FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Remote Supervision System for Aquaculture Platforms

Manuel de Araújo Sousa e Silva

MSc DISSERTATION

Supervisor: Professor Doutor Nuno Alexandre Lopes Moreira da Cruz

Co-Supervisor: Doutor Fernando P. Lima

July 26, 2016

Resumo

A aquacultura é uma atividade que requer meses de trabalho até se atingirem os objetivos pretendidos, e que está altamente depende de fatores imprevisíveis e incontroláveis. Uma variação brusca nas condições ambientais pode ser suficiente para destruir o trabalho de vários meses. Como tal, é fundamental possuir ferramentas que permitam monitorizar atentamente e registar toda a informação relacionada com estes processos, para que dessa forma seja possível perceber e compreender o desenvolvimento da espécie em causa. O rápido acesso a esses dados pode ser de extremo interesse para se resolver situações problemáticas em tempo útil e dessa forma otimizar as condições de produção nesta atividade.

A dissertação apresentada neste documento tem dois objetivos principais. Passando o primeiro pela proposta de uma solução para um sistema de monitorização remota para estações de aquacultura. Sendo o outro objetivo principal, o desenvolvimento de um protótipo cuja implementação permita validar as principais funcionalidades presentes na arquitetura apresentada.

A arquitetura proposta prevê uma rede de sensores cujas comunicações são baseadas no protocolo I2C. Esse protocolo foi escolhido porque não apresenta qualquer tipo de restrição no design da topologia da rede e permite uma fácil reconfiguração da rede de sensores. Esta arquitetura contempla dois tipos de *sensor nodes*, o *main sensor node* que é o mestre na rede I2C, sendo responsável pela coordenação da rede de sensores, pela gestão do processo de recolha de dados, funcionando também como uma *gateway* transmitindo os dados recolhidos para um destino previamente definido. E os *tiny sensor nodes* que funcionam com escravos na rede I2C, assegurando a ligação com os sensores e o correto funcionamento do processo de amostragem.

O processo de implementação descrito neste documento explica como foram desenvolvidas e testadas as principais funcionalidades presentes na arquitetura proposta, tendo o *main sensor node* sido implementado numa *Beaglebone Black* e os *tiny sensor nodes* em microcontroladores *ATtiny85*.

O protótipo desenvolvido permitiu o teste das principais funcionalidades, tendo os resultados obtidos permitido perceber que este sistema se adequa perfeitamente aos objetivos pretendidos.

Abstract

Aquaculture processes typically take a considerable amount of time, usually months, and a single unwanted event can ruin a whole harvest. This loss of several months work usually happens without having means or information to understand its main cause. Therefore the monitoring and logging of those environmental conditions is essential to understand behaviors and dynamics related with this process.

This dissertation had two main objectives, the development of a solution for a Remote Supervision System for Aquaculture Farms and the implementation of a prototype that allowed a practical validation of the proposed solution.

The proposed architecture contemplates the development of an I2C embedded sensor network. This protocol has been chosen since it does not present restrictions in the design of the network topology and allows an easy network reconfiguration. This architecture is composed by two different types of sensor nodes. The main sensor node is responsible for coordinating the whole I2C sensor network, as its I2C master, handling the data collection process, and also serving as a gateway to transmit the collected data to a previously defined destination. The tiny sensor nodes are programmed to operate as I2C slaves that can be added to the bus and are responsible for interfacing with the desired sensors and ensuring the sampling process.

The implementation process described in this document focused on the development and testing of the core features presented in the proposed solution architecture. In the described prototype, the main sensor node has been implemented in a *Beaglebone Black* board, and the tiny sensors nodes are based on *ATtiny85* microcontrollers.

The developed prototype main and critical functionalities have been tested, and it has been concluded that so far the proposed architecture meets the requirements proposed for this system.

Acknowledgments

Esta dissertação é o culminar de 6 anos, ao longo dos quais o caminho nem sempre foi óbvio, mas que foram um período de enorme crescimento e evolução pessoal.

Em primeiro lugar queria agradecer ao Professor Nuno Alexandre Cruz pela excelente orientação e pela enorme disponibilidade. Mas acima de tudo pelo enorme entusiasmo e confiança que me transmitiu ao longo de todo este processo.

Gostava também de realçar o papel do Fernando Lima e Rui Seabra do CIBIO/InBIO, que desde a primeira hora se mostraram disponíveis para qualquer ajuda ou esclarecimento, e por serem um exemplo de que a falta de conhecimento prévio nunca pode ser desculpa para não se tentar, e conseguir atingir os nossos objectivos.

Queria também agradecer ao José Carlos Azevedo pela ajuda preciosa no desenvolvimento das PCBs que são um dos resultados desta dissertação.

Não também posso deixar de agradecer ao João Ventura e ao Bruno Silva, que durante toda esta dissertação tiveram paciência para discutir as minhas dúvidas e problemas técnicos. E pelas inúmeras discussões de cariz técnico ou aleatório.

À minha família mais próxima por toda a paciência, compreensão e por me terem sempre proporcionado todas as condições para que eu atingisse os meus objectivos pessoais. E uma especial palavra ao meu irmão Nuno, por todo o apoio que me deu ao longo desta dissertação.

Manuel Sousa e Silva

"Try and leave this world a little better than you found it and when your turn comes to die, you can die happy in feeling that at any rate you have not wasted your time but have done your best."

Robert Baden Powell

Contents

List of Figures

List of Tables

Abbreviations

ADC	Analog to Digital Conversin
BBB	BeagleBone Black
DIP	Dual In-line Package
DTO	Device Tree Overlay
ESN	Environmental Sensor Networks
FDT	Flattened Device Tree
FEUP	Faculdade de Engenharia da Universidade do Porto
GSM	General System for Mobile
GUI	Graphical User Interface
IDE	Integrated Development Environment
IoT	Internet of Things
IR	Infra-Red
ISR	Interrupt Service Routine
I2C	Inter-Integrated Circuit
PMIC	Power Management Integrated Circuit
OS	Operating System
RF	Radio Frequency
RPi	Raspberry Pi
RTC	Real Time Clock
SOIC	Small Outline Integrated Circuit
μC	Microcontroller
UWSN	Underwater Wireless Sensor Networks
WSN	Wireless Sensor Networks

Chapter 1

Introduction

1.1 Context

The world's population has been constantly increasing in the last 40 years and this has a huge impact in the global food supply chain. The earth natural resources are not unlimited and therefore it is vital to explore solutions to increase the amount of food available worldwide [1].

Aquaculture is a primary activity that consists in the farming of aquatic organisms under controlled conditions.

Although this field already has an important position in the international food supply system, it still presents considerable development and evolution possibilities. In fact, presently half of fish consumed globally comes from aquaculture [2].

Due to its geographical conditions, Portugal has a very significant potential for expanding its open sea aquaculture network. However, being an open air activity, aquaculture is highly dependent on environmental conditions and it is exposed to a wide range of external factors that cannot be controlled. Parameters such as water temperature, water quality, nutrition, and diseases play a critical part in this process [3].

In response to that, producers and researchers are involved in a continuous improvement process, which leads to new solutions, methods and technologies.

1.2 Motivation

The aquaculture processes typically take a considerable amount of time, usually months, and a single unwanted event can ruin a whole harvest. This loss of several months work usually happens without having means or information to understand its main cause.

The Monitoring and Logging of those environmental conditions is essential to understand behaviours and dynamics related with the development of the species. That information could allow the producers to prevent external factors impacting in the production.

In the long term, the analysis of this data could generate knowledge that would increase and improve the know-how of the producers, which would be able to enhance their processes during

the development and growth of the species. Therefore, these systems can be a key factor in the future of this industry.

The logging of these variables can also be very relevant for ecological purposes. Having specific data concerning different environmental parameters about an individual and its surrounding environment enables the study and correlation of all that information, which might be the basis for some evolution in the understanding of ecosystems dynamics. There are a considerable number of solutions available, which perform these tasks. However they still have various constraints, the solutions are not customized for each experiment or environment; usually it is mandatory to use them using the supplier's software; and these solutions usually represent high costs.

This dissertation is the result of a partnership between FEUP and CIBIO/InBIO, a research Institute associated with University of Porto.

CIBIO/InBIO researchers clearly understood the huge importance that electronics can have in the gathering of information for their studies and started developing some systems using low cost electronics products and open source software. This development was always made in response to their immediate needs, but has been thoroughly documented and is published in a scientific journal. Therefore, it is available for fellow researchers, enabling future works [4] [5].

These efforts developed in CIBIO/InBIO are an important basis for this Dissertation, because it demonstrates the existing need for new and different solutions in this area.

CIBIO/InBIO has been involved in a project with the Spanish authorities in Galicia, and has a clear interest in using the sensors they developed, in order to collect data regarding mussel behaviour.

1.3 Objectives

This MSc Dissertation project contemplates two main objectives. The development of a solution for a Remote Supervision System for Aquaculture Farms, that must contemplate the functionalities of both an autonomous data logger and a remote monitoring system.

Another objective is the implementation of a prototype, that will allow a practical validation of the proposed solution. As this project has been developed in partnership with CIBIO/InBIO, the system prototype will be developed taking in consideration its application in a specific context.

However, is is also expected that the system can easily be changed and operational in a different environment, supervising a completely new situation, employing others sensors, and analysing different data.

Finally, the last objective consists in the elaboration of a paper describing the proposed solution and the implementation of the system prototype. If this task is accomplished, the document will be submitted to a conference in order to be published in a scientific journal.

1.4 Document Structure

This document comprises six chapters. This first chapter presents the main factors and motives behind this dissertation, and its objectives. The 2nd chapter focuses on the State of the Art. Chapter 3 aims to present and justify the proposed solution in this MSc dissertation.

Then, the implementation of this architecture main features are presented and discussed in chapter 4. Chapter 5 is dedicated to testing the developed system in order to validate the proposed solution.

Finally, in chapter 6, this work's conclusions are presented, and future works are discussed.

Chapter 2

State of the Art

This chapter starts by presenting a general overview on the field of knowledge of this dissertation. After that, it offers an analysis of related works described in scientific literature, as well as some of the solutions available in the market regarding systems whose functionalities are similar to the ones expected in the prototype.

Finally, this chapter focuses on important concepts central to the development of this dissertation, discussed and presented in order to help the understanding of some choices made later on during the development of the proposed solution.

2.1 Context

Knowledge has always played a fundamental part in the evolution of mankind. Man has always been eager to understand the interactions between the physical world and the environment and master those forces to His benefit.

The huge evolution in electronics enabled the mass production of components, and opened access to several technologies and concepts. The miniaturization of complex electronic circuits, the availability of capable battery solutions, RF modules, microprocessors, commercial sensors, and several others components, represented an important boost for the development of small sized electronic devices.

Researchers started to use simple stand alone devices, the data loggers, to sample environmental data. By the year of 2000, Whiteman *et al.* [6] stated that at that time the advances in electronics miniaturization had finally allowed the commercial development of inexpensive temperature sensor and data logger solutions. Simple and battery powered devices were already capable of operating autonomously for long periods of time, even doing some internal processing of the measured data.

Heidemann and Govidan [7] defined embedded sensor network as a network of embedded computers that interacted with the surrounding environment. This idea entailed three essential concepts: nodes, motes and sensors. Nodes and motes, at different levels, are responsible for the data acquisition, processing, and for the communication with the network. Sensors represent the

lower level in this architecture, as they are responsible for measuring the desired variable. At that time, in the year of 2004, the authors already considered that this area was going to have very significant developments in the upcoming years.

Hart and Martinez [8] deemed these embedded systems revolutionary in earth system science, describing them as Environmental Sensor Networks (ESN). For the authors it is clear these systems have evolved from the initial data loggers, which had some limitations and were not connected to any network. These new trends in earth system monitoring represent an opportunity for biologist and researchers, as it easier to obtain more and better data so they can fully concentrate in their inquiries.

In [8], the authors identified seven main challenges regarding ESN. They believed power management, usability, standardization, data quality, security, data mining and harvesting, and the development of new sensors were the key challenges for anyone working in this field.

Chong and Kumar [9] were capable of efficiently summarize this field of knowledge, as they considered the evolution in sensor network related systems was dependent on three main issues: sensors, communications and computational resources.

Another important aspect when it comes to sensor networks is related with the data transmission, since it is highly dependent on the sensor nodes localization, distance between nodes, and environment characteristics. Sensor nodes might be wired or use wireless communications, and they can be located in different environments that may influence signal transmission.

This new trend of uploading all the data collected to an external database led to a new concept known as Wireless Sensor Networks (WSN). In this paradigm, all the sensing and data logging devices,which may be deployed over a large area, are connected in a wireless network capable of transmitting all the data the sensors gathered to an external system [10].

Gateway is a key concept in this communications networks and it is essential in order to understand the sensor networks paradigm, since it can be seen as a protocol converter. A gateway is a network node capable of interfacing with a different network, which uses another communications protocol. Typically the information is transferred to a web server.

Davis and Chang [11] review about Underwater Wireless Sensor Networks (UWSN) outlined some important factors regarding the implementation of sensor nodes in aquatic environments. The authors introduced a considerable number of difficulties. Power harvesting was presented as a challenging topic, since one of the main power sources, solar energy, is not available underwater. But the main issue lied with the communications, as the environment presents several uncontrolled and unpredictable constraints that affect the reliability and robustness of data transfers between the system nodes.

With the on-going developments concerning the topics already referred, as well as many others, a new concept was established, the Internet of Things (IoT). This concept refers to a global network of physical objects, that are embedded with sensors and anyway connected to a network. All this sensing and monitoring generates enormous amounts of data that has to be carefully stored, analysed, and protected [12].

2.2 Commercial Solutions

There are several solutions available in the market for data logger and remote monitoring systems. This equipments are widely used in activities such as: agriculture, oceanography, aquaculture or ecology. The end purpose might differ, but the technical challenge is almost the same for the manufactures, and with some changes the systems can be used in different applications.

Delta-T [13] offers a wide rage of data loggers orientated for agricultural monitoring. Nexsens products [14] are more orientated to the monitoring of aquatic environments. Campbell Scientific [15] has a wide range of data loggers that can be adapted for almost all the areas referred above. These companies also provide several peripherals that add extra features to the system (sensors,communications modules, weather station or power supplies). The sensors most used by these systems can measure variables such as: temperature, humidity, barometric pressure, pH sensors, water turbidity, soil moisture, and many other variables.

However, these solutions still have several constraints. They are not customized for each experiment or environment, and customization comes at a high cost, since it is only possible to use their own sensors or peripherals. Even tough, the major problem these commercial solutions pose is the mandatory use of propriety software and proprietary format for the data files.

2.3 Related Works

As already referred in section 2.2, the commercial solutions for this kind of purpose are not satisfactory, and the scientific community is constantly developing devices in order to answer to the opportunities not covered by the solutions available in the market, or to propose different, simpler, and cheaper solutions.

This section analyses several projects published in scientific literature. Some of these works are not directly associated with aquaculture, but similar solutions are being employed in completely different areas, such as telemedicine or oceanography. The end purpose might not be the same, but the technical solutions are of high interest to the work under development.

Blank *et al.* [16] presented a prototype of a device projected for telemedicine purposes. The described module is responsible for the data collection and communications management of a network of wearable sensors. Essentially, it is a gateway to enable the transmission of data to a web server. The authors reviewed several approaches regarding sensor data collection and distribution and concluded that the majority of the systems were too customized, and therefore decided to develop a system with both modularization and extensibility.

This system main operation is based in a powerful microcontroller that interfaces with several devices and sensors. It it has three different ways to communicate with the server: Ethernet, Wi-Fi and GSM. However, for applications where the expected tasks to be performed by the system, are more demanding in terms of computational processing, it is possible to attach an embedded Linux computer (Beaglebone). In this situation the Beaglebone assumes the system operation and the microcontroller serves as an interface with all the peripherals.

Alvarez Carasco *et al.* [17] work describes the development of a simple system responsible for the data acquisition of an accelerometer sensor, and the communications with the network gateway. This approach is quite interesting as they use the low dimension, low power, and low cost microprocessor (ATtiny85) as the main processing unit. This enables the utilization of the sensor node as a wearable sensor that does not interfere with the experiment it is supposed to monitor.

Brooke *et al.* [18] present their work in an underwater sensor module for determining ocean pH. The system is essentially a straightforward data logger. The most relevant point in this project is the strategy employed to enhance the system power consumption, thus increasing the sensor module lifetime. A low power microcontroller (ATtiny 85) that can be put to sleep mode, is used to activate the main microprocessor, ADCs and sensors.

Rajgarhia *et al.* [19] ultra low power WSN gateway for outdoor deployments, is also a very interesting work. The authors make a clear distinction between WSN nodes and WSN gateway, as they state the last one is responsible for the data aggregation in its network, local network management, and connecting with an external network. Their solution contemplates an embedded system based on an ARM processor, with several built-in functionalities, and that is capable of interfacing with external peripherals using standard protocols (UART, I2C, SPI, GPIO). This main processor is also responsible for interfacing with WSN nodes and with the GSM modem.

The proposed solution differs from many other similar works, as the software architecture present a very balanced and structured modes of operation that enhanced the system power consumption, since they only activate the power hungry features in specific occasions. Another interesting feature is the 'power status' classification, as the processor receives information regarding the batteries state of charge, via SPI, and it decides whether the system can be 'fully functional', in 'reduced function', a mode where the modem stops, as the communications is the most demanding module in terms of power consumption, or possibly in 'minimal function', where all features are suspended till the processor receives contrary information.

An interesting architecture is proposed by Chianese *et al.* in [20], where the authors present a smart multisensor framework for a sensor network, whose sensor nodes are based in Beaglebone Black boards.

In this work the sensor nodes are all based in identical devices, but they can assume one of four possible roles in the network: Stand-Alone, Client, Monitor or Server, depending only on the software loaded to the board. The role each sensor node assumes in the network depends on the specific application where this framework is employed.

It is important to refer that this is only possible, as the Beaglebone Black has excellent interfacing capabilities, that is fundamental for its operation as a Client collecting sensor data, but also because it has a considerable computational power that can process the software responsible for the operation of the whole network, when the sensor node is working as Monitor or Server.

Manikandan *et al.* [21] work focuses on the application of the I2C protocol in a sensor network. They state that this protocol is mostly used within a PCB and between ICs, but they present

a single master and multi-slave architecture with the several sensors externally connected to an Atmel microcontroller. The authors conclude that the proposed system can be extended, and use an I2C bus as the basis for a large sensor network.

2.4 Relevant Technologies

This section aims to provide an overview regarding some technologies that will be of interest throughout the development of this dissertation project.

2.4.1 Bus Communications

One of the key factors in embedded systems is related with the exchange of data between its components. For this purpose there are several communication protocols defined, carefully structured, and documented.

A protocol might be parallel or serial. Parallel communications allow the transfer of several bits at the same time. This usually requires a dedicated channel for each data transfer, typically a wire between each two end points participating in that communication. Serial communications only allow the transfer of a single bit at a time. This is used generally in networks where the cost of cable is an important factor. This kind of communication can also be divided in synchronous and asynchronous, depending if the protocol has a dedicated clock to trigger the communications or not.

When developing embedded systems, several authors consider serial communications to be a better option, as its implementation is cheaper, because ICs already have serial interfaces, and can easily be integrated in a serial bus [22].

However, it is fundamental to select a protocol that defines the operating rules for all the objects sharing a common bus.

Literature identifies UART, SPI, and I2C, as the main protocols for serial buses [22]. Each as its own singularities being more suited for specific applications.

UART stands for Universal Asynchronous Receiver Transmitter, which is a communications standard developed in the sixties. UARTs are capable of translating data from the computers' parallel internal buses to serial streams that can be transmitted using a single cable (one wire for each direction). The method only allows point-to-point transmission, which represents a handicap for many applications.

Serial Peripheral Interface (SPI) is a protocol developed by Motorola, to serve as synchronous serial data link for the communications between their peripherals and microcontrollers [23]. SPI is officially a "4-wire full duplex synchronous serial data link" and only allows one master in its architectures.

SLCK wire sends a serial clock signal provided by the master to all the slaves. This is vital for the synchronism in the communications.

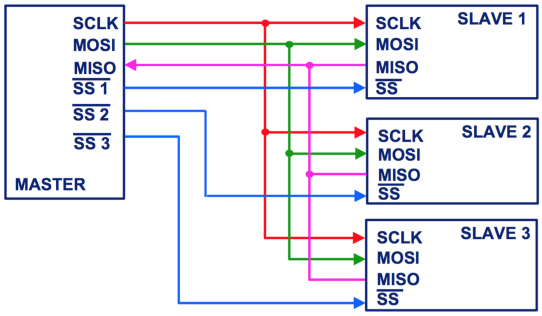

Figure 2.1: SPI typical topology. Extracted from [24].

MOSI stands for Master Out-Slave In, and MISO stands for Master In-Slave Out. These wires are used as data lines between the several objects in the network.

SS is used as a Slave Selector signal, and it allows the master to select the device which is going to be using the MOSI and MISO lines for data transmission.

Although this protocol is described as a 4 wire method, it is effectively a '3+n' wires protocol, with 'n' being the number of slaves connected to the master. This can prove to be an inconvenient when implementing some wired networks using this protocol.

SPI protocol does not have any defined maximum data rate, nor has acknowledgement routines to confirm the success of the data transmissions.

Inter-Integrated Circuit (I2C) is a serial synchronous protocol developed by Philips, which enables communications between ICs, computers and microcontrollers. This protocol is able to manage several masters and several slaves in the same bus and, contrary to SPI, it does not require a specific wire to enable chip selection, as it is done differently. Each IC connected to the bus has a specific and unique address [24].

I2C only requires two bus lines, SDA stands for Serial Data Line, and SCL stands for Serial Clock Line. The data rate may vary between 100kb/s, 400kb/s, and 3.4 Mb/s, depending on the transmission modes, which can be, respectively, standard mode, fast mode, or high-speed mode.

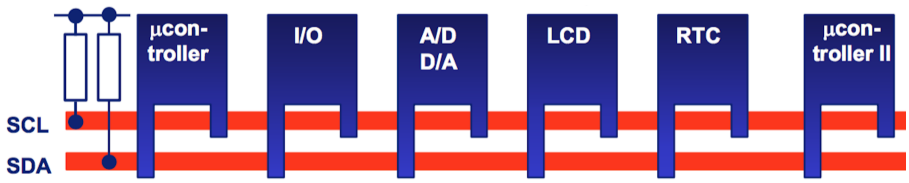

Figure 2.2: I2C typical topology. Extracted from [24].

The previous figure presents an example a typical topology when using an I2C bus in an embedded system. Several peripherals are able to communicate using the same line, and this can be a key factor in applications where the cable cost has influence, and when the data acquisition system has limited inputs.

2.4.2 Wireless Communications

One of the most important modules in a remote monitoring system, is related with data transmission, and as already referred in section 2.1, literature outlines communications as one of three vital factors in sensor networks.

When choosing a communications protocol for a system, there are several factors that must be taken in consideration. First of all, it is vital to know the localization and environmental conditions of the places where the devices which will be networked because range is the key factor.

Nowadays, General System for Mobile (GSM) has widespread coverage and due to that factor, several authors consider it a very compelling option for long range communications in remote data acquisition systems, as it provides reliable data transmission.

GSM provides three data services: Circuit-switched Data (CSD), Short Message Service (SMS), and General Packet Radio Service (GPRS). CSD reserves an audio channel to enable the communication between two devices during a phone call. SMS can be used to send a relatively short message to another device connected to a GSM network. GPRS is packet oriented data service, which can establish an Internet connection between a gateway and a system that serves as a host for a web based server [25] [26].

An Internet connection can also be established via Wi-Fi if any wireless network within the range of the desired device is available and connected to the internet. This protocol is defined in IEEE 802.11 that defines all its specifications [27].

Another communications protocol that is widely described in literature is the ZigBee protocol. It is employed in WSN, as mean of data transmission between sensor nodes and the system gateway.

The ZigBee protocol was developed for embedded systems and battery powered applications, therefore it presents low power consumption. It is a reliable and secure communication protocol, as it was designed to operate in harsh environments.

This method presents a setback, since it has a limited range (comprehended between 10-100 (m)) and ZigBee devices can only communicate within its own protocol, thus the network must have at least a device that works as a gateway. Otherwise it is impossible to transmit data from a monitoring station, that in many situations might be in a far location. Even so, it is considered an efficient method for data transmission between sensor nodes and motes [27] [28].

Chapter 3

System Overview

This chapter provides some insight about this project's main challenge and the associated system requirements. After defining the system development methodology to be used, and reaching a clear understanding of what the main problem was, a suitable technical solution was proposed.

3.1 System Development Methodology

CIBIO/InBIO has been involved in a project with the Spanish authorities in Galicia, and has a clear interest in having a system capable of collecting data regarding mussel behaviour and local environmental parameters, manage and store the acquired data, and transmit that information to a specified destination.

Therefore it is clear that a system needs to be developed, and for that purpose a Systems Engineering [29] approach will be employed, as it is widely considered as solid guideline for this kind of development processes.

The Systems Engineering methodology contemplates several stages that are discussed and presented throughout this document. Initially it is necessary to assess the final system requirements and to study the state of the art on the specific area of knowledge. Only after that a system concept can be proposed as a solution for the initial problem. Then a prototype will be implemented and tested in order to validate the proposed solution.

3.1.1 System Requirements

That project has been the main lever for this dissertation, and was essential in the requirements assessment for the system. From the meetings with the CIBIO/InBIO researchers the following requirements have been identified, as the system needs to:

- Employ a sensing method based on a IR sensor, that allows the measurement of an heartbeat signal.
- Be compatible with analog sensors available in the market.

- Operate autonomously and have low maintenance requirements.
- Capable of collecting underwater data.
- Store all the collected information.
- Transmit the sensor data to a previously defined destination.

Then there are some guidelines that have discussed and can be important in the development of the proposed solution, such as:

- Easily escalate the number of sensors.
- Low power components must be used, so that the system is capable of operating for days or weeks, only powered by batteries.

3.2 Proposed Solution

Taking into consideration the system requirements, the proposed solution contemplates the development of an embedded sensor network composed by several sensor nodes.

In order to fully explain the concept developed, figure 3.1 presents a possible topology for the system. The proposed architecture contemplates a reconfigurable sensor network, where extra sensor nodes can be added to the bus. This network is composed by two different types of sensor nodes:

- 1 Main Sensor Node that serves as the sensor network coordinator, but also functions as a the network gateway.
- 1 to 200 Tiny Sensor Nodes that are responsible for simple sampling tasks.

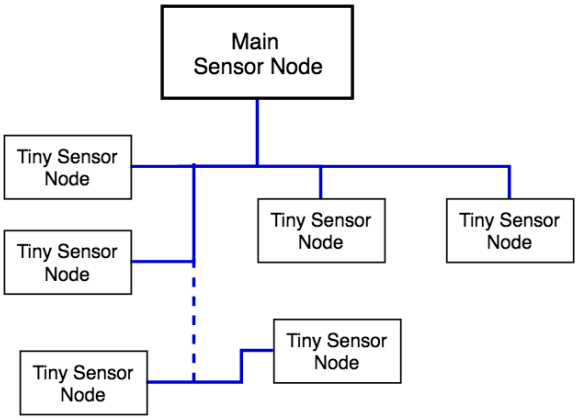

Figure 3.1: Possible topology for the proposed sensor network.

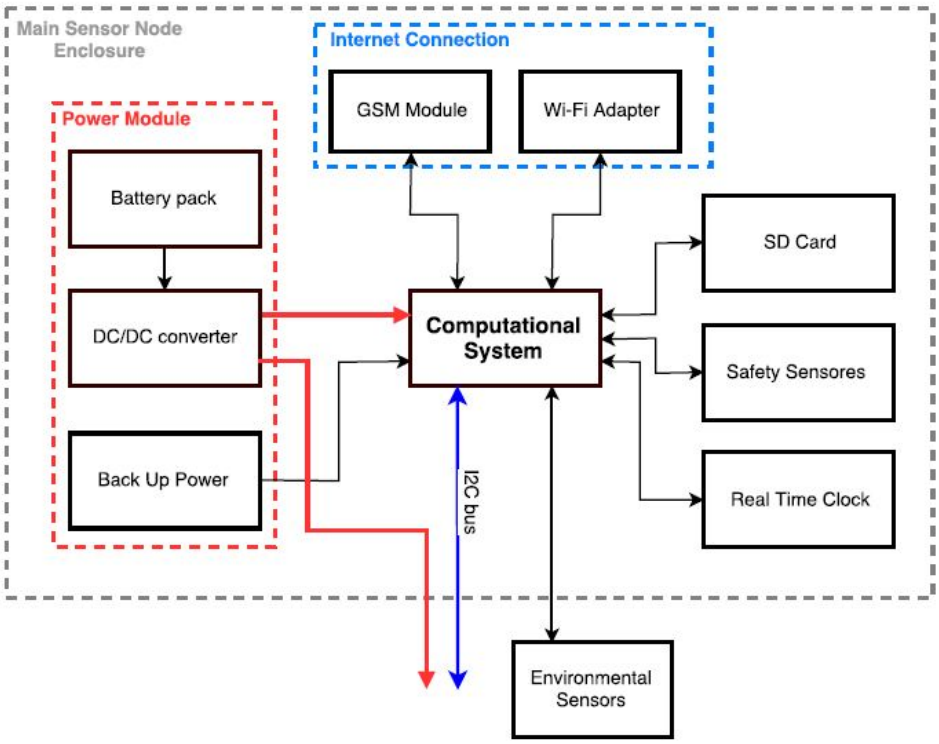

Figure 3.2: Main Sensor Node Architecture

In Figure 3.2 it is possible to see the architecture of the most complex sensor node, which will serve as a gateway for the whole network, as it will be responsible for coordinating the whole sensor network, being capable of gathering the data collected by all the sensor nodes, store that information, and whenever a network is available, transmit the sensor data to a previously defined destination.

Since this system will be responsible for coordinating the sensor network and gathering a considerable amount of data, it will require some computational power. Therefore, the core of this main sensor node will be based in a computational system, that will have I/O capabilities, in order to manage a network constituted by less complex sensor nodes, and to read some local sensors to acquire data regarding its surrounding environment.

Another important task to be ensured is the Internet Connection, that can be achieved either by a Wi-Fi adapter, if any network is available, or by a GSM Module. That can prove to be an interesting option as this kind of system are typically deployed in relatively remote locations where a usable Wi-Fi network is unavailable but the GSM coverage is strong enough to establish a GPRS Internet connection.

To ensure a proper functioning of this system some peripherals are needed. As this system is to be used as a data logger, a Real Time Clock autonomous module ensures the correct time is never lost, and an SD card is used as an external data storage for back up purposes.

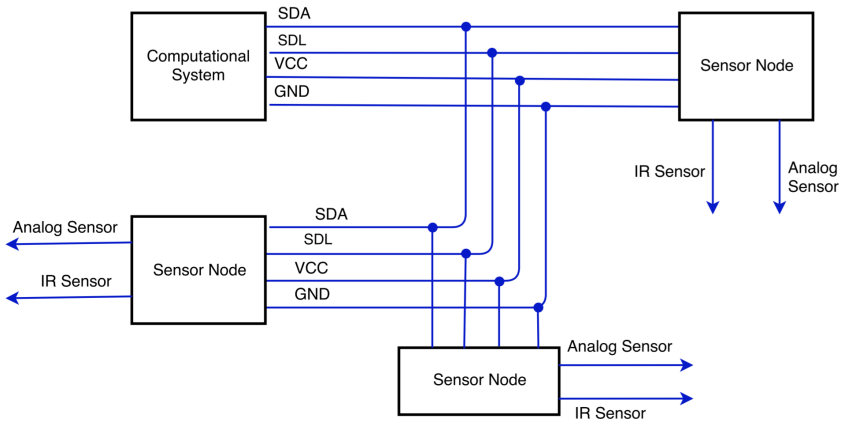

Figure 3.3: Sensor Network Possible Topology

The power module present in this main sensor node is constituted by a battery pack and a DC/DC converter, and it is responsible for powering not only this sensor node but the whole network. Another proposed feature is the inclusion of a back-up battery for the computational system, which will allow the system to finish it operation when the main module runs out of electrical power.

As this main sensor node is to be deployed in an aquatic environment, precautions should be taken, i.e. it can be stored in an IP67 enclosure.

For the sensor network, an I2C bus can be used, as this reduces the wiring complexity and consequently the costs with all the cabling. It is widely considered a solid method for communication between ICs. The main sensor node can be set as the network master and the tiny sensor nodes as slaves since it is possible to attribute an address to each one. Using this communications method it is possible to connect up to 128 nodes in a single bus, with this number depending on the bus capacitance, which can be solved using I2C repeaters for data transmission in long lines [24].

In figure 3.3 it is possible to see a possible topology for an I2C network composed by a main sensor node, that connects to 3 different sensor nodes through a cable with four wires, without any restriction about how the network is designed.

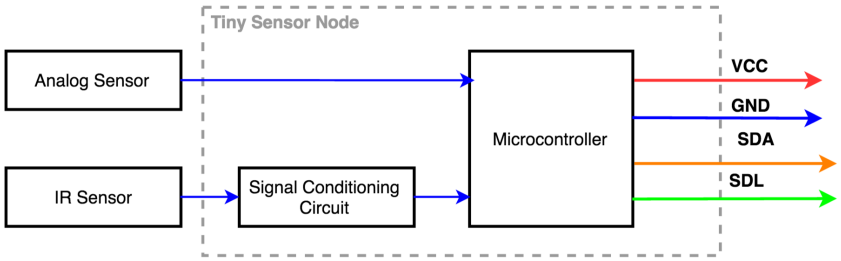

Figure 3.4: Tiny Sensor Node Architecture

In figure 3.4 it is possible to analyze the tiny sensor nodes functional architecture, where a microcontroller unit poses as the central point, as it ensures the execution of this device main tasks such as: reading the sensors, making the ADC conversion, and be part of an I2C based network.

One of the most important requirements for this device is its compatibility with a sensing method that allows the measurement of an heartbeat signal, therefore the tiny sensor nodes also includes the signal conditioning circuitry necessary for obtaining that results using an IR sensor.

These tiny sensor nodes work as I2C slaves that must be connected to a bus coordinated by the main sensor node, as it is possible to see in figure 3.1. As this devices must be able to operate in underwater conditions they are embedded in epoxy resin.

Chapter 4

Implementation

One of the main objectives of this dissertation project was the implementation of a prototype, that allowed a practical validation of the proposed solution. As this project has been developed in partnership with CIBIO/InBIO, the system prototype was developed taking in consideration its application in a specific context.

This chapter starts by providing some insight regarding the technological choices made in order to implement a prototype of the proposed solution. After that, this chapter focus on the technical development of both types of sensor nodes in order to evaluate if the proposed architecture can be a valid and interesting option for the end purpose it was designed for.

4.1 Technological Choices

This section aims to provide some insight regarding the main technological choices made in the development of the system prototype.

Scientific literature on embedded systems describes several works based on embedded Linux boards, as Linux is a well suited operating system for the development of this kind of system, since it is possible to change to the software and adapt it to fulfil the specific needs of each application.

One of the most referred is the Raspberry Pi (RPi), and it is generally regarded as a fine choice. However, authors also refer and use another board with similar capacities, the Beaglebone Black (BBB). When comparing the two products, having in mind the proposed solution requirements, and the details presented in table 4.1 are essential to substantiate the choice.

BBB provides more interface options, which is an important factor when expandability is a desired feature, as it allows the interface with external electronics easily. Another factor in favour of the BBB is the built-in PMIC that manages the board power sources, that can be supplied via three different inputs.

RPi is cheaper, however it requires a memory card to support the board OS, while the BBB has already 4GB of eMMC on-board flash storage, and still allows the usage of a memory card for data storage.

Table 4.1: Comparison between BeagleBone Black and Raspberry Pi 2 Model B.

	BeagleBone Black	Raspberry Pi 2 Model B
CPU Clock Frequency	1 GHz	900 MHz
Memory	512 MB	1 GB
Storage	4 GB eMMC, Expandable	Micro SD Card Required
GPIO pins	65	40
I2C bus	2	1
SPI bus	2	1
UART Ports	4	1
Analog Input	7	0
Power consumption	210-460 mA @ 5V	150-350 mA @ 5V

Due to all the facts presented so far, the Beaglebone Black seemed to be a better choice for this purpose, as it met the proposed system requirements and the main design criteria set for the main sensor node in the proposed architecture, but it still presents potential for future developments.

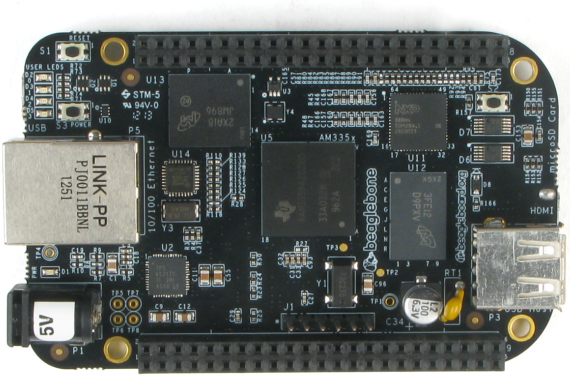

Figure 4.1: Beaglebone Black board. Extracted from [30].

The sensor network is based in the implementation of an I2C bus where all the sensors nodes will be connected, the main motives behind this choice have already been referred in section 3.2, but the expansion possibilities this protocol offers has been decisive.

The tiny sensor nodes are responsible for some very specific tasks, and taking them in consideration, it was vital to choose a microcontroller that fitted those requirements. Being so, a low cost and low power 8-bit microcontroller has been chosen, the ATtiny 85, produced by Atmel.

Its manufacturer describes this device as a *"High Performance, Low Power AVR 8-bit Microcontroller"*. This IC has embedded ADC (10-bit resolution), which is crucial for data acquisition. Furthermore it allows the definitions of timers and Interrupt Service Routines (ISR), that are fundamental to control the sampling frequency, and it is also compatible with I2C protocol.

After analysing its datasheet and reading about its application in several related works, it was clear that this microcontroller presented the desired features.

4.2 Main Sensor Node

In the proposed architecture, the main sensor node is the key part of the whole system, as it is responsible for a considerable numbers of tasks.

The main sensor node is based in a BBB, which is an embedded Linux board that fulfils all the requirements for this specific application, as already discussed in section 4.1.

The board used in this development has the Debian 7.9 distribution, a specifically customized version of Linux for the BBB, installed as its operating system. This Linux distribution has a built-in IDE, Cloud9, that can easily be accessed using Secure Shell (SSH) protocol, which was used to develop the applications responsible for ensuring the main sensor node operation.

C++ programming language was chosen for the development of the applications in charge of performing the main sensor node tasks, as this language allows object oriented programming, which is a suitable framework when programming this kind of devices.

In figure 4.2, it is possible to see a simplified architecture of the software implemented in the Beaglebone Black.

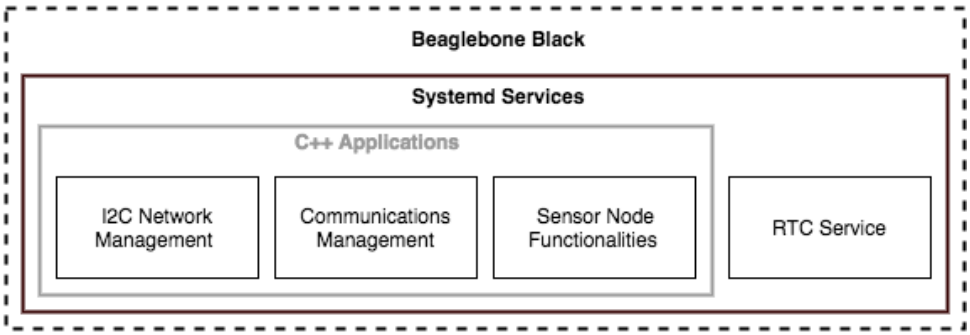

Figure 4.2: Main sensor node software architecture.

This software architecture contemplates three different applications:

- I2C Network Management – Manages the whole sensor network, sending sampling orders to the tiny sensor nodes and retrieving the collected data.
- Communications Management – Is in charge of transmitting the collected data log files to an external destination.
- Sensor Node Functionalities - Simple application to enable the utilization of the BBB as a sensor node itself.

The implementation of this C++ applications will be carefully discussed in sections 4.2.1 to 4.2.3.

Since this system is expected to have an autonomous and automated operation, several services were defined to start on the Beaglebone start-up boot. This services ensured the periodic and coordinated execution of the C++ applications that control the main sensor node tasks. These services are configured in *Systemd*, but further details will be given in section 4.2.4.

4.2.1 I2C Network Management

The main sensor node is responsible for managing the whole sensor network, as it is the master in the I2C network, and a C++ application has been developed to coordinate those operations.

The BBB has 3 I2C buses (I2C0, I2C1 and I2C2), I2C0 is an internal bus for HDMI control, and I2C2 is a general purpose I2C bus that is available by default.

The FDT (Flattened Device Tree) is a data structure that defines the hardware settings in the Beaglebone. If a second public I2C bus, I2C1, is needed, a DTO (Device Tree Overlay) can be loaded to the FDT in order to enable it.

The C++ application developed in this project can be used with both public I2C buses, but since the secondary I2C bus has to be redefined every time the board reboots or a service has to specifically programmed for that purpose. The setup used to test and validate the developed system will only use the bus which is active by default. However both buses have been tested in order to check if they were fully functional.

```
root@beaglebone:/var/lib/cloud9/Teste1# i2cdetect -y -r 1
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- 18 -- -- -- -- -- -- --
20: -- -- -- -- 24 -- 26 -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- 37 -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- UU UU UU UU -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- UU -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
```

Figure 4.3: Available I2C addresses in the active by default I2C bus.

Theoretically, an I2C network can have up to 128 devices connected in a bus, in a scenario where a 7-bit address is used. This number is constrained by the bus capacitance, which is dependent on the cables use in the I2C bus, a problem can be avoided using I2C repeaters throughout the bus.

In figure 4.3, it is possible to see that there only 112 addresses available for I2C interfacing, when the BBB I2C2 bus is checked. In this situation there are 4 tiny sensor nodes connected to the bus (0x18, 0x24, 0x26, 0x37). Then there are some addresses that are reserved by the Beaglebone, which are indicated in the figure with 'UU', since they are already being used for some internal drivers. One example of this is address 0x68 that is considered to be occupied as it is being used by the RTC service.

It is important to refer that Debian distribution maps the I2C2 bus as *'i2c-1'*, which explains why the default bus shown in figure 4.3 has been invoked with that instruction.

```
// @class I2CDevice
// @brief Generic I2C Device class that can be used to connect to
//  any type of I2C device and read or write to its registers

class I2CDevice{
private:
    unsigned int bus;
    unsigned int device;
    int file;
public:
    I2CDevice(unsigned int bus, unsigned int device);
    virtual int open();
    virtual int write(unsigned char value);
    virtual unsigned char readRegister(unsigned int registerAddress);
    virtual unsigned char* readRegisters(unsigned int number, unsigned int fromAddress=0);
    virtual int writeRegister(unsigned int registerAddress, unsigned char value);
    virtual void debugDumpRegisters(unsigned int number = 0xff);
    virtual void close();
    virtual ~I2CDevice();
};
```

Figure 4.4: Wrapper class for I2C device. This library source code is available in [31].

4.2.1.1 I2C Network Management Application Implementation

This application development was based in a C++ class developed by Derek Molloy in [31], that defined with its methods, the general functionalities associated with an I2C bus. This generic class can be adapted and employed to control several generic I2C devices. The used code is open source software and is available in *GitHub* in order to enable future works.

The class I2C device presented in figure 4.4 defines several methods, such as opening and closing the communications with a specific I2C slave, writing registers, read a single or multiple registers. To use this, it is only necessary to include a *<I2CDevice.h>* library in the I2C Network Management C++ application code.

In order to help the understanding of the developed C++ application, its algorithm has been detailed in figure 4.5.

This application must be able to coordinate a single I2C bus as its master, send the sampling orders to all the tiny sensor nodes connected to the bus, collect the sampled data, and then store that information in a specific directory.

The program starts by analysing a *.txt* file, *'SETUP.txt'*, where the system user defined the sampling settings for each tiny sensor node connected to the I2C bus.

After that, the application has all the orders stored in a linked list, and is ready to start sending sampling orders to the tiny sensor nodes using the I2C bus. For that purpose a function has been developed, *sendOrders()*, and its algorithm is detailed in figure 4.6.

After sending all the request to the tiny sensor nodes, the application enters in an idle mode for a period corresponding to the duration of the longer sampling order present on the order list. This safety period ensures that all the sampling processes are concluded before starting the collection of the sampled data. In section 4.3 some specific details will be explained about the tiny sensor node microcontroller ISR, that explain the motives behind this precaution.

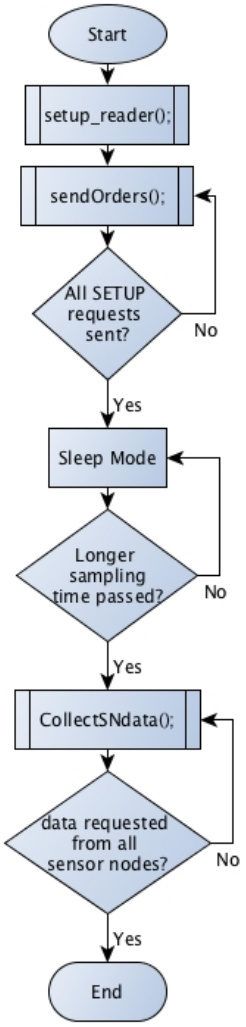

Figure 4.5: I2C Network Management C++ Application flowchart.

Then, the program will request sequentially each tiny sensor node the sampled data, and store that information in a specific log for each device. This process is done by *CollectSNdata()* function, and its operation is detailed in figure 4.6.

When the data from all the devices has been received, the program ends. This application can be programmed to be used periodically as explained in section 4.2.4.

The *sendOrders()* function is responsible for sending a sampling order to a specified tiny sensor node. When called, this function starts by constructing a string that will be sent to the specified device, this string contains the sampling settings that will trigger the data collection process in tiny sensor node, its composition is detailed later in section 4.3.2.

Afterwards the program transmits all the characters from the string, using the *writeRegister()* function, and when the transmission is successfully concluded, the application uses a time stamp function to log the exact time when the sampling process started.

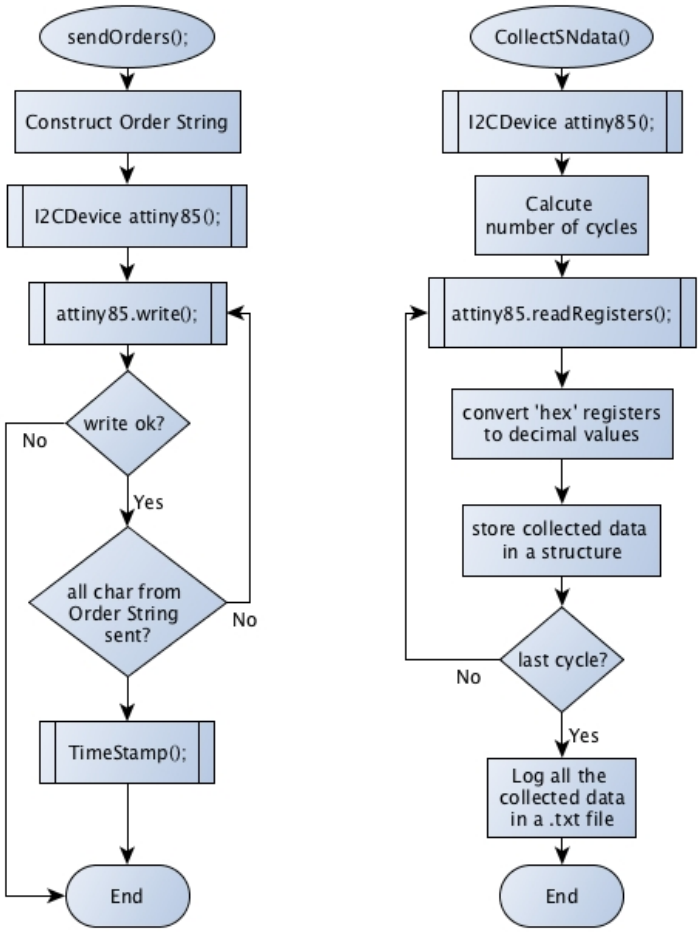

Figure 4.6: C++ functions algorithms from the I2C Network Management C++ Application.

CollectSNdata() is based in the utilization of the the *readRegisters()* function defined in the *I2CDevice.h* library, but as it only allows the retrieval of 16 bytes each time it is invoked, there is a need to calculate how many cycles are needed to collect all the all the samples requested in the sampling order. After receiving each 16 bytes there is the need to concatenate each 2 bytes and convert it to decimal base, to have the desired data and store it in a data structure. When all the samples have been have been retrieved from the specific I2C slave, the collected data is stored in a *.txt* file that is saved in more than one location for safety purposes.

4.2.2 Communications Management

Another important task to be ensured by this main sensor node comprehends the transmission of the gathered data to a desired destination. For this it is essential to have an Internet connection, that can be achieved by three different ways, LAN, Wi-Fi or a GSM connection.

Since the availability of the network depends on the location of the system, it was decided that it was better to develop a C++ application that did not depend on how the Internet connection was

obtained, and that could work whenever an Internet connection was available.

Therefore, a simple program has been implemented to send the log files via email, using a secure simple mail transfer protocol (SMTP) server, to transmit the information to a previously specified address.

For this purpose, it is possible to install a Linux package containing a program that performs the desired functionalities. 'ssmtp' is a Mail Transfer Agent (MTA) responsible for delivering a mail to a mail hub (SMTP Server), such as Gmail.

In the implementation of this system a Gmail account was set up, and after that some changes were made in the *'.conf'* file of the installed ssmtp application, such as inserting the Gmail account settings, and activate TLS encryption. It was also necessary to open port 587 in the BBB, which usually is the port used for outbound SMTP traffic, to allow the service through the BBB firewall.

Then this service was embedded in a simple C++ application, and its algorithm is detailed in figure 4.7.

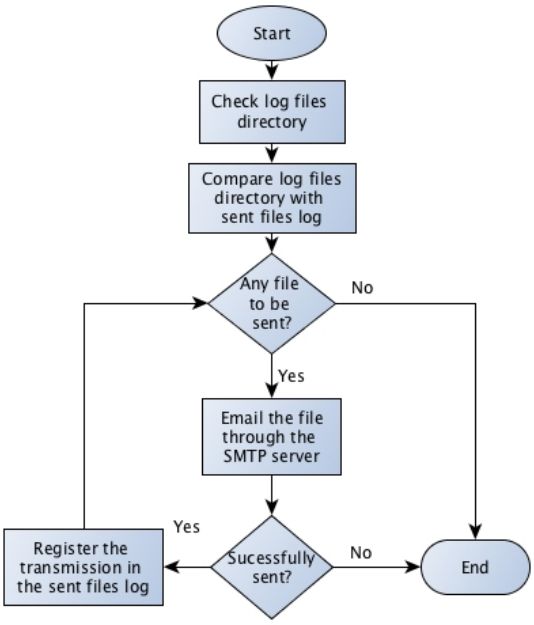

Figure 4.7: Communications C++ Application algorithm.

The program starts by checking the log files directory, then compares it with the sent files log, and if there is any file that has not been sent, it emails it through the SMTP server. If it is successfully sent, it checks again to send any file that may not have been sent yet. In case this communication fails, it means that at the moment there is no Internet connection, therefore the application ends its activity.

This application is designed to operate after the I2C Network Management application, ideally it sends the log files collected in the last time the system setup has been ran, but if there are previous log files that have not been sent, it also tries to upload that data.

4.2.3 Sensor Node Functionalities

This main sensor node has several responsibilities, and many are related with its operation as a gateway gathering information from other devices and transmitting it to an upper level.

However, it is also expected that it is able to collect local information as any other sensor node and store that information in the same directory as the other sensor nodes. It might be a temperature or humidity sensor inside the prototype box for safety purposes, or some other external sensor.

It can be argued that this sensors could be attached to a tiny sensor node and simply attached to the I2C bus, but it is of highly importance that the system can acquire data using different hardware resources. For that purpose, the BBB has 7 analog inputs that can be sequentially used as ADC. This feature is not enabled by default but can be activated if a DTO (Device Tree Overlay) is loaded in the BBB FDT (Flattened Device Tree), in a similar process as the one used to activate the secondary I2C bus described in subsection 4.2.1.

In the development of this prototype, a simple program was used in order to collect the temperature measurements in the specific place where the BBB was located to validate this possibility, and demonstrate this application is possible. This program algorithm is detailed in figure 4.8, and its test and validation is described in 5.4.

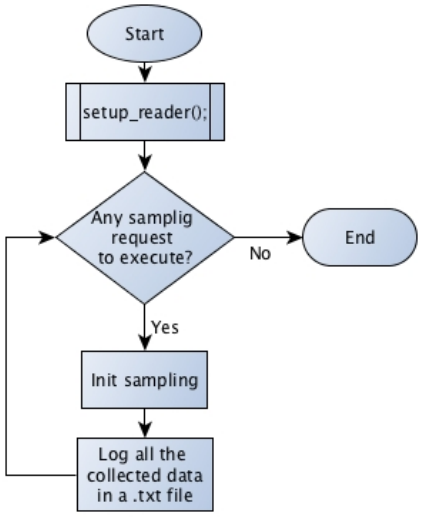

Figure 4.8: Sensor Node Functionalities C++ Application algorithm.

Like the application developed for the I2C network management, this program starts by analyzing a setup file, where the system user defined the analog pins where the desired sensors have connected, the numbers of samples and the sampling frequency associated with each sensor.

Then the sampling processes are executed sequentially, and its data is stored in data log files in a specific directory of the BBB that might be the SD card.

This application is to work in parallel with the I2C Network Management application, as explained later in section 4.2.4.

4.2.4 Beaglebone Autonomous Operation

As already mentioned, this system is expected to have an autonomous operation, with several services starting its operation on the Beaglebone boot. These services are responsible for ensuring the periodic and coordinated execution of the C++ applications that control the main sensor node tasks.

In the Debian distribution running on the board, *Systemd* is a background process, a daemon that manages other daemons. It is the first daemon process to be executed on boot, and it is the last process to finish, before the board shutdown. *Systemd* is responsible for managing several services that control numerous processes in the BBB, as it allows parallel execution of services [32].

When setting up the new services that must be initialized on the board boot, there are some proceedings that must be executed. The executable files resultant from the development of the C++ applications must be placed in a specific directory of the BBB, that will serve the working directory for the services that control the running of each application.

Since this main sensor node is the coordinator of a network responsible for collecting data over time, it is essential that the BBB never loses track of time, therefore a simple service must be defined to request the RTC time at boot and correct the BBB internal time. Then for each service a unit file (*.service*) must be programmed as it encodes the details about the service, it must be written using a specific syntax to define the service settings.

```
[Unit]
Description=MSS dissertation Main Sensor Node operation
After=rtc-ds3231.service

[Service]
Type=simple
WorkingDirectory=/var/logServices
ExecStart=/var/logServices/ServiceTest
SyslogIdentifier=SYSTEM_operation_mss
Restart=always
RestartSec=1200s

[Install]
WantedBy=multi-user.target
```

Figure 4.9: Unit File example.

Figure 4.9 presents an example of a unit file implemented in BBB, and it is possible to see how several settings are defined. The service starts after the RTC Service, its directories are specified, and the service is defined to restart each 30 min.

In figure 4.10 it is possible to see a very simple diagram, to help the understanding of the sequence executed by the main sensor node in its normal operation. After the system boot, when *Systemd* is up and running, and its most important services are already running, a simple service is executed, the RTC Service, to synchronize the BBB time with the external RTC.

Then, the main sensor node starts its cyclic operation. I2C Network Management and Sensor Node Functionalities applications are executed, and when both have finished its execution, the

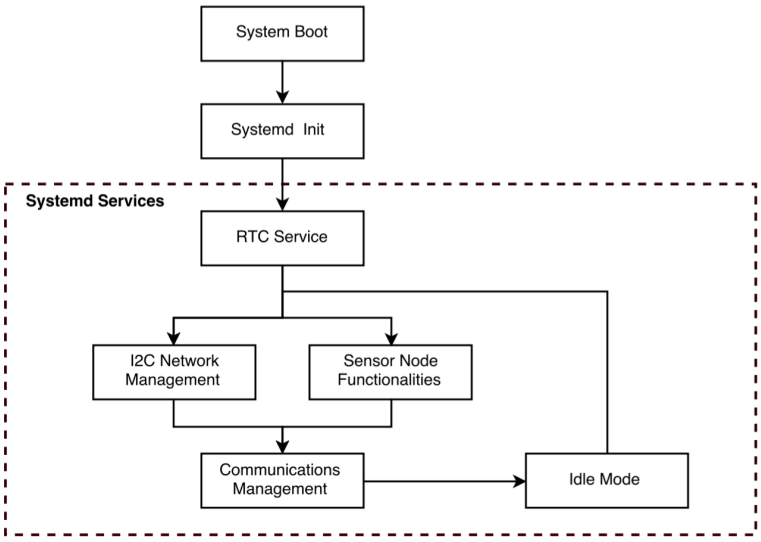

Figure 4.10: Services Execution Sequence.

Communications Management application is executed, followed by an idle time, before staring this process again.

The execution of this services frequency is set up in the unit files where all this processes are set.

4.2.5 Summary

The developed applications were designed to be setup-orientated, operating with little intervention from the system users, and to be capable of working autonomously for long periods of time without any human interference.

The I2C Network Management application is a vital part in the presented architecture, as it is responsible for acquiring the great majority of the data that should be collected by the main sensor.The implemented program is able to send sampling orders and request the the collected data from 1 to 200 I2C devices that might be connected any of the I2C buses.

As mentioned, the Communications Management application was designed to work whenever a Internet connection is available, and with this precaution the developed application is not restricted by the hardware device that ensures the BBB connection to the Internet.

The BBB OS allows the deactivation of several hardware features that are not needed for the board operation as main sensor node, such as disabling the HDMI cape or the USB port. That reduces the BBB power consumption, however as this is still a prototype in development, it is safer to have all the resources available.

4.3 Tiny Sensor Nodes

Each tiny sensor node has to be able to process sampling orders sent by the main sensor node via I2C and to transmit the sampled data when the I2C master requests it.

For this, it has to be able to read the sensors, do some signal conditioning if needed, make the ADC conversion, store the collected data, and be part of an I2C based network.

This section starts by discussing the IR sensor and its signal conditioning circuit, then focuses on the tiny sensor node microcontroller and the development of its firmware. Finally, in the last subsection the developed tiny sensor node board is presented.

4.3.1 IR Sensor and Signal Conditioning Circuit

One of the main requirements of this system consisted in its capacity to sample the heartbeat frequency using an IR sensor to detect variations in the reflection of light due to the heartbeat.

CIBIO/InBIO researchers published a paper where they propose "An improved non-invasive method for measuring heartbeat of intertidal animals" [4] that essentially consists in the utilization of an IR sensor with an IR emitter and an IR detector to detect light variations caused by the changes in the circulatory structure resulting from the heartbeat.

An IR sensor allows that this changes are transduced into changes in electric current, that can be measured and sampled, if a proper conditioning circuit is used.

For that purpose the circuit presented in figure 4.11, which is a modified version of the circuit presented in [33], was used and when tested it achieved the same objectives.The employed sensor was CNY70 [34] from Vishay Semiconductors, which is an IR sensor that comprise the required functionalities for this application.

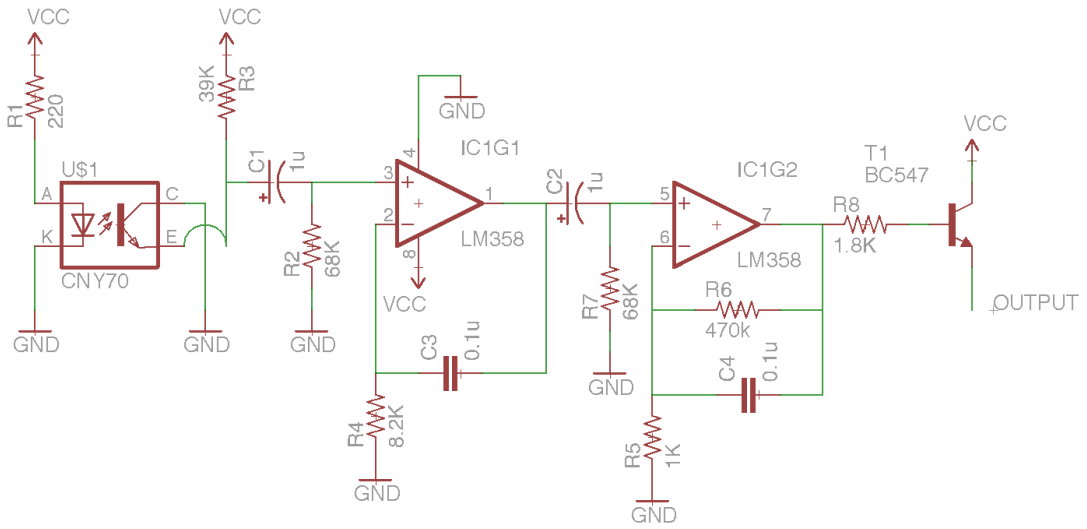

Figure 4.11: Signal Conditioning Circuit for the IR sensor.

The light detector present in the IR sensor is phototransistor, that work as switch, whenever the detector senses light, it allows the passage of current. This current variation causes a voltage drop at the input of the amplifying circuit.

This input signal is then filtered by two consecutive non-inverting operational amplifiers, that are responsible for filtering noise, amplifying the voltage variations that to correspond to the heartbeat peaks.Then the filtered signal is amplified by a NPN transistor.

This circuit has been prototyped in a breadboard. In order to test and validate the use of this conditioning circuit several tests were made using a fingertip as a contact point to collect heartbeat data to be processed by the conditioning circuitry.

The obtained results were plausible and satisfactory, as they have shown that this method supplies a signal that clearly detects the heartbeat peaks.Therefore it was decided that this circuit would be included in the tiny sensor node board to be developed.

4.3.2 Microcontroller

As already detailed in section 4.1, the ATtiny85 was the chosen microcontroller to control the tiny sensor node operation.

For the development of the prototype versions the DIP package version was selected as it allows an easy prototyping in a breadboard, and it also facilitates its programming.

The Arduino IDE was chosen as IDE for the development of the microcontroller firmware, since it is possible to set up this microcontroller hardware settings, using a software package developed by David A. Melis available in [35] that ensures that the developed code will be a program that works properly in the ATtiny85 after being compiled. Another important factor in this choice is related with the considerable number of open source libraries, that facilitate the development of the firmware.

For this purpose it was necessary to set up an Arduino UNO board as an “in-system programmer” (ISP), in order to use it to upload the tiny sensor node firmware to the ATiny85 flash memory. For that an Arduino shield, that can be seen in figure 4.12, has been assembled to facilitate the development and testing of the μC firmware.

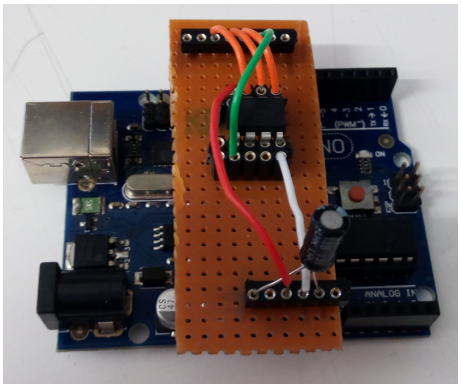

Figure 4.12: Arduino shield used to flash the ATtiny85 microcontrollers.

4.3.2.1 I2C in the Microcontroller

The key feature in the development of the tiny sensor nodes firmware was related with its capacity to be connected to an I2C bus as I2C slaves, in order to receive the parameters that regulate its functioning.

The ATtiny85 is not prepared to use I2C communications by default. However it has a Universal Serial Interface (USI) module, that can be set to enable the Two Wire mode (TWI), which is compatible with the I2C protocol since the USI provides the hardware resources to enable this kind of communication.

Atmel, this chip manufacturers, published an application note [36] regarding the utilization of USI module of its chips as an I2C Slave, and alongside this publication some source files, including a library with several functions developed in C, that ease the use of this USI module with the I2C protocol.

These routines developed by Don Blake, and supported by Atmel, have been adapted for being used in the Arduino IDE and with the ATtiny processors, and that library which is available in [37], have been used in the development of the firmware present in the μC.

The methods defined in this *<TinyWireS.h>* library, defined several functions that alongside the information received from the main sensor node made the I2C bus operational for the expected purposes.

```
// Public Methods //////////////////////////////////////////////////////////////

void USI_TWI_S::begin(uint8_t slaveAddr){ // initialize I2C lib ... }

void USI_TWI_S::send(uint8_t data){  // send it back to master ... }

uint8_t USI_TWI_S::available(){ // the bytes available that haven't been read yet ... }

uint8_t USI_TWI_S::receive(){ // returns the bytes received one at a time ... }

// sets function called on slave write
void USI_TWI_S::onReceive( void (*function)(uint8_t) )
{ ... }
// sets function called on slave read
void USI_TWI_S::onRequest( void (*function)(void) )
{ ... }

void TinyWireS_stop_check()
{ ... }
```

Figure 4.13: Methods defined in *<TinyWireS.h>*, library.

The Interrupt Service Routines (ISR) are high priority tasks that can be triggered by internal or external events, and when triggered interrupt the normal flow of the program to perform specific tasks defined inside the ISR. At the end of the ISR the program resumes its normal operation.

When the μC program is using ISR there are some precautions that must be taken into account, such as declaring the variables that are used both in a ISR function and outside that function, as *volatile* to tell the compiler that variable can be changed at any moment.

There are two functions defined in the <*TinyWireS.h*> that are fundamental for this program, as they are associated with the USI START interrupt of the microcontroller, that interrupts the program flow whenever it detects an I2C start condition. The methods in question are the following:

- *TinyWireS.onRequest()* - This function detects a data request from a Master and calls another function that is responsible for sending the request information to the Master.
- *TinyWireS.onReceive()* - When the I2C slave receives a transmission from a master this function calls another function that deals with the received data.

4.3.2.2 Sampling Process in the Microcontroller

Another important feature in the development of the tiny sensor node firmware is related with the μC capability of sampling with a specific frequency to ensure the quality of the data acquisition process.

For this purpose, the ATtiny85 has the *Timer/Counter1* which is a general purpose *8-bit Timer/-Counter* module, that allows a precise scaling of the timers.

Timer1 uses the μC clock frequency as its clock time base, that can be divided by a specified value, the prescaler. Then a counter can be set to increment each time the specified timer passes. When the counter reaches a specified comparator it triggers an external interrupt and clear the counter value, initiating that way another count. This is the CTC (Clear on Timer Compare) mode.

As already referred, this μC has several built-in ISR that can be triggered by internal events. For this purpose TIMER1 COMPB has been used, which is a ISR that fires when Timer/Counter1 Compare Match B matches the specified comparator value. With this method, the sampling can be done inside the ISR and as this interrupt occurs periodically it ensures the signal can be sampled with a desired frequency.

Using the ATtiny85 with a clock frequency of 1 MHz, the sampling time is calculated with the following expression:

$$SamplingFrenquency = \frac{uCClockFrequency}{(TimerPrescaler * Comparator)}$$

The tiny sensor node receives these parameters in a string that is sent by the I2C master to trigger the sampling process here described.

'c'	samples			presecaler	comparator			selector	<empty>		'f'
0	1	2	3	4	5	6	7	8	9	10	11

content
position

Figure 4.14: Sampling order sent via I2C form the master to a specific slave.

In figure 4.14 it is possible to see how that string is composed. The first and last elements are specific characters set for internal processing purposes, then it contains the number of samples to be collected, the timer1 prescaler, the comparator, a sensor selector byte, and 2 empty bytes that can be used in future developments.

One of the known limitations of the ATtiny85 is related with its memory. The program developed for the tiny sensor node requires several variables for its operation, which occupy almost 100 bytes, leaving only 400 bytes of the SRAM memory for storage of the collected data,before transmitting them to the I2C master whenever it is requested.

Since each sample needs 2 bytes as it needs to be stored in an *int* variable, because the sampling is realized with 10-bit resolution, this means each sample can be a integer ranging form 0 to 1023.

4.3.2.3 Firmware Implementation

In order to help the understanding of the firmware developed for the tiny sensor node, its algorithm has been detailed in a flowchart present in figure 4.15.

When the microcontroller is powered and boots, it starts by initializing several features, and activates the I2C functions.

Then it enters an idle mode where it periodically checks if there is any valid string available. If there is any, the program extracts the sampling details from it, configures the timers and starts the sampling process. When all the desired samples have been collected, the ADC process is stopped and the program returns to the idle mode.

It is important to note that this I2C library for the ATtiny85, whose functions are initialized at the beginning of this program, uses one of this μC interrupts, the USI START, so whenever the slave receives contact from the I2C master, it stops its activity to send or receive data, with *requestEvent()* or *receiveEvent()* functions respectively.

The algorithm of both this functions is detailed in figure 4.16. *RequestEvent()* is responsible for sending 16 bytes to the master containing the information stored in 8 *int* variables, which correspond to analog pin reads sampled with 10-bit resolution that can be an integer value from 0 to 1023.

ReceiveEvent() is called when the μC receives a transmission form the I2C master, that corresponds to a string containing a sampling order. This function receives and analyzes each byte, to construct the order string.

The third function algorithm present in figure 4.16, details the ISR responsible for the sampling process. This ISR is triggered when timer1 matches a comparator defined in the received string, then it collects a samples on the specified pin, store it in a data structure, and increments a counter responsible for controlling the number of samples.

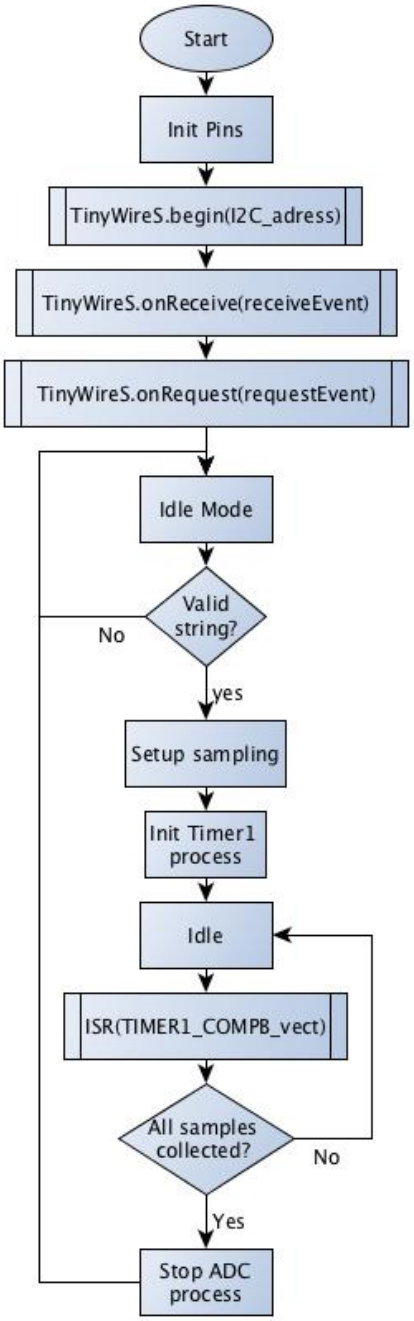

Figure 4.15: Tiny Sensor Mode firmware algorithm flowchart.

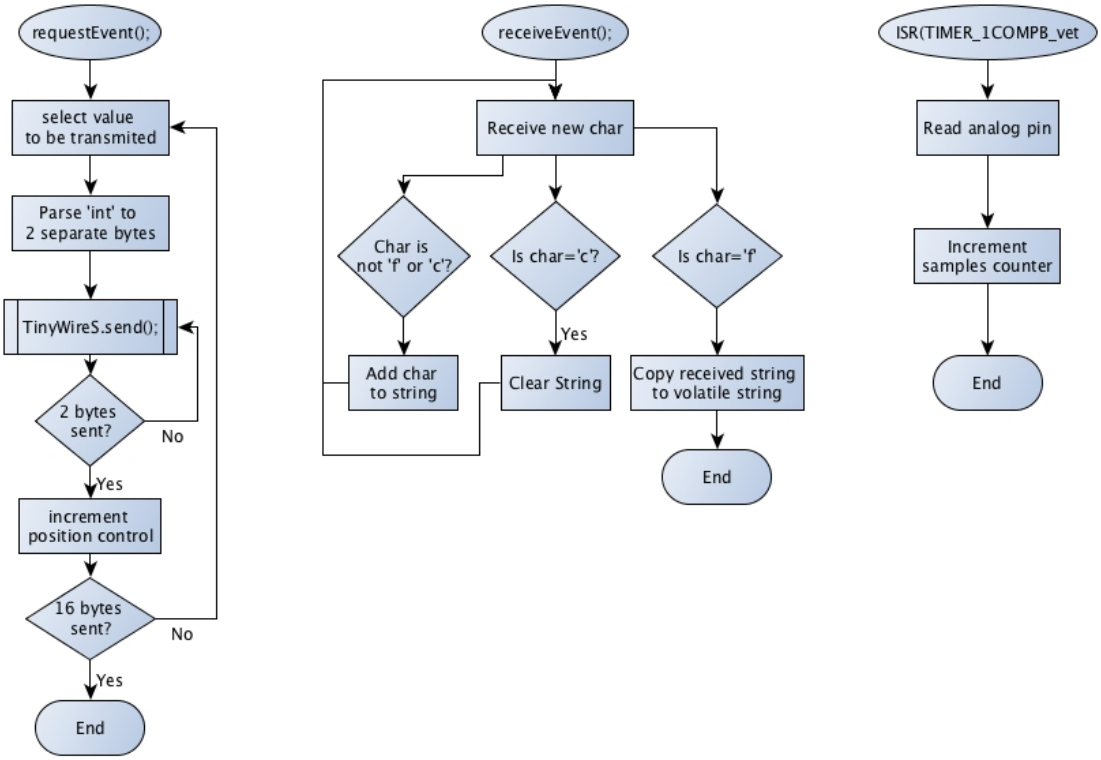

Figure 4.16: Flowchart of some functions from the tiny sensor node firmware.

4.3.3 Tiny Sensor Node Board

Finally, after developing the firmware for the microcontroller and validating the conditioning signal circuit for the IR sensor, there was a need to develop a low dimension print circuit board (PCB) to serve as a tiny sensor node that could be waterproofed using an epoxy resin in order to be used to collect underwater data.

The ATiny85 is an 8-pin microcontroller, which has three ADC pins. However one of them is occupied with the I2C communications, therefore only two sensors can been connected to this integrated circuit. Due to these specific application constraints, one of the pins is dedicated to the IR sensor but the other one is available to be connected to an analog sensor whose voltage variation does not exceed the voltage supplied to the microcontroller, as that is the reference voltage for the ADC process.

The developed board has an 8-pin DIP socket where the ATtiny85 can be connected, as shown in figure 4.17. A SOIC package is a better choice for a board with these requirements, however, as it was developed in parallel with the tiny sensor node firmware development, the DIP version was used in the prototype boards. This allowed that the microcontroller package was connected to and removed from the tiny sensor board throughout the development of the firmware.

In figure 4.18 it is possible to see the results obtained when using the board conditioning circuit to sample the heartbeat of an adult male. It was used with the CNY70 sensor in direct contact with

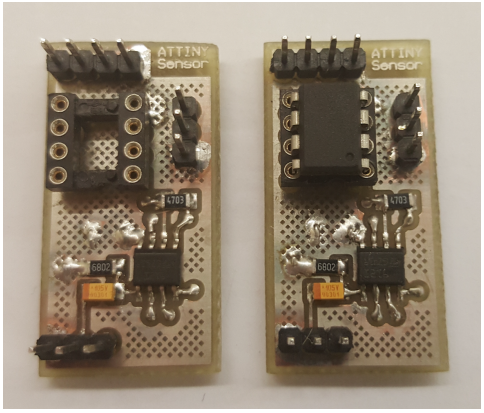

Figure 4.17: Tiny Sensor Nodes.

a fingertip. The circuit was capable of detecting the peaks resulting from the effect the heartbeat has in the circulation of blood, and measured signal frequency of 1.11Hz that according to medical literature is a normal and a expected value for this variable. The presented test is one of many tests, that allowed the validation of the signal conditioning circuit.

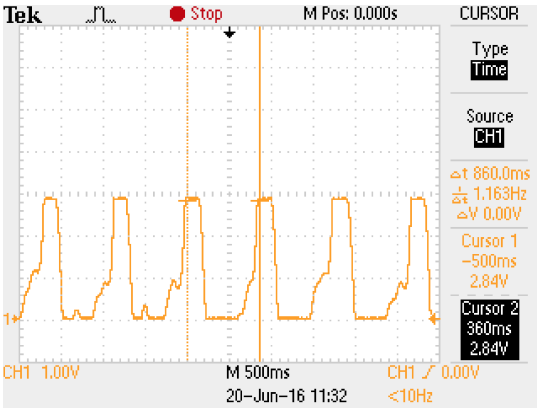

Figure 4.18: IR Sensor conditioning circuit output using the PCB circuit.

This circuit has been tested with both 5V and 3.3V, however as the I2C bus operates at 3.3V the tiny sensor node boards will operate at that voltage level.

As shown in figure 4.19, the tiny sensor nodes have been waterproofed with an epoxy resin, in order to test this devices in the environment they have been designed for. Its test and validation is carefully detailed in chapter 5.

4.3.4 Summary

Regarding this implementation process there are several relevant questions that should be discussed. The development of the tiny sensor nodes above presented is one possible application of

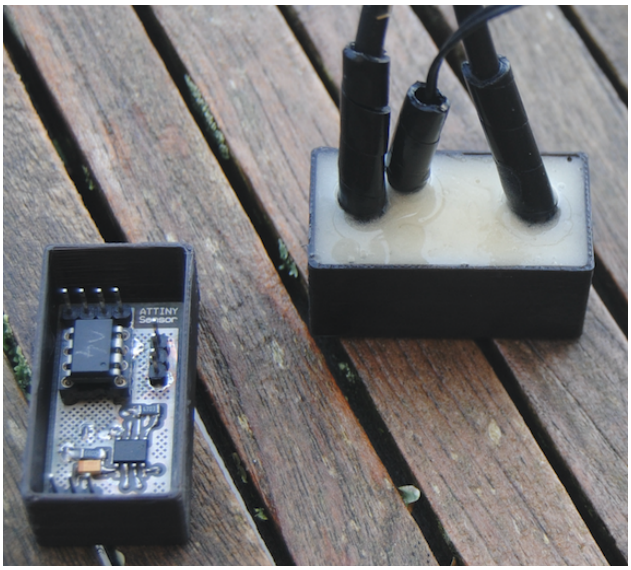

Figure 4.19: Waterproofed Tiny Sensor Node.

the proposed solution. In this situation, everything was orientated for this board to be used with the IR Sensor that required the signal conditioning circuit presented above.

However, this microcontroller can be used in a completely different situation with the developed firmware, e.g. in a smaller board if a different packaging of the ATtiny85 is used. The firmware has been designed so that when an I2C master transmits the specified string to a specific slave, it executes the sampling order considering the designated pin.

If an SOIC package is employed and the analog sensor does not require any specific circuitry for the signal to be measured by the microcontroller, the tiny sensor node board can be small sized, as its major component dimensions will have 5.35 x 5.4 x 2.16 *mm*.

It is important to discuss the limitations associated with this solution. The SRAM memory of the ATtiny85 can store a total 512 bytes, but some of this memory is taken with variables essential in the operation of the tasks the device is responsible for assuring. Therefore, it must be said that for now the tiny sensor nodes are limited to sample a maximum of 180 samples that can be be extended to 200 samples, if the firmware code is optimized.

Another issue that was not discussed in this chapter is related with the implementation of the I2C sensor network in the particular application this solution was proposed, as it requires the installation of the proposed I2C sensor network using long (1-3m) cables. Transmitting I2C data using long communications cables is a delicate issue, as the bus maximum capacitance is under 400pf. However, as NXP Semiconductors, the company who invented this protocol, ensures that this limitation can be overstepped using I2C repeaters throughout the bus, as described in their application note [38], it was assumed this would not be an issue later on when implementing the sensor network

The tiny sensor nodes have been subject to several tests, which will be shown, detailed and discussed in chapter 5.

Chapter 5

Tests and Results

This chapter is dedicated to testing of the developed system in order to validate the proposed solution.

The following tests consist in a series of demonstrations of the features described in Chapter 4. For that purpose some experimental setups have been defined to emulate possible situations in which the system might have to operate.

Each test aims to highlight the results of different features, but as the system is operated as a whole, the main sensor node operation, the I2C network and the tiny sensor nodes, are always validated.

5.1 Test 1 - Sinusoidal Wave

In this test a sinusoidal signal is sampled by a tiny sensor node. Figure 5.1 shows the setup defined for this experiment.

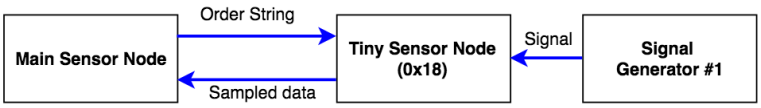

Figure 5.1: Test 1 setup.

A signal generator provides the sinusoidal signal with a frequency of 102 Hz, that is represented in figure 5.2. Then, the tiny sensor node received a sampling order, which contained the information to trigger the sampling of 80 samples with a frequency of 1000 Hz, this meaning one sample each *ms*.

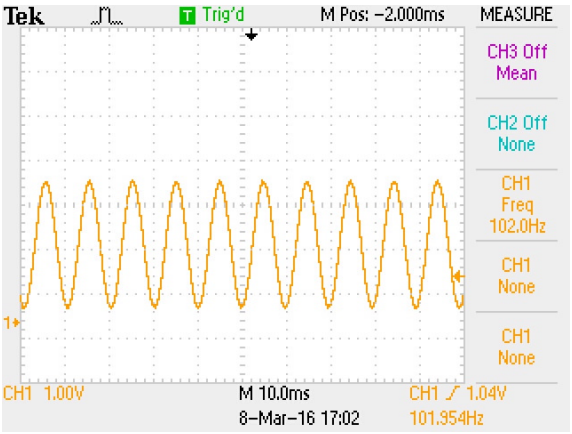

Figure 5.2: Generated signal to be sampled by the tiny sensor nodes.

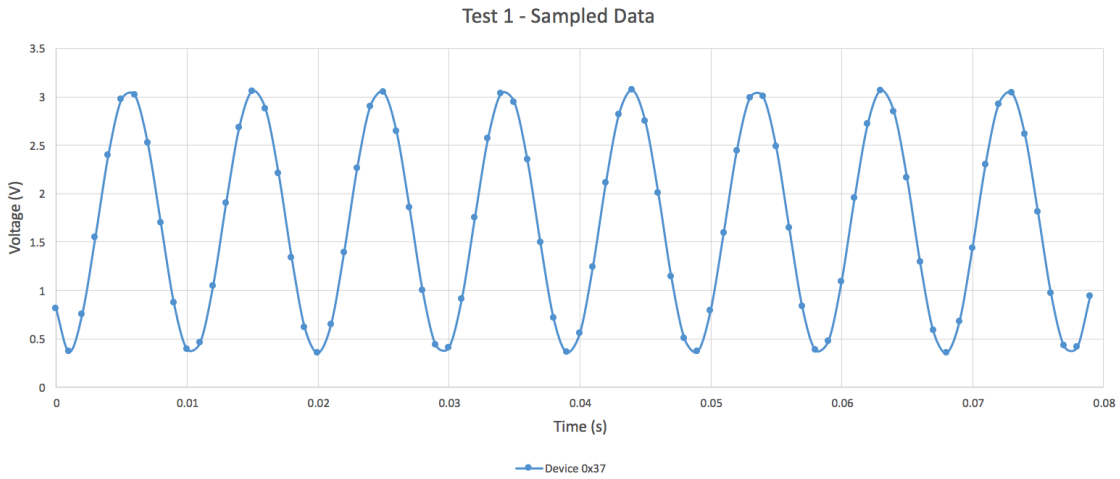

Figure 5.3: Test 1 Results.

The sampling process results present in figure 5.3 are highly satisfactory since the sampled data correspond without a doubt to the signal supplied by the signal generator, and the sampling process occurred exactly as defined in the setup defined in the main sensor node.

5.2 Test 2 - Human Heartbeat

This test aims to validate the tiny sensor node board and its signal conditioning circuit, as well as its sampling capabilities. The diagram presented in figure 5.4 shows the setup used to realize this test.

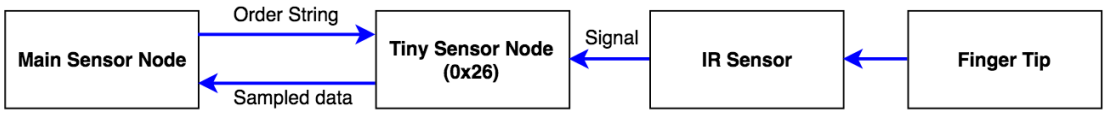

Figure 5.4: Test 2 setup.

In figure 4.18 it is possible to see the results obtained when using the board conditioning circuit to sample the heartbeat of an adult male. It was used with the CNY70 IR sensor in direct contact with a fingertip. The circuit was capable of detecting the peaks resulting from the effect the heartbeat has in the circulation of blood, and measured signal frequency of 1.11Hz that according to medical literature is a normal and a expected value for this variable. The presented test is one of the tests, that allowed the validation of the signal conditioning circuit.Here the heartbeat signal has been sampled with a sampling frequency of 19.93 Hz.

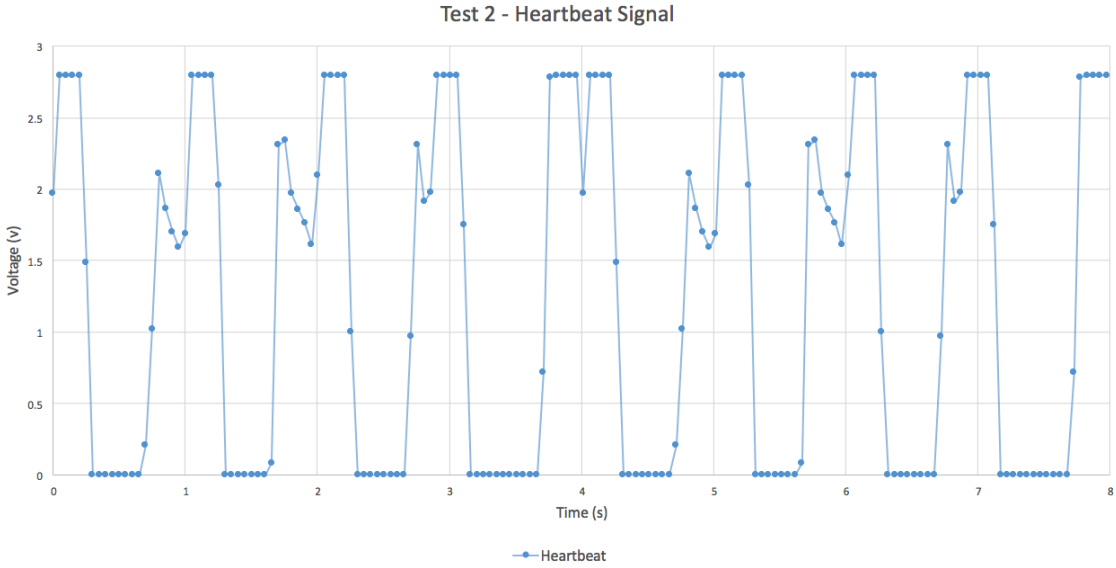

Figure 5.5: Sampled heartbeat signal of human adult male.

The results obtained are coherent, as the circuit clearly detects the heartbeat peaks and the collected value can be processed in order to calculate the heartbeat frequency. This validates the tiny sensor node, as it executed the sampling process as expected and the signal was properly processed before the ADC conversion.

5.3 Test 3 - I2C Sensor Network

This test's main objective was to collect data using different sensor nodes, and therefore demonstrate the system sends the sampling order to the correct tiny sensor node, it performs the sampling correctly, and retrieves the collected data when it is requested.

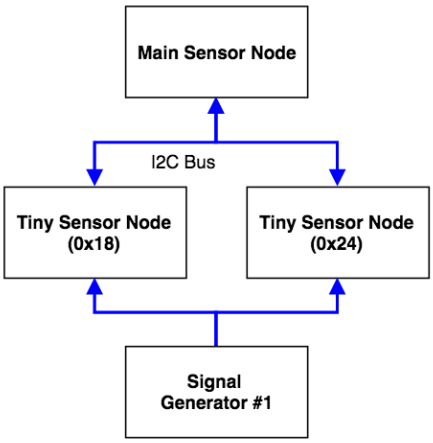

Figure 5.6: Test 3 setup.

In figure 5.6 one can see the experimental setup used to validate the I2C network operation. A signal generator supplying a sinusoidal signal with a frequency of approximately 35 Hz is connected to two different tiny sensor nodes. Then two consecutive and equal sampling orders are sequentially transmitted using the I2C bus. The sampling order requires 120 samples to be collected with a sampling frequency of 500 Hz.

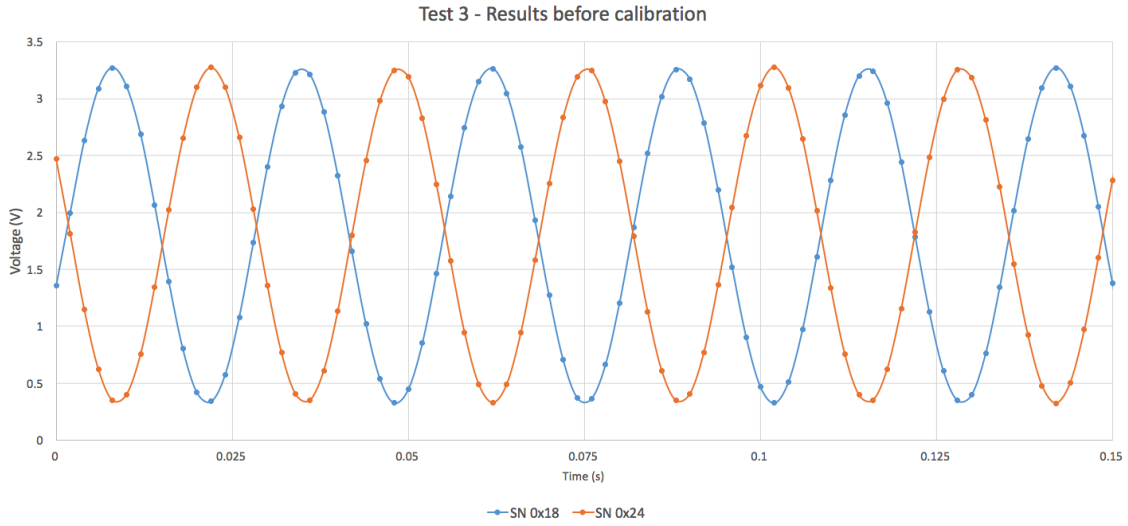

Figure 5.7: Pre-Calibration Results.

The obtained results with this test, presented in figure 5.7, show that both tiny sensor nodes have properly sampled the signal in amplitude. It is also possible to see that there is a time difference from the provided signal and the obtained results. This can easily be explained by the fact, that the main sensor node sends the sampling orders sequentially and it can only speak with one tiny sensor node at the time, as I2C is a serial communications protocol and its bus handles

only one communication at the time. Therefore it makes perfect sense.

However this temporal difference between the start of each sampling process can be measured by the main sensor node, and that time variation value can be included in the log files resultant from the sampling process.

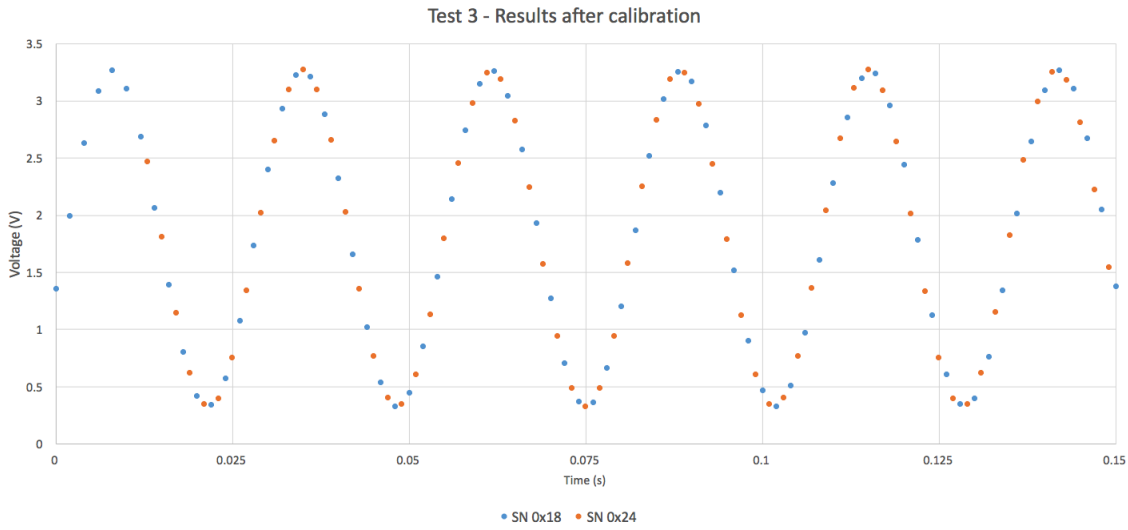

Figure 5.8: Test 3 Results.

Using that information it was possible to calibrate the sampled data, and in figure 5.8 it is possible to see that the results of each sampling process correspond to the same signal, that was supplied by the signal generator. It is possible to see that the signal resultant from the data sampled by *SN 0x24* starts sensibly *11ms* later, which corresponds to the time interval indicated by the main sensor node. However, the measurement of the interval between the start of the sampling process in each tiny sensor node still presents an error of sensibly *1ms*.

5.4 Test 4 - Temperature Measurement

Finally, a test was setup to have the system operating autonomously for a considerable period of time.

This setup, represented in figure 5.9, uses two temperature sensors. One is connected directly to the BBB board, and the other is connected to an underwater tiny sensor node. That way it is possible measure the water temperature and the air temperature in the local the system is operating.

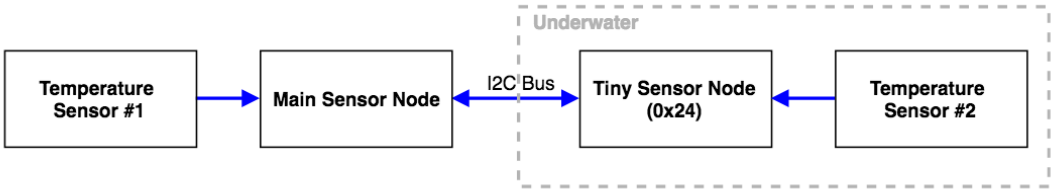

Figure 5.9: Test 4 setup.

The sensor responsible for measuring the air temperature, had a a sampling order defined in the main sensor node, to be executed by the Sensor Node Functionalities C++ application, that defined that the system was to collect 10 samples in 5 seconds (2Hz). With this data the main sensor node calculated the temperature average value at the time, and stored it a log file.

Regarding the measurement of the water temperature, a similar sampling order was sent to the desired tiny sensor node, that executed the data collection, and retrieved the data when requested. That information was stored in a specific log file.

Then, the periodicity of both services that assures the cyclic execution of the Sensor Node Functionalities and I2C Network Management C++ applications are set to start each 10 minutes.

As it is possible to see in figure 5.10, the tiny sensor node used in this test was waterproof and was placed inside a container filled with water.

In figure 5.11 it possible to see the results of this test. The BBB has sampled the outdoors water and air temperature during a period of 57 hours. The obtained results are perfectly plausible, taking in consideration the local environment constraints.

This test validates several features such as: the sensor node functionalities of the main sensor node, the underwater operation of the tiny sensor nodes, and the BBB autonomous operation for a long period of time, which is configured with the services defined in *Systemd*. This means the C++ applications responsible for the sampling processes defined in this test were running less than 30s in each 10 minutes.

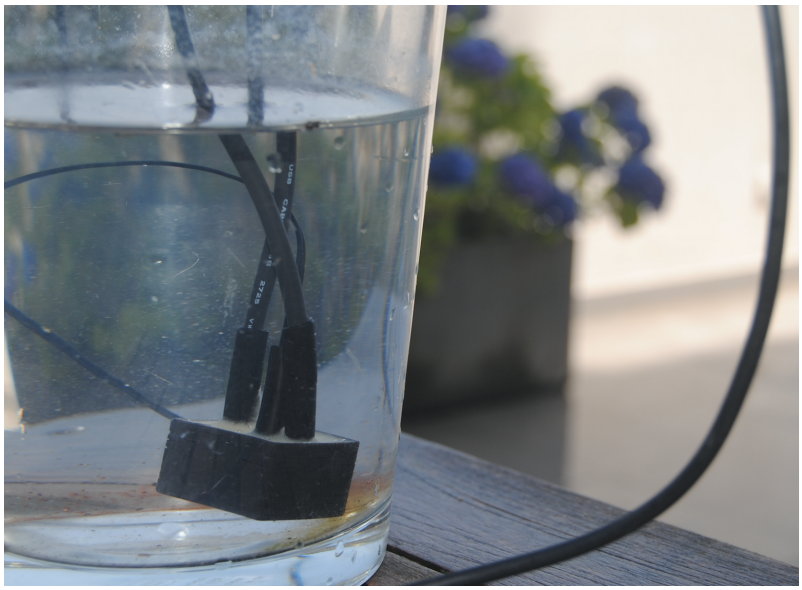

Figure 5.10: Underwater Tiny Sensor Node.

5.5 Summary

As already referred in section 3.1, the developed of this system, used a Systems Engineering approach, and in that methodology the test and validation of the system is essential in order to evaluate if the features and architecture proposed in this MSc dissertation are an interesting solution to solve the problem at hand.

These tests were used to exhibit the utilization of the developed features presented on 4, in order to validate its success. It was possible to confirm that the system is able to operate autonomously for a considerable period of time, execute the defined sampling orders and collect underwater data.

Several others features could have been demonstrated in this chapter. However, it was preferable to focus on the data collection capabilities of the system, since it was its most important task. The three key parts in that process, the main sensor node, the I2C network, and the tiny sensor nodes, have been tested and validated for future utilization.

It is important to highlight that the presented results are only successful experiments, but the failed ones have been more important as they were fundamental throughout the development of the prototype.

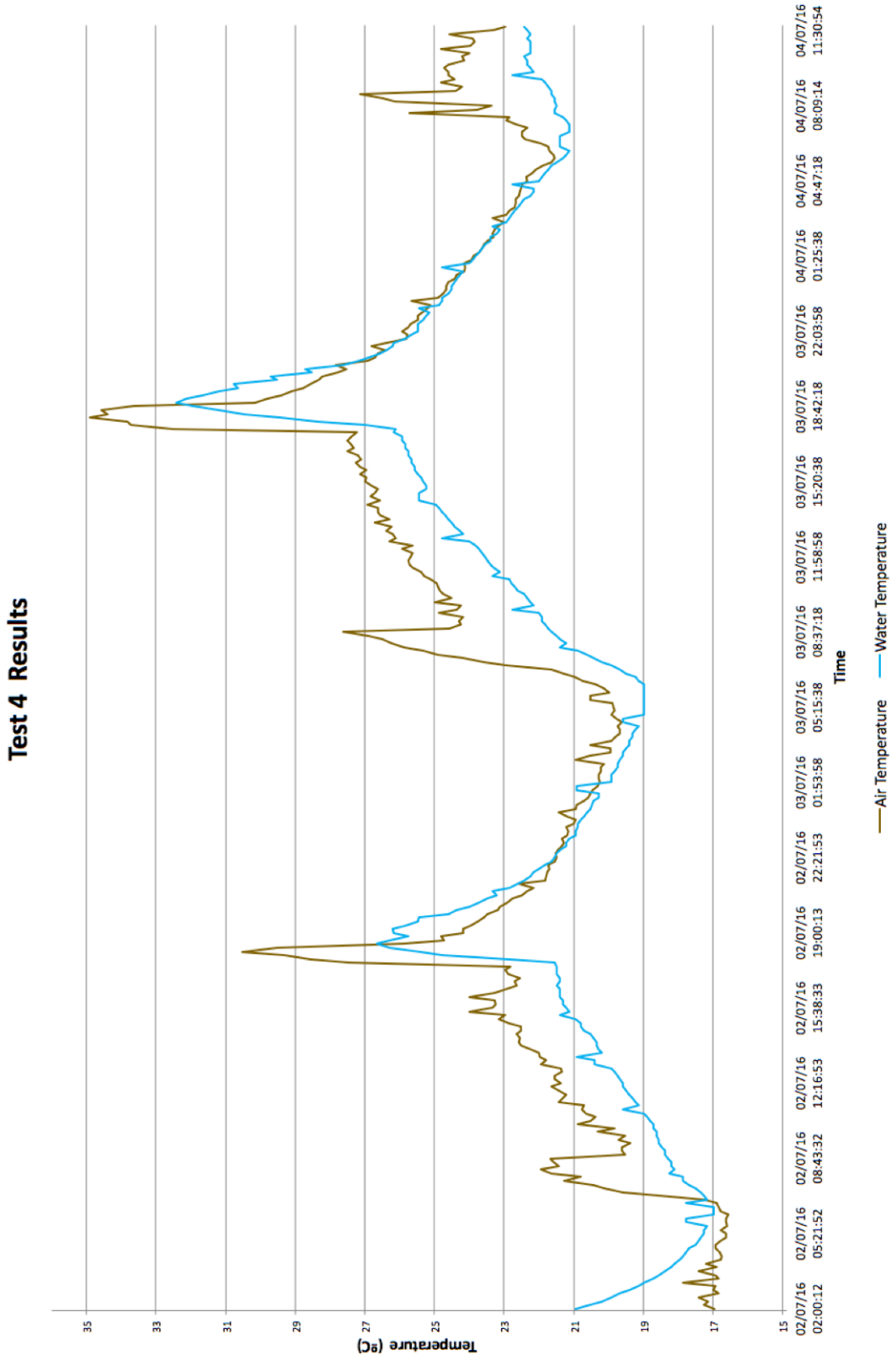

Figure 5.11: Test 4 Results.

Chapter 6

Conclusions and Future Work

Taking the preceding chapters into account, it is now necessary to analyze the final results of the all work developed.

This dissertation had two main objectives, the development of a solution for a Remote Supervision System for Aquaculture Farms, that must contemplate the functionalities of both an autonomous data logger and a remote monitoring system, and the implementation of a prototype that allowed a practical validation of the proposed solution.

Regarding the first main objective, a consistent architecture was proposed in chapter 3, as the proposed solution contemplated a modular, extensible and reconfigurable system that can be applied in the particular context this dissertation is orientated for, but can also be easily adapted for other purposes.

Then the second main objective was the practical implementation of a prototype, and the test and validation the implemented features. That process, presented in chapter 4, focused on the development and test of the core features presented in the proposed solution's architecture, the different sensor nodes and the communications between them.

The developed I2C Network Management application is able to send sampling orders and request the the collected data from 1 to 200 I2C devices that might be connected to different I2C buses. As mentioned in section 4.2.2, the Communications Management application was designed to work whenever a Internet connection is available. This precaution was taken, so that the developed application is not restricted by the hardware device that ensures the Beaglebone Black connection to the Internet.

Another requirement that has been fulfilled is related with the low interaction the system needs to operate. The developed application's execution is only dependant on specific setups that can be reconfigured, but can operate autonomously without any human intervention.

The developed tiny sensor nodes were customized for a specific sensor and its conditioning circuit, as well as another arbitrary analog sensor. However, this microcontroller can be used with the developed firmware, in a smaller board if a different packaging of the ATtiny85 is used. The firmware has been designed so that when an I2C master transmits the specified string to a specific slave, it executes the sampling order considering the designated pin.

The major challenge present in this MSc dissertation project was related with the fact that only functional requirements were defined initially and, being so, the proposed solution had to be designed from scratch, with the added degree of responsibility that represented. For that purpose and as already referred in section 3.1, a System Engineering methodology has been employed, since it allowed a structured and supported development of the final system.

One of the most important steps consisted in the technological choices made before the implementation of the system prototype, the reasons presented in section 4.1 proved to be wise, as the main components performed as expected.

After analyzing the results presented it is possible to assume that the main goals for this work were achieved, since the two main functional blocks and the communications between them have been successfully implemented. This was essential to validate the proposed system architecture.

As a result, a paper abstract describing this work has been submitted to OCEANS'2016 MTS/IEEE Monterey conference[1] and has been accepted in the student competition, this meaning that a poster will be presented in the conference and a full paper will be published in the conference proceedings.

6.1 Future Work

After having implemented and validated the proposed solution main features and the interaction between them, the following step is to have a fully functional system operating in the environment for which this system was designed. For that purpose there are several tasks to be accomplished.

So far the main sensor node software performed well in the tests it was submitted to. However it must be tested using a more complex sensor network, as it has only been tested with six tiny sensor nodes so far, and some unanticipated failures might have to be solved. Then some of its hardware components must be added, configured, and protected against the environmental conditions the system will be exposed to.

The integration of the power module present in the proposed architecture is one of the fundamental steps towards the system's full operation. The Beaglebone Black and the tiny sensor nodes power consumption when running the final versions of the software must be measured in order to make the right option when choosing the batteries responsible for powering the Beaglebone Black and the sensor network.

The tiny sensor nodes firmware must be optimized, so that they are capable of collecting more samples, as already discussed in section 4.3.4. Another important development of high interest for these devices is the implementation of the microcontroller sleep modes, that may not represent a considerable power saving for one tiny sensor node, but if a whole sensor network is considered the system operation time can be highly increased.

Another fundamental step towards a fully operational system is related with the implementation of the I2C sensor network in the particular application this solution was proposed, as it

[1] http://www.oceans16mtsieeemonterey.org/

requires its installation in a situation that requires long cables (1-3m). For this it is necessary to select the adequate cabling for aquatic environments and integrate the I2C repeaters throughout the bus as discussed in section 4.3.4.

However, after a having a fully functional prototype it is always of great interest to enhance the system capabilities with features that are not critical but can be of high interest for its final user. Therefore some objectives are suggested, such as:

- Implementation of a GUI (Graphical User Interface) where the system user can easily define the setups to be processed by the main sensor node and invalids setups are refused, instead of having to carefully edit a text file.
- Development of a method that allows the remote modification of the main sensor node setups.
- Upgrade the communications application currently running in the main sensor node by a an application that uploads the collected data to a relational database management system, as it facilitates the future manipulation of that data.
- Development of a website for displaying all the collected data available in the database.

The last two suggested features are already being developed for another parallel project with CIBIO/InBIO, that is also based in a Beaglebone Black which eases a future integration of both systems.

Also, there is a wider scenario that can be considered, which requires the implementation of special sensor node to coordinate a larger and more complex network composed by several main sensor nodes, with each main sensor node coordinating its sub-network of tiny sensor nodes.

References

[1] FAO. *The state of world fisheries and aquaculture*, volume 2014. 2014. URL: http://scholar.google.com/scholar?hl=en{&}btnG=Search{&}q=intitle:THE+STATE+OF+WORLD+FISHERIES+AND+AQUACULTURE{#}0, arXiv:978-92-5-106675-1, doi:92-5-105177-1.

[2] Stanford University. Half Of Fish Consumed Globally Is Now Raised On Farms, Study Finds – ScienceDaily, 2009. URL: http://www.sciencedaily.com/releases/2009/09/090907162320.htm.

[3] Horst Kaiser, John S Lucas, and Paul C Southgate. *Aquaculture : Farming Aquatic Animals and Plants Aquaculture : Farming Aquatic Animals and Plants*. 2010.

[4] N.P. Burnett, R. Seabra, M. De Pirro, D.S. Wethey, S.A. Woodin, B. Helmuth, M.L. Zippay, G. Sarà, C. Monaco, and F.P. Lima. An improved noninvasive method for measuring heartbeat of intertidal animals. *Limnology and Oceanography: Methods*, 11(FEB):91–100, 2013. doi:10.4319/lom.2013.11.91.

[5] Miguel Gandra, Rui Seabra, and Fernando P Lima. A low-cost, versatile data logging system for ecological applications. *Limnology and Oceanography: Methods*, 13(3):115–126, 2015. URL: http://dx.doi.org/10.1002/lom3.10012, doi:10.1002/lom3.10012.

[6] C. D. Whiteman, J. M. Hubbe, and W. J. Shaw. Evaluation of an inexpensive temperature datalogger for meteorological applications. *Journal of Atmospheric and Oceanic Technology*, 17(1):77–81, 2000. doi:10.1175/1520-0426(2000)017<0077:EOAITD>2.0.CO;2.

[7] John Heidemann and Ramesh Govidan. An Overview of Embedded Sensor Networks. *Handbook of Networked and Embedded Control Systems*, pages 1–20, 2004.

[8] Jane K. Hart and Kirk Martinez. Environmental Sensor Networks: A revolution in the earth system science? *Earth-Science Reviews*, 78(3-4):177–191, 2006. doi:10.1016/j.earscirev.2006.05.001.

[9] Chee Yee Chong and Srikanta P. Kumar. Sensor networks: Evolution, opportunities, and challenges. *Proceedings of the IEEE*, 91(8):1247–1256, 2003. doi:10.1109/JPROC.2003.814918.

[10] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. Wireless sensor networks: a survey. *Computer Networks*, 38(4):393–422, 2002. URL: http://www.sciencedirect.com/science/article/pii/S1389128601003024, arXiv:1004.3164, doi:10.1016/S1389-1286(01)00302-4.

[11] A. Davis and Hwa Chang. Underwater wireless sensor isnetworks. In *Oceans, 2012*, pages 1–5, Oct 2012. doi:10.1109/OCEANS.2012.6405141.

[12] Jayavardhana Gubbi, Rajkumar Buyya, Slaven Marusic, and Marimuthu Palaniswami. Internet of Things (IoT): A vision, architectural elements, and future directions. *Future Generation Computer Systems*, 29(7):1645–1660, 2013. URL: http://dx.doi.org/10.1016/j.future.2013.01.010, arXiv:1207.0203, doi:10.1016/j.future.2013.01.010.

[13] Delta-T Devices Data Loggers. http://www.delta-t.co.uk/default.asp. Accessed: 2016-02-11.

[14] Products - NexSens Technology Inc. http://www.nexsens.com/products. Accessed: 2016-02-12.

[15] Products: Major categories of the Campbell Scientific product line. https://www.campbellsci.com/products. Accessed: 2016-02-11.

[16] Peter Blank, Patrick Kugler, Dominik Schuldhaus, and Bjoern M. Eskofier. A Data Collection and Communication Module for Telemedicine and mHealth Systems. *Proceedings of the 9th International Conference on Body Area Networks*, pages 154–158, 2014. URL: http://www.scopus.com/inward/record.url?eid=2-s2.0-84936014076{&}partnerID=tZOtx3y1, doi:10.4108/icst.bodynets.2014.257018.

[17] Maria Clarissa Carasco, Jan Pierre Pizarro, and Giovanni Tapang. Development of a Microcontroller-based Wireless Accelerometer for Kinematic Analysis. *Journal of Mechatronics, Electrical Power, and Vehicular Technology*, 6(1):1–8, 2015. URL: http://www.mevjournal.com/index.php/mev/article/view/215, doi:10.14203/j.mev.2015.v6.1-8.

[18] Martin Brooke, Eli Cole, Julian Dale, Henry Quach, and Becca Bau. An Ocean Sensor for Measuring the Seawater Electrochemical Response of 8 Metals Referenced to Zinc, for Determining Ocean pH. *Ninth International Conference on Sensing Technology (ICST)*, 2015. URL: http://dukespace.lib.duke.edu/dspace/handle/10161/11159?show=full.

[19] P. Rajgarhia, P. James, V. Nainwal, P. Sowjanya, and S.S. Koshy. Development of an ultra-low power wsn gateway for outdoor deployments. In *Information Communication and Embedded Systems (ICICES), 2014 International Conference on*, pages 1–7, Feb 2014. doi:10.1109/ICICES.2014.7034151.

[20] Angelo Chianese, Francesco Piccialli, and Giuseppe Riccio. *Designing a Smart Multisensor Framework Based on Beaglebone Black Board*, pages 391–397. Springer Berlin Heidelberg, Berlin, Heidelberg, 2015. URL: http://dx.doi.org/10.1007/978-3-662-45402-2_60, doi:10.1007/978-3-662-45402-2_60.

[21] C.Niladri J.Manikandan and V.K.Agrawal. Sensor monitoring using i2c protocol and labview. In *Proceedings of International Conference on "Emerging Research in Computing, Information, Communication and Applications*, 2013.

[22] Paul Horowitz and Winfield Hill. *The Art of Electronics*, chapter 14, pages 1032–1045. Cambridge University Press, 3 edition, 2015.

[23] Frédéric Leens. An introduction to I2C and SPI protocols. *IEEE Instrumentation and Measurement Magazine*, 12(1):8–13, 2009. doi:10.1109/MIM.2009.4762946.

[24] Philips Semiconductors. *I2C manual*, 2003.

[25] Z Mihajlovic and V Milosavljevic. Application of GPRS modules in data acquisition and control of devices for air quality monitoring. ... *(TELFOR), 2012 20th*, pages 1020–1023, 2012. URL: http://ieeexplore.ieee.org/xpls/abs{_}all.jsp?arnumber=6419383.

[26] Chen Xiaorong, Shi Zhan, and Ge Zhenhua. Research on Remote Data Acquisition System Based on GPRS. *Electronic Measurement and Instruments, 2007. ICEMI '07. 8th International Conference on*, pages 2–23, 2007. doi:10.1109/ICEMI.2007.4350652.

[27] Jin-Shyan Lee, Yu-Wei Su, and Chung-Chou Shen. A comparative study of wireless protocols: Bluetooth, uwb, zigbee, and wi-fi. In *Industrial Electronics Society, 2007. IECON 2007. 33rd Annual Conference of the IEEE*, pages 46–51, Nov 2007. doi:10.1109/IECON.2007.4460126.

[28] Maurizio Di Paolo Emilio. *Embedded Systems Design for High-Speed Data Acquisition and Control*. Springer International Publishing, 2014. doi:10.1007/978-3-319-06865-7.

[29] IEEE. *IEEE Standard for Application and Management of the Systems Engineering Process*, volume 1220. 2005. doi:10.1109/IEEESTD.2010.5703195.

[30] Beaglebone black rev c - element14 commercial edition | logic supply. http://www.logicsupply.com/bb-black-c/. Accessed: 2016-06-23.

[31] Derek Molloy. *Exploring BeagleBone: Tools and Techniques for Building with Embedded Linux*. Wiley, 2014. URL: http://www.exploringbeaglebone.com/.

[32] systemd - debian wiki. https://wiki.debian.org/systemd. Accessed: 2016-06-03.

[33] Infrared pulse sensor. http://makezine.com/projects/ir-pulse-sensor/. Accessed: 2016-05-22.

[34] Reflective optical sensor with transistor output. http://www.vishay.com/docs/83751/cny70.pdf. Accessed: 2016-05-18.

[35] Github - damellis/attiny: Attiny microcontroller support for the arduino ide. https://github.com/damellis/attiny. Accessed: 2016-04-03.

[36] Atmel Corporation. *Using the USI module as a I2C slave*, 9 2005.

[37] Tinywire/tinywires at master · rambo/tinywire · github. https://github.com/rambo/TinyWire/tree/master/TinyWireS. Accessed: 2016-02-30.

[38] NXP Semiconductors. *Sending I2C bus signals via long communications cables*, 2 2008. Rev. 01.