future internet

Article

A Novel SDWSN-Based Testbed for IoT Smart Applications

Duaa Zuhair Al-Hamid , Pejman A. Karegar and Peter Han Joo Chong *

Department of Electrical and Electronic Engineering, Auckland University of Technology (AUT), Auckland 1010, New Zealand; duaa.alhamid@aut.ac.nz (D.Z.A.-H.); pejman.karegar@aut.ac.nz (P.A.K.)
* Correspondence: peter.chong@aut.ac.nz

Abstract: Wireless sensor network (WSN) environment monitoring and smart city applications present challenges for maintaining network connectivity when, for example, dynamic events occur. Such applications can benefit from recent technologies such as software-defined networks (SDNs) and network virtualization to support network flexibility and offer validation for a physical network. This paper aims to present a testbed-based, software-defined wireless sensor network (SDWSN) for IoT applications with a focus on promoting the approach of virtual network testing and analysis prior to physical network implementation to monitor and repair any network failures. Herein, physical network implementation employing hardware boards such as Texas Instruments CC2538 (TI CC2538) and TI CC1352R sensor nodes is presented and designed based on virtual WSN- based clustering for stationary and dynamic networks use cases. The key performance indicators such as evaluating node (such as a gateway node to the Internet) connection capability based on packet drop and energy consumption virtually and physically are discussed. According to the test findings, the proposed software-defined physical network benefited from "prior-to-implementation" analysis via virtualization, as the performance of both virtual and physical networks is comparable.

Keywords: wireless sensor networks; software-defined wireless sensor network; Internet of Things; testbed; Texas Instruments

check for updates

Citation: Al-Hamid, D.Z.; Karegar, P.A.; Chong, P.H.J. A Novel SDWSN-Based Testbed for IoT Smart Applications. *Future Internet* **2023**, *15*, 291. https://doi.org/10.3390/fi15090291

Academic Editor: Pascal Lorenz

Received: 23 July 2023
Revised: 20 August 2023
Accepted: 25 August 2023
Published: 28 August 2023

Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

The advancement of the Internet of Things (IoT) has paved the way for new requirements for smart cities. This can be represented by a flexible/adaptable operation and an efficient data monitoring system for various real-world applications that may differ in the degree of mobility and performance quality, such as environmental monitoring and intelligent transportation in a smart city [1,2]. IoT-based wireless sensor network (WSN) events may, to some extent, lack flexible network operations, such as dynamic network events that are transient and require a real-time adaptive process. The flexible orchestration and reorchestration of such demanding dynamic networks plays an important role in monitoring the operation of the dynamic physical environment. Newer data gathering methods, processing, and communication frameworks can be arranged to accomplish effective and intelligent process management based on distinct physical process requirements. In this sense, a cloud-based architecture is a potential solution, as it includes a plethora of software-based computational capabilities, such as virtualization, data and knowledge repositories, and more involved operational and analytical tools [3,4]. Virtualization and softwarization are significant components of cloud architecture in this context, as they contribute to network flexibility of certain applications such as vehicular networks (VNs) and can address issues associated with reactions to any operational events [5]. As a means to achieve this, an intelligent, self-organizing network structure, such as VNs and WSN ground networks seen in forests, can be used to enable rapid response, with the topology reorchestrated by software definition. This can be aligned with advances made in software-defined networking (SDN) when operating various operational network phases, such as self-healing mechanisms. This, in turn, encourages the conceptual development of software-defined wireless sensor networks (SDWSN), in which WSN functions are softwarized and integrated into the network. The three core functionalities are represented by the terms "leaf sensor node", "router node", and "IoT gateway node", which can be modelled and tested on a virtual platform before physical deployment [6]. Herein, cloud-based virtualization is used to develop and provide specified functional configuration parameters to physical nodes. As a result, the cloud can assist with performance analysis via the virtual unit, where possible reorchestration of network behavior can be deployed and tested prior to real-world implementation, paving the way for the cyber–physical system [7–9].

This paper intends to focus on generic and adaptive testbed scenarios that can be used in a variety of advanced IoT applications. Herein, we shed more light on the subtleties of general design features of the IoT architecture, where a software-based virtual model can support physical network configuration prior to hardware implementation. The virtual platform represented by the Contiki Cooja network simulator also supports the softwarization concept utilized in this paper by offering the software specifications of the WSN nodes. Furthermore, it offers a network testing environment prior to actual network implementation using Texas Instruments CC2538 (TI CC2538) hardware. Herein, the Contiki firmware is used with both the Cooja tool and the TI CC2538. Mainly, in this paper, we aim to highlight and cover the components involved in the physical network (i.e., testbed) for any form of application. Furthermore, the paper explores use cases for dynamic and stationary networks, as well as how the suggested testbed could support various applications. The physical network implementation is evaluated in this paper as based on the suggested use cases wherein a virtual model is built and tested prior to hardware implementation. To examine the validity of having a virtual model testing in advance, virtual and physical network metrics such as network capacity and performance measures such as packet loss are employed. The proposed system looks into the design from the perspective of virtual and physical network parameters to offer an attempt towards interoperability.

The main contributions of this work are summarized as follows:

- Applying the concepts of "SDWSN" and "virtualization" to dynamic and stationary networks for various IoT applications such as vehicular networks and environmental monitoring systems. Such networks can be designed and tested on a virtual platform with a variety of functional capabilities to offer network flexibility. A network with nodes that can act as leaf and/or gateway functions, for example, can support the network in rapidly responding to any event through the multiple functions embedded in the nodes.
- Proposing a physical testbed with hardware components that use the same virtual network codes. The technique is a cost-effective solution since it takes advantage of virtual testing to test, evaluate, and recover from networks failures.
- The key components for the leaf node and the coordinator node (i.e., router or cluster head) such as the communication rate are discussed, mirroring the core aspects employed in the virtual platform. This interactive collaboration between virtualization and physical implementation can support the efficient operation of networks such as dynamic networks. This paper delves into the aspects of physical network implementation design, including the number of nodes, sensing variables, and mobility. The RSSI sensing variable is utilized for the test, as it can represent the relative mobility of the nodes with the cluster depending on the communication quality. Also, the collected RSSI values based on node mobility can be utilized for detecting the possible failure/departure of a node from its related cluster based on its distance. This can be used by the virtual network to provide the best network reorchestration service when needed. The testing scenarios are designed to evaluate the capacity of the coordinator node based on scalability and sampling rate in terms of packet loss.

The remainder of this paper is structured as follows: Section 2 discusses the state-of-the-art and related work. Section 3 presents the system model. Section 4 evaluates the testbed based on the proposed use cases. Finally, in Section 5, the conclusion of this work is discussed.

2. Related Work

The system architecture and design criteria for real-world applications such as health and environmental monitoring were evaluated in relation to the WSN testbed. Several researchers have highlighted key testbed elements that can be customized to design specifications, such as hardware deployment capabilities [10–14]. Some researchers have emphasized the use of different hardware components such as Raspberry Pi and Arduino [15], as well as use cases such as sensor node distribution for environmental applications.

Considering a use case of data collection from scattered traps on the ground in New Zealand, the forest vegetation region is separated into four parts based on [16] (Claverley, North Taupo, Leader Valley, Purakaunui). At these four sites, the density of effective traps ranges from two to four traps per hectare. The spacing between traps varies based on the species targeted, according to the New Zealand Department of Conservation [17], and it determines the distance between traps in a trap line for rats, stoats, and possums. According to [17], the initial spacing between possum traps is 20 to 40 m, but can be expanded to 100 m if possum population is low. The work in [18] used 108 brushtail possum monitoring sites to estimate possum occupancy rates in wildlife, and the transect of traps is recognized at a distance of 200 m apart. Therefore, the distance between possum traps needs to be specified. At the node level, trap sensors capture field-level data before sending them to the central gateway via entry points such as unmanned aerial vehicles (UAVs).

According to Faiçal et al. [19], the embedded hardware collects and processes data from each sensor using the Raspberry Pi single-board computer as a gateway. The work in [20] employed ultralow-power TelosB nodes as ground sensor nodes distributed in the field to improve accuracy when spraying pesticides while lowering the risk of human exposure to these products. To reduce energy consumption, a precise and scalable data gathering scheme was designed to enable long-distance communication at low bitrates. In [21,22], the authors employed an Arduino-based data collection implementation for both WSN and UAV, as well as a Raspberry Pi at the base station to collect environmental data.

Owing to the high cost and risk of damage associated with hardware testing without prior operational and functional network design, virtualization has long been a widely accepted solution for performing software-based simulation testing and obtaining adaptations for use in physical networks. Efficient use of underlying physical functions is mainly achieved by abstracting them into logical or virtual functions [23]. Software-driven virtualization offers a testing ground for conducting and analyzing soft trials of network scenarios, such as dynamic behavior. Such parallel co-simulation running in the cloud backend can significantly aid in leaning out the network configuration process by means of obviating the hardware requirement (during the testing process). For example, the network simulator Contiki Cooja was adopted as a virtualization platform for certain target hardware (Motes such as TI CC2538 Evaluation Module) [24]. Acharyya et al. [24] emphasized the importance of virtualization in driving towards a flexible IoT-based WSN organization. Herein, by accessing real physical data for the purpose of modelling and simulating virtual networks, the organization herein benefited from improved flexibility and reduced latency by deriving appropriate feedback generated by the virtualization unit. Cloud-based virtualization has been adopted to plan and test various WSN reorchestration scenarios when a dynamic event occurs prior to actual implementation. Network performance (i.e., packet loss, network downtime, etc.) can be analyzed so that the most appropriate reorchestration structure can be applied to the physical network [3,25]. This can support flexible network operation and lessen the impact of unexpected network behavior [7].

Concepts such as SDN and network virtualization necessitate the use of tools that can model and test the capability of a network to be tested on a virtual platform before the actual implementation to avoid any major adjustments that need to be conducted in a physical network structure. Furthermore, dealing with the virtualization platform can facilitate dynamic planning for possible network reorchestration as demanded. The Contiki Cooja virtualization (network simulator) tool [26] was utilized in the works of [24,25,27,28] to reflect some of the mentioned ideologies. Furthermore, with Cooja acting as a virtualization platform, virtual nodes are created by compiling and configuring the same Contiki operating system firmware that is used to configure the actual targeted hardware platform of the Texas Instruments CC2538 sensor nodes. From utilizing the tool for a given application point of view, Karegar et al. [29] proposed a point-by-point air-to-ground communication system that considers the clustering structure for partitioning the ground network into small clusters of sensor nodes distributed over large spaces, through which efficient communication between sensor nodes and a UAV is supported. Herein, the UAV path flight was relaxed by using the possible dynamics in WSN orchestrations, as suggested in the approach. The Contiki Cooja simulator was utilized to establish a communication dialogue between the UAV and the ground WSN, the communication planning method being based on the distance between the sensor nodes and the UAV highlighted by the RSSI measurements.

From the compatibility of the suggested system's point of view, the proposed system model offers interoperable compatibility between the virtual and physical system models. The method's characteristic in [30] is based on defining a range called compatibility rate that can decide the level of strictness and abstractness of the design, embracing SOA (service-oriented architecture) as the base of concept. This can help a service designer to decide on the level of strictness and abstractness of the design by adjusting the compatibility rate. Therefore, this method reduces the effort and time required for designing an IoT service. Our method is also based on designing dynamic and interoperable architecture once and applying that to multiple use cases due to offering flexibility of network orchestration adaptation based on the use case requirements. The proposed network adaptation capability offers full compatibility for the proposed system model.

Also, authors in [31] used an interpretable architecture to offer communication among the SensorThings API and web processing service, enabling interconnection among environmental sensors, data, and applications. Although this integration may offer real-time access to sensor observations simulation results, it lacks the virtualization capabilities that enables a network testing environment prior to physical network implementation.

In summary, limited information is available pertaining to the generic design of a general-purpose testbed to support various applications ranging from low performance to high performance. Furthermore, the various states of network mobility and connectivity can influence the continual data flow relevant to the different applications. Although significant efforts have been made to develop WSN testbed schemes, the real-time configurability of modules that affect system performance, such as ground communication cost and packet delivery rate, as well as network parameters such as node capacity, has received less attention.

3. System Model

IoT-based WSN system architecture can be looked at based on the sensor node layer, the gateway layer, and the cloud layer. Each tier/layer of the system can offer solutions to the overall challenges of a given application while also making the system more robust and scalable. For instance, cloud interaction with the physical WSN can be utilized to explore future operational improvements with various software scenarios using virtualization, historical data, and learning methods. Considering a vehicular network (VN)-related application as an example, Figure 1 shows the cloud-based architecture for a vehicular network, where the physical network of vehicles updates the cloud with data for real-time monitoring and reorchestration purposes. The acquired data from the physical VN is stored in the cloud database offered by the data storage unit, where access to historical real vehicle data is available. These data can be monitored and analyzed based on the knowledge repository to identify any changes in the data pattern and thus the possibility of dynamic change within the network. Therefore, the virtual platform and software resources are employed to respond to an event that necessitates network reorchestration. The virtual model implemented using the Contiki Cooja simulator tool offers the corresponding orchestration and reorchestration via the softwarization of the nodes involved in the network. The figure also reflects the possibility of a node departure and the cloud's response to that event via the flow of control data to the physical network.

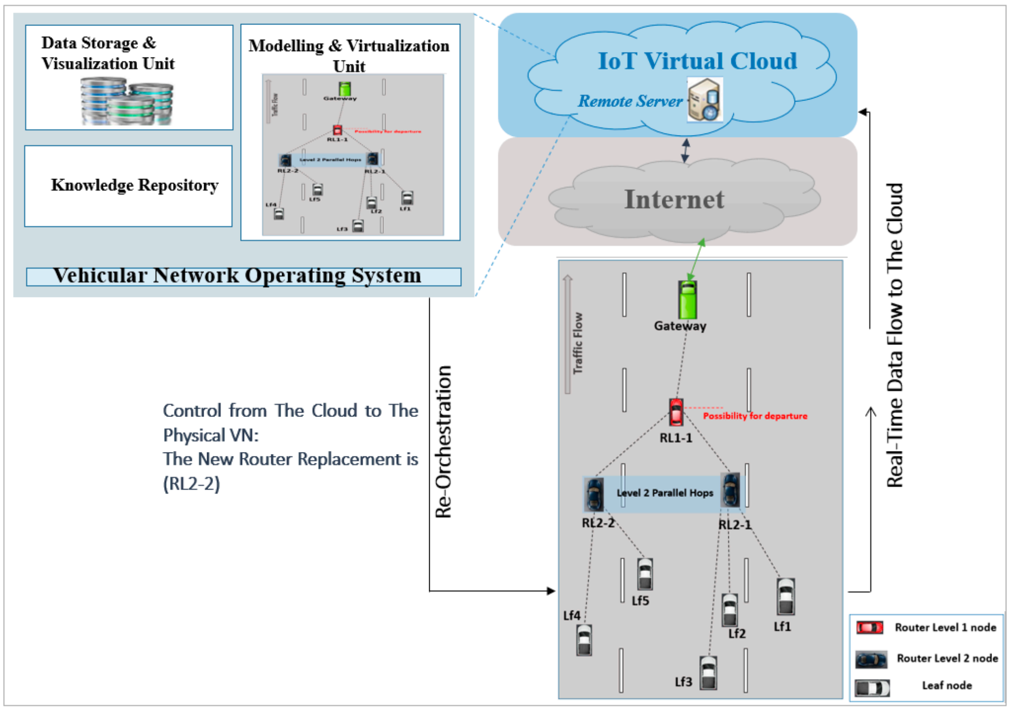

Data Storage & Visualization Unit
Modelling & Virtualization Unit
Knowledge Repository
Vehicular Network Operating System
IoT Virtual Cloud
Remote Server
Internet
Gateway
Traffic Flow
Possibility for departure
RL1-1
Level 2 Parallel Hops
RL2-2
RL2-1
Lf1
Lf2
Lf3
Lf4
Lf5
Control from The Cloud to The Physical VN:
The New Router Replacement is (RL2-2)
Re-Orchestration
Real-Time Data Flow to The Cloud
Router Level 1 node
Router Level 2 node
Leaf node

Figure 1. The interaction between the physical network and the cloud.

From the WSN's point of view, the nodes in our approach can generally be configured with one or more functions. This functional configuration is basically a software component/module. This module is used to configure a specific physical sensor node by enabling the associated hardware libraries. The activities of these functions are presented in Figure 2.

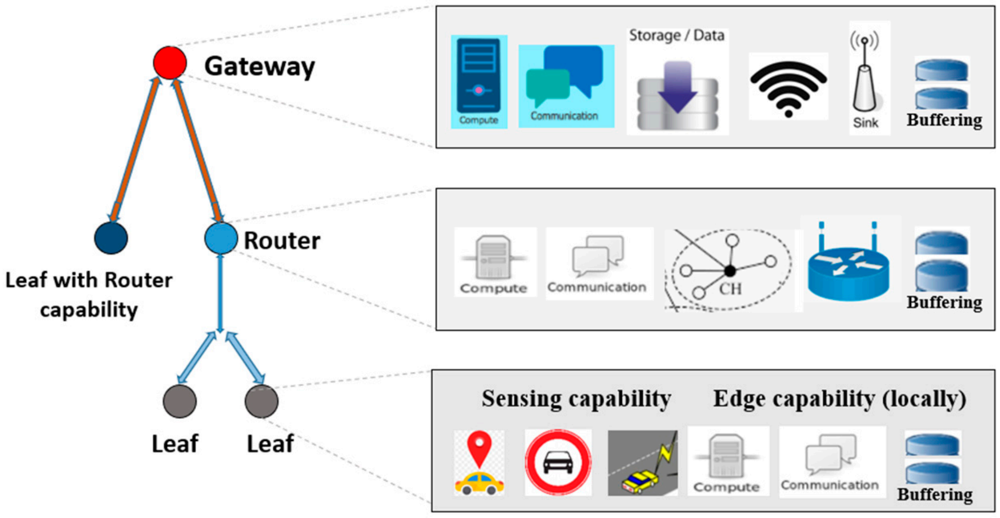


Figure 2. The main activities for WSN functions.

From the communication dialogue that is based on the role/function of the node's point of view, the exchanged data between the nodes can result in the formulation of a star or tree structure depending on the use case. In one of our previous works [7], we presented the communication messages for a self-healing vehicular network. In this paper, we present a general communication dialogue to serve the virtual model as well as the physical network implementation. The roles and communication messages of the participating functional nodes are displayed in a sequence diagram, as illustrated in Figure 3. These are provided in detail as the following:

- To avoid the broadcasting storm in the network, the nodes with routers (coordinator and/or gateway functions) within the LOS (line of sight) can be communicated with to formulate the star or layer(s) of tree structure in some use cases. These nodes are a beacon for message (1), $\mathbf{M_{Hello\_msg}}$, which is a data message that contains the node ID, node type/function, RSSI, etc. Based on these parameters and the number of received "Hello_msg" prompts, the network levels/hops can be allocated.
- Upon allocation of roles to the nodes, the gateway node disseminates message (2), $\mathbf{M_{G\text{-}R}}$, to the router/coordinator node(s) within the LOS. This is an announcement message that indicates the allocation of the gateway node based on the number of received messages.
- The other routers within the LOS disseminate message (3), $\mathbf{M_{R\text{-}G}}$ (acknowledgment), to the gateway node.
- Then, the gateway node disseminates a network formation to the other nodes that are connected it. The network structure could be a star or tree depending on the type of function of the nodes within the transmission range. The message is (4), $\mathbf{M_{G\text{-}Formulating\text{-}Network}}$.
- After establishing the hierarchy of the network, the leaf node(s) can select its connection based on parameters of RSSI and number of connections that the router/gateway can have/already have. For this step, load balancing is required.
- The gateway node disseminates connectivity message (5), $\mathbf{M_{G\text{-}L(n):\ Connectivity}}$, for the unconnected leaf nodes within the LOS. Herein, based on the criteria, the eligible leaf nodes for connectivity send the acknowledgment message (6), $\mathbf{M_{L(n)\text{-}G:\ Acknowledgment}}$.
- The same procedure can be applied for the router/coordinator nodes in connecting to the other leaf nodes, which can discover better connectivity with those nodes. The messages $\mathbf{M_{R\text{-}L(n):\ Connectivity}}$, $\mathbf{M_{L(n)\text{-}R:\ Acknowledgment}}$ are the related messages (7) and (8), respectively.
- As the gateway node is the group head, the other router nodes need to reference or update the gateway with its new connections. Message (9), $\mathbf{M_{R(n)\text{-}G:\ Update}}$, is for the connectivity update.

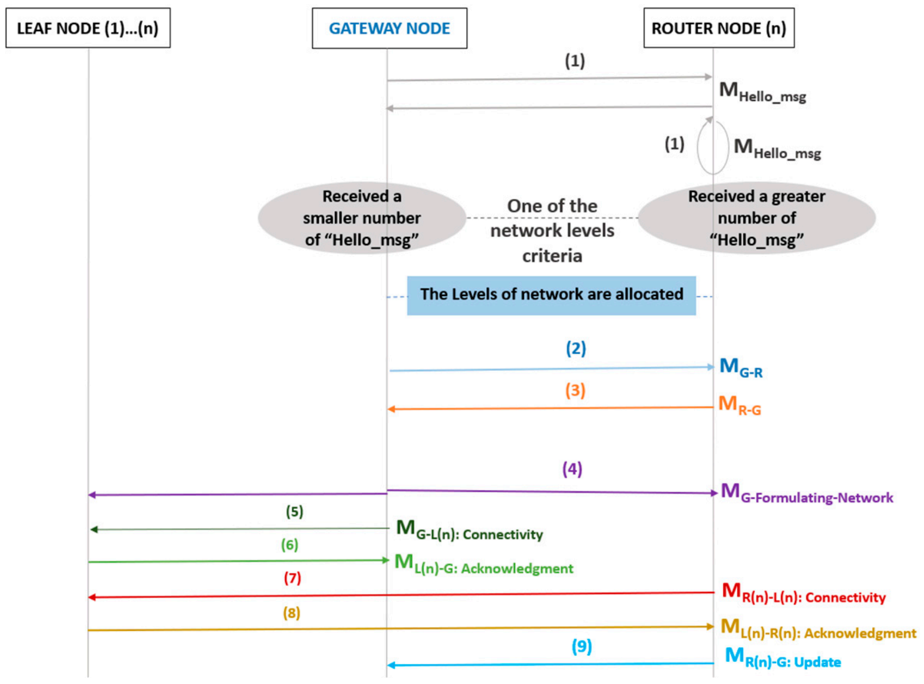

LEAF NODE (1)...(n)
GATEWAY NODE
ROUTER NODE (n)
(1)
M_Hello_msg
(1)
M_Hello_msg
Received a smaller number of "Hello_msg"
One of the network levels criteria
Received a greater number of "Hello_msg"
The Levels of network are allocated
(2)
M_G-R
(3)
M_R-G
(4)
M_G-Formulating-Network
(5)
M_G-L(n): Connectivity
(6)
M_L(n)-G: Acknowledgment
(7)
M_R(n)-L(n): Connectivity
(8)
M_L(n)-R(n): Acknowledgment
(9)
M_R(n)-G: Update

Figure 3. General communication messages for network structure.

The physical WSN testbed is designed to showcase various use cases that can validate the virtual testing available in our earlier works. The testbed for scenarios such as dynamic events and network capacity (size) is highlighted with the use of TI CC2538 due to its alignment with the virtual network, while scenarios such as node distribution with a focus on dual protocols have been seen to be aligned with components such as the TI CC1352R. The use cases as well as the general components of the testbed are further explained in this section.

3.1. The Key Hardware Components for Physical Testbed

The hardware, which represents the physical nodes within the physical network, plays an important role in supporting a given ideology and offering a testing environment for the network. The components that are generally used in testbed design to reflect various use cases and network scenarios are described in the next subsections.

3.1.1. Texas Instruments CC2538 Sensor Nodes

Texas Instruments CC2538 (TI CC2538) wireless transceiver chips consist of on-chip temperature and received signal strength indicator (RSSI) sensors to collect and transmit real-world data. It is an advanced chip with IP configurability. The CC2538 EM is used in conjunction with the SmartRF 06 evaluation board, as shown in Figure 4, for programming purposes, consisting of two on-board sensors for the accelerometer and ambient light sensor. These sensing features, together with the flexibility to configure the TI boards with the same code used for the Cooja simulation motes (i.e., the virtual platform), offer advantages to a variety of use cases wherein network performance can be tested prior to testbed implementation.

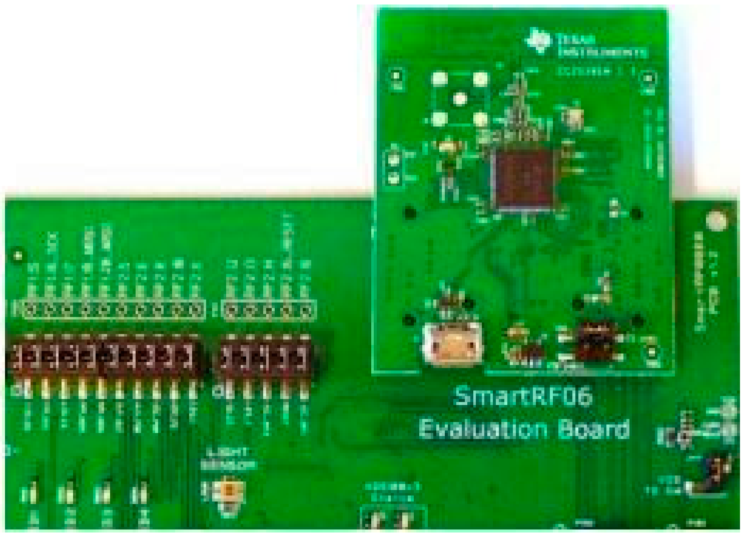


Figure 4. SmartRF06 evaluation board.

These sensor nodes can be configured with either sensing, routing, or/and gateway functionalities through the Contiki IDE. The C code for hardware configuration "include" the related libraries that enable these chips to be configured, such as "#include 'cpu.h'", "#include 'dev/uart.h'", "#include 'dev/adc-sensor.h'", and "#include 'dev/sys-ctrl.h'". Each physical node needs to be assigned a unique ID by accessing the "contiki-conf.h" header file within the Contiki; for example, a coordinator node can have an ID of "0xDA". A channel can also be assigned to the nodes for the purpose of regulating traffic within the shared communication medium by reading the same header file and assigning one of the available and desired channels to the nodes. As an example, channel 25 can be assigned as "#define CC2538_RF_CONF_CHANNEL 25".

From the perspective of mobility, a scenario where one of the nodes is converted into a mobile node by plugging a micro-USB charger into the chip to provide the mobility for conducting outdoor experiments can be used to investigate the influence of RSSI on the physical network. The TI CC2538 evaluation module (EM) with micro-USB is shown in Figure 5.

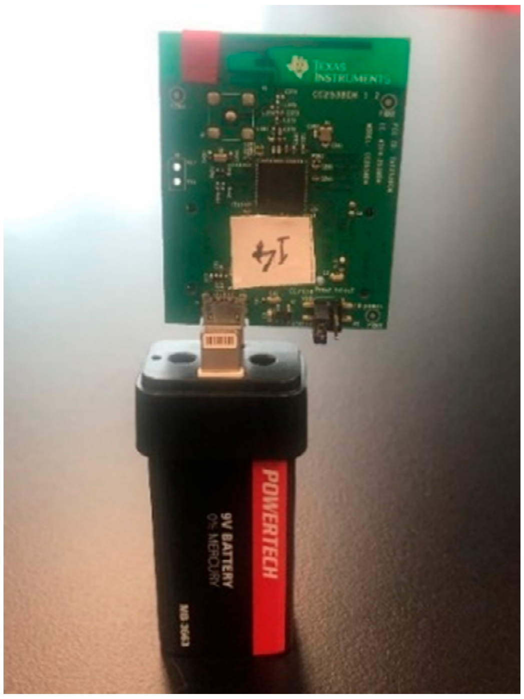


Figure 5. TI CC2538 board with micro-USB.

3.1.2. Texas Instruments CC1352R Sensor Nodes

The LaunchPad SensorTag kit CC1352R, also known as "LPSTK-CC1352R", is a powerful Cortex-M4F MCU with integrated environmental and motion sensors, multiband wireless connectivity, and simple software for prototyping connected applications. Environmental sensors such as humidity and temperature sensors, ambient light and inertia sensors, a Hall effect switch, and a three-axis accelerometer are among the LaunchPad modules that can be used in use cases like trap monitoring in a forest. Mobility is provided by two AAA batteries or a CR2032 coin cell installed on the module. These batteries can provide power for the transmission of their readings in a variety of settings. The LaunchPad modules include a dual-band low-power radio kit that allows for sub-1 GHz (868 MHz/915 MHz) and 2.4 GHz to run concurrently with Bluetooth Low Energy (BLE) in a single-chip solution. The 15.4 Stack protocol offers star topology networks for applications operating at sub-1 GHz or 2.4 GHz. The sub-1 GHz version has several major advantages, including longer range and improved protection against in-band interference, as well as the ability to deliver Bluetooth Low Energy (BLE) beacon packets while running in dual-band mode on a sub-1 GHz 15.4 Stack network. In this case, the dual-band functionality provides the ability to adjust the functionality of a specific SensorTag via over-the-air debugging and BLE configuration. Alternatively, an external programmer–debugger known as the "CC1352R LaunchPad development board" can be used to reconfigure the SensorTag. The LaunchPad SensorTag CC1352R and LaunchPad development board CC1352R are depicted in Figure 6 and Figure 7, respectively.

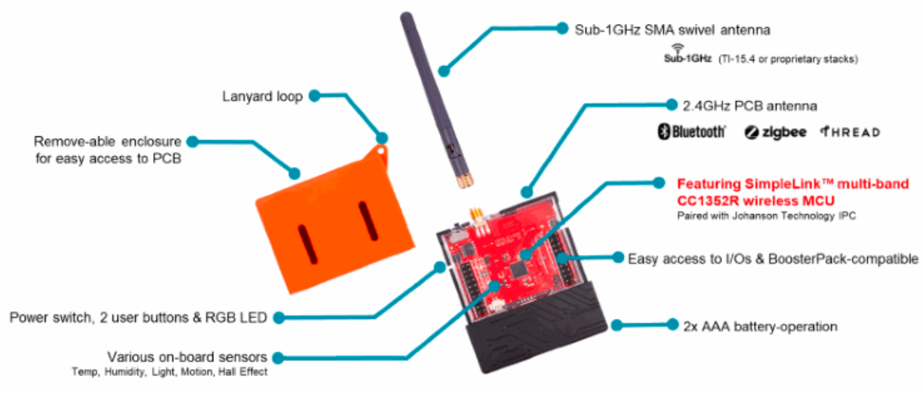

Sub-1GHz SMA swivel antenna
Sub-1GHz (TI-15.4 or proprietary stacks)
Lanyard loop
2.4GHz PCB antenna
Bluetooth zigbee THREAD
Remove-able enclosure for easy access to PCB
Featuring SimpleLink™ multi-band CC1352R wireless MCU
Paired with Johanson Technology IPC
Easy access to I/Os & BoosterPack-compatible
Power switch, 2 user buttons & RGB LED
2x AAA battery-operation
Various on-board sensors
Temp, Humidity, Light, Motion, Hall Effect

Figure 6. The Launchpad SensorTag kit CC1352R.

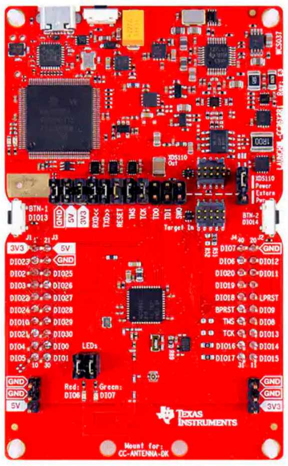


Figure 7. The LaunchPad development board CC1352R.

Code Composer Studio (CCS) software is used as an integrated development environment (IDE) that supports TI's microcontrollers to enable debugging of the targeted module. The CCS provides a package of tools for running, debugging, and altering the functionality of the LPSTK modules, including a C/C++ compiler, source code editor, project build environment, debugger, and profiler. Each LaunchPad SensorTag requires a LaunchPad development kit to be debugged using an ARM 10-pin JTAG cable and a two-wire pair for UART. When the SensorTag is connected to the development board, it allows for full debugging, programming, and UART communication via CCS. The connection between the LaunchPad SensorTag and the LaunchPad development kit to enable debugging of the LaunchPad SensorTag is shown in Figure 8.

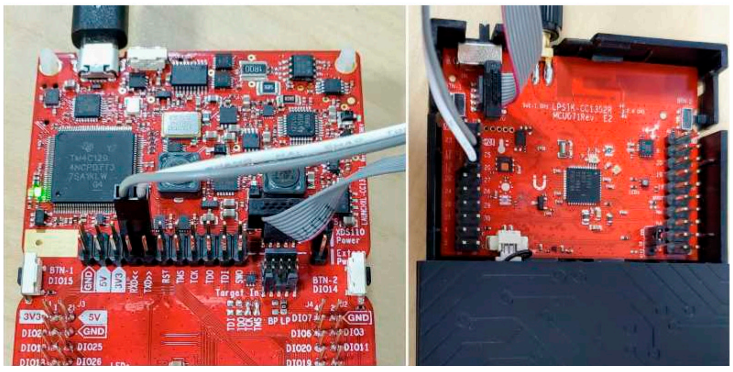


Figure 8. Debugging the LaunchPad SensorTag CC1352R using development board CC1352R.

3.1.3. Raspberry Pi

The widely used Raspberry Pi (RPi) is a small single-board computer that is adaptable, affordable, fully customizable, and programmable with contemporary high-definition multimedia capabilities, as shown in Figure 9, and is equipped with Internet connectivity. It can be seamlessly integrated within the domain of WSN applications for research and prototyping purposes. The RAM memory, CPU (central processing unit), power supply connector, USB ports (where a Wi-Fi USB dongle or adaptor can be attached), ethernet ports, GPU (graphics processing unit), and lower-level peripherals of general purpose input–output (GPIO) pins that can be used for I2C, UART, SPI-based serial communication buses or interfaces, among other components, are all found on a RPi board. It has a slot for an SD card, allowing it to be utilized for datalogging or large-scale data storage (in cases of Internet outage). Operating systems like Raspbian, NOOBS, and others can be used to run RPi. The board can be used as an IoT gateway-capable node to collect, store, and process the data received from sensor nodes in a local database within RPi called SQLite or to communicate the received data from the sensor nodes to a remote database such as MySQL depending on the application. Herein, the stored real data in a remote database can be retrieved and fed into the virtual platform, i.e., the Contiki configuration unit, within the cloud using the appropriate database interface. Contiki can use the real-time data stored in the database from the most recent test of the physical network to compile and configure a given virtual Cooja mote (i.e., the simulation). This can be placed in the simulation window within the simulation platform for virtual testing purposes, thus virtualizing an established physical network. This approach can offer better virtual network testing wherein real data from a physical network is used. The RPi board can also be used as a mobile node by being connected to a portable battery HAT that can be placed on top of the RPI to provide a 5v regulated power supply to the RPi, allowing the board's mobility to meet a specific requirement, mainly for VN scenarios.

Figure 9. Raspberry Pi board.

3.2. Physical Network Setup for WSN-Based Clustering

The TI CC2538 wireless module is one of the hardware targets for developing a physical network structure. The main parameters of sensor data size, sampling rate, network topology, node function, and node mobility all have an impact on physical network performance and possible cloud interaction. This is mainly critical for dynamic networks due to its critical events.

At the Auckland University of Technology (AUT) WZ building, a testbed representing the physical network was deployed to reflect the main functionality of a given topology, e.g., star or tree network scenarios. Within the network, 33 TI CC2538 wireless sensor nodes were configured using the Contiki operating system (OS), with 30 nodes configured as leaf nodes reporting RSSI sensing values within a defined sampling rate and 3 nodes configured as coordinator nodes (cluster heads). The physical network setup is depicted in Figure 10. The physical network setup was designed to log RSSI data for each coordinator node for 24 h to allow for data analysis. Considering that both the hardware and the Cooja motes use the same Contiki firmware, the analysis of the collected data was used to fetch the minimum and maximum RSSI data for the Cooja virtual model in order to update the model based on real data.

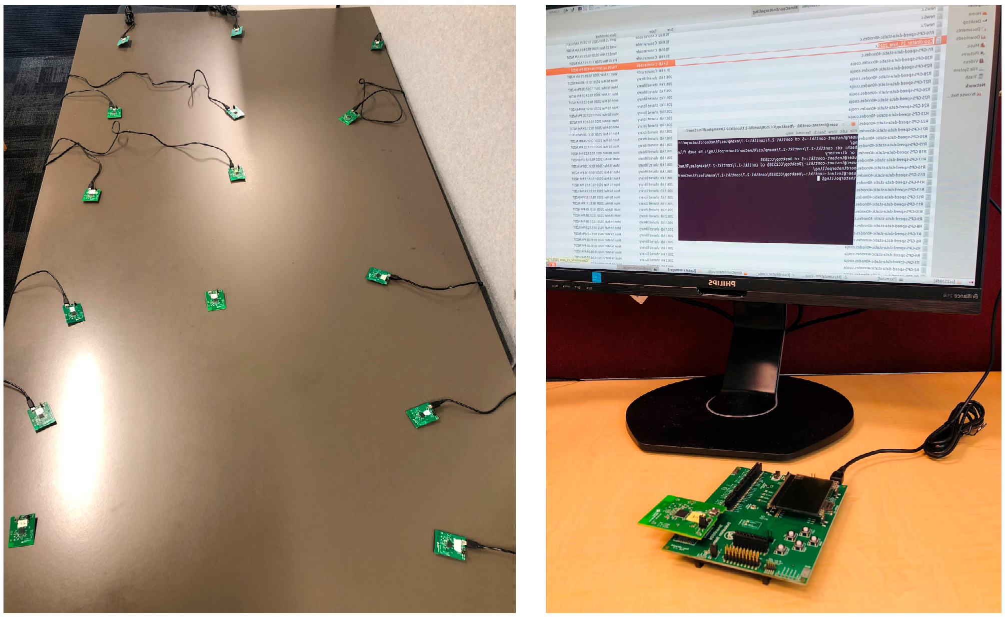


Figure 10. Indoor physical network setup.

The modified values of the RF transmission power of the sensor nodes based on designed indoor and outdoor scenarios can reflect the mobility aspect in the network. Additionally, mobility can be reflected by using a battery attached to one of the nodes to measure RSSI values based on distance, as shown in Figure 11. In addition, the RSSI data are collected at different sampling rates with various number of node scenarios to test the delivery of packets and determine if there is any packet loss. This is mainly to evaluate the capacity of the coordinator (cluster head) node, tested virtually [8,9].

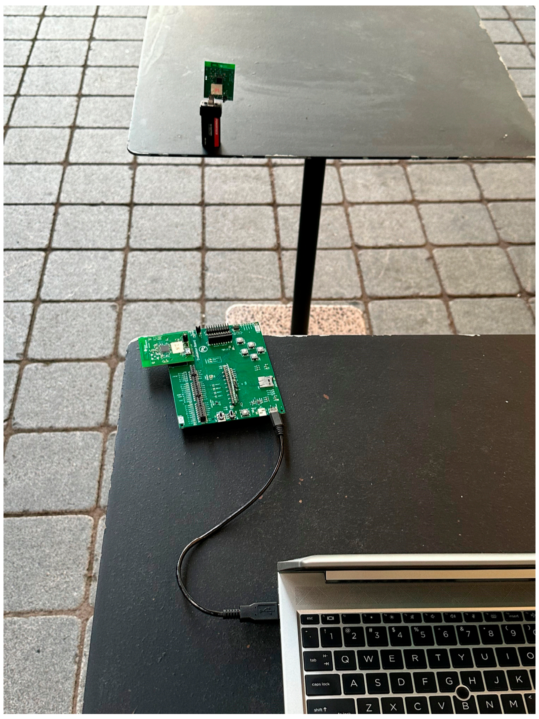


Figure 11. Outdoor physical node mobility testing.

3.2.1. Leaf (End Device) and Coordinator Nodes' Main Components and Data Collection

The same "Contiki OS"-generated codes that are used for compiling and configuring the physical nodes represented by TI CC2538 are used for compiling and constructing virtual nodes. As a result, the logical component of the physical environment's operational dynamics can be accurately reproduced virtually. This allows for flexibility in accessing the physical node code prior to implementation via the virtual mote code. The main components that differ from the virtual mote are the header files, channel allocation, and node ID. Figures 12 and 13 illustrate the pseudocodes for some of the key configuration components of the TI CC2538 leaf node as well as the cluster head (coordinator) node.

```
                                   Leaf Node

RSSI Sensing Function:
rssi=packetbuf_attr(PACKETBUF_ATTR_RSSI) //to acquire the desired RSSI sensing values

Sampling Rate:
etimer_set(&et, CLOCK_SECOND*SAMPLING_RATE_VALUE) // this function is used for setting the value
of the sampling (communication) rate

Buffering the Sensed Data
c[0]=node_ID;  // the node ID is the first element of array 'c'

c[1]=rssi;  // the rssi value is the second element of array 'c'

Transmission to The Group Head Node (Coordinator)

packetbuf_copyfrom(&c, sizeof(c));  //transmitting the array 'c'

               broadcast_send(&bc);

Channel Access Method: TDMA
PROCESS_THREAD(cc2538_demo_process, ev, data)
{
 while(1) {

if(Transmit_Flag==1)
{
packetbuf_copyfrom(&c, sizeof(c));  //transmitting the array 'c'

               broadcast_send(&bc);

Transmit_Flag=0;

}

}

}
```

Figure 12. Pseudocode for configuring TI CC2538 leaf node.

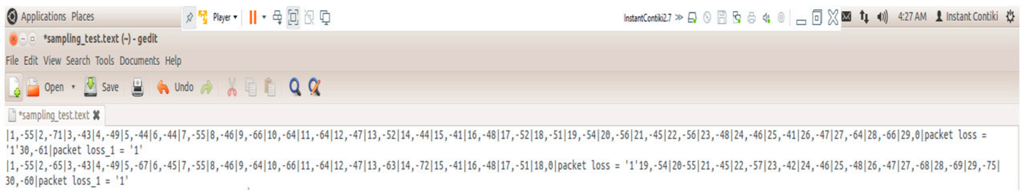
```
                                Coordinator Node

Declaration of Received Data in Arrays
short signed rssi[n]=(0_1,….,0_n ); // the array of 'n' for receiving rssi data from all leaf nodes connected to this
node

Receiving Sensed Variables from Leaf Nodes
//The incoming data from the leaf nodes are stored
int16_t *dataptr_temp1;
dataptr_temp1= (int16_t *)packetbuf_dataptr();
x =dataptr_temp1[0]; //for receiving the node ID of a leaf node
rssi[x] =dataptr_temp1[1]; //receiving rssi data from leaf node 'x' connected to this node

Setting Counter and Printing Data from all The Leaf Nodes
if(counter==number of leaf nodes)
                    {
           counter=(counter % number of leaf nodes);
for (x=1; x<= number of leaf nodes; x++)
{
printf("%d,%d|", x, (rssi[x])); // printing the rssi received by the leaf nodes
}
}

Packet Loss Observation
for (x=1; x<= number of leaf nodes; x++)
{
if (rssi[x]==0)  //setting the condition for considering the packet is lost when the rssi value is not reported by a
leaf node
{
counter_packet_loss++;
printf("packet loss = '%d'", counter_packet_loss);
}
}
printf("packet loss_1 = '%d'", counter_packet_loss);
counter_packet_loss=0;
```

Figure 13. Pseudocode for configuring TI CC2538 coordinator node.

According to Figure 12, the RSSI sensing function, communication/sampling rate that can be set using the same function as for the virtual leaf mote, buffering, and the channel access method at the MAC layer (using the TDMA method in the leaf node as an example) are the main components that reflect the configuration of the leaf sensor node in the established testbed. It is worth noting that the variable "transmit_flag" is not accessible for transmitting the buffer while using the CSMA method.

The declaration of data received from the nodes connected to the coordinator node is shown in an array format in Figure 13. Then, each received sensing datum as well as the node ID of the transmitter node is stored individually. Depending on the number of connected nodes, a counter can be set to print out the data received from the transmitter nodes. In addition, packet loss can be added to the coordinator node by setting any condition that can indicate the loss of a packet.

For RSSI data logging, the data can be sent and received based on the sample codes shown in Figures 12 and 13. A sample of some of the data collected in Contiki in text format is shown in Figure 14. The data received by the coordinator node are displayed, represented by the connected leaf node ID and its RSSI value, along with the amount of packet loss experienced in each round.



```
|1,-55|2,-71|3,-43|4,-49|5,-44|6,-44|7,-55|8,-46|9,-66|10,-64|11,-64|12,-47|13,-52|14,-44|15,-41|16,-48|17,-52|18,-51|19,-54|20,-56|21,-45|22,-56|23,-48|24,-46|25,-41|26,-47|27,-64|28,-66|29,0|packet loss =
'1'30,-61|packet loss_1 = '1'
|1,-55|2,-65|3,-43|4,-49|5,-67|6,-45|7,-55|8,-46|9,-64|10,-66|11,-64|12,-47|13,-63|14,-72|15,-41|16,-48|17,-51|18,0|packet loss = '1'19,-54|20-55|21,-45|22,-57|23,-42|24,-46|25,-48|26,-47|27,-68|28,-69|29,-75|
30,-60|packet loss_1 = '1'
```

Figure 14. A sample of the RSSI data logged to text file.

3.2.2. Mobility Scenario Using Transmission Power

One of the tests involved changing the radio frequency (RF) transmission power output of the TI CC2538 end devices (leaf nodes) based on an indoor scenario (WZ AUT) and an outdoor scenario (around AUT in Auckland) to offer different RSSI values and reflect signal quality. The dynamic alteration of a given node's transmission power output via the C code can be used to evaluate the reliability of communication between the nodes. The hexadecimal value assigned to the selected power mode within the C code can be used to set the intended radio transmission power for the TI CC2538. The power output values used for the indoor and outdoor tests are 22 dBm RF output power with the hexadecimal value 0xFF set in the power function in C code, and 7.5 dBm output power with the hexadecimal value 0 × 42 set in the power function. Figure 15 depicts the power function in C code.

Power function

```
uint8_t
cc2538_rf_power_set(uint8_t new_power)
{
 PRINTF("RF: Set Power\n");

 REG(RFCORE_XREG_TXPOWER) = new_power;

 return (REG(RFCORE_XREG_TXPOWER) & 0xFF); //for 22dBm
}
```

Figure 15. Power function in C code.

The RSSI values for the indoor and outdoor scenarios were logged for three rounds based on the selection of the RF transmission power values modified in the TI CC2538, as shown in Table 1. The signal strength was below −65 dBm in both scenarios (indoor and outdoor), as transmission power was increased. This reflects the reliability in communication as well as the impact of altering transmission power to assume the distance between nodes.

Table 1. Transmission power impact on RSSI indoor and outdoor scenarios.

RF Transmission Power	RSSI Values (dBm) for Indoor Scenario	RSSI Values (dBm) for Outdoor Scenario
22 dBm	−69	−86
	−71	−86
	−67	−87
7.5 dBm	−52	−72
	−55	−78
	−55	−78

3.2.3. Mobility Scenario Using Rechargeable Battery

The RSSI sensing data collected from the physical network are of high importance to dynamic networks as they indicate a node's possible departure when it ventures outside the communication range of the other nodes. The collected RSSI data can be analyzed to observe any variations in the sensed values. A test scenario based on the TI CC2538 network was performed to collect RSSI data from a mobile coordinator node in relation to its leaf node. Herein, the coordinator node was powered using a rechargeable battery to flexibly move away from the leaf node. When the coordinator node moves outside the testing room, the RSSI values range from −72 to −94 dBm, indicating that the node signal strength is degraded. It is worth mentioning that the RSSI-sensed values collected from the physical network can be fed into the virtual network to test any potential network reorchestration. Table 2 shows the RSSI values of the departing coordinator node, which are equivalent to the departed node's distance in meters.

Table 2. RSSI measurements of node movement collected with relation to distance.

RSSI Values of the Departed Node (dBm)	Distance of the Node Moving Away (m)
−72	8
−80	9
−88	11
−92	13
−94	14

3.3. WSN Physical Network Setup for Node Distribution on the Ground

The TI CC1352R can be used for physical network implementation in a variety of applications to allow for sensor node functionality to be modified over the air via cellphone configurability to change some module settings such as bitrate, capture rate, packet size, polling intervals, power, and so on, or to change the entire functionality of a sensor node. The physical network system is softwarized and implemented using LaunchPad SensorTag CC1352R and development boards, as well as Raspberry Pi (RPi). The LaunchPad modules contain a dual-band, low-power radio kit that allows for the coexistence of 15.4 Stack and Bluetooth Low Energy (BLE) on a single chip. The 15.4 Stack is an IEEE 802.15.4e/g RF communication stack that supports star topology networks in either sub-1 GHz (868 MHz /915 MHz) or 2.4 GHz applications. The sub-1 GHz 15.4 Stack is used for ground network connectivity in this paper. The motivation behind this is that the sub-1 GHz version offers numerous advantages, including longer range and improved in-band interference prevention, as well as the ability to broadcast BLE beacon packets while running on a sub-1 GHz 15.4 Stack network in dual-band mode.

The proposed physical ground network topology with the intended dual-band communication functionality is depicted in Figure 16.

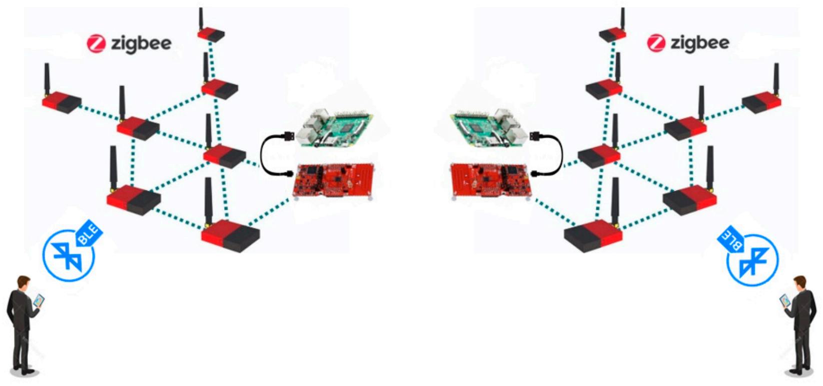


Figure 16. Proposed network architecture for physical network implementation.

The LaunchPad SensorTag CC1352Rs are configured with leaf node firmware, which enables the modules to sense and transmit sensing data to the upper-level coordinators (CC1352R development boards). The gateway nodes are Raspberry Pis with CC1352R development boards connected. The development boards are configured with "Collector firmware" to collect the sensing data from the leaf nodes, while the Raspberry Pi is used for data storage and buffering and is connected to the development board using a USB cable. This physical network topology design has the advantage of enabling dual-band connectivity, allowing the client to reconfigure each physical node using over-the-air debugging (OAD) functionality, which aligns with the softwarization concept. As a result, this network design not only allows for reconfigurability of ground network entities, but it also allows for regular data collection from the ground network within the central RPi station.

3.3.1. Energy Consumption Model for Communication Network

The energy consumption model for enabling communication among the ground network components is calculated based on the proposed model. Hence, the amount of each network components' power consumption for this purpose is as in Equation (1):

$$P_T = P_{Tx} + P_{Rx} + P_{LPM} + P_{Idle} \tag{1}$$

This means the average power consumption of each node is the summation of the average power consumption of a node in four modes: idle mode, low power mode (LPM), and receiving and transmitting modes. The idle mode (P_{Idle}) is activated whenever the node is listening (the time interval during which the CPU is inactive prior to the radio transmitter or receiver becoming active). LPM mode (P_{LPM}) is activated when the sensor node goes to low power mode. Rx mode (P_{Rx}) is activated in radio receive mode and finally Tx mode (P_{Tx}) is activated in transmission mode.

Sensor nodes operate in either active mode or sleep mode. The ratio of the time spent in active mode to a total data period is defined as the duty cycle. In general, sensors mainly consume energy during data receiving and transmitting, and idle listening when they are in active mode. The proposed energy model uses the Contiki power tracker to measure the time intervals that each node spends in these four modes [32]. Hence, the overall energy consumption per node can be calculated considering the equivalent consumed energy for these intervals as in Equation (2):

$$Cost = \sum_{i=1}^{4} P_i \times T_i \tag{2}$$

where P_i represents the value of consumed power within each power mode, T_i is the time spent during a specific mode i.

Power calculation analysis was conducted based on the information explored from the CC2538 datasheet for values of current usages, once the module is in active receiving, transmitting, idle, and low power modes, as shown in Table 3.

Table 3. Power parameters for Cooja mote based on [33].

Active mode TX current consumption	24 mA
Active mode RX current consumption	20 mA
Idle mode	13 mA
Low power mode current consumption	0.6 mA
Supply voltage range	2–3.6 v

3.3.2. Small-Scale Physical Network

This network implementation focuses on non-RPi and RPi network design. The design focuses on the LaunchPad modules, which transfer sensing data such as temperature, RSSI, humidity, accelerometer, and so on to a development board as part of a data collection effort and modifies the functionality of network entities in real time as part of the softwarization process for the network of SensorTags and development boards only. As leaf nodes, Figure 17 displays two sets of modules that include two development boards and two SensorTags. One development board is also assigned the role of coordinator. The SensorTag CC1352R modules are programmed using OAD functionality via Bluetooth Low Energy (BLE) using a smartphone. The leaf nodes provided multiprotocol and dual-band communication over the TI 15.4 Stack and BLE protocols simultaneously. The TI 15.4 Stack is used for communication between leaf nodes and the coordinator, while BLE is used for communication between each module and the smartphone.

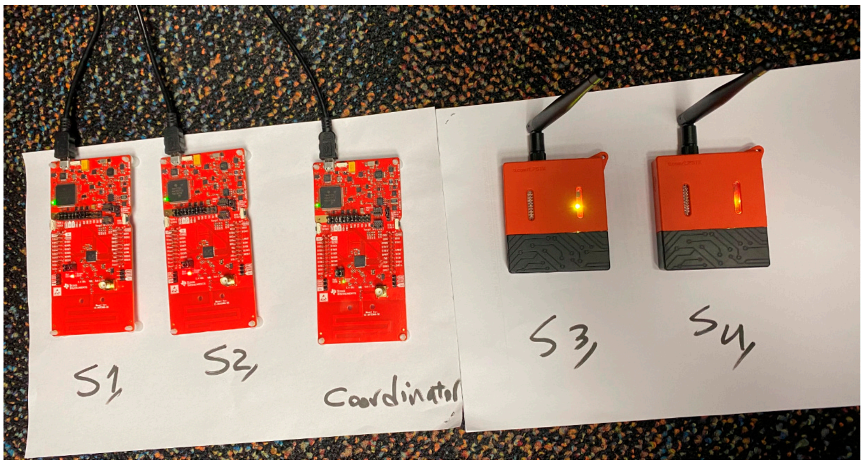


Figure 17. The network configuration for two development boards as sensor nodes and two SensorTags as sensors.

The SimpleLink Starter app for iOS and Android offers significant support for the CC1352R LaunchPad modules, which are used for configurability in this paper. Hence, this app facilitates the OAD to modify the functionality of LaunchPad modules. This app connects the LaunchPad modules to a smartphone through Bluetooth Low Energy (BLE) and supports reading LaunchPad button states, controlling LEDs, and receiving real-time sensing data such as RSSI, temperature, humidity, and accelerometer. It also allows for MQTT cloud communication to the IBM Quickstart server or any other cloud service. This enables a cloud view in which the LaunchPad module can be controlled from any web browser. Figure 18 displays the environment of the Starter application.

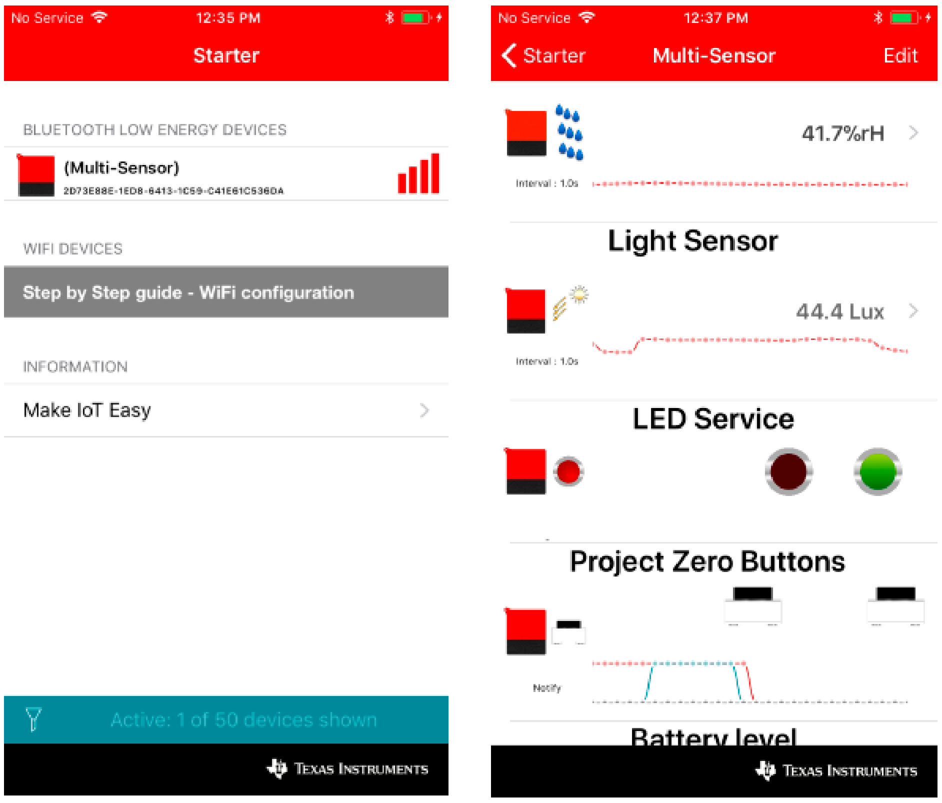
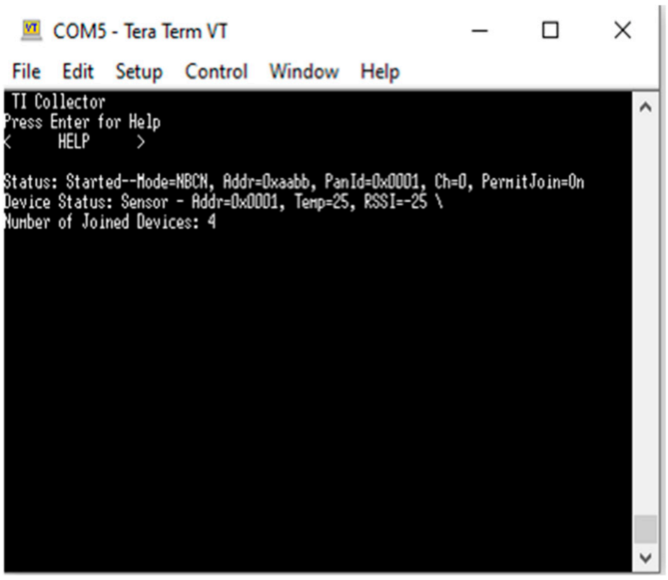


Figure 18. Over-the-air debugging (OAD) and sensing reading on the Starter app.

To program the LPSTK development board as a coordinator, debugging in CCS using the "Collector" program is required. "Tera Term" is used to display the real-time received values after receiving sensing values from various leaf nodes in a coordinator's terminal window using TI 15.4 Stack. The received data from multiple leaf nodes in the coordinator's console is shown in Figure 19. The RSSI and temperature readings regarding this experiment are displayed in the coordinator's console.



```
 TI Collector
Press Enter for Help
<     HELP     >

Status: Started--Mode=NBCN, Addr=0xaabb, PanId=0x0001, Ch=0, PermitJoin=On
Device Status: Sensor - Addr=0x0001, Temp=25, RSSI=-25 \
Number of Joined Devices: 4
```

Figure 19. Temperature and RSSI data received by the coordinator node.

Figure 20 depicts the physical network design with RPi using two SensorTags programmed as leaf nodes and one development board as a coordinator/router node. The development board is connected via USB cable to the RPi. A smartphone is also available to either reconfigure the SensorTags via OAD functionality or to monitor the real-time data of each SensorTag. BLE and TI 15.4 Stack communication protocols are utilized for smartphone–SensorTags and SensorTags–development board interaction, respectively.

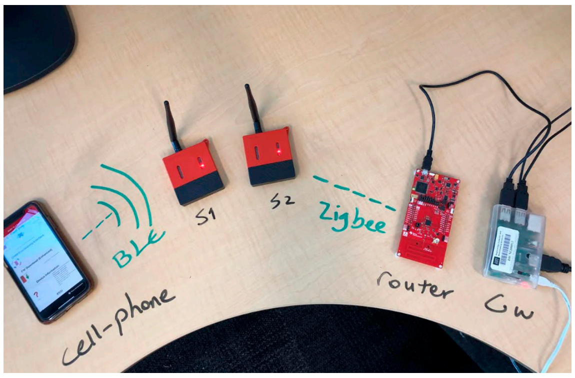


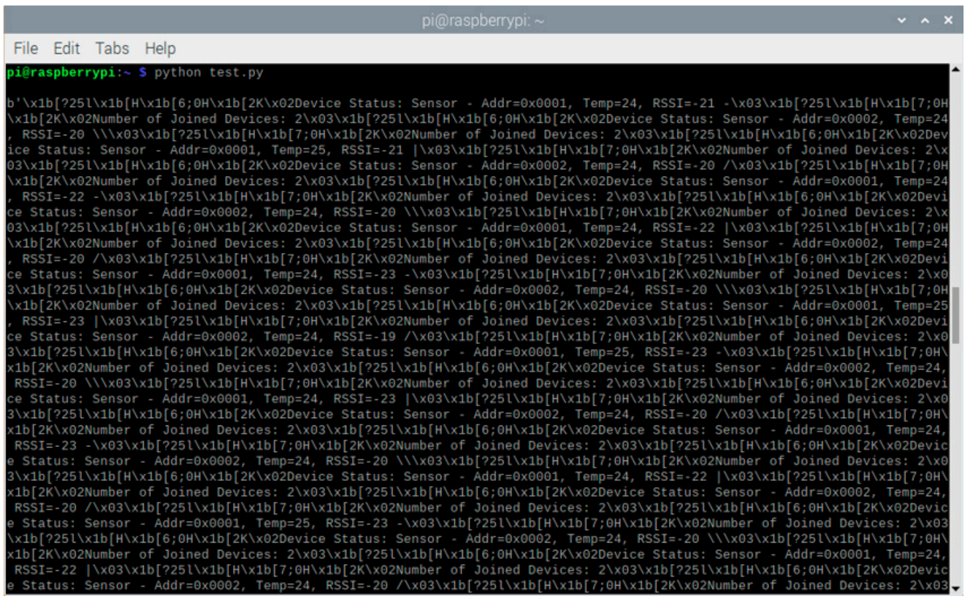
Figure 20. The physical network architecture for TI LaunchPad modules and RPi.

To gain access to the USB port of the RPi and display the real-time outcome on a terminal screen such as "Putty", building a Python script following the Putty installation on "Raspbian" is required, as shown in Figure 21.

```
Import serial
ser=serial.Serial('/dev/ttyACM0',115200)
readedText = ser.readline()
print(readedText)
ser.close()
```

Figure 21. Python script.

The results of running the code in the terminal window, which include the received sensing data from SensorTags on the development board and then on the RPi, are shown in Figure 22.



```
pi@raspberrypi:~ $ python test.py

b'\x1b[?25l\x1b[H\x1b[6;0H\x1b[2K\x02Device Status: Sensor - Addr=0x0001, Temp=24, RSSI=-21 -\x03\x1b[?25l\x1b[H\x1b[7;0H
\x1b[2K\x02Number of Joined Devices: 2\x03\x1b[?25l\x1b[H\x1b[6;0H\x1b[2K\x02Device Status: Sensor - Addr=0x0002, Temp=24
, RSSI=-20 \\\x03\x1b[?25l\x1b[H\x1b[7;0H\x1b[2K\x02Number of Joined Devices: 2\x03\x1b[?25l\x1b[H\x1b[6;0H\x1b[2K\x02Dev
ice Status: Sensor - Addr=0x0001, Temp=25, RSSI=-21 |\x03\x1b[?25l\x1b[H\x1b[7;0H\x1b[2K\x02Number of Joined Devices: 2\x
03\x1b[?25l\x1b[H\x1b[6;0H\x1b[2K\x02Device Status: Sensor - Addr=0x0002, Temp=24, RSSI=-20 /\x03\x1b[?25l\x1b[H\x1b[7;0H
\x1b[2K\x02Number of Joined Devices: 2\x03\x1b[?25l\x1b[H\x1b[6;0H\x1b[2K\x02Device Status: Sensor - Addr=0x0001, Temp=24
, RSSI=-22 -\x03\x1b[?25l\x1b[H\x1b[7;0H\x1b[2K\x02Number of Joined Devices: 2\x03\x1b[?25l\x1b[H\x1b[6;0H\x1b[2K\x02Devi
ce Status: Sensor - Addr=0x0002, Temp=24, RSSI=-20 \\\x03\x1b[?25l\x1b[H\x1b[7;0H\x1b[2K\x02Number of Joined Devices: 2\x
03\x1b[?25l\x1b[H\x1b[6;0H\x1b[2K\x02Device Status: Sensor - Addr=0x0001, Temp=24, RSSI=-22 |\x03\x1b[?25l\x1b[H\x1b[7;0H
\x1b[2K\x02Number of Joined Devices: 2\x03\x1b[?25l\x1b[H\x1b[6;0H\x1b[2K\x02Device Status: Sensor - Addr=0x0002, Temp=24
, RSSI=-20 /\x03\x1b[?25l\x1b[H\x1b[7;0H\x1b[2K\x02Number of Joined Devices: 2\x03\x1b[?25l\x1b[H\x1b[6;0H\x1b[2K\x02Devi
ce Status: Sensor - Addr=0x0001, Temp=24, RSSI=-23 -\x03\x1b[?25l\x1b[H\x1b[7;0H\x1b[2K\x02Number of Joined Devices: 2\x0
3\x1b[?25l\x1b[H\x1b[6;0H\x1b[2K\x02Device Status: Sensor - Addr=0x0002, Temp=24, RSSI=-20 \\\x03\x1b[?25l\x1b[H\x1b[7;0H
\x1b[2K\x02Number of Joined Devices: 2\x03\x1b[?25l\x1b[H\x1b[6;0H\x1b[2K\x02Device Status: Sensor - Addr=0x0001, Temp=25
, RSSI=-23 |\x03\x1b[?25l\x1b[H\x1b[7;0H\x1b[2K\x02Number of Joined Devices: 2\x03\x1b[?25l\x1b[H\x1b[6;0H\x1b[2K\x02Devi
ce Status: Sensor - Addr=0x0002, Temp=24, RSSI=-19 /\x03\x1b[?25l\x1b[H\x1b[7;0H\x1b[2K\x02Number of Joined Devices: 2\x0
3\x1b[?25l\x1b[H\x1b[6;0H\x1b[2K\x02Device Status: Sensor - Addr=0x0001, Temp=25, RSSI=-23 -\x03\x1b[?25l\x1b[H\x1b[7;0H\
x1b[2K\x02Number of Joined Devices: 2\x03\x1b[?25l\x1b[H\x1b[6;0H\x1b[2K\x02Device Status: Sensor - Addr=0x0002, Temp=24,
 RSSI=-20 \\\x03\x1b[?25l\x1b[H\x1b[7;0H\x1b[2K\x02Number of Joined Devices: 2\x03\x1b[?25l\x1b[H\x1b[6;0H\x1b[2K\x02Devi
ce Status: Sensor - Addr=0x0001, Temp=24, RSSI=-23 |\x03\x1b[?25l\x1b[H\x1b[7;0H\x1b[2K\x02Number of Joined Devices: 2\x0
3\x1b[?25l\x1b[H\x1b[6;0H\x1b[2K\x02Device Status: Sensor - Addr=0x0002, Temp=24, RSSI=-20 /\x03\x1b[?25l\x1b[H\x1b[7;0H\
x1b[2K\x02Number of Joined Devices: 2\x03\x1b[?25l\x1b[H\x1b[6;0H\x1b[2K\x02Device Status: Sensor - Addr=0x0001, Temp=24,
 RSSI=-23 -\x03\x1b[?25l\x1b[H\x1b[7;0H\x1b[2K\x02Number of Joined Devices: 2\x03\x1b[?25l\x1b[H\x1b[6;0H\x1b[2K\x02Devic
e Status: Sensor - Addr=0x0002, Temp=24, RSSI=-20 \\\x03\x1b[?25l\x1b[H\x1b[7;0H\x1b[2K\x02Number of Joined Devices: 2\x0
3\x1b[?25l\x1b[H\x1b[6;0H\x1b[2K\x02Device Status: Sensor - Addr=0x0001, Temp=24, RSSI=-22 |\x03\x1b[?25l\x1b[H\x1b[7;0H\
x1b[2K\x02Number of Joined Devices: 2\x03\x1b[?25l\x1b[H\x1b[6;0H\x1b[2K\x02Device Status: Sensor - Addr=0x0002, Temp=24,
 RSSI=-20 /\x03\x1b[?25l\x1b[H\x1b[7;0H\x1b[2K\x02Number of Joined Devices: 2\x03\x1b[?25l\x1b[H\x1b[6;0H\x1b[2K\x02Devic
e Status: Sensor - Addr=0x0001, Temp=25, RSSI=-23 -\x03\x1b[?25l\x1b[H\x1b[7;0H\x1b[2K\x02Number of Joined Devices: 2\x03
\x1b[?25l\x1b[H\x1b[6;0H\x1b[2K\x02Device Status: Sensor - Addr=0x0002, Temp=24, RSSI=-20 \\\x03\x1b[?25l\x1b[H\x1b[7;0H\
x1b[2K\x02Number of Joined Devices: 2\x03\x1b[?25l\x1b[H\x1b[6;0H\x1b[2K\x02Device Status: Sensor - Addr=0x0001, Temp=24,
 RSSI=-22 |\x03\x1b[?25l\x1b[H\x1b[7;0H\x1b[2K\x02Number of Joined Devices: 2\x03\x1b[?25l\x1b[H\x1b[6;0H\x1b[2K\x02Devic
e Status: Sensor - Addr=0x0002, Temp=24, RSSI=-20 /\x03\x1b[?25l\x1b[H\x1b[7;0H\x1b[2K\x02Number of Joined Devices: 2\x03
```

Figure 22. The received data from Temp and RSSI of each SensorTag on RPi terminal.

3.3.3. Large-Scale Physical Ground Network

The data gathering method using the softwarization concept for a ground network [34] is used in the physical network implementation. Herein, the orchestration data gathering phase is validated using a real-world implementation that can be manipulated to be used in applications such as trap monitoring applications. The factor that reflects the dispersing aspect of elected gateway nodes within the ground network after each reconfiguration election process is called sparsity, obtained as in Equation (3):

$$\sigma = \sqrt{\frac{\sum_{i=1}^{N}\left(q_0^i - \mu\right)^2}{N}} \tag{3}$$

wherein σ is the sparsity factor after the election process of i, q_0^i are the coordinates of the elected gateway nodes, μ is the average coordinate of the elected gateway nodes, N is the number of gateway nodes. The simulated post-orchestration data gathering phase is structured to observe the impact of varying the density spread factor parameter on the network's performance. In this section, the same network parameters are used to analyze the physical ground network's performance. Hence, out of 20 nodes distributed on the ground, 17 SensorTags are programmed as leaf nodes, while 3 development boards are programmed as coordinator nodes. Each development board is connected via a USB cable to the RPi node as a gateway. The network organization is based on star network structure. Also, the TI 15.4 Stack communication protocol is used for interaction between the SensorTags and the development board. To validate the softwarization concept in a real-world scenario, 3 coordinator nodes along with 17 leaf nodes were distributed in scenarios with varying densities of gateway nodes, as shown in Figure 23 (σ = 1) and Figure 24 (σ = 2). The data gathering time for each density scenario was set to 60 s. The message rate was also increased from 1 to 20 msg/s. The packet delivery and power consumption were utilized to evaluate the communication performance of the proposed physical network model.

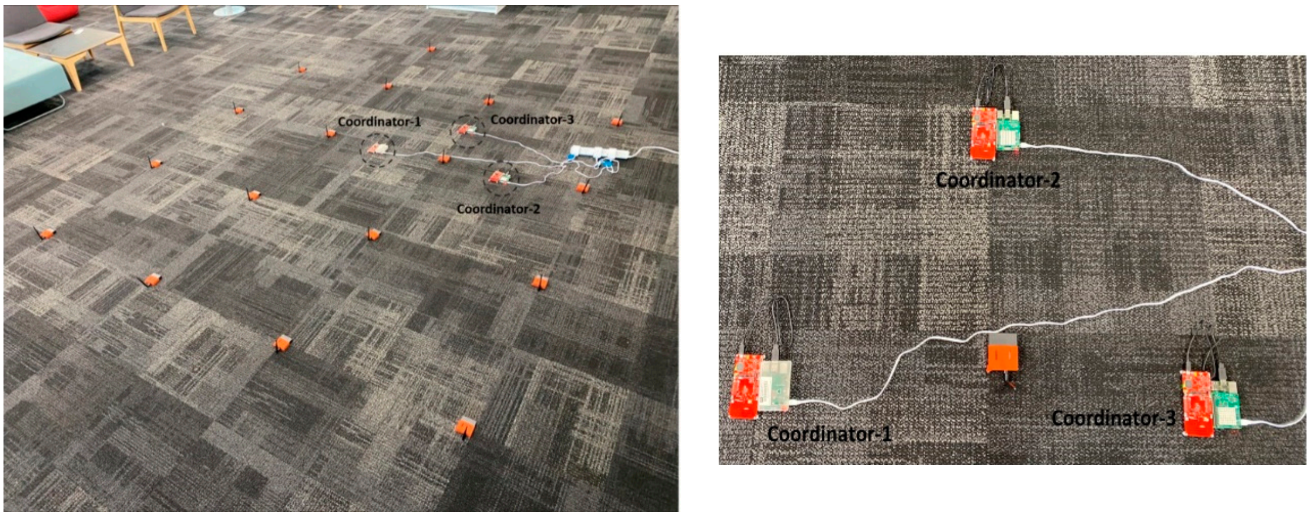


Figure 23. The physical ground network formation for a large-scale network with σ = 1.

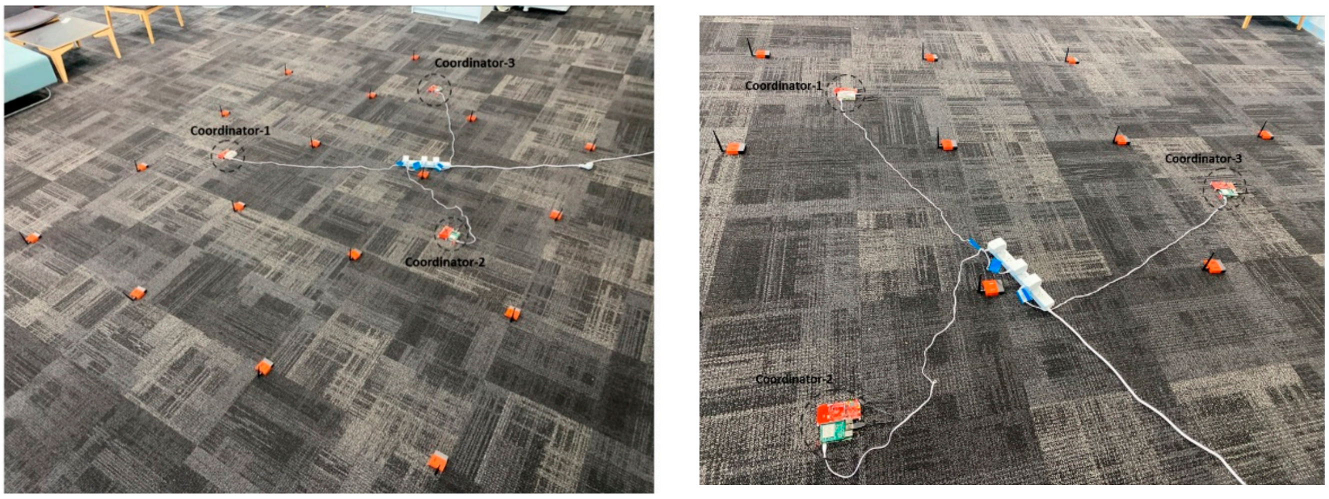


Figure 24. The physical ground network formation for a large-scale network with σ = 2.

4. Model Testing and Evaluation

4.1. Network-Based Clustering Use Case: Router/Coordinator Node Capacity

From a network performance point of view, the network scalability/sparsity scenarios were explored with the increase in sampling rate to reflect the packet loss measurements. Mainly, the testing relates to the network connectivity (router or gateway) capacity tested in the virtual platform using the Contiki Cooja simulator. Herein, the physical network test scenario objective is aligned with our road traffic analysis capacity and virtualization capacity testing scenarios [8,9]. The physical network test was performed, and data were collected when a set of (10 and 20) TI CC2538s connected to a coordinator node reported RSSI values based on a set of sampling rates (1, 2, 3, 4, and 5 samples/s). Figure 25 depicts the total number of packets lost when 10 TI CC2538 nodes are connected to a coordinator node: seven packets at five samples/second. However, the total number of packets lost increases to 12 packets at the same sampling rate when the number of nodes connected to a coordinator node is set to 20 nodes. Although the physical network experienced slightly more packet loss due to real-world factors impacting the quality of the RSSI signal, such as the network deployment environment, obstacles, etc., the physical test indicated the importance of having the virtualization for network soft trials. Variations in sampling rates, the number of nodes, and the type of function of a node in the virtual platform can provide the best scenario for a certain use case that can be realized in the physical network.

4.2. Distributed Nodes for Ground WSN Use Case: Energy Consumption and Packets Received

The physical ground network communication performances including the overall network's energy consumption and the percentage of received packets in coordinator nodes based on a range of gateway sparsity σ and communication message rates are calculated and shown in Figures 26 and 27.

As shown in these two figures, as network density increases, the ground network packet delivery and energy cost for message transmission from leaf nodes (SensorTags) to coordinator nodes (development boards) degrade due to increased interference from other cluster members. Furthermore, as the message rate increases, the percentage of packet delivery in the coordinators' buffers decreases, and the energy consumption of the ground network increases. The reason for this is that as the message rate increases, higher packet intervals of each network component cause the real network components to consume more energy while delivering fewer packets to the coordinator nodes.

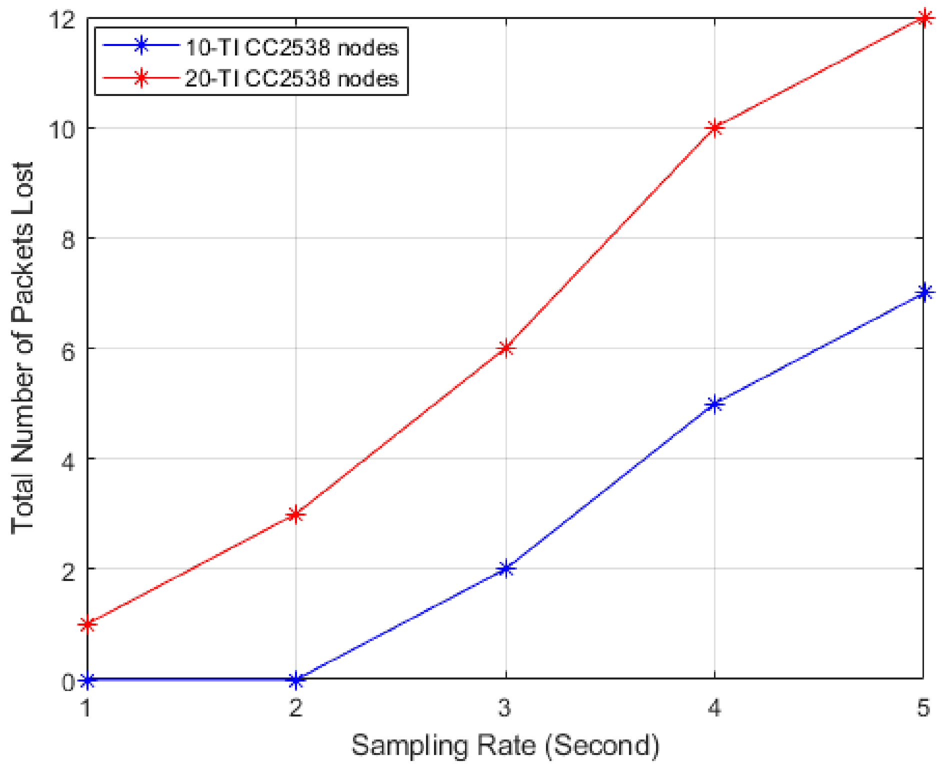


Figure 25. Router capacity based on packet loss.

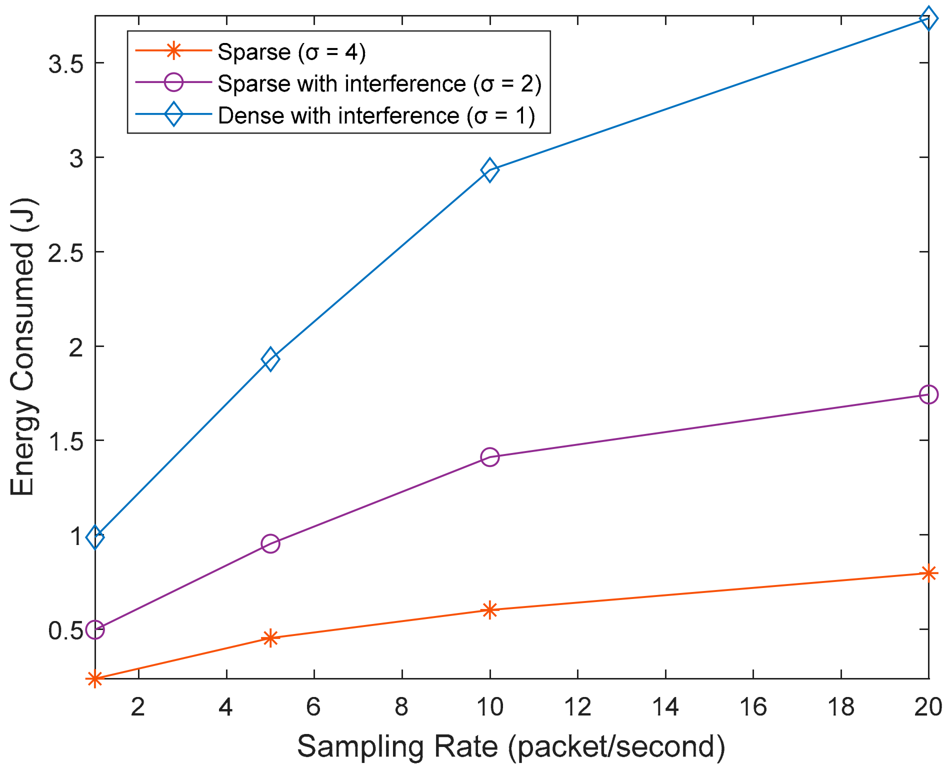


Figure 26. Energy consumption of physical ground network in terms of message rate and gateway nodes' spread factor.

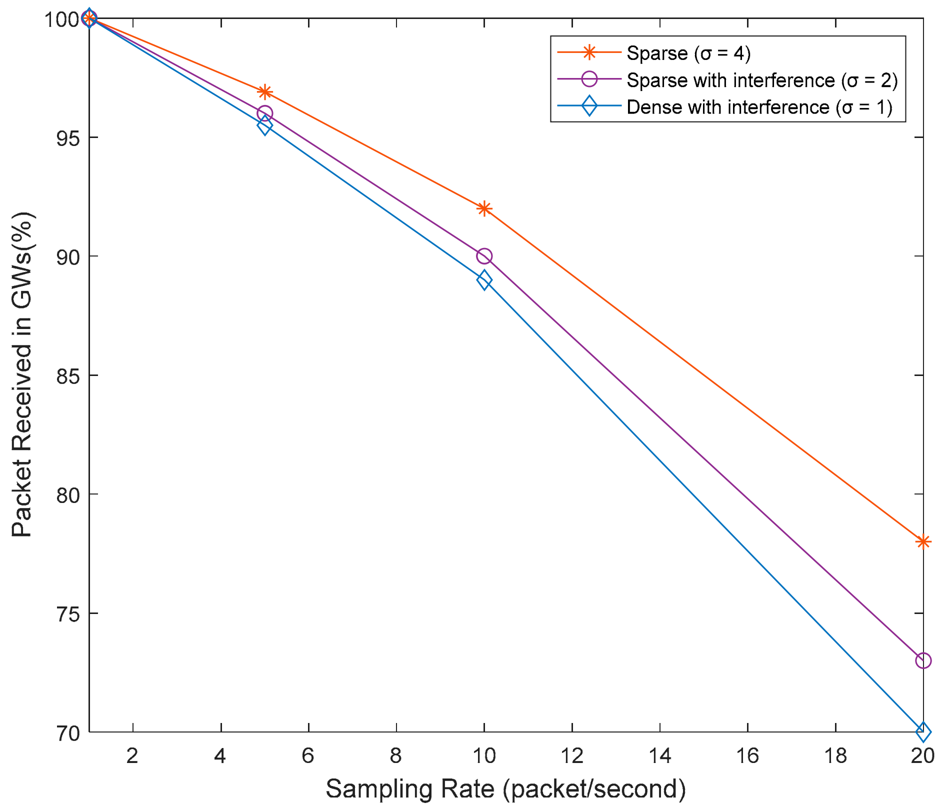


Figure 27. Packets received by coordinators (gateways) from distributed leaf nodes during physical ground network communication.

5. Conclusions

This paper demonstrated physical network implementation using commercial sensor node TI CC2538 with a focus on node capabilities and ability to interact with network operation requirements. Physical network implementation scenarios with a focus on RSSI data were designed to reflect the mobility of the network with the use of rechargeable batteries. This reflected the RSSI values when the node departed from the network and thus the distances were obtained. Also, the change in transmission power indicated the change in RSSI values for indoor and outdoor scenarios. The router/gateway node capacity was tested based on packet loss, indicating slightly more packet loss compared to the virtual network due to real-world factors such as obstacles. This promotes the importance of pretesting on the virtual platform and of enabling interaction between virtual and physical networks. In addition, another use case related to the nodes' distribution on the ground was structured for physical implementation with a focus on stationary scenarios. The proposed physical network topology enables node reconfigurability via virtualization wherein the source code within the virtual platform can be modified, allowing the node to perform various functions depending on the designed scenario. This can be achieved by using the over-the-air debugging (OAD) functionality associated with the proposed TI LaunchPad modules used in this paper. A small-scale network testbed was examined, followed by a large-scale network testbed, using the SensorTag CC1352R LaunchPad, development boards, and Raspberry Pi in order to evaluate network performance variables such as packet delivery rate and consumed energy. The results of physical testing were then compared to the results of simulation testing to validate the proposed concept. This work's limitation is the deployment of the physical network as based on the designed use case. Herein, for future work, the physical network can be implemented on a road to reflect a vehicular network use case as well as in a forest to reflect trap monitoring.

Author Contributions: Conceptualization, D.Z.A.-H. and P.A.K.; methodology, D.Z.A.-H. and P.A.K.; software, D.Z.A.-H. and P.A.K.; validation, D.Z.A.-H.; investigation, D.Z.A.-H., P.A.K. and P.H.J.C.; writing—original draft preparation, D.Z.A.-H.; writing—review and editing, D.Z.A.-H., P.A.K. and P.H.J.C.; visualization, D.Z.A.-H. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Not applicable; this study did not report any data.

Acknowledgments: The authors would like to thank the Department of Electrical and Electronic Engineering in the School of Engineering, Computer and Mathematical Sciences at Auckland University of Technology for providing hardware and advice for this research project.

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

References

1. Saleem, M.A.; Shijie, Z.; Sharif, A.J. Data transmission using IoT in vehicular ad-hoc networks in smart city congestion. *Mob. Netw. Appl.* **2019**, *24*, 248–258. [CrossRef]
2. Memon, I.; Hasan, M.K.; Shaikh, R.A.; Nebhen, J.; Bakar, K.A.A.; Hossain, E.; Tunio, M.H. Energy-Efficient Fuzzy Management System for Internet of Things Connected Vehicular Ad Hoc Networks. *Electronics* **2021**, *10*, 1068. [CrossRef]
3. Ezdiani, S.; Acharyya, I.S.; Sivakumar, S.; Al-Anbuky, A. An IoT environment for WSN adaptive QoS. In Proceedings of the 2015 IEEE International Conference on Data Science and Data Intensive Systems, Sydney, Australia, 11–13 December 2015; pp. 586–593.
4. Ezdiani, S.; Acharyya, I.S.; Sivakumar, S.; Al-Anbuky, A. An Architectural Concept for Sensor Cloud QoSaaS Testbed. In Proceedings of the 6th ACM Workshop on Real World Wireless Sensor Networks, Seoul, Republic of Korea, 1 November 2015; pp. 15–18.
5. Moubayed, A.; Shami, A.J.A.P.A. Softwarization, virtualization, & machine learning for intelligent & effective v2x communications. *IEEE Intell. Transp. Syst. Mag.* **2020**, *14*, 156–173.
6. Karegar, P.A.; Al-Anbuky, A. UAV as a Data Ferry for a Sparse Adaptive WSN. In Proceedings of the 2022 27th Asia Pacific Conference on Communications (APCC), Jeju Island, Republic of Korea, 19–21 October 2022; pp. 284–289.
7. Al-Hamid, D.Z.; Al-Anbuky, A. Vehicular Grouping and Network Formation: Virtualization of Network Self-healing. In Proceedings of the 2018 International Conference on Internet of Vehicles, Paris, France, 20–22 November 2018; Springer: Paris, France, 2018; pp. 106–121.
8. Al-Hamid, D.Z.; Al-Anbuky, A. Vehicular Intelligence: Towards Vehicular Network Digital-Twin. In Proceedings of the 2022 27th Asia Pacific Conference on Communications (APCC), Jeju Island, Republic of Korea, 19–21 October 2022; pp. 427–432.
9. Al-Hamid, D.Z.; Al-Anbuky, A. Vehicular Networks Dynamic Grouping and Re-Orchestration Scenarios. *Information* **2023**, *14*, 32. [CrossRef]
10. Kim, H.; Hong, W.-K.; Yoo, J.; Yoo, S.-E. Experimental research testbeds for large-scale WSNs: A survey from the architectural perspective. *Int. J. Distrib. Sens. Netw.* **2015**, *11*, 630210. [CrossRef]
11. Wu, J.; Jiang, W.; Mei, Y.; Zhou, Y.; Wang, T. A survey on the progress of testing techniques and methods for wireless sensor networks. *IEEE Access* **2018**, *7*, 4302–4316.
12. Fahmy, H.M.A. Testbeds for WSNs. In *Concepts, Applications, Experimentation and Analysis of Wireless Sensor Networks*; Springer: Berlin/Heidelberg, Germany, 2023; pp. 415–545.
13. Mujica, G.; Portilla, J.; Riesgo, T. Testbed architecture and framework for debugging Wireless Sensor Networks. In Proceedings of the 2015 Conference on Design of Circuits and Integrated Systems (DCIS), Estoril, Portugal, 25–27 November 2015; pp. 1–6.
14. Saavedra, E.; Mascaraque, L.; Calderon, G.; Del Campo, G.; Santamaria, A. A Universal Testbed for IoT Wireless Technologies: Abstracting Latency, Error Rate and Stability from the IoT Protocol and Hardware Platform. *Sensors* **2022**, *22*, 4159. [CrossRef] [PubMed]
15. Karegar, M.A.; Kusche, J.; Geremia-Nievinski, F.; Larson, K.M. Raspberry Pi Reflector (RPR): A Low-cost Water-level Monitoring System based on GNSS Interferometric Reflectometry. *Water Resour. Res.* **2022**, *58*, e2021WR031713. [CrossRef]
16. Sweetapple, P.; Nugent, G. Estimating disease survey intensity and wildlife population size from the density of survey devices: Leg-hold traps and the brushtail possum. *Prev. Vet. Med.* **2018**, *159*, 220–226. [CrossRef] [PubMed]
17. Conservation, D.O. Where to put trap and bait lines. Available online: https://www.doc.govt.nz/nature/pests-and-threats/predator-free-2050/community-trapping/trapping-and-toxins/where-to-put-trap-and-bait-lines (accessed on 1 July 2023).
18. Forsyth, D.M.; Perry, M.; Moloney, P.; McKay, M.; Gormley, A.M.; Warburton, B.; Sweetapple, P.; Dewhurst, R. Calibrating brushtail possum (*Trichosurus vulpecula*) occupancy and abundance index estimates from leg-hold traps, wax tags and chew cards in the Department of Conservation's Biodiversity and Monitoring Reporting System. *N. Z. J. Ecol.* **2018**, *42*, 179–191. [CrossRef]
19. Faiçal, B.S.; Freitas, H.; Gomes, P.H.; Mano, L.Y.; Pessin, G.; de Carvalho, A.C.; Krishnamachari, B.; Ueyama, J. An adaptive approach for UAV-based pesticide spraying in dynamic environments. *Comput. Electron. Agric.* **2017**, *138*, 210–223. [CrossRef]
20. Martínez-de Dios, J.R.; de San Bernabé, A.; Viguria, A.; Torres-González, A.; Ollero, A. Combining unmanned aerial systems and sensor networks for earth observation. *Remote Sens.* **2017**, *9*, 336. [CrossRef]

21. Mohamed, N.; Al-Jaroodi, J.; Jawhar, I.; Noura, H.; Mahmoud, S. UAVFog: A UAV-based fog computing for Internet of Things. In Proceedings of the 2017 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computed, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI), San Francisco, CA, USA, 4–8 August 2017; pp. 1–8.
22. Vasisht, D.; Kapetanovic, Z.; Won, J.; Jin, X.; Chandra, R.; Sinha, S.; Kapoor, A.; Sudarshan, M.; Stratman, S. Farmbeats: An IoT platform for data-driven agriculture. In Proceedings of the 14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17), Boston, MA, USA, 27–29 March 2017; pp. 515–529.
23. Khan, I.; Belqasmi, F.; Glitho, R.; Crespi, N.; Morrow, M.; Polakos, P. Wireless sensor network virtualization: Early architecture and research perspectives. *IEEE Netw.* **2015**, *29*, 104–112.
24. Acharyya, I.S.; Al-Anbuky, A. Towards wireless sensor network softwarization. In Proceedings of the NetSoft Conference and Workshops (NetSoft), Seoul, Republic of Korea, 6–10 June 2016; IEEE: Piscataway, NJ, USA; pp. 378–383.
25. Acharyya, I.S.; Al-Anbuky, A.; Sivaramakrishnan, S. Software-defined sensor networks: Towards flexible architecture supported by virtualization. In Proceedings of the 2019 Global IoT Summit (GIoTS), Aarhus, Denmark, 17–21 June 2019; pp. 1–4.
26. Cooja Simulator. Available online: http://anrg.usc.edu/contiki/index.php/Cooja_Simulator (accessed on 1 July 2023).
27. Ezdiani, S.; Acharyya, I.S.; Sivakumar, S.; Al-Anbuky, A. Wireless sensor network softwarization: Towards WSN adaptive QoS. *IEEE Internet Things J.* **2017**, *4*, 1517–1527. [CrossRef]
28. Acharyya, I.S.; Al-Anbuky, A. Software-defined Wireless Sensor Network: WSN Virtualization and Network Re-orchestration. In Proceedings of the Smartgreens, Online, 2–4 May 2020; pp. 79–90.
29. Karegar, P.A.; Al-Anbuky, A. Travel Path Planning for UAV as a Data Collector for a Sparse WSN. In Proceedings of the 2021 17th International Conference on Distributed Computing in Sensor Systems (DCOSS), Pafos, Cyprus, 14–16 July 2021; pp. 359–366.
30. Bin Ahmadon, M.A.; Yamaguchi, S.; Mahamad, A.K.; Saon, S. Physical Device Compatibility Support for Implementation of IoT Services with Design Once, Provide Anywhere Concept. *Information* **2021**, *12*, 30. [CrossRef]
31. Zhang, M.; Yue, P.; Hu, L.; Wu, H.; Zhang, F. An interoperable and service-oriented approach for real-time environmental simulation by coupling OGC WPS and SensorThings API. *Environ. Model. Softw.* **2023**, *165*, 105722. [CrossRef]
32. Amirinasab Nasab, M.; Shamshirband, S.; Chronopoulos, A.T.; Mosavi, A.; Nabipour, N. Energy-Efficient Method for Wireless Sensor Networks Low-Power Radio Operation in Internet of Things. *Electronics* **2020**, *9*, 320. [CrossRef]
33. Texas Instruments. *Datasheet: CC2538 Powerful Wireless Microcontroller System-On-Chip for 2.4-GHz IEEE 802.15.4, 6LoWPAN, and ZigBee Applications*; Texas Instruments Incorporated: Dallas, TX, USA, 2015.
34. Karegar, P.A.; Al-Anbuky, A. UAV-assisted data gathering from a sparse wireless sensor adaptive networks. *Wirel. Netw.* **2022**, *29*, 1367–1384. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.