Robot Locomotion through Tunable Bursting Rhythms using Efficient Bio-mimetic Neural Networks on Loihi and Arduino Platforms

Vijay Shankaran Vivekanand
University of Pittsburgh
Pittsburgh, USA
viv40@pitt.edu

Samarth Chopra
University of Pittsburgh
Pittsburgh, USA
sac345@pitt.edu

Shahin Hashemkhani
University of Pittsburgh
Pittsburgh, USA
shh199@pitt.edu

Rajkumar Kubendran
University of Pittsburgh
Pittsburgh, USA
rajkumar.ece@pitt.edu

Abstract

Rhythmic tasks that biological beings perform such as breathing, walking, and swimming, use specialized neural networks called central pattern generators (CPG). Spiking CPGs have already been implemented to control robot locomotion. This paper aims to take this concept further by designing and implementing a tunable bursting central pattern generator to control quadruped robots for the first time, to the best of our knowledge. Bursting CPGs allow for more granular control over the motion and speed of operation while retaining the low memory usage and latency capabilities of spiking CPGs. A bio-mimetic neuron model is chosen for this implementation which is highly optimized to run real-time on standard (Arduino microcontroller) and specialized (Intel Loihi) hardware. The Petoi bittle is chosen as the model hardware setup to showcase the efficiency of the proposed CPGs even in serial processing architectures. The CPG network is also realized in a completely asynchronous Loihi architecture to illustrate its versatility. The fully connected network running on CPG takes around 10 kilo bytes of memory (33% of Arduino capacity) to execute different modes of locomotion - walk, jump, trot, gallop, and crawl. Benchmarking results show that the bio-mimetic neurons take around 600 bytes (around 2%) more memory than Izhikevich neurons while being 0.02ms (around 14%) faster in isolated neuron testing.

ICONS '23, August 1–3, 2023, Santa Fe, NM, USA

ACM ISBN 979-8-4007-0175-7/23/08…$15.00
https://doi.org/10.1145/3589737.3605965

Keywords: Bursting Central Pattern Generators, Neuromorphic Computing, Network Neuromodulation

ACM Reference Format:
Vijay Shankaran Vivekanand, Samarth Chopra, Shahin Hashemkhani, and Rajkumar Kubendran. 2023. Robot Locomotion through Tunable Bursting Rhythms using Efficient Bio-mimetic Neural Networks on Loihi and Arduino Platforms. In *International Conference on Neuromorphic Systems (ICONS '23), August 1–3, 2023, Santa Fe, NM, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3589737.3605965

1 Introduction

Agile robots that can traverse unknown and dangerous terrains have the potential to enhance disaster response during floods and earthquakes or to access remote and unsafe areas like malfunctioning nuclear plants or space exploration. Mobile robots that rely on self-stabilizing and emergent behaviors can demonstrate decreased reliance on heavy computation and simple but effective ways to exhibit different types of motion using a single underlying framework [4]. Such a framework removes the need for elaborate programming through precise control algorithms demanding larger compute and memory resources. Moreover, there is a significant need to change locomotion behaviors on the fly, such as changing speeds or switching to a different mode, e.g. walking to crawling or trotting. A flexible and tunable control system using the hardware resources efficiently can make robots more versatile [5].

In biology, a key property of animals is the ability to efficiently move in complex environments [3]. Efficient locomotion in both vertebrate and invertebrate mammals is enabled through neural circuits forming Central Pattern Generator (CPG) networks. CPGs produce rhythmic patterns of neural activity without receiving rhythmic inputs or sensory feedback from the peripheral nervous system [6]. CPGs present several interesting properties such as distributed control, robustness and tolerance to perturbations, control loops in action at multiple timescales, and network modulation of

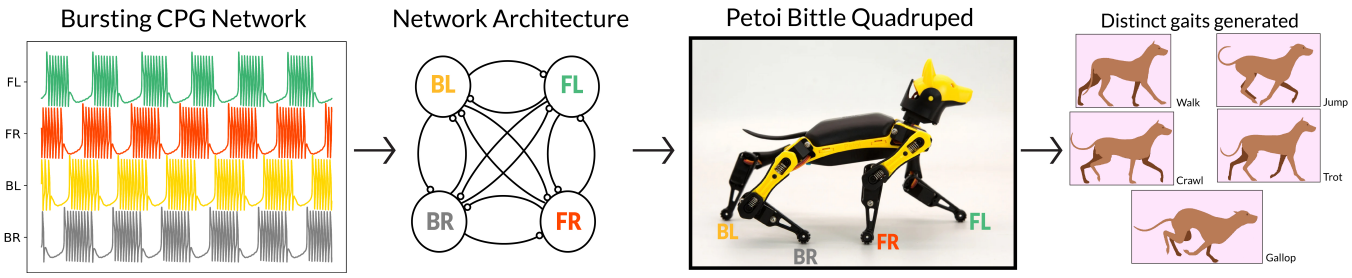

Figure 1. Overview of the proposed CPG network architecture to produce rhythmic bursting patterns for robotic locomotion control using non-linear bio-mimetic neurons. Each neuron is mapped to one limb which generates gait patterns according to the weights assigned (i.e. to synapses between the neurons). These neurons are configured to burst by setting the correct I_{ext}. Here, FL is Front Left, FR is Front Right, BR is Back Right and BL is Back Left.

locomotion using simple control signals. When transferred to mathematical models that can be implemented on electronic hardware, these properties can be realized for efficient locomotion control in robots.

Spiking CPGs have already been implemented to control robot locomotion [7, 12]. This paper aims to take this concept further by designing and implementing a bursting central pattern generator to control quadruped robots for the first time, to the best of our knowledge. Bursting CPGs allow for more granular control over the motion and speed of operation while retaining the ultra-low power and latency capabilities of spiking CPGs. The neuron chosen for this implementation is inspired by the model presented in [10] but highly optimized to run real-time on standard or specialized hardware with minimum resources while still retaining its bio-mimetic properties. The Petoi Bittle is chosen as the model robot with the Arduino hardware platform to showcase the efficiency of the model even in synchronous serial processing architecture. The CPG network is also realized in the neuromorphic asynchronous Loihi architecture, from Intel, to establish a proof-of-concept event-based system with extreme parallelism.

This paper is organized as follows. Section 2 provides a background of the Bio-Mimetic neuron model and robots using spiking CPG for locomtion. Sections 3 and 4 describe the implementation of bursting CPG networks on Arduino and Loihi platforms with the necessary optimizations. Section 5 provides the benchmarking results of the bursting network for the bio-mimetic neuron compared with the performance of Izhikevich neurons. Section 6 summarizes our contributions and possible future work.

2 Background

2.1 Bio-Mimetic Neuron Model

The basis for the neuron is derived from the circuit architecture in [10]. A capacitor and a resistor are connected in parallel with several current sources working at different timescales. These current sources are essentially expected to

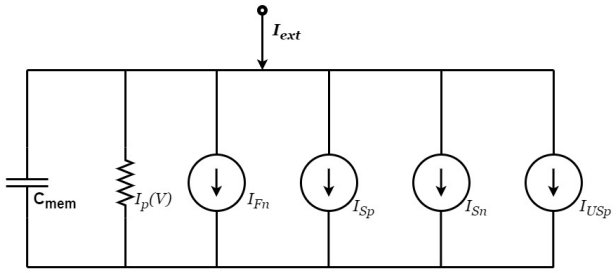

Figure 2. Circuit schematic of the bio-mimetic neuron model. The current sources shown are across multiple timescales with either a positive or negative feedback. Here, F_n stands for fast negative, S_p stands for slow positive, S_n stands for slow negative, US_p stands for ultra-slow positive.

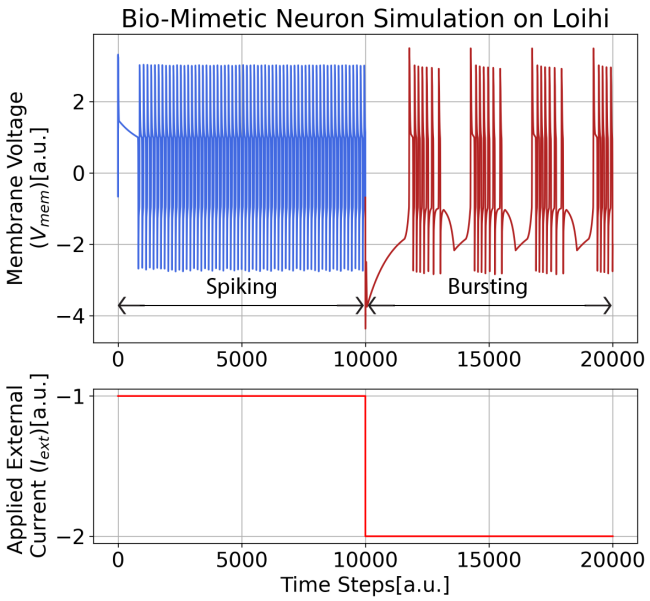

Figure 3. Loihi simulation result showcasing the effect of I_{ext} on intrinsic neuron behavior, transition from spiking to bursting modes. This work uses the burst mode of the neuron predominantly.

mimic the ion channels found in the biological neuron. These current sources are controlled through control voltages with IV characteristics given by:

$$C_{mem}\frac{dV_{mem}}{dt} = I_{ext} - I_p(V_{mem}) - \sum I_x^{\pm} \tag{1}$$

$$where, I_x^{\pm} = \alpha_x^{\pm}\tanh(V_{mem} - \delta_x^{\pm}) and I_p(V_{mem}) = V_{mem} \tag{2}$$

To set delay of different timescales,

$$\tau_x\frac{dV_x}{dt} = V_{mem} - V_x where, \tau_f << \tau_s << \tau_{us} \tag{3}$$

The four time scales chosen here are fast (negative feedback), slow (positive and negative feedback) and ultra-slow (positive feedback). This setup is more than enough to ensure spiking and bursting behavior for a range for applied external current. The only constraint here is to ensure that the time-scales are spaced at least 50 counts apart. This neuromimetic behavior is explained by the fourth order mixed feedback oscillation between the minimum and maximum voltage ranges. For the use case of bursting mode of operation, the important parameters are I_{ext} to set the neuron up for bursting and α_f^- to determine network convergence speed. α_{us}^+, β_{us}^+ are used to control both the inter-burst and intra-burst frequency of the network. Increasing these parameters results in higher inter-burst frequency and lower intra-burst frequency.

2.2 Robot Locomotion using Spiking CPG

Previous work [12] shows a non-linear bio-mimetic neuron being used to control robot locomotion based on spike inputs. This work maps a tightly coupled 4 neuron network onto a physical quadruped and generates different gait patterns by simply swapping out a weight matrix. Different gaits are generated to map different patterns generated by the CPG network. This work [12] shows that even though the applied external current is required to be in the appropriate range for smooth operation it is not mandatory to have these currents fixed to one value which allows for a considerable amount of jitter tolerance without critical failure. Another important feature of this work was the evolution of event-based control input to the motors, i.e. whenever a spike was detected in any one of the neurons, it would cause motor actuation on the limb corresponding to that neuron.

Even though spiking CPGs are easier to tune, simulate and control; one major flaw of using a spiking CPG for robot locomotion is the lack of intermediate states of motion. For instance, when a robot is being controlled though a spiking CPG it essentially gets one input from the controller to execute a large set of commands. Due to any unforeseen circumstance if the sensors detect an anomaly and want to reset to the mean position or handle the exception in some other way, a wait period would ensue till the limb finishes the assigned motion and returns to the mean position. This behavior is undesirable and takes a toll on the responsiveness and agility of the robot. However, if we design a bursting CPG based robot controller, then this would introduce intermediate states which act as checkpoints for quick changes in response to sensory inputs.

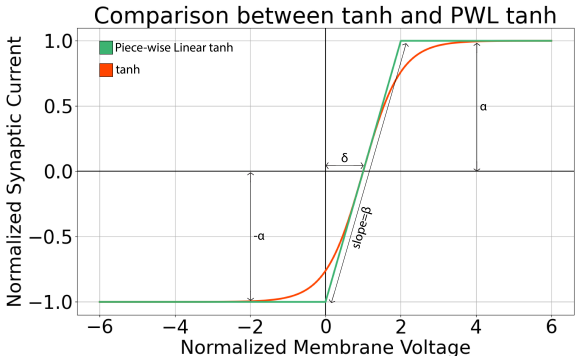

Figure 4. I-V characteristic curve showing the difference between the hyperbolic tangent and piece-wise linear hyperbolic tangent activation functions over a normalized scale.

3 Implementation

3.1 Piece-wise Linear Hyperbolic Tangent Activation

The math library available with Arduino (cmath) though highly efficient in terms of access times to the hyperbolic tangent function, takes up significant memory usage. Hence, we define a piece-wise linear function to replace the tanh function as closely as possible [9]. Here is the equation showcasing the difference between the two functions pictured in Figure 4:

$$I_x = \alpha\tanh(V_{mem} - \delta) \tag{4}$$

(4) can be written as

$$I_x = \begin{cases} -\alpha & \text{if } V_{mem} < -(\alpha/\beta) + \delta \\ \alpha & \text{if } V_{mem} > (\alpha/\beta) + \delta \\ \beta(V_{mem} - \delta) & \text{if } -(\alpha/\beta) + \delta < V_{mem} < (\alpha/\beta) + \delta \end{cases} \tag{5}$$

Here, β is the slope of the piece-wise linear hyperbolic tangent function which is an extra parameter as a result of linearization of the hyperbolic tangent function. The values for β are kept close to α to faithfully mimic tanh behavior.

3.2 Hardware-friendly Differential Equation Solver

Euler's method helps to solve the fourth order differential equation real time by numerical approximation of the state variables. This is way more efficient than trying to solve the differential equation directly or using other complex numerical evaluation methods such as Runge-Kutta. Note that this method does provide less accurate results when compared to other higher order methods but this loss in precision is inconsequential to the outcome of the system, and hence can be safely ignored. Another major advantage

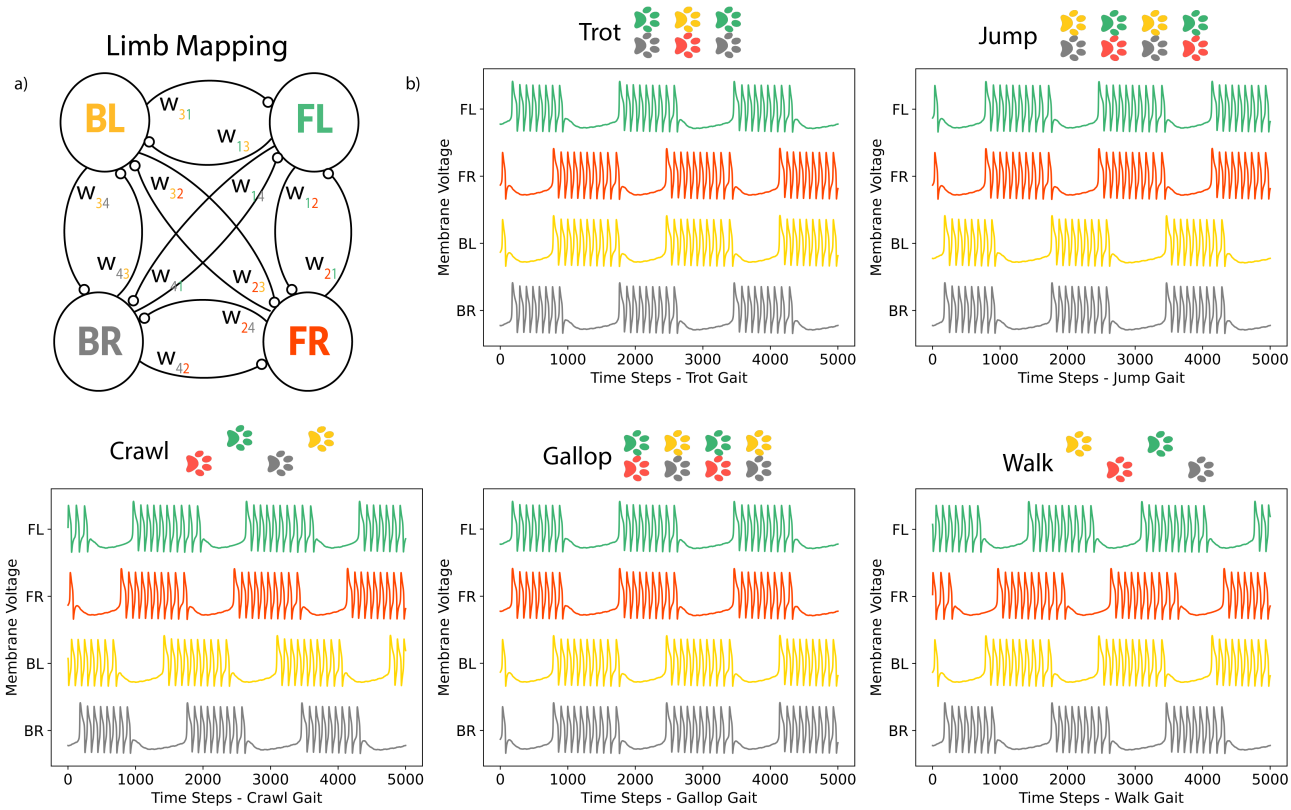

Figure 5. a) Network diagram showing the four neurons and the synaptic connections between them. b) Tuned voltage plots for all the different modes of locomotion with paws representing the achieved gait pattern for each mode.

of using Euler's method over the other evaluation methods is the low compute time per timestep, which results in more fluid and responsive robot locomotion.

3.3 Neuron Parameter Tuning

The primary requirement for the implementation was to choose the right region of operation for the neuron. The neuron has three regions of operation namely spiking, bursting and burst excitable. The process to set the neuron in the correct region is done by tuning α, δ, β and I_{ext}. The major parameter in determining the mode of operation is I_{ext}, and by setting an appropriate value, we can configure the neuron to be in the bursting region.

I_{ext} should be set correctly to determine the mode of operation of the neuron. Once that is fixed, one can tune the other parameters to pace up or slow down the given system. For instance, to increase the inter-burst frequency and to decrease the intra-burst frequency, one can tune α_{us} and β_{us} to high values. This ensures better pattern repeatability and faster convergence. Separation of timescales of operation is another important mode of controlling individual neuron dynamics. This separation between timescales should be chosen very carefully because if it is insufficient then the neuron spike and burst dynamics are not realized properly. Similarly, if its set too high then the neuron dynamics take a long time to take effect, which results in performance loss for a real time system like this one.

3.4 Network Modulation

Our network begins with four neurons, with each neuron mapped to a distinct limb (e.g. neuron one controls front left leg). We could also have assigned one neuron to each motor (one leg has a knee and shoulder motor each). However, that would have resulted in eight neurons to control four limbs, and we avoided this in order to keep the number of microcontroller instructions per cycle (IPC) on the lower end. The neurons are fully connected with plastic synapses and all the weight values are initially assigned to zero to ensure all the individual neurons are in the correct region of operation and are exhibiting expected behavior.

Next, in order to get anti-phase bursts between two neurons, start by initializing them with different external applied current[8]. This causes one neuron to converge faster and start its burst cycle earlier than the other ones. Looking at the gait patterns that are to be generated, [1] it can be seen that some patterns have two distinct phases while others have four distinct phases. The gaits with just two distinct phases essentially mimic a half oscillator, i.e. the weights

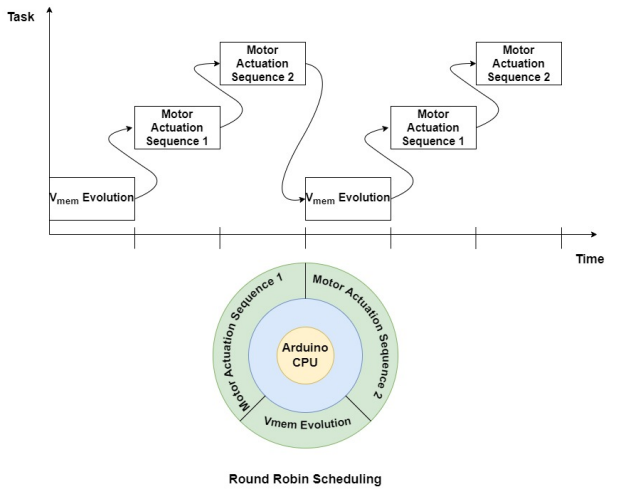

Figure 6. a) Gantt chart showing time allocation for all the tasks when two of the four motors have pending steps. These sequences will be skipped automatically if there are no motor actuations in queue. b) Shows the CPU time allocation division between the tasks for one cycle.

connecting neurons bursting in opposite phase are set as small inhibitory weights which cause the bursts to have a shift in phase. There is a little bit of overlap towards the end of a burst cycle but this is retained to get repeatability and pattern coherence. To expand the two-phase bursting to a four-neuron network, two neurons are initialized to burst early and the other two neurons get a small inhibitory connection from the early neurons in a set order. These late neurons are in turn connected with even smaller synaptic weights (about 80-90% of the original weight) back to the early neurons to make sure that the phase difference generated is locked. For gaits with 4 distinct phases, there is one early neuron which bursts first and has a small inhibitory connection to only the next neuron, which is expected to fire in the set pattern, and this neuron in turn inhibits the third neuron and so on. This sequence is continued till the final neuron inhibits the first neuron back, but the weight is slightly lower (about 80-90% of original weight) this time around. The synaptic weight matrix is modeled as shown in equation 6.

$$W_{i,j} = \begin{bmatrix} 0 & w_{1,2} & w_{1,3} & w_{1,4} \\ w_{2,1} & 0 & w_{2,3} & w_{2,4} \\ w_{3,1} & w_{3,2} & 0 & w_{3,4} \\ w_{4,1} & w_{4,2} & w_{4,3} & 0 \end{bmatrix} \quad (6)$$

This kind of a weight matrix setup causes the neurons to fire slightly out of phase from one another in a fixed order, resulting in a rear-ended overlap between the gait patterns as shown in Figure 5.

3.5 Round Robin Scheduling

Scheduling algorithms are used for multitasking, so that one can parallelize the tasks in a program. These algorithms minimize resource starvation, and make sure all the tasks receive a fair amount of CPU time. The scheduling algorithm that we have utilized here is the round-robin scheduling algorithm [11]. In this algorithm, all the tasks receive a fixed amount of CPU time, and we cycle through them. Once a task has been completed within a certain time segment, we terminate it, and give the remaining time to the other tasks.

For the purpose of this paper, this is highly useful. Since Arduino Uno (NyBoard micro-controller on Petoi Bittle is a modified version of Arduino Uno) does not support multi-threading, it hampers the performance of the robot. Each task has to wait until the previous task is terminated or completed, which is not ideal for the speed of the robot.

Hence, the use of the round robin scheduling algorithm gives the impression of multi-tasking, since doing portions of tasks extremely quickly (around 0.2 ms or so) gives enough time for other tasks as shown in Figure 6. Evolving the V_{mem} of the neurons is a relatively longer process, and it is mandatory to allocate a fair amount of CPU time to the motor commands as well as the evolution of V_{mem} for the 4-neuron network. This allows the robot to be paced appropriately, while being able to check for incoming bursts from the bursting CPG. Advantages of this algorithm is good average response time, since the waiting time is dependent on the number of tasks, as well as balanced throughput and fair allocation of the CPU.

4 Asynchronous CPG on Loihi

The biggest motivation behind proposing neuromorphic algorithms is to achieve bio-realistic behavior and solely showcasing this system on traditional sequential hardware severely limits the capability of the model. Hence, to demonstrate the actual advantage of using a neuromorphic algorithm over a traditional algorithm, we have used one of the most advanced tools available for neuromorphic computing, Intel's Loihi 2.0 platform [2]. This truly parallel asynchronous platform helped evaluate the model's capability and use case in a more realistic scenario rather than just molding the traditional systems to process neuromorphic data. The CPG was faithfully reproduced and run on an actual Loihi simulation platform, LAVA software framework.

Getting the CPG to run on Lava is essentially a two step process. The first step is to implement and simulate the isolated non-linear neuron as a Lava Loihi process to reproduce the expected behavior. This includes showcasing the different modes of operation of the neuron as shown in Figure 3 (spiking and bursting). The second step is to modulate a network by initializing multiple neurons and connecting them with plastic synapses to simulate a CPG. This is shown in Figure 8. Note that obtaining a phase difference and phase locking

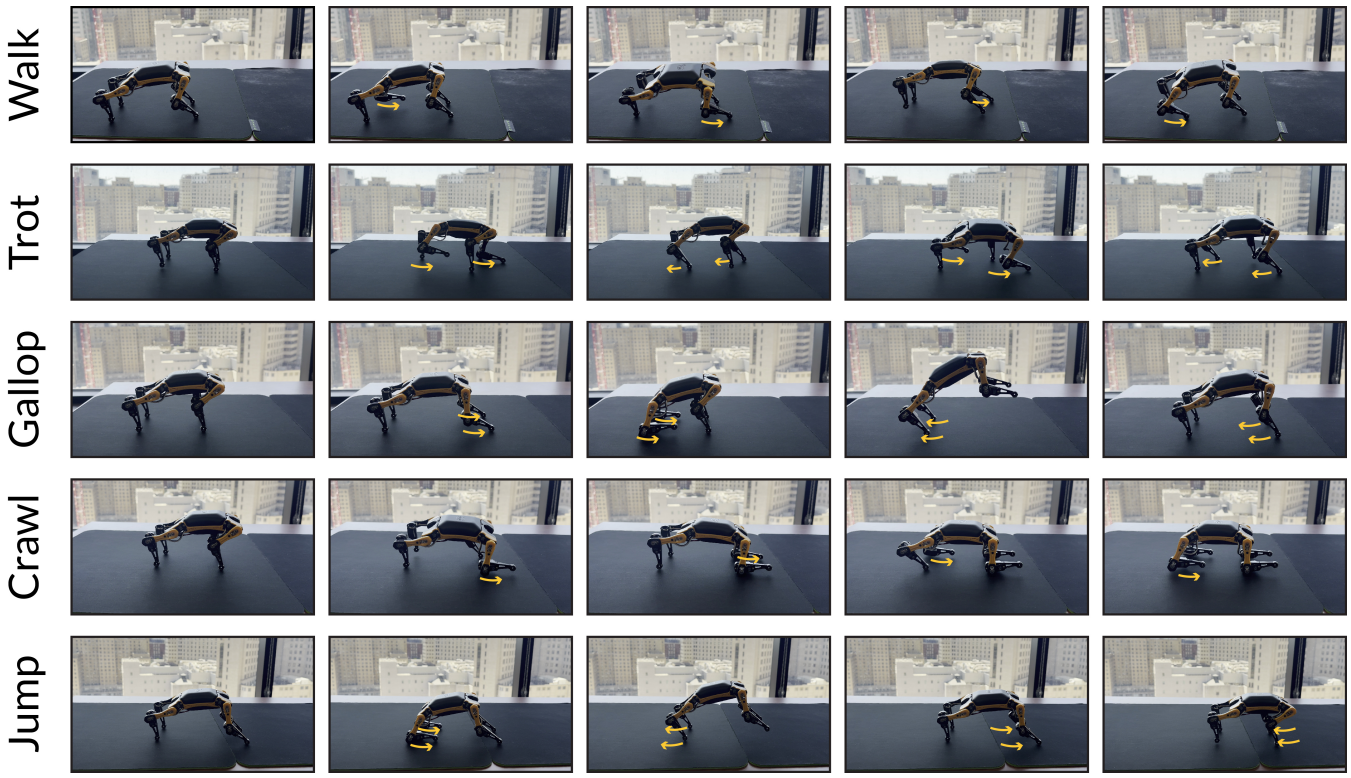

Figure 7. Different modes of locomotion working on the Petoi Bittle quadruped. The arrows indicate the motion of the limbs with respect to the previous frame. These photos are snapshots of the robot actually executing the gait patterns. Demo video can be found at https://youtu.be/9AKoM28ix68

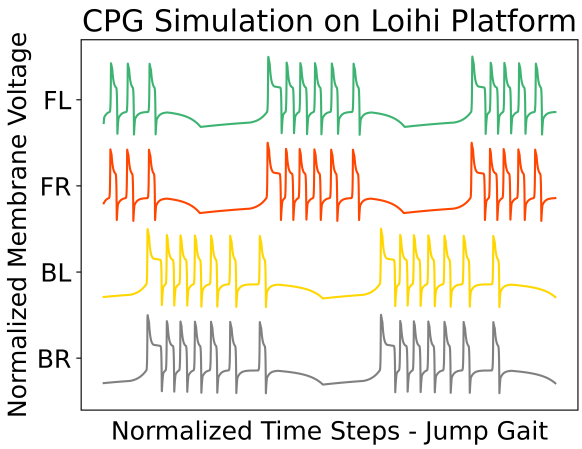

Figure 8. Tuned CPG network showing Jump gait pattern running on Loihi-Oheogulch. The neuron is defined as a custom process and the connections are set up exactly like the network shown in Figure 5.

between the bursts is critical to modulate the network of neurons to produce different patterns for long periods of time.

5 Results

The bursting CPG network is emulated real time on the Petoi Bittle's NyBoard and is shown in Figure 7. This was done using optimization techniques such as Round-Robin Scheduling, Euler's Approximation of a Differential Equation and piece-wise linearized hyperbolic tangent function. The number of spikes per burst, inter-burst frequency, intra-burst frequency was all tuned using I_{ext} and hyperparameters α, β and δ as discussed in [10]. The network architecture used is described with linearly programmed weight matrices for 5 different modes of locomotion in Figure 5. Video demonstration of the robot showing all the different patterns is available at https://youtu.be/9AKoM28ix68

5.1 Benchmarking

In order to evaluate the proposed work, it is important to compare it to the state of the art. The first step in this process is to benchmark the proposed non-linear bio-mimetic neuron against the widely applauded Izhikevich neuron operating in the chattering mode (similar to bursting mode in our neuron). One stark difference between these two models is the method to set the mode of operation of the neuron. In the proposed neuron, the mode of operation of the neuron is set by I_{ext} whereas the mode of operation in the Izhikevich neuron is

Method/ Motion Type		Memory Used % total memory on Arduino)	Latency per time-step (in ms)
Proposed Optimized Biomimetic Neuron	Jump	33%	2.92
	Walk	33%	2.92
	Trot	33%	2.92
	Gallop	33%	2.92
	Crawl	37%	2.92
	Isolated	22%	0.12
Izhikevich	Isolated	20%	0.14

Table 1. Memory and Latency Comparison between the Piece-wise linear Bio-mimetic neuron and Izhikevich neuron.

set by the parameters a, b, c and d and I_{ext} just acts like an on/off switch. The way to achieve modulation on the network level would be different in case of the Izhikevich neuron for this same reason. Izhikevich neurons would require a time controlled I_{ext} to lock in phase and provide bursting rhythms. A time controlled I_{ext} makes it difficult to swap modes of locomotion i.e. one set of time-series I_{ext} would cause the robot to be locked to one gait pattern.

Moreover, when testing an isolated Izhikevich neuron and the bio-mimetic neuron on Arduino it was found that there is a latency versus memory trade-off between the two. The bio-mimetic neuron consumed marginally higher memory (around 2% of on-board Arduino memory) while being reasonably faster (around 14%) in terms of latency in membrane voltage evolution as shown in Table 1. The latency was computed as an average over 1000 time-steps.

6 Conclusion

We have created a tightly coupled bio-mimetic neural network which generates fixed rhythmic burst patterns over a sustained period, that can be used to control the locomotion of a quadruped. These patterns are used to model different animal gaits accurately whilst trying to keep the memory and compute used to an absolute bare minimum. We believe this is the first time a bursting central pattern generator network is implemented as a locomotion control mechanism in any robot.

The underlying neuron chosen is highly non-linear and has been customized to help with performance on standard hardware. It uses approximations of non-linear functions wherever possible while still exhibiting neuro-modulatory behavior. This makes it both practical and an accurate artificial model mimicking biological neurons. We have attempted to showcase several locomotory gait implementations with minimal changes to the synaptic weights between the neurons to ensure rigidity at the network architecture level but still providing headroom for innovative applications. While comparing the bio-mimetic neuron against the Izhikevich neuron, it was found that even though the Izhikevich neuron is 2% more memory efficient, the bio-mimetic neuron does a lot better in terms of latency (around 14% faster). Lower latency is considerably more beneficial for a real-time application like robot locomotion in this case.

Our plan for the future is to use learning algorithms like Spike-Timing Dependent Plasticity (STDP) to achieve perfected gaits not only on quadrupeds but even on other different types of robots. The use of these training methods will potentially enable the use of bio-mimetic neurons to make a robot learn to walk, trot and jump by itself, paving the way for bio-mimetic robots with seamless sensorimotor control.

References

[1] Vicki L. Datt and Thomas F. Fletcher. 2012. Gait Foot-Fall Patterns. http://vanat.cvm.umn.edu/gaits/walk.html

[2] Mike Davies, Narayan Srinivasa, Tsung-Han Lin, Gautham Chinya, Yongqiang Cao, Sri Harsha Choday, Georgios Dimou, Prasad Joshi, Nabil Imam, Shweta Jain, Yuyun Liao, Chit-Kwan Lin, Andrew Lines, Ruokun Liu, Deepak Mathaikutty, Steven McCoy, Arnab Paul, Jonathan Tse, Guruguhanathan Venkataramanan, Yi-Hsin Weng, Andreas Wild, Yoonseok Yang, and Hong Wang. 2018. Loihi: A Neuromorphic Manycore Processor with On-Chip Learning. *IEEE Micro* 38, 1 (2018), 82–99.

[3] Pierre A Guertin. 2009. The mammalian central pattern generator for locomotion. *Brain research reviews* 62, 1 (2009), 45–56.

[4] CHRISTIAN HUBICKI. 2020. Robots That Walk: What the Challenge of Locomotion Says About NextGeneration Manufacturing. In *Frontiers of Engineering: Reports on Leading-Edge Engineering from the 2019 Symposium*. National Academies Press.

[5] Christian Hubicki, Andy Abate, Patrick Clary, Siavash Rezazadeh, Mikhail Jones, Andrew Peekema, Johnathan Van Why, Ryan Domres, Albert Wu, William Martin, et al. 2018. Walking and running with passive compliance: Lessons from engineering: A live demonstration of the atrias biped. *IEEE Robotics & Automation Magazine* 25, 3 (2018), 23–39.

[6] Auke Jan Ijspeert. 2008. Central pattern generators for locomotion control in animals and robots: a review. *Neural networks* 21, 4 (2008), 642–653.

[7] Ashwin Sanjay Lele, Yan Fang, Justin Ting, and Arijit Raychowdhury. 2020. Learning to walk: Spike based reinforcement learning for hexapod robot central pattern generation. In *2020 2nd IEEE International Conference on Artificial Intelligence Circuits and Systems (AICAS)*. IEEE, 208–212.

[8] Álvaro Lozano, Marcos Rodríguez, and Roberto Barrio. 2016. Control strategies of 3-cell Central Pattern Generator via global stimuli. *Scientific Reports* 6, 1 (29 Mar 2016), 23622.

[9] Ashkan Namin, Karl Leboeuf, Huapeng Wu, and Majid Ahmadi. 2009. Artificial neural networks activation function HDL coder. *Proceedings of 2009 IEEE International Conference on Electro/Information Technology, EIT 2009*, 389 – 392.

[10] Luka Ribar and Rodolphe Sepulchre. 2019. Neuromodulation of Neuromorphic Circuits. *IEEE Transactions on Circuits and Systems I: Regular Papers* 66, 8 (aug 2019), 3028–3040.

[11] Sakshi, Chetan Sharma, Shamneesh Sharma, Sandeep Kautish, Shami A. M. Alsallami, E.M. Khalil, and Ali Wagdy Mohamed. 2022. A new median-average round Robin scheduling algorithm: An optimal approach for reducing turnaround and waiting time. *Alexandria Engineering Journal* 61, 12 (2022), 10527–10538.

[12] Vijay Vivekanand, Shahin Hashemkhani, Shanmuga Venkatachalam, and Rajkumar Kubendran. 2023. Robot Locomotion Control using Central Pattern Generator with Non-linear Bio-mimetic Neurons. *International Conference on Automation, Robotics and Applications* 9 (Feb 2023), 102–106. https://doi.org/10.1109/ICARA56516.2023.10125666