chips

MDPI

Article

An Interface Platform for Robotic Neuromorphic Systems

Nicola Russo [1,*], Haochun Huang [2], Eugenio Donati [1], Thomas Madsen [1] and Konstantin Nikolic [1,2,*]

[1] School of Computing and Engineering, University of West London, London W5 5RF, UK
[2] Institute of Biomedical Engineering, Imperial College London, London SW7 2AZ, UK
* Correspondence: 21485661@student.uwl.ac.uk or russo.nik@gmail.com (N.R.); konstantin.nikolic@uwl.ac.uk (K.N.)

Abstract: Neuromorphic computing is promising to become a future standard in low-power AI applications. The integration between new neuromorphic hardware and traditional microcontrollers is an open challenge. In this paper, we present an interface board and a communication protocol that allows communication between different devices, using a microcontroller unit (Arduino Due) in the middle. Our compact printed circuit board (PCB) links different devices as a whole system and provides a power supply for the entire system using batteries as the power supply. Concretely, we have connected a Dynamic Vision Sensor (DVS128), SpiNNaker board and a servo motor, creating a platform for a neuromorphic robotic system controlled by a Spiking Neural Network, which is demonstrated on the task of intercepting incoming objects. The data rate of the implemented interface board is 24.64 k symbols/s and the latency for generating commands is about 11ms. The complete system is run only by batteries, making it very suitable for robotic applications.

Keywords: spiking neural network; neuromorphic computing; SpiNNaker; neuromorphic interface board

Citation: Russo, N.; Huang, H.; Donati E.; Madsen, T.; Nikolic, K. An Interface Platform for Robotic Neuromorphic Systems. *Chips* **2023**, 2, 20–30. https://doi.org/10.3390/chips2010002

Academic Editor: Gaetano Palumbo

Received: 21 October 2022
Revised: 16 December 2022
Accepted: 20 January 2023
Published: 1 February 2023

1. Introduction

Spiking Neural Networks (SNN) run on neuromorphic hardware are an excellent platform to develop new robotic systems, and at the same time get better insight into the operation of biological neural systems [1]. The strength of SNNs lies in the neuron structure, where the biological membrane potential is computed resolving a differential equation, placing this model closer to the biological neuron. Compared to the second generation of Artificial Neural Networks (ANN), Spiking Neural Networks receive discrete signals (spikes), greatly simplifying the computation complexity. A presynaptic neuron fires when the membrane potential exceeds a potential threshold value, emitting a spike to the postsynaptic neurons. After that, a relaxing phase follows, where the membrane potential is restored to the resting value before new input stimuli increase the potential again [2]. The information in SNNs is encoded by the signal itself and its timing [3].

Due to these features, neuromorphic computing exhibits low latencies and low energy consumption. In order to take advantage of the SNN principle and make it available, the Advanced Processor Technologies Research Group (ATP) at the University of Manchester develops a manycore computer architecture SpiNNaker (Spiking Neural Network Architecture) to simulate some operational aspects of the human brain [4]. This computing platform has been deployed in several research fields, such as robotics. One example is the "line follower robot" from ATP [5] where a DVS signal is sub-sampled to match SpiNNaker's processing speed. The system employs a PC to convert protocols between DVS, SpiNNaker, and wheel motors. Another example, PushBot [6], is a mobile robot that employs two microcontroller units (MCUs) that are used to connect with SpiNNaker and other external devices, respectively. It communicates with the other MCU through WIFI to send and receive DVS data and commands for the controlling motor. An autonomous mobile platform [7], based on a 48-chip SpiNNaker board, uses an MCU for communication

between SpiNNaker and other components and a Complex Programmable Logic Device (CPLD) interface board. It is able to achieve two independent tasks: trajectory stabilization using real-time computed optic flow and stimulus tracking with Nengo [8].

Different Interfaces Boards (IB) were implemented for SpiNNaker, examples are the FPGA-based solution from the SpiNNaker team [9] and the MCU interface [10] jointly developed by the University of Manchester and the Technical University of Munich. The former provides a unidirectional data transfer, which means the interface can only send data from input sensors to SpiNNaker and makes it difficult to connect other external devices. In the latter, the data flow is bidirectional, but necessitates two data format conversions and two chips for symbol transmission, resulting in greater power consumption.

The system introduced here represents the first complete prototype which in comparison to the previous work [11] has an integrated power supply and power regulators, a built-in sensor for the reward signal and new communication software for the microcontroller. This is the first self-contained and stable version of our robotic system. The implemented Interface Board links together and powers the whole system consisting of a Dynamic Vision Sensor (DVS128), a SpiNNaker board (SpiNN-3), an MCU (Arduino Due) and a digital motor (Futaba S9257), using batteries. In addition, the board allows multiple inputs from different types of sensors, entirely managed by the MCU. The input signals are encoded in a way that allows communication with SpiNNaker, which is working as "the brain", analyses the signal and sends back the result to the MCU, which is then able to drive a device such as a servo as an output. Figure 1 shows the configuration we used to test our IB.

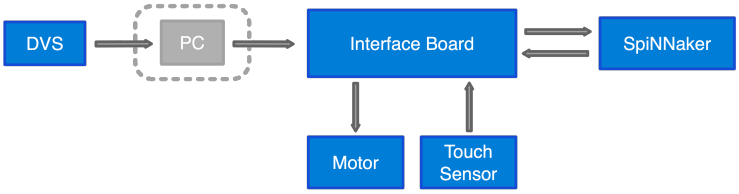

Figure 1. Communication block diagram of a system consisting of different neuromorphic hardware units (DVS and SpiNnaker), and a digital motor, touch sensor, and the Interface Board with an MCU which represent the backbone of the system. Here we tentatively include a PC, which is used during the setup phase to adjust the field of view of the camera and to analyze the produced data and can be removed during the operational phase of the robot.

2. Interface Board Design and Specification

The proposed Interface Board is designed to meet compactness, portability and low power consumption specifics, allowing the use of different types of input sensors (as vision, sound, chemical, temperature sensors), and different output devices (as motors, alarms, lights, actuators). With a size of only 120 mm (width) × 120 mm (length) × 55 mm (height), our platform consists of a PCB board that allow the direct connection of an MCU (using direct pin connection) and a SpiNNaker board (fixed on the board and connected with a dedicated cable). The Arduino Due was selected as MCU for its high number of GPIO ports and adequate processing capacity. It is used to perform communication protocols, so that input sensors, SpiNNaker and the actuators can communicate with each other. The board provides the connection between different robotic components and relevant functions (e.g., inbound links, level shift, voltage regulation). A compact interface board provides a 'spinal cord' which connects the peripheral 'organs' (i.e., vision and sound sensors or actuator) to the 'brain' (SpiNNaker), allows them to talk to each other and pre-processes the signals. Figure 2 shows the data flow of the hardware components introduced previously in Figure 1. It is possible to distinguish three main phases: input signal processing, communication with SpiNNaker and output command execution.

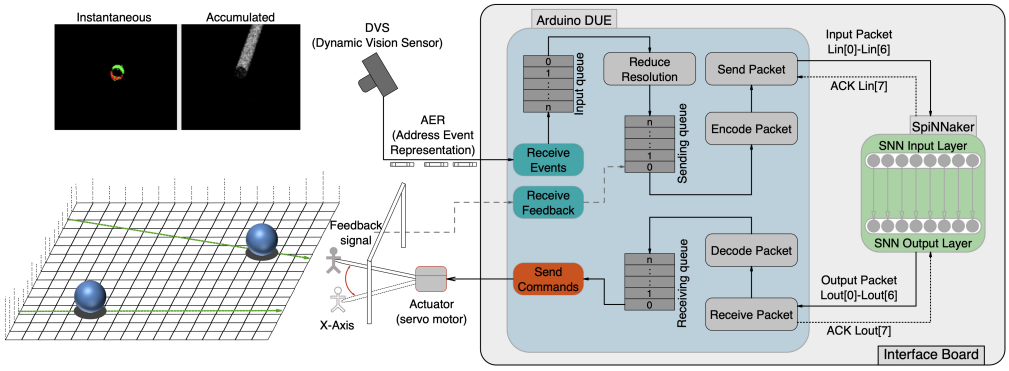

Figure 2. Interface Board connected to different hardware units. The schema shows three main communications pathways: the data flow from the camera (DVS) and the touch sensor to the SpiNNaker via the MCU board, the dataflow from SpiNNaker to MCU, and communication with the goalkeeper (position commands). The Arduino Due board (MCU) is installed on the Interface Board and allows the input–brain–actuator communications, performing data pre-processing, encoding/decoding and command execution. The MCU is linked to a SpiNN-3 board that simulate a simple SNN with 8 input layers and 8 outputs layers. The final position is the SNN computation.

2.1. SpiNNaker

SpiNNaker is a multi-core, real-time computer [12] that simulates brain networks. It uses ARM's high-performance embedded processors [13,14], and thus utilizes their very low power and computational efficiency. Several different SpiNNaker platforms have been developed, we are using the smallest SpiNN-3 board, but the communication protocol works for all the other models. SpiNN-3 board consists of four chips, each having 18 ARM968 processing cores and local memory [15]. The power supply of the SpiNN-3 board is 5 V 1 A via a 2.1mm DC port. For peripheral connections, the SpiNN-3 board has 2 ports: ports J1 and J2 which are 34-way sub-miniature head sockets [16]. The pin assignments are shown in Figure 3. The input part of the link includes inputs Lin(6)–(0) and output LinACK. The output part of the link includes input LoutACK and output Lout(0)–(6).

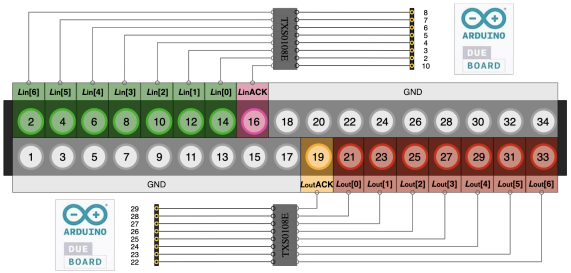

Figure 3. SpiNNaker J1–J2 connector pinout. The image shows the communication link with the MCU (Arduino Due). Two different channels for sending and receiving data are used on the J2 port.

2.2. Arduino Due

Arduino Due board was selected because of the high number of GPIO ports and for the presence of a USB host connection that allows plugging in a USB device (DVS camera sensor). The Arduino Due is based on a 32-bit ARM chip with a clock speed of 84 MHz. The recommended power supply range of the Arduino Due is 7–12 V. On our board, the Arduino is powered with 9 V. To communicate with the J2 link 16 GPIO are necessary, which are covered by the 54 GPIO pins of the Arduino MCU. Thus, there are many unused GPIO pins available to connect with other external devices for further development. The Arduino Due also provides 5 V, 3.3 V and 2 DAC ports to supply power to external devices.

For communication with PC, DVS or eDVS, this board has a UART and 2 micro-USB ports with a maximum 115,200 baud rate.

2.3. Hardware Links

The layout of our PCB for the Interface Board is shown in Figure 4. As stated in the SpiNNaker-3 and Arduino Due sections, they work at different input voltages. In order to use a unified input current for the IB, two different voltage regulators, the MC7809, which is used to provide a stable source of power at 9 V to Arduino, and the MC7805, which supplies power to SpiNNaker at 5 V, were used.

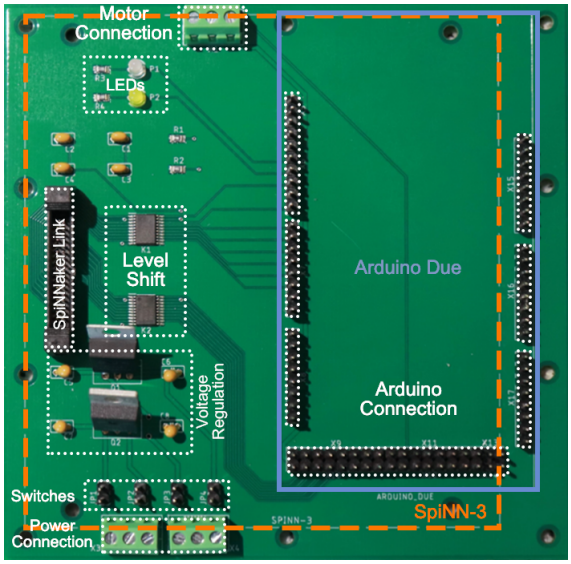

Figure 4. PCB layout. The two-level shift converts the 1.8 V signal from SpiNNaker to the 3.3 V of the Arduino Due board. On the right, the Arduino Due pinout allows the direct connection of the board on the PCB. On the bottom left, two power regulators are used for the batteries connection.

The GPIOs of the Arduino operate at 3.3 V; the SpiNN-3 board port operates at 1.8 V. Thus, level shifters are needed to perform conversions between 3.3 V and 1.8 V, and reverse. TXS0108E chips are used on this PCB for that purpose, which provides 8-way bidirectional level shifts. Therefore, two TXS0108E chips are adequate for 16 GPIO communications. The maximum data rate of this chip is 110 Mbps, which will not be a limitation of the system data rate. Table 1 shows the links between the SpiNNaker-J connector and the Arduino Due ports, passing through the voltage regulators. In Figure 4, (right side) it is possible to see the Arduino Due connection pins, where the MCU is connected directly to the IB. Table 2 shows the GPIO connection between Arduino Due and the main components.

Table 1. SpiNNaker link with Arduino Due ports.

SpiNNaker Link	1	3	5	7	9	11	13	15	17
Arduino Due	GND	GND	GND	GND	GND	GND	GND	GND	GND
SpiNNaker Link	2	4	6	8	10	12	14	16	18
Arduino Due	8	7	6	5	4	3	2	10	GND
SpiNNaker Link	19	21	23	25	27	29	31	33	
Arduino Due	29	28	27	26	25	24	23	22	
SpiNNaker Link	20	22	24	26	28	30	32	34	
Arduino Due	GND	GND	GND	GND	GND	GND	GND	GND	

Table 2. Arduino Due link with others components.

	Arduino Due				Board
	Pins	**N Pins**	**Type**	**Voltage**	**Shifted to**
SpiNNaker	2–8, 10, 22–29	16	Digital	3.3 V	1.8 V
Servo	39 (data), Vin, Gnd (power)	1	PWM	3.3 V	n/a
Touch Sensor	13 (data), Vin, Gnd (power)	1	Digital	3.3 V	n/a
DVS	USB	n/a	Serial	5 V	n/a
Power (batteries)	Vin, Gnd	2	DC	15 V	9 V

3. Communication Protocol

In the previous sections, we described the system from the hardware point of view, focusing on connecting the components (Figure 2). To permit the correct exchange of data between SpiNNaker and the MCU, two different communication protocols were used: the 2-of-7 coding protocol and the 2-phase handshake protocol. For data coding and encoding, a self-timed 2-of-7 coding protocol is used to transmit data in packet form so that the data can be recognized by the SpiNNaker. The 2-phase handshake protocol is used to make sure each packet is successfully transmitted and received by the receiver.

3.1. 2-of-7 Coding

In Spiking Neural Networks information is represented as a time-dependent sequence of spikes, such as a sequence of bits transmitted on a channel. Different coding protocols have been used in SNN each with some limitations. Conventional rate coding counts spikes in fixed time windows to represent each information unit (i.e., encode one alphabet letter or one number), resulting in a relatively slow technique. A faster coding solution is possible with rank order coding, with whom it is possible to use shorter time windows to represent more bits of information. For a real-time scenario, the delay must be as short as possible, prioritizing the transmission time and sacrificing the amount of information that can be represented [17]. N-of-M coding is a protocol that mirrors these properties, allowing a parallel transmission of N bits to represent M bits of information. It is a Non-Return to Zero (NRZ) protocol meaning that the voltage level is not reset to zero after each bit [18]. SpiNNaker uses a 2-of-7 coding protocol to communicate with external devices, where the sender just needs to change the logical level of two wires during each symbol transfer. The states of the other five wires are not changed [19]. In data transfer, there are a total of 17 symbols that can be conveyed, as shown in Table 3. These symbols are able to present 16 hexadecimal digits (from 0 to F) and an EOP sign (which means the end of packets).

Table 3. 2-of-7 coding. The table shows the changing bit to represent HEX numbers plus one extra EOP signal.

Symbol	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**A**	**B**	**C**	**D**	**E**	**F**	**EOP**
bit 1	0	1	2	3	0	1	2	3	0	1	2	3	0	1	2	1	5
bit 2	4	4	4	4	5	5	5	5	6	6	6	6	1	2	3	3	6

3.2. 2-Phase Handshake Protocol

In the 2-phase handshake protocol, the sender first transmits a symbol to the receiver, then waits for the receiver to provide an acknowledgment signal. The receiver will send an acknowledgment signal to the sender after receiving a symbol from the sender.

3.3. Encoding/Decoding Packets

After MCU receives the events from input sensors, MCU will encode the data into packets and send them to SpiNNaker in real time. There are two types of SpiNNaker packets of different sizes: 40 bits and 72 bits [16]. 'EOP' symbols are placed after each packet to show that the transmission of that packet was completed. In comparison to the

40-bit packet, the 72-bit packet contains a 32-bit extra payload. The 72-bit packets are also called nearest neighbor (NN) packets; the 40-bits packets are also called multi-cast (MC) packets. The header and packet data are included in both types of packets. The packet data contains the spike information, while the header contains the packet's specification. The packet type is indicated by the last two bits in the header, where '01' indicates NN packets and '00' indicates MC packets. The parity bit indicated is odd parity, which means that the total number of 1s in the packet, excluding the EOP, is odd. The value of the parity bit is determined after the rest of the packet bits are decided.

The packets are divided into hex numbers and sent in symbols with the 2-of-7 coding protocol. After one packet is received, the hex number of each symbol will be transformed into binary numbers (The SpiNNaker starts to read the packet from the end of the packet, which means the order of the symbols in each part of the packet is reversed and the symbol conversion between hex numbers and binary numbers is also reversed. For instance, the SpiNNaker reading order of 'C000' is '000C', where the hex number 'C' converted into binary numbers is '0011' instead of '1100').

For packets sent from MCU to SpiNNaker, all the packets are MC packets, which have the size of 40 bits. When the SpiNNaker communicates with an external device, the external device is regarded as a virtual chip inside the system for the SpiNNaker. Thus, a virtual routing key is needed (4 hex numbers sent in the second half of the packet data part).

4. Evaluation

The evaluation of the IB (Figure 5) starts with analyzing the communication data rate between Arduino and SpiNNaker (both up-link and down-link). The tests are executed by sending predefined data packets with a time interval of 15 ms.

4.1. Network Topology

For the purpose of this work, a simple neural network was configured (Figure 2 green box). The number of inputs corresponds to the 8 possible positions of the input spikes received from the DVS sensor. The output is composed of 8 neurons that reflect the input to the output without using a learning rule to predict the final position of the target ball. The integration of the touch signal with a learning rule is under consideration for future work.

4.2. Up-Link Data Rate

With a time interval between packets set to 1.5 ms and 0.15 ms, respectively, the communication is able to operate successfully, but the SpiNNaker issues a warning that the time interval between packets is too small. The speed of up-link communication exceeds the processing speed of SNN hosted on SpiNNaker. The SpiNNaker receives 112 packets in 50 ms, which corresponds to 24.64 ksymbols/s or 12.32 kbytes/s.

4.3. Down-Link Data Rate

When the Arduino sends the predefined packets to SpiNNaker at maximum speed, the Arduino receives all the predefined packets from SpiNNaker. Thus, the downlink data rate should be equal to or larger than the up-link data rate. The data rate of the whole system is limited by the up-link data rate, which is 24.64 ksymbols/s.

4.4. Accuracy

The accuracy of the 'robotic goalie' blocking the balls heavily depends on the SNN which runs on SpiNNaker and the speed of approaching balls. To compute the accuracy of our prototype, we take into account the number of saved balls over the total number of launches. The experiment consists of 100 ping-pong balls launched from approximately 1 m with various speeds and directions. When the speed of the approaching ball is relatively low (up to approximately 1 m/s), the 'robotic goalie' can intercept the ball with a 75% success rate. As the ball speed increases, the accuracy of the system gradually decreases. For the balls with fast speed, The robotic system can not maneuver the goalie to the right

location prior to the balls arriving. The reason is that the SNN used in this project has not been taught to anticipate the track of balls on the basis of their movement at the early stages of their approach to the goal.

4.5. Latency

The response latency in the prototype is the time it takes for the DVS to catch the ball's movement and the servo motor to move to the desired location. The minimum time step in DVS is 1 µs. As a result, DVS needs at least 1 µs to produce an event and another 1 µs to communicate the event. Then, the MCU keeps receiving these events (via the serial port) and encodes them into packets every 500 µs. The next relevant parameter is the communication speed between the MCU and SpiNNaker. Sending a packet to SpiNNaker takes $\frac{1}{2240}$ s ≈ 0.45 ms and receiving a packet by the MCU takes 0.45 ms. This is longer than the time it takes for DVS to capture an event and for the MCU to receive it. Furthermore, for every 1 µs, the MCU verifies the condition of all processes. As a consequence, events are received, packets are sent or received, instructions are generated, and commands are executed in parallel. The MCU sends the second converted event to SpiNNaker at the same time it receives the first packet from SpiNNaker. The simulation on the SpiNNaker runs with 1 ms time step. Thus, the minimum time for the SpiNNaker processing data is 1 ms. Furthermore, the packets are queued and the commands are created using the most recent N_{pac} received packets, and the uplink and downlink speeds are virtually identical. As a result, the time spent on operations prior to generating servo control commands (t_{command}) is the sum of DVS events, communication to MCU, encoding time, time to communicate to SpiNNaker, processing and time to send back the response to the MCU with N_{pac} packets:

$$t_{\text{command}} = (0.001\,\text{ms} + 0.001\,\text{ms}) \cdot N_{\text{events}} + 0.032\,\text{ms} + 0.5\,\text{ms} + \frac{1}{2240}\text{s} + 1\,\text{ms} + N_{\text{pac}} \cdot \frac{1}{2240}\,\text{s}$$

For $N_{\text{events}} = 100$ and $N_{\text{pac}} = 20$, we theoretically estimate this time to be approximately 11 ms. If the number of events needed to create an output from the SNN running on SpiNNaker is increased to $N_{\text{events}} = 1000$, the latency will increase only to 13 ms. This we compare with the latency of the executive organ, such as a servo motor. For example, in our robogoalie demonstrator [20] (Figure 6) we use a Futaba digital motor, which has a speed of 60° of angular rotation in 75 ms. Therefore the whole system was reasonably successful (about 85% of the time) in intercepting the ball, with speeds of up to 1 m/s.

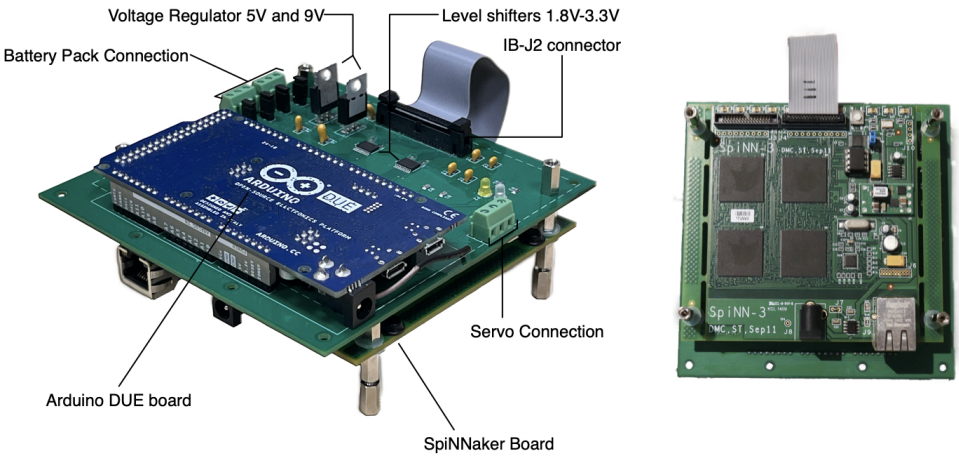

Figure 5. Hardware components installed together. The image shows the connection of the SpiNNaker board at the bottom of the IB and the Arduino Due board on the top, resulting in a compact and portable system.

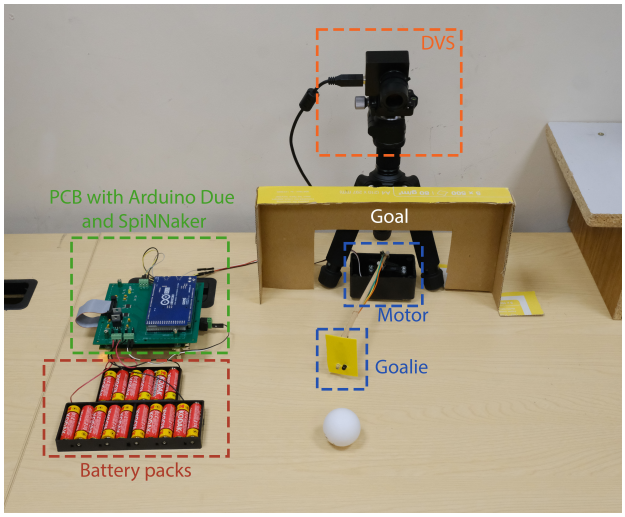

Figure 6. A complete demo setup for the Interface Board, demonstrating a robot goalkeeper [20]. The DVS camera was connected to a PC for data monitoring and the data was forwarded to the Arduino Due board. A video demonstration is available at https://youtu.be/0UqKsB0lQr8 (accessed on 21 October 2022).

5. Discussion

The aim of our work was to implement a compact Interface Board Platform to allow communication between an MCU and a SpiNNaker board. We used an Arduino Due board to manage the communication between the various peripheral hardware and the SpiNNaker board, adapting the signal with two voltage level shifters.

The test environment is represented by the goalkeeper task, where a robotic arm intercepts an incoming ball moving toward the horizontal axis. To "see" the target object, a Dynamic Video Sensor camera was used, exploiting its fast, non-redundant information transmission and low power consumption. However, there is no learning rule that allows the prediction of the final position of the ball (it is under consideration for future works). This lack of the SNN model limits the tests to specific environmental conditions (e.g., light, noise). In fact, a learning rule can work in different conditions, enabling a dynamic adaptation to many conditions. The current implementation of the system considers the horizontal axis only, resulting in a failure interception when a bouncing ball is thrown toward the goal. This is not a limitation because is out of the aim of this work but, it is possible to consider the installation of a two-axis actuator to block the target in the 2D space.

Although there are some limitations in our system for the goalkeeper task, it is possible to find different strong points. Looking at the reaction time, our Interface Board consumes only 13 ms for 1000 events (considering that the DVS have a resolution of 128 $\times$ 128 [21] or 16,384 pixels, 1000 events is reasonable for the tests) to actuate the decision, showing the fast communication between the components. Then, considering that the whole system works with usual batteries, the low power consumption, and its portability are the features that differ from other similar works, breaking down considerably their consumption. Table 4 provides a comparison of our prototype with other neuromorphic projects, tested on the same task or with potential experimentation in solving our similar problem. The main difference, excluding the motors used and weight and only considering the power computation devices, is the power consumption. It is clear how the fast communication of our Interface Board in combination with neuromorphic hardware, completely reduces energy consumption.

Table 4. Comparison of our Interface Board with neuromorphic robotic platforms.

Name	Description	Vision Sensor	Power Consumption	Positioning Time	Accuracy
Quadrupedal Robotic Goalkeeper [22]	The Intel camera is used to track the target ball and send the prediction to the Mini Cheetah. A GPU is used to train the model using the YOLO algorithm.	Intel RealSense D435i	120 W/h (Mini Cheetah max) [23]	0.5 s/4 m field	(sidestep) ∼65% (full) ∼85%
iCub v1.0/v2.0 (Intel ATOM D525) [24]	Humanoid robot with an embedded pc. It is composed of different actuators to simulate human motions.	PointGrey Dragonfly v2 (640 × 480 30 fps)	288 W/h (960 W/h peak)	n/a	n/a
Fetch (and Freight) (Intel i5, Haswell) [25]	Fetch robot is a mobile manipulator to catch and move objects (until 6 kg)	Primesense Carmine 1.09	20 W/h (36 W/h peak)	n/a	n/a
spiNNaLink (this work)	Interface Board Platform to link an MCU with a SpiNNaker board. A DVS camera is used to reveal the ball direction, maintaining a low information rate and low power consumption.	Dynamic Vision Sensor 128	∼7 W/h (Whole system max)	0.150 s/1 m field	75%

6. Conclusions

In this paper, we build spiNNaLink, an Interface Board to link a SpiNN-3 board with an MCU, allowing the usage of multiple input sensors and output actuators. The implemented PCB board works with a set of 1.5V AA rechargeable batteries making it small and portable (Figures 5 and 6). The Arduino Due is utilized to perform communication protocols and data conversion between devices. The use of a specific MCU is not a limitation and it is related only to this particular version of the Interface Board. In fact, considering that the level shifters can work between 1.65 V to 5.50 V and that it is possible to use port extender chips to increase the GPIO ports, future versions of the Interface Board could be used with different MCUs. Furthermore, to overcome the lack of a USB host port, it is possible to consider eDVS cameras communicating directly using the Serial port. The power supply for the entire system, including Arduino and SpiNNaker, is also provided by the interface board. Our tests show that the board allows fast communication link from input sensors (DVS) to the output channel (motor), resulting in a delay of ∼11 ms. The interception accuracy is sensible to the ball speed and direction, due to the simple SNN developed without learning rule. This is because the main purpose of the project was related to the hardware link and the Interface Board. This limit can be treated in future research where the network can be trained to predict the final position of the ball and, additionally, a reward feedback signal can enable self-learning in this task.

The most relevant feature of our prototype is the very low power consumption that compared with similar, or task-related, neuromorphic robots, breaks down completely the consumptions, maintaining a high computational power even with the usage of batteries. Moreover, projecting us into a future scenario where more components will compose a more complex system, it is easy to imagine how the usage of extremely low power consumption hardware can permit the development of bio-inspired, autonomous and energy-independent devices.

Author Contributions: Conceptualization, N.R. and H.H.; methodology, N.R. and H.H.; software, N.R. and H.H.; validation, N.R., E.D., T.M. and K.N.; investigation, N.R. and H.H.; resources, K.N.; data curation, H.H.; writing—review and editing, N.R., T.M. and K.N.; supervision, T.M. and K.N. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by University of West London.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

ACK	Acknowledge Packet
ARM	Advanced RISC Machine Processor
ATP	Advanced Processor Technologies Research Group
CPLD	Complex Programmable Logic Device
DAC	Digital-to-Analog Converter
DC	Direct Current
DVS	Dynamic Video Sensor
EOP	End of Procedure
FPGA	Field Programmable Gate Arrays
GPIO	General Purpose Input Output
IB	Interface Board
MC	Multi Cast
MCU	Micro Controller Unit
NN	Nearest Neighbour
NRZ	Non Return to Zero
PCB	Printed Circuit Board
SNN	Spiking Neural Network
UART	Universal Asynchronous Receiver-Transmitter
USB	Universal Serial BUS

References

1. Ponulak, F.; Kasiński, A. Supervised learning in spiking neural networks with ReSuMe: Sequence learning, classification, and spike shifting. *Neural Comput.* **2010**, *22*, 467–510. [CrossRef] [PubMed]
2. Ghosh-Dastidar, S.; Adeli, H. Spiking neural networks. *Int. J. Neur. Syst.* **2009**, *19*, 295–308. [CrossRef] [PubMed]
3. Wei, D.; Harris, J.G. Signal reconstruction from spiking neuron models. In Proceedings of the 2004 IEEE International Symposium on Circuits and Systems (IEEE Cat. No.04CH37512), Vancouver, BC, Canada, 23–26 May 2004.
4. Furber, B.S.; Galluppi, F.; Temple, S.; Plana, L.A. The SpiNNaker project. *Proc. IEEE* **2014**, *102*, 652–665. [CrossRef]
5. Davies, S.; Patterson, C.; Galluppi, F.; Rast, A.D.; Lester, D.; Furber, B.S. Interfacing real-time spiking i/o with the spinnaker neuromimetic architecture. *Aust. J. Intell. Inf. Process. Syst.* **2010**, *11*, 7–11.
6. iniLabs. Pushbot. Available online: https://inilabs.com/products/pushbot/ (accessed on 1 May 2021).
7. Galluppi, F.; Denk, C.; Meiner, M.C.; Stewart, T.C.; Plana, L.A.; Eliasmith, C.; Furber, S.; Conradt, J. Event-based neural computing on an autonomous mobile platform. In Proceedings of the 2014 IEEE International Conference on Robotics and Automation (ICRA), Hong Kong, China, 31 May–7 June 2014; pp. 2862–2867.
8. Bekolay, T.; Bergstra, J.; Hunsberger, E.; DeWolf, T.; Stewart, T.C.; Rasmussen, D.; Choo, X.; Voelker, A.R.; Eliasmith, C. Nengo: A Python tool for building large-scale functional brain models. *Front. Neuroinform.* **2013**, *3*, 48. [CrossRef] [PubMed]
9. Plana, L.A. Interfacing Aer Devices to Spinnaker Using an Fpga. Available online: http://spinnakermanchester.github.io/docs/spinn-app-8.pdf (accessed on 1 July 2021)
10. Denk, C.; Llobet-Blandino, F.; Galluppi, F.; Plana, L.A.; Furber, S.; Conradt, J. Real-time interface board for closed-loop robotic tasks on the spinnaker neural computing system. In Proceedings of the International Conference on Artificial Neural Networks (ICANN), Sofia, Bulgaria, 10–13 September 2013; pp. 467–474.
11. 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. [CrossRef]
12. Furber, S.; Bogdan, P. *SpiNNaker: A Spiking Neural Network Architecture*; Now Publishers: Delft, The Netherlands, 2020.
13. Furber, S.; Temple, S. Neural systems engineering. *J. R. Soc. Interface* **2006**, *4*, 193–206. [CrossRef] [PubMed]
14. Furber, S.B.; Temple, S.; Brown, A.D. High-performance computing for systems of spiking neurons. In Proceedings of the AISB'06 Workshop on GC5: Architecture of Brain and Mind, Bristol, UK, 3–4 April 2006; Volume 2, pp. 29–36.
15. Temple, S. Spinn-3 Development Board. Available online: http://spinnakermanchester.github.io/docs/spinn-app-1.pdf (accessed on 1 July 2021)
16. Temple, S. Spinnaker Links. Available online: http://spinnakermanchester.github.io/docs/spinn-app-7.pdf (accessed on 1 August 2021)

17. Christensen, D.V.; Dittmann, R.; Linares-Barranco, B.; Sebastian, A.; Gallo, M.L.; Redaelli, A.; Slesazeck, S.; Mikolajick, T.; Spiga, S.; Menzel, S.; et al. 2022 roadmap on neuromorphic computing and engineering. *Neuromorphic Comput. Eng.* **2022**, *2*, 022501. [CrossRef]
18. Mori, H.; Satake, T.K.M.; Matsuoka, T. Communication System with a Plurality of Nodes Communicably Connected for Communication Based on nrz (Non Return to Zero) Code. U.S. Patent 20,120,051,241, 3 January 2012.
19. Brouwer, A.; Shearer, J.; Sloane, N.; Smith, W. A new table of constant weight codes. *IEEE Trans. Inf. Theory* **1990**, *36*, 1334–1380. [CrossRef]
20. 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 October 2022; pp. 249.
21. Lichtsteiner, P.; Posch, C.; Delbruck, T. A 128 × 128 120 db 15 us latency asynchronous temporal contrast vision sensor. *IEEE J. Solid State Circuits* **2008**, 43, 566–576. [CrossRef]
22. Huang, X.; Li, Z.; Xiang, Y.; Ni, Y.; Chi, Y.; Li, Y.; Yang, L.; Peng, X.B.; Sreenath, K. Creating a Dynamic Quadrupedal Robotic Goalkeeper with Reinforcement Learning. *arXiv* **2022**, arXiv:2210.04435.
23. Katz, B.; Carlo, J.D.; Kim, S. Mini Cheetah: A Platform for Pushing the Limits of Dynamic Quadruped Control. In Proceedings of the 2019 International Conference on Robotics and Automation (ICRA), Montreal, QC, Canada, 20–24 May 2019; pp. 6295–6301.
24. Natale, L.; Bartolozzi, C.; Nori, F.; Sandini, G.; Metta, G. ICub. *arXiv* **2021**, arXiv:2105.02313
25. Wise, M.; Ferguson, M.; King, D.; Diehr, E.; Dymesich, D. Fetch and Freight: Standard Platforms for Service Robot Applications. In Proceedings of the Workshop on Autonomous Mobile Service Robots, New York, NY, USA, 11 July 2016.