HardwareX 10 (2021) e00243

Contents lists available at ScienceDirect

HardwareX

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

On the development of flexible mobile multi-sensor units based on open-source hardware platforms and a reference framework

Franklin Oliveira [a], Daniel G. Costa [b,*], Ivanovitch Silva [c]

[a] UEFS-PGCC, State University of Feira de Santana, Feira de Santana, Brazil
[b] UEFS-DTEC, Department of Technology, State University of Feira de Santana, Feira de Santana, Brazil
[c] UFRN-DCA, Department of Computer Engineering and Automation, Federal University of Rio Grande do Norte, Natal, Brazil

ARTICLE INFO

Keywords:
Mobile sensing
Development framework
Open-source hardware platforms
Internet of Things
Smart cities

ABSTRACT

For some IoT applications, mobile entities are considered as the main source of sensed data, requiring the attachment of sensor modules on them. The endowing of sensing capabilities to such mobile entities can be performed in different ways, but the adoption of a reference hardware framework can bring a series of advantages, specially in dynamic complex scenarios. This article exploits the MSensorMob2 multi-sensor hardware framework for monitoring in areas with disconnection periods, comprising sensing, transmission and reconfiguration functions. Comprehensive analyses on multiple open-source hardware platforms are conducted, assessing their costs, deployment constraints and performance issues when implementing this development framework.

Specifications table

Hardware name	*MSensorMob2-based Multi-sensor Monitoring Units*
Subject area	Engineering and material science
Hardware type	Field measurements and sensors; Electrical engineering and computer science
Closest commercial analog	"No commercial analog is available"
Open source license	GNU General Public License (GPL)
Cost of hardware	The costs for all created prototypes are properly presented in this article.
Source file repository	https://doi.org/10.17632/tz4y6s9g5w.1

1. Hardware in context

The maturation of the Internet of Things (IoT) paradigm has the potential to change the way we perceive and interact with the environment. Particularly, with the availability of affordable off-the-shelf hardware platforms, the IoT paradigm

* Corresponding author at: UEFS-PGCC, State University of Feira de Santana, Feira de Santana, Brazil.
E-mail addresses: franklinoliveira40@gmail.com (F. Oliveira), danielgcosta@uefs.br (D.G. Costa), ivanovitch.silva@ufrn.br (I. Silva).

https://doi.org/10.1016/j.ohx.2021.e00243
2468-0672/© 2021 The Author(s). Published by Elsevier Ltd.

has been largely exploited to allow data monitoring from multiple sources, concurrently, creating the fundamentals for a whole new set of disruptive applications. In recent years, emergent IoT-based applications have benefited from such increasing availability of low-cost, tiny and powerful hardware platforms, resulting in a vibrating development scenario [1,2].

For a group of applications, two or more types of data have to be periodically sensed, with each monitoring sample aggregating all sensed data at a given time, together with the GPS position for the corresponding sampling instance and its timestamp. Such samples are expected to be retrieved by multi-sensor hardware units attached to mobile entities in a monitoring scenario, such as cars, bicycles, skates, buses, drones, wearable gadgets, and virtually any generic mobile entity. Since some monitoring scenarios may lack the basic networking infrastructure for continuous online transmission of the samples, either due to costs constraints or coverage holes, such multi-sensor units should operate for long periods of disconnection, demanding them to locally store samples for future transmissions. Therefore, for all aforementioned defined characteristics, there will be some basic functions that will be equivalent for different applications, which push us to the proposal and adoption of standard hardware frameworks [3,4].

In [5], we proposed a framework to support the development of multi-sensor hardware units targeted at GPS-based periodic multi-variable monitoring in scenarios with a discontinuous networking service. The proposed framework in that work defined the hardware architecture and a framework for the execution of a set of basic functions. In this article, we extend that framework a bit further, including a new service for the updating of the configurations of the units. Such updating service is intended to allow dynamic changes in deployed multi-sensor units when the initial monitoring requirements are altered, which may be a reasonable demand in long-lasting applications as in smart cities scenarios.

For this new enhanced framework, referred as MSensorMob2, we assembled a group of multi-sensor units exploiting different open-source hardware platforms, a set of sensor devices, batteries and some basic electronic components for interconnections, always considering affordable off-the-shelf hardware components for Do-It-Yourself (DIY) developments [6]. Additionally, we also employed a high-level hardware module for the Raspberry Pi platform as an easier though more expensive alternative for sensors assembling. Then, we performed a comprehensive quality assessment for this group of multi-sensor units based on the MSensorMob2 framework, comparing their deployment costs, physical configurations and achieved performance. We believe that such multi-level assessments are valuable when indicating how such units could be developed for real applications.

2. Extended hardware framework: MSensorMob2

The MSensorMob2 hardware framework is an improvement of our work in [5], which is intended to support easier and affordable development of flexible multi-sensor mobile units in scenarios with discontinuous networking infrastructures. The operation details of such enhanced framework are described here, indicating how MSensorMob2-complaint units will be created for performance comparisons in this article.

2.1. Hardware architecture

The MSensorMob2 framework defines an architecture within the three-layers IoT generic operation model [7], namely perception, network and application layers. While the lowest conceptual layer is responsible for the perception of the environment, the network layer is located just above the perception layer, interfacing the flow of data from physical to application layer. Since we want to support the development of sensor units to support virtually any type of sensors-based application in IoT monitoring scenarios, the MSensorMob2 architecture implements the perception layer and part of the network layer.

Fig. 1 presents the defined hardware architecture for the creation of multi-sensor units. In general, this architecture was specified to assure energy and operational self-sufficiency, also being easy to create and operate, while keeping it highly adaptable. Moreover, since the developed units will be battery-powered, low-energy consumption was also a project request for the architecture.

The MSensorMob2 architecture is composed of a (core) controller, Wi-Fi interface, SD/microSD card, a GPS (receiver) device, portable power supply (LiPo battery, power bank, etc.), and a number of sensors and components for human–machine interface (particularly, the S-Button and M-Button, besides informative LEDs). This way, we expect to allow the development of units that operate without any support from smartphones or computers. As an important consideration, for the implementation of the user input interface, we assume that they are implemented as conventional push buttons. However, other types of buttons (e.g. switch ON/OFF buttons) could be adopted instead, just adjusting eventual transitions between logical states, without impacting the operation of the framework.

Finally, we adopt the Wi-Fi technology to provide the expected networking services, mostly due to the low-cost and easy-to-use nature of this wireless technology, as well as its high availability as an onboard component on many popular development hardware platforms.

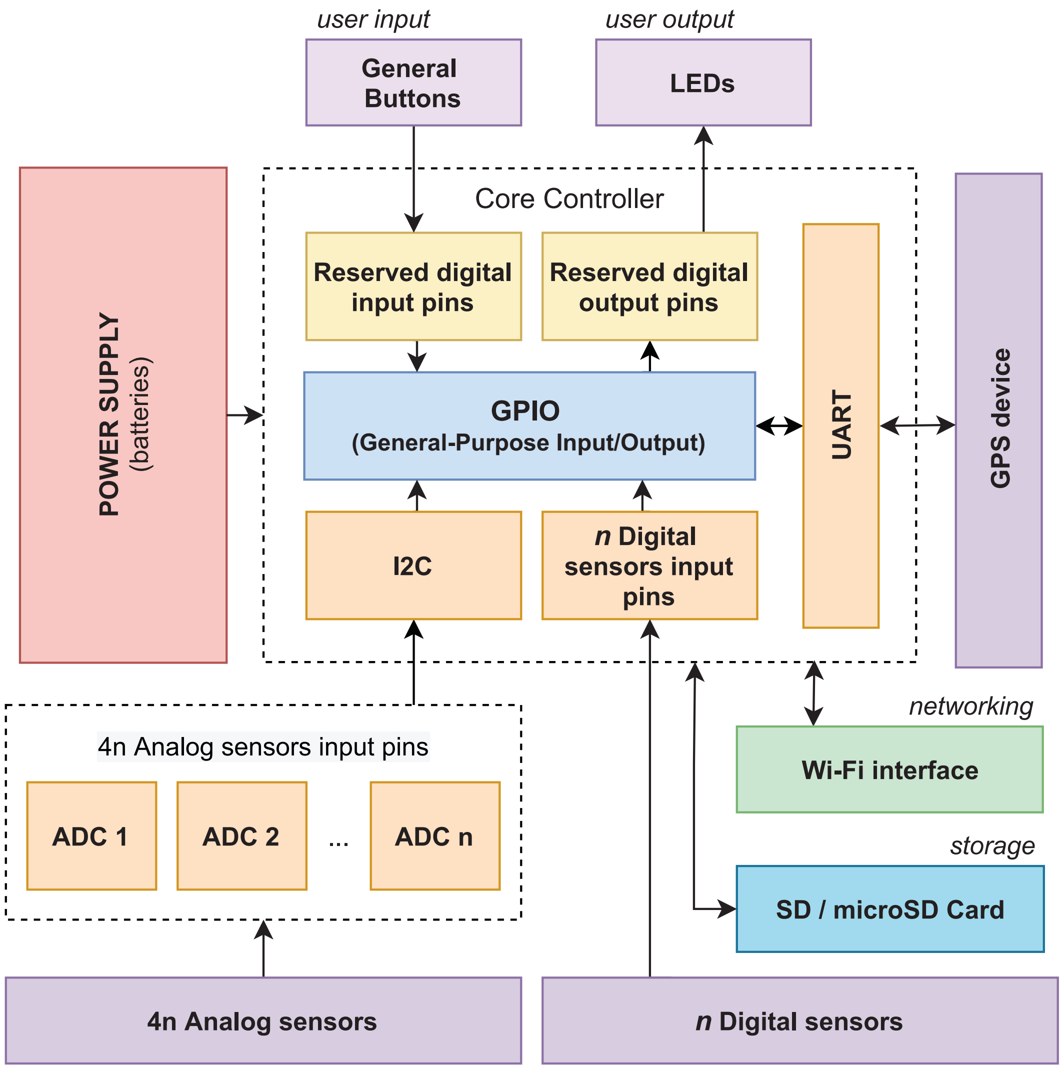


Fig. 1. Hardware architecture for the defined multi-sensor mobile units.

2.2. Hardware firmware

We also defined a software component (firmware) that is responsible for managing the accessing to the multi-sensor unit resources, proposing the standard operation procedures. In short, our original MSensorMob firmware in [5] defines functions for a) automatized networking configuration, b) the types and number of sensor devices, c) the number of analog-to-digital converters (ADC) that will be used, d) the monitoring (sensing) frequency, e) the sampling and storage procedures and f) samples transmission to any cloud-based service. For the MSensorMob2 firmware, besides all mentioned procedures in the original MSensorMob firmware, we also included an updating service to allow dynamic remote re-configuration of the units, which may be valuable for many applications that employ many distributed multi-sensor units in dynamic scenarios.

The MSensorMob2 firmware operates based on six logical states, as depicted in Fig. 2. The use of LEDs is important in this case to visually indicate to the users about the changes in the logical states, while the S-Button and M-Button are user inputs to trigger states changes [5]. As an important remark, the M-Button and S-Button are being implemented as conventional push buttons in that figure, as previously described. This way, states transitions are always triggered when the M-Button is pressed, while a state transition is triggered when the S-Button is pressed from the IDLE state.

The defined six logical states are described in details as follows:

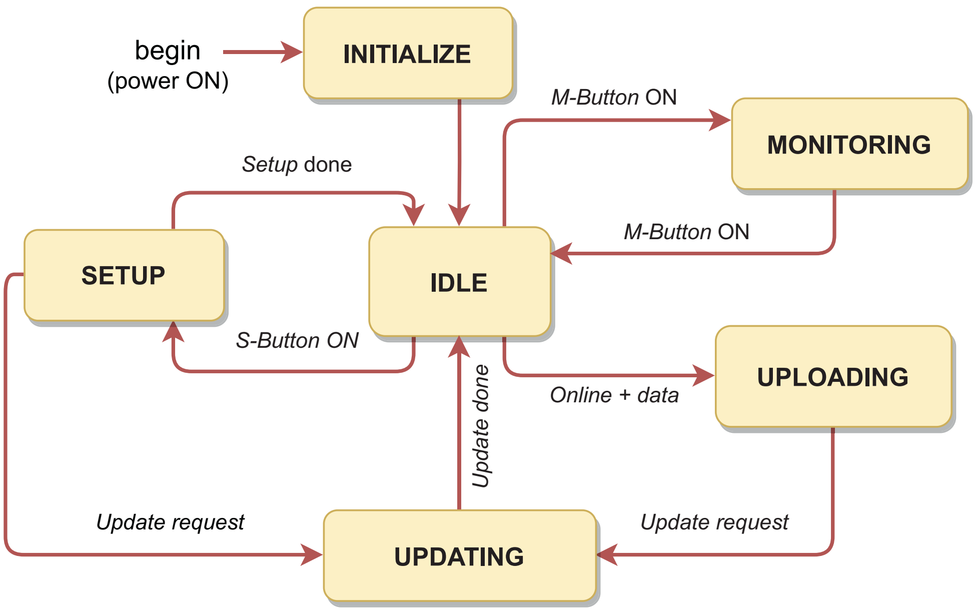


Fig. 2. Defined firmware, with all expected functions for MSensorMob2-based units.

- INITIALIZE: it is an initial state that is executed just after the unit is powered on. It is responsible to read the configuration file stored on the SD/microSD card, which contains the hardware specifications (ID, sample frequency, ADCs reference, digital/analog sensors pins, etc.) as defined in [5]. After that, the unit is ready to be used, moving to the IDLE state to wait for user commands and additional interactions;
- IDLE: this is the standby state, when the unit waits for some user's action or other requests;
- SETUP: a MSensorMob2 unit will sense the environment and store samples to be further transmitted to a cloud server. Such transmission will be performed when the unit is under the coverage radius of its base Wi-Fi station. The SETUP state is responsible for connecting the monitoring unit to a wireless network, exploiting for that the popular WPS (WiFi Protected Setup) technology. When the *S-Button* is pressed, the SETUP procedure is initiated;
- MONITORING: it initiates when the *M-Button* is pressed, indicating that the unit will start collecting data from the configured sensors, storing such data into "csv" sample files (in a daily basis). When the *M-Button* is pressed again, the unit returns to the IDLE state;
- UPLOADING: from the IDLE state, when the unit is connected to its base Wi-Fi station and there are unsent samples stored on it, the unit automatically transitions to the UPLOADING state for transmission. Successfully transmitted samples are removed from the SD/microSD card;
- UPDATING: from both SETUP and UPLOADING states, the MSensorMob2 firmware automatically moves to the UPDATING state. In this state, the monitoring unit requests the "version.txt" file, located in the respective unit's folder in the cloud service used to store the monitoring data. Once the "version.txt" file is received from the cloud storage service, the unit performs a comparison of the indicated firmware version with the current version running on the device. After identifying that the versions are not the same, the monitoring unit also requests, from the cloud storage service, the firmware in the version informed in the "version.txt" file. Finally, the.tar file that stores the firmware is downloaded, extracted and the device is restarted, allowing the new firmware to be executed after booting.

3. Required hardware components

Although the MSensorMob2 framework was designed to address a group of multi-sensor units expected to operate by continuously sensing the environment for long disconnection periods, the framework as a whole can be implemented in different ways, since the basic defined services are provided. Such flexibility allows, for example, the use of different hardware platforms when implementing the Core Controller, as well as the adoption of different numbers, types and configurations of sensor devices and electronic components. As a result, a great number of different, but compatible MSensorMob2 units can be developed, with particular implementation, processing and deployment issues. Thus, describing and comparing some of such possible construction units are of paramount importance when supporting applications based on this framework.

3.1. Employed hardware platforms

Among the hardware possibilities when implementing the units, open-source hardware platforms have been largely used as flexible off-the-shelf options, assuring reasonable processing power while keeping low-energy consumption patterns. Additionally, the popularization of open-source hardware platforms has considerable increased the number of programming resources (libraries) and available compatible hardware components, making them an affordable and effective development choice for emerging applications in areas such as Smart Cities and Industry 4.0 [8,9]. This way, we consider a number of different open-source hardware platforms for the development of MSensorMob2-complaint units.

We exploit two well-defined hardware families when creating the MSensorMob2 prototypes: single board computers and microcontroller boards. Based on those "hardware families", we selected a group of popular boards to be used, as depicted in Fig. 3.

Table 1 summarizes the selected boards to be exploited when creating MSensorMob2 units.

As can be seen, boards with different resources and processing power were selected, allowing us to compare the performance of MSensorMob2 units under different hardware configurations. Additionally, besides the selected boards, we will also consider an enhanced implementation pattern for the Raspberry Pi 3, exploiting the GrovePI + hardware framework when creating the units with a different setting for the sensor devices (presented in Fig. 3).

Finally, as additional parameters, the dimension of the boards (Fig. 4a) and the perspective comparison of I/O pins among them (Fig. 4b) are presented. Such parameters can be helpful when creating real units, specially when the number and types of attached sensors, as well as their final dimensions for deployment, become important development issues to be considered.

3.2. Bill of materials

Setting the global cost for a new hardware product is one of the most critical decisions. It needs to get the pricing estimations as early as possible in order to achieve project budget decisions. The methodology used in this article was based on average cost found on the websites of boards manufacturers and official retail partners. The results are described in Table 2 and Table 3, with amounts displayed in US Dollars (USD).

4. MSensorMob2 prototypes

We specified four groups of MSensorMob2 prototype configurations based on the components presented in Table 1, trying to encompass different processing capabilities, I/O resources, embedded functionalities and budgets. These four configuration settings are described as follows:

- C1-Raspberry Pi prototypes: all prototypes developed using the Raspberry Pi Zero WH, Raspberry Pi 2B, Raspberry Pi 3B and Raspberry Pi 4B. This group of prototypes was created using different versions of the Raspberry Pi boards, varying only their performance according to hardware characteristics. In fact, since they have the same I/O pins, the physical connections of the components are the same. Additionally, they all use the Raspbian Stretch operating system, excepting for the Raspberry Pi 4B that employs the Raspbian Buster system (as defined by the manufacturer);
- C2-GrovePi + prototype: it is composed using the Raspberry Pi 3B with the GrovePi + HAT (Hardware Attached on Top). The GrovePi + is a development framework that facilitates the interaction to analog and digital sensors, making them "plug-and-play". In our prototype, we use a GrovePi + HAT along with Grove sensors and a Grove GPS receiver. As an important remark, the Raspberry Pi 3 was the most recent Raspberry Pi model that we could find updated libraries for this framework, so it was the adopted model for this configuration setting;
- C3-NodeMCU prototype: it is composed using the NodeMCU module, a development board with an integrated ESP8266 microcontroller;
- C4-Arduino prototypes: this configuration setting was created using different Arduino boards, assuming three different models (Nano, Uno and Mega). Although having different hardware configurations and number and types of I/O pins, the schematics for the three models were the same.

Since the implementation cost is an important development concern, Fig. 5 presents the estimated costs taking as reference the information in Table 2 and Table 3. In that figure, the cost of all employed components are accounted, excepting for the costs of wires, resistors and breadboard, as they are only required in this prototyping phase.

Fig. 6 presents the defined hardware configurations for the developed prototypes, highlighting the employed sensor devices and additional components. The ADC is used by all prototype groups, excepting for the C2 group that uses the GrovePi + standard. Moreover, for the C4 group, the NodeMCU was used to provide Wi-Fi communication for the Arduino boards. Particularly for the C4 group, although they have different I/O patterns, the usage of each of the three assessed Arduino models does not impact the construction of the prototype units, excepting for their dimensions, costs and energy consumption, as discussed in this article.

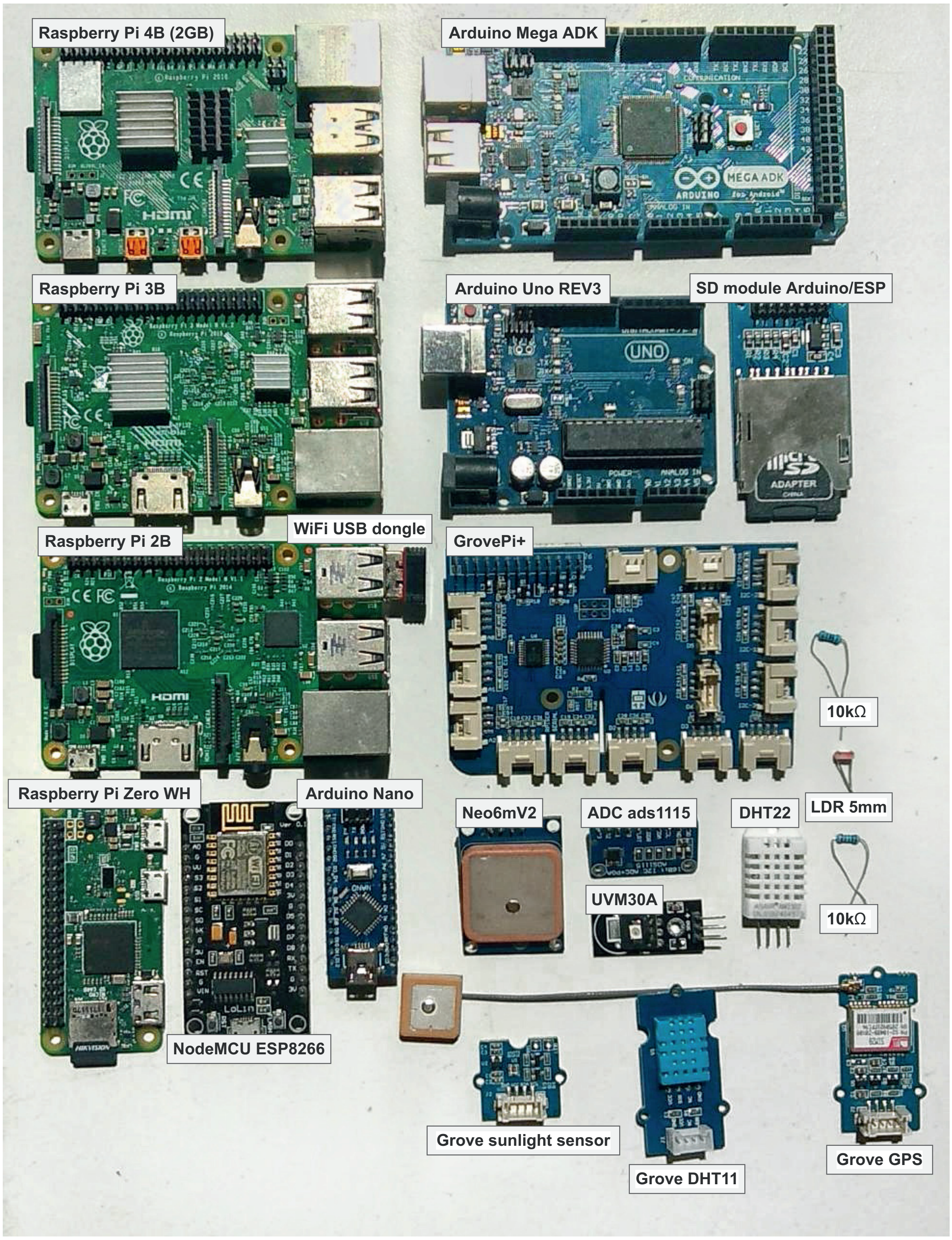
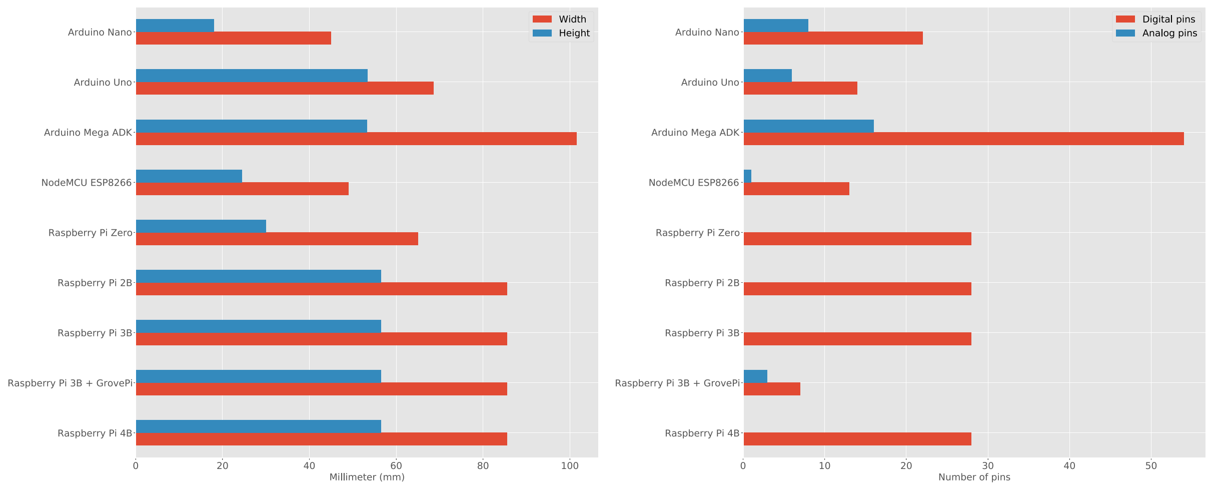


Fig. 3. All development boards and electronics components exploited for the construction of the MSensorMob2 prototypes.

Table 1
Employed development hardware platforms. It is presented the processing frequency of the board (CPU), its local processing memory (RAM), the number and type of the available I/O pins (Digital and Analog), the presence of embedded Wi-Fi capability and the existence of embedded support for a SD/microSD card.

Board	CPU (MHz)	RAM (KB)	Digital	Analog	Wi-Fi	SD card
		Single board computer family				
Raspberry Pi 4B	1500	2,000,000	28	0	YES	YES
Raspberry Pi 3B	1200	1,000,000	28	0	YES	YES
Raspberry Pi 2B	900	1,000,000	28	0	NO	YES
Raspberry Pi Zero WH	1000	500,000	28	0	YES	YES
		Microcontroller board family				
NodeMCU ESP8266	80	64	13	1	YES	NO
Arduino Mega ADK	16	264	54	16	NO	NO
Arduino Uno	16	32	14	6	NO	NO
Arduino Nano	16	32	22	8	NO	NO



(a) Physical dimension. (b) Digital and analog pins.

Fig. 4. Parameters that may affect the deployment of the boards.

Table 2
Average cost for each development board.

Hardware development board		
Board	**Cost (USD)**	**Source**
	Single board computers	
Raspberry Pi 4B (2 GB)	35.00	https://www.sparkfun.com
Raspberry Pi 3B	35.00	https://www.canakit.com
Raspberry Pi 2B	35.00	https://www.canakit.com
Raspberry Pi Zero WH	10.00	https://www.sparkfun.com
	Microcontroller boards	
NodeMCU ESP8266	7.95	https://www.pishop.us
Arduino Mega ADK	40.00	https://vilros.com
Arduino Uno REV3	23.00	https://vilros.com
Arduino Nano	20.50	https://vilros.com

The schematic configuration of the defined prototypes can be seen in Fig. 7. Since the connection of the Grove components is straightforward, requiring simple connections through the specified ports in their datasheets, the C2 configuration setting is not described in that figure.

Table 3
Average cost for each employed electronic component.

Component	Cost (USD)	Source
	Sensors	
DHT22	9.95	https://chicagodist.com
LDR 5 mm	2.30	https://sg.element14.com
UVM30A	18.30	https://abra-electronics.com
	GrovePi Sensors	
Grove DHT11	6.50	https://www.seeedstudio.com
Grove sunlight sensor	11.90	https://www.seeedstudio.com
	GPS devices	
Neo6mV2	17.00	https://www.amazon.com
Grove GPS	25.90	https://www.seeedstudio.com
	Miscellaneous	
ADC ads1115	14.95	https://www.adafruit.com
GrovePI+	34.50	https://www.seeedstudio.com
SD module Arduino/ESP	6.30	https://abra-electronics.com
WiFi USB dongle	19.95	https://www.canakit.com

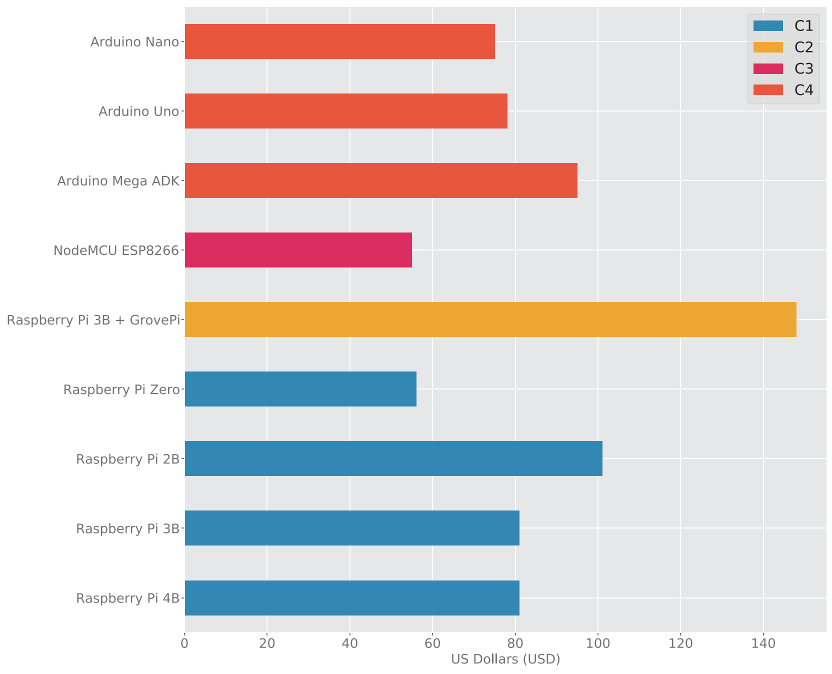


Fig. 5. Total estimated costs for all prototypes, according to the chosen core controller.

5. Performance comparisons

After creating the MSensorMob2 prototypes, we performed a series of comparison tests in order to allow performance assessment for different hardware configurations. Doing so, future implementations and deployments acquire additional information when deciding the most suitable hardware options. For that, we defined a simple simulation scenario, with a single unit for each of the defined prototype types.

The MSensorMob2 development framework defined six different states. Initially, we measured the execution time for the INITIALIZE, SETUP, UPLOADING and UPDATING states, as presented in Fig. 8.

As expected, the reading of configuration files, the processing of the defined procedures and the execution of networking functions will depend on the characteristics of the defined hardware. In Fig. 8, we could see that the Raspberry Pi boards performed better in all states, excepting for the INITIALIZE state. One reason for that is the use of an Operating System by the boards, which adds processing complexities when accessing and managing files. Differently, microcontroller-based units

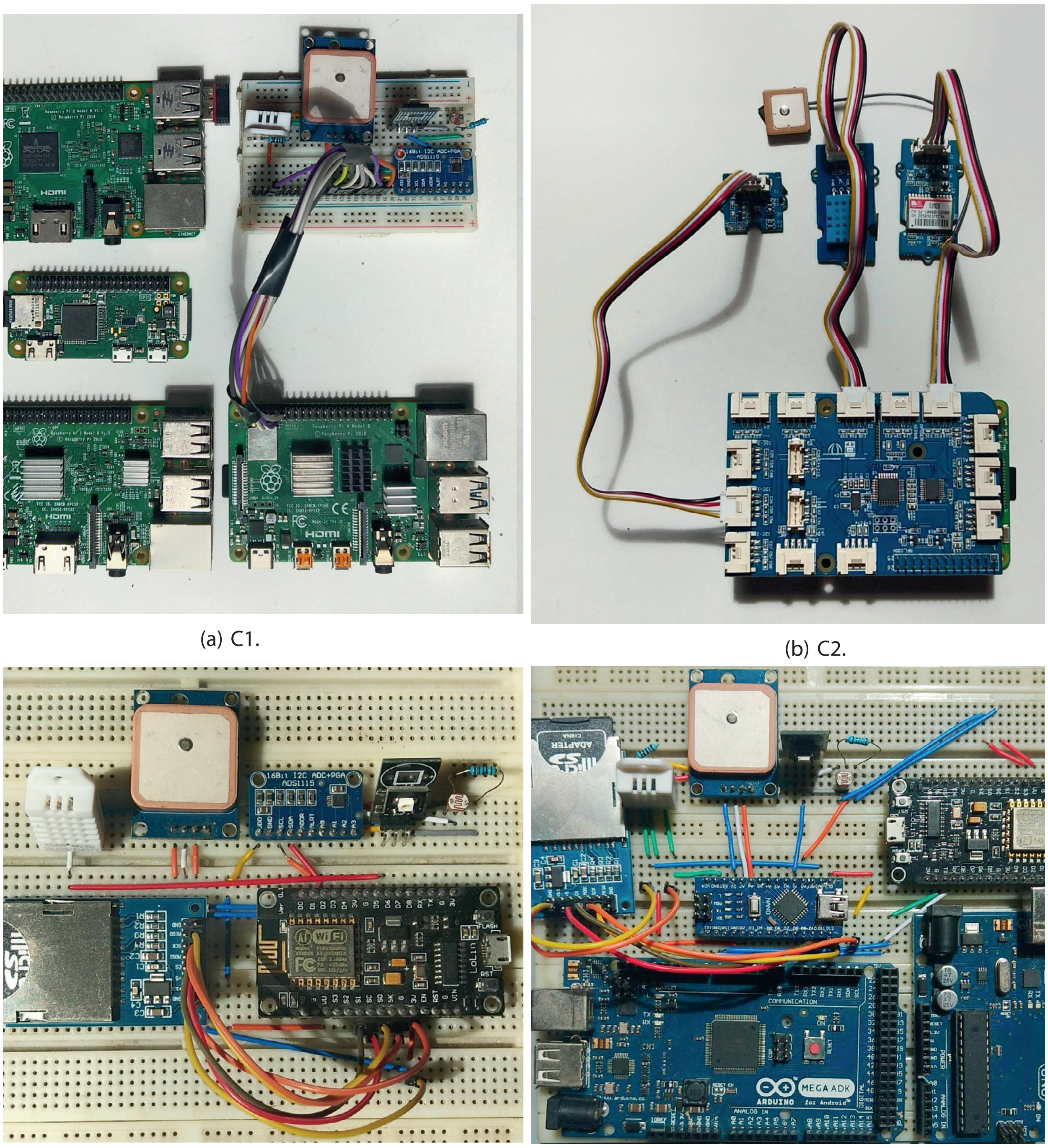
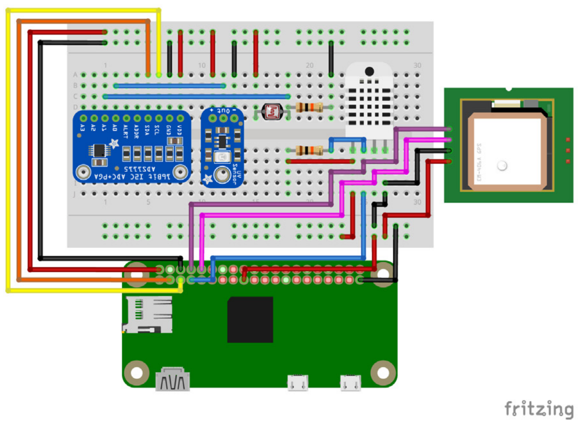

(a) C1. (b) C2. (c) C3. (d) C4.

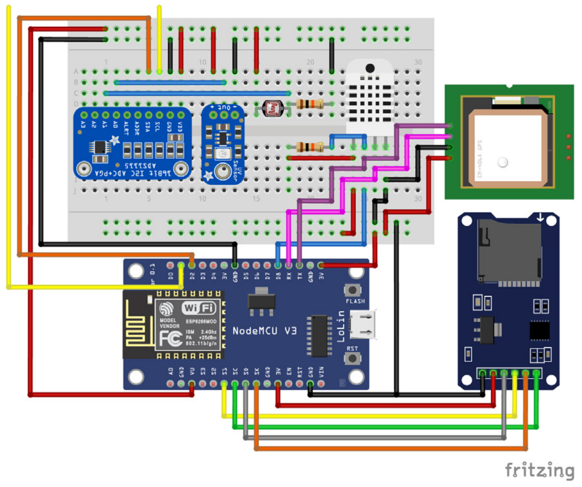
Fig. 6. Adopted hardware configurations for the four configuration settings of the developed prototypes.

have a simpler file system structure, reducing the accessing time to the files even having lower processing power when compared to the Raspberry Pi boards.

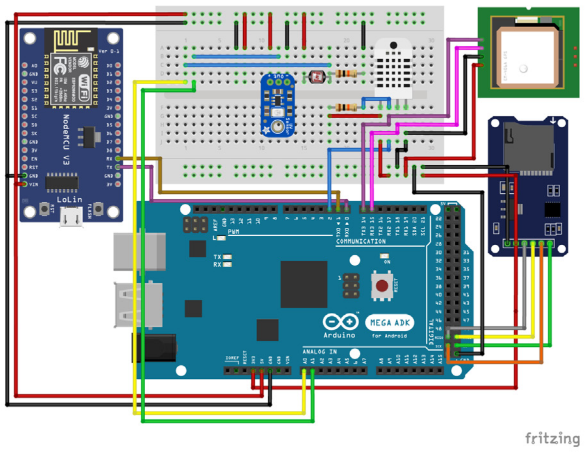
Concerning the UPDATING state, the nature of microcontroller-based boards imposes a natural restriction for updates to the operation and configuration of the units. While single board computers like the Raspberry Pi family operate using an operating system that is flexibly read and executed from a SD/microSD card, the C3 and C4 prototype configurations were based on Arduino and ESP8266 boards that have programs and configurations stored on onboard flash memories, compiled



(a) C1.



(b) C3.



(c) C4.

Fig. 7. Development schemes for the prototypes.

and stored there through proper IDE (Integrated Development Environment). This way, the updating procedure has a limitation in the microcontroller-based prototype configurations due to the impossibility of uploading a new firmware on the flash memory while it executes the current firmware intermittently. Therefore, for the C3 and C4 prototype configurations, it is recommended the execution of the MSensorMob framework as proposed by us in [5], which does not implement the UPDATING state.

When the unit is not in the IDLE state, the MONITORING state is expected to be the most used state in a normal operation. Actually, as defined in our previous work in [5], every unit will be configured with a monitoring frequency *fs*, which will be read during the INITIALIZE state and eventually altered during the UPDATING state. Such sampling rate will indicate that, every *fs* seconds, a new sample procedure will be initiated, resulting in the generation of a sample file composing all monitored data retrieved from all active sensor devices attached to the MSensorMob/MSensorMob2 unit. For that, however, since different hardware configurations may demand different processing times when executing a whole sampling procedure, we have to define t_h as the required sampling processing time for the hardware configuration (prototype) *h*. This way, in order to maintain a strict periodic monitoring frequency, the time between the completion of a monitoring sampling procedure and the next sampling will be equal to $(fs - t_h)$, assuming $fs > t_h$. Therefore, for very small values of *fs*, the minimum acceptable t_h time may be prohibitive for some hardware configurations.

Fig. 9 presents the processing time of an entire sampling procedure, as well the measured energy consumption during that period.

The processing time for the MONITORING state will depend on multiple parameters associated to the hardware configuration of the boards, the software components and the employed sensor devices. For the implemented prototypes, the Rasp-

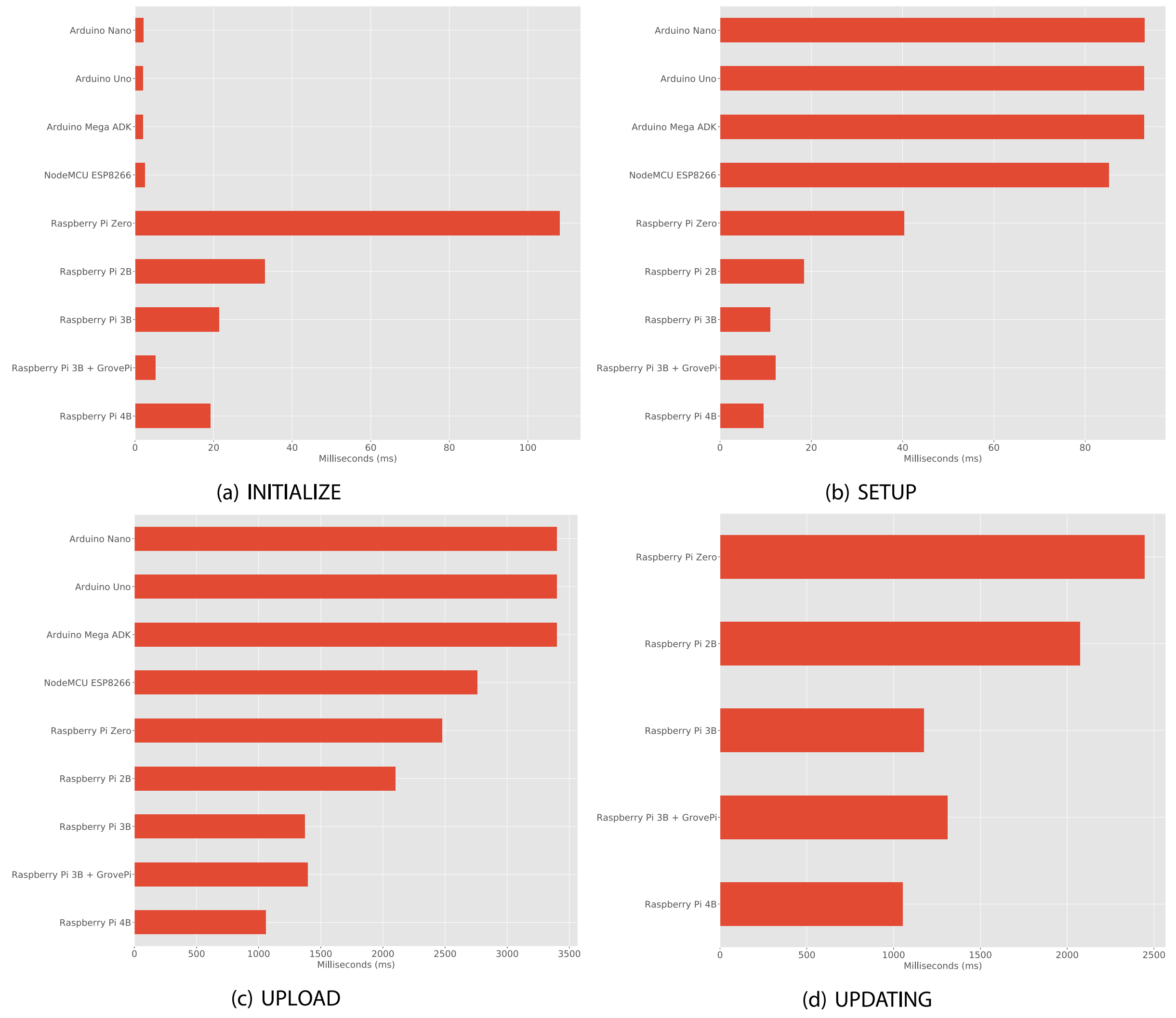


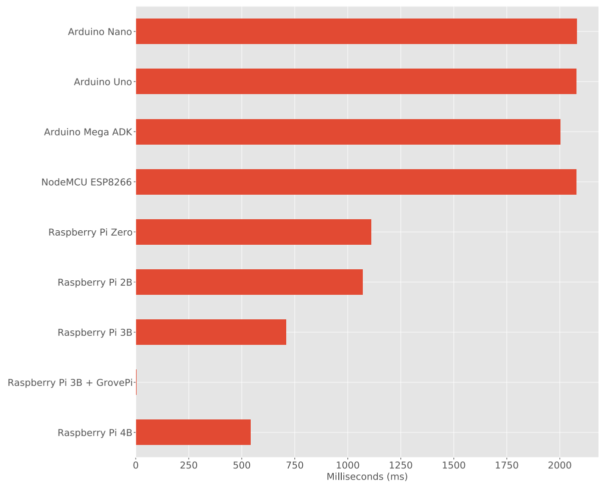
Fig. 8. Time spent in each of the states, excepting MONITORING.

berry Pi boards performed better in terms of processing time, probably due to their enhanced processing power and I/O structure, but they consumed more energy. As an interesting remark, the C2 configuration (Raspberry Pi 3 with GrovePi+) had a very low value of t_h, which is a reflection of the exclusive sensors integration; for the traditional DHT22, which is a temperature/humidity sensor, the reading time is high, but the corresponding Grove DHT11 temperature/humidity sensor operates much faster due to its characteristics and the dedicated microcontroller in GrovePi + HAT for sensors readings.

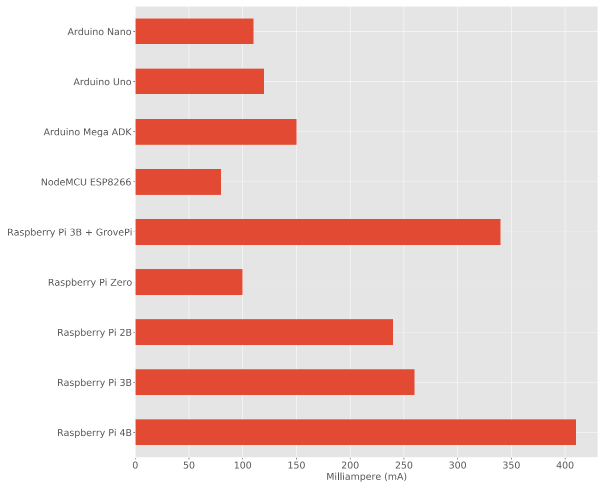
After execution of the tests with the implemented prototypes, we achieved different results that could support the choosing of the best hardware configurations when creating MSensorMob2 units. Actually, although elapsed times and consumed energy are relevant information, there are other important parameters that should be also considered, such as combined costs, dimensions, programming libraries and purchase availability. In this sense, after careful analyses, we concluded that the hardware configuration based on the Raspberry Pi Zero WH board had the better average performance when combining all aforementioned parameters, being indicated for usual multi-sensor monitoring functions in multiple scenarios. However, since this is not a strict recommendation, any other configurations could be adopted, but we favor the ones that can implement all sates of the MSensorMob2 framework.

6. Deployment and security issues

The development of multi-sensor units may be an important breakthrough for the designing and operation of a lot of emerging IoT applications. In this context, sensing applications targeted at the concurrent monitoring of multiple scalar and multimedia data may benefit from the adoption of sensor units that encompass multiple sensor devices. Actually, the



(a) Value of t_h for the defined prototypes.



(b) Energy consumption during the MONITORING state.

Fig. 9. Performance comparisons during the MONITORING state.

scope of such type of applications is wide enough to foster the development of flexible multi-sensor units, with special attention to applications such as emergencies detection [10], smart farming [11], urban monitoring [12], among others.

The performed implementations, experiments and analysis in this article are important contributions to support such scenarios, indicating actual possibilities for real-world applications. However, beside budget, hardware configurations and the operation of the proposed framework, other issues should be also considered when exploiting the flexibility of multi-sensor units. And these issues are particularly relevant when taking into account the mobile nature of a monitoring scenario. As an example, the work in [12] adopts multi-sensor units to monitor environmental variables associated to the quality of cycle paths concerning the health and safety of cyclists. In that work, the units are expected to be attached onto bikes, which naturally adds physical concerns about the proper position of the units, as well as their firm attachment to the bikes (units should not fall on the ground while a bike is moving). In a different perspective, the use of directional sensors like cameras also impose some challenges concerning the position of the cameras and the achieved Field of View. Additionally, obstacles and coverage failures may also jeopardize the effectiveness of the deployed multi-sensor units. In all these cases, the proper planning and execution of deployment strategies, as well as the physical characteristics of the units (as previously discussed), should also be part of the implementation and management details of any proposed approach for mobile multi-sensors monitoring.

Other relevant issue is security. Multi-sensor units will typically be part of a networked system, providing sensed data and sometimes reacting to different types of requests. In the proposed MSensorMob2 framework, for example, while the units are able to transmit stored sensed data using a wireless connection during the UPLOADING state, they also react to re-configuration request in the UPDATING state. In general, for flexible configurable monitoring units in a lot of applications, such behavior may be common, specially for systems designed to operate (and react) during long periods of time. As a consequence, such systems may face different security threats, which may compromise the effectiveness of the systems in unpredictable ways [13]. Actually, although a myriad of solutions have been proposed to enhance the security of Internet-based systems, with efficient solutions to provide authenticity, integrity and confidentiality, IoT systems may behave differently, specially in mobile monitoring scenarios. Moreover, dependability and availability concerns may also emerge in some cases, depending on the networking infrastructure and the level of failures that are expected to be supported by the employed solution. Thus, all these issues should be considered carefully, in order to provide acceptable levels of security/dependability while not reducing the attainable performance of the systems [14,15].

Therefore, although we provided important information to select the proper hardware platform and configurations of the required electronic components, there are other issues that should be considered when designing, implementing, operating and maintaining any monitoring system based on multi-sensor units. Depending on the particularities of the applications, such considerations may also impact the choosing of the hardware components, further extending the conclusions of this work.

7. Conclusions

The development of multi-sensor units may be an important breakthrough for the maturation of emerging IoT applications, specially when hardware frameworks are created to support the development of such units. Nevertheless, the development of architectures and firmwares should be also combined with the analyses of implementation, deployment and performance issues, bringing hardware frameworks to a more realistic setting. In this article, open-source hardware platforms were considered to support the creation of multi-sensor units for mobile applications with long disconnection periods. For that, a previously proposed framework was adopted as reference, being also extended to incorporate a new updating functionality. The resulting designed and implemented prototypes are then good practical examples of how monitoring units following the proposed framework could be implemented.

As future works, we want to further evaluate the construction of MSensorMob2 units, adopting different hardware boards. Additionally, 3D case printing will be also considered when assessing deployment issues of the units. Finally, security issues of connected units will be evaluated and discussed in future experiments.

Funding

This work was partially supported by the Brazilian Agency CNPq (grant 407903/2018-6).

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.

References

[1] M.O. Ojo, S. Giordano, G. Procissi, I.N. Seitanidis, A review of low-end, middle-end, and high-end iot devices, IEEE Access 6 (2018) 70528–70554, https://doi.org/10.1109/ACCESS.2018.2879615.
[2] J.M. Pearce, Economic savings for scientific free and open source technology: a review, HardwareX 8 (2020), https://doi.org/10.1016/j.ohx.2020.e00139 e00139.
[3] T. Lieske, B. Pfundt, S. Vaas, M. Reichenbach, D. Fey, System on chip generation for multi-sensor and sensor fusion applications, in: International Conference on Embedded Computer Systems: Architectures, Modeling, and Simulation (SAMOS), 2017, pp. 20–29, https://doi.org/10.1109/SAMOS.2017.8344607.
[4] M.G. Samaila, J.B.F. Sequeiros, T. Simões, M.M. Freire, P.R.M. Inácio, IoT-HarPSecA: A Framework and Roadmap for Secure Design and Development of Devices and Applications in the IoT Space, IEEE Access 8 (2020) 16462–16494, https://doi.org/10.1109/ACCESS.2020.2965925.
[5] F. Oliveira, D.G. Costa, I. Silva, P. Andrade, A. Dias, MSensorMob: A Multi-Sensors Hardware Framework to Support the Development of Adaptable Monitoring Units in Mobile Applications. in IEEE International Workshop on Metrology for Industry 4.0 & IoT (MetroInd4.0&IoT), 2021, pp. 648–653. https://doi.org/10.1109/MetroInd4.0IoT51437.2021.9488435..
[6] D.G. Costa, C. Duran-Faundez, Open-source electronics platforms as enabling technologies for smart cities: recent developments and perspectives, Electronics 7 (12) (2018), https://doi.org/10.3390/electronics7120404.
[7] N. Kaur, S.K. Sood, An Energy-Efficient Architecture for the Internet of Things (IoT), IEEE Syst. J. 11 (2) (2017) 796–805, https://doi.org/10.1109/JSYST.2015.2469676.
[8] W. Rafique, X. Zhao, S. Yu, I. Yaqoob, M. Imran, W. Dou, An application development framework for internet-of-things service orchestration, IEEE Internet Things J. 7 (5) (2020) 4543–4556, https://doi.org/10.1109/JIOT.2020.2971013.
[9] G. Vieira, J. Barbosa, P. Leitão, L. Sakurada, Low-cost industrial controller based on the raspberry Pi platform, in: IEEE International Conference on Industrial Technology (ICIT), 2020, pp. 292–297, https://doi.org/10.1109/ICIT45562.2020.9067148.
[10] D.G. Costa, F. Vasques, P. Portugal, A. Aguiar, A distributed multi-tier emergency alerting system exploiting sensors-based event detection to support smart city applications, Sensors 20 (170) (2020), https://doi.org/10.3390/s20010170.
[11] M. Mahbub, A smart farming concept based on smart embedded electronics, internet of things and wireless sensor network, Internet Things 9 (2020), https://doi.org/10.1016/j.iot.2020.100161 100161.
[12] F. Oliveira, D.G. Costa, L. Lima, I. Silva, iBikeSafe: a multi-parameter system for monitoring, evaluation and visualization of cycling paths in smart cities targeted at cycling adverse conditions, Smart Cities 4 (4) (2021) 1058–1086, https://doi.org/10.3390/smartcities4030056.
[13] X. Li, P. Jiang, T. Chen, X. Luo, Q. Wen, A survey on the security of blockchain systems, Future Gener. Comput. Syst. 107 (2020) 841–853, https://doi.org/10.1016/j.future.2017.08.020.
[14] J. Leng, M. Zhou, J.L. Zhao, Y. Huang, Y. Bian, Blockchain security: a survey of techniques and research directions, IEEE Trans. Serv. Comput. (2020), https://doi.org/10.1109/TSC.2020.3038641.
[15] M.A. López-Peña, J. Díaz, J.E. Pérez, H. Humanes, DevOps for IoT systems: Fast and continuous monitoring feedback of system availability, IEEE Internet Things J. 7 (10) (2020) 10695–10707, https://doi.org/10.1109/JIOT.2020.3012763.

Franklin Oliveira received the B.Sc degree in computer engineering from the State University of Feira de Santana, UEFS (2019), Brazil, with an research period at the University of Porto, Portugal (2018/2019), and a M.Sc degree in Computer Science also from UEFS (2021). He has acted as a researcher at the Advanced Applications and Networks Lab (LARA), contributing to the development of Embedded Systems and applications focused on Internet of Things for Smart Cities.

Daniel G. Costa is an Associate Professor at the Department of Technology of the State University of Feira de Santana, Brazil. He got a D.Sc. degree in Electrical Engineering at Federal University of Rio Grande do Norte, Brazil, in 2013, with research internship at the University of Porto, Portugal. A B.Sc. degree in Computer Engineering and a M.Sc. degree in Electrical Engineering were both obtained from Federal University of Rio Grande do Norte in 2005 and 2006. He has served many committees of distinguished international conferences and acted as reviewer for high-quality journals. He is author or co-author of more than 100 papers in the areas of computer networks, industrial communication systems, Internet of Things, smart cities and sensor networks. Daniel currently coordinates the Advanced Networks and Applications Lab (LARA) at the State University of Feira de Santana.

Ivanovitch Silva received the B.S. (2006), M.Sc. (2008), and Ph.D. (2013) degrees in Electrical and Computer Engineering from the Federal University of Rio Grande do Norte (UFRN), Brazil and a co-participation from the University of Porto (Sandwich). Additionally, he concluded a shorttechnological course in Big Data & Social Analytics by the Massachusetts Institute of Technology (MIT, 2016), and IEEE Senior Member (2021). Since 2013 is a professor at Digital Metropolis Institute of UFRN. In 2021 was nominated as a Visiting Professor in the Universitá degli Studi di Brescia, Italy. He is currently working as vice-coordinator of the Graduate Program in Electrical and Computer Engineering (PPgEEC) of UFRN. His research interests include scientific modeling and analysis, Internet of Intelligent Vehicles, Industry 4.0, and sustainable cities.