information

Article

Performance Assessment of ESP8266 Wireless Mesh Networks

Luís Santos [1], Tiago Costa [1], João M. L. P. Caldeira [1,2] and Vasco N. G. J. Soares [1,2,*]

[1] Instituto Politécnico de Castelo Branco, Escola Superior de Tecnologia, Avenida do Empresário, 6000-767 Castelo Branco, Portugal; luis.santos1@ipcbcampus.pt (L.S.); tiago.costa1@ipcbcampus.pt (T.C.); jcaldeira@ipcb.pt (J.M.L.P.C.)
[2] Instituto de Telecomunicações, Universidade da Beira Interior, Rua Marquês d'Ávila e Bolama, 6201-001 Covilha, Portugal
* Correspondence: vasco.g.soares@ipcb.pt

Abstract: This paper presents a wireless mesh network testbed based on ESP8266 devices using painlessMesh library. It evaluates its feasibility and potential effectiveness as a solution to monitor perishable goods, such as fresh fruit and vegetables, which are often stored and transported inside refrigerated containers. Performance testing experiments with different numbers of nodes and traffic loads and different message payload sizes are conducted under unicast transmission. The impact on network performance is evaluated in terms of delivery ratio and delivery delay, which, consequently, affect the energy consumption and, hence, network lifetime. The results of this investigation are an important contribution to help researchers to propose mechanisms, schemes, and protocols to improve performance in such challenging networks.

Keywords: wireless mesh networks; testbed; performance assessment; painlessMesh; ESP8266

Citation: Santos, L.; Costa, T.; Caldeira, J.M.L.P.; Soares, V.N.G.J. Performance Assessment of ESP8266 Wireless Mesh Networks. *Information* **2022**, *13*, 210. https://doi.org/10.3390/info13050210

Academic Editor: Abderrezak Rachedi

Received: 24 February 2022
Accepted: 19 April 2022
Published: 20 April 2022

1. Introduction

The contribution of wireless mesh networks (WMNs) [1], in the fast-growing context of the Internet of Things (IoT), extends from agriculture to the most demanding industries and services, collecting and processing timely data and aiming to provide real-time information to help decision making [1]. They are a viable, low-cost, and highly scalable tool, making them a valuable alternative for this type of implementation [2]. Mesh networks are characterized by the aggregation of nodes in a mesh topology, making them actively involved in data transmission, regardless of whether they are switches, bridges, or another device [3]. This lack of hierarchy, which contrasts with commonly used topologies, such as star topologies, makes the network more efficient and fault tolerant, since the nodes are not dependent on each other [4]. It is also extremely easy to add new nodes to an existing network since it is not necessary to change the rest of the network [5].

Cold chains, which are the refrigerated portions of the supply chain, help keep products fresh, nutritious, and safe as they are moved from farm to fork, thus, improving food security, reducing food waste, and enhancing income for farmers [6]. Cold chains can benefit from IoT adoption. The PrunusPós project [7] aims to contribute to this goal by studying the optimization of processes of storage cold conservation, active and/or intelligent packaging, and traceability of food quality in the post-harvest stage of endogenous stone fruit products of the Beira Interior region of Portugal (e.g., cherry and peach). Among other objectives, this project: (1) experimentally characterizes the storage, conventional cold conservation, and packing in the post-harvest stage of cherry and peach and studies innovative techniques for this sector; (2) quantifies reference operation times and parameters in the different stages of conservation and storage in order to extend the shelf life of fruit products, using experimental evaluation in cold storage chambers with controlled atmosphere and packaging with modified atmosphere and numerical simulation; (3) develops a provisional computational tool that, through the function of different operative

parameters and specificity of the process, allows the optimization of the permanence time in each stage of the conservation and storage process, assuring the quality of the product.

Accordingly, it is necessary to develop a wireless mesh sensor network (WMSN) in the context of the PrunusPós project that allows the collection of data about the fruits' storage temperature and relative humidity, as well as the storage duration. The evolution of cherry and peach quality depends directly on these parameters.

The ESP8266 [8] devices with integrated temperature and humidity sensors are selected to be used on the PrunusPós project WMSN. This hardware is chosen because it fulfils the requirement to be a simple, low-cost, effective solution. The ESP8266 devices are incorporated inside the innovative fruit crates (i.e., intelligent packages) also developed in the context of PrunusPós project to prolong fruit life. The devices communicate wirelessly, forming a mesh network which gathers and sends data to a server on the Internet. Some of the fruit crates are removed from the refrigerated truck load during transport for delivery to different stores. Thus, not all sensor nodes are always present in the network.

The ESP8266 [8] devices are commonly used in many projects together with the painlessMesh library [9], which is a tool available for the Arduino [10] programming environment that implements layer 3 (network layer) of the open system interconnection (OSI) model. This library allows nodes to exchange unicast or broadcast messages in a WMN. The network creation and connectivity are managed by the library itself. It is only necessary to configure the devices to have a matching service set identifier (SSID).

A previous work from other authors [11] attempted to measure the performance of ESP8266 devices using the painlessMesh library. However, we argue that the study was incomplete and unclear and lacked an in-depth explanation of the setup and configurations used. Moreover, the implementation and initial tests of the PrunusPós project WMSN showed different results from the ones presented in that work [11].

Therefore, this paper aims to address the previously identified shortcomings. It presents a wireless mesh network testbed setup and provides a measurement-based performance evaluation. Performance experiments are conducted with different numbers of nodes and traffic loads and different message payload sizes using unicast communications. The impact on network performance is evaluated in terms of delivery ratio and delivery delay. This study will guide the future development of mechanisms, schemes, and protocols that provide efficient bandwidth utilization and energy-efficient computing for the PrunusPós project WMSN. It is crucial to reduce the need for human intervention to recharge the batteries of the devices incorporated inside the innovative fruit crates.

The rest of the paper is organized as follows. Section 2 presents the concept of wireless mesh networks and introduces the painlessMesh library. Section 3 describes the technical details of the testbed hardware implementation and configuration and presents the algorithms implemented. Section 4 presents the performance evaluation results, together with a results discussion. Finally, Section 5 concludes the paper and provides guidelines for future work.

2. Wireless Mesh Networks

Wireless mesh networks [1] are dynamically self-organized and self-configured. The nodes in the network (with sensor, communication, storing, and processing abilities) automatically establish an ad hoc network and maintain the mesh connectivity [12].

WMNs can provide broadband communication without relying on a wired backhaul infrastructure. This has the benefit of rapid and low-cost network deployment. WMNs remain functional even if one of the nodes fails, since they are not dependent on each other. It is also extremely easy to add new nodes without having to change the rest of the network [3,5]. WMNs differ from traditional topologies found in wired networks, such as star or tree topologies, in that they do not form a hierarchy between devices, such as switches and bridges. This way, all nodes participate in data transmission, making the network more efficient and more fault tolerant [4,13].

WMNs were originally developed for military applications but have gained considerable popularity and now have numerous applications for the environment, health, home, and commerce [14]. They can even be used for industrial wireless networks, city-wide broadband Internet access, all-wireless offices, and even rural networks [15,16].

In spite of the increasing interest in the study of WMNs, many challenges still need to be addressed. Many concerns have been raised about the data link layer. Media access control (MAC) protocols are used for single-hop communication, so this layer should be designed in such a way that allows multipoint communication between nodes [17–19]. Scalability and quality of service (QoS) are other challenges [20]. Data transmission using the transmission control protocol (TCP) also seems to be a problem for single-hop communication, which becomes worse in this type of network [17].

This paper provides a description of a WMN testbed setup and a measurement-based performance evaluation. The results analysis provides important insights for PrunusPós [7], a smart precision agriculture project, which aims to extend the shelf life of peaches and cherries in the Beira Interior region in Portugal. These fruits are highly seasonal and deteriorate rapidly after harvest. Storage under controlled temperature and humidity can slow down their decay, but even slight variations may compromise this process. In the PrunusPós project, sensors are integrated into the crates or other containers used to store and transport the fruit, which allows the formation of a WMN to provide continuous feedback on the ambient conditions. This system facilitates individualized data collection from the moment the fruits are packaged to their delivery.

The testbed is based on ESP8266 devices [8] connected in a wireless mesh network using the painlessMesh library version 1.4.9 [10,21], which is a library available for the Arduino programming environment [22]. The painlessMesh library takes care of creating a simple WMN using ESP8266 and ESP32 hardware. The programmer does not have worry about how the WMN is structured or managed, as it automatically connects nodes with the same SSID. The painlessMesh is not IP networking; it does not create a TCP/IP network of nodes. It is implemented as a layer-3 protocol (i.e., network layer of the OSI model), connectionless and without confirmation. Nodes are not identified by their Internet protocol (IP) address, but, instead, by a 32-bit address taken from their MAC address. Each node knows the network topology from the neighboring nodes, which is updated every 3 s (Figure 1). The messages exchanged between network nodes can be transmitted through broadcast or unicast and use the JavaScript object notation (JSON) format. The use of JSON format facilitates user comprehension and integration into web applications but may decrease network performance [9].

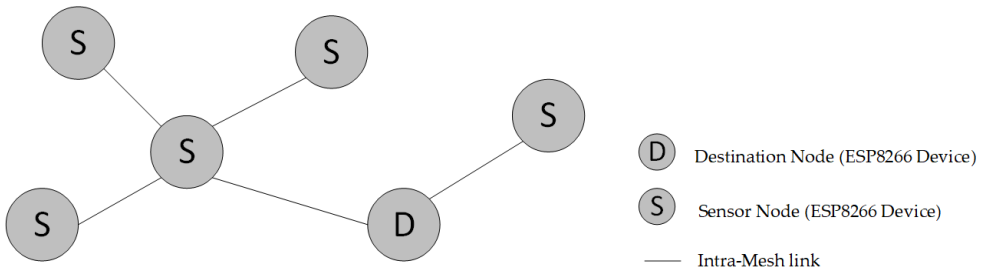

Figure 1. Example of a painlessMesh topology.

3. Testbed

The testbed implemented in this study was based on ESP8266 devices shown in Figure 2. These devices have 16 MB of Flash memory and a built-in wi-fi antenna, which transmits at a frequency of 2.4 GHz [8]. It is also possible to connect an external antenna to these devices to amplify the signal. In terms of processing, the ESP8266 operates at 80 MHz or 160 MHz frequency, set according to the programmer's selection [23]. These devices also have a battery connector so they can operate in standalone mode [23].

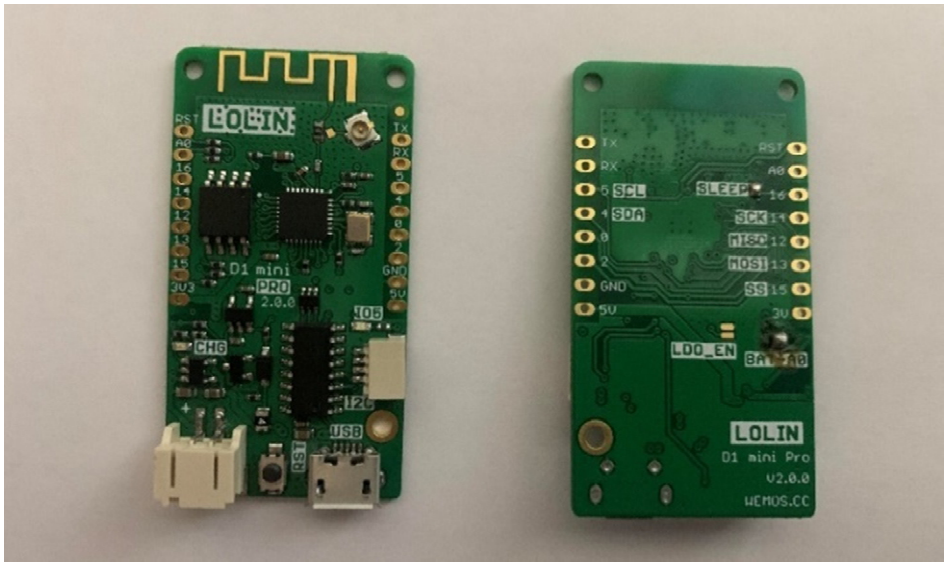

Figure 2. ESP8266 device.

Despite their small size (48 × 25.4 mm), the ESP8266 devices find a wide range of implementations in the field of the Internet of Things. Some examples range from small projects for smart irrigation systems [24], measuring energy consumption [25], or fire-detection alarms [26] to more robust solutions such as synchronizing information on the availability of charging stations for electric vehicles (facilitating their scheduling during trips) [27], the control of meteorological data in photovoltaic stations [28], air quality control systems in metallurgical factories [29], or in scenarios of disease prevention related to COVID-19 [30].

The configuration and programming of these devices was performed through the Arduino IDE [22]. It is also possible to program them using the MicroPython [31] or NodeMCU [32] environments [23]. Flowcharts are presented next to illustrate the configuration process based on the node type (i.e., sensor or destination) and the network settings to be configured (e.g., payload, number of messages per second, or *IwIP Variant*).

After installing the painlessMesh library in the Arduino IDE, the first step was to choose the parameters that are set on all nodes to form the mesh network: the SSID, the port, and the password to access the network. The port needed to be selected so as to not conflict with those used by well-known applications and protocols. The steps are illustrated in Figure 3.

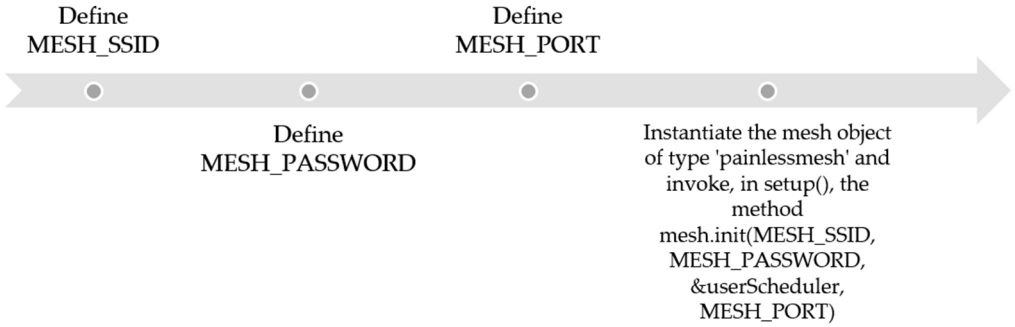

Figure 3. Initial setup process of a painlessMesh network.

The configuration of the *IwIP Variant* shown in Figure 4 is another important parameter that requires attention, due to its potential impact on the network performance. It allows the selection of one of two options: lower memory or higher bandwidth. The first one considers a maximum segment size (MSS) of 536 bytes. The second one considers a MSS of

1460 bytes, thus, requiring more computational and memory resources. MSS is defined as the largest size that a TCP payload can have after subtracting the header sizes [33]. Both options consider a maximum transfer unit (MTU) of 1480, which is the maximum size that each IP payload can be including IP and TCP headers [8].

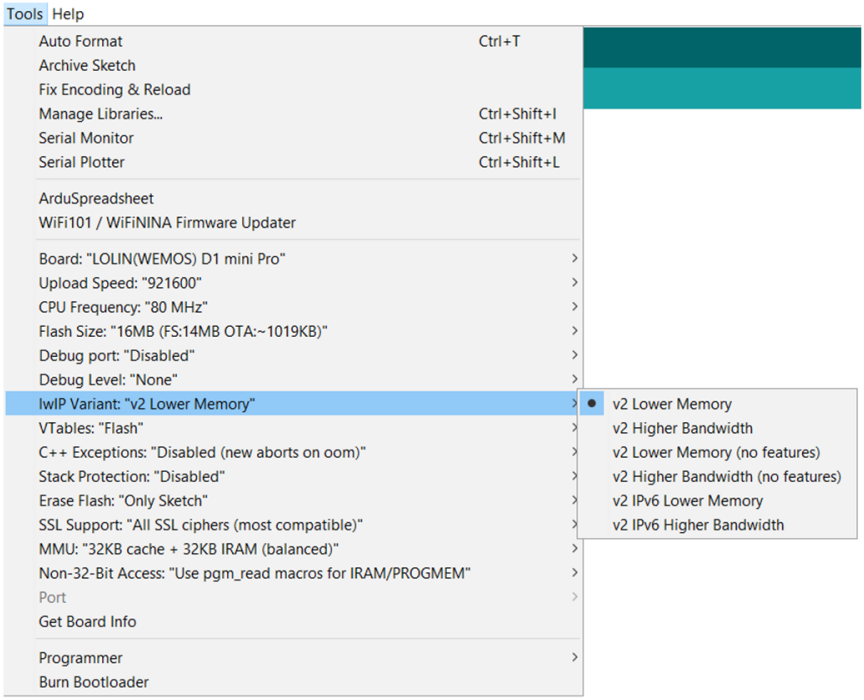

Figure 4. lwIP Variant configuration on Arduino IDE.

Due to the requirements of the PrunusPós project, it was assumed that a network composed of one or more sensor nodes and a single destination node was required, as illustrated in Figure 1. The use of painlessMesh library guaranteed that each sensor node is able to connect to several other sensor nodes and/or to the destination node, thus, forming the mesh, and, if one drops out of the network, its neighbors simply find another route. The drop out of sensor nodes may occur due to battery depletion, a hardware problem, or if some sensor nodes, which are integrated in the fruit pallets, leave the truck (e.g., fruit delivery to a market). The following describes the algorithms implemented in the testbed to evaluate message delivery ratio and the transmission delay between different numbers of sensor and one destination node.

3.1. Message Exchange

Figure 5 illustrates the process of pairing one or more sensor nodes and the destination node and the flow of message exchange. Figure 6 describes the algorithm used for message exchange. After setting the parameters of the experiment (i.e., message send rate, payload size, and type of communication), sensor(s) and destination nodes paired. Then, message transmission was initiated and executed during a predefined period of time. Both sensor(s) and destination nodes counted the number of messages successfully sent or received during that time. The destination node had additional time to process and count them. The count was printed at the end. Unicast messages were sent using the *sendSingle()* method from the painlessMesh library. This method needs to identify the ID of the destination node.

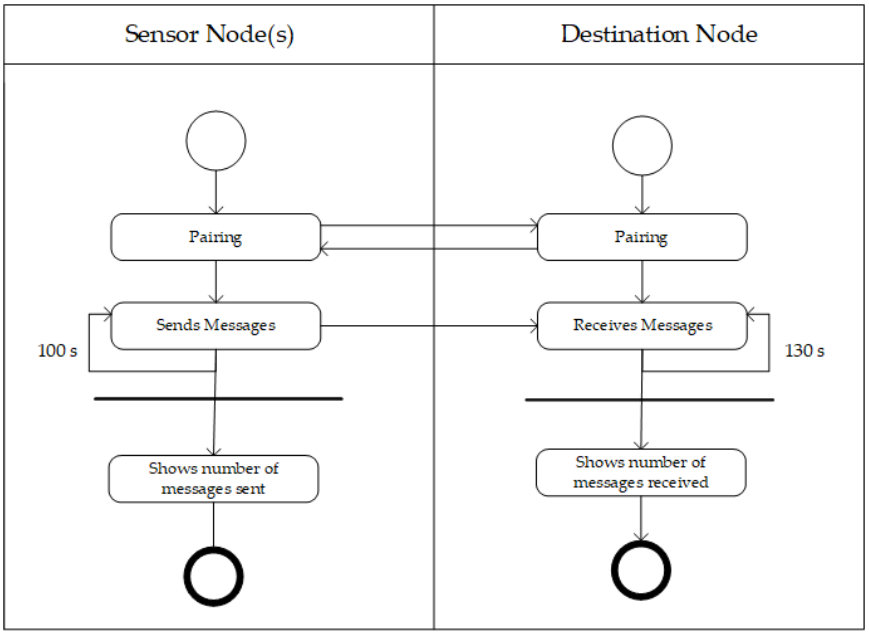

Figure 5. UML modelling of message exchange.

Message exchange and count

1. ***Setting the parameters:***
   - Set the message send rate (i.e., messages per second).
   - Set the message payload size (i.e., number of bytes).
   - Set the type of communication (unicast).
2. ***Pairing sensor and destination:***
   - Destination starts counting down from 130 s.
   - If it's the Sensor's first connection:
     - Sensor starts counting down from 100 s.
     - Sensor enables message sending.
3. ***Message sending task:***
   - If the 100-s period has elapsed:
     - Sensor stops the message sending task.
     - Sensor prints the number of messages sent.
   - Else:
     - Sensor sends message.
     - Sensor increments the number of messages sent.
4. ***Message receiving task:***
   - If the 130-s period has already elapsed:
     - Destination prints the number of messages received.
   - If Destination receives a message:
     - Destination increments the number of messages received.

Figure 6. Algorithm used for message exchange and count.

3.2. One-Way Delivery Delay

Figure 7 illustrates the process of pairing the sensor(s) and destination nodes and the message exchange between them that allowed the determination of the one-way delivery delay. Figure 8 describes the algorithm used to obtain information about the one-way delivery delay. According to painlessMesh API documentation [34], this library provides *startDelayMeas()* and *nodeDelayCallbak_t()* methods to measure the one-way delivery delay. The first method (*startDelayMeas()*) was called by the destination node to send a message to the sensor node. Then, the destination node waited for the sensor node's response to that message. The sensor node, after receiving the message from the destination node, returned a response with the value of delivery delay obtained by the *startDelayMeas()* method. This response message triggered the callback method (*nodeDelayCallbak_t()*) on the destination node, and the delivery delay value was extracted from the received message.

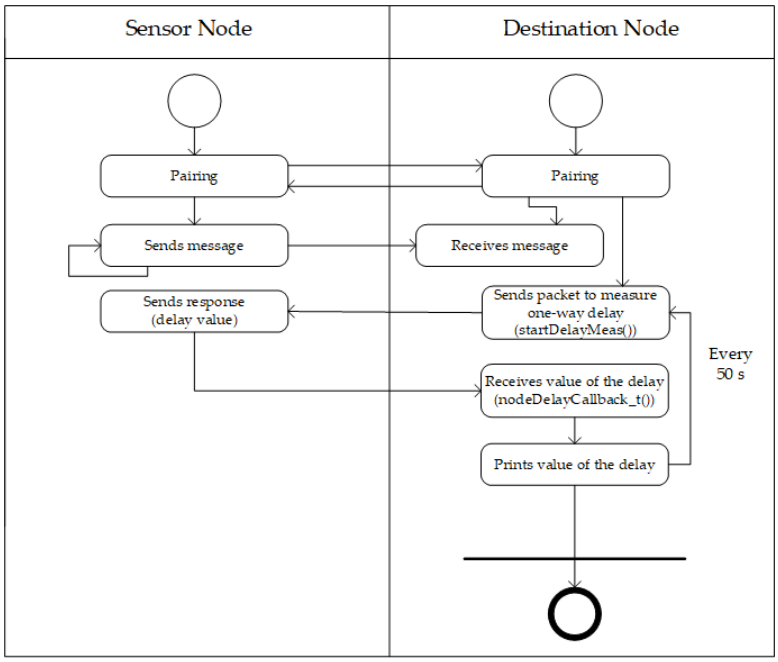

Figure 7. UML modeling for calculating one-way delivery delay.

One-way delivery delay

1. Pairing sensor and destination:

- Destination identifies the nodeId of the sensor nodes it pairs with.

2. Sending the message for delay computing

- Destination sends a delay measurement message to all sensor nodes in the network (*startDelayMeas()*).

3. Receiving the Delay value:

- Destination receives the value of the delay (*nodeDelayCallbak_t()*) from all sensor nodes.
- Destination prints the value of the delay.

Figure 8. Algorithm used for one-way delivery delay calculations.

4. Performance Assessment

This section presents the performance assessment of a WMN testbed based on ESP8266 devices connected using the painlessMesh library. The next subsections describe the testbed setup, the performance metrics considered in this study, and the results analysis and discussion.

4.1. Setup

To not bias the results, all tests were conducted with device configurations requiring the minimum computing and memory requirements. Therefore, the tasks performed by ESP8266 devices were only those strictly necessary for their correct operation. The responsibility for the network mesh connection was delegated to the painlessMesh library, as was assumed by other authors in previous work [11]. It should also be noted that no root node was defined.

Due to the requirements of the PrunusPós project [7], the performance of the WMN was evaluated in the scenarios illustrated in Figure 9. The network topologies presented in Figure 9 should be considered as a mere example, since this task is part of painlessMesh library management. The scenarios were composed of a varying number of sensor nodes (1 to 5), which repeatedly send messages, and a single destination node that attempts to receive them. The nodes were evenly positioned inside an area of 7 m $\times$ 2.5 m similar to the size of a 26 ft truck's refrigerated container.

Different traffic loads (1 to 10 messages per second) and message payload sizes (10, 25, 50, 100, 250, 500, 750, and 2000 bytes) were considered under unicast transmissions. The *taskSendMessage()* method was responsible for sending messages. This method was scheduled in the algorithm at fixed time intervals. Thus, the greater the number of messages to be sent, the shorter that interval was.

The two configurations available for the *IwIP Variant* (MSS = 536 bytes and 1460 bytes) were also tested. All the other settings were set by default, namely, the CPU frequency, which was kept at 80 MHz. The configuration of the devices was performed entirely through the Arduino IDE. The algorithm developed for the nodes was prepared to output data through serial communication. Thus, the serial monitor feature of the Arduino IDE and the PuTTY application [35] were used to present that data.

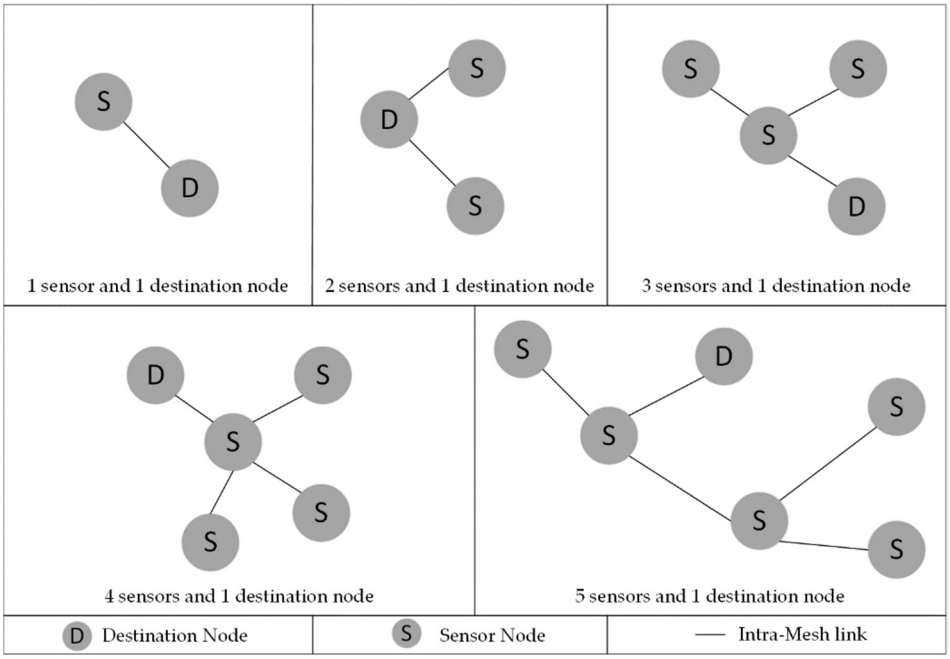

Figure 9. Number of sensor and destination nodes.

4.2. Performance Metrics

The impact on network performance was evaluated in terms of delivery ratio and one-way delivery delay. The delivery ratio is a key performance indicator, as it tells the percentage of successfully received messages. It results from the quotient between the number of messages received by the destination node and the total messages sent by the sensor node. One-way delay is the time it takes for a successful message delivery. Each experiment was run 30 times, and the average results are reported with a 95% confidence interval.

4.3. Results and Discussion

To evaluate the performance assessment, several tests were conducted using the described scenarios and configurations. The following results were obtained for each test and are presented, analyzed, and discussed.

4.3.1. MSS Size Variation (IwIP Variant Option)

The first test performed was intended to determine the delivery ratio as a function of the MSS size: 536 bytes (Figure 10) or 1460 bytes (Figure 11). The scenario used for this test was composed of a sensor node and a destination node (1S1R) exchanging messages.

The MSS = 536 bytes configuration showed the highest consistency in sending one message per second among all payload levels. At a sending rate of five messages per second, there were two significant differences; MSS = 536 bytes achieved better results sending 100 bytes payloads (94.1% against 69.5%), but MSS = 1460 bytes achieved a delivery ratio of over 95% with a 500 bytes payload. MSS = 536 bytes only achieved a delivery ratio of 69.4% at the same level. When the device was sending ten messages per second, MSS = 536 bytes performed better when sending 100 bytes payload messages (41.6%). However, MSS = 1460 bytes showed greater performance when sending 500 bytes payloads (49.0%). There was no statistical evidence demonstrating that there are benefits to using the MSS = 1460 bytes configuration, especially considering, on the contrary, that it requires more computational power from the device, which would surely impact the battery lifetime.

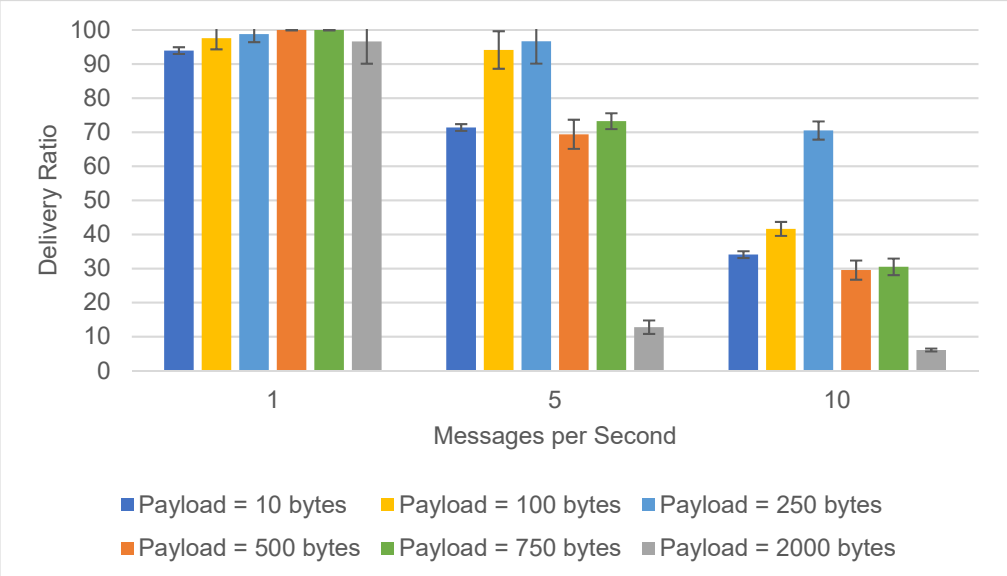

Figure 10. Delivery ratio (%) of messages (1 sensor and 1 destination, MSS = 536 bytes).

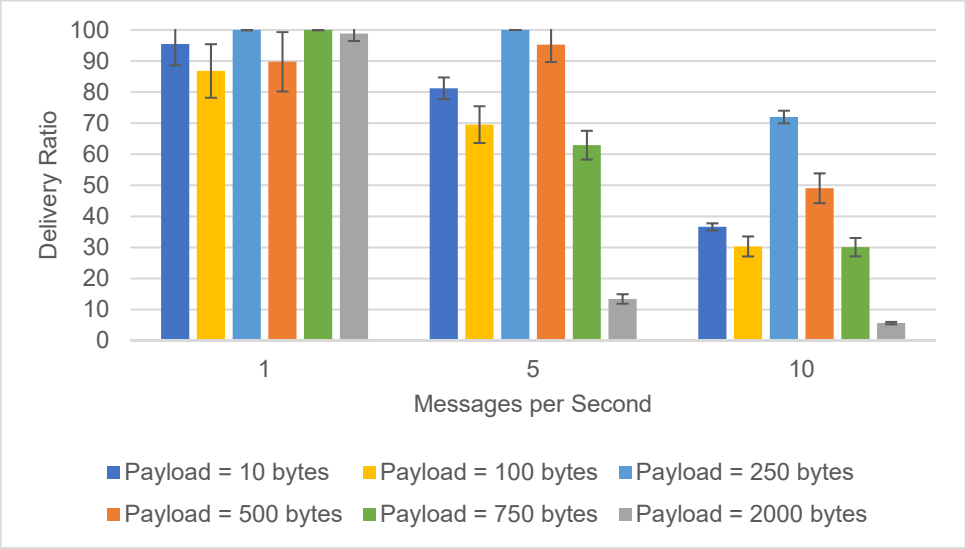

Figure 11. Delivery ratio (%) of messages (1 sensor and 1 destination, MSS = 1460 bytes).

4.3.2. Number of Nodes Variation

The following test focused on the variation of the number of nodes in the network. The results were obtained using the following scenarios: two nodes (1 sensor and 1 destination) exchanging messages (Figure 12); two sensor nodes (2 sensors and 1 destination) sending messages (Figure 13); three sensor nodes (3 sensors and 1 destination) sending messages (Figure 14); four sensor nodes (4 sensors and 1 destination) sending messages (Figure 15); and five sensor nodes (5 sensors and 1 destination) sending messages (Figure 16). All scenarios considered the existence of only one destination node receiving the messages.

The results obtained with a single sensor node and a destination node let us anticipate that the number of messages sent per second has a clear impact on network performance. However, the scenarios with four and five sensor nodes did not lead to a drop in network efficiency as pronounced as the one recorded in scenarios with two and three sensor nodes. This, ultimately, indicates that the balance that the mesh network finds depends on the increase in the number of nodes.

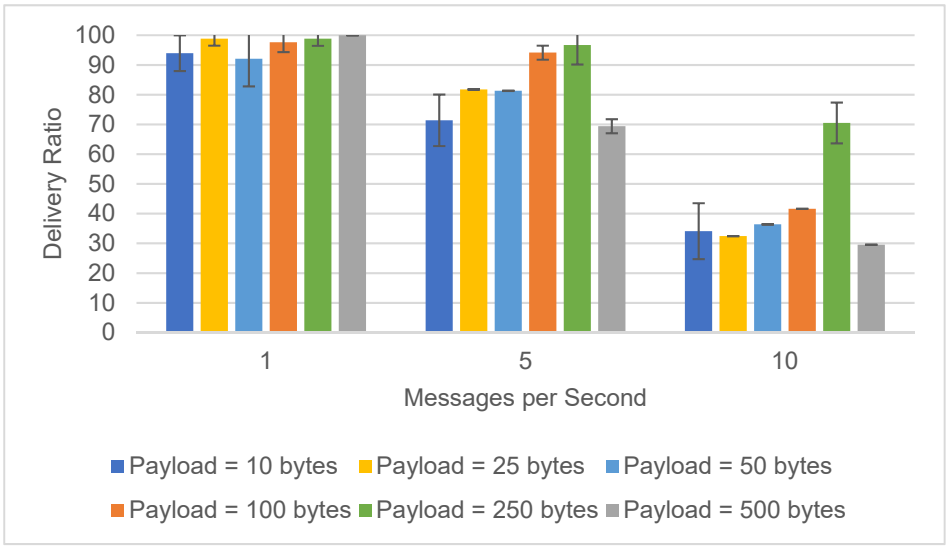

Figure 12. Delivery ratio (%) of messages (1 sensor and 1 destination).

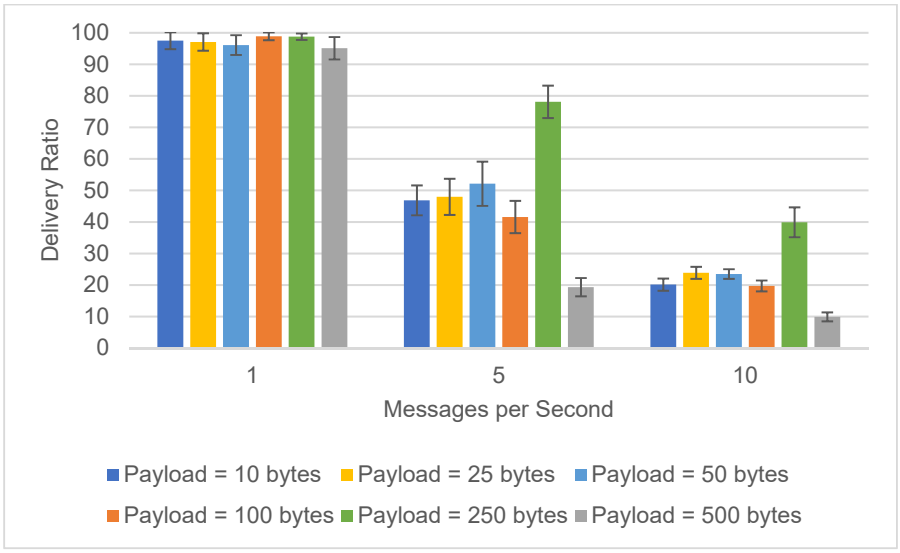

Figure 13. Delivery ratio (%) of messages (2 sensors and 1 destination).

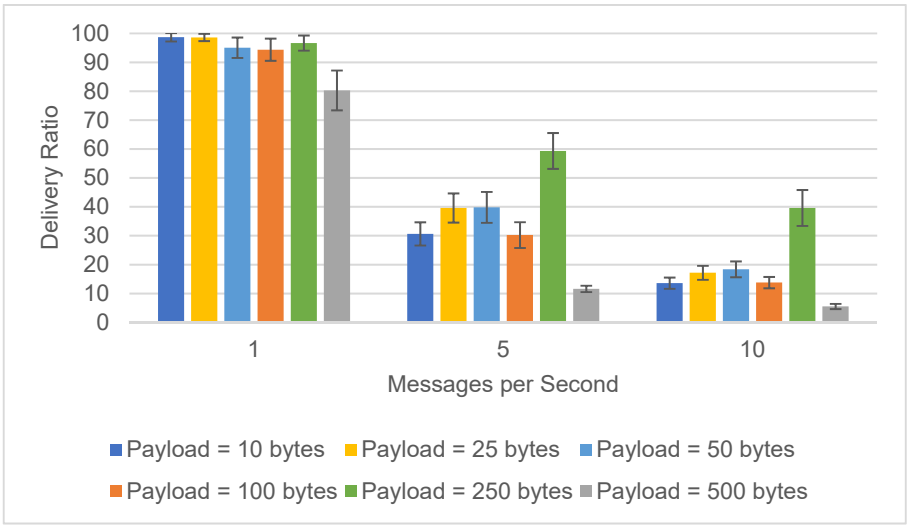

Figure 14. Delivery ratio (%) of messages (3 sensors and 1 destination).

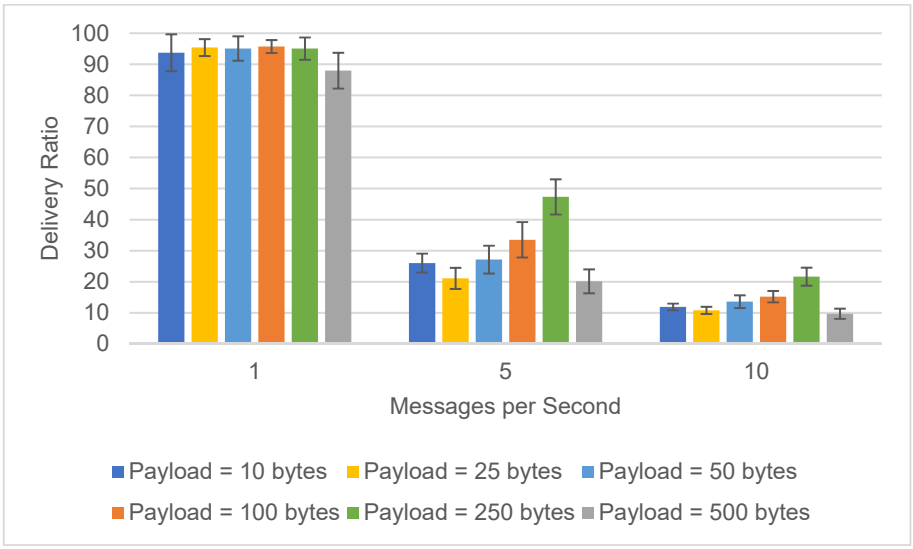

Figure 15. Delivery ratio (%) of messages (4 sensors and 1 destination).

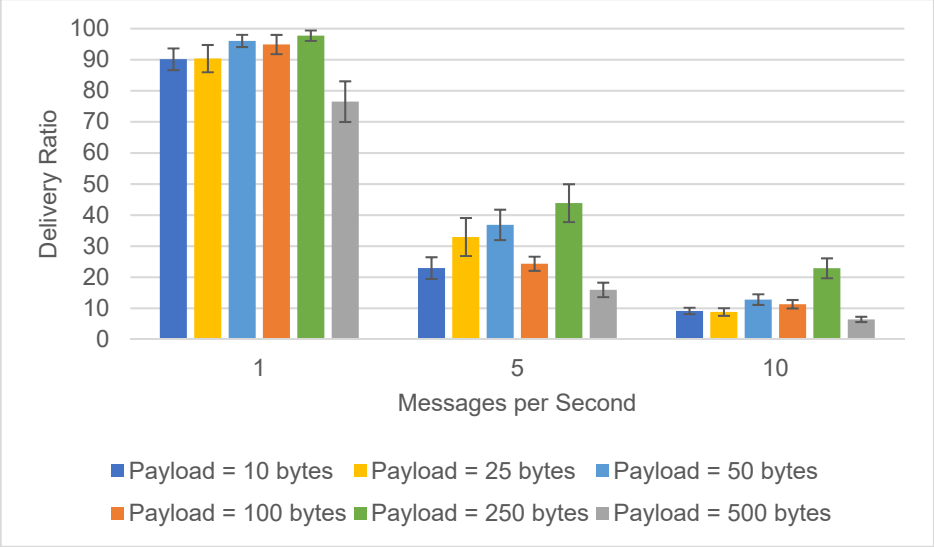

Figure 16. Delivery ratio (%) of messages (5 sensors and 1 destination).

An occurrence common to all scenarios was verified when using 250 bytes payload messages. Regardless of the number of nodes in the network, this was the payload that registered the highest delivery ratio, sending five or ten messages per second. This situation cannot be dissociated from the fact that the packets had a maximum segment size of 536 bytes and were, therefore, the ones that took better advantage of this dimension.

4.3.3. One-Way Delivery Delay

The results for the one-way delivery delay measurement experiments (Figure 17) are presented as follows. This experiment was performed with different numbers of sensor nodes in the network (between 1 and 5). It is important to mention that the mesh network topology was managed by the painlessMesh library itself. To evaluate the impact of network congestion on delivery delay, message exchange between nodes was introduced. The message sending rate varied from no messages exchanged to 1, 5, and 10 messages exchanged per second by each sensor. Due to the previous findings, this experiment used the MSS = 536 bytes configuration and a 250 bytes payload messages.

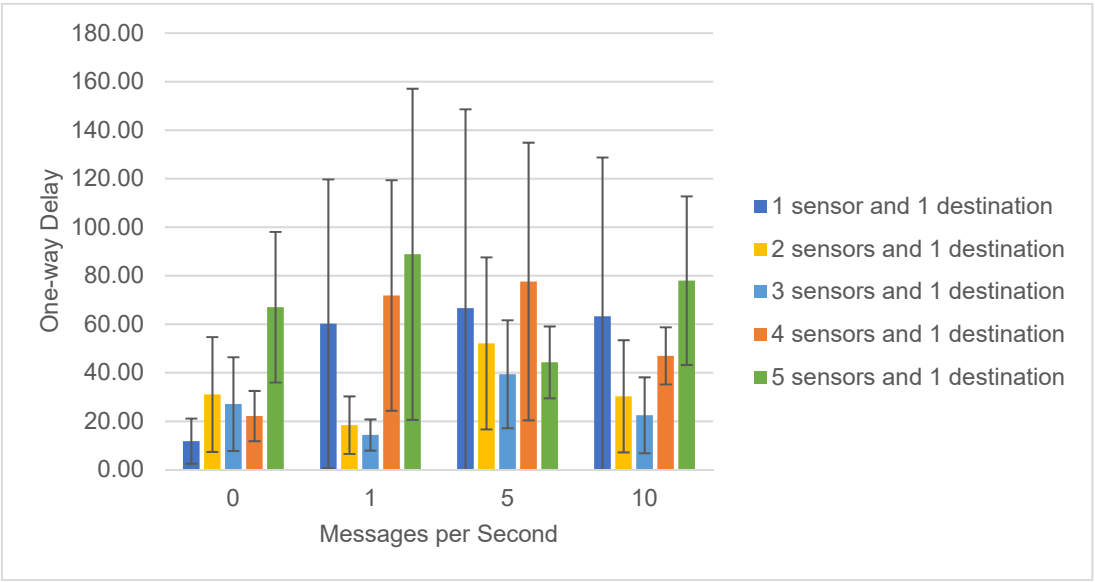

Figure 17. One-way delivery delay (s) as a function of the number of network nodes.

The results showed a clear volatility in the network delivery delay, with a very high confidence error range. There was evidence that the introduction of additional nodes led to increased congestion in the network. In fact, scenarios with four and five nodes generally led to performance losses. The same behavior was observed with the introduction of load on the network, i.e., increasing the number of messages exchanged between nodes. Contrary to expectations, scenarios using only one sensor node always led to very high average delivery delays, especially when compared to scenarios using two or three sensor nodes. Other unexpected results, such as the inconstancy in delivery delay at various network congestion scenarios, can be explained by the uncontrolled network topology during and between tests. It should be remembered that the management of the mesh network is part of the painlessMesh library operation.

5. Conclusions

Wireless mesh networking is a continuously growing technology that has an important role in the future vision of smart agriculture. This paper presented the setup and performance assessment of a wireless mesh network testbed developed within the context of PrunusPós project to collect data such as the temperature and humidity of fruit crates or containers when stored or transported in refrigerated chambers.

The testbed was based on ESP8266 devices connected in a wireless mesh network using the painlessMesh library. An extensive performance evaluation study was conducted with different numbers of nodes and traffic loads and different message payload sizes and IwIP Variants under unicast communication. The impact on network performance was evaluated in terms of delivery ratio and one-way delivery delay.

It was possible to conclude that there is no statistically significant benefit in using messages with larger MSS sizes (536 or 1460 bytes); that would imply greater computational effort. Additionally, the performance results were affected with the increase in the number of sensor nodes, the message send rate, and the message payload size, as expected. The addition of a second and a third sensor nodes to the network had a greater impact on performance than the addition of a fourth and a fifth node. It was also observed that, regardless of the number of sensor nodes in the network, the best performance was always achieved with 250 bytes messages. Regarding the delivery delay, it was possible to conclude that there is high volatility, perhaps justified by the network topology management performed by the painlessMesh library.

This performance assessment study significantly complements and carries forward the previous work from other authors [11]. It serves as basis for future work on mechanisms, schemes, and protocols to improve the performance of the painlessMesh in terms of energy consumption; thus, it enhances the network lifetime of the PrunusPós project wireless mesh sensor network. It may also be interesting to explore the possibility of using vehicular networks to attempt to send the sensed data to a server in the cloud [36,37].

Author Contributions: Conceptualization, L.S. and T.C.; methodology, L.S. and T.C.; validation, V.N.G.J.S. and J.M.L.P.C.; formal analysis, V.N.G.J.S. and J.M.L.P.C.; investigation, L.S., T.C., V.N.G.J.S. and J.M.L.P.C.; resources, L.S. and T.C.; data curation, L.S. and T.C.; writing—original draft preparation, L.S. and T.C.; writing—review and editing, V.N.G.J.S. and J.M.L.P.C.; supervision, V.N.G.J.S. and J.M.L.P.C. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by FCT/MCTES through national funds and, when applicable, co-funded by EU funds under the project UIDB/50008/2020.

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

References

1. Barros, F.S.; Sousa, L.M.; Dias, J.P.; Ferreira, H.S. A Preliminary Study on Mesh Networks and IoT. In *Livro de Resumos do 11.° Encontro de Jovens Investigadores da U.PORTO*; Universidade do Porto: Porto, Portugal, 2018; p. 22.
2. Varandas, L.; Faria, J.; Gaspar, P.D.; Aguiar, M.L. Low-Cost IoT Remote Sensor Mesh for Large-Scale Orchard Monitorization. *J. Sens. Actuator Netw.* **2020**, *9*, 44. [CrossRef]

3. Muhendra, R.; Rinaldi, A.; Budiman, M. Development of WiFi Mesh Infrastructure for Internet of Things Applications. *Procedia Eng.* **2017**, *170*, 332–337. [CrossRef]
4. Cilfone, A.; Davoli, L.; Belli, L.; Ferrari, G. Wireless Mesh Networking: An IoT-Oriented Perspective Survey on Relevant Technologies. *Future Internet* **2019**, *11*, 99. [CrossRef]
5. Mezher, A.M.; Rivera, P.E.; Cárdenas-Barrera, J.; Meng, J.; Guerra, E.C. Efficient strategy to optimize key devices positions in large-scale RF mesh networks. *Ad Hoc Netw.* **2020**, *106*, 102192. [CrossRef]
6. Simões, M.P.A.F.; Veloso, A.; Gaspar, P.D.; Espírito Santo, C.; Silva, P.D.; Andrade, L.P. *PrunusPÓS—Otimização de Processos de Armazenamento, Conservação em Frio, Embalamento Ativo e/ou Inteligente, e Rastreabilidade da Qualidade Alimentar no Pós-Colheita de Produtos Frutícolas*; COTHN—Centro Operativo e Tecnológico Hortofrutícola Nacional, 2021; pp. 404–471. Available online: https://repositorio.ipcb.pt/bitstream/10400.11/7791/1/PrunusPOS-livro%20GO%20de%20Fruticultura%202021.pdf (accessed on 23 February 2022).
7. PrunusPós. GO Prunus Pós. Available online: https://prunospos.webnode.pt/ (accessed on 23 February 2022).
8. Grokhotkov, I. ESP8266 Arduino Core Documentation—Release 3.0.2-30-gbf2882d8. 13 November 2021. Available online: https://readthedocs.org/projects/arduino-esp8266/downloads/pdf/latest/ (accessed on 1 March 2022).
9. GitLab. PainlessMesh Technical Documentation. 24 July 2019. (GitLab). Available online: https://gitlab.com/painlessMesh/painlessMesh/-/wikis/home (accessed on 27 October 2021).
10. Arduino. What Is Arduino? Available online: https://www.arduino.cc/ (accessed on 1 March 2022).
11. Yoppy Arjadi, R.H.; Setyaningsih, E.; Wibowo, P.; Sudrajat, M.I. Performance Evaluation of ESP8266 Mesh Networks. *J. Phys. Conf. Ser.* **2019**, *1230*, 012023. [CrossRef]
12. Krenz, R. IEEE 802.16 wireless mesh networks capacity estimation using collision domains. In Proceedings of the 2009 Second International Conference on Advances in Mesh Networks, Athens, Greece, 18–23 June 2009; pp. 115–119. [CrossRef]
13. Liu, Y.; Tong, K.-F.; Qiu, X.; Liu, Y.; Ding, X. Wireless Mesh Networks in IoT Networks. In Proceedings of the 2017 International Workshop on Electromagnetics: Applications and Student Innovation Competition, London, UK, 30 May–1 June 2017; pp. 183–185. [CrossRef]
14. Köbel, C.; Garcia, W.B.; Habermann, J. A survey on Wireless Mesh Network applications in rural areas and emerging countries. In Proceedings of the IEEE Global Humanitarian Technology Conference, San Jose, CA, USA, 20–23 October 2013; pp. 389–394. [CrossRef]
15. Akyildiz, I.F.; Wang, X.; Wang, W. Wireless mesh networks: A survey. *Comput. Netw.* **2005**, *47*, 445–487. [CrossRef]
16. Akyildiz, I.F.; Wang, X. A survey on wireless mesh networks. *IEEE Commun. Mag.* **2005**, *43*, S23–S30. [CrossRef]
17. Shahdad, S.Y.; Sabahath, A.; Parveez, R. Architecture, issues and challenges of wireless mesh network. In Proceedings of the 2016 International Conference on Communication and Signal Processing (ICCSP), Melmaruvathur, India, 6–8 April 2016; pp. 557–560. [CrossRef]
18. Akintade, O.O.; Yesufu, T.K.; Kehinde, L.O. Development of Power Consumption Models for ESP8266-Enabled Low-Cost IoT Monitoring Nodes. *Adv. Internet Things* **2019**, *9*, 90552. [CrossRef]
19. Cotrim, R.S.; Caldeira JM, L.P.; Soares VN, G.J.; Azzoug, Y. Power saving MAC protocols in wireless sensor networks: A survey. *TELKOMNIKA Telecommun. Comput. Electron. Control* **2021**, *19*, 1778–1786. [CrossRef]
20. Portmann, M.; Pirzada, A.A. Wireless Mesh Networks for Public Safety and Crisis Management Applications. *IEEE Internet Comput.* **2008**, *12*, 18–25. [CrossRef]
21. GitHub. Intro to PainlessMesh. 24 July 2019. (GitHub). Available online: https://github.com/gmag11/painlessMesh (accessed on 1 March 2022).
22. Arduino. Arduino IDE 2 Tutorials. Available online: https://docs.arduino.cc/software/ide-v2 (accessed on 23 February 2022).
23. WEMOS. D1 Mini Pro. Available online: https://www.wemos.cc/en/latest/d1/d1_mini_pro.html (accessed on 23 February 2022).
24. Kumar, K.N.; Pillai, A.V.; Narayanan, M.B. Smart agriculture using IoT. *Mater. Today Proc.* 2021, *in press*. [CrossRef]
25. Kumar, L.A.; Indragandhi, V.; Selvamathi, R.; Vijayakumar, V.; Ravi, L.; Subramaniyaswamy, V. Design, power quality analysis, and implementation of smart energy meter using internet of things. *Comput. Electr. Eng.* **2021**, *93*, 107203. [CrossRef]
26. Mahgoub, A.; Tarrad, N.; Elsherif, R.; Al-Ali, A.; Ismail, L. IoT-Based Fire Alarm System. In Proceedings of the Third World Conference on Smart Trends in Systems, Security and Sustainability (WorldS4), London, UK, 30–31 July 2019. [CrossRef]
27. Savari, G.F.; Krishnasamy, V.; Sathik, J.; Ali, Z.M.; Abdel Aleem, S.H. Internet of Things based real-time electric vehicle load forecasting and charging station recommendation. *ISA Trans.* **2020**, *97*, 431–447. [CrossRef] [PubMed]
28. Pereira, R.I.S.; Jucá, S.C.S.; Carvalho, P. IoT embedded systems network and sensors signal conditioning applied to decentralized photovoltaic plants. *Measurement* **2019**, *142*, 195–212. [CrossRef]
29. Balashanmugam, T.; Elayaraja, P.; Srinivasan, P.; Kumarganesh, S. IoT based air quality measurement and alert system for steel, material and copper processing industries. *Mater. Today Proc.* 2021, *in press*. [CrossRef]
30. Chawla, M.S.; Prakash, D.; Jindal, S. Design of system for measuring air properties for help during COVID-19 scenario. *Mater. Today Proc.* **2021**, *45*, 4472–4476. [CrossRef]
31. George, D. MicroPython. 2018. Available online: https://micropython.org/ (accessed on 23 February 2022).
32. NodeMcu. NodeMcu—Connect Things EASY. 2018. Available online: https://www.nodemcu.com/index_en.html (accessed on 1 October 2021).

33. Borman, D. TCP Options and Maximum Segment Size (MSS), RFC 6691. 2012. Available online: https://datatracker.ietf.org/doc/html/rfc6691 (accessed on 23 February 2022).
34. GitLab. PainlessMesh API. 24 July 2019. (GitLab). Available online: https://gitlab.com/painlessMesh/painlessMesh/-/wikis/api (accessed on 23 February 2022).
35. Putty. Download PuTTY—A Free SSH and Telnet Client for Windows. Available online: https://www.putty.org/ (accessed on 23 February 2022).
36. Fabian, P.; Rachedi, A.; Guéguen, C. Programmable objective function for data transportation in the Internet of Vehicles. *Trans. Emerg. Telecommun. Technol.* **2020**, *31*, e3882. [CrossRef]
37. Rachedi, A.; Badis, H. BadZak: An Hybrid Architecture Based on Virtual Backbone and Software Defined Network for Internet of Vehicles. In Proceedings of the 2018 IEEE International Conference on Communications (ICC), Kansas City, MO, USA, 20–24 May 2018.