CASE STUDY

Energy Storage

Battery monitoring system at the edge

dos Santos AAF[1, 2, 3] | Alonso RH[2] | Simplício RS[2] | Zuffo MK[1, 2]

[1]Department of Electrical Engineering, University of São Paulo, São Paulo, SP, 05508-010, Brazil

[2]Interdisciplinary Center in Interactive Technologies, University of São Paulo, São Paulo, SP, 05508-020, Brazil

[3]Centro de Ciência de dados (C²D), University of São Paulo, São Paulo, SP, 05508-010, Brazil

Correspondence
Adolpho Augusto Ferreira dos Santos, Department of Electrical Engineering, University of São Paulo, São Paulo, SP, 05508-010, Brazil
Email: adolphoafsantos@usp.br

Funding information
This work was financed in part by the *Coordenação de Aperfeiçoamento de Pessoal de Nível Superior* (CAPES Finance Code 001), Brazil, and and by *Itaú Unibanco S.A.* through the *Programa de Bolsas Itaú* (PBI) program of the *Centro de Ciência de Dados* (C^2D) of *Escola Politécnica* of USP.

This paper studies how an edge recurrent neural network can improve a Battery Monitoring System (BMS). The proposed monitoring measures current, voltage, and temperature and infers the State of Charge (SoC) and State of Health (SoH) values through machine learning in an embedded system. The study relies on two test cases: a theoretical one using NASA's battery dataset, where the high volume of data is best suited for a study, and a second one with a system built in the University of São Paulo where this paper can analyze more profound the practical results of its use. This system of the second test case consists of peripheral sensors integrated into an Internet of Things (IoT) platform, sending the data collected from a VRLA battery to a Single Board Computer (SBC) via Bluetooth Low Energy (BLE). The edge SBC concentrates this received data - from one or more IoT nodes - generating new data for supervision and enabling control. The SBC communicates with a Web server in a one-way route to send the battery data without any data request. The algorithm developed for the SoC uses a dense recursive network with Mean Absolute Error (MAE) up to 0.2, and for SoC above 10%, the Mean Absolute Percentage Error (MAPE) can reach 0.16%. As an SoH calculation method, the battery replacement performs the methodology when the capacity reaches 80% of its nominal capacity. It is essential to highlight that these results are from a devices with limited hardware availability without cloud communication.

KEYWORDS
Battery monitoring, Battery Management System, Machine learning, Internet of Things, State of Charge, 4.0 Substation

1 | INTRODUCTION

Battery banks are essential in several applications. It can be used as a primary energy source in isolated applications or as a secondary source of energy - backup - in case of failures in the electrical grid. The availability of energy by battery banks directly depends on the quality and integrity of its elements, and, in case of failure, the energy supply would be interrupted. For that reason, monitoring these battery banks is extremely important for their continuous operation.

In the Brazilian scenario where this research has taken place, due to the difficulty of providing Internet connection in some hard-to-reach places in Brazil, such as isolated electrical substations, and the necessity of a high accuracy level of Battery Monitoring System (BMS), this system should not rely on this type of connection. Nevertheless, this scenario should not be considered a limitation but an opportunity to develop a system that can solve a broader spectrum of BMS use cases. Thus, this paper seeks a system that does not rely on an Internet connection for a proper function, while the accuracy of a machine learning algorithm is more than welcome.

The BMS consists of sensors that measure the magnitudes of the battery, especially current, voltage, and temperature [1], transmitting them to a concentrator unit that will work these data. The concentrator unit operates this data to obtain two other critical measurements: state of charge (SoC) and state of health (SoH), which cannot be directly measured. Although this is the most usual BMS formation, there are different ways to do it, and many other variables can be monitored.

SoC, as [1] defines it, is a unitless measurement ranging from 0 to 1 - or more commonly in percent - denoting a cell's residual energy. The measure dictates that the total possible amount of energy the battery can provide corresponds to 1, or 100%, while the minimum amount of energy the battery can provide is 0 or 0%. Thus, the SoC is defined as the percentage of energy that the battery has and can provide.

Another measure addressed in this system is the SoH. According to the battery usage, it will degrade, causing an internal resistance increase and thus decreasing the amount of charge the battery can provide. In valve-regulated lead acid (VRLA) batteries, this degradation over time has an almost linear degradation until it reaches the maximum load of 80%

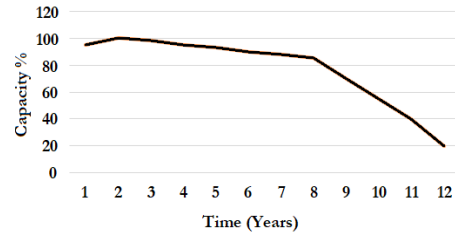

FIGURE 1 VRLA SoC over time [2]

of the nominal capacity. After this range, the degradation stops to have a linear pattern. The degradation becomes unpredictable and with higher degradation over time, as shown in Fig. 1. This pattern is why changing the battery when it reaches this mark is instructed.

One crucial point to notice is that there are many ways to infer the SoC value, and it can be classified as direct, model-based, learning algorithm, and hybrid methods, as seen in [3]. This paper considers machine learning one of the most efficient ways, as seen in [4], and proposes edge machine learning as a future technology to be explored. The machine learning approach is even more attractive as the technology advances, and together, the processing capacity at the edge, as [5] also states. This paper uses machine learning due to its simplicity, accuracy, and capacity to achieve great results, but this has to add the possibility of the machine learning running in low-power consumption devices without internet connections, hence the edge machine learning.

One crucial point in designing the BMS architecture is to use the top-of-the-notch architecture that can leverage the system. Nowadays, Internet of Things (IoT) devices provide that. This concept can counterattack the unpredictability in the final architecture of the battery system to be analyzed, leveraging the scalability. This study will analyze if the use case of IoT devices can effectively supply the needs for a BMS even if running an edge machine learning or if it is necessary to downgrade its potential without cloud computing. Also, this study will analyze what kind of data is necessary for better accuracy in BMS and what the data can teach us for a better fit between data, devices, and technologies to reach a better BMS.

In order to accomplish that, this paper has two main goals: propose the next-generation BMS architecture using technolo-

gies already available but studying for a better fit in devices and technology for the BMS, possibly having a new configuration, and study the machine learning approach for the SoC inference in this new scenario, pointing out what the data can teach us, what this technology can support, and the future holds for this configuration.

The contributions of this article are listed below:

- BMS model with machine learning on edge, producing scalability without using the cloud and without inherent performance loss
- Architecture that can leverage edge machine learning for BMS
- Demonstration of the ability to implement a complete BMS using IoT as an edge device

The article follows with the related works section presenting an overview of the BMS development. Subsequently, the paper presents the IoT device and software architecture. After that, it explains the network training, followed by validation and testing, demonstrating the experiments. Finally, the results are analyzed, and the conclusion discusses the results achieved with this work.

2 | RELATED WORK

The main problem with measuring the SoC is that it cannot be acquired directly, as voltage can, so it is necessary to infer the value. This necessity occurs because the value possible to acquire refers to the energy flowing out of the battery, but the SoC measures the percentage of energy in a battery. Knowing this, SoC still has other obstacles besides knowing the maximum and minimum value of the amount of charge that the battery can concentrate, such as the fact that the internal resistance of the battery has a direct correspondence with the SoC.

The correspondence with the internal resistance brings another challenge because, as [1] also points out, the value can not be accurately acquired when measured with the load connected, which is the case for many use cases. Besides disconnecting the battery, it needs a resting time - usually indicated as half an hour - to measure accurately.

Such a configuration is infeasible for an accurate real-time monitoring system, which is one reason to seek other methods of inferring SoC without using the battery's internal resistance.

Knowing these challenges, it is important to learn what are the currently existing methods to acquire the SoC, which are:

- direct methods such as counting coulombs and open-circuit voltage
- model-based method as extended kalman-filters
- learning algorithm as feed-foward neural networks, recurrent neural networks and deep learning algorithms
- hybrid methods, where it is a combination of the others, as a kalman-filter with a back-propagation algorithm

However, it is essential to understand that there are other methods, such as the model-based extended kalman-filter that is highly used as in [6], and [7], and the hybrid methods with filters and techniques that can increase the accuracy for a system, as in [8], this article focuses in the learning algorithms.

The computational model using machine learning carries significant benefits for SoC inference, as it does not require parameterizing the battery as to the state of minimum and maximum charge, despite being a point of interest to build more accurate algorithms. In addition, it is unnecessary to know the technical data of the battery, as in the case of direct models as the equivalent models and even the counting coulombs.

Despite not using all the elements presented in this energy conversion of the battery and the approximations used for the optimization of the model, it reaches more accuracy in constructing the mathematical model. This means that this model is also essential because, as mentioned, it does not need to calculate the internal resistance or model it, being a value that the machine learning itself can infer without calculating it correctly. That is suitable because, as [9] states, it is impossible to find the correspondence.

Within this machine learning model, we can also highlight that besides the use of different algorithms, as [10] points out, with neural networks [11], recurrent neural networks [8], and [12], up to more powerful models like Deep Learning [3] and Deep Reinforcement Learning [13], we also have the variation

of the hardware used, and it is architecture.

One can classify into three major groups: the methods using low computational power in cheaper and more devices with limited hardware availability as in [14], [15], and [16], usually using algebraic models or even simpler machine learning models such as linear regression. A second group uses heavy computational models, like Reinforcement Neural Network (RNN) up to XGboost, but being used in personal computers, like em [6], [17], [18], [19] and [20], which also directs the thought to the higher power consumption. Moreover, a third group with the use of these same heavier algorithms on intermediate hardware, as the IoT used here, but which makes use of cloud computing to make the inference, as in [7] and [13], which also does not satisfy the restricted application environment of this paper.

This paper introduces a model that does not fit into these large groups, with a heavier computational model - in this case, the RNN - used on intermediate hardware but without the need for cloud computing. It is worth mentioning that [11] has already done the technique of using a microcontroller with pre-trained neural network weights. However, the paper does not present temperature input with valuable information, does not use a recursive neural network that can leverage the system besides a heavier computational algorithm, and presents errors between 1.59% and 5.847%. A critical point here is that the methods using RNNs and their variations present better results, as pointed out in [8], [10], and [17], and the training method used can be of great value. Table 1 briefly summarises some of the papers analyzed.

For the SoH measurement, this paper, as well as [6] and [24], aims to find the moment when the capacity is reduced to 80% of the nominal value as the point to change the battery, considering the end of its useful life. Although this method has good performance with little use of computational resources, it is essential to note that other methods exist and are improving, as [5] points out, with complex methods, as in [13] employing genetic algorithms.

The analyses of the related works solutions identified several possible improvement points in the monitoring algorithms. Among the articles that do not require the battery to be disconnected or to remain at rest are those using machine learning. Many machine learning solutions require high computational power or cloud computing. Some methods still disregard the variables for use in a real-case application and do not use temperature as input data or only a fixed temperature, reducing their accuracy. Other methods still need to contemplate scalability for different application architectures or to couple BMS functions such as maintenance. Thus, this proposed algorithm based on the IoT edge architecture using edge RNN aims to end these problems and also brings an essential point to the discussion: the energy consumption and the price cost of the system.

This paper accomplishes these results because the architecture does not consume high energy or computational power and does not require Internet connections. Also, this work considers temperature as a factor that influences the battery's internal resistance, which can be quickly escalated by coupling more node devices or functions to the edge device. There is still room for improvement for a complete BMS. All of that is still trying to minimize the system's energy consumption.

3 | ARCHITECTURE

This section will present the architecture of the BMS for the second test case, developed at the University of São Paulo. To minimize energy consumption and increase the project's scalability, it uses an edge system with low-power IoT devices that, as [26] and [23] explain, this system can provide this efficiently. Firstly, to understand the installed system, it is essential to know which IoT devices this research uses.

3.1 | Edge Device

This study uses the Labrador as the edge device, an IoT platform developed at the University of São Paulo. This Single Board Computer has a quad-core of 64 bits, 16 Gigabytes of memory, 2 Gigabytes of RAM, and consumes 3 Watts [27].

3.2 | Node Device

This research uses the Pulga module as the node device. This IoT has 1 megabyte of memory, 256 kilobytes of RAM, and consumes 0.6 Watts with energy harvest capability, all in a round shape of 24.26 millimeters in diameter [28].

Once the limitation of the hardware availability issue is

TABLE 1 Literature review

Author	Comments
[21]	SoC in Electric model. BMS is in DQN without edge-node
[22]	SoC in Extreme Learning Machine using Cloud computing
[23]	It is a BMS in an IoT with a edge-node
[11]	IoT with SoC using NN for battery charge without temperature
[13]	IoT with SoC in H-infinity, and has SoH using Cloud Computing
[24]	Uses the 80% technique for SoH
[17]	SoC in Recurrent Gaussian Process Regression, SoH, not in IoT
[8]	It has SoC with KF-RNN and uses only at 25ºC
[7]	IoT with SoC using Extended-KF, SoH uses clouding computing
[16]	IoT with SoC using SVR-LR and do not compare with RNN
[15]	IoT using Zigbee for ES monitoring
[3]	Uses Deep Learning and has state-of-art accuracy
[19]	Uses a host computer
[20]	Uses a host computer with values little higher than the best ones
[25]	Join the KF with neural network and uses only 25ºC

clarified, we can move to the architecture. For a better understanding, two parts divide the architecture. The first part comprehends the communication architecture, where it will be shown at a higher level how the IoTs transfer the data. The second part is the software architecture, which shows how the software processes the data, the machine learning used, and its configuration. Thus, the following two subsections will present communication architecture and software architecture.

3.3 | Communication Architecture

Fig.2 illustrates the high-level operation of this system. It displays the system where the node device is attached to the battery and measures current, temperature, and voltage. Then, it sends these measurements via Bluetooth Low Energy (BLE) along with a time sample to the edge device that acts as a concentrator of the information and as the edge device. Then, the node IoT sleeps until the following readings. A crucial benefit of BLE is that it does not require new requests between devices for data exchange; the devices are on standby, waiting for new communication. Another significant benefit is that BLE consumes less energy than Bluetooth communication.

Once the data is in the edge device, it must look for missing data cases, considering both data unsent - resulting in missing data - and data with values that do not correspond to actual readings - with values above or below the possible ones - remembering that short circuit values, for example, may be atypical but are possible. Then, using the treated data, it calculates other valuable ones, power and energy. After that, the edge device can infer the SoC value from a pre-trained RNN machine learning algorithm. Using the SoC as a base, the edge device can calculate SoH, and the edge device performs power management.

The relevant data from this battery monitoring can be sent one-way to a web server via the Internet. Then, it redirects to an application under development, where the data flow is only for supervision but does not allow information to be inserted into the manager through the application for security reasons. This method ensures that the data used for managing the battery is from the BMS calculation and is not fake. However, it still allows the notification of the battery conditions outside this secure location.

3.4 | Software Architecture

Two functions divide the software architecture, the SoC inference and SoH calculation, as Fig. 3 represents. This figure helps to understand that machine learning is used in this two-

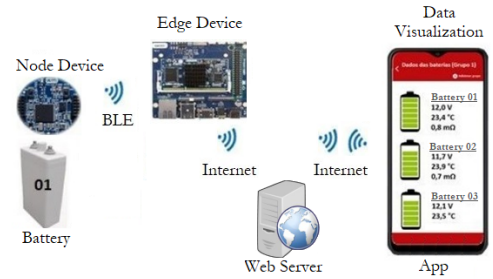

FIGURE 2 Hardware architecture

layer configuration to infer the SoC, with 20 neurons in the first one and 5 in the second. One important thing to notice is that this figure illustrates the software for the BMS installed at the University of São Paulo. The configuration for the study using the NASA dataset [29] - the first test case - does not have this 20-5 configuration. It will be adequately addressed in the next section and does not appear in this figure for an easier understanding, but as an earlier mention, the study using the [29] dataset has a three-layer configuration using 500-50-10.

The Fig. 3 explains the architecture of the second test case. The first test case is interely exploring and analyzing the [29] dataset in a Google Colab environment. Thus, the first test case comprehends only the assigned Google Colab part. The following section explains the configuration and machine learning training in the first and second test cases.

4 | MACHINE LEARNING TRAINING

The flowchart in Fig. 4 shows the top-down machine learning process from the data to the software deployment. This flowchart demonstrates all the steps taken and includes more details of the process from left to right. This figure is specific for the second use case, where software deployment is required, though the steps in Google Colab are the same for the first test. In the first test, the software loads the dataset in Google Colab and processes the data by reading, standardizing, and creating the variables. After that, it builds the machine learning, and it is possible to analyze the result and, if necessary, build a new version of the machine learning. This process occurs a few times during a validation, trying to reach a better result.

Now, we can analyze the software for the second test case, which uses the system at the University of São Paulo. It begins with acquiring the dataset, where all the voltage, current, temperature, and time readings from the sensors are summarized. After that, the data processing begins, consisting of three steps: dataset reading, standardization, and the variable creation. The created variables were power and energy, combining some previously acquired values, as seen in the equations 1 and 2. After that, the machine learning training begins. The machine learning training comprehends training, testing, and validation. After these three steps, the test and validation performances of the machine learning models are compared. If it is the best model, it is recorded, and when the model is good enough, the Mean Absolute Percentage Error (MAPE) is compared with the state-of-the-art benchmark. The machine learning training restarts if any of the tests do not produce a good enough model.

All these steps are the same as in the first case, as mentioned. After achieving a well-suited model, the Python software is exported, with the data processing and pre-trained machine learning model that infers the SoC. This software is deployed to the edge and evaluates the time consumed and if all the weights behave as expected. Here, we have a specific situation, analyzing the behavior in a real case scenario.

$$P = U * I \tag{1}$$

$$E = \frac{P * t}{1000} \tag{2}$$

The equation 1 represents the Power equation used in this paper, where the power P in Watts is given by the voltage U in Volts times current I in Ampere. In the 2 equation is presented the Energy equation, where the energy E in kWh is calculated by power times time t in hours and divided by a thousand.

The software for SoC and SoH calculation has been developed using python and the Google Colab platform; however - as the edge device is the final destination - the software was deployed in python and executed in the edge device for practical results. The test software assumes that the edge device

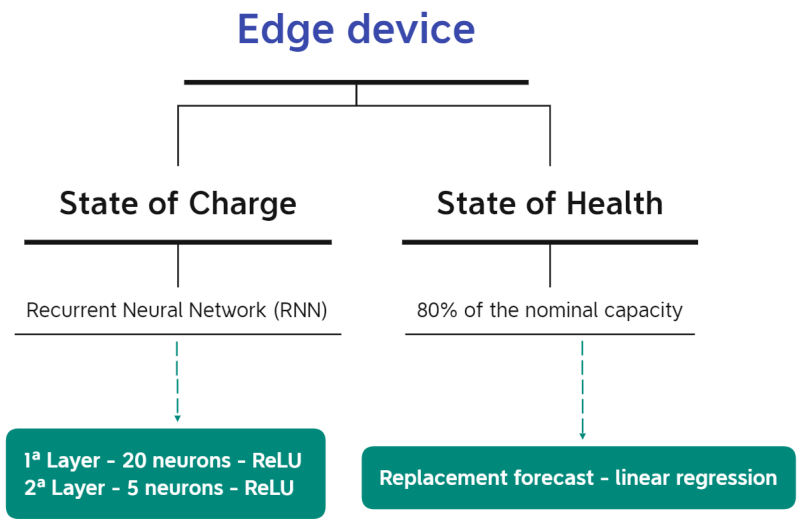

FIGURE 3 Software architecture

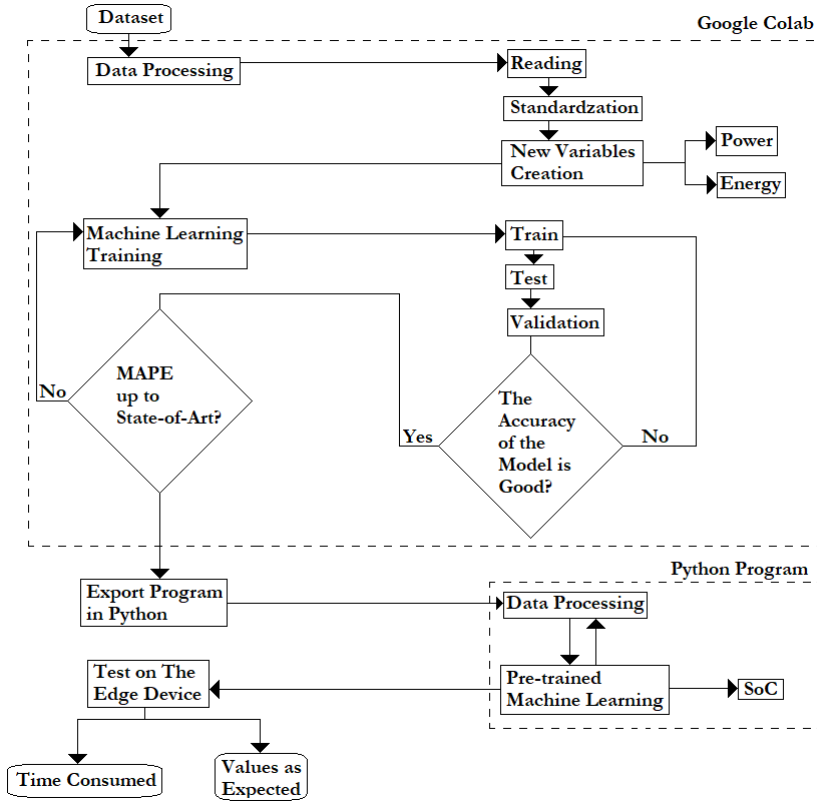

FIGURE 4 Software flowchart

receives the data by BLE, so practical tests of the connection between the node and edge devices were performed, as mentioned in Fig. 5. The BLE test software was made in python and reproduced in the edge device, validating that the communication of readings goes as it should. Also, the software needs a missing data search function that is not included at this phase of the project because, as the tests were performed on a known database without missing data, it was not required to deal with missing data in the software at this moment.

4.1 | Data Processing

TABLE 2 VRLA dataset

t (s)	Vol (V)	C (A)	Cap (Ah)	P (W)	E (Wh)
0	2.04	25	0	51.10	0
1	2.04	25	0	51.08	0
3	2.04	25	0	51.05	0
4	2.04	25	0	51.01	0.1
6	2.04	25	0	50.98	0.1
8	2.04	25	0.1	50.94	0.1
10	2.04	25	0.1	50.90	0.1
12	2.04	25	0.1	50.87	0.2
14	2.03	25	0.1	50.83	0.2
17	2.03	25	0.1	50.77	0.2

As Fig. 4 shows, the first step is data processing. It is noteworthy in table 2 that there are just a few data currently available for the battery used, which is a problem in this principle, causing restrictions in the treatment of the data and, consequently, in the software's performance. For each temperature value, there are about 75 occurrences. As the amount of acquired data grows, it can improve the model. The result analysis will address this point.

One crucial point to notice is that the variables used for the test were slightly modified, which makes the performance worse. Another essential detail is the presence of power and energy. However, this is fine here, as the software in the second test case creates both variables as soon as it receives the data from the readings. It is also possible to notice that the current has a constant value. It is an important variable and demands to be appropriately handled in applications that do not use constant current discharge. Although the table 2 does not show the temperature's information, it is present as each temperature has its dataset. Each dataset is separated by temperature: -30,-20, -10, 0, 10, 25, and 40 ºC.

TABLE 3 Inputs from the datasets

	Collected	Training data	Wanted
Time	X		X
Voltage	X	X	X
Current	X		X
Temperature	X	X	X
Power		X	X
Energy		X	X
Capacity		X	X
New_capacity			X

This first part of the program, data processing, is also one of the most important. In this part, the program must manipulate the data to leverage the dataset's best characteristics concerning its target. The first step is to choose the variables that will be used and standardized. In table 3, we can see the collected variables. Time had to be excluded to avoid overfitting, as the machine learning program could learn the time and voltage relationship to estimate the capacity and perform inappropriately with new data. The current has an almost constant value due to the characteristic wanted for the tests made, that is the same in the first test that the dataset has an almost constant current discharge characteristic as well. The current adds no new information to the machine learning training and has no need in those use case. It is essential to address that because if the current is variable, it has to be handled and has critical value to the model.

The second step is to create the power variables and energy in Watts and Watts-hour. As the database already has these data, we proceed to the next step for training, but it is important to note here that the four initial inputs created two more. After creating these data, the min-max equation standardized the voltage and power features. Below is represented the min-max equation.

$$X_{sc} = \frac{X - X_{min}}{X_{max} - X_{min}} \tag{3}$$

4.2 | Training

For training, it is crucial to remember that the database is restricted. Thus, in order to not allow overfitting, where the model created by the machine learning algorithm learns the data pattern but is not able to generalize and perform well with new data, and to search for results from temperatures not yet seen, the -20ºC temperature dataset was reserved as a test and the 10ºC temperature dataset as validation. TensorFlow Keras API was used to train the network. Variations were made in the number of epochs, neurons, and seeds, seeking better values; however, there are possibly even better results with other combinations. Another critical factor is that this combination will not be the best with another data arrangement.

As pointed out in Fig. 3, this program consists of a dense RNN with two layers, twenty neurons in the first and only five in the second, and a third layer with an output neuron, all with activation by ReLU. The supervised learning algorithm was chosen because this paper has data regarding input and output states; therefore, it is best suited to supervised learning. Previous studies demonstrated that recurrent networks produced excellent results for SoC inference within these algorithms and pointed out to be better researched by [10]. This project evaluated the cost-benefit of other simpler algorithms before using the RNN. Although the recurrent networks do not present such expressive gains when there is a more refined feature engineering process, a more complex and precise system was chosen because of a well-thought architecture with simple hardware and good computational capacity.

4.3 | State of Charge

With the network trained, the regressor created can calculate the SoC. The output value will be the capacity in ampere-hours (Ah), and the created min-max formula can transform the capacity into SoC, substituting the minimum value for 0 and the maximum for the current maximum capacity for the corresponding temperature as 1.

Two details here are important. If the battery exceeds the maximum capacity value for the temperature, the SoC displayed must equal 1, and the same is true for negative SoC with a default value of 0. Also, as the battery degrades, the capacity decreases, and the maximum capacity needs to be updated to the current value of the corresponding battery and reset when the battery is changed. It is crucial to note that the data in those datasets does not contemplate this capacity variation over time, so these two systems are not implemented with new data in the experimental testing phase. The SoC can be calculated as follows:

$$SoC = \frac{Q(t)}{Q_{total}} \tag{4}$$

$$z(t) = z(0) - \frac{1}{Q}\int_0^t \eta i(\tau)\, d\tau \tag{5}$$

Equation 4, is presented as the SoC calculation. The Q is the capacity at time t, and Q_{total} is the cell total capacity in ampere seconds (coulombs) between 0% and 100%. In equation 5, cell current is positive on discharge and negative on charge. The η is cell coulombic efficiency nearly 1, but always less than 1. This estimation is called “coulomb counting.” As we can see, it does not use the temperature, although the temperature enunciates the SoC.

4.4 | State of Health

As explained earlier, the method used to calculate the SoH is to know when the battery’s capacity reduces to 80% of its nominal capacity. It is important to note that, due to the limited amount of data in the second case that does not allow us to trace a usage pattern to estimate its life, this test was not performed; only the algorithm was tested.

Therefore, the equation below infers the SoH and if it is necessary to replace the battery. The nominal capacity is known; for example, the VRLA used in the second case is 300 Ah. After using the regressor, the output is the capacity value in Ah. As such, by doing the same min-max equation of the battery capacity when the battery’s SoC is at 1, it can get the value of SoH. The min-max equation corresponding to SoH is shown below. In this formula, C corresponds to the capacity

displayed by the regressor, C_{nom} corresponds to the nominal capacity, and if the SoH value is below 0.8, it indicates the exchange.

$$SoH = \frac{C}{C_{nom}} \tag{6}$$

Knowing how the project of this paper performed the training, it is interesting to show the differences between this and another application, where predicting when the battery replacement may be required. For that, it is necessary to use a new capacity variable. This variable corresponds to the new SoC value equal to 1 for the battery in use, as this battery can still hold a SoC at one even without reaching the same 300 Ah again. This way, the new capacity variable calculates the battery capacity drop over time, comparing the percentage value that the new capacity holds against the nominal capacity. Suppose this approach constantly updates the prediction of the new battery capacity for different temperatures. In that case, it can make a linear regression with temperature, time, and this new capacity input.

4.5 | Data Preservation

Once all this data is generated, it is coupled to an internal edge device database so it can easily export when required, and a second backup and monitoring can be placed in the cloud; for the test case, both kinds of backup were implemented. Another important use of this data history is the use in the further tuning of the RNN weights by a development team, something that will also not be a burden due to the ease in this re-training and in the time consumed, which is low - about few minutes to train the new neural network.

5 | TESTS AND VALIDATION

This study has three tests to ensure the expected outcome and validate the software, as shown in Fig. 5. The first test evaluates the communication between the edge and the node device. This test uses software deployed in python, where it is possible to pair via BLE the edge device to a node device in a notification configuration.

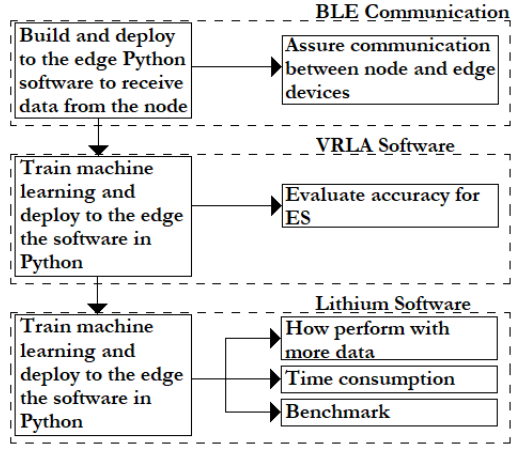

FIGURE 5 Flowchart test's goals

After that, this paper analyzed the accuracy performance of the system in a VRLA battery and the time consumption, studying the feasibility of building the system. In this stage, a machine learning algorithm in Google Colab was created with high precision, and python software was deployed with the trained machine learning to an edge device, evaluating the time consumption and testing the values to ensure the calculations of the SoC had the expected outcome. The Fig. 4 shows the second stage.

The third stage of this study is to evaluate the feasibility of this system in the future and build a benchmark of the architecture. For that, the same structure seen in Fig. 4 is repeated but now using a lithium battery dataset. This study uses this approach because using this larger dataset can assure that the accuracy will grow as the dataset of the VRLA battery grows, so as this happens, the machine learning algorithm will grow and still perform well. The second important point is to evaluate if the time consumption in the edge device is still lower than 1 minute, the time considered in this study to allow the system to work without causing further problems in synchronizing the readings and monitoring. The last crucial point is that a lithium dataset well publicized as [29] can provide a benchmark to evaluate whether this architecture performs well considering other SoC studies.

One way to look at this lithium test is to visualize the performance of the VRLA in the future, besides evidence of the performance to a larger broad of use cases. To conclude,

this study is built on the SoH calculation to find when the battery's maximum capacity will be reduced to 80% of the nominal capacity. The batteries used for the VRLA test are the battery 5 OPzV 250 [30]. Based on these batteries, was generated the dataset [31], which was used in the VRLA test.

5.1 | Bluetooth Low Energy validation

To start the validation test, as the program assumes that the edge device is receiving the reading values from the node IoT, this must be validated first. So, a Python program was built directly on the edge device. In this program, the edge device turns on BLE, searches for nodes, and connects to the selected device. The node selected - through advertising -sends data to paired devices. The edge device then receives this signal, converts it to data, and prints it already in data format, validating the communication with the Pulga that made the advertisement via BLE.

5.2 | State of Charge validation

After the BLE validation, it is necessary to validate the SoC software. An important point here is that, for the data to be better compared with other research papers, it is necessary to perform the same training procedures with another database where the number of readings and the variable dimensions are the same. Therefore, the answers and metrics will be analyzed using NASA's "BatteryAgingARC" database [29]. This database enables this paper to compare the procedures and results with research using lithium batteries, which have more papers and data available.

As table 4 shows, the database has the values of voltage measured at the terminal, current measured at the terminal, temperature, voltage measured at the load, the current measured at the load, and time, respectively. The data processing began by converting the time to time in seconds, starting at 0. After that, the software creates the capacity variable by multiplying the average current between the current at the time "t" and the current at the time "t-1" by the time in hours between the same two values. The value of this average capacity is then added to the previous capacity, thus completing the capacity at the current time. Below is shown this equation. After that, the software calculates the energy and power variables. For training, the features are voltage measured at the terminal, current measured at the terminal, temperature, time, energy, and power, and the target is the capacity.

TABLE 4 Battery Aging Ames Research Center dataset

Vol (V)	C (A)	T (°C)	C_load (A)	V_load (V)	t (s)
3.873	-0.001	24.655	0.00	3.00	0
3.479	-4.030	24.666	-4.03	1.57	2532
4.000	1.512	24.675	1.50	4.72	5.5
4.012	1.509	24.693	1.50	4.74	8344
4.019	1.511	24.705	1.50	4.75	11125
4.025	1.512	24.718	1.49	4.75	13891
4.030	1.511	24.731	1.50	4.76	16672
4.035	1.510	24.741	1.50	4.76	19.5
4.039	1.507	24.759	1.50	4.77	22282
4.043	1.507	24.766	1.50	4.77	25063

$$C_t = [\frac{(C_t + C_{t-1})}{2} \times \frac{(t_t + t_{t-1})}{2}] + C_{t-1} \quad (7)$$

The best network combination acquired for SoC was a dense RNN layer followed by three dense MLP layers and one output MLP. The RNN has 500 neurons, and the MLP networks have 500, 50, 10, and 1 neurons, respectively, all activated by ReLU. At the same time, the RNN for the VRLA is a 20-neuron RNN layer followed by a 5-neuron dense layer.

5.3 | Validation in the edge

As shown in Fig. 4, the VRLA battery software referenced in the training section validated the performance in the edge device. However, before exporting the program, still in Colab, a trained neural network model was generated and saved. This software was then changed to no longer read the database, model the data, train, and infer the SoC; now, it loads the neural network rather than train the model. This new software was then deployed in Python to edge devices with the trained neural network and the VRLA database.

This software was reproduced in the edge using the python software and, from the data processing to the inference of

the SoC, took 8 seconds. This time consumed is satisfactory, considering the interval between a series of readings of the node, its sleep, and waking up again for new readings is 5 minutes. This result validates the performance in the edge, where the time consumed and the correctness of the data were evaluated from end-to-end.

6 | RESULT ANALYSIS

As mentioned before, due to the small amount of data in the VRLA database, obtaining more concrete results for comparing the state-of-the-art systems is impossible, although as the VRLA battery has many usage applications nowadays, it is noteworthy. So, the system is compared in three ways: its performance for its application, taking into account its metrics; its performance concerning a more robust database, which was done in the validation; and its performance concerning state of the art. As metrics of state of the art, it was used reference taken from [4]. It is indispensable to point out that when analyzing the article [4], it was required to have a clear distinction between Mean Absolute Error (MAE) and MAPE, which could lead to a mistake in interpreting the results. The values were certified, and [4] obtained an excellent margin for state of the art with MAPE 0.3 to 1%, besides the static errors with some analyzed works. A comparison of this paper and the state of the art pointed out by [4] is seen at 5.

TABLE 5 Comparing Results

	MAPE
[29]**Lí-íon Test**	**0.5**
[4] Lí-íon Test	**0.3**
This article's VRLA Test [31]	**4.8**
This article's VRLA Test at SoC >10%	**0.18**

It is interesting to note the results here. The constructed regressor does not perform in its prime for the entire discharge cycle. This performance is because the database lacks representativeness for the model to learn the battery pattern at a small SoC. Some methods can be used to achieve better performance for battery SoC at low levels; however, the VRLA batteries are not meant to discharge by completion; this can damage the battery, making another compelling reason not to focus on this region of operation. The crucial thing is that the battery must always be ready for the moment it will act and should not be discharged at too low levels. This way, the high precision for higher SoC values is more interesting. For values above the critical state of SoC equal to 10%, the MAPE of the regressor is only 0.18, which qualifies this system as state-of-the-art for general cases.

For the model using the lithium-ion batteries database, which can expand for a more significant number of scenarios, the Mean Squared Error (MSE) was below 0.0001. The same tests were performed for this database but with a different amount of data. This test confirmed that the amount of data enhances the system prediction as it increases, a positive point for the primary VRLA battery model, which should present even better results after more data is collected.

7 | CONCLUSION

This paper presented a BMS with edge computing that can effectively infer the SoC using edge machine learning without performance loss. The proposed BMS collects the main battery parameters to infer the SoC and SoH values. It also calculates power and energy that can help a maintenance team regarding the battery's physical condition and indicate the best moment for maintenance or replacement. Besides, energy is a valuable feature for optimization of the SoC inference. This BMS also has high scalability - vertically and horizontally -, and allows adjusting the number of batteries and configurations in the bank. In order to minimize energy consumption and increase the project's scalability, it uses an edge system with low-power IoT devices that can provide this efficiently.

The presented system has state-of-the-art MAPE, especially considering the expected performance area for VRLA that can not be fully discharged. The software with the "Li-ion Battery Aging Datasets" database, where it was possible to obtain more research about SoC inference, still demonstrates that this method brings good answers. The system should have even better results as the amount of data grows, including regarding low values of SoC for VRLA. It is important to remember that the value of the state-of-the-art was analyzed considering IoT lithium-ion battery works but using the cloud,

which helps to create more complex algorithms capable of obtaining even better results, and even algorithms run directly on personal computers. Although those systems using a computer or mandatory internet access were not considered fit as a solution for this paper, they were considered to build a state-of-the-art system.

This BMS can solve broad use cases, not only for the most presented in a daily basis BMS applications but also in hard-to-reach places in Brazil, such as isolated electrical substation. This work made possible a system with the high precision of machine learning without adding significant cost or energy consumption, considering that the IoT edge device has 3 W consumption and the node has energy harvesting. This model still does not need cloud computing and adds an extra layer of protection against possible attacks. Thus, it can still maintain high precision in this process with the security of not having a system susceptible to remote invasion, something fundamental in a system that is essential to society.

It is worth mentioning that the SoC inference system requires more practical tests, being validated at first only with information from a database but not with actual data from other batteries together, where the noises should be different from the ones present in this database. Within the limitations, it is considered a promising project with remarkable results regarding the inferences for long-scale projects and reliability to many use cases, but still with great possibilities for improvement, as mentioned, the potential to couple prescriptive maintenance.

This paper does not address the prescriptive maintenance deeply due to the necessity of more tests in the field, as in the second test case of this paper, but it is safe to say that this BMS can support the allocation of this function. The tests performed for this conclusion used reinforcement learning in a Double Q-Learning algorithm in a MatLab environment deployed to the edge IoT in C++. This first validation test was more concerned with the ability to perform the function than the action performed by the reinforcement learning agent.

In the preliminary results, this paper presents a device that fulfills its function and will have more performance tests for a complete BMS using edge machine learning. This system can also improve smart grid applications and even mobile and personal computer SoC efficiency by its software thinking method, possibly with a complex pre-trained machine learning software deployment or a stacking method afterward for fine-tuning. This study comprises precise SoC and SoH systems on the edge and cloudless, supports complete BMS, saves a considerable amount of energy, does not require Internet for the main operation, has low cost for implementation, and is still capable of improvements as the amount of collected data advances. The next step is to evaluate the SoC under different practical values and test the reinforcement learning in the prescriptive maintenance function in the natural system at the University of São Paulo in a controlled environment, thus finalizing the BMS.

NOMENCLATURE

BMS	Battery Management System
SoC	State of Charge
SoH	State of Health
VRLA	Valve-Regulated Lead Acid
IoT	Internet of Things
RNN	Recurrent Neural Network
DQN	Double Q-Learning
NN	Neural Network
KF-RNN	Kalman Filter - Recurrent Neural Network
Extended-KF	Extended Kalman Filter
SVR-LR	Supporting Vector Machine - Linear Regressor
ES	Electrical Substation
KF	Kalman Filter
RAM	Random-Access Memory
BLE	Bluetooth Low Energy
NASA	National Aeronautics and Space Administration
MAPE	Mean Absolute Percent Error

E	Energy
P	Power
t	time
kWh	Kilowatt-hour
ReLU	Rectified Linear Unit
Ah	Ampere-hour
Q	Capacity
Q_{total}	Total Capacity
η	Cell Coulombic Efficiency
C	Capacity
C_{nom}	Nominal Capacity
Vol	Voltage
C	Current
T	Temperature

ACKNOWLEDGEMENTS

This work was financed in part by the *Coordenação de Aperfeiçoamento de Pessoal de Nível Superior* (CAPES Finance Code 001), Brazil, and and by *Itaú Unibanco S.A.* through the *Programa de Bolsas Itaú* (PBI) program of the *Centro de Ciência de Dados* (C^2D) of *Escola Politécnica* of USP.

ORCID

Adolpho Augusto Ferreira dos Santos
https://orcid.org/0000-0002-6225-9195
Rafael Herrero Alonso
https://orcid.org/0000-0001-5009-8147
Roberto Silva Simplício
https://orcid.org/0000-0001-7231-631X
Marcelo Knörich Zuffo
https://orcid.org/0000-0002-2973-4476

REFERENCES

[1] Plett GL. Battery management systems, Volume I: Battery modeling. Artech House; 2015.

[2] PowerShield, Monitoring VRLA State of Health; 2018. https://info.powershield.com/blog/monitoring-vrla-state-of-health.

[3] El Fallah S, Kharbach J, Hammouch Z, Rezzouk A, Jamil MO. State of charge estimation of an electric vehicle's battery using Deep Neural Networks: Simulation and experimental results. Journal of Energy Storage 2023;62:106904.

[4] Ali MU, Zafar A, Nengroo SH, Hussain S, Alvi MJ, Kim HJ. Towards a smarter battery management system for electric vehicle applications: A critical review of lithium-ion battery state of charge estimation. Energies 2019;12:446.

[5] Xiong R, Li L, Tian J. Towards a smarter battery management system: A critical review on battery state of health monitoring methods. Journal of Power Sources 2018;405:18–29.

[6] Shrivastava P, Soon TK, Idris MYIB, Mekhilef S, Adnan SBRS. Combined State of Charge and State of Energy Estimation of Lithium-Ion Battery Using Dual Forgetting Factor-Based Adaptive Extended Kalman Filter for Electric Vehicle Applications. IEEE Transactions on Vehicular Technology 2021;70:1200–1215.

[7] Adhikaree A, Kim T, Vagdoda J, Ochoa A, Hernandez PJ, Lee Y. Cloud-based battery condition monitoring platform for large-scale lithium-ion battery energy storage systems using Internet-of-Things (IoT). IEEE; 2017. p. 1004–1009.

[8] Zhao W, Kong X, Wang C. Combined estimation of the state of charge of a lithium battery based on a back-propagation–adaptive Kalman filter algorithm. Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering 2018;232:357–366.

[9] Busa B. Predictive Machine Learning Maintenance of VRLA Batteries based Uninterruptible Power Supply Systems 2021;.

[10] Cui Z, Wang L, Li Q, Wang K. A comprehensive review on the state of charge estimation for lithium-ion battery based on neural network. International Journal of Energy Research 2022;46:5423–5440.

[11] Trinandana GA, Pratama AW, Prasetyono E, Anggriawan DO. Real time state of charge estimation for lead acid battery using artificial neural network. IEEE; 2020. p. 363–368.

[12] Li S, Ju C, Li J, Fang R, Tao Z, Li B, et al. State-of-charge estimation of lithium-ion batteries in the battery degradation process based on recurrent neural network. Energies 2021;14(2):306.

[13] Li W, Rentemeister M, Badeda J, Jöst D, Schulte D, Sauer DU. Digital twin for battery systems: Cloud battery management system with online state-of-charge and state-of-health estimation. Journal of energy storage 2020;30:101557.

[14] Miao Z, Xu L, Disfani VR, Fan L. An SOC-based battery management system for microgrids. Ieee transactions on smart grid 2013;5:966–973.

[15] Firdaus, Aditya DO, Pramono WB. DC backup power supply monitoring in substation based on wireless sensor network; 2016. p. 97–100.

[16] Mousavi N, Aksanli B, Akyurek AS, Šimunič Rosing T. Accuracy-resource tradeoff for edge devices in Internet of Things; 2017. p. 581–586.

[17] Sahinoglu GO, Pajovic M, Sahinoglu Z, Wang Y, Orlik PV, Wada T. Battery State-of-Charge Estimation Based on Regular/Recurrent Gaussian Process Regression. IEEE Transactions on Industrial Electronics 2018;65:4311–4321.

[18] Zhang ZL, Cheng X, Lu ZY, Gu DJ. SOC Estimation of Lithium-Ion Batteries with AEKF and Wavelet Transform Matrix. IEEE Transactions on Power Electronics 2017 10;32:7626–7634.

[19] Chai H, Gao Z, Jiao Z, Yang C. State of charge estimation for lithium-ion batteries based on an adaptive fractional-order cubature Kalman filter with initial value compensation. Journal of Energy Storage 2023;68:107544.

[20] Xia L, Wang S, Yu C, Fan Y, Li B, Xie Y. Joint estimation of the state-of-energy and state-of-charge of lithium-ion batteries under a wide temperature range based on the fusion modeling and online parameter prediction. Journal of Energy Storage 2022;52:105010.

[21] Li W, Cui H, Nemeth T, Jansen J, Ünlübayir C, Wei Z, et al. Deep reinforcement learning-based energy management of hybrid battery systems in electric vehicles. Journal of Energy Storage 2021;36:102355. https://www.sciencedirect.com/science/article/pii/S2352152X21001158.

[22] Li S, Zhao P. Big data driven vehicle battery management method: A novel cyber-physical system perspective. Journal of Energy Storage 2021;33:102064. https://www.sciencedirect.com/science/article/pii/S2352152X20318971.

[23] Siva AS, Parthasarathy P, Gijipriya K, Vinothini N, Stonier AA. Anticipation of Battery Condition Using Internet of Things. IOP Conference Series: Materials Science and Engineering 2021 2;1055:012157.

[24] Zainuri A, Wibawa U, Rusli M, Hasanah RN, Harahap RA. VRLA battery state of health estimation based on charging time. TELKOMNIKA (Telecommunication Computing Electronics and Control) 2019;17:1577–1583.

[25] Wang C, Zhang X, Yun X, Fan X. A novel hybrid machine learning coulomb counting technique for state of charge estimation of lithium-ion batteries. Journal of Energy Storage 2023;63:107081.

[26] Costa LCP, Rabaey J, Wolisz A, Rosan M, Zuffo MK. Swarm os control plane: an architecture proposal for heterogeneous and organic networks. IEEE Transactions on Consumer Electronics 2015;61:454–462.

[27] Loucos C, Labrador 64 bits; 2023. https://caninosloucos.org/en/labrador-64-en/.

[28] Loucos C, Pulga Core V2.0; 2023. https://caninosloucos.org/en/pulgacore-v2-en/.

[29] Center NAR, Li-ion Battery Aging Datasets; 2023. https://data.nasa.gov/dataset/Li-ion-Battery-Aging-Datasets/uj5r-zjdb.

[30] EnerSys, Battery 5 OPzV 250; 2023. https://www.enersys.com/en/products/batteries/powersafe/powersafe-opzv/.

[31] dos Santos AAF, VRLA Datasets; 2023. https://github.com/Adolpho-Ferreira/Machine_learning_projects/tree/main/VRLA.

Adolpho Augusto Ferreira dos Santos

Master's degree student at the University of São Paulo with a scholarship from the Itaú Scholarship Program (PBI), linked to the Data Science Center (C2D). He received his Control and Automation Engineering degree from the Federal Institute of São Paulo in 2018, with a sandwich period at the Dublin Institute of Technology. His main research interest is machine learning, and he is currently researching and developing machine learning applications in batteries.

Marcelo Knörich Zuffo

Marcelo Knörich Zuffo (Member, IEEE) received the Electrical Engineering degree and the master's and Ph.D. degrees from the Universidade de São Paulo (USP), São Paulo, Brazil, in 1989, 1993, and 1997, respectively.,He is a Full Professor with the Escola Politécnica, Universi-

dade de São Paulo (USP), São Paulo, Brazil, where he is the Chair of the Interdisciplinary Center on Interactive Technologies and a Researcher with the Laboratory of Integrated Systems. He is currently the Coordinator of the Interdisciplinary Center on Interactive Technologies, USP, where he leads research and develops projects on consumer electronics, including virtual reality, digital image processing, digital health, multimedia hardware, embedded computing, and Internet of Things.,Prof. Zuffo is also a member of the Brazilian Computer Society, ACM SIGGRAPH, and Union of Engineers of the State of São Paulo and Brazil. He intensely contributed to the Brazilian Digital TV System, adopted by most countries in Latin America.

Rafael Herrero Alonso

Rafael Herrero is an Electrical Engineer graduated from FEI (2004), with a Master's (2009), Doctorate (2015) and Postdoctoral (2021) from Escola Politécnica-USP. With more than 10 years of experience in R&D projects, he was responsible for the development of Wi-Fi Solar (2009), participated in the Carport/PV project (500kW) in Parque-Público/SP (2012-2017) and was the research responsible for the design of the PV system, BESS and Solarimetric Station, integrated into a multidisciplinary monitoring platform at POLI/USP (2019-2021), for training students and training masters and doctors. Today he has been working on research on DERs, performance analysis of hybrid systems, SE4.0, computational modeling, IoT and AI.

Roberto Silva Simplício

Graduated in Electrical Engineering from the Polytechnic School of the University of São Paulo. Since then, he has been working in R&D projects at CITI/USP. His main research area is solar energy focused on agrivoltaic systems; PVSyst simulation; shading and soiling analysis; forecast of energy production; and economic feasibility.