



UWL REPOSITORY

repository.uwl.ac.uk

Self-learning neuromorphic robot based on reward-driven Spiking Neural Network

Russo, Nicola, Madsen, Thomas ORCID logoORCID: <https://orcid.org/0000-0001-9354-0935> and Nikolic, Konstantin ORCID logoORCID: <https://orcid.org/0000-0002-6551-2977> (2025) Self-learning neuromorphic robot based on reward-driven Spiking Neural Network. In: IEEE International Symposium on Circuits and Systems (ISCAS), 25-28 May 2025, London, UK.

<http://dx.doi.org/10.1109/ISCAS56072.2025.11044049>

This is the Accepted Version of the final output.

UWL repository link: <https://repository.uwl.ac.uk/id/eprint/13919/>

Alternative formats: If you require this document in an alternative format, please contact: open.research@uwl.ac.uk

Copyright:

Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

Take down policy: If you believe that this document breaches copyright, please contact us at open.research@uwl.ac.uk providing details, and we will remove access to the work immediately and investigate your claim.

Rights Retention Statement:

Self-Learning Neuromorphic Robot Based on Reward-Driven Spiking Neural Network

Nicola Russo, Thomas Madsen, and Konstantin Nikolic

School of Computing and Engineering, University of West London, London W5 5RF, UK

Nicola.Russo@uwl.ac.uk; Thomas.Madsen@uwl.ac.uk; Konstantin.Nikolic@uwl.ac.uk

Abstract—While there are adequate tools available to simulate Spiking Neural Networks (e.g. Brian2, snnTorch), as well as the tools for simulating robots and their environments, there remains a need for integrated tools that enable researchers to jointly simulate realistic brain models, robots, and sensory-rich environments. This work introduces a comprehensive neuromorphic robotic system, which combines neuromorphic computing with neuromorphic (and conventional) sensory and motor devices. We emulate the neuromorphic computing on a conventional low-power CPU, specifically a Virtual Machine on a Raspberry Pi 5, integrating Python and specialised packages for real-time Spiking Neural Networks (SNN) simulations. We achieve: (i) a cost-effective alternative to dedicated neuromorphic hardware, (ii) built-in GPIO and USB ports for seamless sensor and motor interfacing. We have built a demonstrator system: a robotic goalkeeper, using a DVS camera, a digital servo motor, and a touch sensor for a reward signal. The SNN uses a combination of unsupervised and supervised (reinforcement) learning. The system off-line and on-line learning was demonstrated, and some performance metrics reported.

Index Terms—robotics, spiking neural networks, neuromorphic computing, neuromorphic hardware, low-power systems.

I. INTRODUCTION

Neuromorphic hardware has shown significant potential in bioinspired robotics [1]–[3], offering low-power and low-latency solutions for tasks like object tracking [4], object interception [5], obstacle avoidance [6], [7], and emulation of biological behavior [8], [9]. Conventional platforms like microcontrollers [10], [11], FPGAs [7], and cloud computing are still commonly used, but dedicated neuromorphic platforms such as SpiNNaker [12], Loihi [13], and TrueNorth [14] could offer performance advantages, though their application in robotics is still developing [15].

Neuromorphic sensory systems, such as silicon retina (Dynamic Vision Sensors (DVSs)) or silicon cochlear [16], also play a crucial role in enabling efficient robotic operations. They complement traditional sensory technologies like ultrasonic and tactile sensors [17], providing enhanced capabilities for real-time applications.

Spiking Neural Networks (SNNs) are at the heart of many neuromorphic systems. Unlike traditional ANNs, SNNs communicate via asynchronous spikes between neurons, replicating the temporal dynamics of biological neural networks [18]. This spike-based paradigm is particularly effective for dynamic, real-time tasks, where temporal information is critical for decision-making [1]. The challenge lies in training SNNs on robotic platforms, which has led to the development of

learning algorithms such as Spike-Timing-Dependent Plasticity (STDP) [19], [20], Reinforcement STDP (R-STDP) [21], and Dopamine-Modulated STDP [6]. The backpropagation algorithms like SpikeProp [22] were adapted for SNNs recently, however they remain challenging to implement in robotics.

Neuromorphic systems offer two key advantages: creation of low-power intelligent systems inspired by brain function, and the facilitation of studying biological system through simulation. However, access to dedicated neuromorphic hardware is still limited. Platforms like SpiNNaker, Loihi, and TrueNorth are not widely available, and their integration with external devices is often complex. Deployment of the NVIDIA Isaac AI robot development platform with NVIDIA Jetson might also be unaffordable [23]. This work addresses these limitations by exploring the use of conventional low-power hardware, such as Raspberry Pi 5 [24], to implement neuromorphic robotic systems using software simulation.

Here we describe a neuromorphic robotic system based on Python simulation platform running on a Raspberry Pi 5, combined with peripheral devices (DVS, and touch sensor), and actuators (a servo motor). The novelty of this approach lies in combining the widely accessible Python libraries for SNN simulation, such as Brian2 [25] and snnTorch [26], with the Raspberry Pi platform for executing the code and communicating with external devices. This setup demonstrates the feasibility of using conventional hardware for neuromorphic robotics, offering an affordable alternative to specialized neuromorphic platforms. Importantly, we demonstrate the ability of simulation platforms such as Brian2 and snnTorch to handle real-time inputs effectively, making them suitable for applications which require processing of incoming stream of data such as robotics.

While neuromorphic platforms often employ specialised operating systems optimised for partitioning and mapping SNNs [27], we take a different approach by utilising a conventional operating system running on low-power processors. This method allows the use of standard programming languages (e.g. Python), and offers an accessible solution for researchers.

II. MATERIALS AND METHODS

A. A Self-Learning Neuromorphic Robotic System.

Our self-learning robotic system is designed to observe an incoming object and intercept it using a combination of visual input from an event-based camera DVS128 [28] and actuator output to control a servo motor (Futaba S9257), Figure 1.

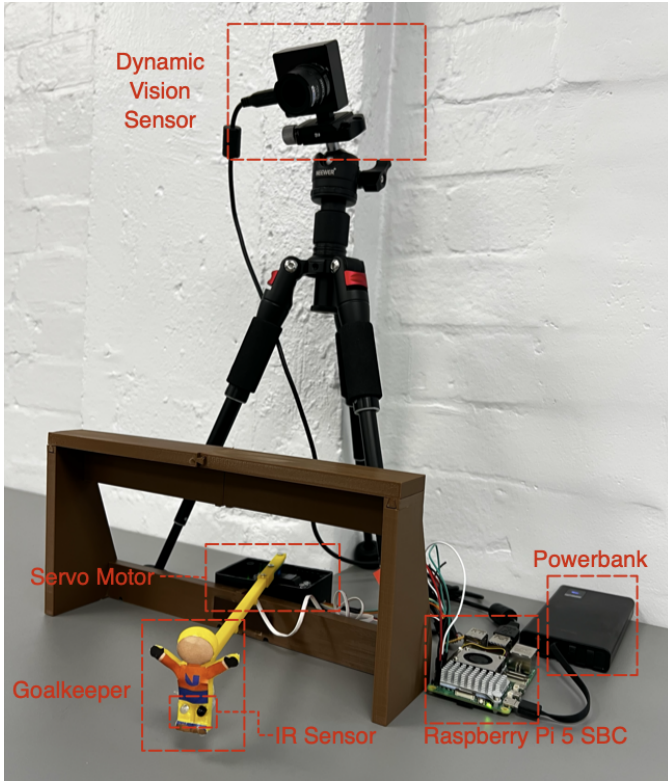


Fig. 1: System configuration: Raspberry Pi 5 powered by USB-C power bank (12,500 mAh capacity), the DVS camera, the servo motor with the goalkeeper, and the IR touch sensor.

The information about whether the decision where to position the goalkeeper was correct is reported back to the SNN by using a touch sensor. In our case the sensor was realised as a combination of an IR-LED and a photodiode sensor. It provides touch feedback, which is used in the SNN model for reinforcement learning.

All these components are integrated with a Raspberry Pi 5 via USB and GPIO interfaces. The Raspberry Pi 5 runs an SNN model on a Virtual Machine, which processes real-time input from the sensors to predict the goalkeeper’s position.

B. The Hardware and Software

The core of our system is based on a low-power Single Board Computer (SBC), specifically the Raspberry Pi 5, which serves as the neuromorphic processing unit. The SBC runs a Virtual Machine (VM) that simulates Spiking Neural Networks (SNNs) using the Brian2 or snnTorch frameworks. The system design leverages the SBC’s physical connections to interface with external sensors and devices, making it a flexible and modular platform, Figure 2.

We have introduced the concept of transforming a conventional low-power CPU into neuromorphic hardware due to the need for low-cost alternatives to specialised neuromorphic chips [29]. By using a Raspberry Pi 5 with a Broadcom BCM2712 quad-core Arm Cortex A76 processor and 8 GB of RAM, we can simulate neuromorphic computations efficiently.

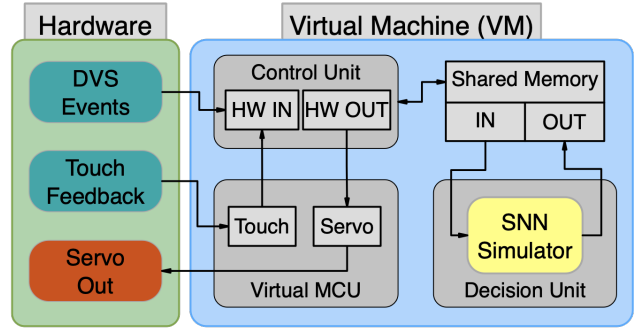


Fig. 2: The Neuromorphic Robotic system data flow. The Control Unit (CU) handles all inputs and outputs, and communicates with other units via a Shared I/O Memory. The Decision Unit runs the SNN simulation (yellow box). A Virtual MicroController Unit (MCU) drives the digital Servo motor and handles the Touch sensor signals.

This approach also benefits from the use of standard interfaces provided by the SBC, such as USB for sensor connections and General Purpose Input/Output (GPIO) ports for actuators. The external devices are managed by native drivers installed on the VM, ensuring smooth data transfer between hardware components and the SNN model.

The data flow begins with the DVS128 camera, connected via USB, which sends Address-Event Representation (AER) packets to the Raspberry Pi, Figure 2. These packets are handled by the *libcaer* driver, which also manages aspects such as noise reduction and packet batching, ensuring that only relevant events are processed by the system. The raw data is received by the Control Unit for preprocessing and converting into a rate-based representation. The preprocessing is essential to reduce the number of spikes sent to the SNN while preserving critical information about the scene. The data is then stored in shared memory, allowing the Decision Unit (which runs the SNN simulation), to access the data in real-time. The introduction of Shared memory is a key innovation for the real-time operation.

The Decision Unit manages the SNN simulation and handles communication with the Virtual Microcontroller Unit (MCU), which controls the servo motor. This Virtual MCU is responsible for receiving feedback from the infrared (IR) sensor. Communication between the Decision Unit and the Virtual MCU is handled through socket connections, enabling real-time feedback and control.

A key challenge in real-time SNN simulation is synchronising the simulation time with real time to ensure precise predictions and seamless actuation. To answer this, we adjust the execution by integrating time deltas between simulation steps, enabling the system to account for any delays. The input from the DVS camera is converted into spike frequencies and directed to SNN Input Layer. The simulation outputs are then promptly processed and transmitted to the actuators through shared memory, ensuring the system responsive to input.

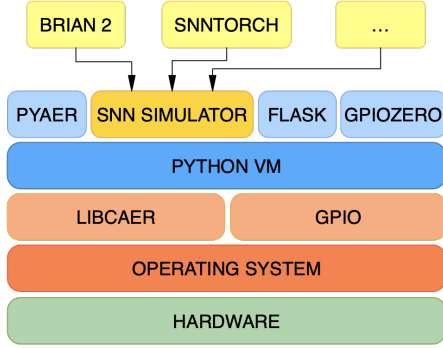


Fig. 3: System stack illustrating abstraction levels. Level 1: hardware (SBC, DVS, sensors, actuator). Level 2: OS (Linux) runs services and virtual units. Level 3: drivers (libcaer, gpio) interface with hardware. Level 4: Virtual Machine runs units, supported by PyAer for video input, SNN simulators (Brian2, snnTorch) for prediction, gpiozero/Flask for positioning.

The Virtual MCU handles the servo motor’s pulse-width modulation (PWM) signals using the *gpiozero* [30] library, which simplifies GPIO control on the Raspberry Pi.

In terms of software architecture, our system is built on a four-layer stack (Figure 3): (L1) consists of the hardware components, including the Raspberry Pi 5, DVS128 camera, IR sensor and servo motor, (L2) the operating system (Linux), which manages the drivers and libraries required to interface with the hardware, (L3) the device drivers and communication protocols, and (L4) the Python VM, which hosts the neuromorphic computation and robotic control logic. Here, the PyAer [31] library reads the DVS input (from libcaer driver [32]), and the *gpiozero* library, together with the *Flask* web service [33], manage the actuator communication. The SNN simulator could be realised in Brian2, snnTorch or any other SNN simulation framework. This layered approach ensures modularity, making it easier to adapt the system to new hardware components or modify the neuromorphic model without affecting the lower layers.

C. The SNN model

We use the Brian2 and snnTorch frameworks, which are well-established tools for simulating SNNs, but implemented here to operate on incoming stream of data in real-time. Since real-time prediction is critical for robotic applications, we synchronise the simulation time with the real-time.

The SNN topology consists of three layers: Input (I), Hidden (H) and Output (O), and an additional neuron for the reward (R), Figure 4, with all-to-all connectivity between the layers and lateral inhibition within the H and O layers. We have implemented a two-phase learning scheme: (I) a winner-takes-all unsupervised learning with STDP for I-H synapses to automatically classify trajectories in the hidden neurons, and (II) supervised reward based learning (for H-O synapses) to map the final position to the specific trajectories. The second phase is based on a Delayed-STDP, Figure 5, that shifts in time

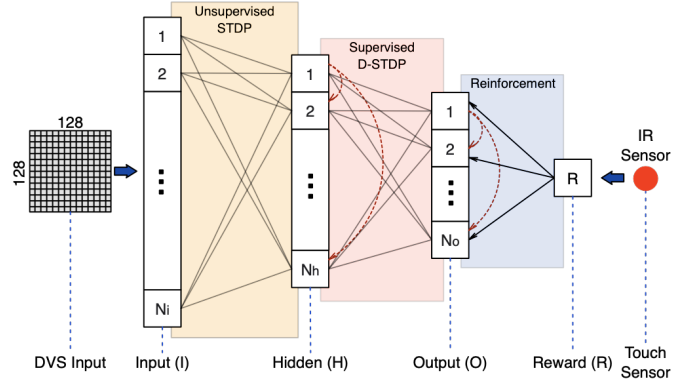


Fig. 4: The SNN topology: Input (I), Hidden (H) and Output (O) layers. The DVS input is first preprocessed (downsizing, noise filtering, spike encoding) and then fed into the Input. The touch sensor signal is fed into the Reward (R) neuron. Synapses $I \rightarrow H$ and $H \rightarrow O$ are plastic (until fixed after the training), whereas $R \rightarrow O$ is always fixed.

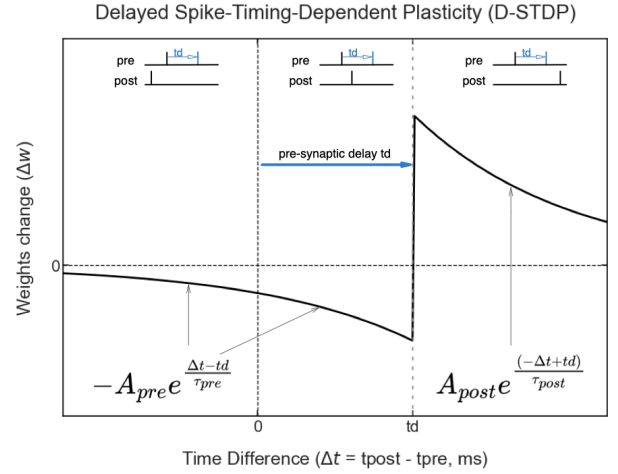


Fig. 5: Delayed-STDP: the synaptic weight change vs time difference between the post and pre synaptic spikes. The time shift (delay) is t_d .

the pre-synaptic spike trace. In this way we allow the output neurons to spike on time for the SNN to make a decision, but to delay the STDP process to take into account the reward signal. This approach ensures that the wrong classification for the goalkeeper’s position is not reinforced through the STDP, since every selected path is first depressed, and then reinforced if the reward signal is triggered. Although there are more well established algorithms for reinforcement learning [34], they are much more complicated and power demanding for online robotics learning.

III. RESULTS

Our primary success metric was to demonstrate the operationability of our Self-Learning Robotic platform. Then we have evaluated the performance by estimating accuracy, latency, and power usage for real-time applications.

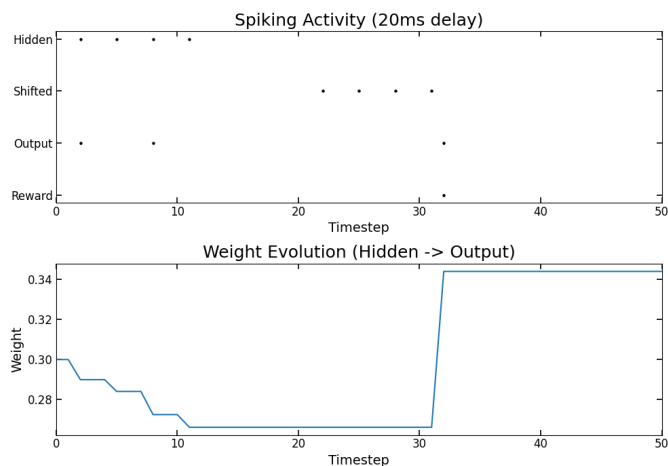


Fig. 6: A demonstration of the supervised learning phase. (top) A randomly chosen Hidden (H_i) and Output neuron (O_j) spikes are shown, together with a Reward spike (and shifted H_i spikes). (bottom) The change of the weight for the synapse $H_i \rightarrow O_j$.

A. Demonstration of the Off-line and On-line Learning

We first demonstrate the robot functionality. The off-line training has been done on the visual data created by simulations of balls moving on the screen. In this case the reward signal is generated also by the code when the visual classification by the SNN is correct. Due to limited space we only touch on the supervised learning phase here. Figure 6 shows the weight change for a selected synapse linking a spiking hidden neuron (H_i) to an output one (O_j). When O_j spikes, lateral inhibition strongly hyperpolarises the rest of O neurons. Then if R neuron spikes (meaning successful guess), it creates excitatory post-synaptic potential in all O neurons, but only O_j will spike again (or have a short burst). This will strengthen the synapses to those hidden neurons which fired just before O_j . But note that the initial spike of O_j will depress all these synapses, because of delayed STDP, but the overall effect will be the synaptic weight increase, Figure 6. If there is no reward signal, the synapses will remain depressed.

Then we train and test the robotic system using real balls and the touch sensor. For the SNN we have: the number of input neurons $N_i = 256$, hidden $N_h = 8$, and output $N_o = 8$ (corresponding to 8 positions for the goalkeeper), for both off-line and on-line training cases.

B. Accuracy, Resource Consumption, Latency

After the training was completed, the synaptic weights were fixed, and the system’s accuracy was evaluated by counting successful ball blocks over 100 random trials.

We have achieved an overall accuracy of 80%, similar to tests in a controlled offline environment using simulated ball movements. Results showed consistency between real-world and simulated scenarios, further validating the system’s predictive capability.

The total latency includes: receiving the input signal, pre-processing it, delivering it to the SNN, then SNN operation to predict the goalkeeper’s position, communication with the servo and servo motor movement. The DVS camera is generating events every $1 \mu\text{s}$, and the system preprocesses the input in 1 ms batches. The SNN operates in 50 ms batches. With minimum delays of 1 ms for input and output, the overall computation latency is 3 ms. Additional delays come from arm positioning (75 ms for 60°) and HTTP communication, resulting in a total latency of 154 ms, optimized to 100 ms with position reset.

The system was tested by using a 5 V, 12,500 mAh battery. During execution, the CPU operated at 1.2 V, with average current of 6 A (range was 4–7 A). The usage was consistently over 80%, and the temperature stabilised at $60\text{--}65^\circ\text{C}$. The whole system operated at 5 V, and consumed on average 3.8 A, leading to a total battery consumption of approximately 20 W.

IV. DISCUSSION AND CONCLUSION

In this work we have developed a neuromorphic robotic system capable of Self-Learning, by integrating a reward signal into the Spiking Neural Network. Importantly, the robot processing unit is built on a Raspberry Pi 5, which is a very affordable solution, offering easy interfacing with peripheral hardware (both neuromorphic and conventional), but still very efficient in terms of low-power consumption and speed of operation. The system achieved real-time performance with synchronisation mechanisms and maintained power consumption around 20 W, comparable to similar platforms. It demonstrated 88% offline and 80% online accuracy, improving on previous work. A novel SNN solution, which combines unsupervised and supervised learning with reinforcement learning, has been successfully developed and demonstrated.

From a software perspective, the system utilises the standard Python libraries (such as Brian2 and snnTorch) for real-time SNN simulation, offering full control over the neural network model and preventing spike loss, a common issue in hardware-based neuromorphic systems. SNNs are particularly suited to event-based inputs, unlike frame-based CNN models, which are less efficient in real-time scenarios.

Our system also offers scalability. Hardware scalability depends on the number of available ports, while the SNN’s scalability is highly flexible due to its software-based simulation, unlike neuromorphic hardware where network size is constrained by physical resources. Furthermore, the system can be expanded into a cluster of SBCs to boost computational power when needed.

Despite these advantages, there are limitations. Real-time performance is impacted by the complexity of the SNN model; finer time-steps, such as $dt = 0.5 \text{ ms}$, improve detail but may slow down the simulation. Additionally, an increasing number of neurons and synapses increases computational demands.

ACKNOWLEDGEMENTS

NR thanks the University of West London Vice Chancellor’s PhD Scholarship.

REFERENCES

- [1] Bing, Z.; Meschede, C.; Röhrbein, F.; Huang, K.; Knoll, A.C. A Survey of Robotics Control Based on Learning-Inspired Spiking Neural Networks. *ront. Neurorobot.* 2018, 12, 35.
- [2] Ivanov, D.; Chezhegov, A.; Kiselev, M.; Grunin, A.; Larionov, D. Neuromorphic artificial intelligence systems. *Front. Neurosci.* 2022, 16, 959626.
- [3] Basu, A.; Acharya, J.; Karnik, T.; Liu, H.; Li, H.; Seo, J.-S.; Son, C. Low-Power, Adaptive Neuromorphic Systems: Recent Progress and Future Directions. *IEEE J. Emerg. Sel. Top. Circuits Syst.* 2018, 8, 6.
- [4] Lobov, S.A.; Mikhaylov, A.N.; Shamshin, M.; Makarov, V.A.; Kazantsev, V.B. Spatial Properties of STDP in a Self-Learning Spiking Neural Network Enable Controlling a Mobile Robot. *Front. Neurosci.* 2020, 14, 88.
- [5] Russo, N.; Huang, H.; Nikolic, K. Live Demonstration: Neuromorphic Robot Goalie Controlled by Spiking Neural Network. In Proceedings of the 2022 IEEE Biomedical Circuits and Systems Conference (BioCAS), Taipei, Taiwan, 13–15 October 2022; pp. 249–249.
- [6] Liu, J.; Lu, H.; Luo, Y.; Yang, S. Spiking neural network-based multi-task autonomous learning for mobile robots. *Eng. Appl. Artif. Intell.* 2021, 104, 104362.
- [7] Liu, J.; Hua, Y.; Yang, R.; Luo, Y.; Lu, H.; Wang, Y.; Yang, S.; Ding, X. Bio-Inspired Autonomous Learning Algorithm with Application to Mobile Robot Obstacle Avoidance. *Front. Neurosci.* 2022, 16, 905596.
- [8] Clawson, T.S.; Ferrari, S.; Fuller, S.B.; Wood, R.J. Spiking Neural Network (SNN) Control of a Flapping Insect-Scale Robot. In Proceedings of the 2016 IEEE 55th Conference on Decision and Control (CDC), Las Vegas, NV, USA, 12–14 December 2016; pp. 3381–3388.
- [9] Deng, X.; Weirich, S.; Katzschnmann, R.; Delbruck, T. A Rapid and Robust Tendon-Driven Robotic Hand for Human-Robot Interactions Playing Rock-Paper-Scissors. In Proceedings of the IEEE RO-MAN 2024, Pasadena, CA, USA, 26–30, August 2024.
- [10] Cheng, R.; Mirza, K.B.; Nikolic, K. Neuromorphic robotic platform with visual input, processor and actuator, based on spiking neural networks. *Appl. Syst. Innov.* 2020, 3, 28.
- [11] Russo, N.; Huang, H.; Donati, E.; Madsen, T.; Nikolic, K. An Interface Platform for Robotic Neuromorphic Systems. *Chips* 2023, 2, 20–30.
- [12] Furber, S.B.; Galluppi, F.; Temple, S.; Plana, L.A. The SpiNNaker Project. *Proc. IEEE* 2014, 102, 652–665.
- [13] Davies, M.; Srinivasa, N.; Lin, T.-H.; Chinya, G.; Cao, Y.; Choday, S.H.; Dimou, G.; Joshi, P.; Imam, N.; Jain, S.; et al. Loihi: A Neuromorphic Manycore Processor with On-Chip Learning. *IEEE Micro* 2018, 38, 82–99.
- [14] Akopyan, F.; Sawada, J.; Cassidy, A.; Alvarez-Icaza, R.; Arthur, J.; Merolla, P.; Imam, N.; Nakamura, Y.; Datta, P.; Nam, G.-J.; et al. TrueNorth: Design and Tool Flow of a 65 mW 1 Million Neuron Programmable Neurosynaptic Chip. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* 2015, 34, 1537–1557.
- [15] Natale, L.; Bartolozzi, C.; Nori, F.; Sandini, G.; Metta, G. iCub. *arXiv* 2021, *arXiv:2105.02313*.
- [16] Liu, S.-C.; Delbruck, T. Neuromorphic sensory systems. *Curr. Opin. Neurobiol.* 2010, 20, 1–8.
- [17] Chou, T.-S.; Bucci, L.D.; Krichmar, J.L. Learning touch preferences with a tactile robot using dopamine modulated stdp in a model of insular cortex. *Front. Neurorobot.* 2015, 9, 6.
- [18] Maass, W. Networks of Spiking Neurons: The Third Generation of Neural Network Models. *Neural Netw.* 1997, 10, 1659–1671.
- [19] Diehl, P.; Cook, M. Unsupervised Learning of Digit Recognition Using Spike-Timing-Dependent Plasticity. *Front. Comput. Neurosci.* 2015, 9, 99.
- [20] Linares-Barranco, B.; Serrano-Gotarredona, T.; Camuñas-Mesa, L.A.; Perez-Carrasco, J.A.; Zamarreño-Ramos, C.; Masquelier, T. On Spike-Timing-Dependent-Plasticity, Memristive Devices, and Building a Self-Learning Visual Cortex. *Front. Neurosci.* 2011, 5, 26.
- [21] Juárez-Lora, A.; Ponce-Ponce, V.H.; Sossa, H.; Rubio-Espino, E. R-STDP Spiking Neural Network Architecture for Motion Control on a Changing Friction Joint Robotic Arm. *Front. Neurorobot.* 2022, 16, 904017.
- [22] Bohté, S.M.; Kok, J.N.; Poutré, H.L. SpikeProp: Backpropagation for Networks of Spiking Neurons. In: ESANN 2000 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, 26-28 April 2000, Bruges (Belgium).
- [23] Ren, P. et al., "Infinite World: A Unified Scalable Simulation Framework for General Visual-Language Robot Interaction, *arXiv:2412.05789* 2024
- [24] Raspberry Pi 5 Single Board Computer. Available online: <https://www.raspberrypi.com/5> (accessed on 3 February 2024).
- [25] Stimberg, M.; Brette, R.; Goodman, D.F. Brian 2, an Intuitive and Efficient Neural Simulator. *eLife* 2019, 8, e47314.
- [26] Eshraghian, J. K. et al., "Training Spiking Neural Networks Using Lessons From Deep Learning," in Proceedings of the IEEE, vol. 111, no. 9, pp. 1016-1054, Sept. 2023, doi: 10.1109/JPROC.2023.3308088.
- [27] Xue, J.; Xie, L.; Chen, F.; Wu, L.; Tian, Q.; Zhou, Y.; Ying, R.; Liu, P. EdgeMap: An Optimized Mapping Toolchain for Spiking Neural Network in Edge Computing. *Sensors* 2023, 23, 6548.
- [28] Lichtsteiner, P.; Posch, C.; Delbruck, T. A 128 × 128 120 dB 15 Latency Asynchronous Temporal Contrast Vision Sensor. *IEEE J. Solid-State Circuits* 2008, 43, 566–576.
- [29] Russo, N.; Madsen, T.; Nikolic, K. An Implementation of Communication, Computing and Control Tasks for Neuromorphic Robotics on Conventional Low-Power CPU Hardware. *Electronics* 2024, 13, 3448.
- [30] Raspberry Pi Foundation. GPIO Zero Documentation. Available online: <https://gpiozero.readthedocs.io> (accessed on 19 March 2024).
- [31] Yue, D. PyAer Documentation. GitHub. Available online: <https://github.com/duguyue100/pyaer> (accessed on 14 March 2024).
- [32] iniVation AG. Libcaer Documentation. Available online: <https://libcaer.inivation.com> (accessed on 5 February 2024).
- [33] Grinberg, M. Flask Web Development: Developing Web Applications with Python; O'Reilly Media: Sebastopol, CA, USA, 2018.
- [34] Kormushev, P.; Calinon, S.; Caldwell, D. G. Reinforcement Learning in Robotics: Applications and Real-World Challenges, *Robotics* 2013, 2(3), 122–148.