TECHNICAL UNIVERSITY OF CRETE
ELECTRICAL AND COMPUTER ENGINEERING DEPARTMENT
TELECOMMUNICATIONS DIVISION

Design and Implementation of a Solar Powered Wireless Sensor Network for Autonomous Inference

by

Athanasios Nichoritis

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DIPLOMA OF

ELECTRICAL AND COMPUTER ENGINEERING

July 2021

Abstract

This work studies implementation of an ambiently-powered, distributed wireless sensor network (WSN) capable for in-network inference, using low-cost, low-power embedded nodes. It is found that for 10.8 dBm transmission power, at least 280 meters point-to-point communication link is possible, with packet error rate (PER) less than 2%. Furthermore, a 16-node distributed message passing network is presented and compared to a Zigbee centralized network. It is found that the distributed network provides more flexibility and larger communication range, while the centralized network provides faster execution time at the expense of shorter communication range and coverage. Last but not least, Gaussian Belief Propagation and Average Consensus are implemented in the distributed message passing, solar-powered WSN, demonstrating why asynchrony in message passing is the key.
Thesis Supervisor: Professor Aggelos Bletsas

Acknowledgements

First of all I would like to thank my supervisor Prof. Aggelos Bletsas for his guidance and support throughout this work and beyond that.

My friends and colleagues, P.Vasilakopoulos, V.Papageorgiou for all the help they provided during my work on the laboratory.

My family, my close friends and especially Lampros, nothing would be the same without them.

Contents

List of Figures

Chapter 1

Introduction

1.1 Internet of Things and Wireless Sensor Networks

Wireless sensor networks (WSN) have recently attracted immense research interest with a broad spectrum of possible applications. With the advent of the so-called Internet of Things (IoT), this interest has skyrocketed as by nature WSNs and IoT are closely related. To achieve a large network, each device must communicate with the outside world using low cost and ideally, ultra low-power technology.

Recent advances have enabled the development of low-cost and low-power wireless sensors. Sensors can monitor air quality and track environmental pollutants, wildfires, or other natural or man-made disasters. An important application of sensor networks is in security monitoring and surveillance for buildings, or other critical infrastructure. Microphone or orientation sensors can be very useful in identifying and tracking moving entities, although they require higher-bandwidth communication links. Last but not least, wide area monitoring is possible using environmental sensors such as humidity or temperature sensors. These networks can help farmers have the control of their crops at any time, they can reduce the cost and at the same time that leads to water conservation by irrigating only when and where it is necessary.

In fact, environmental monitoring is one of the earliest applications of sensor networks. An important consideration is the durability of sensors in an unattended environment over an extended period of time.

A WSN is a self-organized wireless network composed of a large number of sensors [4]. WSN nodes typically use an independent power supply, which means they can be easily deployed in large-scale, complex environments.

However, this also means that WSN nodes have very limited energy, memory, and computing resources. So converting the WSN nodes to ambiently powered nodes is a great challenge.

1.2 Ambiently Powered Distributed network

Recent advances on scatter radio sensor networks, have demonstrated feasibility of μWatt power and low-cost, joint sensing and wireless networking; all is needed at the transmitter side is a radio frequency (RF) transistor, a reliable antenna and a low-cost micro controller unit (MCU).

As packet switching set the basic principles of computer networking architecture decades ago, ambient micro-energy harvesting defines new principles for battery-less computing and in-network inference.

Ambient energy, from sun, motion, temperature or even from living animals such as insects or plants has fixed density per cm^2. Consequently, WSNs over an extended area could in principle harvest a large amount of energy (Table 1.1).

Instead of having a central device (like a personal computer or cloud) , which can lead to an increase of cost and complexity, the key is the distributed way. By balancing the WSN computation and communication load of the (distributed) message passing/inference task across various (distributed in space) nodes and by exploiting ultra-low power wireless communication and sensing principles (e.g., scatter radio), autonomous, in-network decisions could be achieved using solely ambient energy.

In-network decisions can be useful for plenty applications. Recent advances on powerful message passing algorithms (e.g., sum-product, max-product), also known as belief propagation, have offered concrete examples on how decision making and inference can be facilitated through communication at carefully crafted graphs.

The vision for this work is the implementation of an ambiently powered, batteryless WSN over an agricultural field which can decides itself where and when to irrigate, without any cloud/edge computing support.

Table 1.1: Ambient Energy Availability & Harvesting Capacity

Energy source	**Ambient Energy Availability**	**Current-Technology Offered Electric Power** (after conversion, incorporating efficiency)
Light/ Solar	35mWatt/cm^2	135mWatt (Polycrystalline Blue Solar Cell 5.4cm x 4.3cm efficiency 16.5%)
Chemical/ Biologic	Voltage from an Avocado plant (*Persea Americana*) 60cm tall	1.15μ Watt @ 21°C, 12.00 pm 1.05μWatt @ 19.5°C, 16.00 pm
RF	0.1μWatt/cm^2 (GSM band)	0.88μWatt (efficiency 6% dipole antenna)
RF	-40dBm/FM station (FM band)	0.018μWatt (efficiency 3% harvesting from six FM stations)
RF	-40dBm/FM station (FM band)	0.003μWatt (efficiency 3% harvesting from one FM station)

In this work conventional, low-cost development kits are used, capable of forming a WSN (Ch. 2). Moreover, a centralized and a distributed network are compared and the advantages of each network are presented (Ch. 3). An energy harvester is designed and implemented, which can exploit the solar energy (Ch. 3). A proof of concept is also provided, for a WSN which can harvest energy from the environment and process the collected information in a distributed manner, by converting the (network) inference task to a probabilistic, message passing problem (Chap. 5). Work is concluded in Chap. 6 in which future plans for extending this work are presented.

Chapter 2

Embedded Hardware

It is nowadays feasible to build a low-power and low-cost WSN [5] by exploiting new technology and techniques. All is needed is a reliable tool to build that system, taking into consideration all the aforementioned parameters.

A sensor node, also known as a mote (mainly in North America), is a node in a sensor network that is capable of performing some processing, gathering sensory information and communicating with other connected nodes in the network.

Wireless sensor nodes communicate via their radio modules. Two nodes are directly connected if they can transmit/receive data to/from each other. A sensor communication model is a mathematical model that quantifies the direct connectivity between sensor nodes.

A commonly assumed communication model is the disk connectivity model according to which, a sensor node can communicate with other nodes located within a disk itself centered within the radius of its communication range.

All the above lead to the conclusion that portables, low-cost, low-power sensor nodes are needed, which must also have large communication range.

Silicon labs' Thunderboard sense 2 [6] meets the aforementioned criteria.

This work studies whether reliable communication and in-network inference are possible, when using the aforementioned low-cost and low-consumption kits as network's nodes.

2.1 Specification

Thunderboard Sense 2 is a compact, feature-packed development platform [6]. The development platform combines a broad range of sensors with the powerful EFR32 radio. Thunderboard Sense 2 also has an on-board J-Link debugger and is fully supported in Simplicity Studio.

The core of the Thunderboard Sense 2 is the EFR32MG12 Mighty Gecko Wireless SoC. The board also contains a variety of sensors, including various environmental sensors and a motion sensor, all connected to its MCU (EFR32MG12). The user interface components include three push buttons, a two colour LED, and four high brightness RGB LEDs.

Wireless Gecko EFR32™ provides :

- Wireless SoC with multi-protocol radio
- ARM Cortex® M4 core with $256kB$ RAM and $1024kB$ Flash
- Low Energy Consumption
- Flexible MCU preipheral interfaces

The Kit Features are the following:

- 2.4 GHz radio configuration with on-board antenna
- Segger J-Link Integrated Debugger
- USB CDC virtual serial port
- $8Mbit$ SPI ultra low power NOR flash for local storage

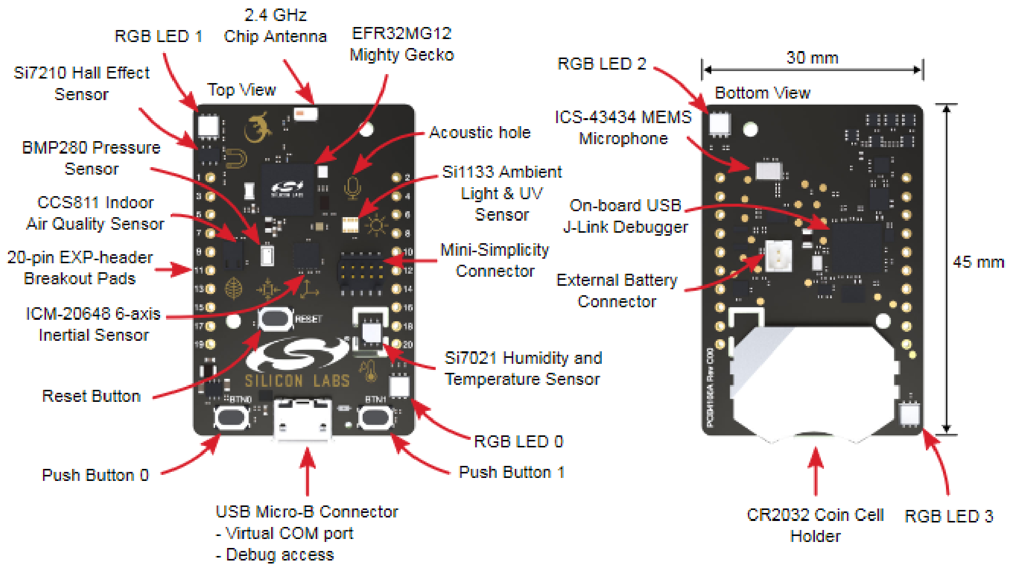
It is obvious from the above, that sensor nodes are vital component in order to build a reliable and large-scale WSN. Thunderboard's embedded sensors provide plenty of options for a lot of real life applications.

In particular Thunderboard Sense 2 provides [2] :

- Relative Humidity and Temperature Sensor Si7021
- UV and Ambient Light Sensor Si1133
- Pressure Sensor BMP280
- Indoor Air Quality and Gas Sensor CCS811
- 6-axis Inertial Sensor ICM-20648

- Digital Microphone ICS-43434
- High brightness LEDs
- Hall-effect Sensor Si7210

The layout of the Thunderboard Sense 2 is shown below.



2.4 GHz Chip Antenna
RGB LED 1
EFR32MG12 Mighty Gecko
Si7210 Hall Effect Sensor
Top View
BMP280 Pressure Sensor
CCS811 Indoor Air Quality Sensor
20-pin EXP-header Breakout Pads
ICM-20648 6-axis Inertial Sensor
Reset Button
Push Button 0
Acoustic hole
Si1133 Ambient Light & UV Sensor
Mini-Simplicity Connector
Si7021 Humidity and Temperature Sensor
RGB LED 0
Push Button 1
USB Micro-B Connector
- Virtual COM port
- Debug access
RGB LED 2
30 mm
Bottom View
ICS-43434 MEMS Microphone
On-board USB J-Link Debugger
External Battery Connector
45 mm
CR2032 Coin Cell Holder
RGB LED 3

Figure 2.1: Thunderboard sense 2 Hardware Layout [2]

In normal operation, power can be applied using either a USB cable connected to a power source, or a battery connected to one of the battery connectors. The 5V power from the USB bus is regulated down to 3.3V using a low-dropout regulator. Batteries can be connected to the Thunderboard Sense 2 using either the CR2032 coin cell holder or the external battery connector.

Silicon laboratories provides a software tool called Simplicity Studio for configuring Thunderboard. The tool performs automatic code generation depending on the desired radio configuration.

Programming the Thunderboard Sense 2 is easily done using a USB Micro-B cable. A USB virtual COM port provides a serial connection to the target application. Included on the board is a 8Mbit serial flash non-volatile memory.

Connecting external hardware to the Thunderboard Sense 2 can be easily done using the 20 breakout pads, which present peripherals from the EFR32MG12 Mighty Gecko such as SPI, UART, I2C and GPIOs.

Silicon Labs provides also a free mobile application for both Android and IOS, using Bluetooth, in order to test the board features, such as sensors' values in real time.

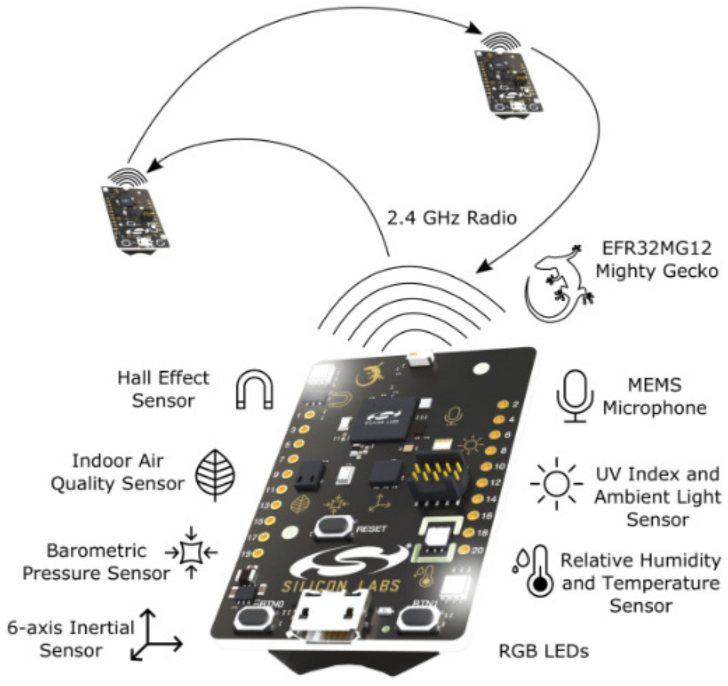

2.4 GHz Radio
EFR32MG12 Mighty Gecko
Hall Effect Sensor
MEMS Microphone
Indoor Air Quality Sensor
UV Index and Ambient Light Sensor
Barometric Pressure Sensor
Relative Humidity and Temperature Sensor
6-axis Inertial Sensor
RGB LEDs

Figure 2.2: Thunderboard sense 2 [2]

The board has plenty of advantageous characteristics which makes it a useful tool to build a large-scale and reliable WSN.

First and foremost, Thunderboard Sense 2 is significantly low-cost given all the aforementioned capabilites. It costs 19.99 $.

Thunderboard's operation at 2.4 GHz and the 32-bit MCU are two big advantages compared to other low-cost and low-power boards. One of the primary advantages of a 32-bit microcontroller over an 8-bit microcontroller is its superior processing speed. A typical 8-bit microcontroller usually runs at 8 Mhz while a 32-bit microcontroller, such as Thunderboard Sense 2, can be clocked up to hundreds of Mhz. A 32-bit microcontroller could be used in more complicated calculations, while, a 8-bit microcontroller is basically used to execute simple logical operations such as addition, subtraction, multiplication, division, etc.. Comparing to 32-bit microcontroller, 8-bit is way smaller in storage space and its calculating capability is weaker.

This is a multi-protocol capable kit for developing connected IoT devices, supporting proprietary stacks and standard protocols such as Zigbee, Thread, and Bluetooth Low Energy (BLE).

Lots of mini software examples are provided in Simplicity Studio, with minor differences depending on the kit used (different MCU, antenna etc). Also, these examples cover a wide variety of different applications, where Silicon Lab's kit can be useful. From a simple transmit-receive example, to more complicated ones, such as the basic Zigbee network. That means it is easier for a new developer to get started with programming hardware kits.

This work takes advantage of RAIL proprietary SDK, but also uses Zigbee protocol. In Chap. 3.2 the centralized network is presented using Zigbee protocol. In Chap. 3.1, the distributed network is presented which takes advantage of the proprietary SDK.

2.2 Point-to-Point communication

The first step implemented was the point-to-point communication between two nodes (terminals). To achieve this communication, Radio Abstraction Interface Layer (RAIL) proprietary SDK was used [7]. The procedure resembles to ping pong. Each terminal sends a message and waits for a response, while the other terminal follows the reverse procedure.

Frequency domain implementation was selected for the communication (similarly to the distributed network in Chap. 3.1). That means, each terminal listens at its own channel (a specific frequency) and transmits messages to a different channel, the one that the target terminal listens to.

The frequency band provided by Silicon Labs is flexible enough in order to build a multi-node WSN. The center frequency the system operates is 2.450MHz. Channel spacing was set manually at 500kHz. Using lower channel spacing value, Cross Frequency Coupling was observed. Given the aforementioned value, the band allows 128 channels. In frequency domain, that means that a 128-node WSN can be implemented, in which, each terminal listens at a different channel. A drawback noticed with the method above, was that some frequencies interfere with some others when the physical distance between the nodes is small. So, some nodes must be at least two-three meters apart (e.g channel 7 with channel 13).

RAIL SDK provides a user friendly interface for the radio configuration. The parameters and their values for the point to point communication (which will be generalized to a large WSN in Chap. 3.1) are shown in Table 2.1.

Table 2.1: Radio Configuration

Parameter	Value
Modulation	2-FSK
Data Rate	2.4 kbps
Center Frequency	2.450 MHz
Frequency Deviation	±10.8 kHz
RX Bandwidth	160 kHz
TX Power	10.8 dBm

The simple transmit-receive (trx) software example provided by Silicon Labs was taken into consideration, in order to achieve point to point communication between two nodes. In such a communication, one node starts the transmission (tx) of a message, then goes to receive (rx) mode and waits for the response. The other node starts in receive mode, and as soon as it receives a packet, it sends its answer back. The embedded LEDs were enabled (toggle) on each node when receiving a packet, for confirmation.

After indoor communication has been succeeded, the next goal was the implementation of an outdoor and reliable communication link.

In order to achieve that communication, Frequency Shift Keying (FSK) modulation was used. In binary FSK modulation, bit rate equals to data rate. A wide and reliable wireless communication system, especially for area monitoring, requires wide communication range. So, a very crucial attribute for this board, was whether long range communication could be achieved.

2.2.1 Radio Configuration

Fundamentally, the range is dictated by the communication data rate. So, a low bit rate of 2.4kbps has been chosen for the proposed link.

In real cases, the noise floor grows due to other contributors such as: Environmental noise, PCB generated noise. Environmental noise and PCB generated noise also degrade the performance of the receiver by increasing the noise level. This kind of noise is dependent on the environment, the frequency band and the PCB and it is more subtle to characterize. PCB generated noise, is a parameter that needs to be investigated in order to achieve the optimal radio receiver performance.

Additionally, having a narrow filter at the receiver effectively lowers the noise floor. The spectral density of noise is given by,

$$N_0 = k \cdot T \text{ watt/Hz}, \tag{2.1}$$

where $k = 1.38 \cdot 10^{-23}$ Joule/K the Boltzman's constant, and T the temperature in Kelvin. For a temperature of 27°C (300 K), the above equation results:

$$10 \log N_0 = -174 \text{ dbm/Hz}. \tag{2.2}$$

The above value represents the power density of noise, measured in dBm per unit of bandwidth, at room temperature (27°C). If that value is multiplied by the bandwidth at which the receiver operates, the result is the noise floor. This noise floor is further increased by a factor called Noise Figure (NF). Noise Figure is a measure of how much noise does the operation of the system itself, add to the the weak received signal. So the noise floor seen by the receiver is:

$$P_{noise} = N_0 \cdot BW \cdot NF. \tag{2.3}$$

Anything above P_{noise} level will be picked up by the receiver as useful signal. However, some systems request higher input signal to achieve a certain “quality” (in digital communication the quality may be quantified with the Bit Error Rate) in the recovered signal. The amount of how much higher above the noise floor the signal must be, is defined as minimum Signal to Noise Ratio (SNR_{min}).

The quantity that indicates the minimum power a signal may have, so that the receiver may work with it, is called *Receiver Sensitivity* and it will be referred to as P_{min},

$$P_{min} = P_{noise} \cdot SNR_{min}, \tag{2.4}$$

where :

$$P_{min}(dBm) = \underbrace{10\log(N_0)}_{-174\ \text{dbm/Hz @ } 27°\text{C}} + 10\log(BW)(Hz) + 10\log(NF) + SNR_{min,dB}. \tag{2.5}$$

expressed in a log scale using dB.

Having mentioned the above, it is obvious now that the less bandwidth a receiver is working on, the less power a signal induced in its antenna may have to be acceptable and so greater communication ranges could be achieved.

The minimum value for receiver’s bandwidth in which the largest point to point communication was observed is 160kHz. During the experiments, smaller values of bandwidth were set, but the range achieved was not the largest possible.

2.2.2 Outdoor communication link

Thunderboard’s 2.4 GHz ceramic on-chip antenna was used for the communication between the nodes. After setting the parameters (Table 2.1) that maximize the communication range, point to point communication was tested outdoor, and the antenna had to be checked whether it was sufficient enough to build a WSN, or an external antenna was needed.

The nodes have been placed in small boxes, and relied on 1.40m tall timbers, in order to maximize the communication range and minimize the noise (Fig.2.3). Minor changes in communication range were observed when the kit's antennas were in line, compared to random orientation.

When using the maximum transmission power (10.8dBm), the range measured between two Thunderboard Sense 2 nodes which exchange messages, was 280m. No packet error was observed up to 280m. After that point, some packet errors were observed and the packet success drops to about 50% at 290m. Hence, it is clear the performance of on-chip antennas at 2.4 GHz turned out to be sufficient to satisfy the communication link requirements, and there is no need for an external antenna.

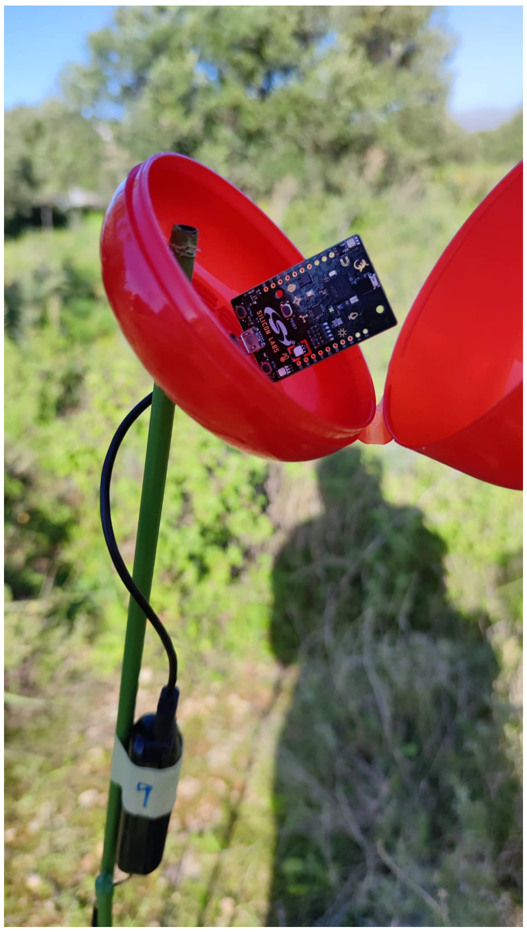


Figure 2.3: Thunderboard setup outdoor.

The model assumes that between the transmitter and receiver an unobstructed path(line-of-sight path) exists. In a real-world environment, a single direct path between the transmitter and receiver is unlikely and the most common case is when obstacles between the transmitter and receiver modify the propagation of the radio signal.

Chapter 3

Embedded Networking

The comparison of the centralized and the distributed network has been a topic of debate for many years. This work attempts to implement both networks, in order to solve the aforementioned dilemma.

Nowadays, every parameter of a network is taken into account. A wireless sensor network must be low-cost, low-power, low-complexity, user friendly, and at the same time it has to be effective and solve the challenges of the current period.

This work is occupied with the design and implementation of both types of networks in order to form a message passing network. The advantages and drawbacks of both methods are presented, taking into consideration the parameters above.

3.1 Distributed message passing platform

Recent advances, show the importance of a distributed WSN. A distributed network is capable for in network inference, where the whole system can decide itself and solve a problem, without using a central device (such as a personal computer, or cloud).

Distributed processing in WSNs can help reduce energy consumption, make efficient use of network bandwidth, and improve system response time. That leads to less complexity and cost reduction.

In this chapter, this work attempts to design and implement a multi-node message passing distributed network , capable for in-network inference, using Thunderboard's embedded environmental sensors.

3.1.1 Implementation

In a large distributed system, no node can have a global view of the entire system at any time. Instead, every node has a local view of the system only, and has to base its decisions on this local information. Many tasks can be solved completely locally, for instance, a node can figure out the lowest measured humidity in its neighbourhood by simply communicating with its neighbours (Fig. 3.1).

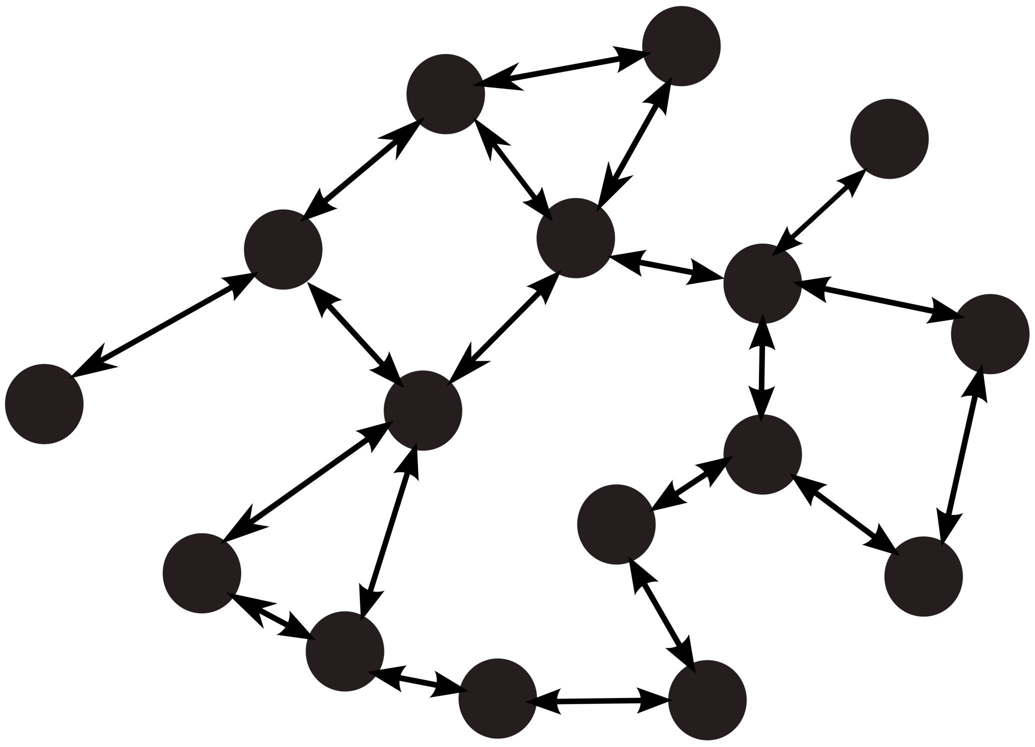


Figure 3.1: An example distributed network of 16 nodes, where each agent communicates with a set of nodes in its neighborhood

A vital characteristic of a distributed WSN is the ability of each node to find its neighbor nodes in the network. In a distributed network, a node only communicates with its neighbors, so being able to detect the presence of other nodes is vital. The neighborhood of a terminal in a WSN is defined as the area of its communication range.

The distributed network to be analyzed, is an extension of the point-to-point communication presented in detail in Ch. 2.2. As a consequence, the network operates in frequency domain, and the radio configuration is also the same (Table 2.1).

Firstly, the number of nodes in the network is defined. Each node has a unique node id which is its identifier. The node id is the same number as the channel a node listens on. For example $node\_id = 5$ also listens on channel 5.

Beginning from $node\_id = 1$, the sending node sends sequentially the same message for neighbor discovery, to all the other nodes in network. If a node receives that message , which means it does not sleep (further information in Chap. 3.1.3) and is located within the communication range of the sending node, then it sends back a message to inform that is its neighbor (acknowledgment). The obvious drawback of this procedure is that it takes more time than a time domain implementation.

After sending to all nodes in network, the sending node knows exactly which nodes are located inside its neighborhood. Then, it sends the information message (for example the humidity measurement) to these nodes sequentially. Then the procedure continues with $node\_id = 2$ etc..

3.1.2 Packet Layer Configuration

The message length was adjusted to problem's requirements. A 13-byte payload message is sent each time, using also 4 bytes synchronization word to maximize packet success.

Fixed payload length must be used during the whole procedure. There are three types of messages in the network. The one that the sending node sends to the other nodes, in order to find the ones in its neighborhood, the one that the nodes send in response to this message and the information message that the node send to its neighbors.

During neighbor discovery, the messages only contain the message identifier and the sending node's id. The rest of the message is covered with zeros.

The payload of the information message is presented in Fig 3.2. Using the

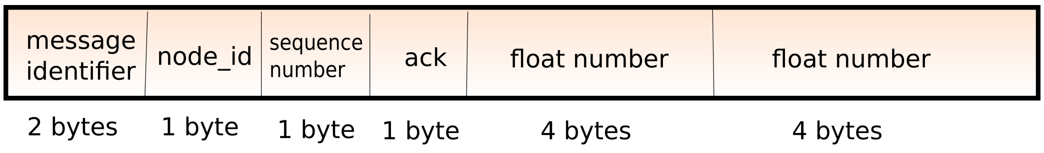


Figure 3.2: Message payload

aforementioned payload length, the network is capable of sending two float numbers which can be the mean or the variance values of a probabilistic model, or the measurements from the embedded sensors.

This work ensures that each node in the WSN is also capable of detecting which node is the closest neighbor. So, the implementation can be easily modified to a network in which each terminal communicates *only* with the closest neighbor.

During neighbor discovery, after each response from the neighbor nodes, the sending node (using the function GetRssi() provided by RAIL SDK), estimates the Received Signal Strength Indicator (RSSI). These values are stored into a buffer and at the end of the procedure, the sending node sorts this buffer and can easily detects the closest neighbor. The precision of the rssi values is pretty high, as the node is capable of detecting which node is closer even when the physical distance between them is less than 50cm. However, this feature does not meet network's requirements so it is not included in network's capabilities.

Very simple mini software examples for the initialization and the measurement of each sensor are provided by Silicon Labs. Thus, it is very easy for the developer to make use of the embedded (in the kit) sensors. A challenge this work faced, is the measurement of multiple sensors simultaneously. Thunderboard Sense 2 provides 8 different sensors but only some of them share the same driver. It is noticed though, that sensors from different driver cannot be enabled simultaneously.

3.1.3 Sleep Mode

After the implementation of the message passing network, the work attempts to achieve an ambiently powered WSN (more in Sec. 3.3). A necessary prerequisite for a ambiently powered node is the sleep mode. During sleep mode, the power consumption of the kits is significantly low, so that helps the system charge the energy harvester storage unit.

Proprietary SDK provides RAIL libraries which help the developer to program Thunderboards to sleep for a specific time. Four energy modes are provided from Silicon Labs [8]. The requirements of the network implemented (for example the system must keep the sequence number), lead us to select the energy mode 2.

In Energy Mode 2 (EM2), the EFM32 microcontrollers offer a high degree of autonomous operation, while keeping energy consumption low. The high frequency oscillator is turned off in EM2; however, a 32 kHz oscillator and the real time clock are available for the low energy peripherals.The MCU performs advanced operations in sleep mode. The peripherals run autonomously due to intelligent interconnection of the modules and memory, the wake-up time to EM0 is only $2\mu s$ and low-leakage RAM ensures full data retention.

Silicon Labs provides enough documentation to guide the developer implement the sleep mode. Each project generated in Simplicity Studio includes a quiet friendly interface for enabling/disabling/changing the software components such as sleep libraries. Also, a sleep timer is provided which is the only process that works during sleep mode. It is enabled when the kit enters sleep mode, and when the timer expires, an interrupt is occurred, and the kit wakes up.

The purpose of sleep mode is energy saving, so the energy consumption must be measured to make sure it is significantly low. The consumption for a supply voltage of 3V was measured at 8μA, which is a very satisfactory value.

3.1.4 Test setup

After taking into consideration all the necessary parameters for a reliable network, a 16 node wireless sensor network was implemented and tested outdoor. The experiment took place in a wide area full of trees inside TUC's campus, where line-of-sight path between nodes did not always exist.

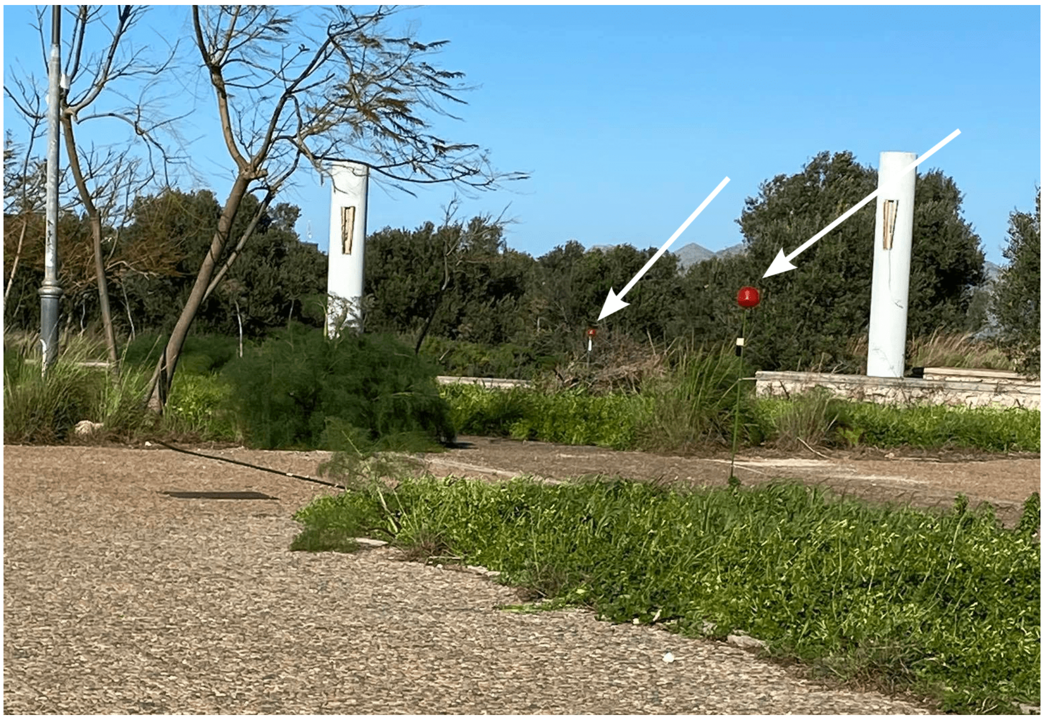


Figure 3.3: Experimental setup.

It was observed that the coin cell batteries did not cover kits' needs for power. More specifically, the 3V lithium 2032 coin cells, could not provide the necessary power during the current spikes occurred during transmission. Thus, power banks were used instead, to power the kits.

Despite the fact that the communication range achieved was 280m, the nodes were placed about 50 meters apart so each node has plenty of neighbors (Fig. 3.3). In comparison to a network where each terminal communicates only with one node, more interaction between the nodes (when each node communicates with more neighbors), leads to faster convergence.

According to Google Maps, the area covered (shown at Fig. 3.4) is 300m x160m. This is a total of $48000m^2$.

The time needed for all the 16 nodes to find their neighbors and send their information (one iteration) is 48 seconds including sleep mode. The procedure is repeated periodically until the network reaches convergence. Given the pretty low bit rate (2.4 kbps) and the large payload length this is

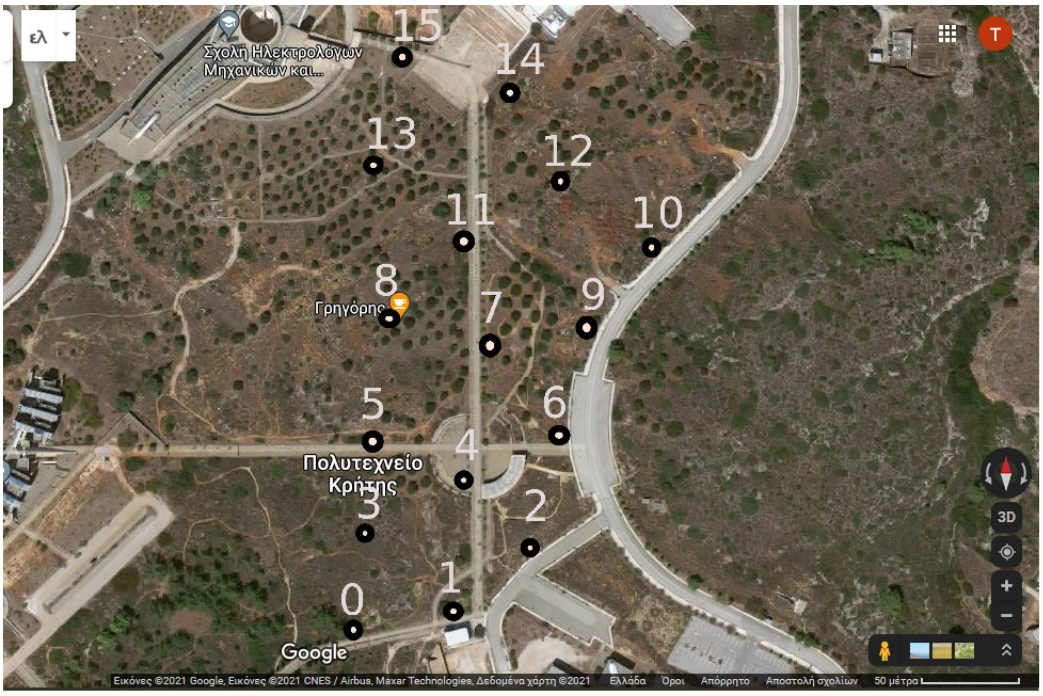


Figure 3.4: Experimental Topology.

a significantly fast procedure.

A major challenge this work faced, was to find the exact time slot during the period of time (T) where the nodes must sleep, without losing crucial messages that may effect the functionality of the network. The decision was, to put each node to sleep for a duration of 10 seconds, 20 seconds after sending its information message.

A duty cycle or power cycle is the fraction of one period in which a signal or system is active. In this work, $\frac{10\text{secs}}{48\text{secs}} * 100\% = 20.3\%$ of the time, the nodes sleep. So duty cycle of the implementation is 79.7%. That's a high duty cycle but the value can be modified depending on the network's needs.

The packet error rate observed given the above setup was less than 2%. In conclusion, this work offers a proof of concept of a distributed WSN using low-power, low-cost kits and exploiting the embedded environmental sensors. This message passing network can be used for in network inference, either the area monitoring using the embedded environmental sensors, or to convert the inference task to a probabilistic, message passing problem for in-network inference. Two applications of this WSN for in network inference will be analytically presented in Ch. 5.

3.2 Centralized Zigbee Network

3.2.1 What does Centralized Network mean?

Centralized formation techniques are suitable for networks in which the processing power capacity relies mostly on a unique device (personal computer, cloud etc). In such cases, this device is responsible for the processing, coordination, and management of the sensed information activities. It also forwards this data to a sink node. A centralized network is scalable and hence can accommodate any new nodes or devices at any time; it is also flexible and hence open to physical partitions.

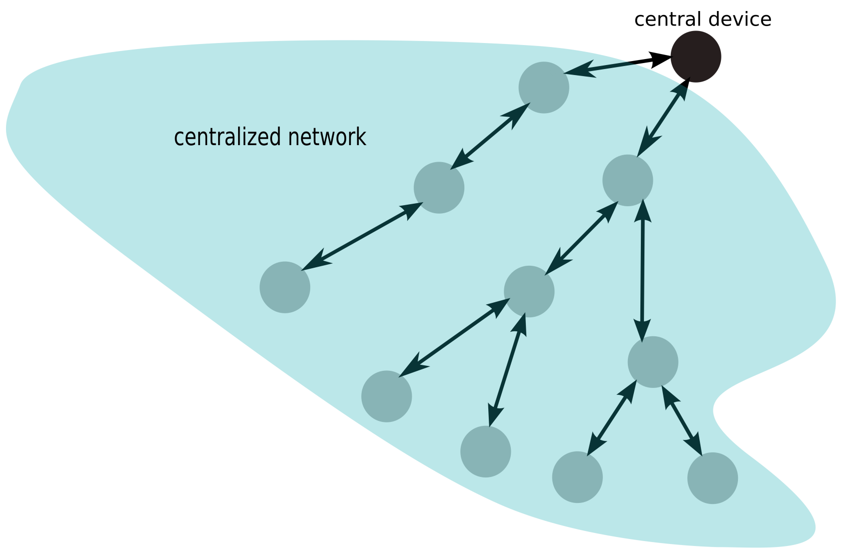


Figure 3.5: Centralized Network.

This work uses Zigbee 3.0 for the implementation of the centralized network. Silicon Labs EmberZNet PRO Zigbee networking protocol stack is a Zigbee protocol software package containing all the elements required for mesh networking applications on Silicon Labs' Ember platforms. However, the provided SDK is not open source. The serial number of a Silicon Lab's kit is needed to access it. That means that the support is quiet deficient, and the sources are limited to the documentation files and the Zigbee forum.

A key component of the Zigbee protocol is the ability to support mesh networking. In a mesh network, nodes are interconnected with other nodes so that multiple pathways connect each node. Connections between nodes are dynamically updated and optimized through built-in mesh routing table.

Mesh networks are decentralized in nature; each node is capable of self-discovery on the network. Also, as nodes leave the network, the mesh topology allows the nodes to reconfigure routing paths based on the new network structure. The characteristics of mesh topology and ad-hoc routing provide greater stability in changing conditions or failure at single nodes.

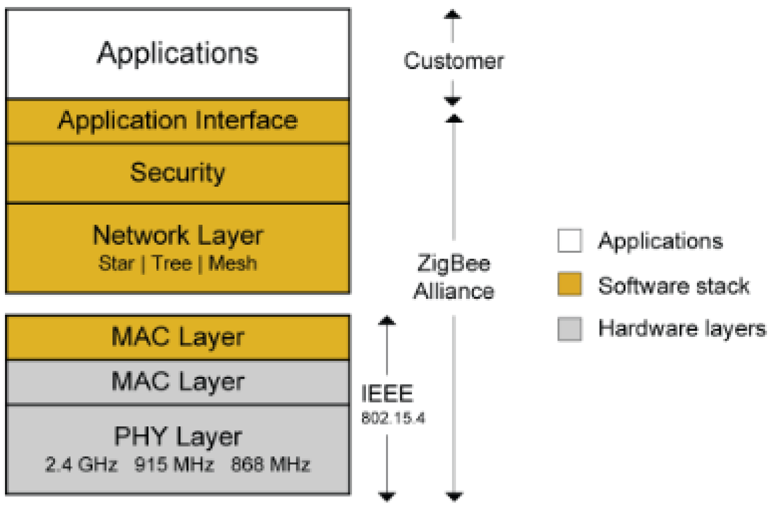


Figure 3.6: Zigbee Architecture [3].

Zigbee provides a very important part of the centralized network to the developer as a black box (Fig. 3.6). Zigbee networks [3] are based on the IEEE 802.15.4 MAC and physical layer. The 802.15.4 standard operates at 250 kbps in the 2.4 GHz band. The 802.15.4 MAC layer is used for basic message handling and congestion control. This MAC layer includes mechanisms for forming and joining a network, a carrier sense multiple access (CSMA) mechanism for devices to listen for a clear channel, as well as a link layer to handle retries and acknowledgment of messages for reliable communications between adjacent devices. The Zigbee network layer builds on these underlying mechanisms to provide reliable end-to-end communications in the network.

3.2.2 Implementation

Zigbee's centralized network is composed of three roles. The coordinator (C) of the network, the routers (R) and the end device (ED).

The centralized network provided by Zigbee is a multi - hop network. The topology looks like a tree (Fig. 3.5). The coordinator is the master of the network and communicates with the external device (such as personal computer or cloud). Coordinator does not have a parent node, and only has children. The leaves of the tree are the end devices which just send their messages (sensor value) to the closer node. These nodes do not have children. The nodes between coordinator and end device(s) are the routers of the system. These nodes have two roles: to forward the messages received from their children (range extender) and to send their own message with the sensor value to their parent (router or coordinator).

The radio configuration is standard by Zigbee, and the communication bit rate is set to 250kbps. It also operates at 2.4GHz. The radio settings cannot be changed by the user.

The implementation is the following: Each node attempts to send a message to coordinator, using the hops, unless node and coordinator are within the communication range. In this case, the node will send a direct message to coordinator, without using the routers. If a node is not inside the range of the coordinator, it tries to find the fastest path, using the neighbors routers as hops, so the message finally reaches coordinator with multi-hop procedure.

To form a centralized network Zigbee requires the following serial commands:

A) plugin network-creator form 1 (for centralized), panId, radioTxPower, channel which searches for an unused Channel and Pan Id (personal area network).

Each network is defined by a unique PAN identifier (PAN ID), which is common among all devices of the same network. The below command (only used by the coordinator) automatically forms a network on the best unused Channel and Pan Id it finds. Furthermore, the transmission power is set using this command.

After the network is formed, the coordinator of the network can open access to the WSN nodes. The time window for a node to join the network is restricted to 256 seconds due to safety reasons. After that period of time, the user can execute the command again. The command is the following:

B) plugin network-creator-security open-network,

When the network is formed and there is an open access for the rest of the WSN nodes, the routers and the end devices can join the network by executing the below command.

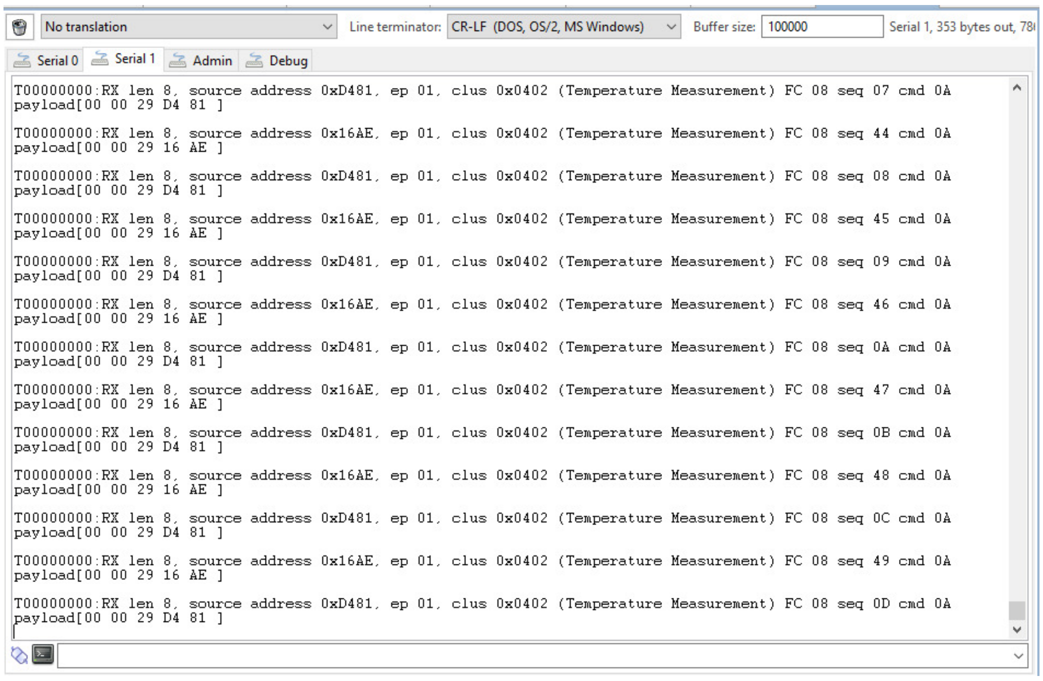
C) plugin network-steering start 1,

Using the above simple serial commands in Simplicity Studio's terminal, a basic centralized network is formed. At that point, the network has no functionality.

Given the very high data rate that the network operates, very high transmission power is needed in order to achieve the largest communication range possible. 10.8 dBm transmission power is selected , same as the distributed network's, so that the comparison is fair.

At first, a 3 node setup was implemented using one coordinator, one router and one end device.

The end devices and the routers in the network, were set to send every 5 seconds, periodically, a message to coordinator. The payload of the message contains the cluster ID, and the node id of the sending node, for recognition. The messages received by the coordinator are shown in 3.7.



No translation   Line terminator: CR-LF (DOS, OS/2, MS Windows)   Buffer size: 100000   Serial 1, 353 bytes out, 78

```
T00000000:RX len 8, source address 0xD481, ep 01, clus 0x0402 (Temperature Measurement) FC 08 seq 07 cmd 0A
payload[00 00 29 D4 81 ]
T00000000:RX len 8, source address 0x16AE, ep 01, clus 0x0402 (Temperature Measurement) FC 08 seq 44 cmd 0A
payload[00 00 29 16 AE ]
T00000000:RX len 8, source address 0xD481, ep 01, clus 0x0402 (Temperature Measurement) FC 08 seq 08 cmd 0A
payload[00 00 29 D4 81 ]
T00000000:RX len 8, source address 0x16AE, ep 01, clus 0x0402 (Temperature Measurement) FC 08 seq 45 cmd 0A
payload[00 00 29 16 AE ]
T00000000:RX len 8, source address 0xD481, ep 01, clus 0x0402 (Temperature Measurement) FC 08 seq 09 cmd 0A
payload[00 00 29 D4 81 ]
T00000000:RX len 8, source address 0x16AE, ep 01, clus 0x0402 (Temperature Measurement) FC 08 seq 46 cmd 0A
payload[00 00 29 16 AE ]
T00000000:RX len 8, source address 0xD481, ep 01, clus 0x0402 (Temperature Measurement) FC 08 seq 0A cmd 0A
payload[00 00 29 D4 81 ]
T00000000:RX len 8, source address 0x16AE, ep 01, clus 0x0402 (Temperature Measurement) FC 08 seq 47 cmd 0A
payload[00 00 29 16 AE ]
T00000000:RX len 8, source address 0xD481, ep 01, clus 0x0402 (Temperature Measurement) FC 08 seq 0B cmd 0A
payload[00 00 29 D4 81 ]
T00000000:RX len 8, source address 0x16AE, ep 01, clus 0x0402 (Temperature Measurement) FC 08 seq 48 cmd 0A
payload[00 00 29 16 AE ]
T00000000:RX len 8, source address 0xD481, ep 01, clus 0x0402 (Temperature Measurement) FC 08 seq 0C cmd 0A
payload[00 00 29 D4 81 ]
T00000000:RX len 8, source address 0x16AE, ep 01, clus 0x0402 (Temperature Measurement) FC 08 seq 49 cmd 0A
payload[00 00 29 16 AE ]
T00000000:RX len 8, source address 0xD481, ep 01, clus 0x0402 (Temperature Measurement) FC 08 seq 0D cmd 0A
payload[00 00 29 D4 81 ]
```

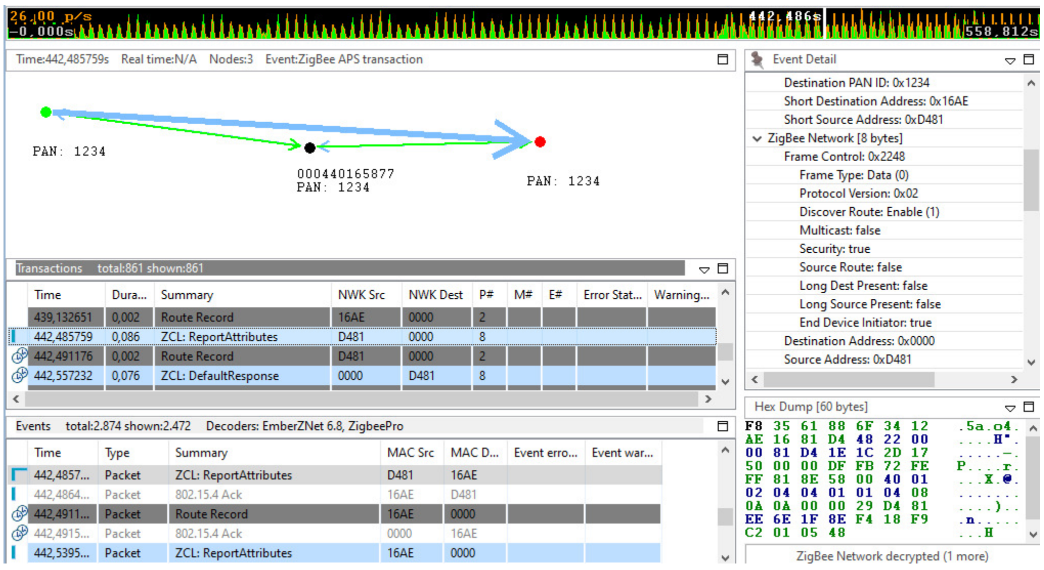
Figure 3.7: Coordinator's console.

The range of the communication link is a crucial factor for a centralized network.

Zigbee's standard radio configuration leaves no space for developer's changes. Using the maximum transmission power provided, the range achieved outdoors between two devices was 30m. Thus, it is clear that the centralized network provided by Zigbee is ideal for smart homes and buildings applications. However, it is not suggested for an outdoor WSN which requires wide communication range e.g. for area monitoring, like the distributed network analyzed above.

In Fig. 3.8, where the end device is located outside coordinator's communication range, the router is used as a hop (range extender) to help the message reaches coordinator. That is clear if we observe the network source and destination compared to MAC source and destination in the figure.

Zigbee provides sleep mode only for the end devices. The sleepy end device implemented, wakes up the end device only for the communication. However



Time:442,485759s Real time:N/A Nodes:3 Event:ZigBee APS transaction

Transactions total:861 shown:861

Time	Dura...	Summary	NWK Src	NWK Dest	P#	M#	E#	Error Stat...	Warning...
439,132651	0,002	Route Record	16AE	0000	2				
442,485759	0,086	ZCL: ReportAttributes	D481	0000	8				
442,491176	0,002	Route Record	D481	0000	2				
442,557232	0,076	ZCL: DefaultResponse	0000	D481	8				

Events total:2.874 shown:2.472 Decoders: EmberZNet 6.8, ZigbeePro

Time	Type	Summary	MAC Src	MAC D...	Event erro...	Event war...
442,4857...	Packet	ZCL: ReportAttributes	D481	16AE		
442,4864...	Packet	802.15.4 Ack	16AE	D481		
442,4911...	Packet	Route Record	16AE	0000		
442,4915...	Packet	802.15.4 Ack	0000	16AE		
442,5395...	Packet	ZCL: ReportAttributes	16AE	0000		

Event Detail

```
    Destination PAN ID: 0x1234
    Short Destination Address: 0x16AE
    Short Source Address: 0xD481
  ZigBee Network [8 bytes]
    Frame Control: 0x2248
      Frame Type: Data (0)
      Protocol Version: 0x02
      Discover Route: Enable (1)
      Multicast: false
      Security: true
      Source Route: false
      Long Dest Present: false
      Long Source Present: false
      End Device Initiator: true
    Destination Address: 0x0000
    Source Address: 0xD481
```

Hex Dump [60 bytes]

```
F8 35 61 88 6F 34 12    .5a.o4.
AE 16 81 D4 48 22 00    ....H".
00 81 D4 1E 1C 2D 17    ......-.
50 00 00 DF FB 72 FE    P....r.
FF 81 8E 58 00 40 01    ...X.@.
02 04 04 01 01 04 08    .......
0A 0A 00 00 29 D4 81    ....)..
EE 6E 1F 8E F4 18 F9    .n.....
C2 01 05 48             ...H
```

ZigBee Network decrypted (1 more)

Figure 3.8: The multi hop network.

the majority of the time, the end device sleeps, for energy saving. The routers or the coordinator cannot sleep in a Zigbee network. If a router exits the network for any reason, the communication may corrupt, because all the end devices relied on that router, will not be able to communicate with the rest of the network and thus with the coordinator; a crucial part will leave the network.

That leads to the conclusion that another very important mode, which is crucial for an ambiently powered WSN as described above, cannot be implemented using Zigbee's protocol. That means, Zigbee's centralized network is useful for other types of applications such as home monitoring, although it is not a perfect solution for a wide wireless sensor network, where the nodes must be ambiently powered and their batteries must last for long periods.

3.3 Energy Harvester

Wireless sensor networks (WSNs) research has pre-dominantly assumed the use of a portable and limited energy source, like batteries, to power sensors [9]. Without energy, a sensor is essentially useless and cannot contribute to the utility of the network as a whole. Consequently, there are emerging WSN applications where sensors are required to operate for much longer durations (like years or even decades) after they are deployed. Examples include environmental monitoring where batteries are hard or even impossible to replace. Lately, an alternative to powering WSNs is being actively studied, which is to convert the ambient energy from the environment into electricity to power the sensor nodes. Sensor nodes need to exploit the sporadic availability of energy to quickly sense and transmit the data.

Although that energy harvested is small and in the order of μWatt, it can provide enough power for wireless sensors and other low-power applications [10]. In the environment there is a lot of pure energy (especially solar energy) provided for free, and can be converted into electricity to power the various circuits and represents a potentially low-cost source of power. This technology applied in a wireless sensor network (WSN) and devices on the IoT, will eliminate the need for network-based energy and conventional batteries, will minimize maintenance costs, eliminate cables and batteries and is ecological. Energy harvesting will promote environmentally friendly technologies that will save energy, will reduce CO_2 emissions, which makes this technology indispensable for achieving next-generation smart cities and sustainable society.

This work presents an energy harvester customized for Thunderboard Sense 2, which exploits the solar energy.

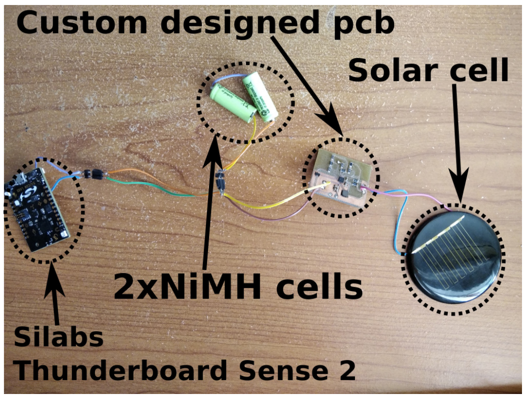


Figure 3.9: A terminal of the WSN testbed; it consists of a custom solar energy harvester and a Thunderboard Sense 2 module.

3.3.1 Implementation [1]

Each WSN node/terminal consists of a Silabs Thunderboard Sense 2 embedded module and a custom solar energy harvester [1], using rechargeable NiMH cells, as storage elements (Fig. 3.9).

The initial vision was to implement 16 energy harvester circuits to power all the nodes. Due to time constraints, 3 energy harvesters were implemented. So, the following analysis refers to a 3 node network. Average Consensus Algorithm implementation (Sec. 5.2) was powered by this energy harvester.

In order to design the energy harvester, the WSN terminals (Fig. 3.9) were first characterised with respect to their power consumption. For a supply voltage of 3 V, the current consumption was measured under four different modes of operation. It is noted that during the current measurements, the humidity sensor was enabled. In both idle and receive modes, each terminal consumed 13 mA, while in sleep mode $I_{\mathrm{sl}} = 8$ μA. The transmit mode was the most power demanding and presented current spikes in the order of 50 mA (and thus the average current in active mode is $I_{\mathrm{A}} = 14.54$ mA). EM3 and EM4 provide lower energy consumption during the sleep mode, although the order of μA consumption that EM2 provides, makes it the perfect choice and at the same time meets the requirements the network sets.

A single iteration of the algorithm (in Sec. 5.2) requires 24 seconds of operation; each WSN terminal spends 14 seconds in its active operation mode and 10 seconds in sleep/ultra-low-power mode, offering a duty cycle of $D = 58.3\%$ and an average current consumption of 8.48 mA.

The harvester design consists of a 26.42 cm^2 solar-cell connected to a custom circuit designed using the TI BQ25504 IC. Two 1.2 V, $C_s = 100$ mAh NiMH cells connected in series, were utilized as the storage element at the output of the harvesting circuit. This design features an efficiency of 91% and can store energy up to 864 J (240 mWh). A detailed schematic of the system is shown in Fig. 3.11.

The probability for an outage event was defined as the percentage of time that a WSN terminal fails to operate, due to insufficient energy. The aforementioned percentage was calculated based on the availability of solar power during a day.

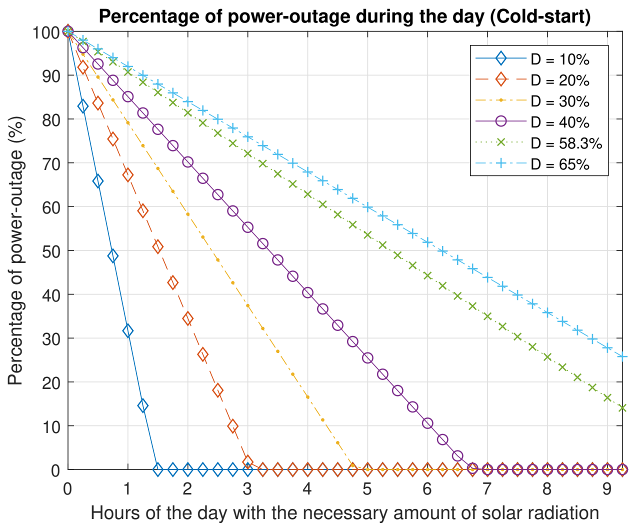


Figure 3.10: Probability of power outage as a function of number of hours during which the solar irradiance constraint is satisfied [1].

During sleep mode, the current drawn by each terminal ($I_{sl} = 8\ \mu A$) can be considered negligible (compared to the charging current provided to the storage cell pack from the harvester). Thus, it can be assumed that for a time portion of D, the terminal operates while drawing current (mainly) from the cells; for a time portion $1 - D$ the current provided by the harvester is (mainly) used for charging the cells.

The pair of NiMH cells was depleted of charge (combined voltage 1.37 V) and the time required for reaching a state of full charge was measured at 4 hours (for $D = 0$, i.e., no load connected). Under the aforementioned assumptions it can be directly shown that the time required for fully charging the storage cells, when the load operates with a duty cycle D, is given by $4/(1 - D)$ hours.

In order for the harvester to both charge the cells and provide power to the terminal, the available solar radiation must be greater than 368 W/m^2. In a different case, the terminal will be solely powered by the storage cells. It is noted that the batteries must be at a state of at least 5% of their full capacity, in order for the aforementioned analysis to be valid.

For a typical summer day in Chania Crete, it was concluded that 11 hours of exposure, would satisfy the constraints described above and allow the correct operation of the node. The probability of outage as a function of the hours with solar irradiation greater than 368 W/m^2 is presented in Fig. 3.10. For a duty cycle ratio of $D = 58.3\%$, the probability of outage within a day (24 hours) is about 5%. The probability of outage can be decreased if smaller duty cycle ratios are adopted, at expense of increased overall processing delays (more sleep time).

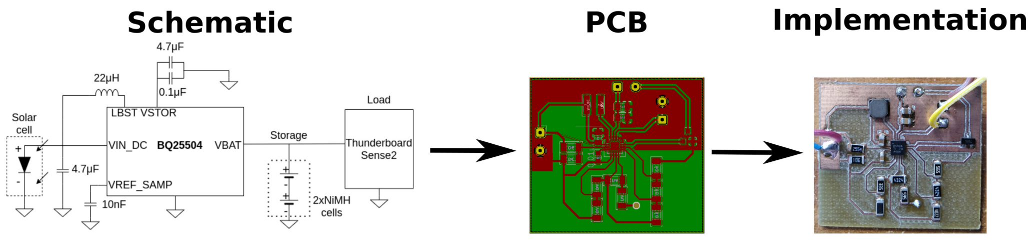


Figure 3.11: The solar harvester designed and implemented for powering the WSN terminals [1].

The aforementioned design can be easily used in the network presented in Ch. 3.1 changing only the average current in active mode due to the larger network.

Chapter 4

WSN as Inference Platform

4.1 Description

Let real vectors $\mathbf{x}^{(0)}, \mathbf{b} \in \mathbb{R}^K$ and real square matrix $\mathbf{A} \in \mathbb{R}^{K \times K}$. A large class of inference algorithms, including Gaussian belief propagation (GBP) and consensus, as well as matrix inversion (Jacobi and Gauss-Seidel) are formulated according to the following recursion:

$$\mathbf{x}^{(l)} = \mathbf{A}\mathbf{x}^{(l-1)} + \mathbf{b}, \tag{4.1}$$

with $l = 1, 2 \ldots$. The solution of this problem, i.e., the *fixed point*, is denoted by $\mathbf{x}^* \triangleq \lim_{l \to \infty} \mathbf{x}^{(l-1)} = \lim_{l \to \infty} \mathbf{x}^{(l)}$, provided that such fixed point exists. The k-th elements of $\mathbf{x}^{(l)}$ and $\mathbf{b}^{(l)}$ are denoted by $\mathbf{x}_k^{(l)} \equiv x_k^{(l)}$ and $\mathbf{b}_k^{(l)} \equiv b_k^{(l)}$ respectively, while $a_{ij} = \mathbf{A}_{ij}$ denotes the element of $\mathbf{A}$ at the i-th row and j-th column. From Eq. (4.1),

$$x_k^{(l)} = \sum_{j=1}^{K} a_{kj}\, x_j^{(l-1)} + b_k,\ \ k \in \{1, 2, \ldots K\}, \tag{4.2}$$

which requires all variables $\{x_j^{(l-1)}\}$ (with respective $a_{kj} \neq 0$) from previous iteration $(l-1)$ to be readily available at iteration (l), in order to update the value of x_k. This constraint results to *synchronous* operation, since no output update is possible for *any* element of the vector $\mathbf{x}$ at a specific iteration, before all necessary input is available for *all* variables.

As far as *asynchronous* operation is concerned, i.e., possibility for a variable $x_k^{(l)}$, to update its value or keep the same value for the following iteration, without waiting for *all* variables $k \in \{1, 2, \ldots, K\}$ to collect input information before update, the following random variable and notation is adopted,

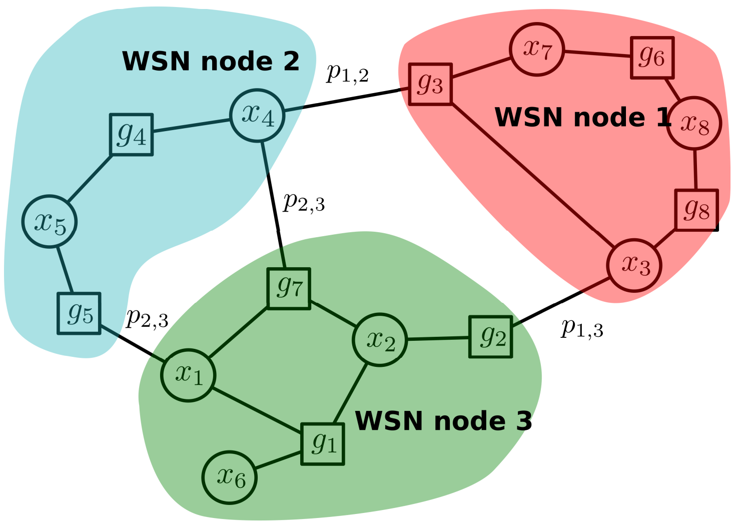


Figure 4.1: Factor graph assignment into a WSN with 3 nodes.

from seminal work in [11]:

$$\psi_k^{(l)} = \begin{cases} 1, & \text{if } x_k \text{ is updated at iteration } l, \\ 0, & \text{otherwise} \end{cases}. \tag{4.3}$$

Let $\boldsymbol{\psi}^{(l)} \in \mathbb{B}^K$ the binary vector obtained if all $\{\psi_k^{(l)}\}, k \in \{1, 2, \ldots K\}$ are stacked and $\boldsymbol{\Psi}^{(l)} = \{\boldsymbol{\psi}^{(l)}\}$ the respective diagonal matrix, with $\boldsymbol{\psi}^{(l)}$ at its diagonal. In that case, the update of Eq. (4.1) becomes as follows:

$$\begin{aligned} \mathbf{x}^{(l)} &= \boldsymbol{\Psi}^{(l)} \left(\mathbf{A}\mathbf{x}^{(l-1)} + \mathbf{b}\right) + \left(\mathbf{I} - \boldsymbol{\Psi}^{(l)}\right) \mathbf{x}^{(l-1)} \\ &= \left(\boldsymbol{\Psi}^{(l)}\mathbf{A} + \mathbf{I} - \boldsymbol{\Psi}^{(l)}\right) \mathbf{x}^{(l-1)} + \boldsymbol{\Psi}^{(l)}\mathbf{b}. \end{aligned} \tag{4.4}$$

Notice that if $\psi_k^{(l)} = 1, \forall k$ (or equivalently $\boldsymbol{\Psi}^{(l)} \equiv \mathbf{I}_K$), the above equation is simplified to the synchronous case of Eq. (4.1). In sharp contrast to [11], where $\{\psi_k^{(l)}\}$ are assumed independent and identically distributed (i.i.d.) across k and l, this work assumes a more general model and allows these variables to be correlated and non-identically distributed. In other words, in this work $\mathbb{E}\left[\boldsymbol{\Psi}^{(l)}\right] = \mathbf{P} \neq p\,\mathbf{I}_K$, in the general case.

Fig. 4.1, depicts a factor graph, allocated to 3 WSN terminals [12]. The factor graph encodes a probability density function (pdf), involving 8 random variables (denoted by circles) and 8 factor nodes (denoted by rectangles) and has the expressive power to describe a variety of message passing (inference) algorithms.

For example, GBP-based message passing in this factor graph results to a exchange of pairs of real numbers from factor to variable nodes $\{\mu_{g_i \to j}, v_{g_i \to j}\}$. There are 19 factor-to-variable links in the depicted factor graph and GBP-based message passing results to $K = 19$ variables in Eqs.(4.1), (4.4), where $\mathbf{x}$ stacks the 19 (real) values of $\{\mu_{g_i \to j}\}$ and $\mathbf{A}$ depends on the connectivity of the factor graph, as well as the fixed point values of $\{v_{g_i \to j}\}$ (and thus, matrix $\mathbf{A}$ is constant).

In Fig. 4.1, there are messages $\mu_{g_5 \to 1}$, $\mu_{g_7 \to 4}$, $\mu_{g_3 \to 4}$, $\mu_{g_2 \to 3}$ (which assume real scalar values); these messages correspond to scheduling variables $\{\psi\}$, with expected value $p_{2,3}$, $p_{2,3}$, $p_{1,2}$, $p_{1,3}$, respectively. Thus, for the depicted mapping of the graph/inference algorithm to the actual WSN network in Fig. 4.1, diagonal matrix $\mathbf{\Psi}^{(l)}$ consists of ones in the main diagonal, apart from the indices corresponding to the aforementioned $\{\psi\}$, with expected values the probabilities given above. In other words, $\mathbb{E}\left[\mathbf{\Psi}^{(l)}\right] = \mathbf{P}$ of Fig. 4.1 is a diagonal matrix, with all ones in the main diagonal, apart from two places which assume the value of $p_{2,3}$, one place that assumes the value of $p_{1,2}$ and another with $p_{1,3}$. It is noted that scheduling variables corresponding to $\mu_{g_5 \to 1}$, $\mu_{g_7 \to 4}$ are fully correlated (with the same expected value) [11].

If p_1 is the probability of terminal 1 being in energy outage, the probability of successful communication between 1 and 2 is given by $p_{1,2} = (1-p_1)(1-p_2)$. Similar methodology holds for $p_{2,3}$, $p_{1,3}$ and $p_{1,4}$. In Sec. 3.3, a method to calculate energy outage probability p_i per terminal i was given, assuming solar (ambient) energy harvesting circuitry.

For completeness, two inference algorithms tested in the numerical results, are briefly presented; these algorithms serve as the distributed inference engine, running at the ambiently-powered WSN.

4.2 Algorithm 1: Gaussian Belief Propagation

Consider the joint Gaussian pdf

$$p(\mathbf{x}) \propto \exp\left\{-\frac{1}{2}\mathbf{x}^T\mathbf{J}\mathbf{x} + \mathbf{h}^T\mathbf{x}\right\}, \tag{4.5}$$

where $\mathbf{J} \succ \mathbf{0}$ and $\mathbf{h}$ the information matrix and potential vector, respectively. This pdf can be written as

$$p(\mathbf{x}) \propto \prod_{i=1}^{n} f_i(x_i) \prod_{j=1}^{m} g_j(\mathcal{X}_j). \tag{4.6}$$

where $\mathcal{X}_j \subseteq \{x_1, x_2, \ldots, x_n\}$. If at least one of $g_j(\mathcal{X}_j)$ contains more than two variables, the above is a **high-order factorization** of the joint pdf. In [11] a general factorization of the aforementioned matrix and vector is considered, where $\mathbf{J} = \mathbf{\Lambda} + \mathbf{\Xi}^T\mathbf{\Sigma}\mathbf{\Xi}$ and $\mathbf{h} = \mathbf{\Lambda}\xi + \mathbf{\Xi}^T\mathbf{\Sigma}\mathbf{u}$. It is assumed that $\mathbf{\Lambda} \triangleq (\eta_1, \eta_2, \ldots, \eta_n)$, $\mathbf{\Sigma} \triangleq (\zeta_1, \zeta_2, \ldots, \zeta_m)$, where $\eta_i \geq 0\ \forall i$ and $\zeta_j > 0\ \forall j$

So, it is

$$\begin{aligned} p(\mathbf{x}) &\propto \exp\left\{-\frac{1}{2}\mathbf{x}^T(\mathbf{\Lambda} + \mathbf{\Xi^T\Sigma\Xi})\mathbf{x} + (\mathbf{\Lambda}\xi + \mathbf{\Xi^T\Sigma u})^T\mathbf{x}\right\} \\ &\propto \exp\left\{-\frac{1}{2}(\mathbf{x}-\xi)^T + \mathbf{\Lambda}(\mathbf{x}-\xi)\right\} \times \exp\left\{-\frac{1}{2}(\mathbf{\Xi x} - \mathbf{u})^T\Sigma(\mathbf{\Xi x} - \mathbf{u})\right\} \\ &\propto \prod_{i=1}^{n} \exp\left\{-\frac{1}{2}(\eta_i - x_i\xi i)^2\right\} \times \prod_{i=1}^{n} \exp\left\{-\frac{1}{2}(\zeta_j\mathbf{\Xi_j x} - u_j)^2\right\}. \end{aligned} \tag{4.7}$$

Using Eqs. (4.7), (4.6), it is concluded that

$$f_i(x_i) \propto \exp\left\{-\frac{1}{2}\eta_i(x_i\Xi_i)^2\right\}. \tag{4.8}$$

$$g_j(\mathcal{X}_j) \propto \exp\left\{ -\frac{1}{2}\zeta_j(\sum_{k\epsilon\mathcal{V}_j} \Xi_{jk}x_k - u_j)^2 \right\}. \tag{4.9}$$

As shown in [11], the message means of Gaussian Belief Propagation under high-order factorization and asynchronous scheduling with i.i.d. scheduling variables are updated as follows:

$$\mu^{(l)}_{g_j\to x_i} = \begin{cases} \boldsymbol{\Xi}^{-1}_{ji}u_j - \sum_{k\in\mathcal{V}_j\setminus i} \frac{\boldsymbol{\Xi}^{-1}_{ji}\boldsymbol{\Xi}_{jk}(\eta_k\xi_k + \Sigma_{k'\in\mathcal{G}_k\setminus i}v^{(l-1)}_{g_k\to x_k}\mu^{(l-1)}_{g_k\to x_k}}{\eta_k + \Sigma_{k'\in\mathcal{G}_k\setminus i}v^{(l-1)}_{g_k\to x_k}}, \text{ if } \eta_k + \Sigma_{k'\epsilon\mathcal{G}_k\setminus i}v^{(l-1)}_{g_k} > 0, \forall k \in \mathcal{V}_j \setminus i \\ 0, \text{otherwise} \end{cases} \tag{4.10}$$

$$v^{(l)}_{g_j\to x_i} = \frac{\boldsymbol{\Xi}^2_{ji}}{\zeta^{-1}_j + \Sigma_{k\in\mathcal{V}_j\setminus i}\boldsymbol{\Xi}^2_{jk}(\eta_k + \Sigma_{k'\in\mathcal{G}_k\setminus i}v^{(l-1)}_{g_k})^{-1}}. \tag{4.11}$$

Eq. (4.10) can be also expressed as

$$\begin{aligned} \boldsymbol{\mu}^{(l)} &= \boldsymbol{\Psi}^{(l)}\left(\mathbf{A}\boldsymbol{\mu}^{(l-1)} + \mathbf{c}\right) + \left(\mathbf{I} - \boldsymbol{\Psi}^{(l)}\right)\boldsymbol{\mu}^{(l-1)} \\ &= \left(\boldsymbol{\Psi}^{(l)}\mathbf{A} + \mathbf{I} - \boldsymbol{\Psi}^{(l)}\right)\boldsymbol{\mu}^{(l-1)} + \boldsymbol{\Psi}^{(l)}\mathbf{c}. \end{aligned} \tag{4.12}$$

where analytical expression of $\boldsymbol{\mu}$ as well as construction of $\mathbf{A}$ and $\mathbf{c}$ can be found in Eq. (13) and (19) of [11].

Assume that the WSN goal is to solve the following linear equation:

$$\mathbf{Mx} = \mathbf{s}, \tag{4.13}$$

where $\mathbf{M} \in \mathbb{R}^{C\times D}(C \geq D)$ is a full rank matrix and $\mathbf{s} \in \mathbb{R}^C$; the solution of the aforementioned system is offered by:

$$\mathbf{x} = \left(\mathbf{M}^T\mathbf{M}\right)^{-1}\mathbf{M}^T\mathbf{s}. \tag{4.14}$$

Setting $\mathbf{\Lambda} = \mathbf{0}_{n \times n}$, $\mathbf{\Xi} = \mathbf{M}$, $\mathbf{\Sigma} = \mathbf{I}$, $\boldsymbol{\xi} = \mathbf{0}$ and $\mathbf{u} = \mathbf{s}$, Gaussian Belief Propagation can be utilized for the following distribution:

$$p(\mathbf{x}) = \mathcal{N}\left(\mathbf{x}; \left(\mathbf{M}^T\mathbf{M}\right)^{-1}\mathbf{M}^T\mathbf{s}, \left(\mathbf{M}^T\mathbf{M}\right)^{-1}\right), \tag{4.15}$$

whose inference of mean value will yield the desired solution of the problem.

Notice that the same factorization of $\mathbf{J}$ can be exploited in GBP-based inference of minimum mean squared error (MMSE) estimation [13].

4.3 Algorithm 2: Average Consensus

Decentralized consensus algorithms involve a large number of agents that exchange messages in a distributed manner [12]. One of the algorithms is the so called average consensus, where all agents converge to the same value, which is the average of all sensor measurements. In particular, under certain conditions, it can be proved that the following iteration,

$$\mathbf{x}^{(l)} = \mathbf{W}\mathbf{x}^{(l-1)}, \tag{4.16}$$

where $\mathbf{W}$ is a weight matrix that encapsulates both the connectivity and the weights of the network, reaches a fixed point $\mathbf{x}^*$ where $x_i^* = \frac{1}{N}\sum_{j=1}^{N} x_j$, $\forall i = 1, 2, \ldots, N$.

This work considers an asynchronous variant of update Eq. (4.16), given as follows:

$$\mathbf{x}^{(l)} = \left(\mathbf{\Psi}^{(l)}\mathbf{W} + \mathbf{I} - \mathbf{\Psi}^{(l)}\right)\mathbf{x}^{(l-1)}. \tag{4.17}$$

If $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is the graph that stems from the topology of the WSN, where $\mathcal{V}$ its vertices and $\mathcal{E}$ its edges, then the matrices that contain the *max-degree* weights of $\mathcal{G}$, as follows,

$$\mathbf{W}_{ij} = \begin{cases} \dfrac{1}{d+1}, & i \neq j, \{i,j\} \in \mathcal{E}, \\ 1 - \dfrac{d_i}{d+1}, & i = j, \\ 0, & i \neq j, \{i,j\} \notin \mathcal{E} \end{cases} \tag{4.18}$$

where d is the degree of $\mathcal{G}$ and d_i the degree of vertex i, is a family of matrices that guarantee convergence [14]. Other choices of $\mathbf{W}$ are possible.

Chapter 5

Experimental results

5.1 Implementation of Gaussian Belief Propagation (GBP)

The Gaussian Belief Propagation algorithm was implemented in a 3 node WSN, depicted in Fig. 5.1. The network topology refers to the factor graph depicted in Fig. 4.1.

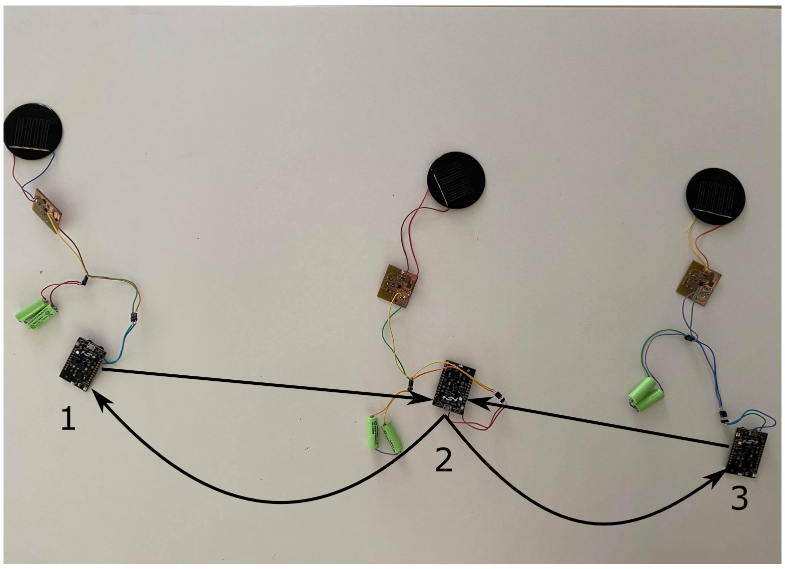


Figure 5.1: Real WSN with 3 terminals/nodes.

Consider an ambiently powered wireless sensor network with N (physical) terminals [12]. Assume that at each iteration, WSN terminal i is responsible for updating a unique subset of the elements $\{x_k^{(l)}\}$ of $\mathbf{x}^{(l-1)}$, denoted by $\mathbf{x}_{\mathcal{I}_i}^{(l)}$, where $\cap_{i=1}^{N}\mathcal{I}_i = \emptyset$ and $\cup_{i=1}^{N}\mathcal{I}_i = \{1, \ldots, K\}$. In other words, each WSN terminal is responsible for updating a subset of the variables, all vari-

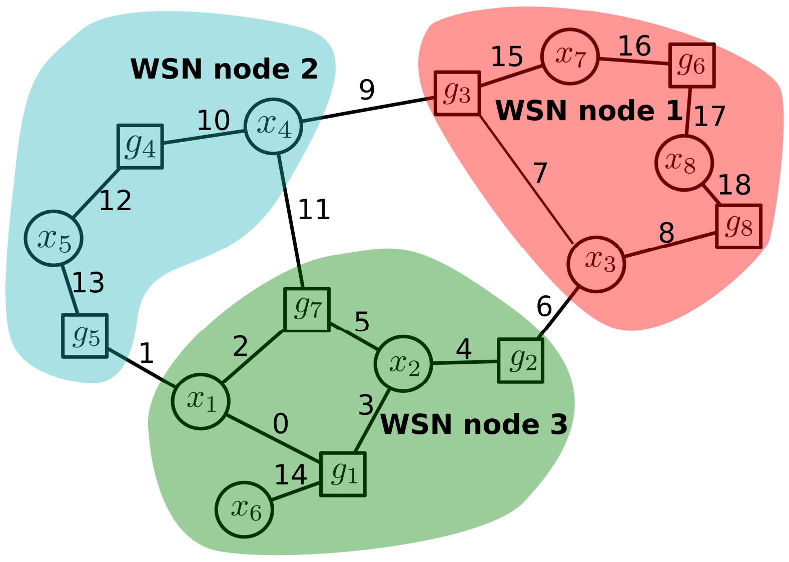


Figure 5.2: Edges representation in factor graph.

ables are allocated to specific WSN terminals and no variable is allocated to more than one WSN terminal. Given that the input for each variable update (according to Eq. (4.2)) might require variables that are allocated to other WSN terminals, communication between the WSNs is required; such message passing may fail, simply because the WSN terminal does not have sufficient energy; thus, the necessary message passing is interrupted probabilistically, with respective probabilities that depend on ambient energy harvesting (and consumption) at each WSN terminal. For simplicity, we assume that messages between WSN terminals fail to be communicated only due to energy outage at either the transmitter or the receiver.

In the actual WSN, the factor graph was divided in three sub factor graphs (Fig.5.2). Each sub factor graph represents a WSN node. Each node computes the necessary factor-to-random variable messages (the mean and the variance), for subsequently calculating the belief of each random variable. However, as shown above Eqs. (4.10), (4.11), the computations implemented by each node, may depend on messages located in different nodes. For example, message 11 depends on message 9. Although, node 3 does not possess information about the value (mean and variance) of that message. So, node 1 must send the mean and the variance of message 9 to node 3.

Therefore, the communication between the nodes in the actual WSN involves the exchange of the values (mean and variance) necessary, for the edges values computations. More specifically, using the aforementioned equations, for the specific vector **M** (Eq. (5.1)), it is concluded that node 1 sends the values (mean and variance) of the edges $7, 8, 9$ to node 3 and the values of the message no.9 to node 2. Node 2 sends the values of the message no.10 to node 1 and the values of the messages no.$1, 10$ to node 3. Node 3 send the values of the messages no.$6, 11$ to node 1 and the values of the messages no.$0, 2, 11$ to node 2.

The procedure is composed of two rounds, one for information transmission and another one for performing the necessary computations. During the information transmission, the number of messages each node sends can vary from node to node, depending on how many edges a node needs. Also, unicast communication takes place, as the nodes send different values to different nodes.

In all experiments, the following values are used [11]:

$$\mathbf{M} = \begin{bmatrix} 3.63 & -6.12 & 0 & 0 & 0 & -2.61 & 0 & 0 \\ 0 & -10.65 & 7.59 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1.92 & 7.05 & 0 & 0 & -10.46 & 0 \\ 0 & 0 & 0 & 0.18 & 3.27 & 0 & 0 & 0 \\ -2.01 & 0 & 0 & 0 & -0.97 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 8.01 & 0.37 \\ 5.18 & -1.86 & 0 & 4.63 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.91 & 0 & 0 & 0 & 0 & 0.13 \end{bmatrix} \tag{5.1}$$

and

$$\mathbf{s} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}^T \tag{5.2}$$

resulting in the factor graph with 8 random variables and 8 factors that can be seen in Fig. 4.1. GBP is executed with two different schedules: 1) a **synchronous** scheduling, where no communication failures occur and 2) a **asynchronous** scheduling, where the messages between terminals are possibly lost with a predefined probability due to communication error.

Fig. 5.3 depicts the expected value $\mathbb{E}[||\mathbf{e}^{(l)}||_2] \triangleq \mathbb{E}[||\epsilon^{(l)} - \epsilon^*||_2]$ as a function of iteration number l, where $\epsilon^{(l)}$ and ϵ^* are the belief means of distribution Eq. (4.15) at iteration l and after reaching the fixed point $\boldsymbol{\mu}^*$, respectively.[1] It is assumed that all communication links for the asynchronous scheduling have probability of successful message exchange $p = 0.4317$, which was set randomly.

In the actual WSN, the outage event is implemented by "flipping a coin". Before each information transmission, each node creates a number from 0 to 1 randomly. If the value is bigger than the probability of the message success (in this case $p = 0.4317$), then we assume there is an outage event in that node, and the node sends the message sent during the previous message passing iteration, without updating the new values. This is the asynchronous operation presented in Eq.(4.4). Otherwise, the updated values are sent to the target nodes.

In case of a message failure, the sending node repeats the message transmission, to make sure the message finally reaches the target node, as it does not possess information about the state of the that node. That can minimize the packet error rate.

Although the synchronous Gaussian belief propagation diverges, it is experimentally shown that when there is a packet error due to an outage event, with the aforementioned probability, GBP leads to convergence after 120 iterations, as the residual of belief from the actual mean value is of order 10^{-2}. However, a drawback of the implementation is the overall delay observed, because each iteration requires 20 seconds, and only the information transmission iterations are presented in Fig. 5.3.

[1]The detailed expression of the elements of the belief means stem from the standard equations of sum-product and and can be seen in Eq. (20) of [11].

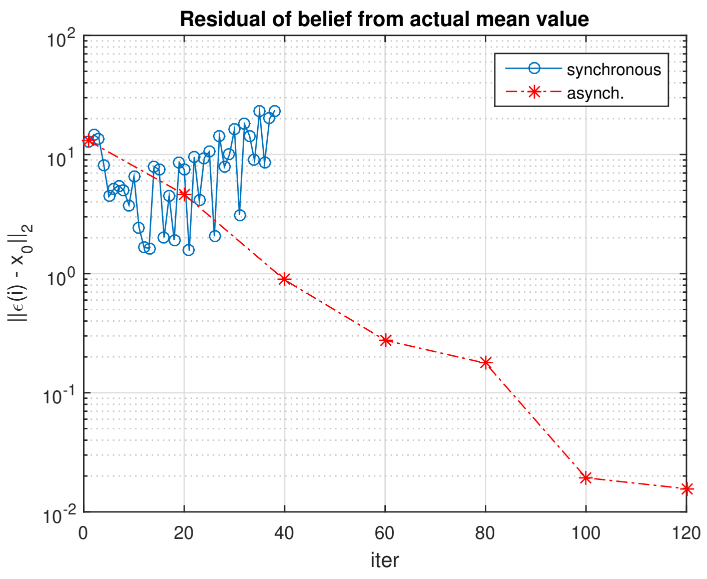


Figure 5.3: Convergence of the 3 node experimental GBP.

5.2 Implementation of Average Consensus Algorithm

The consensus algorithm was also implemented in a chain WSN, depicted in Fig. 5.1; connectivity among the terminals is also depicted.

Given the chain communication between the three nodes, the following **W** of Eq. (4.16) and (4.17) is chosen:

$$\mathbf{W} = \begin{bmatrix} 2/3 & 1/3 & 0 \\ 1/3 & 1/3 & 1/3 \\ 0 & 1/3 & 2/3 \end{bmatrix}. \tag{5.3}$$

The message passing WSN analyzed above (Ch. 3.1) was used for the implementation. At first, all terminals perform a measurement of environmental humidity. Then, the network aims to estimate the average value of the humidity in the area of the nodes. The procedure is composed again of two rounds, one for information transmission and another one for performing

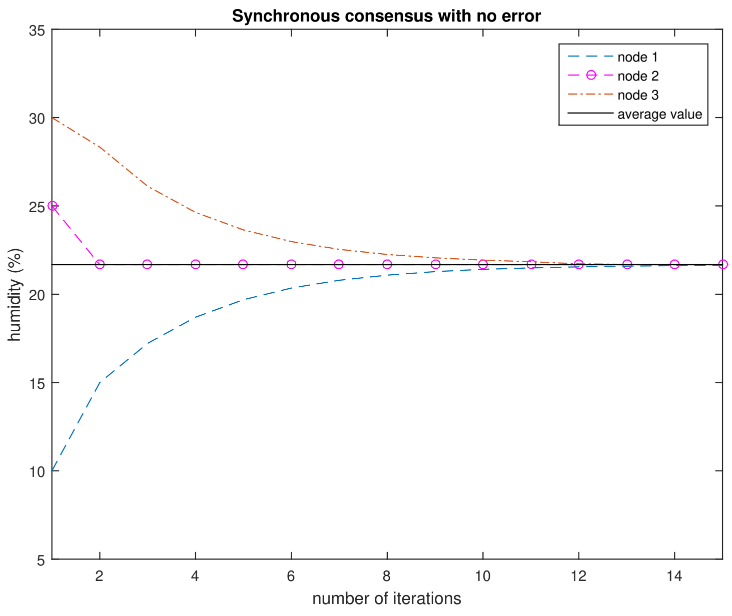


Figure 5.4: Convergence of the 3 node experimental consensus.

the necessary computations. The number of iterations shown in the tables refers *only* to the rounds necessary for information transmission.

Table 5.1: Results with fixed, known "sensor values". Specifically, node 1 "humidity" was set to 10%, node 2 to 25% and node 3 to 30%.

Type	Iterations	True value	Est. value
no message failure	15	21.67	21.67
node 1 off for 20 secs	17	21.67	22.89
node 2 off for 20 secs	17	21.67	22.78
node 3 off for 20 secs	17	21.67	18.77

Table 5.1 offers the results of an experiment performed with fixed, known values (instead of using actual humidity measurements). When the network operates without an outage event, the average consensus converges to the true average value (Fig. 5.4). The figure shows how the algorithm converges, utilizing the synchronous update of Eq. (4.16) and $\mathbf{x}^{(0)} = \begin{bmatrix}10 & 25 & 30\end{bmatrix}^T$. As expected, after 15 iterations, all terminals reach agreement and converge to the arithmetic mean $(10 + 25 + 30)/3 = 21.667 \triangleq \bar{\mathbf{x}}^{(0)}$. However, there is

Table 5.2: Results using actual humidity sensor measurements.

Values	Type	Iterations	True value	Est. value
node 1: 42.96% node 2: 36.28% node 3: 36.11%	no message failure	12	38.44	38.44
node 1: 61.31% node 2: 57.77% node 3: 60.53%	node 1 off for 20 secs	15	59.87	61.11
node 1: 26.58% node 2: 24.72% node 3: 26.45%	node 2 off for 20 secs	6	25.91	30.89
node 1: 61.83% node 2: 58.64% node 3: 61.14%	node 3 off for 20 secs	13	60.53	61.34

an overall delay, as it is noticed that the convergence was reached after 9 minutes.

At the case of an outage event, it is shown that when the node (at which the outage event took place) "rejoins" the network, the network will converge close to the true average value, at the expense of a slightly increased delay.

Instead of fixed, known values, the nodes were subsequently programmed to utilize the actual measurements taken by an embedded (in the node) humidity sensor (Si7021 Relative Humidity sensor) and the experiment of the previous paragraph was repeated. The results are offered in Table 5.2 and Fig. 5.5. The nodes are placed two meters apart which leads to limited deviation of the humidity measurements and thus, the convergence rate is increased (compared to the case of using fixed values). Increased divergence for the case of an outage event in node 2, is attributed to the change in the environmental conditions during said event. It is emphasised that the reported values stem from a *single* run (and not as the arithmetic mean of several consensus experiments).

If multiple experiments take place, the mean of the convergence values will be very close to the actual mean value, despite the outage event [12]. However, that will leads to an overall delay.

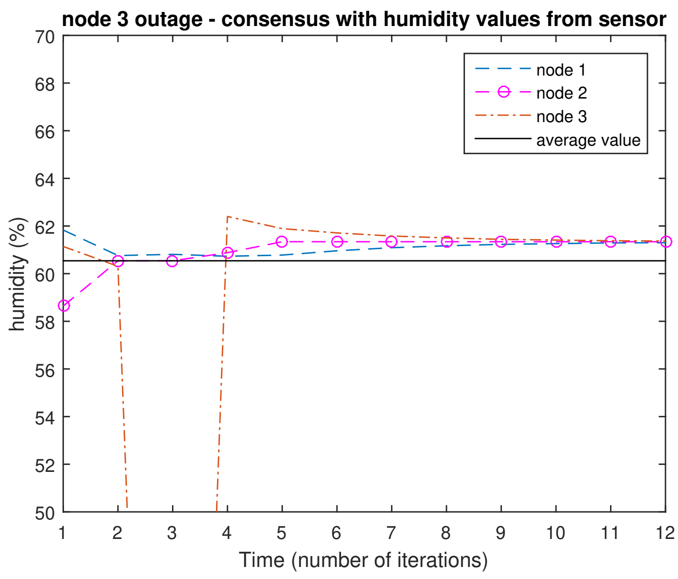


Figure 5.5: Experimental behaviour of the Consensus algorithm when an outage event occurs at node 3.

It is clear now, that, instead of an outage event in a node, or a communication error between the nodes, the asynchronous way can lead to faster convergence in GBP, and a result close to the actual value in Consensus. Thus, it is proven that an actual WSN, using Thunderboards Sense 2, can provide reliable in-network decisions, for both the Gaussian Belief Propagation algorithm and the Average Consensus Algorithm.

Chapter 6

Conclusions

6.1 Conclusion

This work offered a proof-of-concept of how commercial low-cost, low-power nodes, can be exploited to create a wireless sensor network capable for in network inference, with large communication ranges, at 10.8 dBm transmission power. Also, it was presented how a energy harvester can exploit the solar energy and power the Thunderboards with low outage probability. Also, Zigbee's centralized network was presented and it was justified why it is not the ideal solution for a wireless sensor network. Furthermore, it is proven that in network inference in a distrbuted manner is possible using the aforementioned nodes, and it was explained why the asynchronous way provides better results.

6.2 Future Work

The main focus of this work was to implement a distributed network that will exchange humidity values for autonomous decision. Also, a centralized network for humidity monitoring may be implemented.

Security applications can be possible using the implemented message passing WSN, by exploiting Thunderboard's embedded orientation sensors.

Also, the implementation of 16 energy harvesters is possible. Due to time constraints, only 3 energy harvesters were implemented, and so the 16 node message passing network in Sec. 3.1 was limited to a 3 node network in Ch. 5.

Finally, the future work should focus on reducing the overall delay especially of the distributed network, as time is a vital factor in a network operation.

Appendix

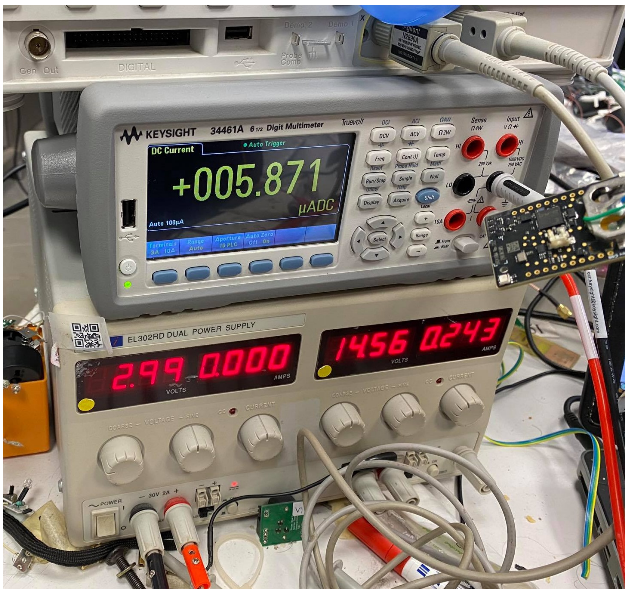

KEYSIGHT 34461A 6½ Digit Multimeter
DC Current
+005.871
µADC
EL302RD DUAL POWER SUPPLY
2.99 0.000
14.56 0.243

Figure 6.1: Sleep mode consumption.

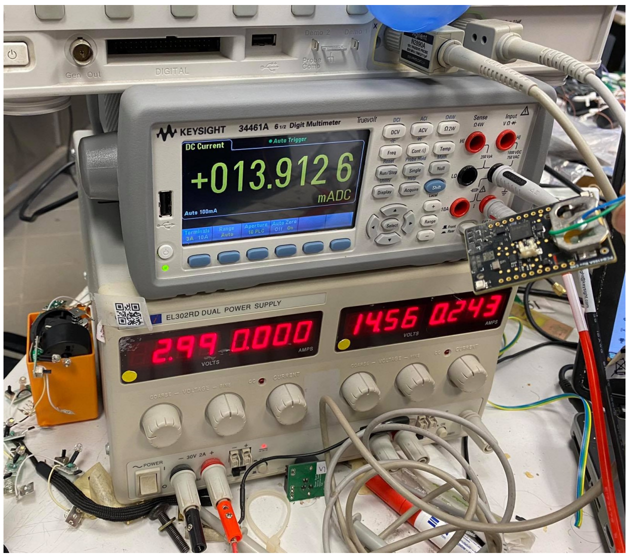

KEYSIGHT 34461A 6½ Digit Multimeter
DC Current
+013.912 6
mADC
EL302RD DUAL POWER SUPPLY
2.99 0.000
14.56 0.243

Figure 6.2: Active mode consumption.

Bibliography

[1] P. Vasilakopoulos, “Design and implementation of a low-cost bidirectional embedded backscatter link,” Jul. 2021, undergraduate Thesis at Technical University of Crete, Supervisor A. Bletsas.

[2] “Ug309: Thunderboard sense 2 user’s guide.” [Online]. Available: https://www.silabs.com/documents/public/user-guides/ug309-sltb004a-user-guide.pdf

[3] “Ug103.2: Zigbee fundamentals.” [Online]. Available: https://www.silabs.com/documents/public/user-guides/ug103-02-fundamentals-zigbee.pdf

[4] H. Xiaoci, J. Yi, S. Chen, and X. Zhu, “A wireless sensor network-based approach with decision support for monitoring lake water quality,” *Sensors*, vol. 15, pp. 29 273–29 296, 11 2015.

[5] S. S. Sonavane, V. Kumar, and B. P. Patil, “Designing wireless sensor network with low cost and low power,” in *2008 16th IEEE International Conference on Networks*, 2008, pp. 1–5.

[6] “Thunderboard sense 2 sensor-to-cloud advanced iot kit.” [Online]. Available: https://www.silabs.com/development-tools/thunderboard/thunderboard-sense-two-kit

[7] “Radio abstraction interface layer (rail) sdk.” [Online]. Available: https://www.silabs.com/developers/flex-sdk-radio-abstraction-interface-layer

[8] “Efm32 32-bit mcu ultra efficient energy modes.” [Online]. Available: https://www.silabs.com/mcu/32-bit/efm32-energy-modes

[9] W. K. Seah, Z. A. Eu, and H.-P. Tan, "Wireless sensor networks powered by ambient energy harvesting (wsn-heap) - survey and challenges," in *2009 1st International Conference on Wireless Communication, Vehicular Technology, Information Theory and Aerospace Electronic Systems Technology*, 2009, pp. 1–5.

[10] A. Pop-Vadean, P. P. Pop, T. Latinovic, C. Barz, and C. Lung, "Harvesting energy an sustainable power source, replace batteries for powering WSN and devices on the IoT," in *Materials Science and Engineering Conference Series*, ser. Materials Science and Engineering Conference Series, vol. 200, May 2017, p. 012043.

[11] B. Li and Y.-C. Wu, "Convergence analysis of gaussian belief propagation under high-order factorization and asynchronous scheduling," *IEEE Trans. Signal Process.*, vol. 67, no. 11, pp. 2884–2897, Jun. 2019.

[12] V. Papageorgiou, A. Nichoritis, P. Vasilakopoulos, G. Vougioukas, and A. Bletsas, "Towards Ambiently Powered Inference on Wireless Sensor Networks: Asynchrony is the Key!" in *5th IEEE International Workshop on Wireless Communications and Networking in Extreme Environments*, July 2021.

[13] D. Jian, S. Ma, Y.-C. Wu, S. Kar, and J. Moura, "Convergence analysis of distributed inference with vector-valued gaussian belief propagation," *Journal of Machine Learning Research*, vol. 18, 11 2016.

[14] L. Xiao, S. Boyd, and S.-J. Kim, "Distributed average consensus with least-mean-square deviation," *Journal of Parallel and Distributed Computing*, vol. 67, no. 1, pp. 33–46, 2007. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0743731506001808