	100.1	100 ± 0
Measured by oscilloscope	90.0	110.0	100 ± 3	90.0	110.0	100 ± 3

A 100μs signal, generated by the computers, was recorded with each computer and an external oscilloscope. Minimum, maximum, mean and standard deviation values of the recorded signal period are displayed at the table. When compared to the signal period measured by the computers, the oscilloscope registered a ±10μs inaccuracy in the precision of the computers clocks.

TABLE 3 | Usability and accessibility characteristics for each real-time solution explored in this study.

	RTAI	Xenomai	Preempt-RT
Installation	Kernel patching	Kernel patching	- Kernel patching - Pengutronix Debian repositories
Programming	API for C language	Various APIs emulators, including POSIX	Standard POSIX code (works also without real-time)
Documentation	Scarce and old	- Up-to-date - Few examples but active mailing-list	Plenty: standard POSIX documentation
Support and maintenance	Discontinued	Currently active	Currently active (Linux Foundation project)

the user community is not very large and there are not many examples to be found, there is a mailing-list where questions can be asked. The project is currently active (Xenomai 3 was released in October 2015) and it is maintained and updated frequently. An Ubuntu 16.04 distribution already patched with Xenomai 3.0.5 can be downloaded from our website[3].

3.1.2.3. Preempt-RT

Similarly to the dual-kernel implementations, the typical way to install Preempt-RT is by patching and compiling a vanilla kernel, following the instructions that can be found at the official website, these being significantly simpler than the ones for RTAI and Xenomai. An alternative possibility for Debian distributions is to install it from its repositories as any other package. Despite the real-time patch, the system is still a normal Linux, so the standard POSIX library can be used and all its documentation is valid. In 2015 the project was transferred to The Linux Foundation, becoming the "official" Linux real-time solution.

3.1.3. RTOS Benchmark Analysis

RTAI, Xenomai and Preempt-RT were tested using the method described in 2.2.1. Each trial consisted in a 5 min run of the test program under stress, with a frequency of 20kHz, and was repeated twice on every platform. We measured the maximum, minimum and mean latency values, as well as the jitter, and the results can be seen in **Figure 2**. Distribution of latency values during these tests is shown in **Figure 3**.

RTAI obtained the best performance scores, getting $3.65\mu s$ as maximum latency, even running on an older machine. Xenomai

---

[3]Ubuntu 16.04 with Xenomai 3.0.5 Live CD/USB: http://arantxa.ii.uam.es/~gnb/rtubuntu-16.04.4-1.0-desktop-amd64.iso

was not far from this performance, with a maximum latency of $5.66\mu s$. Preempt-RT had slightly worse results, reaching a maximum latency of $15.94\mu s$, but still acceptable for our purposes. The system without real-time is not reliable when millisecond precision or below is required, as it goes over the millisecond barrier in these tests (indicated in red in **Figure 3**). Nevertheless, as mentioned in section 2.2.1, Linux operating systems allow to isolate a processing core and bind a specific task to it. In this scenario, the task running over the isolated core will never be interrupted by any other users tasks, but system processes can still use this core. When this is done in RTAI or Xenomai it has little impact on their already good latency values, but we observe a remarkable improvement of latency values with both non real-time and Preempt-RT operating systems. Without an RTOS, it can be an useful tool in soft real-time environments.
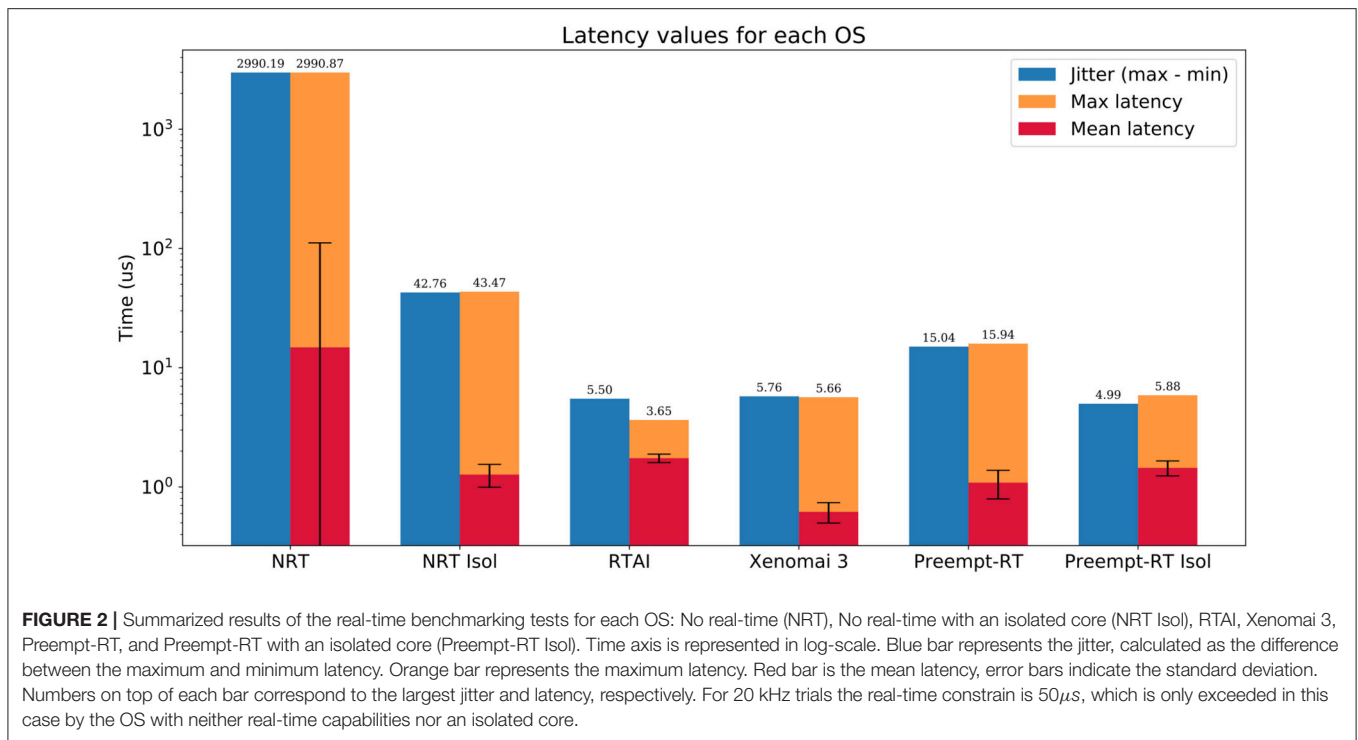
Unfortunately, regarding the analyzed RTOS, the better their performance is, the worse their usability and user-friendliness. As summarized in **Table 3**, RTAI is quite difficult to install and use, even for experimented users, while Preempt-RT is the most accessible, since there are not many differences with a normal GPOS. Due to this results, RTHybrid was developed to run over both Xenomai and Preempt-RT to balance performance and user-friendliness.

3.2. RTHybrid

3.2.1. Software Design and Implementation

RTHybrid is designed to be an user-friendly and accessible real-time tool for any researcher, regardless of their budget or computer science and programming knowledge. It is an open-source project that can be downloaded for free from its Github

FIGURE 2 | Summarized results of the real-time benchmarking tests for each OS: No real-time (NRT), No real-time with an isolated core (NRT Isol), RTAI, Xenomai 3, Preempt-RT, and Preempt-RT with an isolated core (Preempt-RT Isol). Time axis is represented in log-scale. Blue bar represents the jitter, calculated as the difference between the maximum and minimum latency. Orange bar represents the maximum latency. Red bar is the mean latency, error bars indicate the standard deviation. Numbers on top of each bar correspond to the largest jitter and latency, respectively. For 20 kHz trials the real-time constrain is $50\mu s$, which is only exceeded in this case by the OS with neither real-time capabilities nor an isolated core.

repository[4]. Any Linux distribution is supported, including those running Xenomai 3 and Preempt-RT real-time patches. The code is written using C/C++ language and compiled with GCC 6.3 and QMake 3.0. Relevant information regarding the hybrid circuit experiment, such as neuron (both living and model) and synapse types employed, parameters and latency values is registered in log files. Detailed instructions on how to download, configure and install both RTHybrid and all its dependencies are provided in its user manual [5].

Three processing threads have been used to build the program's architecture in order to address an optimized real-time software implementation (see **Figure 4**). Two execution modes are available: graphical user interface (GUI) mode or script mode. In the former, an intuitive GUI (see **Figure 5**) is displayed when the application is launched, where the user can select the desired models and modify all their parameters as well as set the experiment configuration (duration, DAQ channels, sampling rate, etc). This GUI has been designed using the Qt 5.10 framework[6]. The second option allows to automatically load all experimental parameters from an XML text file[7] and then sequentially execute various experimental protocols using scripts. Instructions on how to use the program and examples of XML configuration files can be found in RTHybrid user manual. Whatever option is chosen, the first and main process is in charge

of gathering the information from either source and starting two new threads with it.

The second thread performs all tasks that need to be completed with real-time precision in a periodic loop. Within each interval of this loop, key operations are fulfilled: interaction with the DAQ device, synapse and neuron models computation, automatic calibration, drift adaptation, etc. (see **Figure 6**). Communication with DAQ devices is achieved through Comedi open-source drivers for National Instruments' and several other manufacturers' hardware (or Analogy drivers in the case of Xenomai) (Schleef et al., 2012). Automatic calibration and experiment automation algorithms are included to deal with the differences between models and living neurons in terms of temporal scale and amplitude (Reyes-Sanchez et al., 2018). They also cover other possible experimental complications such as the presence of signal drift.

Some tasks are computationally too expensive to be carried out within the temporal restrictions established in a real-time interval, including essential ones as writing data to a file or printing it on screen. A third thread is used to write experiment data to files without disturbing the real-time performance. An inter-process message queue is utilized to send information from one thread to the other. Since it is not a real-time process, it will wait until there are enough computer resources available, and only then reads from the queue and stores the data into the files.
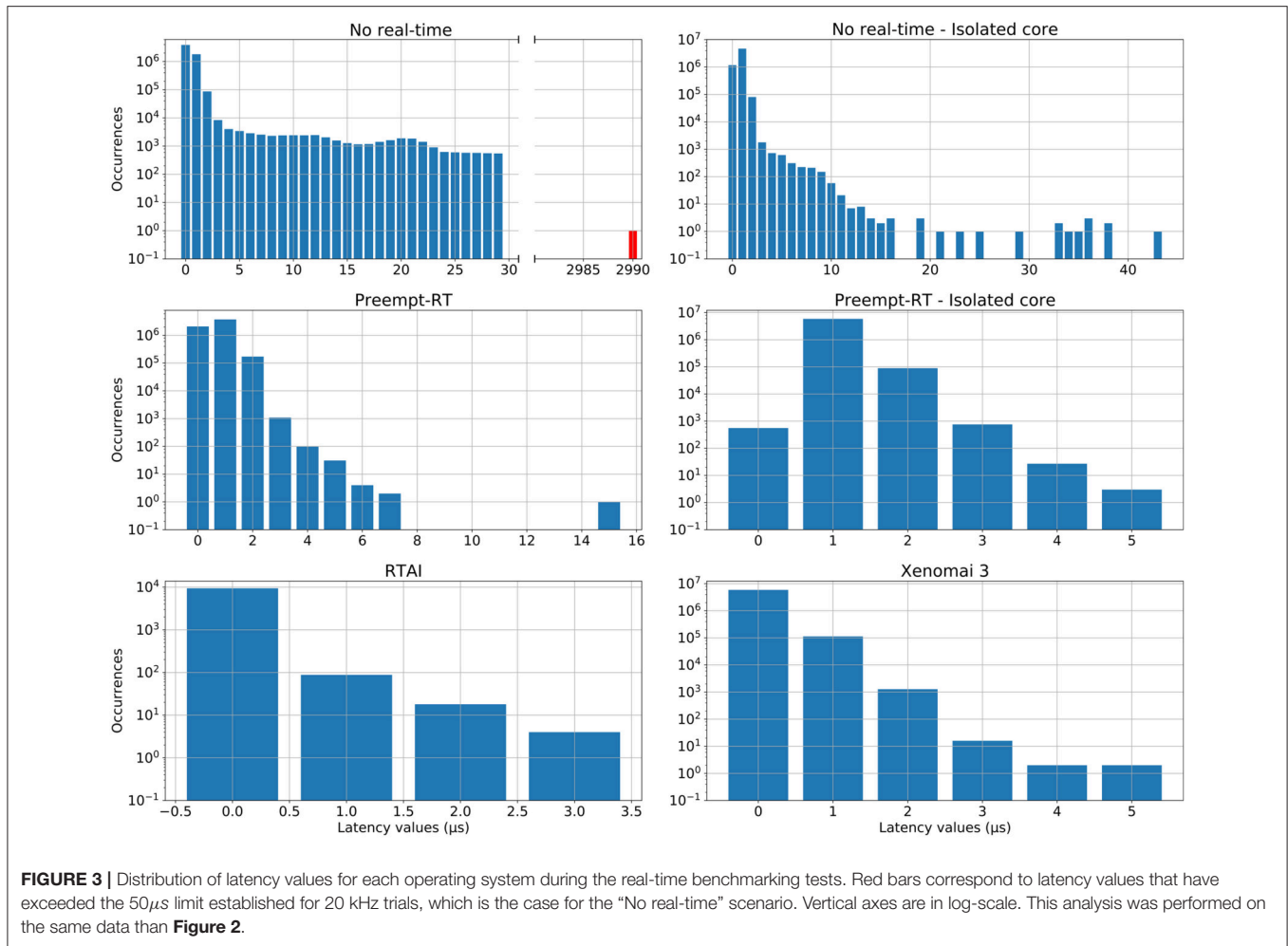
3.2.2. Model Library
RTHybrid includes a customizable library of neuron and synapse models to build a wide variety of hybrid circuits. **Table 4** shows the currently included neuron models, which have been selected due to their rich intrinsic dynamics and suitability to

---

[4]Source code for RTHybrid: www.github.com/GNB-UAM/RTHybrid
[5]RTHybrid User Manual: www.github.com/GNB-UAM/RTHybrid/blob/master/docs/RTHybrid_User_Manual.pdf
[6]Qt5 website: www.qt.io/
[7]libxml2 library website: http://xmlsoft.org/

FIGURE 3 | Distribution of latency values for each operating system during the real-time benchmarking tests. Red bars correspond to latency values that have exceeded the $50\mu s$ limit established for 20 kHz trials, which is the case for the "No real-time" scenario. Vertical axes are in log-scale. This analysis was performed on the same data than **Figure 2**.

build hybrid circuits. All of them have different characteristics and require also specific adaptations to work in real-time, which are handled automatically by auto-calibration algorithms (Reyes-Sanchez et al., 2018). Several numerical integration methods can be chosen by the user, including Euler, Heun, Order 4 Runge-Kutta (Press et al., 1988) and (6)5 Runge-Kutta (Hull et al., 1972). **Table 5** lists the types of synapse models included in the library at the moment and the different input parameters that they require to work in a hybrid circuit implementation.

Models described with differential equations require minimum integration steps and specific adaptations to match the duration of events such as spikes or bursts in a given living neuron. This is the case of models such as Izhikevich (2003), Hindmarsh and Rose (1984), Ghigliazza and Holmes (2004), and Wang (1993). Other neuron models, such as the Rulkov map (Rulkov, 2002), generate activity events such as bursts using a little number of points. In this case, interpolation of the values produced by the model is required to match the living neuron's event resolution at the chosen sampling rate. Synapse models also present differences in their implementation, both in the input parameters required and the

computational cost. A simple linear electrical synapse model is included along with more complex non-graded and graded chemical synapse paradigms (Destexhe et al., 1994; Golowasch et al., 1999). New models can be easily added to the library using C language.

Figure 6 illustrates the average time consumed by each task at each iteration of the loop executed by RTHybrid real-time thread. This study was performed for several models of the RTHybrid library. The models were run at a 10 kHz frequency for 5min (300s), i.e., each model test contained 3 million intervals of $100\,\mu s$ duration. Neuron models were bidirectionally connected through a graded chemical synapse model to a hardware-implemented Hindmarsh-Rose model that generated bursting activity at the same characteristic rate of a pyloric CPG cell (Pinto et al., 2000). Synaptic conductances were set to $g = 0.02\mu S$ and all other parameters from neuron and synapse models were fixed to produce bursting behavior. Burst duration for the models was set at 1s per burst. The computer used for these tests was the one referred as Computer 1 in **Table 1**, with both Preempt-RT and Xenomai 3 and the same DAQ device and board described in section 2.2.2. Models selected for this analysis included one low-cost differential equation implementation (Hindmarsh and Rose,
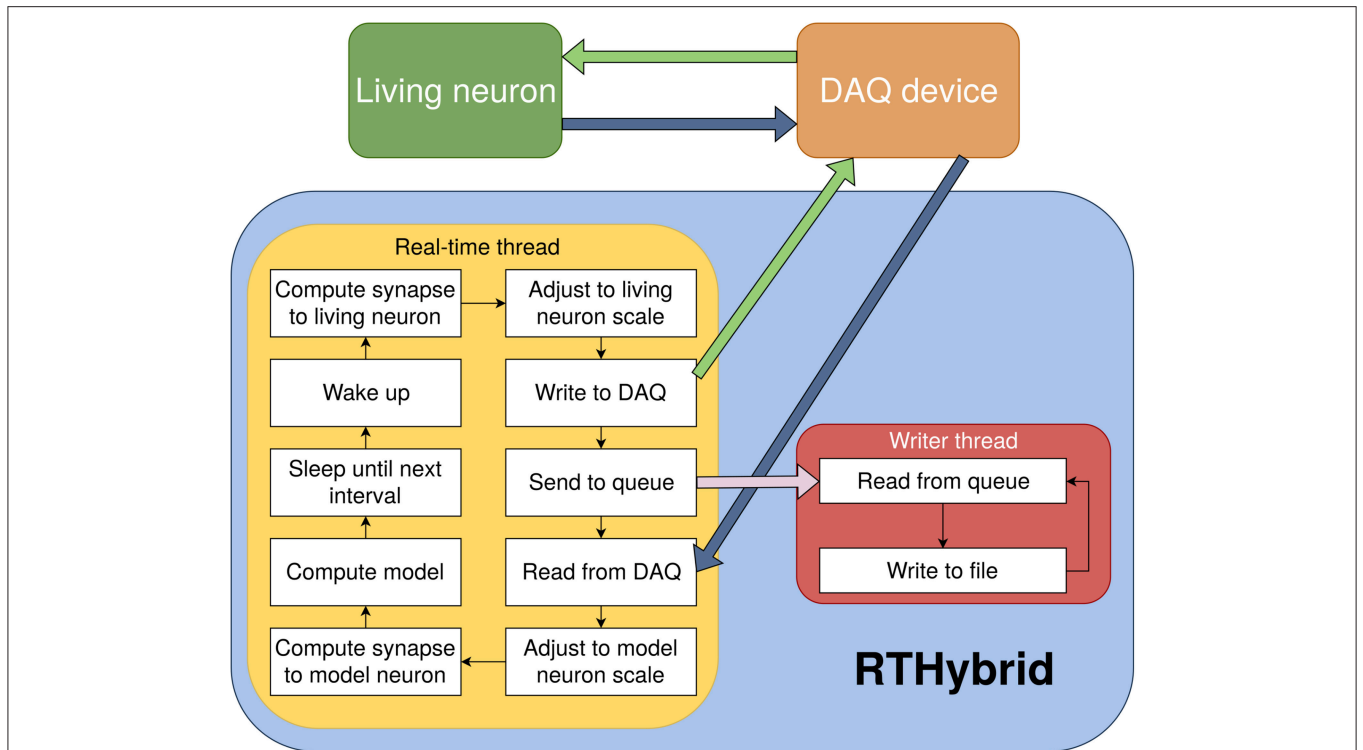
FIGURE 4 | Diagram of RTHybrid architecture. The computer receives the membrane voltage from the living neurons through the DAQ device, computes the neurons and synapses models and sends current back to the biological cells. The main program process creates two sub-threads, *rt_thread* and *writer_thread*, to manage the different tasks of the program. Both threads communicate through inter-process communication (IPC) message queues.
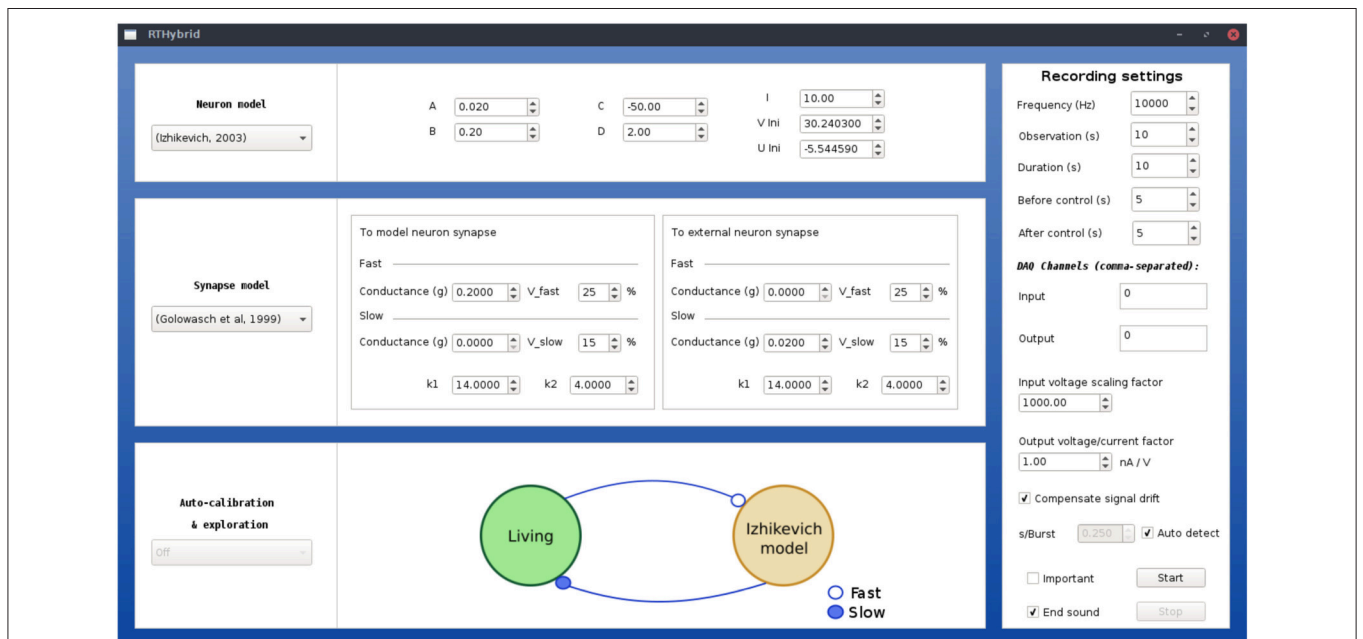

FIGURE 5 | Illustration of RTHybrid graphical user interface to build hybrid circuit interactions. Users can select neuron and synapse models, as well as their parameters, and the experiment settings, such as input/output DAQ channels, sampling frequency, duration, etc.
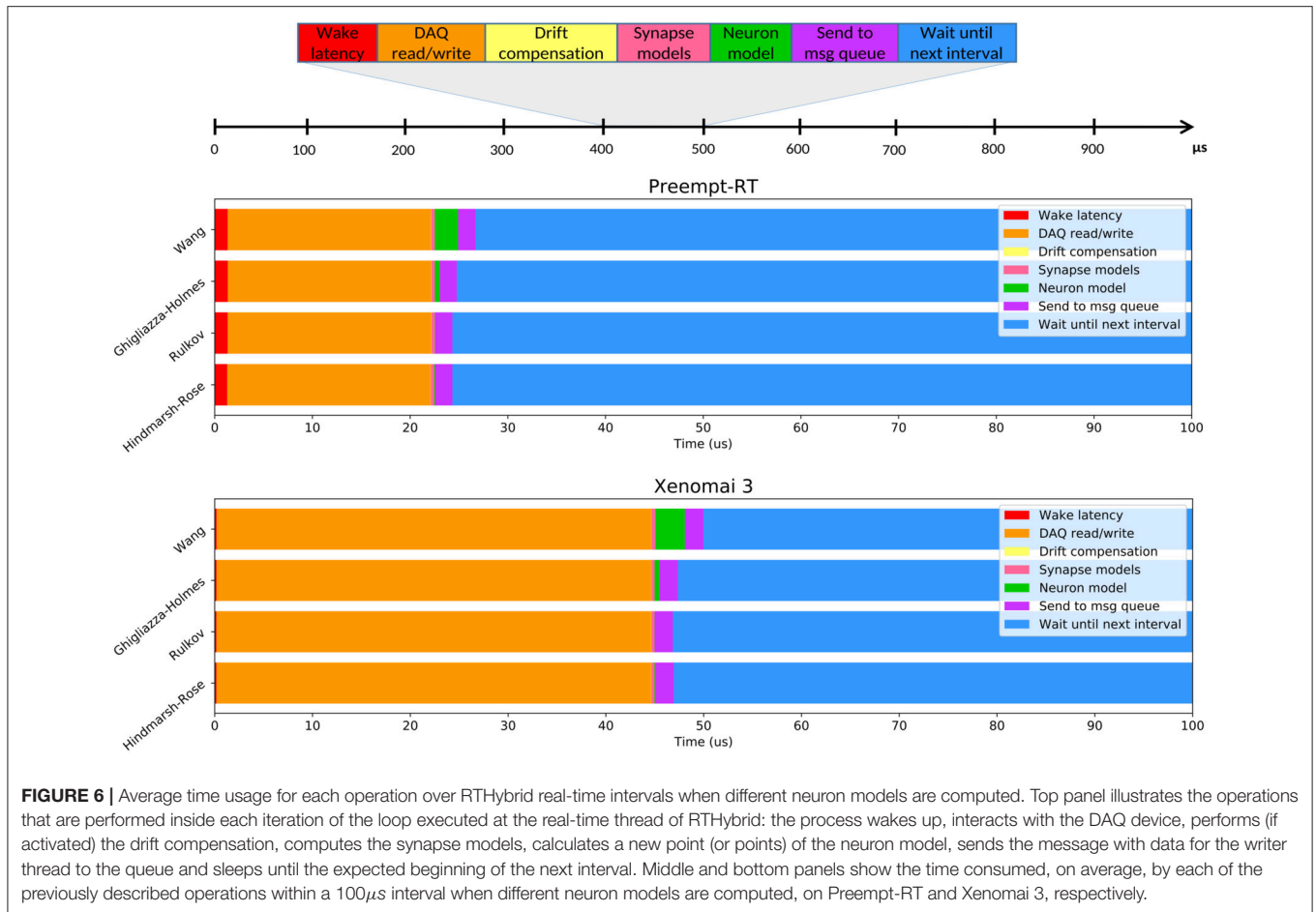
FIGURE 6 | Average time usage for each operation over RTHybrid real-time intervals when different neuron models are computed. Top panel illustrates the operations that are performed inside each iteration of the loop executed at the real-time thread of RTHybrid: the process wakes up, interacts with the DAQ device, performs (if activated) the drift compensation, computes the synapse models, calculates a new point (or points) of the neuron model, sends the message with data for the writer thread to the queue and sleeps until the expected beginning of the next interval. Middle and bottom panels show the time consumed, on average, by each of the previously described operations within a $100\mu s$ interval when different neuron models are computed, on Preempt-RT and Xenomai 3, respectively.

TABLE 4 | Currently available neuron models in RTHybrid.

Neuron model	Computational cost	Adaptation to RT method
Rulkov[a]		Interpolation
Izhikevich[b]	Low	
Hindmarsh-Rose[c]		Select best integration step
Conductance-based[d,e]	Expensive	
		Tabulation (in some cases)

Different models have different computational costs and need to be adapted to real-time performance using distinct methods. [a]Rulkov (2002), [b]Izhikevich (2003), [c]Hindmarsh and Rose (1984), [d]Ghigliazza and Holmes (2004), [e]Wang (1993).

TABLE 5 | Currently available synapse models in RTHybrid.

Synapse model	Input parameters
Electrical	V post and V pre
Chemical[a]	Presynaptic spike time and V post
Graded[b]	V post and V pre

Different kinds of model may require different input parameters to work in a hybrid circuit configuration. V refers to membrane potential of a living or model neuron. [a]Destexhe et al. (1994), [b]Golowasch et al. (1999) .

1984), a fast map model (Rulkov, 2002) and two conductance-based models with exponential non-linearities (Wang, 1993; Ghigliazza and Holmes, 2004).

Variability of the time consumed by the operations performed inside each interval (see **Table 6**) is constrained by the use of RTOS, being Xenomai 3 more effective in this task than Preempt-RT. Xenomai 3 achieved much lower latency values, but the use of the Analogy framework, instead of Comedi, translated into a higher cost when interacting with the DAQ device. Some models have on average low computation times

which may increase occasionally due to their nonlinearities or to a non stable regime for the chosen integration method. This is the case for the Hodgkin-Huxley type model from Wang (1993), which on <10 occasions on every trial, out of the three million iterations described previously for **Figure 6**, had an abnormally high computation time. Other models can always be computationally expensive because of different reasons such as their high dimensionality or multi-compartmental nature. One way to reduce their integration time and make them more efficient and suitable for real-time environments is to tabulate their nonlinearities with high resolution taking advantage or the large memory capacity of modern computers. When the operations within an interval exceed the temporal margin

TABLE 6 | Duration analysis for each operation performed inside the real-time cycle described in **Figure 6** on every iteration.

	Preempt-RT		Xenomai 3	
	Mean ± Std(μs)	Max (μs)	Mean ± Std(μs)	Max (μs)
Wake latency	1.3 ± 0.1	64.649	0.18 ± 0.05	19.844
DAQ read/write	20.8 ± 0.3	55.807	44.5 ± 0.3	49.973
Drift compensation	0.022 ± 0.006	8.654	0.022 ± 0.007	0.711
Synapse models	0.35 ± 0.04	12.486	0.4 ± 0.1	12.740
Send to queue	1.77 ± 0.09	37.449	1.88 ± 0.08	5.599
NEURON MODELS				
Hindmarsh-Rose	0.10 ± 0.01	5.810	0.099 ± 0.005	0.832
Rulkov	0.03 ± 0.01	9.319	0.021 ± 0.008	0.709
Ghigliazza-Holmes	0.49 ± 0.04	4.937	0.51 ± 0.05	6.159
Wang	2.4 ± 0.2	136.114	3 ± 1	122.707

Mean and maximum times are shown in microseconds. The use of RTOS constrains the variability of these times, with Xenomai 3 being more efficient at this than Preempt-RT. Duration of each model computation differs greatly from the others due to their distinct mathematical descriptions.

established, RTHybrid tries to minimize the negative impact on the system by taking time from the next interval and reducing the sleeping period, as can be seen in **Figure 7**. The presence of latency values above the real-time threshold can be detected from the log file.

3.2.3. Validation With Hybrid Circuits

Proper performance of RTHybrid neuron and synapse models was tested building hybrid circuits as detailed in section 2.2.2. Living neurons from the pyloric CPG were bidirectionally connected through chemical graded synapse models with neuron models. Four trials per model were conducted, each of them 5 min long, with one a minute long control period before and after the hybrid circuit interaction. The first two trials were performed with a sampling frequency of 10 kHz, thus cycle interval duration was $100\mu s$. The remaining two trials were run at 20 kHz, and the interval was $50\mu s$ long. Any latency value exceeding that limit was considered a real-time failure. All four trials per model were repeated both in Preempt-RT and Xenomai 3, without core isolation.
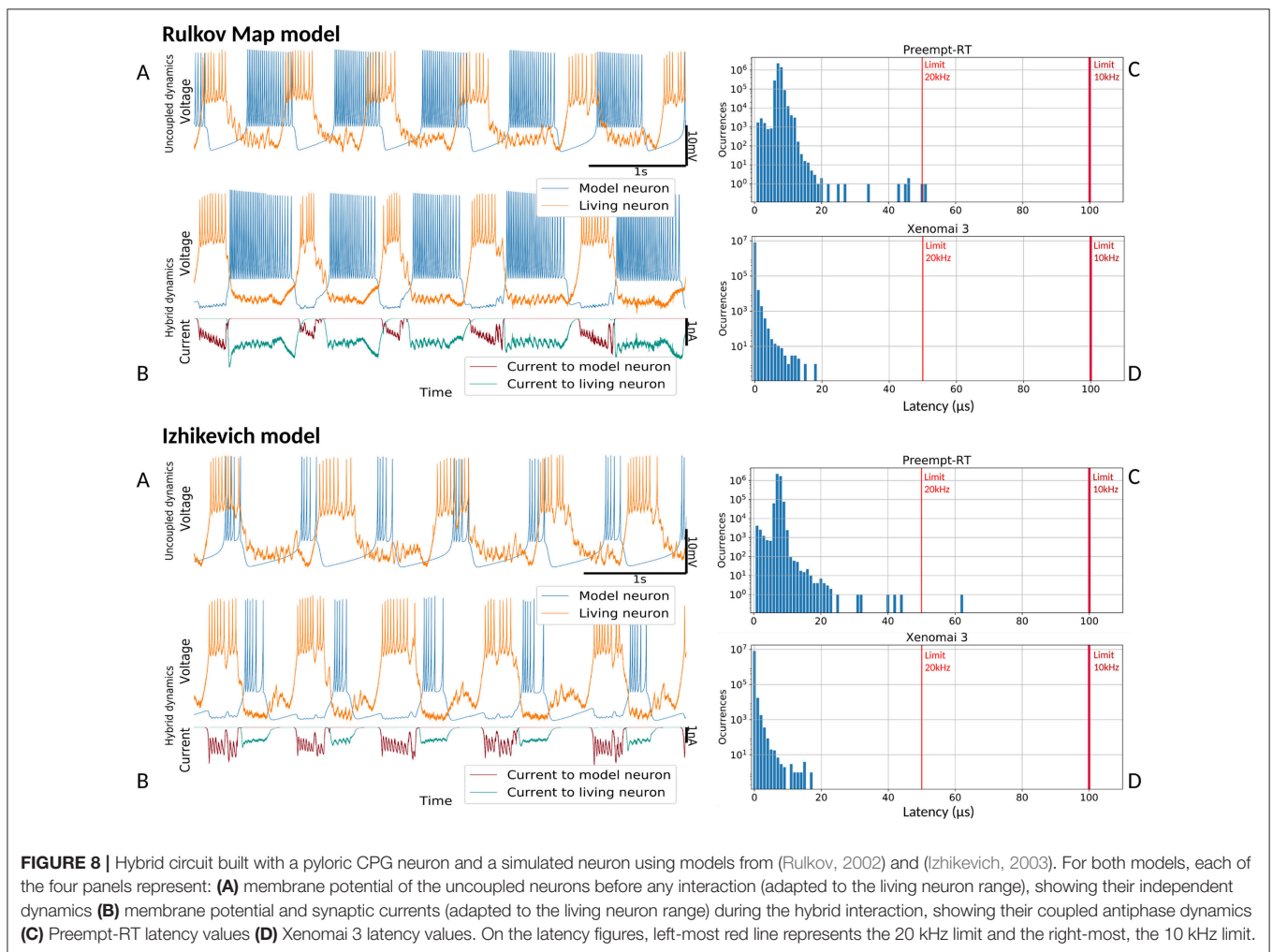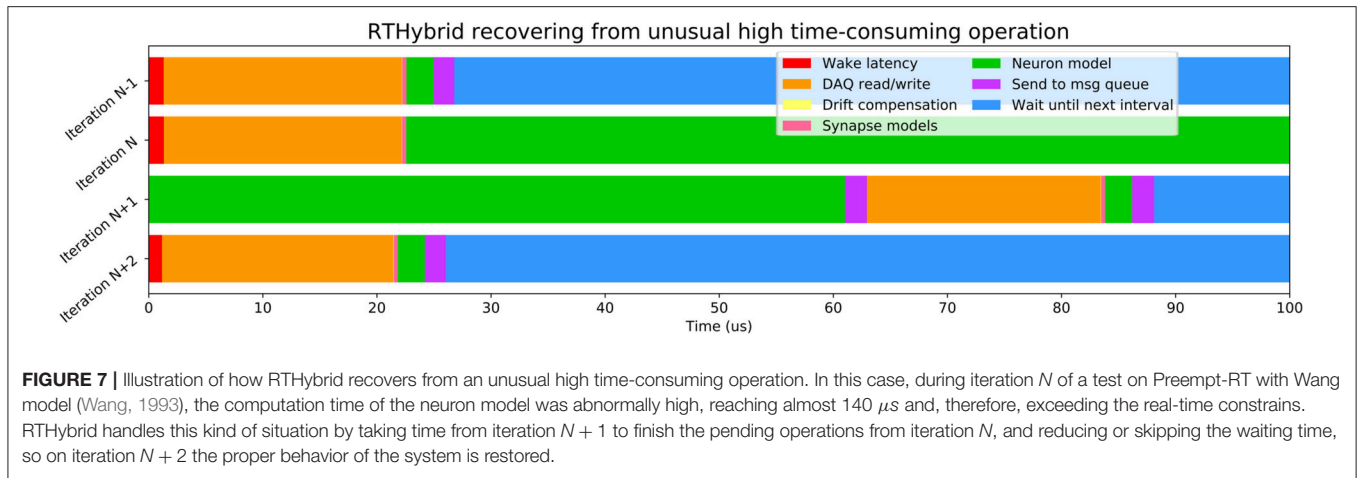
Connections in these hybrid circuits mimicked graded chemical synapses with fast and slow dynamics (Golowasch et al., 1999). The connection from the model to the living neuron was built with a slow synapse. A fast graded synapse was used for the connection from the living neuron to the model. Our target was to achieve rhythmic antiphase behavior between the neurons so we set inhibitory synapses in both directions. The slow synapse had a conductance of $g = 0.2\mu S$, $V_{th} = 15\%$ and $s = 1\%$ of the maximum amplitude range and kinetic parameters $k_1 = 14.0$ and $k_2 = 4.0$, except when using Hindmarsh-Rose model that the conductance was $g = 0.1\mu S$. The fast synapse had $V_{th} = 50\%$ and $s = 5\%$ of the maximum amplitude range and different conductances depending on the neuron model at use: for Izhikevich and Ghigliazza-Holmes models the conductance was $g = 0.8\mu S$, for Hindmarsh-Rose model it was $g = 1.0\mu S$ and for Rulkov was $g = 0.2\mu S$. Input voltage scaling factor was set to 100 and output current/voltage conversion factor to

$10nA/V$. Neuron model parameters were set to produce bursting behavior and the duration of each burst was set to automatically match the living neuron activity. Signal amplitude and temporal scaling was automatically performed by RTHybrid calibration algorithms (Reyes-Sanchez et al., 2018). Validation tests are shown in **Figures 8**, **9**.

Figure 8 shows the results of the validation tests for the Rulkov and Izhikevich models, and **Figure 9** displays the results with the Hindmarsh-Rose and Ghigliazza-Holmes models. For each model, sorted by increasing complexity, there are four panels displaying different information about the hybrid interaction. The worst performance trials for each model, understood as those with higher latency values, were selected for the analysis. Recorded membrane potential when both living and model neurons are uncoupled is displayed in (**Figures 8A**, **9A**). Voltage is scaled to the living neuron range, showing the independent behavior of each neuron's bursting activity when there is no interaction. Membrane potential during the hybrid interaction along with the synaptic current injected in both directions is shown in (**Figures 8B**, **9B**), portraying the robust antiphase rhythm achieved due to the bidirectional inhibitory graded chemical connectivity. (**Figures 8C**, **9C**) represents Preempt-RT test latency values, showing that in all cases the $100\mu s$ limit established for 10 kHz trials, represented by the right-most red vertical line, was fulfilled. However, this RTOS was not able to keep 20 kHz constrains, indicated by the left-most red vertical line, during these experiments. This was not the case for Xenomai 3, portrayed at (**Figures 8D**, **9D**), whose latency values were far under the $50\mu s$ barrier set during 20 kHz trials. Due to this outcome, the results displayed correspond to Preempt-RT 10 kHz and Xenomai 20 kHz trials.
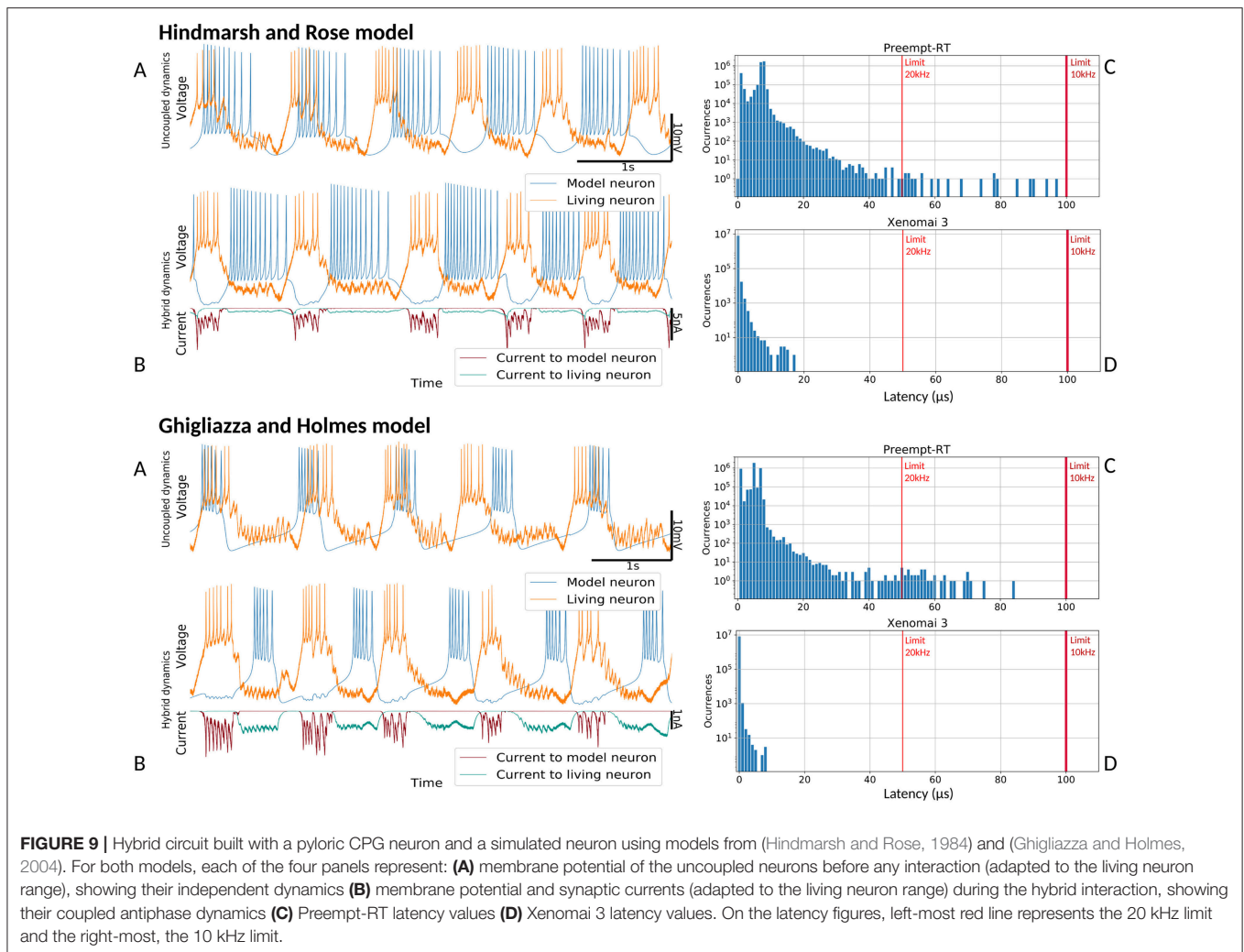
4. DISCUSSION

Characterization and control of neural systems dynamics, as well as experimental protocol automation can largely benefit from the use of closed-loop techniques, and specifically from hybrid circuits built by connecting model neurons and synapses to living cells. RTHybrid provides a neuron and synapse model library aimed to build hybrid circuits in an easy and user-friendly manner with large flexibility provided by a real-time software approach that takes advantage of the computational and memory capacity of modern computers. Developed for Linux, this program is open-source and can be downloaded for free from www.github.com/GNB-UAM/RTHybrid, where installation guide, user manuals and examples can also be found. RTHybrid has a simple GUI to design and configure the hybrid experiments. This tool also incorporates a command-line mode where configuration XML files can be loaded, a useful feature for experiment automation and scripting for activity-dependent exploration of neural dynamics. Beyond a standard Comedi compatible DAQ board, no additional hardware requirements are needed. Thus, standard computers used for data acquisition in any lab can run this software, for example in a dual-boot configuration with another hard-drive with a GPOS setup, e.g., Windows, to keep existing protocol configurations.

FIGURE 7 | Illustration of how RTHybrid recovers from an unusual high time-consuming operation. In this case, during iteration N of a test on Preempt-RT with Wang model (Wang, 1993), the computation time of the neuron model was abnormally high, reaching almost 140 μs and, therefore, exceeding the real-time constrains. RTHybrid handles this kind of situation by taking time from iteration $N + 1$ to finish the pending operations from iteration N, and reducing or skipping the waiting time, so on iteration $N + 2$ the proper behavior of the system is restored.

FIGURE 8 | Hybrid circuit built with a pyloric CPG neuron and a simulated neuron using models from (Rulkov, 2002) and (Izhikevich, 2003). For both models, each of the four panels represent: **(A)** membrane potential of the uncoupled neurons before any interaction (adapted to the living neuron range), showing their independent dynamics **(B)** membrane potential and synaptic currents (adapted to the living neuron range) during the hybrid interaction, showing their coupled antiphase dynamics **(C)** Preempt-RT latency values **(D)** Xenomai 3 latency values. On the latency figures, left-most red line represents the 20 kHz limit and the right-most, the 10 kHz limit.

Temporal precision requirements for closed-loop interactions in RTHybrid are fulfilled by using hard real-time software technology. An extensive analysis of available RTOS was conducted to select the most suitable platforms to implement RTHybrid. The software has been developed to run over Preempt-RT and Xenomai 3 frameworks for Linux due to

their balance between performance and accessibility. RTHybrid model library includes a wide variety of neuron models: from computationally-unexpensive paradigms, such as Rulkov (2002), Izhikevich (2003), and Hindmarsh and Rose (1984), to realistic conductance-based Hodgkin-Huxley type (Wang, 1993; Ghigliazza and Holmes, 2004). The library also contains several

FIGURE 9 | Hybrid circuit built with a pyloric CPG neuron and a simulated neuron using models from (Hindmarsh and Rose, 1984) and (Ghigliazza and Holmes, 2004). For both models, each of the four panels represent: **(A)** membrane potential of the uncoupled neurons before any interaction (adapted to the living neuron range), showing their independent dynamics **(B)** membrane potential and synaptic currents (adapted to the living neuron range) during the hybrid interaction, showing their coupled antiphase dynamics **(C)** Preempt-RT latency values **(D)** Xenomai 3 latency values. On the latency figures, left-most red line represents the 20 kHz limit and the right-most, the 10 kHz limit.

synapse models: from simple gap junctions implementations to configurable chemical synapses (Destexhe et al., 1994; Golowasch et al., 1999). All of them are adapted to work under hard real-time restrictions and additional neuron, network and synapse models can be easily added using C language. Moreover, calibration algorithms are integrated in the library to automate the adaptation of model amplitude and time scales to the living neuron behavioral range (Reyes-Sanchez et al., 2018). Core isolation can also be employed as validated in the tests reported in this paper.

Models currently included in the RTHybrid library are suitable for a wide variety of hybrid circuit experiments implemented using dynamic clamp. Beyond electrophysiological protocols, RTHybrid can also be easily generalized to drive open- and closed-loop interactions in optogenetic and drug microinjection paradigms. Future development in parallelization and GPU computing will be considered to implement large scale network or highly-realistic biophysical models.

Many researchers and laboratories overlook closed-loop techniques despite their advantages due to the difficulties in the installation and use of the required technology. With RTHybrid, we aim to encourage the use of open-source, standardized and user-friendly real-time software tools, available in different platforms, to facilitate the implementation of closed-loop experimentation in neuroscience research.

ETHICS STATEMENT

AUTHOR CONTRIBUTIONS

RA, MR-S, and PV conceived of and designed the project. RA and MR-S wrote, tested, and documented the software. RA, MR-S, and IE and performed the experiments. RA, MR-S, IE, FR, and PV analyzed the data and wrote the paper.

FUNDING

REFERENCES

Abbott, D. (ed.). (2006). *Linux for Embedded and Real-Time Applications, 2nd Edn.* Oxford: Newnes.

Aroca, R. V., and Caurin, G. (2009). "A real time operating systems (RTOS) comparison," in *WSO-Workshop de Sistemas Operacionais*, 12. (Bento Gonçalves).

Arrouët, C., Congedo, M., Marvie, J.-E., Lamarche, F., Lécuyer, A., and Arnaldi, B. (2005). Open-ViBE: a three dimensional platform for real-time neuroscience. *J. Neurother.* 9, 3–25. doi: 10.1300/J184v09n01_02

Arsiero, M., Lüscher, H. R., and Giugliano, M. (2007). Real-time closed-loop electrophysiology: towards new frontiers in *in vitro* investigations in the neurosciences. *Arch. Ital. Biol.* 145, 193–209. doi: 10.4449/aib.v145i3.186

Biró, I., and Giugliano, M. (2015). A reconfigurable visual-programming library for real-time closed-loop cellular electrophysiology. *Front. Neuroinform.* 9:17. doi: 10.3389/fninf.2015.00017

Broccard, F. D., Joshi, S., Wang, J., and Cauwenberghs, G. (2017). Neuromorphic neural interfaces: from neurophysiological inspiration to biohybrid coupling with nervous systems. *J. Neural Eng.* 14:041002. doi: 10.1088/1741-2552/aa67a9

Brochini, L., Carelli, P. V., and Pinto, R. D. (2011). Single synapse information coding in intraburst spike patterns of central pattern generator motor neurons. *J. Neurosci.* 31, 12297–12306. doi: 10.1523/JNEUROSCI.1568-11.2011

Chamorro, P., Levi, R., Rodríguez, F. B., Pinto, R. D., and Varona, P. (2009). Real-time activity-dependent drug microinjection. *BMC Neurosci.* 10(Suppl. 1):P296. doi: 10.1186/1471-2202-10-S1-P296

Chamorro, P., Muñiz, C., Levi, R., Arroyo, D., Rodriguez, F. B., and Varona, P. (2012). Generalization of the dynamic clamp concept in neurophysiology and behavior. *PLoS ONE* 7:e40887. doi: 10.1371/journal.pone.0040887

Christini, D. J., Stein, K. M., Markowitz, S. M., and Lerman, B. B. (1999). Practical real-time computing system for biomedical experiment interface. *Ann. Biomed. Eng.* 27, 180–186. doi: 10.1114/1.185

Ciliberti, D., and Kloosterman, F. (2017). Falcon: a highly flexible open-source software for closed-loop neuroscience. *J. Neural Eng.* 14045004. doi: 10.1088/1741-2552/aa7526

Desai, N. S., Gray, R., and Johnston, D. (2017). A dynamic clamp on every rig. *eneuro* 10:ENEURO.0250-17.2017. doi: 10.1523/ENEURO.0250-17.2017

Destexhe, A., and Bal, T. (2009). "Dynamic-clamp: from principles to applications," in *From Principles to Applications*, Vol. 1, eds A. Destexh and T. Bale (New York, NY: Springer-Verlag), 443.

Destexhe, A., Mainen, Z. F., and Sejnowski, T. J. (1994). An efficient method for computing synaptic conductances based on a kinetic model of receptor binding. *Neural Comput.* 6, 14–18. doi: 10.1162/neco.1994.6.1.14

Dietrich, S.-T., and Walker, D. (2005). "The evolution of real-time linux," in *7th Real-Time Linux Workshop*, 3–4.

Dorval, A. D., Christini, D. J., and White, J. A. (2001). Real-time linux dynamic clamp: a fast and flexible way to construct virtual ion channels in living cells. *Ann. Biomed. Eng.* 29, 897–907. doi: 10.1114/1.1408929

Elices, I., Levi, R., Arroyo, D., Rodriguez, F. d. B. B., and Varona, P. (2018). Robust dynamical invariants in sequential neural activity. *bioRxiv [Preprint]*. doi: 10.1101/379909

Forlim, C. G., Pinto, R. D., Varona, P., and Rodríguez, F. B. (2015). Delay-dependent response in weakly electric fish under closed-loop pulse stimulation. *PLoS ONE* 10:e0141007. doi: 10.1371/journal.pone.0141007

Franke, F., Jäckel, D., Dragas, J., Müller, J., Radivojevic, M., Bakkum, D., and Hierlemann, A. (2012). High-density microelectrode array recordings and real-time spike sorting for closed-loop experiments: an emerging technology to study neural plasticity. *Front. Neural Circ.* 6:105. doi: 10.3389/fncir.2012.00105

Furht, B., Grostick, D., Gluch, D., Rabbat, G., Parker, J., and McRoberts, M. (1991). "Introduction to real-time computing," in *Real-Time UNIX Systems* (Boston, MA: Springer), 1–35.

Gerum, P. (2004). "Xenomai-Implementing a RTOS Emulation Framework on GNU/Linux". in *Proceedings* Nucleus 1–12.

Ghigliazza, R. M., and Holmes, P. (2004). Minimal models of bursting neurons: how multiple currents, conductances, and timescales affect bifurcation diagrams. *Soc. Indus. Appl. Math.* 3, 636–670. doi: 10.1137/030602307

Golowasch, J., Casey, M., Abbott, L. F., and Marder, E. (1999). Network stability from activity-dependent regulation of neuronal conductances. *Neural Comput.* 11, 1079–1096. doi: 10.1162/089976699300016359

Grashow, R., Brookings, T., and Marder, E. (2010). Compensation for variable intrinsic neuronal excitability by circuit-synaptic interactions. *J. Neurosci.* 30, 9145–9156. doi: 10.1523/JNEUROSCI.0980-10.2010

Hambarde, P., Varma, R., and Jha, S. (2014). "The survey of real time operating system: RTOS," in *2014 International Conference on Electronic Systems, Signal Processing and Computing Technologies (IEEE)*, (Nagpur), 34–39.

Hazan, H., and Ziv, N. E. (2017). Closed loop experiment manager (CLEM)–an open and inexpensive solution for multichannel electrophysiological recordings and closed loop experiments. *Front. Neurosci.* 11:579. doi: 10.3389/fnins.2017.00579

Hindmarsh, J. L., and Rose, R. M. (1984). A model of neuronal bursting using three coupled first order differential equations. *Proc. R. Soc. Lond. B Biol. Sci.* 221, 87–102. doi: 10.1098/rspb.1984.0024

Hooper, R. M., Tikidji-Hamburyan, R. A., Canavier, C. C., and Prinz, A. A. (2015). Feedback control of variability in the cycle period of a central pattern generator. *J. Neurophysiol.* 114, jn.00365.2015. doi: 10.1152/jn.00365.2015

Hull, T. E., Enright, W. H., Fellen, B. M., Sedgwick, A. E., and Sedgwickt, A. E. (1972). Comparing numerical methods for ordinary differential equations. *Source* 9, 603–637. doi: 10.1137/0709052

Izhikevich, E. M. (2003). Simple model of spiking neurons. *IEEE Trans. Neural Netw.* 14, 1569–1572. doi: 10.1109/TNN.2003.820440

Kispersky, T. J., Economo, M. N., Randeria, P., and White, J. A. (2011). GenNet: a Platform for Hybrid Network Experiments. *Front. Neuroinform.* 5:11. doi: 10.3389/fninf.2011.00011

Krook-Magnuson, E., Armstrong, C., Oijala, M., and Soltesz, I. (2013). On-demand optogenetic control of spontaneous seizures in temporal lobe epilepsy. *Nat. Commun.* 4:1376. doi: 10.1038/ncomms2376

Lareo, A., Forlim, C. G., Pinto, R. D., Varona, P., and Rodriguez, F. d. B. (2016). Temporal code-driven stimulation: definition and application to electric fish signaling. *Front. Neuroinform.* 10:41. doi: 10.3389/fninf.2016.00041

Le Masson, G., Le Masson, S., and Moulins, M. (1995). From conductances to neural network properties: analysis of simple circuits using the hybrid network method. *Progr. Biophys. Mol. Biol.* 64, 201–220. doi: 10.1016/S0079-6107(96)00004-1

Le Masson, G., Renaud-Le Masson, S., Debay, D., and Bal, T. (2002). Feedback inhibition controls spike transfer in hybrid thalamic circuits. *Nature* 417, 854–858. doi: 10.1038/nature00825

Levi, T., Bonifazi, P., Massobrio, P., and Chiappalone, M. (2018). Editorial: Closed-loop systems for next-generation neuroprostheses. *Front. Neurosci.* 12:26. doi: 10.3389/fnins.2018.00026

Linaro, D., Couto, J., and Giugliano, M. (2014). Command-line cellular electrophysiology for conventional and real-time closed-loop experiments. *J. Neurosci. Methods* 230, 5–19. doi: 10.1016/j.jneumeth.2014.04.003

Mantegazza, P., Dozio, E. L., and Papacharalambous, S. (2000). RTAI: real-time application interface. *Linux J.* 2000, 10.

Monteiro, J. (2008). Building your way through RTAI. *Language* 1–14. Available online at: https://code.ua.pt/attachments/download/2171/Starting_With_RTAI4.pdf

Müller, J., Bakkum, D. J., and Hierlemann, A. (2013). Sub-millisecond closed-loop feedback stimulation between arbitrary sets of individual neurons. *Front. Neural Circ.* 6:121. doi: 10.3389/fncir.2012.00121

Muñiz, C., Arganda, S., de Borja Rodríguez, F., de Polavieja, G. G., and Varona, P. (2005). "Realistic stimulation through advanced dynamic-clamp protocols," in *Mechanisms, Symbols, and Models Underlying Cognition*, eds J. Mira and J. R. Álvarez (Berlin; Heidelberg; Springer Berlin Heidelberg), 95–105.

Muñiz, C., Forlim, C. G., Guariento, R. T., Pinto, R. D., Rodriguez, F. B., and Varona, P. (2011). Online video tracking for activity-dependent stimulation in neuroethology. *BMC Neurosci.* 12(Suppl. 1):P358. doi: 10.1186/1471-2202-12-S1-P358

Muñiz, C., Levi, R., Benkrid, M., Rodríguez, F. B., and Varona, P. (2008). Real-time control of stepper motors for mechano-sensory stimulation. *J. Neurosci. Methods* 172, 105–111. doi: 10.1016/j.jneumeth.2008.04.017

Muñiz, C., Rodríguez, F., and Varona, P. (2009). RTBiomanager: a software platform to expand the applications of real-time technology in neuroscience. *BMC Neurosci.* 10(Suppl. 1):P49. doi: 10.1186/1471-2202-10-S1-P49

Norman, S. E., Butera, R. J., and Canavier, C. C. (2016). Stochastic slowly adapting ionic currents may provide a decorrelation mechanism for neural oscillators

by causing wander in the intrinsic period. *J. Neurophysiol.* 116, 1189–1198. doi: 10.1152/jn.00193.2016

Nowotny, T., Szűcs, A., Pinto, R. D., and Selverston, A. I. (2006). StdpC: a modern dynamic clamp. *J. Neurosci. Methods* 158, 287–299. doi: 10.1016/j.jneumeth.2006.05.034

Nowotny, T., and Varona, P. (2014). "Dynamic clamp technique," in *Encyclopedia of Computational Neuroscience*, eds D. Jaeger and Jung R (New York, NY: Springer), 1–4. doi: 10.1007/978-1-4614-7320-6_126-2

Nowotny, T., Zhigulin, V. P., Selverston, A. I., Abarbanel, H. D. I., and Rabinovich, M. I. (2003). Enhancement of synchronization in a hybrid neural circuit by spike-timing dependent plasticity. *J. Neurosci.* 23, 9776–9785. doi: 10.1523/JNEUROSCI.23-30-09776.2003

Oprisan, S. A., Prinz, A. A., and Canavier, C. C. (2004). Phase resetting and phase locking in hybrid circuits of one model and one biological neuron. *Biophys. J.* 87, 2283–2298. doi: 10.1529/biophysj.104.046193

Patel, Y. A., George, A., Dorval, A. D., White, J. A., Christini, D. J., and Butera, R. J. (2017). Hard real-time closed-loop electrophysiology with the real-time eXperiment Interface (RTXI). *PLoS Comput. Biol.* 13:e1005430. doi: 10.1371/journal.pcbi.1005430

Pinto, R., Elson, R., Szücs, A., Rabinovich, M., Selverston, A., and Abarbanel, H. (2001). Extended dynamic clamp: controlling up to four neurons using a single desktop computer and interface. *J. Neurosci. Methods* 108, 39–48. doi: 10.1016/S0165-0270(01)00368-5

Pinto, R. D., Varona, P., Volkovskii, A. R., Szücs, A., Abarbanel, H. D. I., et al. (2000). Synchronous behavior of two coupled electronic neurons. *Phys. Rev. E* 62, 2644–2656. doi: 10.1103/PhysRevE.62.2644

Potter, S. M., El Hady, A., and Fetz, E. E. (2014). Closed-loop neuroscience and neuroengineering. *Front. Neural Circ.* 8:115. doi: 10.3389/fncir.2014.00115

Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P. (1988). "Chapter 16: Integration of ordinary differential equations," in *Numerical Recipes in C: The Art of Scientific Computing* (Cambridge University Press), 710–714.

Prinz, A. A., Abbott, L., and Marder, E. (2004). The dynamic clamp comes of age. *Trends Neurosci.* 27, 218–224. doi: 10.1016/j.tins.2004.02.004

Prsa, M., Galiñanes, G. L., and Huber, D. (2017). Rapid integration of artificial sensory feedback during operant conditioning of motor cortex neurons. *Neuron* 93, 929.e6–939.e6. doi: 10.1016/j.neuron.2017.01.023

Racciu, G. and Mantegazza, P. (2006). RTAI 3.4 User Manual. Available online at: https://www.rtai.org/userfiles/documentation/documents/RTAI_User_Manual_34_03.pdf

Rana, M., Varan, A. Q., Davoudi, A., Cohen, R. A., Sitaram, R., and Ebner, N. C. (2016). Real-time fMRI in neuroscience research and its use in studying the aging brain. *Front. Aging Neurosci.* 8:239. doi: 10.3389/fnagi.2016.00239

Reyes-Sanchez, M., Amaducci, R., Elices, I., Rodriguez, F. B., and Varona, P. (2018). Automatic adaptation of model neurons and connections to build hybrid circuits with living networks. *bioRxiv [Preprint].* doi: 10.1101/419622

Robinson, H. P., and Kawai, N. (1993). Injection of digitally synthesized synaptic conductance transients to measure the integrative properties of neurons. *J. Neurosci. Methods* 49, 157–165. doi: 10.1016/0165-0270(93)90119-C

Roth, E., Sponberg, S., and Cowan, N. J. (2014). A comparative approach to closed-loop computation. *Curr. Opin. Neurobiol.* 25, 54–62. doi: 10.1016/j.conb.2013.11.005

Rulkov, N. F. (2002). Modeling of spiking-bursting neural behavior using two-dimensional map. *Phys. Rev. E* 65, 041922. doi: 10.1103/PhysRevE.65.041922

Schleef, D., Mori, F., Bruyninckx, Porr, B., Abbott, I., and Piel, É. (2012). Comedi Documentation. Available online at: http://www.comedi.org/doc/

Sharp, A. A., O'Neil, M. B., Abbott, L. F., and Marder, E. (1993). The dynamic clamp: artificial conductances in biological neurons. *Trends Neurosci.* 16, 389–394.

Shin, K., and Ramanathan, P. (1994). Real-time computing: a new discipline of computer science and engineering. *Proc. IEEE* 82, 6–24. doi: 10.1109/5.259423

Siegle, J. H., López, A. C., Patel, Y. A., Abramov, K., Ohayon, S., and Voigts, J. (2017). Open Ephys: an open-source, plugin-based platform for multichannel electrophysiology. *J. Neural Eng.* 14:045003. doi: 10.1088/1741-2552/aa5eea

Sitaram, R., Ros, T., Stoeckel, L., Haller, S., Scharnowski, F., Lewis-Peacock, J., et al. (2016). Closed-loop brain training: the science of neurofeedback. *Nat. Rev. Neurosci.* 18, 86–100. doi: 10.1038/nrn.2016.164

Stallings, W. (2012). "Chapter 10: Multiprocessor and real-time scheduling," in *Operating Systems: Internals and Design Principles*, 7 Edn, eds M. Hirsch, T. Dunkelberger, and M. Haggerty (Boston, MA: Prentice Hall), 430–474.

Szücs, A., Varona, P., Volkovskii, A. R., Abarbanel, H. D. I., Rabinovich, M. I., and Selverston, A. I. (2000). Interacting biological and electronic neurons generate realistic oscillatory rhythms. *Neuroreport* 11, 563–569. doi: 10.1097/00001756-200002280-00027

Tessadori, J., Bisio, M., Martinoia, S., and Chiappalone, M. (2012). Modular neuronal assemblies embodied in a closed-loop environment: toward future integration of brains and machines. *Front. Neural Circ.* 6:99. doi: 10.3389/fncir.2012.00099

Thounaojam, U. S., Cui, J., Norman, S. E., Butera, R. J., and Canavier, C. C. (2014). Slow noise in the period of a biological oscillator underlies gradual trends and abrupt transitions in phasic relationships in hybrid neural networks. *PLoS Comput. Biol.* 10:e1003622. doi: 10.1371/journal.pcbi.1003622

Torres, J. J., and Varona, P. (2012). "Modeling Biological neural networks," in *Handbook of Natural Computing*, eds G. Rozenberg, T. Back, and J. N. Kok(Berlin; Heidelberg: Springer Berlin Heidelberg), 533–564.

Varona, P., Arroyo, D., Rodríguez, F., and Nowotny, T. (2016). "Online event detection requirements in closed-loop neuroscience," in *Closed Loop Neuroscience 2016*, ed A. El Hady (London: Elsevier), 81–91. doi: 10.1016/B978-0-12-802452-2.00006-8

Varona, P., Torres, J. J., Abarbanel, H. D., Rabinovich, M. I., and Elson, R. C. (2001). Dynamics of two electrically coupled chaotic neurons: experimental observations and model analysis. *Biol. Cybern.* 84, 91–101. doi: 10.1007/s004220000198

Wang, S., Chandrasekaran, L., Fernandez, F. R., White, J. A., and Canavier, C. C. (2012). Short conduction delays cause inhibition rather than excitation to favor synchrony in hybrid neuronal networks of the entorhinal cortex. *PLoS Comput. biol.* 8:e1002306. doi: 10.1371/journal.pcbi.1002306

Wang, X. J. (1993). Ionic basis for intrinsic 40 Hz neuronal oscillations. *Neuroreport* 5, 221–224. doi: 10.1097/00001756-199312000-00008

Yaghmour, K. (2003). "Chapter 13," in *Building Embedded Linux Systems 1st Edn.* ed. A. Oram (Sebastopol, CA: O'Reilly).

Yarom, Y. (1991). Rhythmogenesis in a hybrid system-interconnecting an olivary neuron to an analog network of coupled oscillators. *Neuroscience* 44, 263–275. doi: 10.1016/0306-4522(91)90053-Q

Conflict of Interest Statement: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.