electronics

MDPI

Article

CaBIUs: Description of the Enhanced Wireless Campus Testbed of the Ionian University †

Aikaterini Georgia Alvanou [1,*], Alexandros Zervopoulos [1], Asterios Papamichail [1], Konstantinos Bezas [1], Spiridon Vergis [1], Andreana Stylidou [1], Athanasios Tsipis [1], Vasileios Komianos [2], Georgios Tsoumanis [3], George Koufoudakis [1] and Konstantinos Oikonomou [1]

1 Department of Informatics, Ionian University, GR-49100 Corfu, Greece; azervop@ionio.gr (A.Z.); aspapa@ionio.gr (A.P.); kbezas@ionio.gr (K.B.); svergis@ionio.gr (S.V.); astylidou@ionio.gr (A.S.); atsipis@ionio.gr (A.T.); gkoufoud@ionio.gr (G.K.); okon@ionio.gr (K.O.)

2 Department of Audio and Visual Arts, Ionian University, GR-49100 Corfu, Greece; vkomianos@ionio.gr

3 Department of Informatics and Telecommunications, University of Ioannina, GR-47100 Arta, Greece; gtsoum@uoi.gr

* Correspondence: akorina@ionio.gr

† This paper is an extended version of our paper published in the Proceedings of the 4th South-East Europe Design Automation, Computer Engineering, Computer Networks and Social Media Conference (SEEDA 2019), Piraeus, Greece, 20–22 September 2019.

Received: 4 February 2020; Accepted: 5 March 2020; Published: 8 March 2020

check for updates

Abstract: Technological evolution and in particular the development of the Internet of Things (IoT) has paved the way for material prosperity and a better standard of living. A critical factor in the effectiveness of emerging IoT applications, which heavily rely on sensor information flow, is the development of a functional and efficient Wireless Sensor Network. Additionally, the levels of automation are conducive to usability and time efficiency by reducing the need for human intervention, as well as increasing the rate at which experiments can be carried out. In current work, an already installed infrastructure on the Ionian University campus is considered and enhanced, with the goal of elevating accessibility and user-friendliness, by designing a web platform. The presented platform enables the remote development, execution and monitoring of simple but necessary network-based algorithms using a custom language, without requiring code to be uploaded to remote nodes. As a proof of concept, three information dissemination algorithms are implemented and provided as example templates for users, promoting simultaneously ease of use.

Keywords: Internet of Things; wireless sensor network; testbed; web platform; information dissemination; upload automation; programming language design; experimental environment

1. Introduction

Significant technological developments are being observed in the context of exponentially increasing human demands, one of them undoubtedly being the Internet of Things (IoT) [1]. An imperative need for proper operation is a well-organized and efficient Wireless Sensor Network (WSN), as a wide variety of sensors are embedded in interconnected IoT devices. By extension, the abstraction of the WSN's lower level details to provide high level functionality and improve usability is of major importance for the adoption of various IoT applications [2,3].

With regard to WSNs, low energy consumption, efficient communication between heterogeneous nodes, independent mobility and ultimately ease of use play a key role in their success. More specifically, their ease of use can be further enhanced by minimizing human intervention, while the suitability of the equipment consisting a WSN is also a crucial factor. The previously mentioned parameters have to be taken into account when implementing WSN-based IoT applications, and therefore there is a need for thorough experimentation regarding them.

A preparatory step for such experimentation is the design and installation of a testbed facilitating the operation of a WSN. A testbed reflects the architecture of a WSN and can serve as a stepping stone in the conduction of experiments for its evaluation. The design of a testbed intended to resemble the conditions of a WSN is predominantly oriented by the appliance of theoretical findings regarding this type of network, while its installation is determined by a sequence of decisions. However, the two aforementioned stages both underpin the proper operation, high efficiency, and long-term viability of the testbed.

During the testbed's installation process, a critical factor is the selection of a reasonable location, which can influence the quality of the network nodes' connection and thus the final topology. The physical security [4] of the location that will host the testbed has to be taken into account, so as to avoid malicious damage or stealing, while preventive measures also have to be taken to address these issues (e.g., by enclosing the devices in containers, by placing them far from plain sight, etc.), Moreover, the physical location is equally crucial to be studied from the perspective of its effects on the testbed's behavior, under the light of prevailing environmental conditions (e.g., humidity, temperature, dust, brightness, etc.) that are capable of affecting the accuracy and representativeness of the results.

Additionally, another principal factor is the specification of the hardware to be used has to take place, as it not only affects the hardware's efficiency (e.g., in terms of memory, computational capability, etc.), but the scope of the entire system too. Therefore, an optimal solution is one that meets the above requirements and is distinguished for its simplicity of use, suitability for a multitude of applications, and low cost.

As for the value and utility of a testbed, providing an appropriate infrastructure to support multiple experiments and users [4] would be considered an ideal, but hard to realize, feature, emphasizing its interdisciplinary dimension. Furthermore, such a system can be optimized by introducing automation techniques, such as remote device and process control, as well as task scheduling. This is often achieved by developing frameworks facilitating easier management, faster system control, as well as data visualization. In general, the automation of processes is a feature of utmost importance, as it facilitates both the user and the researcher, reducing the likelihood of mismanagement, while saving time.

One of the main parts of an experimental research lies in achieving the same (or almost the same) conditions among the multiple common experiments conducted. Achieving reproducibility is rather difficult due to physical causes governing the system's operation—thus hindering the recreation of fully identical conditions among multiple common experiments (e.g., connectivity between nodes may deteriorate due to different weather conditions). Furthermore, it is necessary for the testbed to be able to meander in the characteristics it can reproduce, so as to support the evaluation of a wider plethora of applications.

Given the ever-present need for applications to be developed and evaluated in an academic community within a timely manner, a testbed infrastructure serves as the means of filling the gap between simulation environments and real-world conditions. Supporting experimentation and evaluation through user-friendly and accessible means helps promote the educational aspects of the academic community. In this paper, the existing infrastructure [5] of the CAmpus TestBed of the Ionian UniverSity (CaBIUs) is extended. While the developed testbed provides a certain degree of security, given that the nodes are installed indoors within protective enclosures, the necessity for automation of procedures arises.

More specifically, the network setup and topologies that emerge from the composed system are further analyzed in order to better inform and facilitate future studies. Moreover, the main contribution of this work is the design and implementation of a web platform to enable remote development and control of the deployed system and enhance its usability and accessibility, thus expediting administrative actions. This web platform's architecture comprises lightweight low-cost components that seamlessly fit into most emerging IoT applications, utilizing Arduino prototyping boards equipped with XBee transceivers as WSN nodes and a Raspberry Pi as the web server host. The developed platform aims at the remote execution of algorithms coded in a custom programming language, without requiring an upload process, which would otherwise require additional devices—and a fairly lengthy process, given the nodes' enclosure. The scope and breadth of the designed framework and language are investigated, providing examples of potential algorithms that can be implemented, while some common examples are compiled and provided as templates to be used within the platform.

The rest of this paper is organized as follows. An overview of related literature is presented in Section 2, while the testbed under consideration is described in Section 3, including the utilized devices and resulting network configuration. Section 4 specifies the developed web platform and implementation. Discussion and future work is laid out in Section 5. Finally, conclusions are drawn in Section 6.

2. Past Related Work

The appropriateness, contribution, and utility of WSNs to a large number and variety of industries has attracted the interest of several members in the scientific community [6–8]. In particular, over the past few years, WSN performance has been examined and evaluated on several applications, such as military and agricultural applications [9,10], smart environments [11,12], emergency situations [13,14], environmental monitoring [15,16], and health applications [17,18].

Imran et al. [19] focus on evaluating WSN protocols and algorithms, through emulators, simulators, and testbeds, providing vertical and horizontal analysis, with the ultimate goal of helping researchers choose the most appropriate tool for their needs. On the other hand, Steyn and Hancke [20] concentrate on both laboratory and real-world testbeds, providing information on their architecture and function characteristics, and suggesting possible combinations of them for controlling interference sources while experimenting with a variety of wireless channel conditions.

In a parallel manner, Horneber and Hergenröder [21] first present various architectural testbeds for different types of requirements, and eventually raise general issues of development of such systems. In addition, Kim et al. [22] collect once more the essential requirements associated with such systems, categorize the testbeds by architecture, and ultimately evaluate them. Important information is also summarized in [23], bringing out the requirements, challenges, and design aspects of enhanced testbeds, which have a high degree of control over reproducability and thus accountability, as well as federation with other testbeds.

One of the fundamental studies on testbed implementation is MoteLab [24], which is intended for WSNs, offering permanently installed Ethernet-connected nodes on Harvard University's campus, enabling a web-interface to organize and coordinate experiments, retrieve data, and interact with specific nodes.

Another representative testbed example is WISEBED [25], which can support heterogeneous nodes by providing researchers with the opportunity to conduct remote experiments via the Internet. At the same time, it supports a repository of protocols, algorithms, and mechanisms to facilitate the process of extracting measurements from experiments.

Furthermore, a different approach to the issue is made with FlockLab [26], in which sensor nodes are paired with observer nodes. Observer nodes can manipulate the sensor nodes extensively, while maintaining a LAN in which the experimental data are securely available. Additionally, they accurately adjust and monitor pins and power metrics.

The w-iLab.t testbed [27] exhibits multi-dimensional behavior as it is suitable for large-scale wireless sensor and actuator networks and WiFi mesh and ad hoc experiments. The system is accessible to authorized users through a web-based interface where users can configure the experiments, completely customize the experimentation process, and collect the results that can be visualized on graphs or maps in real time or asynchronously with the experiments.

On the other hand, the PotatoNet testbed [28] has been implemented in an outdoor environment—where conditions continuously shift—and aims to provide a durable and at the same time flexible system. Moreover, PotatoMesh [29] extends PotatoNet to better manage energy reserves, even in cases of large volumes of data generation.

In relation to more recent studies, MobiLab [30] focuses on the phenomenon of mobility in a WSN. In particular, this WSN comprises three types of nodes: static relay nodes, mobile nodes, and sniffer nodes, which are not involved in experiments but are simply providing additional information by monitoring the wireless communication channel. The overall architecture of the system, except for the WSN, includes a control station, which functions as a system–user interface (both command-line and web-based interface), a node manager that connects each node of the network to the control station and a WiFi backbone network, made up of node managers and the control station, designed to exchange information during experiments. It is noteworthy that this testbed does not require a specific network size or topology to perform experiments.

Likewise, Indriya2 [31] extends the original version of "Indriya" by enriching the capability of supporting heterogeneous, higher data-rate, multi-experiments of multi-users and providing architecture for accessing the outputs in real time through an MQTT server. It is worth noting that it offers REST APIs for connecting to other testbeds, ensuring the required levels of security.

A different approach to the issue is presented in SDNWisebed [32], where WSN management is simplified and implemented with Software-Defined Networking Solutions. SDNWisebed includes a controller that manages network operation and sensor nodes, which aim to transfer packets to the WSN and interact with the environment, sensing environmental indicators or performing some action. They essentially define the data plane. Information exchange between controller and sensor nodes is achieved via the border router. In addition, it offers dynamic information, such as traffic statistics, that enables better packet routing.

Additionally, HATBED [33] is made up of targets, observers, and a controller that is essentially a computer server that controls the entire testbed automatically. Targets are the sensor nodes, which are monitored by observers—as their name implies—that are able to interact with each other. Its main purpose is to aid in profiling, testing, and debugging code without impacting the nodes' performance, utilizing hardware assisted tracing.

Gu et al. [34] concentrate on optimizing an existing testbed, through virtualization technology to better utilize resources and enhance scalability and flexibility, to meet the growing demands and needs of large-scale IoT experiments. At the same wavelength and better IoT needs, NEWSBED is met [35], a proposed platform designed to facilitate the testing and implementation of ideas, despite the difficulties that may arise in such an environment.

Huang and Yu [36] are studying the issue of node positioning, providing a demonstration of an innovative range-based positioning testbed that can meet the high demands of node heterogeneity, while Lattanzi et al. [37] focus on monitoring indoor human comfort in a university campus in Italy, designing a testbed the sensors of which are multitasking, performing independent actions.

A recent study presented in [5] thoroughly describes the installation and implementation of a prototyping testbed, while taking into consideration environmental parameters that may affect the connectivity of the resulting network topology. The current work further extends the implementation and capabilities of this particular testbed, the soundness of which has been proven in earlier studies [38–40].

3. Testbed Description

In this section, a comprehensive listing of the equipment utilized for the presented testbed is provided, along with some of the resulting topological information that may be of interest to future applications.

3.1. Equipment

Choosing the equipment needed to build a WSN, and by extension a testbed, is a crucial process in terms of performance. The ease of use and configuration of the devices, along with their low cost, reflect the major requirements for implementing a testbed for IoT application experiments. In order to satisfy these requirements, the following devices are selected: (i) Arduino Mega 2560 Rev3; (ii) Arduino Wireless SD Shield; (iii) XBee S2C Zigbee module with wire antenna; (iv) Raspberry Pi 3 Model B; and (v) Sensory Devices.

Each of these devices offers a decisive contribution to the system by virtue of their capabilities that will be showcased next. In more detail, the Arduino Mega 2560 is a microcontroller board based on the Atmega2560, which can be programmed to bring to fruition a plethora of projects. Its technical specifications are presented in Table 1 [41].

Table 1. Arduino Mega Specifications.

Component	Value
Microcontroller	ATmega2560
Operating Voltage	5 V
Input Voltage (recommended)	7–12 V
Input Voltage (limit)	6–20 V
Digital I/O Pins	54 (of which 15 provide PWM output)
Analog Input Pins	16
DC Current per I/O Pin	20 mA
DC Current for 3.3 V Pin	50 mA
Flash Memory	256 KB of which 8 KB used by bootloader
SRAM	8 KB
EEPROM	4 KB
Clock Speed	16 MHz
LED_BUILTIN	13
Length	101.52 mm
Width	53.3 mm
Weight	37 g

The interface between the Arduino Mega and the antenna is achieved through a mediator, whose role is played by the Arduino Wireless SD Shield. This Shield contributes to wireless communication and allows the data to be stored on a micro-SD card [42].

A vital component for the aforementioned network is the transceiver, with the help of which wireless communication of the nodes is established. Hence, the XBee S2C Zigbee transceiver has been chosen because of its high specifications and universality in a variety of applications. More extensively, Zigbee, the protocol on which the antenna's functionality is based, is intended for personal wireless networks and provides a maximum transmission rate of 250 kbps [43], extending the 802.15.4 protocol for mesh topologies.

When the network's operation is based on the Zigbee protocol, there are three types of nodes that constitute it [44]. Each type plays a unique role, the details of which are presented below. The first type is the "Coordinator", which is vital to the existence and configuration of the network. However, there is a strict limitation on the existence of a single Coordinator in the whole network. Upon the fulfillment of its role/task in setting up the network, the Coordinator's functionality is equivalent to that of a "Router's", which is the second type of node.

Apart from the Coordinator, the network can consist of a large number of Routers whose main concern is the control and execution of the packet routing process, including all the actions that are necessary to accomplish this. A common feature among these two types mentioned above is that they can function both as sources and destinations in the data transmission process.

The third type is the "End device", which sleeps periodically and can only communicate with the node that has the property of being its parent (i.e., a Router or a Coordinator), in order to transmit and receive data. Consequently, it requires lower energy, compared to the rest of the types.

In terms of XBee module modes of operation [44], "Transparent" mode is first encountered. Transparent mode operates, using a set of user-defined settings and parameters, preventing their modification during the execution of the program. Such parameters include options, e.g., the destination address that are crucial for more complex applications. Furthermore, when data are sent to the module, they are transmitted and the receiving nodes do not acquire any additional information, such as the source address.

Another mode of operation is the "Application Programming Interface (API)", where communication between nodes relies on the exchange of data packets, which represent functions or events for the device, making it possible to modify various parameters, even during its operation. It also allows identifying the source address of a packet, transmitting data to different addresses, and receiving information about the success or failure of a packet transmission. Thanks to these abilities, it has been selected for the purposes of this study.

Continuing with the next component of the equipment used, Raspberry Pi 3, despite its low cost and small size (almost the same with the size of a credit card), acts as a computer in terms of enhancing conventional computers [45]. Its technical characteristics are presented in detail in Table 2 below [46].

Table 2. Raspberry Pi 3 specifications.

Component	Value
SoC	Broadcom BCM2837 64 bit
CPU	4 × ARM Cortex-A53, 1.2 GHz
GPU	Broadcom VideoCore IV
RAM	1 GB LPDDR2 (900 MHz)
Networking	10/100 Ethernet, 2.4 GHz 802.11n wireless
Bluetooth	Bluetooth 4.1 Classic, Bluetooth Low Energy
Storage	microSD
GPIO	40-pin header, populated

As is evident, a WSN does not exist without the sensors; thus, in the implemented testbed, a wide range of sensors has been used, corresponding to the most common requirements and serving the most common purposes. More specifically, accelerometers, humidity, temperature, and ultraviolet radiation sensors have been installed [47–49]. Furthermore, GPS modules and Lithium Polymer batteries are provided, in order to meet higher standards. Note that the sensor nodes are powered by power banks [50], which are available for all kinds of experiments in the described system. The final form of the system's node, integrating all aforementioned components and stored in a case to ensure its physical security, is depicted in Figure 1.

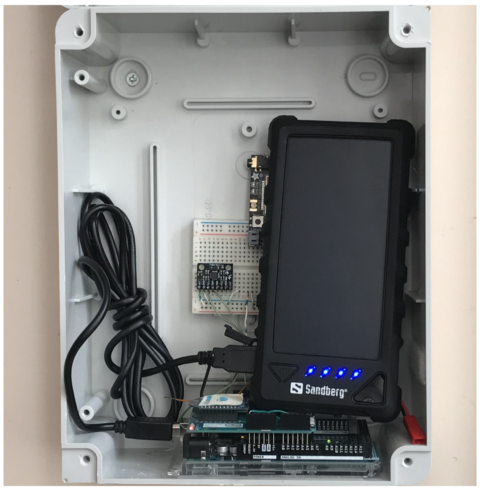


Figure 1. A node of the developed system enclosed in its protective case, along with its battery.

Moreover, with respect to a node's cost, the price of each of the system's individual components is listed in Table 3. A single node, without any sensors or power sources, comprised of just an Arduino Mega 2560, a Wireless SD Shield, and an XBee S2C module costs close to $100. Depending on the number of sensors used, the total cost of each node can go up to $160. When it comes to power sources, a durable power bank with solar recharging costs more than $80, whereas more lightweight lithium polymer batteries can cost less than $20. Raspberry Pi 3 boards cost close to $50, although these are not required for each node and only a single one is used in this study. These prices represent the current state of the market for the particular models referenced from relatively reliable sources.

Table 3. Cost of each of the utilized components.

Component	Price (USD)
Arduino Mega 2560 Rev3	38
Arduino Wireless SD Shield	20
XBee S2C Zigbee Module with Wire Antenna	34
Raspberry Pi 3 Model B	47
Accelerometer Sensor	5
Humidity and Temperature Sensor	10
Ultraviolet Radiation Sensor	3
GPS Module	40
Lithium Polymer Battery	16
Power Bank	87

3.2. *Topology Configuration*

The testbed described was established on the Ionian University campus. It consists of a set of up to 30 sensor nodes, which are located in scattered areas of the University premises, as depicted in Figure 2. The installation points have been selected, based on parameters that may affect the performance of the system, taking into account the connectivity of the nodes. More specifically, the installation was carried out on walls, where the predominant material is stone, which can turn message transmission into a barrier process.

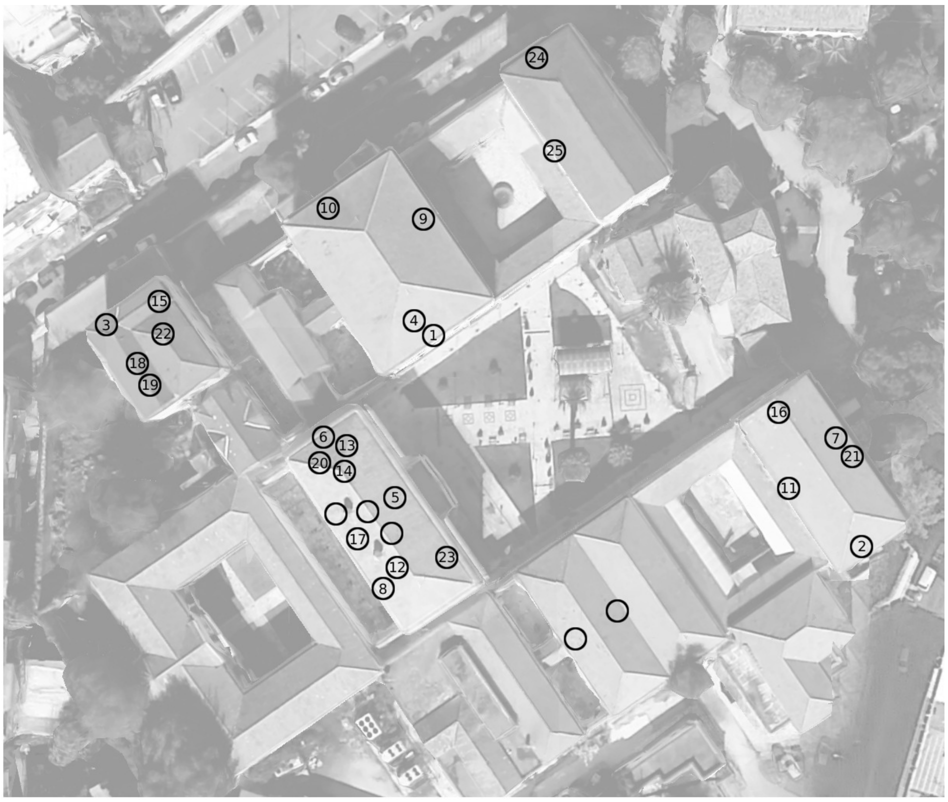


Figure 2. The positions of CaBIUs' nodes from a satellite view, indicated by circles. The numbers are the corresponding node's ID. In the topology instance under consideration, the IDs of the nodes, which were not installed, have been omitted.

Therefore, the control of fruitful communication was tested during installation, with the help of XCTU Software, with the ultimate goal of avoiding one or more fully-isolated nodes. This was accomplished with the transceivers operating utilizing their maximum transmission power capabilities. While considering safety along with security of the equipment, nodes are placed in locations that are easily accessible, but not in common view. The devices are housed in protective enclosures, especially for the duration of the experiment's conduction, where security is guaranteed by the researchers' presence. The rest of the time, the devices are removed from the containers and stored in a safe location, which is a temporary solution until CaBIUs is fully established. In the sequel, additional details regarding the network's topology are given.

The network's topology is likely to incur changes as link quality can fluctuate based on environmental conditions, which was studied in previous work [5]. Due to this fact, only one particular topology instance is examined and presented here, but do note that most of the topology alterations, which have been observed in previous experiments [38–40], occur in the links between nodes on the network's outskirts. Additionally, as far as applications are considered, neighbor discovery processes should be employed, since the XBee module's Network Discovery (ND) process generally provides asymmetric links, and, from the authors' view, is unreliable in certain cases.

An undirected graph $G(V, E)$, where V denotes the set of nodes and E the links among them, may be formed from the particular topology instance so as to formulate graph theoretic concepts, which might be of use when designing and analysing applications. Thus, the network comprises $|V| = 25$ nodes and $|E| = 72$ links among them, derived from the neighbor discovery presented in [38–40], so as to provide reliable symmetric links. The adjacency matrix of the resulting topology is depicted in Table 4. The average degree of the network's nodes is $\delta = 5.76$, while the maximum degree is $\Delta = 12$. The network's diameter $d = 5$ and radius $r = 3$ are defined as the maximum and minimum of the nodes' eccentricity, respectively. The network's density is

$$D = \frac{2|E|}{|V|(|V|-1)} = 0.24.$$

Table 4. Adjacency matrix of the selected topology instance.

ID	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
1	0	0	0	1	1	0	0	0	1	0	0	0	1	0	0	1	0	0	1	0	0	0	1	1	1
2	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	1	0	0	1	0	1	0	1	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1	0	0	1	0	0	0
4	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1
5	1	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	1	0	0	1	0	0	1	0	0
6	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	1	0	1	1	0	0	1	0	0
7	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0
8	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	1	0	0	1	0	0	1	0	1
9	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	1	0	0
12	0	0	0	0	1	1	0	1	0	0	0	0	1	1	0	1	1	0	0	0	0	0	1	0	0
13	1	0	0	0	0	1	0	0	0	0	0	1	0	1	1	1	0	0	1	1	0	0	0	0	0
14	0	0	0	0	0	1	0	0	0	0	0	1	1	0	0	0	1	0	0	0	0	0	1	0	0
15	0	0	1	0	0	1	0	0	0	0	0	0	1	0	0	0	0	1	1	0	0	1	0	1	0
16	1	1	0	0	1	0	1	0	0	0	1	1	1	0	0	0	0	0	1	0	0	0	1	0	0
17	0	0	0	0	1	1	0	1	0	0	0	1	0	1	0	0	0	0	0	0	0	0	1	0	0
18	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	1	0	0	0
19	1	1	1	0	0	1	0	0	0	0	0	0	1	0	1	1	0	1	0	0	0	1	0	0	0
20	0	0	0	0	1	1	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
21	0	1	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
22	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1	0	0	0	0	0	0
23	1	1	0	1	1	1	0	1	0	0	1	1	0	1	0	1	1	0	0	0	0	0	0	1	0
24	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	1
25	1	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

4. An Enhanced Remote Development Platform

Literature indicates a clear trend in the direction of minimizing the need for human intervention when developing a testbed infrastructure. This may be accomplished under various utilities, with web-based frameworks and architectures being evidently prominent in this regard. These web applications aid in the remote and automated uptime status monitoring, code upload, execution and visualization, as well as multiple other facets of the system's overall operation. As such, a web-based platform is developed for the underlying WSN in order to assist with the remote execution and monitoring of algorithms.

4.1. Architecture of the Proposed Platform

The designed architecture fully utilizes the components presented in Section 3. Arduino boards along with XBee modules and accompanying sensors play the role of the WSN's nodes. The Zigbee network's Coordinator acts as the sink node, while also maintaining a serial connection with a Raspberry Pi. The latter also hosts the web server, which, by communicating with the WSN's sink (i.e., the Coordinator), exchanges data and commands. The architecture is depicted in Figure 3.

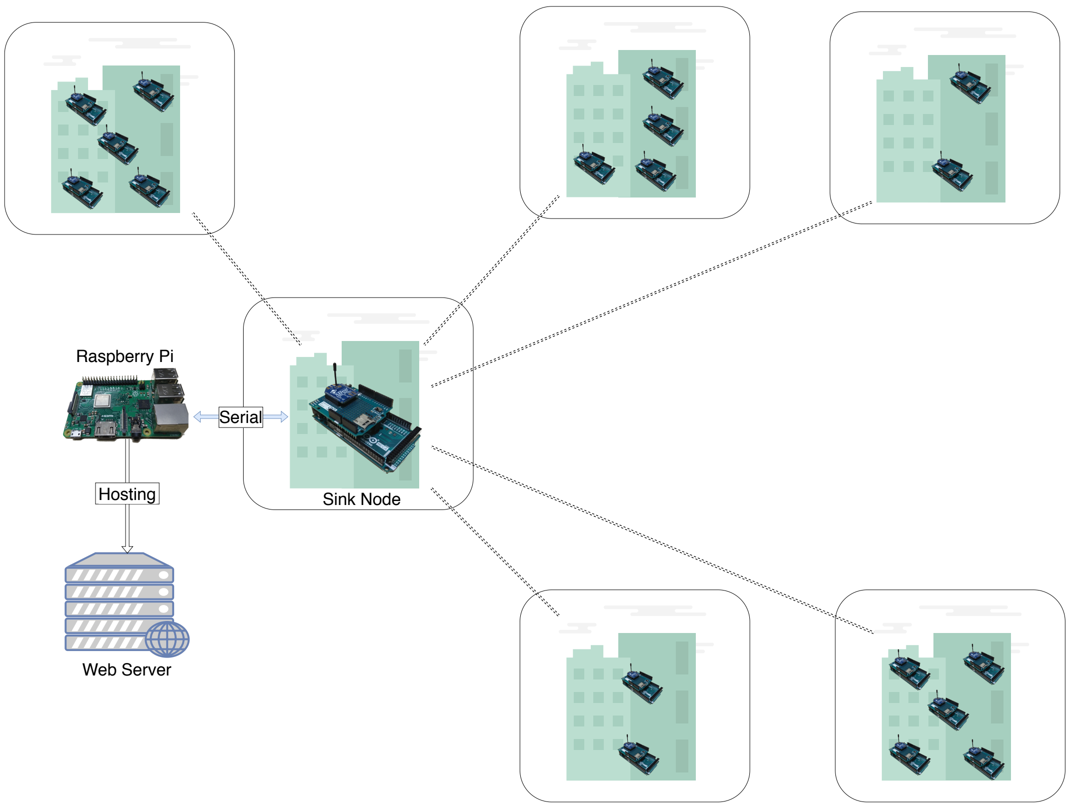


Figure 3. Depiction of the system's architecture on the Ionian University campus.

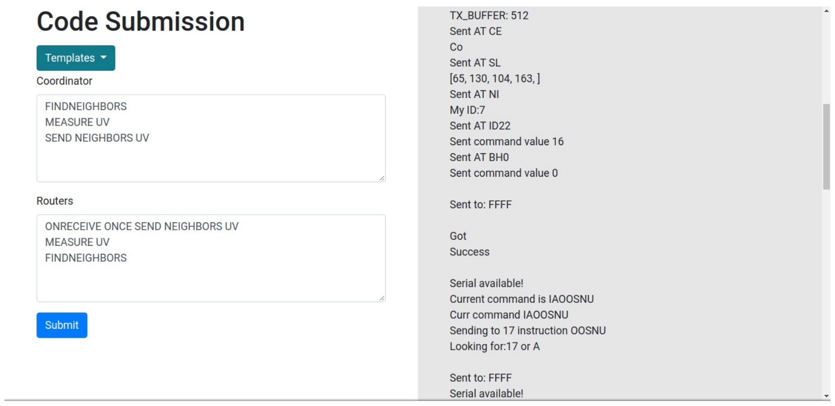
According to this particular architecture, the web server handles incoming connections, which are able to send instructions to the WSN's sink. These custom instructions represent common WSN-related instructions, such as transmitting, receiving, or measuring. The input instructions are interpreted and validated by the server. If they are valid, they are encoded in the form of a string and sent to the Arduino, which decodes the instructions and executes the appropriate actions. In order to program remote nodes, i.e., the network's Routers, the Coordinator transmits the received encoded instructions, which are then handled by the addressed node. The Coordinator's serial output is captured by the web server, so that clients may monitor the algorithm's execution.

It is worth pointing out that the Arduino Wireless SD Shield typically occupies the Arduino's main serial port, which complicates writing and reading through the connected computer's serial connection. Normally, the XBee modules utilize the Arduino's serial buffers in order to receive and transmit messages. If the same serial port is also utilized by the connected device, transmitted data are certain to get muddled. In order to address this, the shield's RX and TX pins are slightly bent, so that they are no longer inserted into the Arduino's corresponding pins. The bent pins are hence routed to one of the Arduino Mega's other serial pins, so that the XBee module may once again be fully operational. Since two serial connections are required, memory costs slightly increase as well to accommodate the RX and TX buffers, although the rest of the framework has trivial memory requirements, so an Arduino Mega should suffice for reasonably large networks.

Regarding the efficiency in various aspects of the system, the proposed architecture can certainly be improved. For instance, the Raspberry Pi can make for a poor web server, due to its limited computational capabilities and I/O transfer rates. The Ionian University's servers could be utilized instead, which also share Gigabit Ethernet connection with the campus's premises. However, the particular Raspberry Pi model shares the same controller for both USB and Ethernet ports, which could prove to bottleneck the system's performance, in which case it would have to be replaced by a more powerful unit.

4.2. Web Framework of Proposed Platform

The main purpose of the developed web framework is to allow users to create and test simple algorithms intended for WSN applications. Thus, it primarily consists of two distinct interface components with the goal of permitting the user to input a specific algorithm, in the form of a sequence of custom-designed commands, as well as a method for monitoring the provided algorithm's execution. The website's interface is showcased in Figure 4.



```
FINDNEIGHBORS
MEASURE UV
SEND NEIGHBORS UV
```

```
ONRECEIVE ONCE SEND NEIGHBORS UV
MEASURE UV
FINDNEIGHBORS
```

```
TX_BUFFER: 512
Sent AT CE
Co
Sent AT SL
[65, 130, 104, 163, ]
Sent AT NI
My ID:7
Sent AT ID22
Sent command value 16
Sent AT BH0
Sent command value 0

Sent to: FFFF

Got
Success

Serial available!
Current command is IAOOSNU
Curr command IAOOSNU
Sending to 17 instruction OOSNU
Looking for:17 or A

Sent to: FFFF
Serial available!
```

Figure 4. The implemented web framework's main interface. The submitted code is displayed on the left panel, whereas the Arduino's output is on the right.

The user's algorithm is forwarded to the server through the use of two separate forms, one for the Coordinator's and one for the Routers' actions to execute. Once the forms have been submitted, the contents of both are passed to the parser through the web server and validated. The user is redirected back to the same page and an appropriate error message is displayed, if something went wrong. Alternatively, if the program's code is valid, the instructions are encoded and sent via the serial interface to the connected Arduino board, while the Arduino's output is captured, displayed, and updated live in the web page.

The encoding process is fairly straightforward, aiming to minimize the amount of data transmitted over serial and Zigbee. In particular, only the first letter of each token is retained. As of yet, no ambiguity occurs among commands' first letter, although this might have to change later on, when the selection of commands is further extended. Once the encoded forms' content has been forwarded to the Arduino, they are decoded and any Router operations are transmitted to the specified remote Router node. The decoding process on the receiving nodes is made significantly easier, with a linear computational complexity relative to the encoded command's length, thanks to the validity of the program's structure being guaranteed by the web framework.

In terms of communication overhead, the instructions of a program can be aggregated into a single message, provided they fit within the 255 bytes that Zigbee packets are restricted to. If a program exceeds the limit of 255 bytes, which is rare using the current grammar, it can be fragmented into multiple messages. It should be noted that Zigbee only allows an average of one broadcast per second, so larger programs would take a while to propagate.

Traditional WSN related algorithms have been implemented and made available for using as templates. These templates are stored in the website's database, which can be selected and loaded through an additional field. The selection of available algorithms currently includes flooding, flooding with feedback (Echo), as well as single and multiple random walkers.

The selected algorithms differ widely with respect to their characteristics, providing trade-offs between the required number of transmissions and completion time. More specifically, flooding requires a large number of transmissions to cover the entire network, but is guaranteed to terminate within a small time period [51]. Similarly, the Echo algorithm involves flooding; however, feedback is also provided by its termination, making it the default option when gathering data. Finally, random walkers are capable of covering a network by sending a comparatively smaller amount of messages, but the termination time is difficult to predict [52]. The varying traits of these algorithms showcase the platform's capability in granting a versatile environment.

Considering that the currently installed system only consists of a single Zigbee network, there is a restriction of only a single user being able to run a program on the platform, although no restrictions are imposed on the number of users accessing the Coordinator's serial output. As such, several mechanisms have to be implemented in future work that will lock write permissions to the serial connection, so as to not interrupt a program's execution. This is not currently an issue, as the website is not yet publicly available.

4.3. Custom Designed Language of the Proposed Platform

The custom language designed for use in the implemented platform can be utilized to implement a fairly extensive set of basic programs. The defined instructions include essential actions that are commonly required in distributed networking systems. The full set of available instructions is listed in Table 5.

Table 5. The LL(1) grammar of the language designed for the developed architecture. Terminal symbols are capitalized, while *ID* refers to an unsigned 1-byte integer indicating a node's assigned ID.

Instruction		**Parameters**
Stmt_list	$\rightarrow$	Stmt Stmt_list | ε
Stmt	$\rightarrow$	Remote_Instruction | Local_Instruction
Remote_Instruction	$\rightarrow$	INSTRUCT Who Local_Instruction
Local_Instruction	$\rightarrow$	Receive_response | Node_operation
Receive_response	$\rightarrow$	ONRECEIVE Frequency Node_operation
Node_operation	$\rightarrow$	FINDNEIGHBORS | MEASURE Key | SEND Who Node_data
Node_data	$\rightarrow$	Key | Metadata
Key	$\rightarrow$	TEMP | HUMIDITY | UV | ACCEL
Metadata	$\rightarrow$	*ID* | NEIGHBORS | CLOCK
Who	$\rightarrow$	*ID* | NEIGHBORS | ALL | RANDOM
Frequency	$\rightarrow$	ONCE | ALWAYS

Although the available commands are fairly self-explanatory, a more detailed description is provided:

1. FINDNEIGHBORS: scan for neighbors using Zigbee's ND process.
2. MEASURE: get the measurement value returned by one of the connected sensors determined by token Key, including temperature (TEMP), humidity (HUMIDITY), UV radiation (UV) or accelerometer (ACCEL).
3. SEND: send data acquired by a node. This data includes metadata, i.e., a node's ID, its neighbors' IDs or its clock in milliseconds, or a measurement determined by Key. The message is sent to the node(s) specified by Who. Available options include: (i) a node with a specific ID; (ii) a node's neighbors; (iii) all of the network's nodes (utilizing Zigbee's broadcast transmissions); and (iv) a random neighbor.
4. ONRECEIVE: indicates a Node_operation to be executed as a response upon receiving a message. The response command is stored and may be reused or consumed after the first message, as indicated by FREQUENCY.
5. INSTRUCT: send a Local_instruction (any instruction other than INSTRUCT) to node WHO.

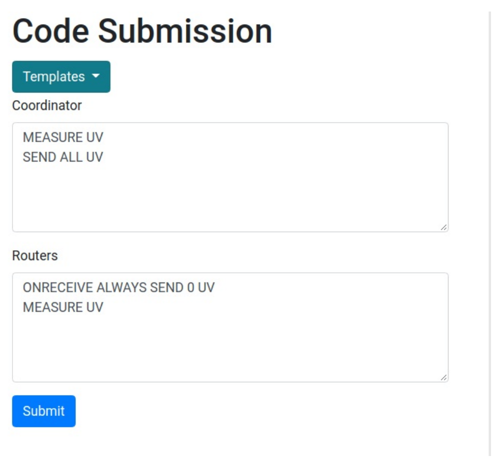
All Local_instruction statements are self-contained commands executed by each node. The INSTRUCT command is separated, so as to restrict its use solely to the Coordinator and avoid potentially endless rebroadcasts. In particular, this command is only executed in order to transmit to remote Routers the input algorithm's instructions; end users' direct use of this command is not permitted. When a Router receives the specified INSTRUCTION in the form of an encoded string, it is either immediately executed or stored in a variable to be executed when a message is received.

Even though this is a very limited selection of commands, they can be combined to facilitate some fundamental network functionality. For instance, two algorithms, namely Echo and the deployment of a random walker, are implemented by utilizing these commands and are displayed in Figure 5. These algorithms, along with a flooding implementation, are available as templates for the website's users.

The main limiting factor for an endeavor, such as developing a custom language designed for Arduinos and transmitted over Zigbee, is the control of remote nodes' variables. Variable names cannot be explicitly defined, partly due to Arduinos using a compiled language—thus, variables cannot be defined during runtime—and partly because of limitations imposed by Zigbee on the length of a transmitted message. A way to address this issue could be predefining dictionary-like structures on the Arduinos and transmitting the variables' names and values.

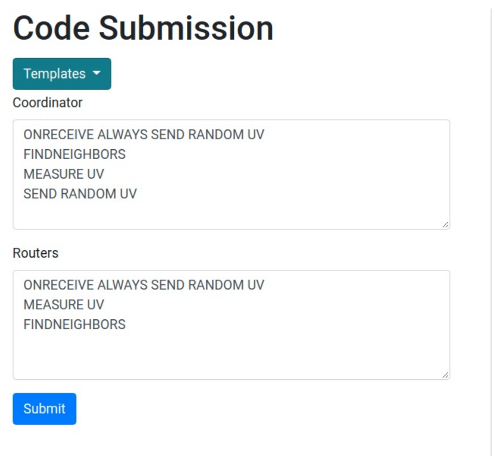
In more detail, a naive implementation of this concept would involve the definition of two parallel arrays for each data type, with the first holding variable names as strings and the second maintaining the corresponding variables' value. This is not very practical, as the memory requirement for the first array would be equal to the length of the variable's name multiplied by the number of "defined" variables. Do note that each variable access would also require lookup in the first array, which is not too computationally efficient. Furthermore, since these variables would have to be transmitted through Zigbee, they would take up much of the available bandwidth, not to mention a large proportion of each message's maximum length of 255 bytes. If compound types are to be considered, such as arrays, matters are further complicated and real-world implementation becomes increasingly unrealistic.

Other commonly used approaches that would improve performance could utilize hash tables and similar dictionary-like structures. However, these structures can take up a significant amount of memory, which is already a scarce resource. Therefore, this may be an option for future work, but issues are still certain to arise with actual deployment seeming unrealistic. Thus, this custom language will likely remain a fairly basic implementation and tool.


Code Submission

Coordinator
```
MEASURE UV
SEND ALL UV
```
Routers
```
ONRECEIVE ALWAYS SEND 0 UV
MEASURE UV
```

(**a**)


Code Submission

Coordinator
```
ONRECEIVE ALWAYS SEND RANDOM UV
FINDNEIGHBORS
MEASURE UV
SEND RANDOM UV
```
Routers
```
ONRECEIVE ALWAYS SEND RANDOM UV
MEASURE UV
FINDNEIGHBORS
```

(**b**)

Figure 5. Execution of two programs developed in the custom designed language: (**a**) the Echo algorithm; (**b**) deployment of a single random walker.

5. Discussion and Future Work

During the design of CaBIUs' refinement process through the implemented remote development platform, plenty of food for thought was brought about. More specifically, shortcomings observed in the present system defined the final scope and shape of the implemented architecture. Such shortcomings revolve around the inability of remotely uploading code—partly due to a lack of required devices and partly because of the nodes' impermanent installation in their enclosures—and the lengthy process of having to collect the nodes, in order to arm them with code. Thus, these costly, in terms of time, requirements are addressed through the creation of a framework to remotely transmit commands through the only available wireless communication means, Zigbee.

Though the design of a language transmitted through Zigbee and executed over Arduino imposes certain restrictions, the provided syntax needs to be user-friendly and adequately comprehensible to contribute to the system's overall accessibility. Additionally, there is always room for improvement and expansion of the available system, with respect to both software and hardware that the language needs to be able to adapt to and support. For instance, since a program in the developed language needs to be disseminated through broadcasts across the entire network, which is inefficient in large-scale networks, more efficient techniques could be considered. Of particular importance in this domain is clustering, allowing for more efficient information flow and error-prone architectures [53,54].

Aside from the aforementioned considerations that drove to the design and completion of the web framework, some issues that are yet unaddressed came to light. To start with, the impermanent positioning of nodes leads to limitations of interoperability among CaBIUs' constituents. By extension, resolving this issue would permit the development of complementary tools (e.g., remote upload, visualization) or even the network's partitioning to support multiple concurrent users.

It is noteworthy that the testbed's enrichment with other kinds of devices and sensors, conducing to a heterogeneous networking environment and providing a common subset of devices, hence enabling comparison between testbeds' effectiveness and performance. The development of web APIs would be another step in the direction of federation with other testbeds, while also being complementary to collaboration among different universities.

Providing the developed platform alongside a functional testbed is largely conducive to the promotion of educational matters in a university—for instance, the platform's already fairly simplified programming language can be augmented through the use of block-structured programming interfaces. This further abstracts implementation details, making IoT application development accessible to less experienced undergraduate students, through a more interactive and hands-on approach than more advanced, systems-oriented programming courses.

Furthermore, the framework and testbed can be propelled in multiple directions by the students themselves, including the addition of more advanced networking functionality, user-friendly interface design, development of backend APIs, and integration of concurrency mechanisms to support multiple users. These activities could be integrated into courses spanning a multitude of subject areas, requiring varying levels of knowledge and technical skills. All in all, this environment also helps stimulate up-and-coming researchers, expediting the cultivation of critical thinking, problem solving and decision-making in real-world applications.

6. Conclusions

Providing a testbed helps to conduct experiments and tests to verify scientific theories and to evaluate the performance of multiple tools and emerging technologies. Consequently, such a system prevails in the evolution and advancement of IoT technologies and has drawn heavy interest from the scientific community.

By extension, each preparatory step and decision during the implementation as well as installation of a testbed is extremely critical, from the equipment selected to the site in which it will be planted. Among them, the high efficiency of the system is determined by both low cost and low energy consumption, as well as ease of use and accessibility.

The need for quick and easy experimentation by members of an academic community, such as university students, has fueled the creation of a testbed for testing techniques and finding effective solutions in a WSN. The testbed described here was installed on the Ionian University campus and meets all of the above requirements.

In this context, the existing infrastructure is further analyzed to provide graph theoretic metrics of resulting topologies. Additionally, a web-based platform is established to provide elevated levels of automation and remote control of the installed system. The web architecture is described in detail and aims at facilitating the platform's users by granting a set of tools that enable remote monitoring and execution of WSN-related algorithms.

Author Contributions: Conceptualization, V.K., G.K. and K.O.; Data curation, A.Z. and V.K.; Formal analysis, A.Z.; Funding acquisition, K.O.; Investigation, A.P.; Methodology, A.P., K.B., A.T. and G.T.; Project administration, K.B.; Resources, S.V.; Software, S.V. and G.K.; Validation, A.S.; Visualization, A.G.A. and A.S.; Writing—original draft, A.G.A.; Writing—review & editing, A.T., G.T. and K.O. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Acknowledgments: This work was supported in part by the "A Pilot Wireless Sensor Networks System for Synchronized Monitoring of Climate and Soil Parameters in Olive Groves" (MIS 5007309) project, which is partially funded by European and National Greek Funds (ESPA) under the Regional Operational Program "Ionian Islands 2014–2020."

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

Abbreviations

The following abbreviations are used in this manuscript:

CaBIUs	CAmpus TesBed of the Ionian UniverSity
IoT	Internet of Things
WSN	Wireless Sensor Network
API	Application Programming Interface

References

1. Xia, F.; Yang, L.T.; Wang, L.; Vinel, A. Internet of things. *Int. J. Commun. Syst.* **2012**, *25*, 1101. [CrossRef]
2. Mainetti, L.; Patrono, L.; Vilei, A. Evolution of wireless sensor networks towards the Internet of Things: A survey. In Proceedings of the SoftCOM 2011, 19th International Conference on Software, Telecommunications and Computer Networks, Split, Croatia, 15–17 September 2011; pp. 1–6.
3. Gubbi, J.; Buyya, R.; Marusic, S.; Palaniswami, M. Internet of Things (IoT): A vision, architectural elements, and future directions. *Future Gener. Comput. Syst.* **2013**, *29*, 1645–1660. doi:10.1016/j.future.2013.01.010. [CrossRef]
4. Gluhak, A.; Krco, S.; Nati, M.; Pfisterer, D.; Mitton, N.; Razafindralambo, T. A survey on facilities for experimental internet of things research. *IEEE Commun. Mag.* **2011**, *49*, 58–67. doi:10.1109/MCOM.2011.6069710. [CrossRef]
5. Papamichail, A.; Alvanou, A.G.; Zervopoulos, A.; Bezas, K.; Vergis, S.; Koufoudakis, G.; Oikonomou, K.; Tsoumanis, G. Description of the Ionian University's Campus Wireless Network Testbed Infrastructure. In Proceedings of the 2019 South Eastern European Design Automation, Computer Engineering, Computer Networks and Social Media Conference (SEEDA-CECNSM), Piraeus, Greece, 20–22 September 2019.
6. Akyildiz, I.; Su, W.; Sankarasubramaniam, Y.; Cayirci, E. Wireless sensor networks: A survey. *Comput. Netw.* **2002**, *38*, 393–422. doi:10.1016/S1389-1286(01)00302-4. [CrossRef]
7. Ali, A.; Ming, Y.; Chakraborty, S.; Iram, S. A Comprehensive Survey on Real-Time Applications of WSN. *Future Internet* **2017**, *9*, 77. doi:10.3390/fi9040077. [CrossRef]
8. Yick, J.; Mukherjee, B.; Ghosal, D. Wireless sensor network survey. *Comput. Netw.* **2008**, *52*, 2292–2330. doi:10.1016/j.comnet.2008.04.002. [CrossRef]
9. Diamond, S.M.; Ceruti, M.G. Application of Wireless Sensor Network to Military Information Integration. In Proceedings of the 2007 5th IEEE International Conference on Industrial Informatics, Vienna, Austria, 23–27 June 2007; Volume 1, pp. 317–322, doi:10.1109/INDIN.2007.4384776. [CrossRef]

10. Wark, T.; Corke, P.; Sikka, P.; Klingbeil, L.; Guo, Y.; Crossman, C.; Valencia, P.; Swain, D.; Bishop-Hurley, G. Transforming Agriculture through Pervasive Wireless Sensor Networks. *IEEE Pervasive Comput.* **2007**, *6*, 50–57. doi:10.1109/MPRV.2007.47. [CrossRef]
11. Li, M.; Lin, H. Design and Implementation of Smart Home Control Systems Based on Wireless Sensor Networks and Power Line Communications. *IEEE Trans. Ind. Electron.* **2015**, *62*, 4430–4442. doi:10.1109/TIE.2014.2379586. [CrossRef]
12. Nandury, S.V.; Begum, B.A. Smart WSN-based ubiquitous architecture for smart cities. In Proceedings of the 2015 International Conference on Advances in Computing, Communications and Informatics (ICACCI), Kochi, India, 10–13 August 2015; pp. 2366–2373, doi:10.1109/ICACCI.2015.7275972. [CrossRef]
13. Gayan, S.; Weeraddana, D.M.; Gunathillake, A. Sensor network based adaptable system architecture for emergency situations. *Lect. Notes Inf. Theory* **2014**, *2*, 85–91. [CrossRef]
14. Li, Y.; Wang, Z.; Song, Y. Wireless Sensor Network Design for Wildfire Monitoring. In Proceedings of the 2006 6th World Congress on Intelligent Control and Automation, Dalian, China, 21–23 June 2006; Volume 1, pp. 109–113, doi:10.1109/WCICA.2006.1712372. [CrossRef]
15. Barroca, N.; Borges, L.M.; Velez, F.J.; Monteiro, F.; Górski, M.; Castro-Gomes, J. Wireless sensor networks for temperature and humidity monitoring within concrete structures. *Constr. Build. Mater.* **2013**, *40*, 1156–1166. doi:10.1016/j.conbuildmat.2012.11.087. [CrossRef]
16. Yu, T.C.; Lin, C.C.; Chen, C.C.; Lee, W.L.; Lee, R.G.; Tseng, C.H.; Liu, S.P. Wireless sensor networks for indoor air quality monitoring. *Med. Eng. Phys.* **2013**, *35*, 231–235. doi:10.1016/j.medengphy.2011.10.011. [CrossRef] [PubMed]
17. Neethirajan, S. Recent advances in wearable sensors for animal health management. *Sens. Bio-Sens. Res.* **2017**, *12*, 15–29. doi:10.1016/j.sbsr.2016.11.004. [CrossRef]
18. Sandhu, M.; Javaid, N.; Jamil, M.; Khan, Z.; Imran, M.; Ilahi, M.; Khan, M. Modeling mobility and psychological stress based human postural changes in wireless body area networks. *Comput. Hum. Behav.* **2015**, *51*, 1042–1053. doi:10.1016/j.chb.2014.09.032. [CrossRef]
19. Imran, M.; Said, A.M.; Hasbullah, H. A survey of simulators, emulators and testbeds for wireless sensor networks. In Proceedings of the 2010 International Symposium on Information Technology, Kuala Lumpur, Malaysia, 15–17 June 2010; Volume 2, pp. 897–902. doi:10.1109/ITSIM.2010.5561571. [CrossRef]
20. Steyn, L.P.; Hancke, G.P. A survey of Wireless Sensor Network testbeds. In Proceedings of the IEEE Africon'11, Livingstone, Zambia, 13–15 September 2011; pp. 1–6, doi:10.1109/AFRCON.2011.6072072. [CrossRef]
21. Horneber, J.; Hergenröder, A. A Survey on Testbeds and Experimentation Environments for Wireless Sensor Networks. *IEEE Commun. Surv. Tutorials* **2014**, *16*, 1820–1838. doi:10.1109/COMST.2014.2320051. [CrossRef]
22. Kim, H.; Hong, W.K.; Yoo, J.; eun Yoo, S. Experimental Research Testbeds for Large-Scale WSNs: A Survey from the Architectural Perspective. *Int. J. Distrib. Sens. Netw.* **2015**, *11*, 630210. doi:10.1155/2015/630210. [CrossRef]
23. Ma, J.; Wang, J.; Zhang, T. A Survey of Recent Achievements for Wireless Sensor Networks Testbeds. In Proceedings of the 2017 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC), Nanjing, China, 12–14 October 2017; pp. 378–381, doi:10.1109/CyberC.2017.55. [CrossRef]
24. Werner-Allen, G.; Swieskowski, P.; Welsh, M. MoteLab: A Wireless Sensor Network Testbed. In Proceedings of the 4th International Symposium on Information Processing in Sensor Networks, Los Angeles, CA, USA, 24–27 April 2005.
25. Chatzigiannakis, I.; Fischer, S.; Koninis, C.; Mylonas, G.; Pfisterer, D. WISEBED: An Open Large-Scale Wireless Sensor Network Testbed. In *Sensor Applications, Experimentation, and Logistics*; Komninos, N., Ed.; Springer: Berlin/Heidelberg, Germany, 2010; pp. 68–87.
26. Beutel, J.; Lim, R.; Meier, A.; Thiele, L.; Walser, C.; Woehrle, M.; Yuecel, M. The FlockLab Testbed Architecture. In Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems, SenSys '09, Berkeley, CA, USA, 4–6 November 2009; pp. 415–416, doi:10.1145/1644038.1644129. [CrossRef]
27. Bouckaert, S.; Vandenberghe, W.; Jooris, B.; Moerman, I.; Demeester, P. The w-iLab.t Testbed. In *Testbeds and Research Infrastructures. Development of Networks and Communities*; Magedanz, T., Gavras, A., Thanh, N.H., Chase, J.S., Eds.; Springer: Berlin/Heidelberg, Germany, 2011; pp. 145–154.

28. Kulau, U.; Schildt, S.; Rottmann, S.; Gernert, B.; Wolf, L. Demo: PotatoNet–Robust Outdoor Testbed for WSNs: Experiment Like on Your Desk. Outside. In Proceedings of the 10th ACM MobiCom Workshop on Challenged Networks, Paris, France, 7–11 September 2015; pp. 59–60, doi:10.1145/2799371.2799374. [CrossRef]
29. Gernert, B.; Rottmann, S.; Wolf, L.C. PotatoMesh: A Solar Powered WSN Testbed: Poster. In Proceedings of the 17th ACM International Symposium on Mobile Ad Hoc Networking and Computing, MobiHoc '16, Paderborn, Germany, 5–8 July 2016; pp. 391–392, doi:10.1145/2942358.2942411. [CrossRef]
30. Wen, J.; Ansar, Z.; Dargie, W. MobiLab: A Testbed for Evaluating Mobility Management Protocols in Wireless Sensor Networks. *EAI Endorsed Trans. Ubiquitous Environ.* **2017**, *4*, e1. [CrossRef]
31. Appavoo, P.; William, E.K.; Chan, M.C.; Mohammad, M. Indriya2: A Heterogeneous Wireless Sensor Network (WSN) Testbed. In *Testbeds and Research Infrastructures for the Development of Networks and Communities*; Gao, H., Yin, Y., Yang, X., Miao, H., Eds.; Springer International Publishing: Cham, Switzerland, 2019; pp. 3–19.
32. Schaerer, J.; Zhao, Z.; Carrera, J.; Zumbrunn, S.; Braun, T. SDNWisebed: A Software-Defined WSN Testbed. In *Ad-Hoc, Mobile, and Wireless Networks*; Palattella, M.R., Scanzio, S., Coleri Ergen, S., Eds.; Springer International Publishing: Cham, Switzerland, 2019; pp. 317–329.
33. Yi, L.; Ma, J.; Zhang, T. HATBED: A Distributed Hardware Assisted Testbed for Non-invasive Profiling of IoT Devices. In Proceedings of the 2nd Workshop on Benchmarking Cyber-Physical Systems and Internet of Things, CPS-IoTBench '19, Montreal, QC, Canada, 15 April 2019; pp. 13–17, doi:10.1145/3312480.3313172. [CrossRef]
34. Gu, R.; Zhang, H.; Pei, D.; Zhang, J. A Scalable and Virtualized Testbed for IoT Experiments. In Proceedings of the 12th EAI International Conference on Testbeds and Research Infrastructures for the Development of Networks & Communities, Dalian, China, 28–29 September 2017; pp. 24–33.
35. Huh, J.H.; Kim, D.H.; Kim, J.D. NEWSBED: The Internet of Things testbed platform. In Proceedings of the 2017 International Conference on Information Networking (ICOIN), Da Nang, Vietnam, 11–13 January 2017; pp. 492–494. doi:10.1109/ICOIN.2017.7899542. [CrossRef]
36. Huang, M.; Yu, B. Demo Abstract: RPTB: Range-based Positioning TestBed for WSN. In Proceedings of the IEEE INFOCOM 2019—IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), Paris, France, 29 April–2 May 2019; pp. 999–1000, doi:10.1109/INFCOMW.2019.8845273. [CrossRef]
37. Lattanzi, E.; Dromedari, M.; Freschi, V. A Scalable Multitasking Wireless Sensor Network Testbed for Monitoring Indoor Human Comfort. *IEEE Access* **2018**, *6*, 17952–17967. doi:10.1109/ACCESS.2018.2818191. [CrossRef]
38. Alvanou, A.G.; Skiadopoulos, K.; Giannakis, K.; Oikonomou, K.; Tsoumanis, G. Random Walkers Coverage Experimentation and Evaluation in Low-Cost Wireless Home Networks. In Proceedings of the 2019 10th International Conference on Information, Intelligence, Systems and Applications (IISA), Patras, Greece, 15–17 July 2019.
39. Zervopoulos, A.; Skiadopoulos, K.; Giannakis, K.; Oikonomou, K.; Komianos, V.; Tsoumanis, G. Constructing Virtual Backbones over Low-Cost Wireless Networks for Smart Tourism Services. In Proceedings of the 2019 10th International Conference on Information, Intelligence, Systems and Applications (IISA), Patras, Greece, 15–17 July 2019.
40. Zervopoulos, A.; Komianos, V.; Skiadopoulos, K.; Tsoumanis, G.; Spiggos, A.; Giannakis, K.; Oikonomou, K. Constructing Minimal Maintenance Virtual Backbones over Low-Cost Wireless Networks. In Proceedings of the 2019 South Eastern European Design Automation, Computer Engineering, Computer Networks and Social Media Conference (SEEDA-CECNSM), Piraeus, Greece, 20–22 September 2019.
41. Arduino. Available online: https://store.arduino.cc/arduino-mega-2560-rev3 (accessed on 25 October 2019).
42. Arduino. Available online: https://store.arduino.cc/arduino-wirelss-sd-shield (accessed on 25 October 2019).
43. Ferdoush, S.; Li, X. Wireless Sensor Network System Design Using Raspberry Pi and Arduino for Environmental Monitoring Applications. *Procedia Comput. Sci.* **2014**, *34*, 103–110. [CrossRef]
44. Digi. XBee® /XBee-PRO S2C Zigbee® RF Module User Guide. Available online: https://tinyurl.com/y5posdyh (accessed on 25 October 2019).
45. Sachdeva, P.; Katchii, S. A review paper on raspberry pi. *Dimensions (Wash.)* **2014**, *85*, x56mm.

46. Raspberry, P. Available online: https://www.raspberrypi.org/products/raspberry-pi-3-model-b/ (accessed on 26 October 2019).
47. InvenSense. MPU-6050. Available online: https://www.invensense.com/products/motion-tracking/6-axis/mpu-6050/ (accessed on 14 December 2019).
48. SparkFun. Humidity and Temperature Sensor—RHT03. Available online: https://www.sparkfun.com/products/10167 (accessed on 14 December 2019).
49. Vishay. VEML6070 UVA Light Sensor With I^2C Interface. Available online: https://www.vishay.com/ppg?84277. (accessed on 14 December 2019).
50. Sandberg. Sandberg Outdoor Solar Powerbank 16000. Available online: https://sandberg.it/en-mt/product/Outdoor-Solar-Powerbank-16000. (accessed on 14 December 2019).
51. Alotaibi, E.; Mukherjee, B. A survey on routing algorithms for wireless Ad-Hoc and mesh networks. *Comput. Netw.* **2012**, *56*, 940–965. doi:10.1016/j.comnet.2011.10.011. [CrossRef]
52. Skiadopoulos, K.; Oikonomou, K.; Avlonitis, M.; Giannakis, K.; Kogias, D.; Stavrakakis, I. Multiple and replicated random walkers analysis for service discovery in fog computing IoT environments. *Ad Hoc Netw.* **2019**, *93*, 101893. doi:10.1016/j.adhoc.2019.101893. [CrossRef]
53. Lin, C.R.; Gerla, M. Adaptive clustering for mobile wireless networks. *IEEE J. Sel. Areas Commun.* **1997**, *15*, 1265–1275. doi:10.1109/49.622910. [CrossRef]
54. Tsiropoulou, E.E.; Paruchuri, S.T.; Baras, J.S. Interest, energy and physical-aware coalition formation and resource allocation in smart IoT applications. In Proceedings of the 2017 51st Annual Conference on Information Sciences and Systems (CISS), Baltimore, MD, USA, 22–24 March 2017; pp. 1–6.