HardwareX 12 (2022) e00340

Contents lists available at ScienceDirect

HardwareX

journal homepage: www.elsevier.com/locate/ohx

Hardware Article

Motrol: A hardware-software device for batch benchmarking and profiling of in-lab mobile device clusters

Juan Manuel Toloza *, Matías Hirsch, Cristian Mateos, Alejandro Zunino

ISISTAN-UNICEN-CONICET, Tandil, Argentina

ARTICLE INFO

Article history:
Received 7 April 2022
Received in revised form 13 July 2022
Accepted 17 July 2022

Keywords:
Android
Benchmarking
Dew computing
NodeMCU
Smartphones

ABSTRACT

Motrol is a simple device that satisfies functional requirements necessary for the automatic execution of battery-driven tests and profiling of connected mobile devices. It is specifically a hardware/software platform that allows Dew Computing researchers and developers to automate performance tests on Android-based smartphones. The hardware is based on a NodeMCU Esp8266 microcontroller that runs a firmware for managing the outputs. This software allows enabling/disabling the relays that connect the sockets that power the chargers of up to 4 mobile devices minimizing the need for human intervention. The firmware runs a web server that serves Rest requests from a Rest client with the commands to drive the digital outputs. These digital outputs activate or deactivate the relays to allow current to pass or not to the sockets. Such capability is essential to automate the study of battery behavior on battery-driven devices such as smartphones. Motrol is easy to assemble, knowledge in electronics or programming languages is not necessary, it is constructed with open hardware, and it is cheap, being its total cost ~USD 30.

Specifications table

Hardware name	Motrol (MObile ConTROL)
Subject area	• Educational tools and open source alternatives to existing infrastructure • General
Hardware type	• Electrical engineering and computer science
Closest commercial analog	ITEAD's Sonoff® DIY smart switches (e.g. Basic R2, Basic R2, RFR2, RFR3) (https://itead.cc/product-category/smart-home/diy-smart-switches) and specifically Basic R2 was the first device used for this purpose but it was not automatable at the time of designing the device (https://sonoff.tech).

(*continued on next page*)

* Corresponding author.
E-mail addresses: juanmanuel.toloza@isistan.unicen.edu.ar (J.M. Toloza), matias.hirsch@isistan.unicen.edu.ar (M. Hirsch), cristian.mateos@isistan.unicen.edu.ar (C. Mateos), alejandro.zunino@isistan.unicen.edu.ar (A. Zunino).

https://doi.org/10.1016/j.ohx.2022.e00340
2468-0672/

	Another close analog on the market is Yepkit USB Switchable Hub https://www.yepkit.com/products/ykush but it did not reach Argentina.
Open source license	CC-BY 4.0
Cost of hardware	∼USD 30
Source file repository	http://doi.org/10.17632/9ndz5b9ccv.3

Hardware in context

Smartphones industry permanently launch new models to the market improving hardware and software, which results in better performance and energy efficiency capabilities compared to previous models, and this motivates the need to keep fresh benchmarking and profiling data. Benchmarking means establishing a point of comparison between executing different variants of an application on the same device – e.g. by applying different code optimizations– or the same application on mobile devices with different characteristics [1]. Profiling on the other hand concerns tracking an application behavior to characterize its execution time or energy consumption [1]. Moreover, in a context of results validation, where testbeds integrated by several smartphones are used to compare the behavior and performance of several job scheduling algorithms, it is required a mechanism to recreate the same initial experimental conditions including battery level of each smartphone in the testbed. These testbeds are essentially clusters of smartphones at the edge, i.e. the base infrastructure enabling a high-performance computing paradigm proposed recently and termed Dew Computing [2]. Note that Edge Computing is an umbrella term that refers to any distributed computing paradigm where computations, originated at an end device D, are carried out not on distant Clouds but on devices and computational resources as close to D as possible. Besides Dew Computing, another prominent example is Fog Computing.

Solutions to perform battery-driven benchmarking and profiling of cell phones have been presented. As for commercial devices, Sonoff® can easily activate or deactivate an output but it has some disadvantages compared to Motrol:

- Sonoff® can only be controlled through a proprietary application and you must have an Internet connection,
- You must add a socket to Sonoff® so as to plug a cell phone using its original power cord and plug. Use of the original charger provided with the product is recommended by the manufacturer for safety reasons and this applies to smartphone chargers.
- Finally, the price of this device to manage 4 channels, in Argentina, is ∼USD 100.

Regarding another alternative named Yepkit USB, you need to run an application on a host to which it must be connected, and the application handles the task of turning the USB ports on and off. In this case, only the USB power cables can be connected to the cell phones without the possibility to use the original chargers. Although this solution is used to connect devices that transfer information, each port can be used as a charging port when powered. A drawback of Yepkit USB is that it is able to handle up to three ports and it costs barely more than Motrol (∼USD 40).

Motrol (short for MObile conTROL) is better for several reasons: first, no proprietary software is needed for its use, since knowing the IP address assigned to it and the protocol, it can be used; second, all its components can be purchased at any electronics store; and third, some of the above commercial devices are not available in various parts of the world, such as Argentina.

Hardware description

Motrol is a smartphone benchmarking and profiling hardware built using open and standard hardware/software components. Motrol allows the user to enable/disable the current flow pass through any of its hardware sockets programmatically (Rest API), on demand, and without the need to physically interact with the hardware. It is an advantage over a standard timer switch whose on/off program should be performed by the user manually. Another advantage is the multiple multi-norm sockets allowing the user to plug up to four electrical devices to control their current flow and independently of each other.

When plugged devices use rechargeable batteries as the main power source to operate, Motrol facilitates the control of charging/discharging periods. In the context of Dew experimentation with smartphones, concretely, software that runs on smartphones, this capability is essential to exercise battery-driven automatic tests. Precisely, Motrol is useful to perform automatic tests that involve programmatically enable/disable current flow -e.g., to dynamically charge/discharge batteries- in order to exercise battery consumption in smartphones under different device load conditions [1].

Motrol plus its firmware plus the base platform-level experimentation software[1] facilitate the automation of long lasting duties including profiling, benchmarking and real testbed setup with an easy to operate, cheap and open source software/hardware solution.

[1] From now on, we will refer to this software as Benchmarking Software.

Essentially, such automatic tests are useful to obtain elementary data that serves as input to simulations and analyses within Dew Computing, a sub-area of mobile distributed computing, which aims at scavenging idle computing resources from a group of smartphones to complete a set of jobs collaboratively. As an emerging distributed computing area, Dew Computing faces important challenges concerning experiment methodologies that allow reproducible experiment conditions, e.g, mimic battery behavior [3]. Motrol contributes in realizing such an experimental methodology because its capability to programmatically control the periods of charging/discharging, which is used by researchers to:

- extract smartphones battery profiles and benchmarking data of their computing, sensing and data transfering capabilities [4]. It means, for instance, to record battery events reported by the smartphone OS while running a program that maintains a configured CPU usage and screen state for the time it takes to consume battery from level X to level Y where X > Y. Resulting profiles and benchmarking data characterize a device model, which means that to build a rich dataset and keep it fresh with data from recently launched devices, it is required hardware support such as Motrol to automate the whole procedure, which is time consuming.
- reset experimental conditions where they involve initiating tests with pre configured smartphones battery level. When performing live tests, i.e., using real (not emulated) smartphones testbeds, for instance for evaluating, job scheduling battery-awareness [5], it is necessary to vary the heuristic for distributing jobs but start each test with the battery level in each smartphone pre-configured in the test scenario. This contributes to achieving reproducibility.

As mentioned, Motrol does not need software other than its firmware to operate. By means of an HTTP request processed by the Motrol Firmware, it is possible to activate/deactivate the charge of a particular mobile without affecting the rest, to activate/deactivate all plugs at the same time, or any possible combination.

When used for the experimentation purposes described previously, the HTTP requests are triggered by the Benchmarking Software which acts as a client to Motrol as a whole. This software accepts profiling plans in the form of plain-text files, that obey a predefined JSON structure, to be queued for execution by all or any of the connected mobile devices. By consuming profiling plans from a queue that the Benchmarking Software administrates, it initiates/terminates benchmarks (or Profiles) when the battery level of each device matches with the battery level configured in the profile description. The Benchmarking Software does it by interacting with a software counterpart component running on mobile devices (which is mainly used to report devices current battery level) and with Motrol Firmware. After executing a profile, the Benchmarking software is responsible for charging/discharging (preparing) mobile devices battery accordingly to assure the battery level and battery state of the next profile to be executed.

With regard to its hardware elements, Motrol is designed with a 4-Channel Relay Module made up of SRD-05VDC-SL-C 5V [6] relays that allow switching power loads. The relay contacts are designed to switch loads up to 10A and 250VAC (30VDC), although voltage levels below these limits are recommended. The control inputs are isolated with optocouplers to minimize the noise perceived by the control circuit while switching the load. The control signal can come from any TTL or CMOS control circuit such as a microcontroller. This module is ideal for switching alternating current loads connected to the electrical network, and supports all microcontrollers, applications in industrial areas, PLC control, among others. This module is capable of controlling various high current equipment for a long time. It can be controlled by many microcontrollers such as Arduino, 8051, AVR, PIC, DSP, ARM, MSP430, and TTL.

Compared to another recent alternatively of our own, called Motrol 2.0 [2], in this paper, we introduce two major differences, a) we provide a detailed blueprint of Motrol hardware components, so interested readers can fully reproduce our solution, and b) we provide a fully-functional hardware device, since Motrol 2.0 it is still in the prototype stage due to the lack of market standardization regarding the use of different types of USB connectors such as mini and C. According to the European Union [7], not sooner than 2024 will manufactured mobile devices use USB C as a standard. At that time, we plan to continue the development of Motrol 2.0 to support the standard.

Design files summary

Design file name	File type	Open source license	Location of the file
Connection diagram file	Fritzing file	CC-BY 4.0	http://doi.org/10.17632/9ndz5b9ccv.3

In the connection diagram file, a schematic was designed with the Fritzzing [8] software to connect a part of the components, as can be seen in Fig. 1.

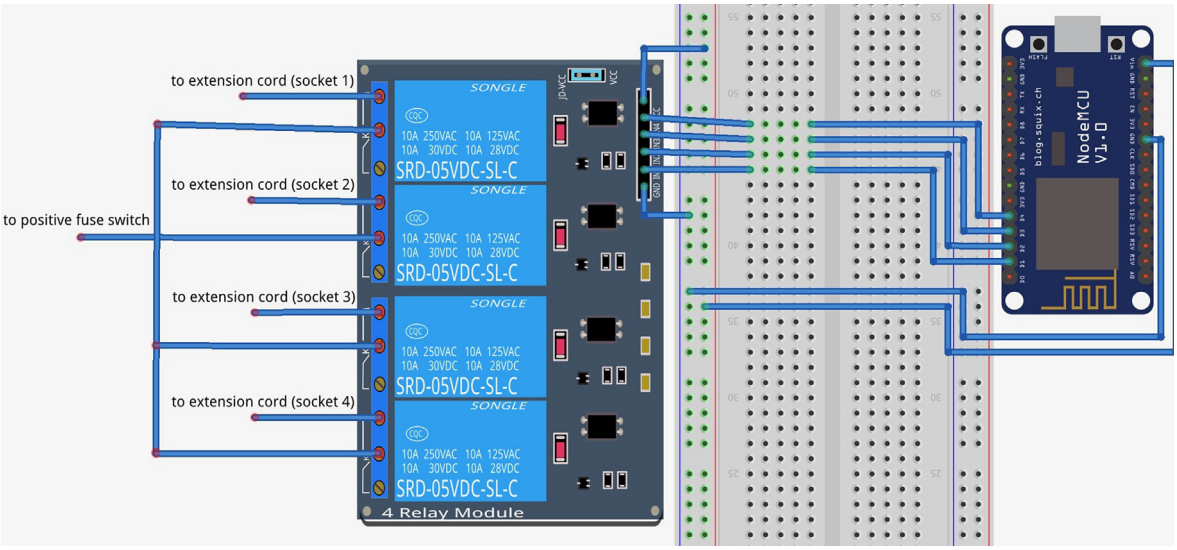

Fig. 1. Connection Diagram.

Bill of materials summary

Designator	Component	Number	Cost per unit-currency	Total cost-currency	Source of materials	Material type
1	NodeMCU Esp8266 (Fig. 2)	1	$ 3,50	$ 3,50	https://tinyurl.com/cym4y8pu	PCB board
2	4 Channel 5 V Relay Module (Fig. 3)	1	$ 5,12	$ 5,12	https://tinyurl.com/2p8pwx57	Other
3	40P DuPont 2.54 mm Rainbow Cable Ribbon (Fig. 4)	1	$ 1,89	$ 1,89	https://tinyurl.com/mr3avefw	Semiconductor
4	Waterproof Electronic Project Box Enclosure (Fig. 5)	1	$ 7,99	$ 7,99	https://tinyurl.com/2p8c5tea	Plastic
5	AC Inlet Male Plug Power Socket With Fuse Switch (Fig. 6)	1	$ 2,54	$ 2,54	https://tinyurl.com/y8uxtxcm	Other
6	Multi plug Electrical Adapter Power Strip with 4 USB Hexagon Extension Socket (Fig. 7)	1	$ 8,69	$ 8,69	https://tinyurl.com/2avrcfxs	Other

Build instructions

Step 1:

Disassemble the Multi plug Electrical Adapter Power Strip (6) to access each plug and release the connections from one of the pins of the sockets since that is the one that is going to be bridged by the relay. Also, desolder the 220 V cable and the power supply from the USB board. (Fig. 8).

Step 2:

From the 220v cable that was separated, choose one and cut into parts of no more than 20 cm each, in order to connect each plug of the Multi plug Electrical Adapter Power Strip (6) with each NO (normal open) output belonging to each relay on the 4-Channel Relay Module (2). Repeat this step for the 3 remaining sockets with the 3 relays that were free. In Fig. 9 you can see how everything is connected.

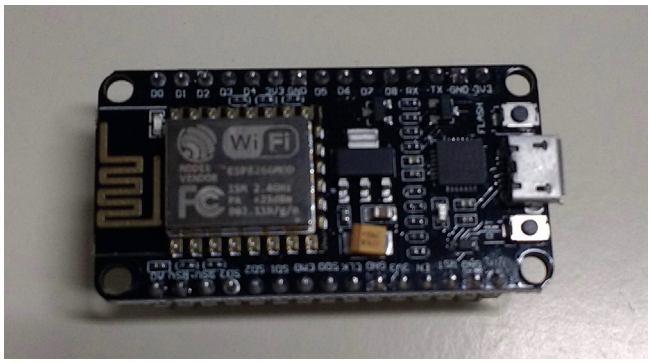

Fig. 2. NodeMCU Esp8266 (1).

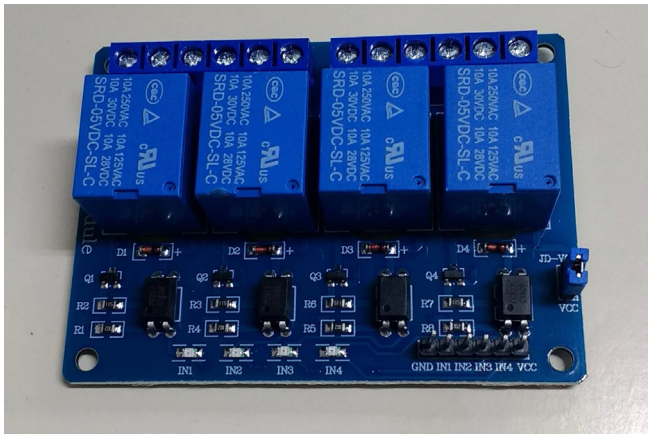

Fig. 3. 4-Channel Relay Module (2).

Fig. 4. 40P DuPont 2.54 mm Rainbow Cable Ribbon (3).

Fig. 5. Waterproof Electronic Project Box Enclosure (4).

Fig. 6. 220 V/110 V Power Supply Switch Adapter Module with Socket Fuse (5).

Fig. 7. Multi Plug Electrical Adapter Power Strip with 4 USB Hexagon Extension Socket (6).

Step 3:

Take another wire from the 220v power cable and cut 5 small parts, 4 to connect the COM (Common) output of each of the relays and 1 to connect everything and connect it to the positive terminal of the Fuse Switch (5) as seen in Fig. 9.

Step 4:

Cut 2 more parts of the cable to connect the ground and neutral of the Fuse Switch (5) as shown in Fig. 10. These two cables go to the Multi plug Electrical Adapter Power Strip (6) to connect the 4 current outlets and remain as shown in Fig. 11.

Step 5:

Drill the container box in one of the sides to embed the Fuse Switch (5) and in the cover to pass the cables to the Multi plug Electrical Adapter Power Strip (6) (Fig. 12).

Step 6:

Assemble the 4-Channel Relay Module (2) inside the container box (4) leaning on its base as shown in Fig. 13.

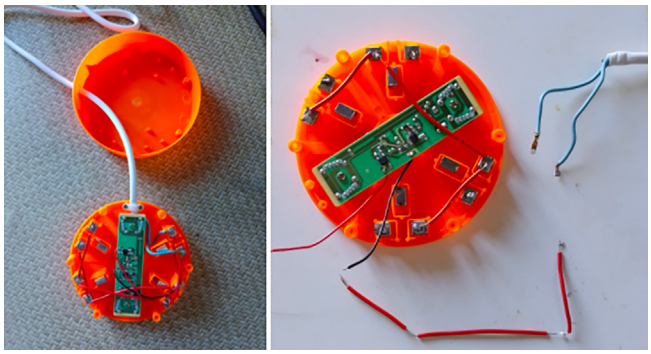

Fig. 8. Multi plug Electrical Adapter Power Strip disassembled with cables disconnected.

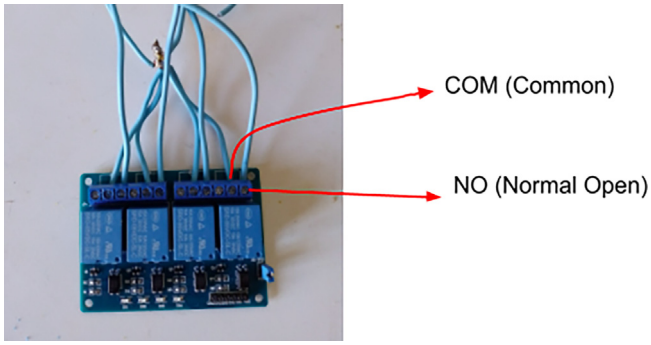

Fig. 9. Wires connected to 4-Channel Relay Module (2).

Fig. 10. Front view of the Fuse Switch (left), rear view (middle) and side view (right).

Step 7:

Embed the Fuse Switch (5) on the side of the container box and pierce the Multi plug Electrical Adapter Power Strip (6) at its base to pass the cables that come from the box (Fig. 14).

Step 8:

Connect the digital outputs of the NodeMCU Esp8266 (1) to the pins of the 4-Channel Relay Module (2) with the cables (3) as shown in Fig. 15.

Step 9:

Connect the *Vin* input of the NodeMCU ESP8266 (1) to the *Vcc* pin of the USB board that has the Multi plug Electrical Adapter Power Strip (6). Add to this wire an output to connect to the 4-Channel Relay Module (2) on the VCC pin. Connect the GND of the ESP8266 with the one of the USB board of the Multi plug Electrical Adapter Power Strip (6). In this way, the ESP8266 can be powered without the need to connect to a USB port.

Step 10:

Assemble the NodeMCU Esp8266 (1) inside the container box (4) as shown in Fig. 16.

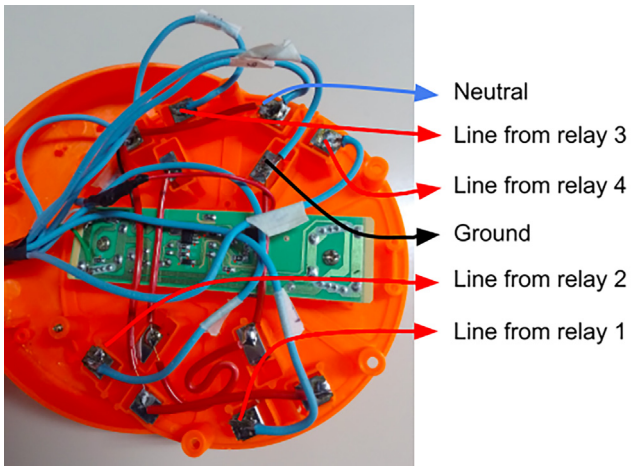

Fig. 11. Connections to the 4 outlets in the Multi plug Electrical Adapter Power Strip.

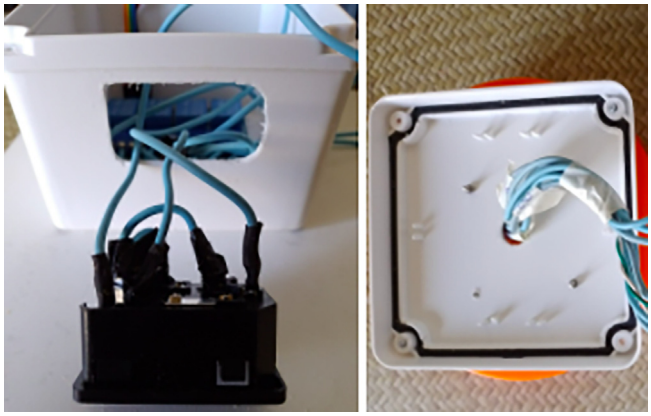

Fig. 12. Container box holes to embed the Fuse Switch and pass the cables to the Multi plug Electrical Adapter Power Strip.

Step 11:

Connect the NodeMCU Esp8266 (1) microcontroller to the host to download the control program.The final result is shown in Fig. 17 and in Fig. 18 you can see how all the parts are interconnected to make use of Motrol.

Operation instructions

To start using Motrol, you must first plug in a 220 V cable to the Fuse Switch. Then connect the USB cable to the serial port of the computer that is going to be used as host. By entering an SSID and its password through the serial port, the WiFi network to which it will be connected is configured. If it can connect successfully, the device will return the IP assigned by the router. Then turn on the corresponding button.

The entrypoint used to manage the 4 relay board (connected to Multi plug Electrical Adapter Power Strip) consists of a IP address with a parameter that specifies which relay to turn on or off:

http://IP_ADDRESS/OnOff? handle=N. N is a character between 0 and 9.

For example, a request to turn on the socket where mobile 1 is connected is:

IP_ADDRESS/OnOff?handle=1.

whereas it can be turned off with handle = 5. This protocol is explained in Table 1.

Motrol is controlled by a program running on the NodeMCU Esp8266 (1) microcontroller. This firmware runs a web server that receives an HTTP GET request with a digit character between 0 and 9 to handle the connection or disconnection of each relay.

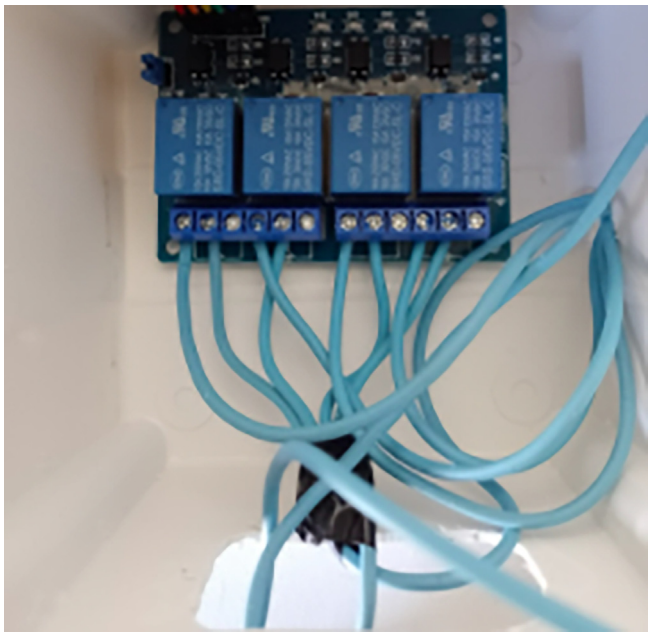

Fig. 13. 4-Channel Relay Module inside the box container.

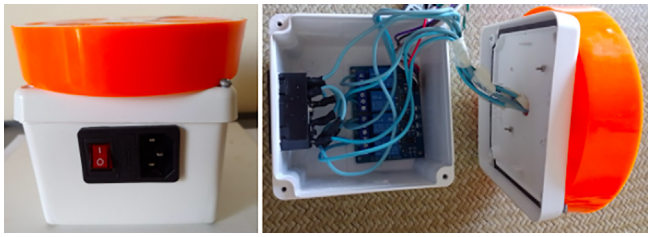

Fig. 14. Built-in fuser (right) and slipper attached to the lid of the container box (left).

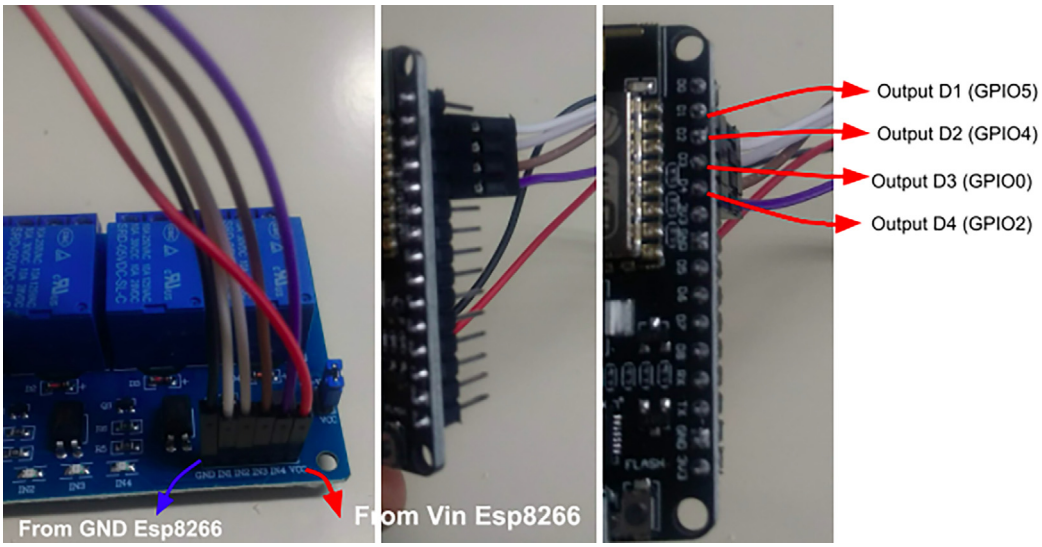

Fig. 15. Connections to the digital outputs of the ESP8266 with the 4-Channel Relay Module.

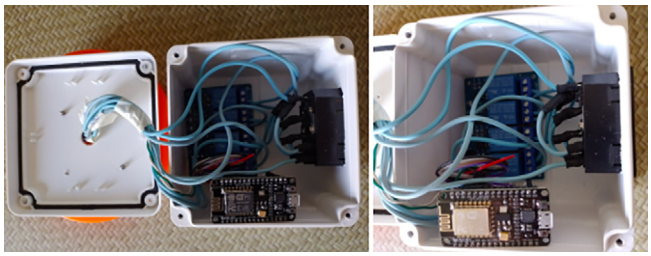

Fig. 16. ESP8266 embedded inside the container box.

Fig. 17. Fully-assembled Motrol.

The protocol created for this purpose is the following shown in Table 1, where *t-1* means that the relay state of the previous time is maintained:

Validation and characterization

Motrol is equipped with SRD-05VDC-SL-C 5 V relays, which have a useful life of operations: mechanical (100,000) and electrical (10,000,000). Their operation time is a maximum of 10 ms and a maximum release time of 5 ms. Also, due to the limitation of 10 amp (details in section 2), it is not convenient to use a power strip extension cord to extend the number of plugs because it can overload the circuit. In that case the fuse in 220 V/110 V Power Supply Switch Adapter Module with Socket Fuse (5) will blow, indicating overheating. Then the first step is to unplug the power strip extension cord and then replace the fuse.

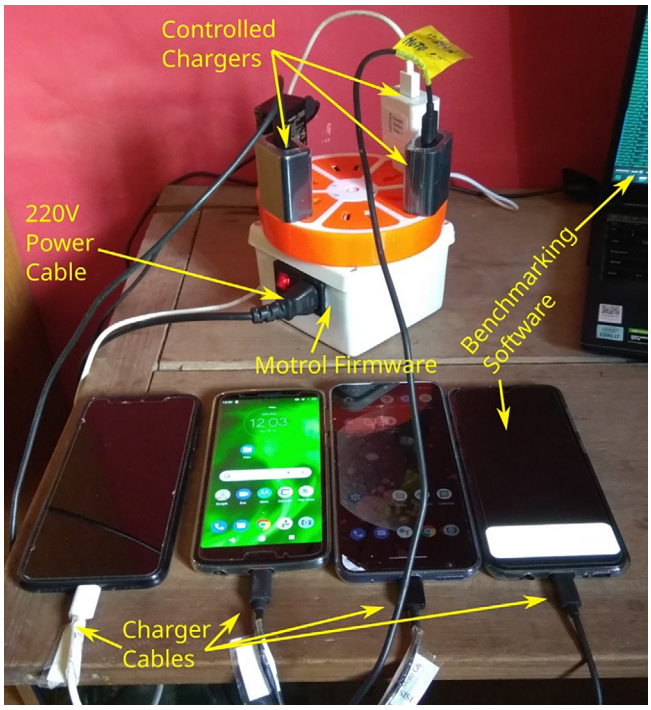

Fig. 18. Using Motrol.

Table 1
Protocol adopted to manage the connection / disconnection of each mobile device via its charger.

Command	Mobile 1	Mobile 2	Mobile 3	Mobile 4
0	off	off	off	off
1	on	t-1	t-1	t-1
2	t-1	on	t-1	t-1
3	t-1	t-1	on	t-1
4	t-1	t-1	t-1	on
5	off	t-1	t-1	t-1
6	t-1	off	t-1	t-1
7	t-1	t-1	off	t-1
8	t-1	t-1	t-1	off
9	on	on	on	on

Fig. 19 illustrates the usefulness of Motrol 1.5. On the top, there is a high level workflow describing the benchmarking/profiling steps applied to a single device. In research duties, the workflow is frequently exercised for several devices simultaneously. Commonly, tests are configured with initial battery level and battery state preconditions. Only when such preconditions are met, the test is able to start. This logic corresponds to the "prepare device" activity, whose workflow is shown at the bottom part of the image. On the left side of the bottom part (without Motrol) it can be seen that human intervention is present in all sub activities. By contrast, the right side (with Motrol) shows that human intervention is not necessary and was replaced by software, which listens for battery updates obtained by means of the device battery API, and Motrol 1.5 which is in charge of interrupting/enabling current flow to accommodate battery charging state appropriately. Without Motrol, manual operation can result in a tedious and error prone duty. With Motrol, all steps are executed automatically. The benefits are major when a series of benchmarking/profiling tests should be run on several devices, and specially when running *synchronized* tests, which are tests where several devices are required to meet certain battery level and battery state synchronously as precondition for starting the test. In this video[2] you can see how the operation is. The necessary scripts[3] to configure everything to start an experiment are also provided.

[2] https://data.mendeley.com/datasets/9ndz5b9ccv/3/files/3694c2b7-376a-4997-9fbb-10a0df5c4aa3.
[3] https://doi.org/10.17632/9ndz5b9ccv.3.

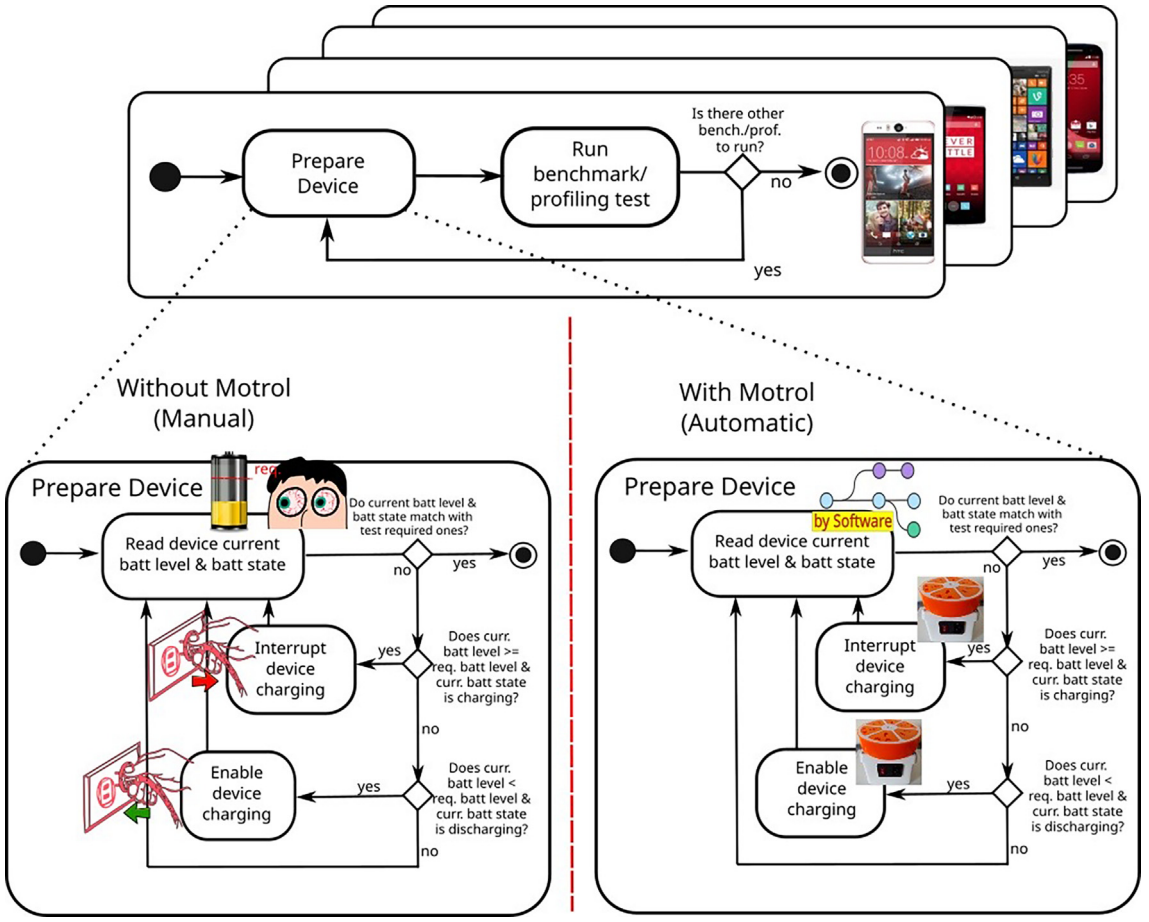

Fig. 19. Switching operation without Motrol (left) and with Motrol (right).

Ethics statements

Our work involves neither human subjects nor animal experiments.

CRediT authorship contribution statement

Juan Manuel Toloza: Writing – original draft, Software, Validation. **Matías Hirsch:** Conceptualization, Investigation, Writing – review & editing, Validation. **Cristian Mateos:** Conceptualization, Methodology, Writing – review & editing, Supervision. **Alejandro Zunino:** Conceptualization, Methodology.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

We acknowledge the financial support by CONICET through grant PIP no. 11220170100490CO and by ANPCyT through grant 2 PICT-2018-03323.

Appendix A. Supplementary data

Supplementary data to this article can be found online at https://doi.org/10.1016/j.ohx.2022.e00340.

References

[1] M. Hirsch, C. Mateos, A. Zunino, J. Toloza, A platform for automating battery-driven batch benchmarking and profiling of Android-based mobile devices, Simul. Model. Practice Theory, Vol. 109 (February 2021). 10.1016/j.simpat.2020.102266.
[2] C. Mateos, M. Hirsch, J. Toloza, A. Zunino, Motrol 2.0: A Dew-oriented hardware/software platform for batch-benchmarking smartphones, in: 2021 IEEE 45th Annual Computers, Software, and Applications Conference (COMPSAC), 2021, pp. 1772–1777, https://doi.org/10.1109/COMPSAC51774.2021.00265.
[3] M. Hirsch, C. Mateos, J.M. Rodriguez, A. Zunino, Dewsim: A trace-driven toolkit for simulating mobile device clusters in dew computing environments, Softw. Pract. Exper. 50 (5) (2020) 688–718.
[4] M.A. Hoque, M. Siekkinen, K.N. Khan, Y. Xiao, S. Tarkoma, Modeling, profiling, and debugging the energy consumption of mobile devices, Acm Comput. 19 Surv. 48 (3) (2015) 1–40.
[5] M. Hirsch, J.M. Rodriguez, A. Zunino, C. Mateos, Battery-aware centralized schedulers for CPU-bound jobs in mobile Grids, Pervasive Mobile Comput. 1 (29) (2016) 73–94.
[6] https://datasheetspdf.com/pdf-file/720556/Songle/SRD-05VDC-SL-C/1 Last accessed Apr-4 2022.
[7] https://www.consilium.europa.eu/en/press/press-releases/2022/01/26/council-approves-a-common-charger-for-electronic-devices/ Last accessed Jun-9 2022.
[8] https://fritzing.org/ Last accessed Apr-4 2022.

Ph. D. Juan Manuel Toloza graduated as Systems Engineer in march 2009 at UNICEN (Argentina) and got the Ph.D. (Computer Science) in 2013 at UNLP (Argentina). He is a member of ISISTAN Research Institute and LabSET (Embedded Systems Laboratory) at UNICEN. Works at CONICET as Research and Development Support Staff. His research interests include sensor data analysis, Internet of Things, embedded systems, robotics and agtech solutions. He is teacher assistant at the Computer Science Department - UNICEN