Raspberry Pi based Modular System for Multichannel Event-Driven Functional Electrical Stimulation Control

# Raspberry Pi based Modular System for Multichannel Event-Driven Functional Electrical Stimulation Control

Andrea Prestia, Fabio Rossi, Andrea Mongardi, Danilo Demarchi, Paolo Motto Ros

***Abstract*— This paper describes the implementation and testing of a modular software for multichannel control of Functional Electrical Stimulation (FES). Moving towards an embedded scenario, the core of the system is a Raspberry Pi, whose different models (with different computing powers) best suit two different system use-cases: user-supervised and stand-alone. Given the need for real-time and reliable FES applications, software processing timings were analyzed for multiple configurations, along with hardware resources utilization. Among the results, the simultaneous use of eight channels has been functionally achieved (0% lost packets) while minimizing system timing failures (excessive processing latency). Further investigations included stressing the system using more constraining acquisition parameters, eventually limiting the usable channels (only for the stand-alone use-case).**

***Index Terms*— Functional electrical stimulation, Raspberry Pi, Embedded system, Real-time computing, Rehabilitation engineering**

## I. Introduction

Embedded systems are emerging technologies thanks to their limited power consumption, dimensions, cost, and complexity [1]–[5]. In recent years, the use of embedded systems in the biomedical field has increased, especially for telerehabilitation purposes, as a result of the Covid-19 pandemic [6], [7]. Among the employed treatments in physiotherapy practice, Functional Electrical Stimulation (FES) offers several advantages for recovering motion functionalities thanks to the not invasive stimulation of skeletal muscles [8], [9]. From an engineering point of view, embedded solutions for controlling stimulation application are based on inertial sensors to capture body movements [10], [11], or on the surface ElectroMyoGraphy (sEMG) technique [12], [13] to assess the muscles activation by recording the electrical signals generated during myofibers contraction [14].

Among other architectures, state-of-the-art embedded systems for FES control often include the use of Raspberry Pi (RPi) [15] machines as wearable controllers to support post-stroke gait [16], or cycling for spinal cord injury victims [17], also aiming to sport competitions [18]. On the other hand, among the future perspectives of the investigated works is the use of an RPi for greater modularity of the described system [19]. Main reasons for using an RPi machine as the control platform lie in its versatility, size, and cost [20] compared to a common laptop: indeed, an RPi can be integrated into a portable device to provide a user-friendly solution for healthcare personnel [21]–[23] or, alternatively, it can be adopted as a wireless collector node to stream data to a remote central unit (e.g., telerehabilitation) [24]–[26].

This work was not supported by any organization.

The authors are with the Department of Electronics and Telecommunications (DET), Politecnico di Torino, 10129 Turin, Italy (corresponding author e-mail: andrea.prestia@polito.it)

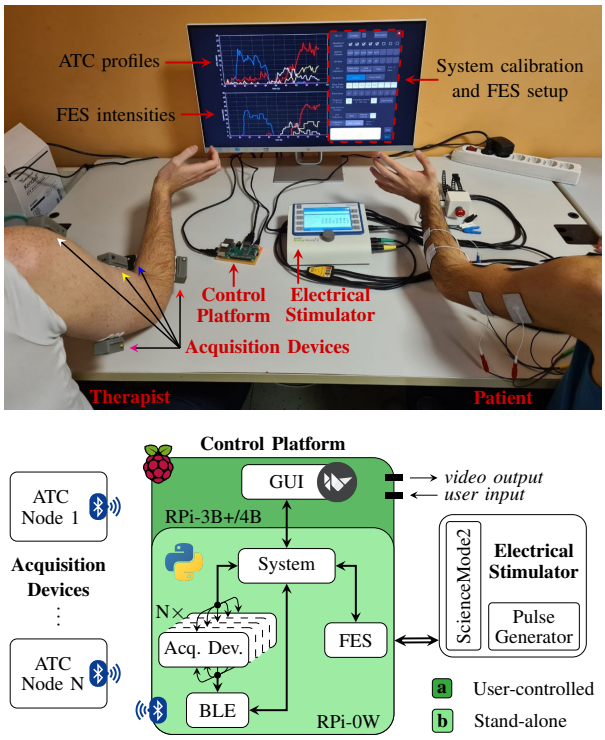


Fig. 1. (**top**) Standard system usage setup with a GUI; (**bottom**) The flow of the ATC-FES operation starts from the ATC processing of the sEMG signals on the acquisition devices and ends with the application of FES, thanks to the translation of the muscular data into stimulation patterns performed by the control platform. Here, the software routines manage extra- (BLE, FES, and GUI) and intra- (among objects) modules communication and operations by implementing a multi-threaded approach.

The RPi-based system involved in this work (see Fig. 1) results from an embedded implementation for modulating FES according to muscle activity [27], [28], using the event-based Average Threshold Crossing (ATC) technique to process the sEMG signal directly on the acquisition board [29]. ATC consists in generating a digital event every time the analog sEMG signal crosses a predefined threshold. The number of events generated within a time window is transmitted as input to the FES control system instead of the whole sEMG signal, which is therefore not even sampled [30]. The information synthesis resulting from the ATC simplifies the data processing phases [31], thus leading the FES control towards the minimization of the processing latency between

the input signal and the output stimulation, and improving real-time operations [13]. The interaction between the ATC acquisition devices and the electrical stimulator is provided by an RPi, acting as the control platform of the system. This single-board computer is equipped with suitable hardware resources for inter-operating the system modules and for the development of a custom Graphical User Interface (GUI) for system management.

Complementary to what we reported in [28], which investigates how the previous version of our system performs in controlled rehabilitative sessions involving 17 healthy subjects, this work analyzes the software performance of our new implementation from a technical point of view. Therefore, different test benches have been set up for several configurations, varying the computational effort, in two different use cases, and taking into account up to three RPi machines, with different computing powers, as control platform.

## II. System Overview

Fig. 1 (top) depicts the typical using condition of the system, where the muscular activity of the first subject modulates the stimulation applied to the second one to obtain the replication of the performed movement. In a rehabilitation scenario, the two subjects can be a therapist and her/his patient, respectively [32]. The functional architecture of the ATC-FES system can be conceptually schematized as a cascade flow of three parts (see Fig. 1 (bottom)). The acquisition devices (each working as a standalone module) extract the information from muscle contractions by ATC-processing the sEMG signal and stream the result through Bluetooth Low Energy (BLE) communication [29]. The control platform receives the ATC data from all the available boards and defines an appropriate stimulation profile based on muscular activation [27]. The electrical stimulator (i.e., RehaStim 2 [33]) closes the loop by inducing the generated biomimetic pattern [13].

Considering the above process, we can notice how the control platform represents the central core of the system, making possible to bridge input and output devices effectively. As a consequence, the performance of the machine running the control software determines the limit of system applicability, especially when the processor has to be selected suitable for embedded computation [1]. Following these considerations, in this paper we propose and analyze our implementation for two use-cases:

a) *User-controlled interface*: the system is managed by means of a Graphical User Interface (GUI), which allows the user to actively supervise the stimulation session and provides a visual feedback on muscular activation and FES pulses generation.
b) *Stand-alone system*: user interaction is limited to remote control, and main ATC-FES operations run independently from a direct user supervision.

Clearly, the first application requires adequate devices and graphical (hardware) supports to work properly, demanding high-performance machine resources w.r.t. the stand-alone application, which relaxes the computational constraints by minimizing user-system interactions. We took into consideration these dissimilarities by selecting as control platforms, belonging to the RPi family, the most appropriate boards, which ensure the compatibility with our test benches due to their portable, open design, wireless connectivity, and system-on-chip features [15]. In particular, looking at the RPis reported in Table I, we chose the RPi-4B and RPi-3B+ machines to run application a), also aiming to verify if the software operates unaltered while scaling down the platform performance. Instead, the lighter requirements of application b) allowed us to opt for the RPi-0W device, whose reduced computing capacity better adapts to the power constraints of an embedded scenario.

TABLE I
Raspberry Pi boards features for control platform

| Model | #core | Architecture | $f_{CLK}$ (GHz) | RAM (GB) | BLE |
|---|---|---|---|---|---|
| RPi-4B | 4 | Cortex-A72 (64-bit) | 1.5 | 4 | 5.0 |
| RPi-3B+ | 4 | Cortex-A53 (64-bit) | 1.4 | 1 | 4.2 |
| RPi-0W | 1 | ARM11 (32-bit) | 1 | 0.5 | 4.1 |

## III. Software Implementation

The control software has to fulfill multiple technical requirements, to be compliant with different biomedical applications. In particular, it must be scalable, to perform efficiently even with multiple interfaced devices, it should be modular, according to object-oriented programming [34], to ease the development of each module, and it has to be sufficiently reliable, to minimize failure probability and to ensure users' safety [35]. Moreover, the real-time control of FES represents an additional and essential application requirement [36]. We decided to design the system software using the Python programming language because of its versatility and rapidity of development, which sped up the evolution of the control system across different hardware platforms and Operating Systems (OSs), from common computers to embedded systems [13].

Regarding the management of peripherals communication, the control of the on-board BLE transceiver has been implemented taking advantage of the Bluepy library [37], which relies on BlueZ [38] (the official GNU/Linux Bluetooth stack), and the RehaStim2 electrical stimulator has been controlled through the ScienceMode2 communication protocol [39]. In this second case, a custom library implementation, relying on the Pyserial [40] module, allows the user to conveniently update the stimulation parameters (e.g., pulse amplitude, width and frequency) during on-going stimulation.

On the other hand, the main system data flow and the computation of FES parameters are handled by means of a completely custom module. In particular, ATC input values are transformed into stimulation intensities using a Look-Up Table (LUT), whose parameters are defined by a linear regression algorithm normalized on the maximum output

values obtained during an initial calibration [13], [27]. Our custom LUT implies a very fast and responsive ATC processing, thus not impacting on the overall computational timings and providing the output parameters with the lowest latency possible [13].

Then, for the user-controlled case (see Sec. II), we chose to develop the GUI using the Kivy Python-compatible framework [41], leveraging the GPU for performance maximization. In particular, considering the reliability requirement, *ad-hoc* Kivy buttons and graphs have been combined together to provide the end-user with the complete control of the system.

All above considered, the architecture of the software is divided into multiple modules (Fig. 1). The BLE and FES modules handle communications with the external devices, and one object is instantiated for each connected acquisition board, storing the features of each sensing units. The System core is responsible for all the data processing and internal communications between software modules, while the GUI (whenever available, depending on the use-case) allows the user to supervise the system behavior and to intervene if needed.

The tasks of these modules have been assigned to multiple threads, in order to concurrently execute operations and achieve an overall real-time performance. In particular, one thread per module has been created, resulting in a minimum of 4 threads when in the use-case b) with only 1 acquisition device linked (System + FES + BLE + 1 Acq. Dev.) and a maximum of 12 operative threads when using the implementation a) with 8 devices involved (System + FES + BLE + GUI + 8 Acq. Dev.).

Multiple thread-safe queues [42] have been configured to obtain proper communication among objects, with software data exchanges only on user-defined event-triggers, also promoting system scalability by easing the concurrent management of multiple channels. Furthermore, this implementation allows the system to operate in synchronous or asynchronous mode. In the first case, the ATC processing is performed only after the proper reception of all the ATC values from the corresponding queues of all the connected boards. Vice versa, in the asynchronous case, the thread does not wait for the availability of all the data, thus updating the FES parameters only for the channels whose ATC packet is received. This behavior depends on the implicit time-shift between each device, working independently from each other, and on the thread handling the Bluetooth communication.

Fig. 2 shows the processing dataflow representing the entire procedure from the ATC data reception to the acknowledgement of the FES parameters update command. The first step is to retrieve the ATC data from the queue of each active acquisition device. Then, the new data are appended to the matrix containing the two most recently acquired ATC values for each channel, and a median operation is performed for noise robustness [13]. The result is used as the index for the LUT to identify the modulated stimulation parameter (e.g., pulse amplitude) for the associated stimulation channel. In order to create the packet containing the pulse parameters, both the results obtained from the LUT and the pre-defined not modulated stimulation parameters are provided as input. The ScienceMode2 communication protocol [39] requires using a 1-byte checksum through Cyclic Redundancy Check (CRC) and the use of byte stuffing to avoid the presence of bytes equal to the start or stop sequence of the packet. Finally, the last steps are the serial transmission (to the electrical stimulator) of the created packet and the corresponding reception of the acknowledgment sent back.

## IV. Test Setup

Processing times were measured from the end of the phase of getting data from the acquisition queues to the acknowledgment reception from the stimulation device (Fig. 2).

Since the purpose of this work is a hardware and software performance assessment, the tests were performed on different system configurations and did not involve the participation of human subjects. The combinations of tested configurations consist of the use of multiple acquisition devices, different windows for the ATC technique, synchronous/asynchronous processing of received data, and the optional use of the GUI.

The number of tested acquisition devices ranges from 1 to 8 (maximum number of channels managed by the electrical stimulator) with a one-to-one control. In particular, for the sake of synthesis and without loss of generality, the tested configurations involved 1, 3, 5, or 8 channels.

Processing synchronization was evaluated to satisfy possible application requirements, which may need to apply FES only when data are received from all connected acquisition devices.

Since shrinking the ATC window is among our future perspectives, tests were also performed by halving its value (i.e., from 130 ms to 65 ms) for the most critical system configurations with 5 and 8 active channels.

Last, although RPi-3B+/4B involve the use of a GUI, the configurations with 8 channels have also been tested without

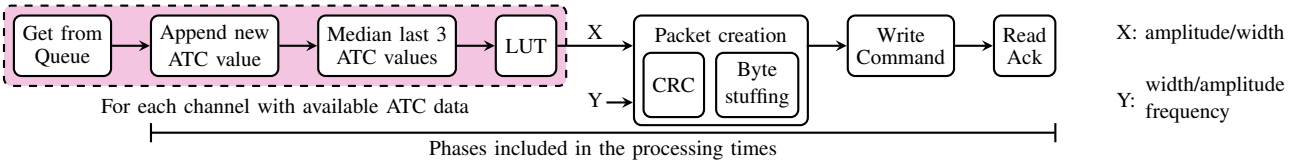


Fig. 2. Following the first phase of retrieving ATC data from the queue of each acquisition device, the modulated stimulation parameter X is computed through three processing steps and provided as input to the (ScienceMode2) packet creation block together with the not modulated parameters Y. The measured processing times start after the get from the acquisition queues, and end at the acknowledgment reception from the stimulation device.

| ID | RPi | N | Sync | Win (ms) | GUI | CPU (%) | RAM (MB) | Losses (%) | Delays (%) |
|---|---|---|---|---|---|---|---|---|---|
| C1 | 4B | 1 | no | 130 | yes | 9.55 | 85.01 | 0 | 0 |
| C2 | | 3 | no | 130 | yes | 12.35 | 86.32 | 0 | 0 |
| C3 | | | yes | 130 | yes | 10.57 | 86.31 | 0.03 | 0 |
| C4 | | 5 | no | 130 | yes | 15.57 | 87.44 | 0 | 0 |
| C5 | | | yes | 130 | yes | 11.93 | 87.3 | 0.1 | 0 |
| C6 | | | no | 65 | yes | 17.35 | 87.36 | 0 | 0 |
| C7 | | | yes | 65 | yes | 14.21 | 87.56 | 0 | 0 |
| C8 | | 8 | no | 130 | yes | 17.9 | 88.97 | 0 | 0 |
| C9 | | | no | 130 | no | 5.6 | 23.2 | 0 | 0 |
| C10 | | | yes | 130 | yes | 15 | 88.8 | 0.1 | 0 |
| C11 | | | yes | 130 | no | 3.38 | 22.89 | 0.14 | 0 |
| C12 | | | no | 65 | yes | 18.93 | 89.33 | 0 | 0 |
| C13 | | | no | 65 | no | 6.95 | 22.84 | 0 | 0 |
| C14 | | | yes | 65 | yes | 17.32 | 88.98 | 0.1 | 0 |
| C15 | | | yes | 65 | no | 4.3 | 23.12 | 1.31 | 0 |
| C16 | 3B+ | 1 | no | 130 | yes | 74.72 | 138.48 | 0 | 0 |
| C17 | | 3 | no | 130 | yes | 71.17 | 139.85 | 0 | 0.07 |
| C18 | | | yes | 130 | yes | 71.83 | 139.42 | 1.53 | 0 |
| C19 | | 5 | no | 130 | yes | 67.62 | 139.98 | 0.14 | 0.11 |
| C20 | | | yes | 130 | yes | 69.03 | 139.41 | 3.04 | 0.07 |
| C21 | | | no | 65 | yes | 65.33 | 140.27 | 7.06 | 12.98 |
| C22 | | | yes | 65 | yes | 66.92 | 139.99 | 12.78 | 5.82 |
| C23 | | 8 | no | 130 | yes | 63.3 | 140.71 | 0.93 | 3.11 |
| C24 | | | no | 130 | no | 10.55 | 23.09 | 0 | 0 |
| C25 | | | yes | 130 | yes | 65.82 | 140.39 | 3.63 | 0.72 |
| C26 | | | yes | 130 | no | 5.7 | 22.92 | 0.18 | 0 |
| C27 | | | no | 65 | yes | 60.45 | 140.95 | 13.4 | 16.99 |
| C28 | | | no | 65 | no | 13.68 | 23.09 | 0 | 0 |
| C29 | | | yes | 65 | yes | 62.88 | 140.41 | 17.87 | 7.08 |
| C30 | | | yes | 65 | no | 8.4 | 23.21 | 0 | 0 |
| C31 | 0W | 1 | no | 130 | no | 37.6 | 22.99 | 0 | 0 |
| C32 | | 3 | no | 130 | no | 64.3 | 22.58 | 0 | 0 |
| C33 | | | yes | 130 | no | 37.7 | 22.85 | 0 | 0 |
| C34 | | 5 | no | 130 | no | 77.9 | 22.98 | 0.02 | 0 |
| C35 | | | yes | 130 | no | 56.2 | 23.21 | 0.02 | 0 |
| C36 | | | no | 65 | no | 89 | 22.76 | 3.36 | 0 |
| C37 | | | yes | 65 | no | 79.7 | 23.26 | 0.12 | 0.11 |
| C38 | | 8 | no | 130 | no | 92.8 | 23.19 | 0.03 | 0 |
| C39 | | | yes | 130 | no | 81.4 | 23.55 | 0.08 | 0 |
| C40 | | | no | 65 | no | 90.6 | 23.21 | 26.56 | 99.66 |
| C41 | | | yes | 65 | no | 90.6 | 23.2 | 26.94 | 99.72 |

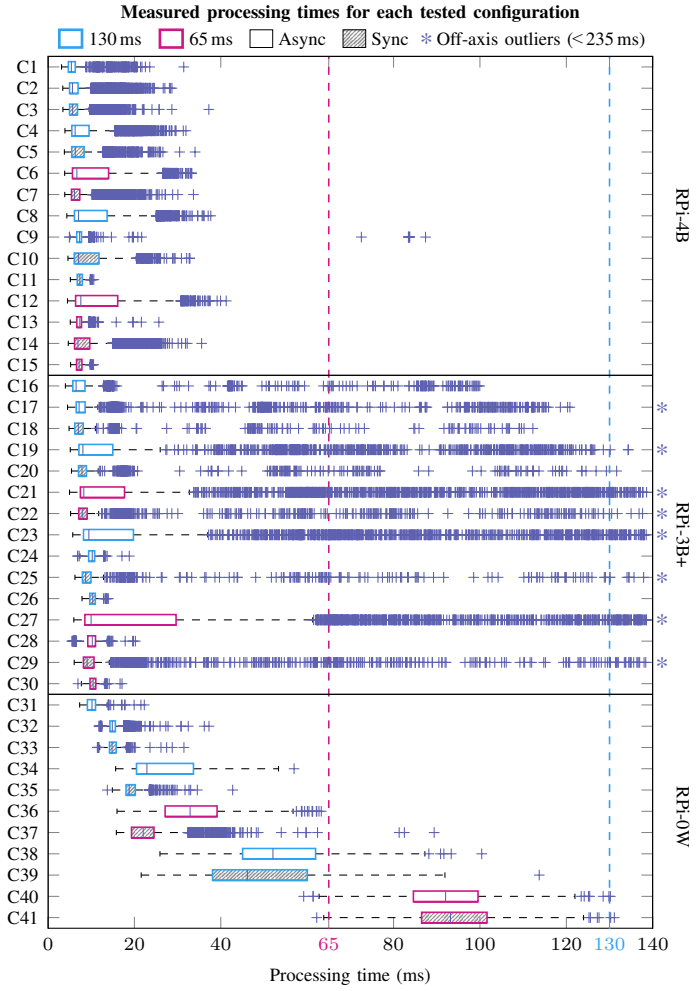


Fig. 3. Obtained results for each tested configuration. The table on the left reports the CPU and RAM usage, the percentages of lost packets, and those for which the processing times exceeded our constraint (i.e., ATC window). The boxplots on the right show the measured processing latencies as described in Section IV.

it for the sake of completeness and to understand its effect on the system. In particular, the tests without GUI were carried out by communicating with the system via the command line, and so having one less thread since ATC and FES profiles were not plotted.

Each test included 3 min of continuous operation, with CPU and RAM usage continuously monitored.

## V. Results and Discussion

This section presents the characterization of the system across the multiple tested configurations, whose results are reported in Fig. 3.

For both system implementations (i.e., the user-controlled and the stand-alone system), the increase of acquisition devices results in longer processing times. This is mainly due to two factors: each input device is managed by a thread, so the number of active threads increases; the command packet size is proportional to the number of initialized stimulation channels, thus requiring more time to build the packet.

Enabling the synchronization among input channels relaxes the processing phase, as the control platform needs to build the command packet fewer times. However, since waiting for data from all queues may take longer, the probability of queued data piling during subsequent processing steps increases: in these cases, only the most recent data are processed (for each input device), while older data are discarded. The percentage of data lost, w.r.t. those streamed by the acquisition devices, is reported in the *Losses* column in the table on Fig. 3. As expected, this percentage gets higher when the number of input devices increases and the ATC window decreases.

The use of the halved window (i.e., 65 ms) results in more processing cycles per time unit, which also reduces the operating time before the next ATC value(s). The percentage of data that took longer than one ATC window to be processed is reported in the *Delays* column of Fig. 3.

The user-controlled implementation of the system features negligible losses and zero delays when RPi-4B is employed as the control platform. Combined with a maximum CPU

utilization of 18.93 % in the most critical configuration (i.e., 8 channels, asynchronous processing, and 65 ms window), our measurements confirm the RPi-4B as an ideal platform for this implementation. On the other hand, the use of the RPi-3B+ involves more losses and delays (as evidenced by the high presence of outliers in Fig. 3), which reflect in an increase of RAM utilization w.r.t. the RPi-4B. A reasonable explanation for this behavior is related to the less performant GPU available on the RPi-3B+, which does not fully support the graphical requirements of our application. Although its performance decrease, considering that the interquartile range of the processing times is always within the ATC window, the use of the RPi-3B+ is still a feasible solution for use-case a).

The stand-alone system implementation (i.e., based on the RPi-0W) exhibits good performances for all tested configurations, except those involving 8 channels with the halved window. In these cases, the percentage of discarded data exceeds 26 %, while more than 99 % of the processed data takes longer than 65 ms. Comparing to what was obtained with RPi-3B+ (when the GUI was not used), the amount of RAM utilization is the same (about 23 MB), while the percentage of CPU usage is much higher (92.8 % versus 10.55 % in the configuration with 8 channels, asynchronous processing, and 130 ms window). Despite the higher CPU usage, if the ATC window is not halved, all data are processed on time, thus allowing the implementation b) to be used even with 8 channels.

## VI. Conclusion and Future Perspectives

In this paper, we proposed and analyzed the ATC-controlled FES workflow on different RPi machines for two use-cases: a user-supervised version and a stand-alone one.

Since real-time operability is a fundamental prerogative of FES systems, the computational latencies of the software were examined, emphasizing the system scalability in terms of input devices and considering the usage of RPi hardware resources. As a perspective of future developments of the ATC technique, tests were performed on different acquisition windows. The obtained results confirmed the feasibility of using the RPi-3B+/4B as the control platform when the maximum number of active channels (i.e., eight) is selected, even with the reduced ATC window. With the same channels configuration, the RPi-0W, acting in the stand-alone application, is able to manage up to eight-channels only with the longer (common) window.

The outcomes of this study allowed us to take a further step towards adopting our system for rehabilitation purposes, promoting its use for daily life activities involving multiple muscle groups (e.g., human gait and reaching exercises), thanks to the verified real-time and multichannel operation.

Next steps will also include the development of an IoT solution to complete the remote control of the stand-alone system, making it totally suitable for telerehabilitation.

## References


[1] M. Peter, *Embedded System Design: Embedded Systems Foundations of Cyber-Physical Systems, and the Internet of Things*. Springer Nature, 2021.

[2] A. Mongardi, P. Motto Ros, F. Rossi, M. Ruo Roch, M. Martina, and D. Demarchi, "A low-power embedded system for real-time sEMG based event-driven gesture recognition," in *2019 26th IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, 2019, pp. 65–68.

[3] M. Capra, S. Sapienza, P. Motto Ros, A. Serrani, M. Martina, A. Puiatti, P. Bonato, and D. Demarchi, "Assessing the feasibility of augmenting fall detection systems by relying on UWB-based position tracking and a home robot," *Sensors*, vol. 20, no. 18, 2020. [Online]. Available: https://www.mdpi.com/1424-8220/20/18/5361

[4] X. Wang, M. Hersche, B. Tömekce, B. Kaya, M. Magno, and L. Benini, "An accurate EEGNet-based motor-imagery brain–computer interface for low-power edge computing," in *2020 IEEE International Symposium on Medical Measurements and Applications (MeMeA)*. IEEE, 2020, pp. 1–6.

[5] S. F. L. Romero and L. Serpa-Andrade, "Low-cost embedded system proposal for EMG signals recognition and classification using ARM microcontroller and a high-accuracy EMG acquisition system," in *International Conference on Applied Human Factors and Ergonomics*. Springer, 2020, pp. 422–428.

[6] G. Dere, "Biomedical applications with using embedded systems," in *Data Acquisition-Recent Advances and Applications in Biomedical Engineering*. IntechOpen, 2021.

[7] M. N. Sadiku, R. A. Jaiyesimi, J. B. Idehen, and S. M. Musa, *Emerging Technologies in Healthcare*. AuthorHouse, 2021.

[8] K. Masani and M. Popovic, *Functional Electrical Stimulation in Rehabilitation and Neurorehabilitation*, 01 2011, pp. 877–896.

[9] C. Marquez Chin and M. Popovic, "Functional electrical stimulation therapy for restoration of motor function after spinal cord injury and stroke: a review," *BioMedical Engineering OnLine*, vol. 19, 05 2020.

[10] A. L. Basith, A. Arifin, F. Arrofiqi, T. Watanabe, and M. Nuh, "Embedded fuzzy logic controller for functional electrical stimulation system," in *2016 International Seminar on Intelligent Technology and Its Applications (ISITIA)*, 2016, pp. 89–94.

[11] S. Marzetti, V. Gies, V. Barchasz, H. Barthelemy, H. Glotin, E. Kussener, and R. Vauche, "Embedded learning for smart functional electrical stimulation," in *2020 IEEE International Instrumentation and Measurement Technology Conference (I2MTC)*, 2020, pp. 1–6.

[12] Z. Bi, Y. Wang, H. Wang, Y. Zhou, C. Xie, L. Zhu, H. Wang, B. Wang, J. Huang, X. Lü, and Z. Wang, "Wearable EMG bridge — A multiple-gesture reconstruction system using electrical stimulation controlled by the volitional surface electromyogram of a healthy forearm," *IEEE Access*, vol. 8, pp. 137 330–137 341, 2020.

[13] F. Rossi, P. Motto Ros, R. M. Rosales, and D. Demarchi, "Embedded bio-mimetic system for functional electrical stimulation controlled by event-driven sEMG," *Sensors*, vol. 20, no. 5, p. 1535, Mar 2020. [Online]. Available: http://dx.doi.org/10.3390/s20051535

[14] R. Merletti and S. Muceli, "Tutorial. Surface EMG detection in space and time: Best practices," *Journal of Electromyography and Kinesiology*, vol. 49, p. 102363, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1050641119302536

[15] S. J. Johnston and S. J. Cox, "The Raspberry Pi: A technology disrupter, and the enabler of dreams," *Electronics*, vol. 6, no. 3, 2017. [Online]. Available: https://www.mdpi.com/2079-9292/6/3/51

[16] B. Sijobert, C. Azevedo, J. Pontier, S. Graf, and C. Fattal, "A sensor-based multichannel FES system to control knee joint and reduce stance phase asymmetry in post-stroke gait," *Sensors*, vol. 21, no. 6, 2021. [Online]. Available: https://www.mdpi.com/1424-8220/21/6/2134

[17] B. Sijobert, R. Le Guillou, C. Fattal, and C. Azevedo Coste, "FES-induced cycling in complete SCI: A simpler control method based on inertial sensors," *Sensors*, vol. 19, no. 19, 2019. [Online]. Available: https://www.mdpi.com/1424-8220/19/19/4268

[18] A. Bo, L. da Fonseca, J. Guimaraes, E. Fachin-Martins, M. Paredes, G. Brindeiro, A. de Sousa, M. Dorado, and F. Ramos, "Cycling with spinal cord injury: A novel system for cycling using electrical stimulation for individuals with paraplegia, and preparation for cybathlon 2016," *IEEE Robotics Automation Magazine*, vol. 24, no. 4, pp. 58–65, 2017.

[19] G. Ricarte, L. de Macĕdo Pinheiro, and A. P. Lanari Bó, "Intuitive and modular software architecture for functional electrical stimulation rehabilitation," in *2020 Latin American Robotics Symposium (LARS),*

*2020 Brazilian Symposium on Robotics (SBR) and 2020 Workshop on Robotics in Education (WRE)*, 2020, pp. 1–6.

[20] N. S. Dhillon, A. Sutandi, M. Vishwanath, M. M. Lim, H. Cao, and D. Si, "A Raspberry Pi-based traumatic brain injury detection system for single-channel electroencephalogram," *Sensors*, vol. 21, no. 8, 2021. [Online]. Available: https://www.mdpi.com/1424-8220/21/8/2779

[21] R. Bayareh, A. Vera, L. Leija, and J. Gutiérrez-Martínez, "Development of a thermographic image instrument using the Raspberry Pi embedded system for the study of the diabetic foot," in *2018 IEEE International Instrumentation and Measurement Technology Conference (I2MTC)*. IEEE, 2018, pp. 1–6.

[22] F. Stradolini, A. Tuoheti, P. Motto Ros, D. Demarchi, and S. Carrara, "Raspberry Pi based system for portable and simultaneous monitoring of anesthetics and therapeutic compounds," in *2017 New Generation of CAS (NGCAS)*, 2017, pp. 101–104.

[23] T. Shaown, I. Hasan, M. M. R. Mim, and M. S. Hossain, "IoT-based portable ECG monitoring system for smart healthcare," in *2019 1st International Conference on Advances in Science, Engineering and Robotics Technology (ICASERT)*. IEEE, 2019, pp. 1–5.

[24] V. V. Garbhapu and S. Gopalan, "IoT based low cost single sensor node remote health monitoring system," *Procedia computer science*, vol. 113, pp. 408–415, 2017.

[25] V. Pardeshi, S. Sagar, S. Murmurwar, and P. Hage, "Health monitoring systems using IoT and Raspberry Pi — a review," in *2017 international conference on innovative mechanisms for industry applications (ICIMIA)*. IEEE, 2017, pp. 134–137.

[26] H. Hamil, Z. Zidelmal, M. S. Azzaz, S. Sakhi, R. Kaibou, S. Djilali, and D. Ould Abdeslam, "Design of a secured telehealth system based on multiple biosignals diagnosis and classification for IoT application," *Expert Systems*, p. e12765, 2021.

[27] F. Rossi, P. Motto Ros, S. Cecchini, A. Crema, S. Micera, and D. Demarchi, "An event-driven closed-loop system for real-time FES control," in *2019 26th IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, 2019, pp. 867–870.

[28] A. Prestia, F. Rossi, A. Mongardi, P. Motto Ros, M. Ruo Roch, M. Martina, and D. Demarchi, "Motion analysis for experimental evaluation of an event-driven fes system," *IEEE Transactions on Biomedical Circuits and Systems*, pp. 1–1, 2021.

[29] F. Rossi, A. Mongardi, P. Motto Ros, M. Ruo Roch, M. Martina, and D. Demarchi, "Tutorial: A versatile bio-inspired system for processing and transmission of muscular information," *IEEE Sensors Journal*, vol. 21, no. 20, pp. 22 285–22 303, 2021.

[30] F. Rossi, P. Motto Ros, S. Sapienza, P. Bonato, E. Bizzi, and D. Demarchi, "Wireless low energy system architecture for event-driven surface electromyography," in *Applications in Electronics Pervading Industry, Environment and Society*, S. Saponara and A. De Gloria, Eds. Cham: Springer International Publishing, 2019, pp. 179–185.

[31] P. Motto Ros, A. Sanginario, M. Crepaldi, and D. Demarchi, "Quality-energy trade-off and bio-inspired electronic systems," in *2018 IEEE International Conference on the Science of Electrical Engineering in Israel (ICSEE)*, 2018, pp. 1–5.

[32] K. Shima and K. Shimatani, "A new approach to direct rehabilitation based on functional electrical stimulation and EMG classification," in *2016 International Symposium on Micro-NanoMechatronics and Human Science (MHS)*, 2016, pp. 1–6.

[33] HASOMED GmbH, *Operation Manual RehaStim 2, RehaMove 2*, 09 2012.

[34] R. K. Ege, *Programming in an Object-Oriented Environment*. Academic Press, 1992.

[35] R. Oshana and M. Kraeling, *Software engineering for embedded systems: Methods, practical techniques, and applications*. Newnes, 2019.

[36] Z. Li, D. Guiraud, D. Andreu, M. Benoussaad, C. Fattal, and M. Hayashibe, "Real-time estimation of FES-induced joint torque with evoked EMG," *Journal of neuroengineering and rehabilitation*, vol. 13, no. 1, pp. 1–11, 2016.

[37] I. Harvey. Bluepy. Accessed: April 11, 2022. [Online]. Available: http://ianharvey.github.io/bluepy-doc/

[38] B. Project. BlueZ. Accessed: April 11, 2022. [Online]. Available: http://www.bluez.org/

[39] B. Kuberski, *ScienceMode2, RehaStim2 Stimulation Device, Description and Protocol*, 12 2012.

[40] C. Liechti. Pyserial. Accessed: April 11, 2022. [Online]. Available: https://pyserial.readthedocs.io/en/latest/index.html

[41] K. Project. Kivy. Accessed: April 11, 2022. [Online]. Available: https://kivy.org/#home

[42] P. S. Foundation. Queue — A synchronized queue class. Accessed: April 11, 2022. [Online]. Available: https://docs.python.org/3/library/queue.html