NTNU – Trondheim
Norwegian University of
Science and Technology

Data Acquisition with Unmanned Aerial Vehicle (UAV) from Floating Sensor Nodes

Remus-Gabriel Barbatei

Embedded Computing Systems
Submission date: July 2014
Supervisor: Tor Arne Johansen, ITK
Co-supervisor: Amund Skavhaug, ITK

Norwegian University of Science and Technology
Department of Engineering Cybernetics

Data Acquisition with Unmanned Aerial Vehicle (UAV) from Floating Sensor Nodes

Remus-Gabriel Barbatei

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in the

European Masters in Embedded Computing Systems
Department of Engineering Cybernetics
Norwegian University of Science and Technology

7 July 2014

Supervisor: Professor Tor Arne Johansen
Co-Supervisor: Associate Professor Amund Skavhaug

Preface

This Master Thesis was carried out as a part of the study program European Masters in Embedded Computing Systems at the Department of Engineering Cybernetics at the Norwegian University of Science and Technology. The work was done in the time period between January and July 2014.

This thesis is a continuation of a previous project done by the author at the Norwegian University of Science and Technology during the fall semester of 2013. In order to clearly describe the underlying theoretical aspects some parts of the fall project have been rewritten in this current work.

This project is one of the many projects carried out at the UAV Lab at the Norwegian University of Science and Technology It is also supported by the Center for Autonomous Marine Operations and Systems (AMOS).

Acknowledgments

I want to give thanks to my parents and very good friends, for the moral support I received throughout my entire academic development, I say thank you with all my heart!

I gratefully acknowledge the support and resources made available to me by my supervisor, Professor Tor Arne Johansen, for his patience, careful guidance and support at all times. Without him I would have not come this far. It has been a beneficial opportunity for me to work on a project that is part of the UAV lab which he is the director of.

Special thanks go to Associate Professor Amund Skavhaug for his continual encouragement and constant motivation, pushing me to do more and always keep an ambitious attitude towards work. He has been more than a good supervisor to me, offering much more than academic advice for which I am most grateful.

My gratitude also goes to the staff of the Department of Engineering Cybernetics for all the support that they offered me, with special emphasis to: John Olav Horrigmo, Per Inge Snildal, Terje Haugen, Lars Semb, Torkel Hansen and Janne Karin Hagen.

I would be ungrateful if I fail to acknowledge the support I had from Nhan Nguyen, Kristian Klausen, Frederik Stendahl Leira and even fellow colleagues from the University of Porto in Portugal.

Title: Data Acquisition with Unmanned Aerial Vehicle (UAV) from Floating Sensor Nodes

Student: Remus Gabriel Barbatei

Problem Description:

One of our research visions is coordinated data acquisition from marine sensor networks that forms a heterogeneous network of Unmanned Aerial Vehicle nodes (UAVs), Autonomous Underwater Vehicle nodes (AUVs), floating sensor nodes, and surface ships.

The project will focus on the interaction between UAVs and floating sensor nodes having built-in sensors and radio communication capabilities and, in the future, hydro-acoustic communication as well. The first step is the development of the floating sensor node design, hardware and software alike. The software is based on 2 multi-node coordination open-source software frameworks, NEPTUS and DUNE that have been developed by the University of Porto.

The main tasks and objectives of the candidate are:

1. Carry out a study of required background theory and previous related work
2. Propose a solution for the general system
3. Design and implement the proposed solution, both hardware and software.
4. Perform tests (both in the lab and in the field) with the built hardware and the software system that comes with it
5. Evaluate and discuss the results
6. Draw practical conclusions from what has been built, programmed and tested
7. Suggest further work and improvements

Supervisor: Professor Tor Arne Johansen
Co-Supervisor: Associate Professor Amund Skavhaug

Summary

Short background, results and conclusions

In the last decades an increase in the need for ocean monitoring capabilities has arisen. To be able to meet these requirements a low-cost, automated solution is needed. It should be easy to maintain and to reconfigure as well as environmentally friendly.

To this extent, a small-sized floating sensor node would be suitable for the task. The node would be similar to a floating buoy but having a size comparable to that of a soda can. Furthermore it should be easily deployable and have the possibility to be recuperated from the ocean surface using UAVs with fixed-wing or using multicopters but these aspects are not part of this current work.

There is also a need to transfer the data between the UAVs, sensor nodes and the end user. This is done using robust hardware, reliable software frameworks and well established protocols. For the interface and communication between the sensor nodes and UAVs a software framework, called DUNE, is used and for the end-user interface NEPTUS is employed. Both of these will be described in a later chapter.

In order to achieve a unified communication protocol among the devices in the system the IMC (Inter-Module Communication) message formatting is used, detailed later on.

The work touches topics in disciplines such as *Engineering Cybernetics*, *Electronics Engineering* and *Computer Science*.

Analysis and exploration has been carried out in order to analyze current existing solutions that addresses the problem, reveal what kind of measurements are possible to do with a small-sized sensor node and what kind of sensors can be used.

It is important to mention that the software frameworks that have been used, DUNE and NEPTUS are projects currently being developed. They are considered to be research topics and they are also viewed as unique as compared to other similar frameworks in development today. These shall be described throughout this report. They are open source projects and have been developed during the last 10 years and are constantly being improved. They were

originally used by the University of Porto and now more and more Universities are gaining inters in them.

Objectives that have been achieved by the candidate throughout this work:

- perform research on the type of hardware components needed
- mechanically design a physical case for the device
- design custom printed circuit boards for the proposed hardware
- write the firmware that runs on the embedded device
- program the necessary software used to communicate with the built embedded device
- perform verification and validation on the system
- perform measurements on data integrity and range over the wireless medium, power consumption and hardware limitations
- carry out lab and field tests in order to prove the functionality of the proposed solution
- analyze and interpret the results

The total work carried out, tests and development that were carried out are split into two main categories, *hardware* and *software*.

The hardware is composed of a sensor node with all required components (passive and active) and several boards which form a payload for the UAV. The UAV acts as a data sampler and communication relay between the sensor node (in the water or on ground) and a base station. Building the sensor node is part of the hardware category. The sensor node has been built (and tested) to be buoyant, directionally stable in water and hermetically sealed.

Among the software tasks that come into the proposed solution are the programming of the microcontroller that runs the code on the embedded device, programming parts of the software framework (DUNE) that is used to communicate with the embedded device and prepare the intercommunication between DUNE and the end-user application, NEPTUS.

A simple conclusion on the work that has been carried out so far, after running field tests with the proposed system is that the proposed solution can be a viable solution good for the scientific community and with more work it could even be developed into a working product.

Contents

List of Figures

List of Acronyms

API Application programming interface

AUV Autonomous underwater vehicle

CDMA Code division multiple access

CPU Central processing unit

DMA Direct memory access

GPS Global Positioning System

GSM Global System for Mobile Communications

I²C or I2C Inter-Integrated Circuit

IMC Intermodule Communication

MCU, µC Microcontroller

OO Object oriented

OS Operating System

PCV Printed Circuit Board

POSIX Portable Operating System Interface

RTOS Real-time operating system

SD Secure Digital

SPI Serial Peripheral Interface bus or

UART Universal Asynchronous Receiver/Transmitter

UAV Unmanned Aerial Vehicle

UML Unified modeling language

1. Introduction

1.1 Background of the task

Global wheatear changes have become a significant concern in our life over the last few years. It is estimated that the sea level is rising at high rates every year, up to 3mm/year. Along with this, the Arctic ice cover is shrinking and there is an increase of temperature in high latitude areas. Sometimes lives are lost by extreme weather events and there are enormous burdens on local organizational and insurance companies and administrative departments in the countries that are close to the sea shore. Since the scientific community began recording statistics, around 1860, we had the 8 out of the 10 warmest years globally, only in the last decade. The main reason for these effects are a mixture of natural variability and long-term climate change. Sometimes, the impact of these changes can be both good and bad. It can be good because the growing seasons are longer in some areas and also, there are new trading opportunities due to the opening of Arctic shipping routes. But these changes can also have a negative impact. They can cause coastal flooding, more frequent and extreme heat waves, intense and dangerous droughts and, in some cases, severe phenomenon like tropical cyclones can be observed [1].

The ocean's ecosystem is also impacted by human activities. Entire habitats are severely affected yearly and most of these are either because of human negligence or a lack of care for the environment. Oil spills problems on the ocean have started to gain more and more attention from the world in the last 20 years. This is because disasters caused by such chemicals in the water greatly impact sea life and actually can destroy entire species of fish and birds that depend so much on clean water. For example, oil can destroy the insulating ability of the fur in some mammals (like sea otters). The water insulating capability of the feathers on some birds is also compromised by oil making them vulnerable to the harsh elements of the ocean and increasing their probability to die because of hypothermia. Some bird species also can ingest oil along with the water they use to clean themselves, poisoning them in the process. Of course the ocean contains many kinds of fish and shellfish and even though they might not be exposed directly to oil spills they do come in contact with the water and this can lead to severe effects on their health in the future. For example, when adult fish comes in contact with water contaminated by oil they can experience changes in heart and

respiration rates, reproduction impairment, reduced growth, fin erosion and other effects. Oil in water can also have bad effects on fish eggs and larval survival rate [2].

1.2 Motivation

Having a proper understanding of occurring oceanic phenomenon and eventually predicting them is required in order to guide international actions, to shape industrial strategies and to optimize governmental policies against these issues. To make these predictions possible an accurate model of the climate of the earth is required, particularly the climate of the oceans. The data for this kind of sea model has to be collected somehow. Lack of sustained observations about the ocean have set back greatly the development of valid climate models of the ocean. There are many examples in which sea monitoring was required in order to avoid serious disasters. Furthermore monitoring oil spills and other kind of substances or garbage that is illegally dumped on the ocean surface has started to be of great value in the last decade.

Not just large bodies of waters such as seas and oceans need to be monitored. Small lakes, rivers and even man-made reservoirs require observation. Measuring parameters such as water purity of a spring, the amount of minerals in some lakes and even monitoring fish in fish farms are some more usage scenarios which require an adequate system to solve this.

The issues specified have to be addressed somehow, in an efficient and cost effective manner that first of all does not further impact the environment in a negative way.

2. Solutions to address this problem

2.1 Current solutions

2.1.1 Argo Sensor nodes

Several attempts at monitoring the sea have been made, out of which the most mentionable of all is ARGO. It is a global array composed of around 3000 profiling sensor nodes that are free-drifting and constantly measure the temperature and salinity of the world's oceans. They are not only floating but can submerge to up to 2000m under the ocean, allowing for continual monitoring of salinity, temperature and the velocity of the upper ocean. All the data they gather they send via satellite to a centralized database and this is publicly available [1].

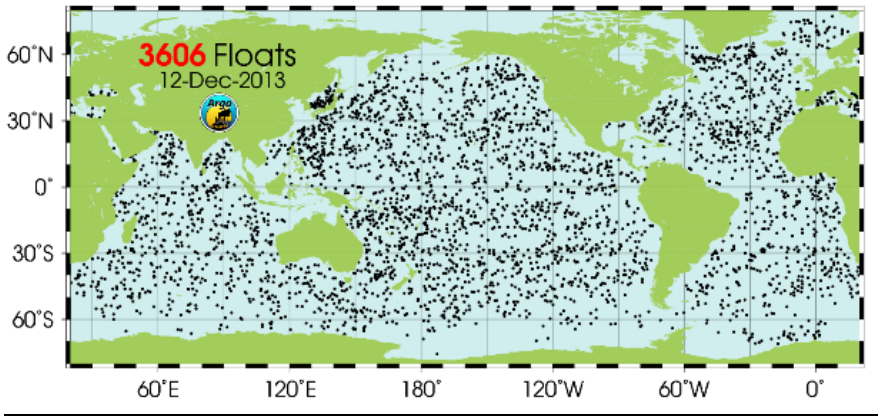


Fig. 2.1 - ARGO nodes around the world [1]

The project is well appreciated in the scientist community as many organizations and weather monitoring facilities use it daily. *Fig. 2.1* shows a rough estimate of where each node is located in the world's oceans, based on their last reported GPS coordinates.

Fig. 2.2 - ARGO node and ship [1]

Some key characteristics of these nodes include their size and weight. They are approximately 1.3m long and about 20cm in diameter. They also weigh a hefty 40kg. Also because of their size and weight they are difficult to maintain and usually when something happens to one of them they are simply replaced.

This combined with their total cost of $15,000 makes reusability and expandability an issue. They also require ships to be deployed and collected from the ocean surface (*Fig. 2.2*) further adding to the overall cost of using such devices.

However even with all of these factors this project continues to expand and receive more and more involvement from governments in many countries around the world.

2.1.2 Starmon mini - underwater temperature recorder

Another mentionable device, this time on a smaller scale is the Starmon mini - underwater temperature recorder. It is a miniaturized sensor (25mm x 130mm) that is capable of long period temperature measurements and logging underwater [3]. The next image *(Fig.2.3)* shows the sensor. One first disadvantage can be seen immediately: to get the sampled data, the sensor needs to be picked up from the water. Nevertheless some applications are well suited for this kind of sensors even if it's not that convenient if you would need to download the

sampled data from 100 such sensors at one time. A centralized way of collecting the data from such sensors is needed.

Fig. 2.3 - Miniaturized Underwater Temperature Sensor [3]

Fig. 2.4 - Submersible Acoustic Transmitter [4]

2.2.3 Star Oddi FishCall

Another kind of device that is used to interact with the marine biodiversity is a submersible acoustic transmitter *(Fig.2.4)*. It is a low frequency transducer, loudspeaker and sounder, developed to attract fish. The main usage scenarios for such a device include the ability to train fish so that they can be easily relocated and monitoring their lifespan [4].

Being a relatively large device it might be difficult to handle and maintain. The immediately observable characteristic of this device is its size. In an ever miniaturizing world as the world we live in today, size tends to matter more and more meaning that such monitoring devices are viewed better if their size is small. Small size generally means lower power, leading to longer battery life, and a longer life cycle.

2.1.3 MATAKI - low-cost, wirelessly-enabled tracking platform

Other similar solution for monitoring, but this time, not particularity floating or water-related have also proven to solve the problem of tracking and monitoring in an elegant and efficient way. One example of such a solution is called MATAKI [6]. They are small boards fitted with sensors, GPS modules and RF communication. They are low power and can be adapted and used in various applications.

2.1.4 Real-Time Deep-Ocean Tsunami Measuring, Monitoring, and Reporting System

Another solution for deep opcean monitoring is the second-generation Deep-Ocean Assessment and Reporting of Tsunamis system know as DART II. Tsunami data from the DART system can be combined with seismic data ingested into a forecast model to generate accurate tsunami forecasts for coastal areas.[5]

The sistem is composesd of two physical components: a tsunameter on the ocean floor and a surface buoy with iridium satellite network telecommunications capability.

The DART II systems have bi-directional communication links and are thus able to send and receive data from the Tsunami Warning Center and others via the Internet.

The Tsunameter incorporates a divers set of componentes. The computer, designed around the 32-bit, 3.3 volt Motorola 68332 microcontroller, reads presure readings, runs a tsunami detection algorithm, and sends and recivies commands and data tto and from the buoy via an acuoustic modem(A Benthos ATM-880 Telesonar acoustic modem with an AT-421LF directional transducer). The pressure sensor is a 0-10,000 psi model 410K Digiquartz® unit manufactured by Paroscientific, Inc.

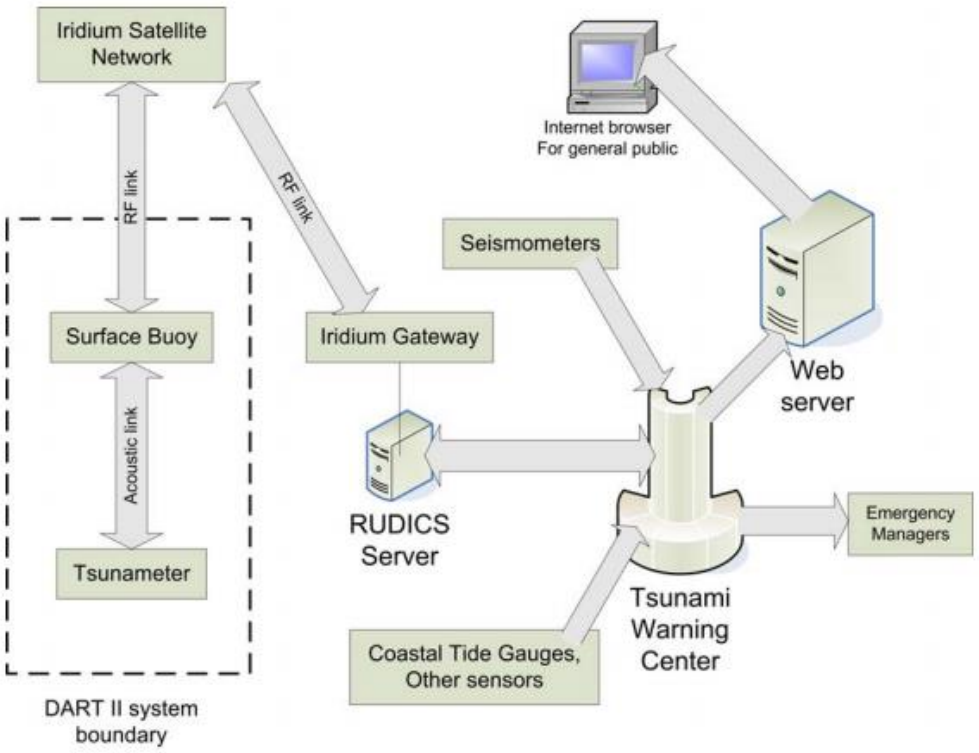


Fig. 2.5 - Context diagram showing a DART II system and the related telecommunication nodes. [5]

2.2 Proposed solution and objectives

Our solution to address the problem of ocean monitoring in a cheap and eco-friendly way involves a new kind of sensor node. It is small-sized, no more than 30cm in height, 6cm in diameter and low-weight (around 500 grams). The beauty of this is that it is deployable and recoverable using unmanned aerial vehicles (UAVs).

The sensor node is also modular, so that it can have a multitude of sensors and functionalities making it vary in size, depending on what kind of sensors it has connected to it. It encapsulates small and low power sensors, GPS positioning, wireless communication capabilities and a non-volatile storage medium.

2.3 Outline of the report, structure of solution description

In the next chapters the proposed solution shall be discussed. At first the physical characteristics of the sensor node will be treated then, moving on further to the electric characteristics and a description of what components are inside the node. After that the software part is also described.

This software part is split into 3 main categories: the embedded software that runs on the device itself, the software framework that is used to communicate with the device and the end-user interface software that displays the sensor node. Besides all this, a protocol will be described in detail as to how the low-level software on the sensor node communicates with the higher level software.

In the latter chapter, tests and experiments will be described and discussed. The basic features of the built prototype of the sensor node have been successfully tested and proved to work. All of the results will be treated and discussed in details.

Finally, since this is a rather complex project, future goals and suggestions have a separate chapter.

3. Proposed Solution

3.1 Introduction, General Overview

The proposed solution to deal with the problem described in the previous chapter involves more than just a simple sensor node with embedded software running on it. It is a composition of robust hardware carefully designed to withstand more than normal working conditions. After all, some of the operating environments in which the sensor should work can involve both warm and cold ocean waters, strong winds and even mechanical shocks.

Along with this kind of hardware, a carefully designed software framework is part of the solution. The whole system uses well thought-out communication protocols adding a certain level of completeness to the whole implementation.

The solution, first of all, has to require as little maintenance as possible. Monitoring of the ocean needs to be done in an automated way, with as less human involvement as possible. This solution aims at removing the need for a boat placing or picking-up the sensors in the ocean. This implies having the sensor nodes picked-up and removed from the ocean automatically with the help of Unmanned Aerial Vehicle. UAVs are also to be used to collect data from the sensor node, to store it and forward it to a ground base station where it can be processed by meteorologist, biologists, physicists, etc.

The next figure (*Fig 3.1*) shows a representation of how the whole system functions. UAVs are launched from the ground station with the purpose of deploying sensors on the ocean, to collect the data that has already been sampled by previously placed sensors or to pick up the sensor.

Sensor deployment and recovery from the ocean surface are not part of this current work and will not be treated in great detail. However, the whole communication and system interaction will be described in detail throughout this report.

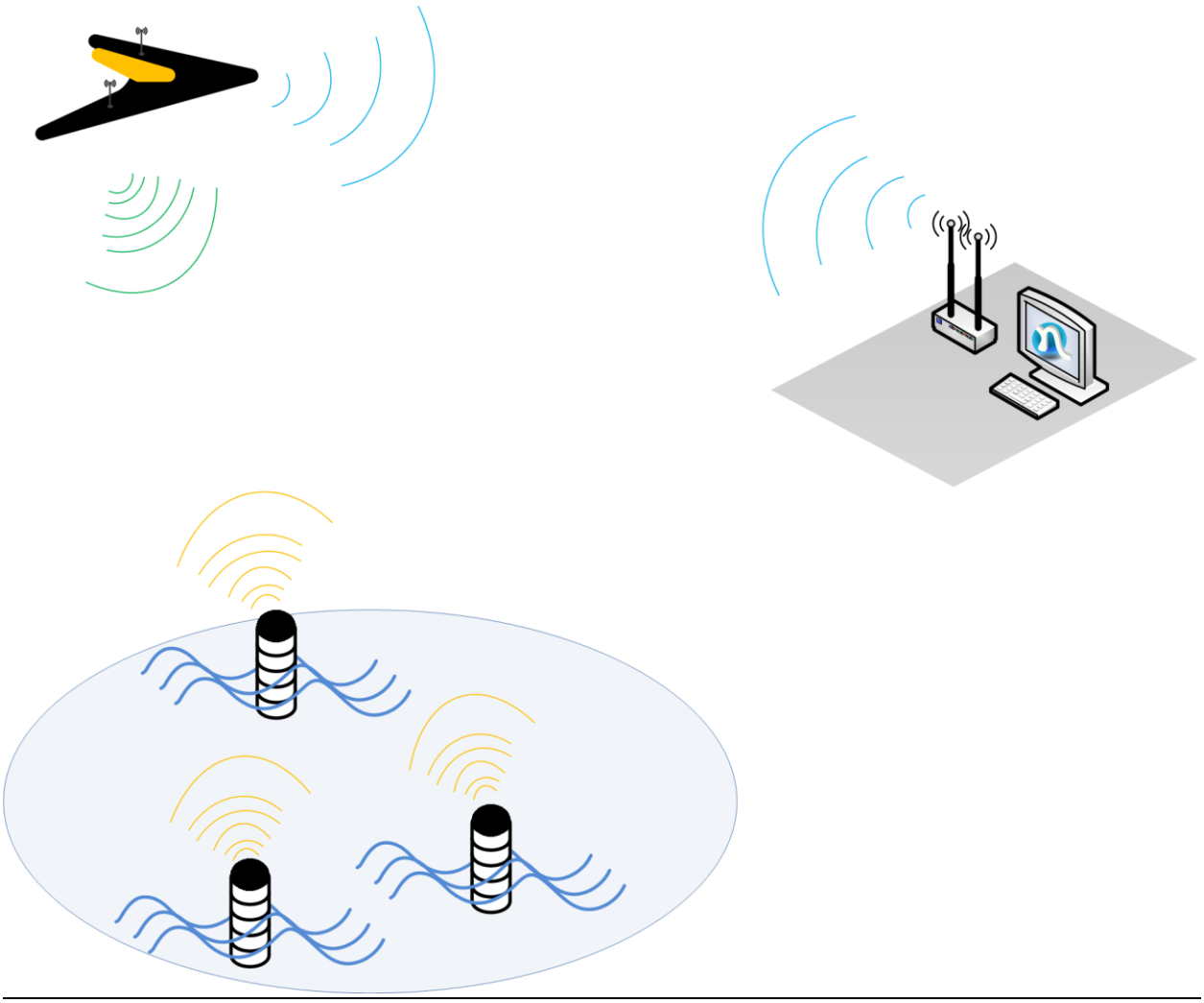


Fig. 3.1 – General overview of system components and interaction

3.2 General characteristics

The main physical characteristics of the device are buoyancy, and water tightness and being directionally stable in water. Even though these seem like common sense, all design must begin and end with these in mind. In the water it will stay upright, having most of its case completely submerged and the other part (which contains the antennas) will be floating. This characteristic of being directional stable in water is achieved by carefully placing the components inside in such a way that the heaviest ones are situated at the bottom and at the top part of the cylindrical sensor node there are only antennas and optionally a deployment and recovery mechanism. These mechanisms can be considered to be a combination of special

hooks. A rough representation of how the device's modularity is organized is shown in the following figure (*Fig.3.2*).

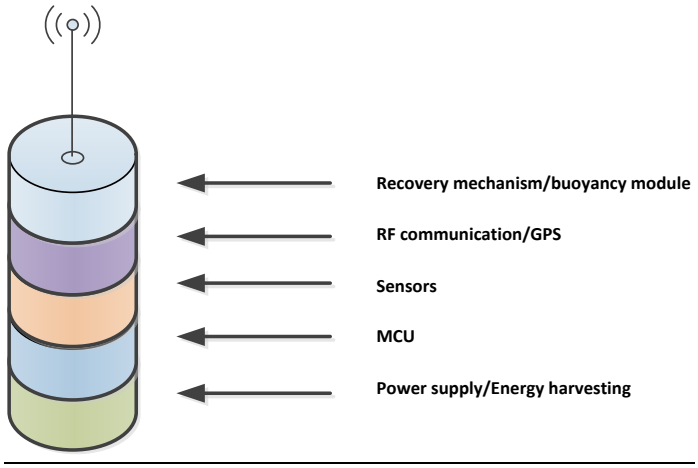


Fig. 3.2 - Modular concept of the sensor node

One of its important characteristics of the sensor node is that it should have a modular design. Modular in the sense that it should be easy to add new sensors to it, or remove old ones in the case that they become defective. Also, this modularity must be combined with the very important property of being a watertight sensor node. To achieve this kind of modularity a cylindrical design composed of many cylinder segments is to be used. An example of this is shown in *Fig. 3.3*.

Fig. 3.3 – Possible Physical Design of the Sensor Node

In the figure, each section represents a different functionality. To be able to satisfy the properties described earlier (directionally stable in water and buoyant) some sections of this node can be entirely dedicated for this, containing low density, high volume material that floats. For example the upper most section is composed of materials that offer buoyancy to the entire cylinder while the lower most part of the node has the heaviest components like the battery.

The way the sections are connected to each other to provide a safe seal is also very important. Some of the ways in which this can be achieved are to have threads on each section and screw the sections together like a normal bottle screw or just by joining them together, without screwing them into position, but keeping them joined with metallic or plastic small screws. They should also contain rubber O-rings for water sealing. Because each section has a dedicated task and functionality there is clear need of all being connected electrically to the power source and to the main MCU board. This calls for a way of interconnecting the sections in a way that does not interfere with the screwing in place and assembling the whole device.

As mentioned earlier the device has GPS positioning and wireless communication capabilities with the UAVs. There is a minimum of 2 antennas that need to be fitted onto the node: one for the GPS module and one for the wireless communication. The normal way to proceed in this situation is to have 2 separate antennas, one of which is a simple straight (flexi) antenna (the one used for wireless communication, the GPS antenna is standard). But as theory states flexi antennas are designed to have the range of effect on a horizontal plane and not on a vertical one.

Naturally, an entire network of such sensors deployed on the ocean surface is the end goal. Ideally these sensors should be able to communicate and exchange data with each other but there is one big problem with this: because of the sea waves certain nodes will be out of line-of-sight range and can be blocked from communicating with one another, as shown in *Fig. 3.4.*

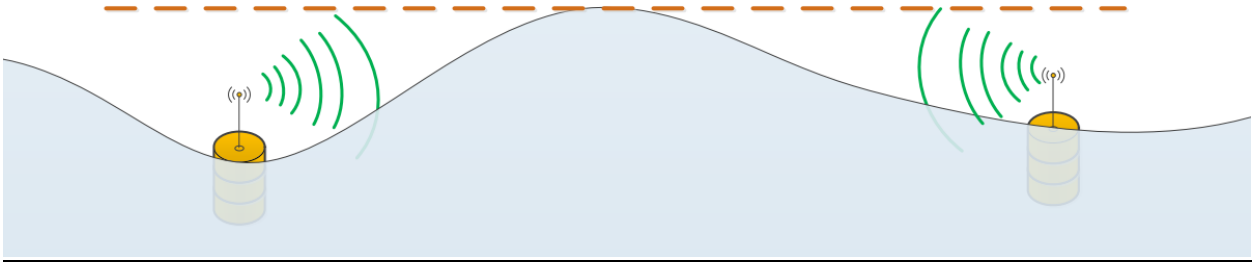


Fig. 3.4 - Example of disrupted communication because of the ocean waves

Even in this situation an actual data link might be made and communication could be established but it comes with great power costs. Since we want the sensor nodes to be low power this leads us to make the decision of designing the nodes in such a way that they don't exchange data with each other but only communicate with the UAVs. This again leads to other issues caused by the type of antenna we use. As mentioned earlier normal flexi-type antennas are designed to have horizontal functionality range and not vertically as we need it. This is cleverly solved by using a puck antenna.

Fig. 3.5 - Combo puck antenna [7]

These kind of antennas are designed to have an effective range in the shape of a semi-sphere making them perfect for ensuring the communication channel between the node and the UAV. Because of the lack of space a clever way of satisfying both communication needs and size constraints is to use a combo puck antenna *(Fig. 3.5*). This kind of antennas feature 2 built in antennas in one package, one for GPS and one for GSM, and CDMA type wireless communication capabilities [7].

Since the sensor node will be carried, deployed and recovered by UAVs it needs to have some certain weight and size constrains previously mentioned. Also it must tolerate around 15G of acceleration and shock forces since it will be dropped from the UAVs when deployed and

rapidly pulled out of the water when recovered. These requirements can be safely satisfied if we use a strong plastic case for the node. Also, when going for plastic as the material for the case it should be pointed out that plastic is not affected by salty water corrosion, is resistant to temperatures and is also low cost.

Low power is of great importance for this project and wave energy harvesting could also greatly impact the way the node is designed. Many tweaks and design considerations for having a low-power system will be discussed throughout the document but at the end, as a future goal, there is a special chapter dedicated to wave energy harvesting.

4. Hardware Description

4.1 Introduction

This next section is meant to describe the hardware components and architecture of the sensor node. Details such as values of the passive components and other related facts are omitted. The focus is mainly on what components are used how they are interconnected and communicate with one another.

4.2 General Hardware Description

4.2.1 Main Microcontroller Board, Atxmega192C3

The core of the hardware lies in an 8/16bit microcontroller by Atmel, the ATxmega C3 core family chip. The main reasons why the decision to use this particular vendor for the microcontroller are as follows: First of all we needed something that is low cost and low power. And also the speed of the device is really not a very important issue since its main purpose is monitoring. The µC runs at maximum 32MHz and at that speed it can consume maximum 15mA. This gives us a lot of flexibility in terms of speed and power consumption. For example the speed of the device can be dynamically lowered or increased at runtime. The chip has the characteristic that after reset, it will always start up running from its 2MHz internal oscillator. Also, during normal operation, the system clock source and prescalers can be changed from software at any time and in turn lowering or increasing its power consumption.

Other important characteristics that were taken into consideration when choosing this µC involve the existence of a DMA bus and controller, further helping with lowering the power consumption by not loading the CPU too much during simple monitoring operations.

Like most microcontrollers, it has many peripheral interfaces for communicating with sensors, memory devices and even other microcontrollers. These include SPI, I2C, and UART which are used in our system.

4.2.2 Peripherals interfaces used

In this section a brief description of each peripheral device attached to the microcontroller will be provided.

First of all, since we are talking about mostly monitoring functionalities of the device there has to be a safe and cheap way of storing the monitored data. For this purpose we are using a micro SD card. These cards are cheap and can be easily interfaced with the microcontroller using the SPI interface.

Localization of the device is also needed and for this purpose a GPS module is connected to the µC. Most GPS modules have standard communication interfaces to work with a microcontroller and for our application we will be using the UART interface with the module. Implementation details for how this is achieved are not so relevant at this point since most GPS modules work very similarly.

Special attention must be given to the wireless communication module. Each sensor node must communicate with UAVs both for sending previously saved monitored data and for commands and configuration parameters. These parameters can be anything from specific measurement time intervals patterns or even modifications that alter the functionality of the node. For example, disabling a sensor due to a fault or inconsistent data reports. All of this information must be transmitted fast and over a relatively large distance. After all, fixed wing UAVs cannot fly really low or slow and they cannot wait in case of slow communication. Therefore the decision of using a RF communication module provided by Radiocrafts has been made. It is the Radiocraft-rf1280 module. In its US variant (433MHz) 2-4km in range can be achieved and this would be equivalent to around 1-2Km for the European version of the model (868MHz). It has a data rate of 4.8Kbits/sec and most important a low current consumption: 21mA for receiving and 28mA for transmitting the data. Further details for this module will be described in a later chapter. Another key important feature that makes it

suitable for our need is the fact that it has a standard UART interface and the device by itself takes care of data transmission and reception. The user only has to read or write data via the serial interface to it and the device takes care of all communication. Naturally, there are certain configuration parameters that need to be configured but all-in-all it is not complicated to work with such a device.

The microcontrollers is also fitted with an internal analog to digital converter and the main board is designed is a way that it allows easy access to the microcontroller pins that are connected to the ADC. Various analog sensors can be connected using this interface.

What makes this device unique is its modular design. Sensors need to be added and removed in a way that would not require any hardware modifications to the main board. For this reason all sensors that are to be connected to the device will use either an I2C interface or an industrial RS-485 interface. It is well known that both of these interfaces offer bus connection that can support multiple devices making them ideal for our need. Each new added sensor will just be a new device on the bus and modifications will only be carried out on the software part, substantially reducing the workload required for adapting the sensor node to new needs.

4.2.3 Overall architecture

Taking into consideration the previously described devices and communication interfaces a rough hardware architecture is presented in the following figure (*Fig. 4.1*). It shows a typical configuration with multiple sensors attached to the main microcontroller.

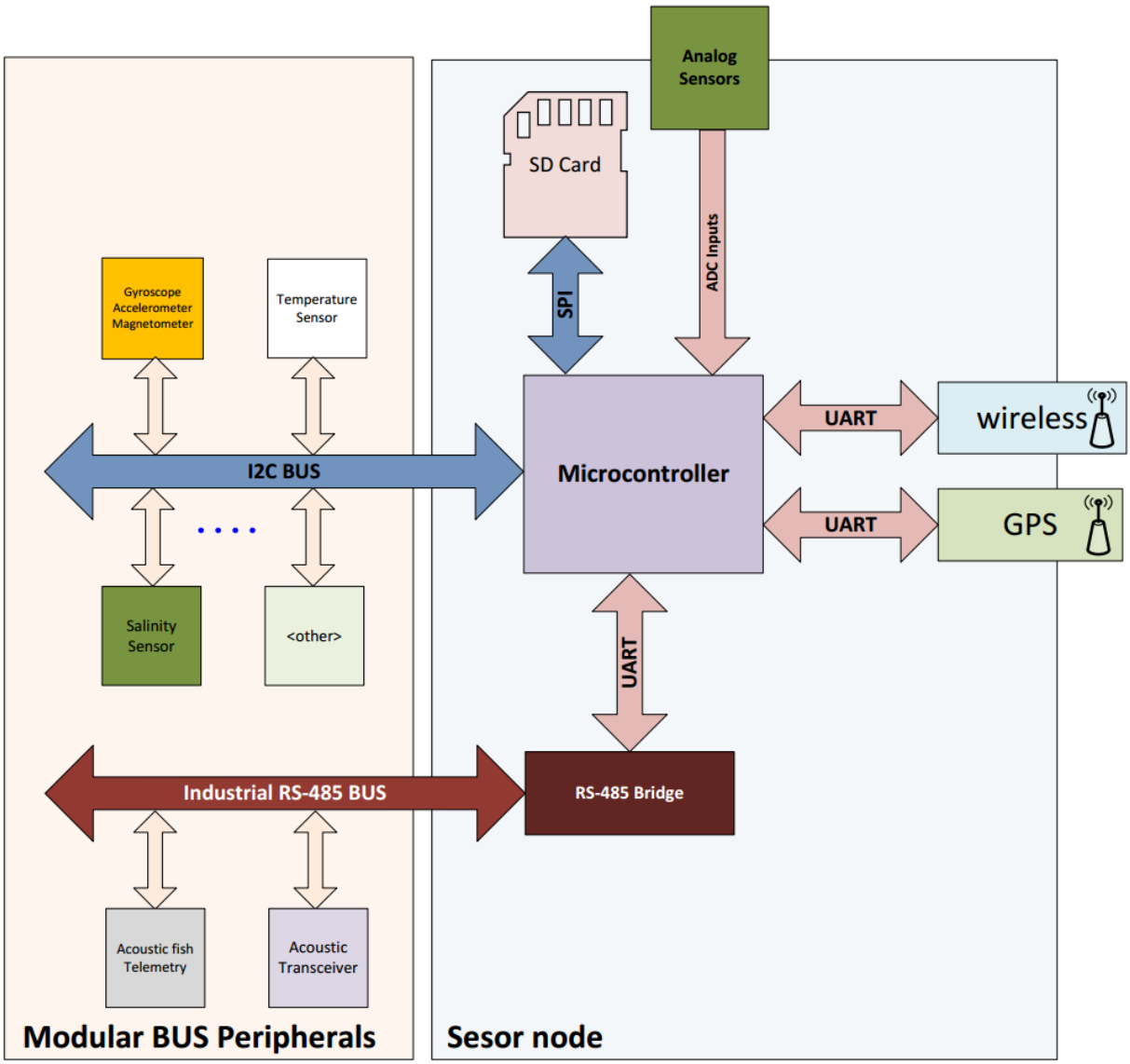


Fig. 4.1 - Overall hardware architecture

4.2.4 Other aspects

It is important to state that there is a minimum configuration of components that all sensor nodes must have. All nodes are to be equipped with data storage (micro SD card), wireless communication and a GPS module. The rest of the sensors can be chosen according to the

monitoring requirements. For example, some nodes can be specially designed to monitor oil spills while others can be designed to monitor sea life and sea conditions such as the level of chlorophyll in the water, nutrients, dissolved oxygen and pH. All the sensors that measure these are to be connected to the I2C bus, to the RS-485 bus or to the analog input pins. Therefore the printed circuit board must be designed to permit this. For example it might be necessary to modify the values of the pull-up resistors of the bus, depending on the number of devices connected on the bus.

Another key feature of the device includes self-monitoring. The sensor node is to be designed in such a way that the internal temperature of its components can also be viewed. For example it might be important to know the temperature of the battery and that of the microcontroller, since the whole device might be required to work in extreme temperatures.

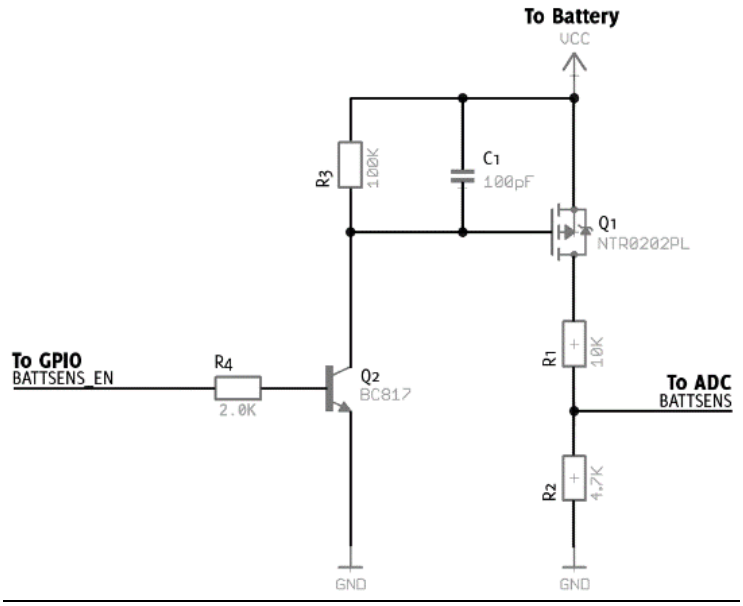


Fig. 4.2 - Battery sensing circuit [8]

Besides monitoring the temperature of the battery, the charge level of the battery must also be measured. For this we can either chose a smart battery that has internal circuitry for monitoring (and also connect it to the microcontroller via some interface). Or, we can add a chip that takes care of these functionalities or we can just create a simple circuit for battery voltage sensing and do the rest of the processing on the main µC. A simple battery measurement circuit is presented in *Fig.4.2.* As seen in the figure the circuit is designed in such a way that it does not consume battery energy unless it is used (enabled) [8].

Since the environment in which the sensor node function is a relatively cold one, techniques for extending battery life are very important and should be considered. These techniques are not discussed in detail here since there is literature on this particular topic [9].

4.3 Detailed Hardware Description

4.3.1 Physical case construction

The production of the proposed prototype was done in several steps.

First, the shape of the device an issue: should it be spherical? Cylindrical? Or Like a cube? The decision to make it have a cylindrical form has been made because it is easy to manufacture and, most importantly, most buoys have a cylindrical shape. So if something is already proved to work and it is used in many different scenarios, then it is a good idea to also use that design.

The second part of the design was to decide on the size of the device. This had to be done taking into consideration several issues such as the ability of being deployed in water easily, the size of the battery that can be fitted inside, and the size of the electronics that go inside.

After careful consideration on the availability and size of the hardware that can be fitted inside, the next step in designing the case was to use a 3D printer and print an initial prototype. This step was repeated 2 times until the size of the device was fine-tuned in an optimal way.

The last step involved making the cylindrical shape out of PVC water resistant plastic. Not much effort has been put in optimizing the weight of the device. It was made from materials that were available at that time at the workshop of the university. The work was done by the employees there.

The next figure shows a vertical section of two sections of the sensor node. It shows how the design decision has been made so that when joining them they will form a watertight seal. There are basically two important dimensions. These are marked in the figure as D1 and D2. One section is made of a cylinder which has its bottom outer diameter D2 and its top outer diameter D1. They are designed in such a way that they would fit into one another (D2<D1).

Each section comes at the bottom part with a rubber O-ring. When joining together two section this O-ring acts as a good water sealant. Furthermore after uniting 2 modules, they are to be fastened together with screws (blue color in the figure). The screws are made of

metal and they are corrosion resistant so that they can withstand the harsh conditions of the ocean waters. The position of the screws is above that of the O-rings. This has been done so that there would be no holes drilled inside the compartment that is to be water-tight. In case any water goes in through the screw hole it will stop when it gets to the rubber O-ring.

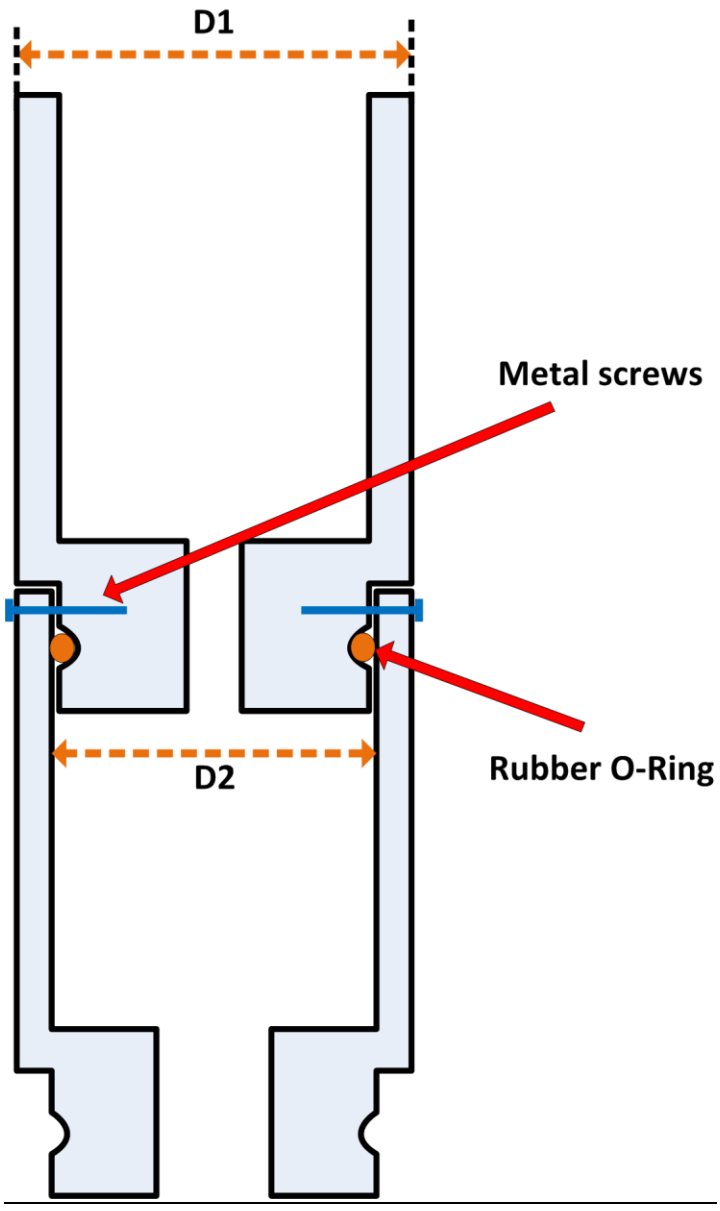


Fig. 4.3 – Vertical section through the sensor node

This design was proved to work and was tested to be water-tight, therefore the final protoype has been realised in this way.

The next figure shows a side-by-side comparison with what has been initially designed on a computer and a picture of what has been realised in the end. As it can be observed, this was a case when expectations met realiy and what has been theoretically designed at first, has been implemented, tested and proved to work in the end.

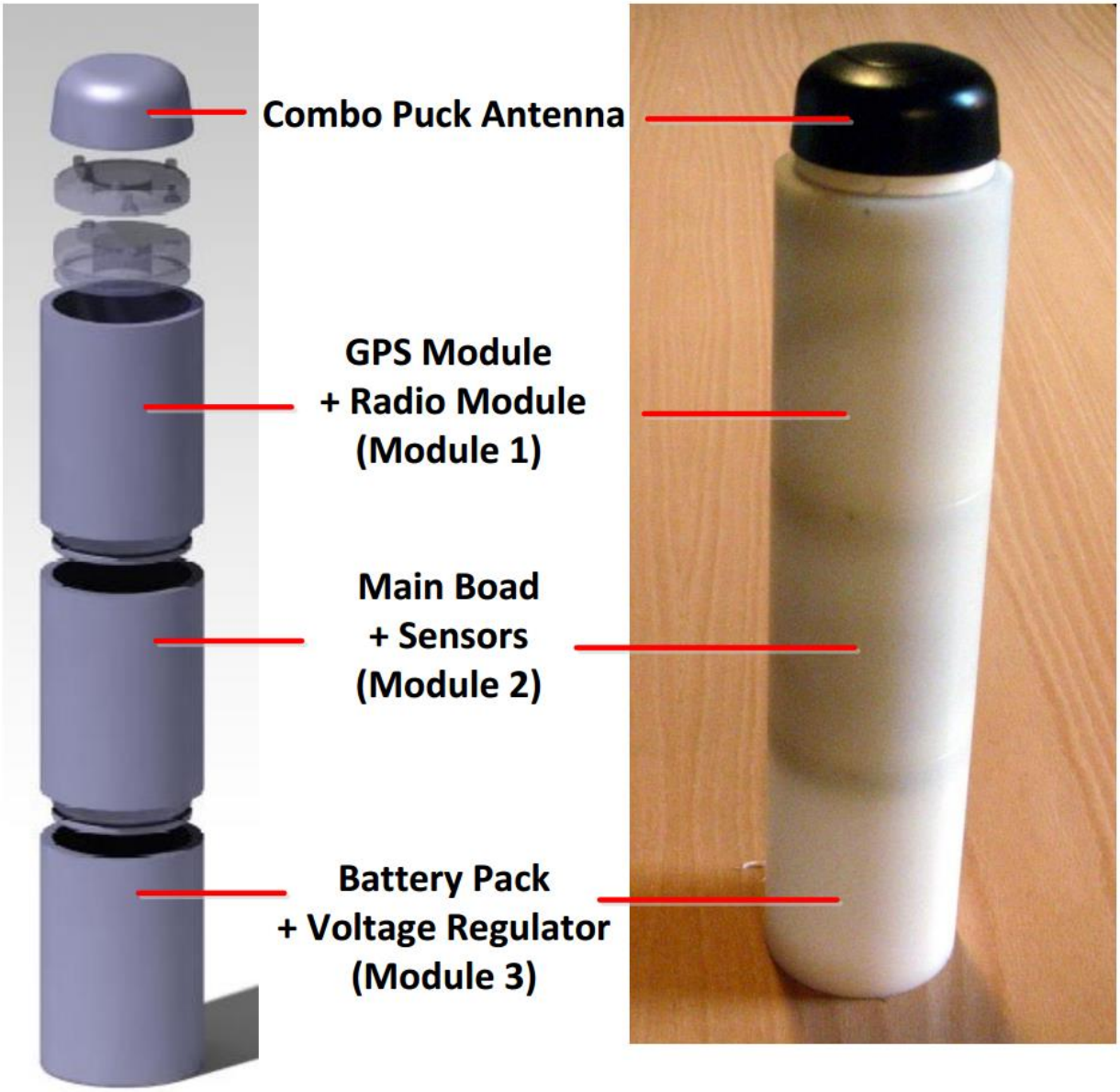


Fig. 4.4 – 3D Graphic design of the device (left); Manufactured prototype (right)

The tests carried out to prove the water-tightness of the device were of 2 kinds. First test was to fill it with water and then place high sensitivity paper around the joints. If the test would fail and any drop of water were to reach those pieces of paper it would have been clearly observed. The device was left for around 10 hours under normal pressure and temperature conditions and it was a success.

The second test carried out involved completely submerging the device in water and again, live if for around 10 hours. This time, humidity indicator paper was placed inside the sensor node and the results were similar, it proved to be water tight. This test was carried out with the device submerged only a few centimeters under the water.

The device is modular, therefore each section has a specific purpose. The developed prototype has 3 main sections (as it can be seen in the previous image) which will be described in detail in the next subchapters.

An overview of the entire hardware can be seen in the next figure.

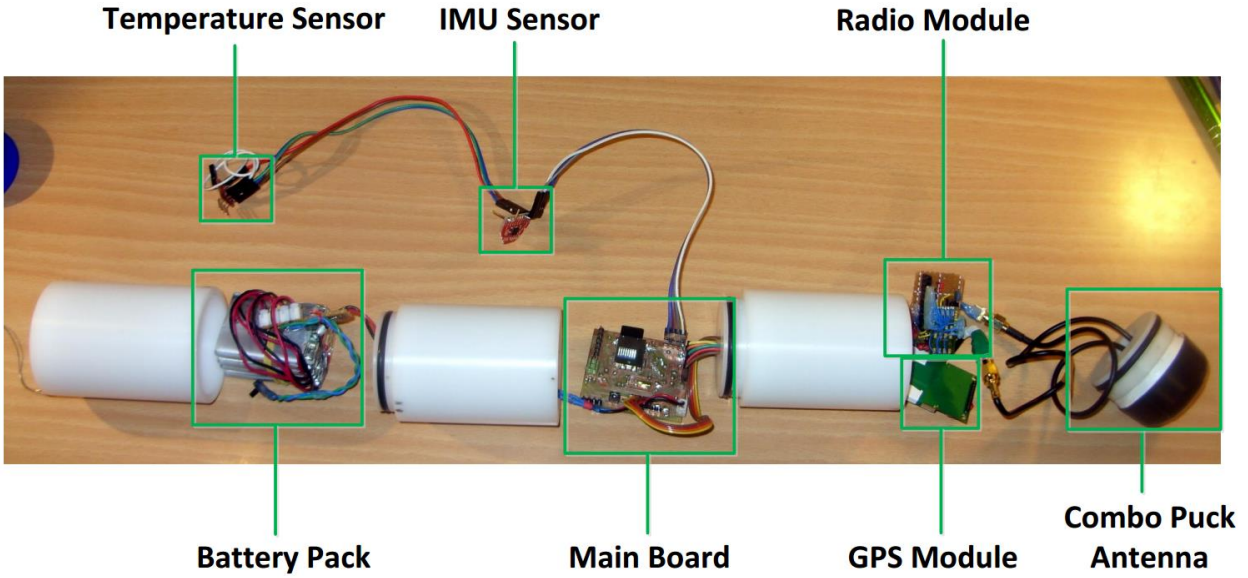


Fig. 4.5 – Sensor node prototype and attached components

4.3.2 Top Module Description (Module 1)

The top module of the device contains 3 main parts: the radio communication module, The GPS module and the antenna for these.

GPS Module

The GPS module used in this project is a generic one (GPS-610F provided by RF solutions) It has standard UART interface and the protocol used is NMEA. The reason behind this choice for this kind of module is its ease of working with and the fact that it can support an external antenna (passive or active). The main GPS chip is already soldered on an interface board and it offers easy access to the most important pins.

The downside of choosing this particular model is that it does not have a power-off pin to shut it down for power saving purposes.

Radio module

In choosing the appropriate wireless radio communication module some investigation has been made. To start with, the output power of the module and the frequency at which it worked had to be open and regulated and approved to work in Europe. Also it should not interfere with other frequencies that are used in the UAV for telemetry.

The following table (*Table 1*) shows a number of devices that were considered for this project. Very detailed comparisons and tests for the performance of all of these was not carried out since it is beyond the scope of this current project. Just the basic characteristics were considered when the choice was made. Cost, range, power consumption and, very important, module and antenna size and shape were key factors that influenced the final decision.

Option	Name	VCC	Power - TX	Power - RX	Range	Data rate	cost (euro)
1	XBee-PRO 868	3.3V	500mA	65mA	40km	24kbps	65
2	3DR Radio 433Mhz "Air" module (Europe)	3.3V	100mA	25mA	1.6km	250kbps	25
3	Zigbee/802.15.4 865/868LP RF	2.7 to 3.6 VDC	48mA	27mA	4km max	80kbps	20
4	RADIOCRAFTS - RC1280	2.8 to 5.5 VDC	28mA	21mA	1-2km	4.8kbps	60
5	RADIOCRAFTS - RC1280HP	3.2 to 4.2 VDC	700mA	21mA	5-6km	4.8kbps	80

Table 1 – Main characteristics of several radio modules

Out of the 5 possible devices some were immediately not considered: Module number 2 from the table is the same module used for the UAV telemetry therefore it was the first to be removed. Also with that, the 433 MHz frequency became unavailable. Even though several devices can use the same base frequency, to try to avoid as much as possible interference it is good not to have many devices running at the same frequency.

It is reasonable to say that the remaining devices were of 2 categories: XBee/Zigbee modules and Radiocrafts modules.

The main difference between these is the protocol they are using. Module number 1 and 3 use an XBee - ZigBee based protocol and the Radiocrafts module use an embedded RC232 protocol.

It is well known that XBee/ZigBee type modules are mainly used when extra network functionalities are needed. They provide functionalities for devices to work as Coordinator, routers or end devices. Besides all this, they have different topologies that can be adapted to many scenarios.

Our current usage scenario does not require that much protocol overhead and configuring such devices to fit our current needs might be unpractical so the more robust and lightweight protocol of the Radiocrafts module was considered to be a better option. Therefore the selected

module to be used was the Radiocrafts - RC1280 module. The high power one is more expensive both money-wise and power-wise. Range and data loss tests for the RC1280 module have been performed and will be described in a next chapter.

Another feature that was taken into account when choosing this module was the fact that its interface is a standard UART interface. Although the datasheet specifies the working voltage for the UART interface of the module as being 3.3V tolerant the truth is that the UART levels are at 2.7V. Therefore a level shifter needs to be used to be able to properly interface the module (see Appendix A).

The next figure *(Fig.4.6)* shows the main differences between these 2 protocols.

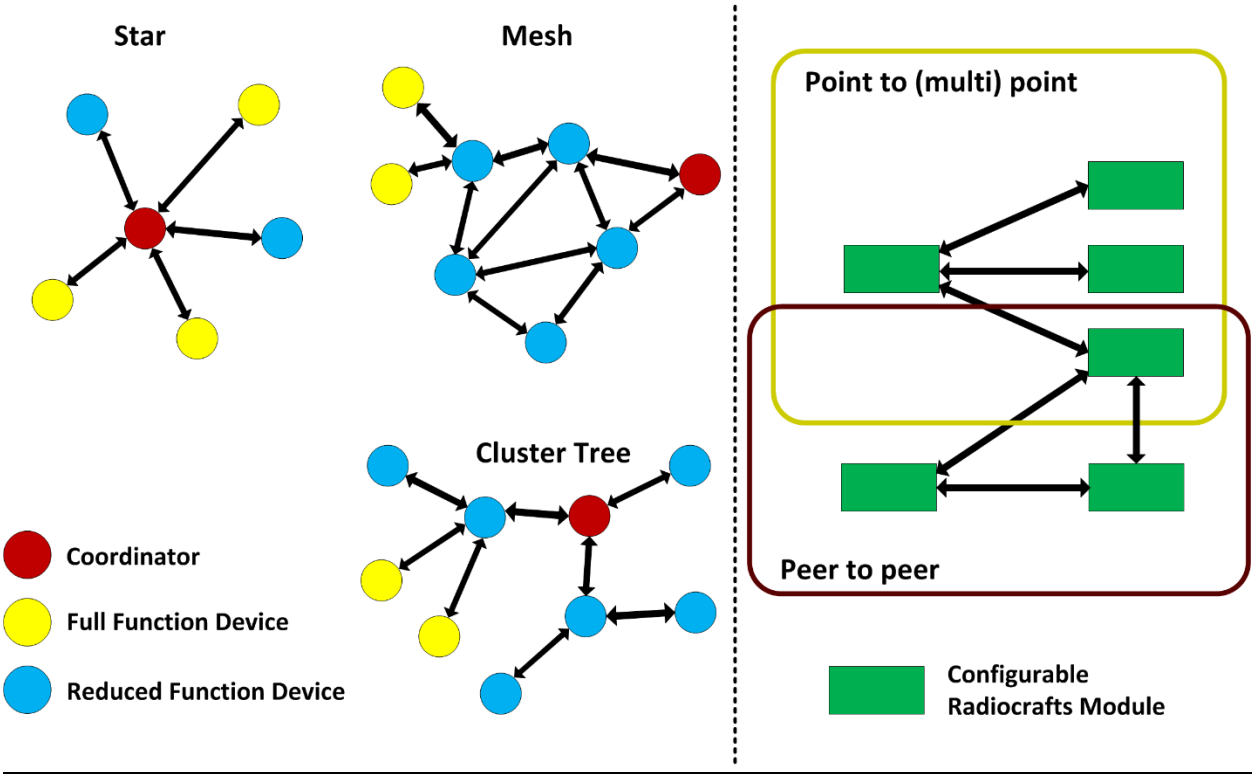


Fig. 4.6 – Zigbee network topologies (left) and Radiocraft network topologies (right)

Antenna

The antenna used is a GSM combo puck antenna. It supports a variety of frequencies and it encapsulates two physical antennas in one package. One was used for the GPS module and the other one for the radio module. This kind of package was suited for the required space constraints of the sensor node.

4.3.3 Main Controller Module (Module 2)

The main controller board of the embedded device has at its core the ATxmega 192C3 8bit microcontroller. The board size is 5cm x 4cm and it is a 2 layer standard width circuit board. The board has been custom designed to be used in the sensor node as it was made to be as small as possible and to offer a good modularity and adaptability in the future. The manufacturing of the custom PCB was done using a milling machine, therefore several limitations on vias under chips exist. All the details (schematic, PCB Layout and detailed pictures) can be seen in appendices.

Besides the main microcontroller the board has a SD card slot, a RTC oscillator and a RS-485 bridge circuit. Most pins of the microcontroller have either been separated and designated to a dedicated module or bus or have been laid out and exposed on the board for future expandability.

Two prototypes of the board have been designed and soldered. The second version is the version currently in use and it is an improvement of the first variant. Still, both versions of the PCBs have been tested to work and partially work (the first one having some limitations) and that will be presented later on.

The board is powered directly by a regulated 3.3V source. All the pins of the microcontroller that are to be used to connect the different modules are placed near the edge to facilitate the connection of other devices to it. Power and ground to the external components connected to the board have also been placed near the edge. A programming header is also present on the board making it easy to flash and debug.

In this module of the constructed prototype some sensors have also been placed: a temperature sensor and an IMU (accelerometer, gyroscope and magnetometer).

Both sensors are connected on the same I2C bus and they act as 2 slaves while the microcontroller is the master. Both of the devices have been tested and proved to work using the same shared bus.

The main application in which an IMU sensor can be used in such a device is for calculating wave amplitude and frequency. This kind of application can also be known known in the scientific community as localization based on accelerometer and gyroscope. [28][29].

The next graph *(Fig.4.7)* shows the raw, unfiltered data sampled from the gyroscope. The sample rate is 5Hz. It is easy to observe that the data is nowhere near the point at which it can be used for meaningful wave height calculations. Filtering of the data and other compensations are required to achieve that.

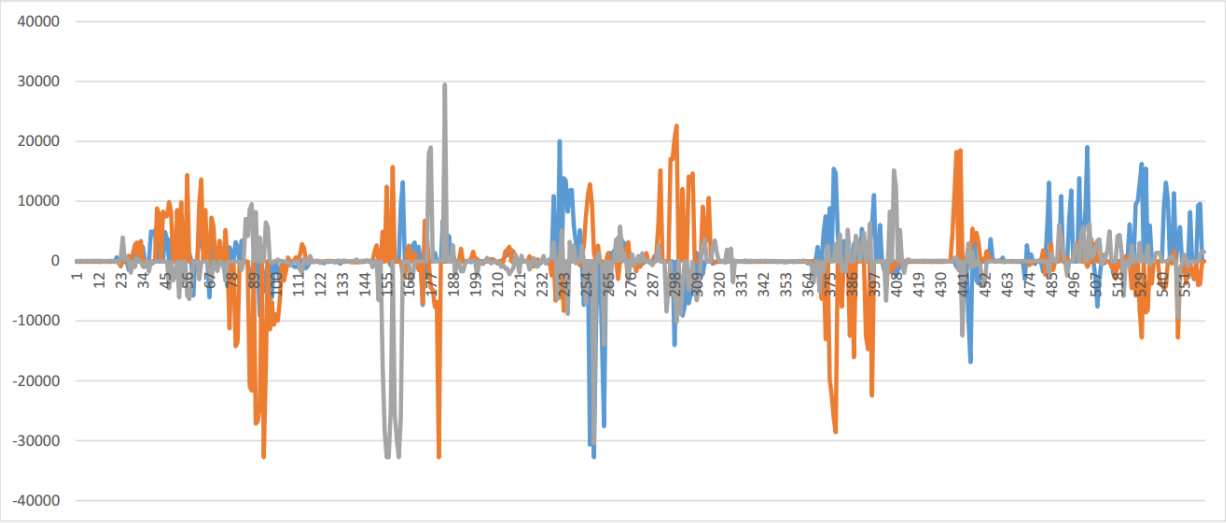


Fig.4.7 – Plotted gyroscope measurements

An example of how an algorithm for relative position calculation can look can be seen in the following figure *(Fig.4.8)* It is largely based on the work seen in [28] and [28] and will not be presented in greater detail since it's beyond the scope of this work.

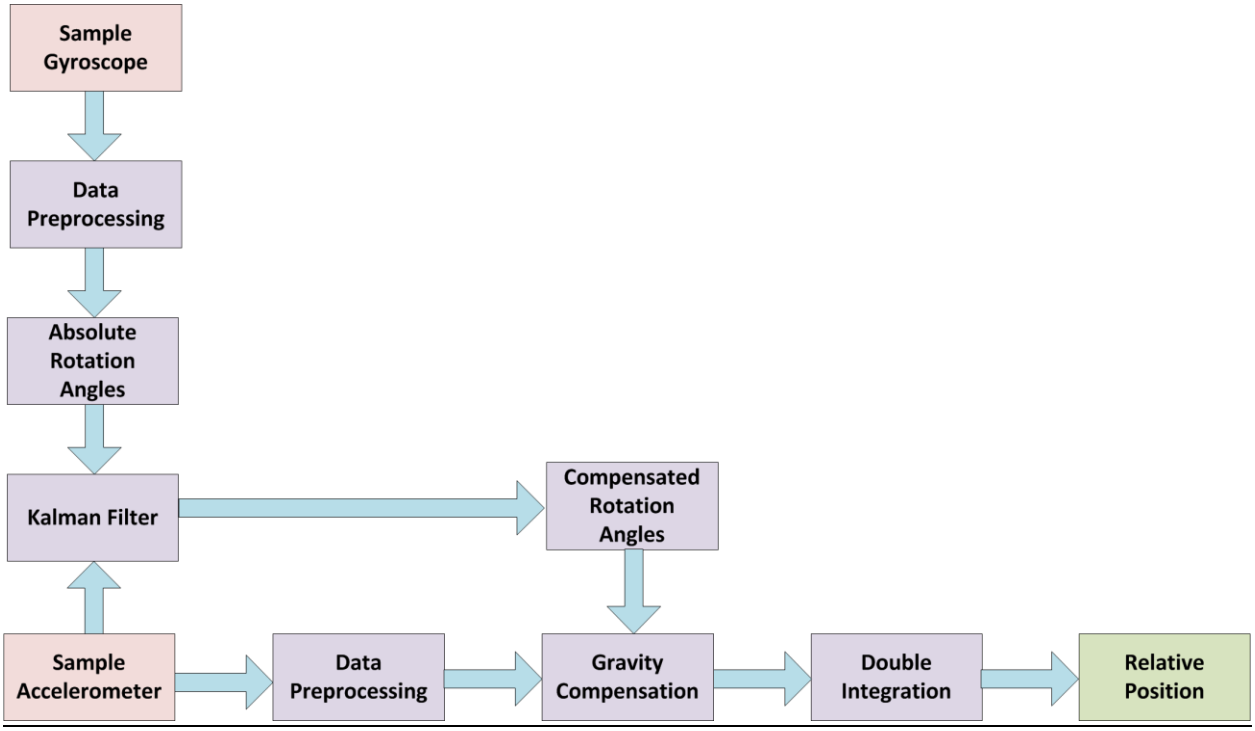


Fig..4.8 – overview of relative positioning algorithm

4.3.4 Power Supply Module (Module 3)

The third segment built as part of the embedded device is responsible for supplying all the circuits. It has a battery pack and a 3.3V voltage regulator. (see appendix A)

In choosing the voltage regulator two options were considered:

The first one was to use a step-down converted and the second one was to use a low-dropout linear regulator. Both of these options were compared and tested.

The next table shows measured and calculated values for the current consumption of the devices. Some of the values were measured in the lab and where measurement was not possible in a straightforward way, the datasheets values were used.

Component(s)	Mode/Characteristics	Consumption (mA)	Measurement Method
GPS module	no attached antenna, searching for sattelites (Enhanced-mode Acquisition)	74	Lab measurments
GPS module	attached active antenna, searching for sattelites (Enhanced-mode Acquisition)	88	Lab measurments
GPS module	attached active antenna, searching for sattelites (Tracking mode lock on 6 sattelites)	66	Lab measurments
GPS module	module in RESET mode	37	Lab measurments
IMU sensor	normal operation mode	4	Lab measurments
Radio module	receive mode + consumption of level shifter	26	Lab measurments
Temperature sensor	normal operation mode	<1	Datasheet
RS-485 bridge	no load	2	Datasheet
All components	normal sampling and transmitting mode, + linear regulator, no writes to SD card	220	Lab measurments
All components	normal sampling and transmitting mode, no linear regulator, no writes to SD card	180	Datasheet
DC Step-down converter	efficiency is minimum 90% depending on the charge	varies with voltage	Datasheet
Linear regulator (LDO)	Quiescent Current max	12	Datasheet

Table 2 – current consumption for some of the devices

A comparison on the voltage regulators used can be made using the table above.

- Computing power loss for the step-down converter:

If the efficiency is 90% then it means 10% (L) of the energy gets wasted.

This means:

$$L \times V \times I_{total} = \text{DC-loss}$$

$$10\% \times 3.7V \times 220mA = 81.4mW$$

Where:

L - The percentage of energy lost
V – Battery voltage
Itotal – Measured current consumption of the entire device (see Table 2)

- Computing power loss for the step-down converter:

$$V \times I_{total} - V_{cc} \times I_{device} = \text{LDO-loss}$$

$$3.7V*220mA - 3.3V * 180mA = 220mW$$

Where:

V – Battery voltage
Itotal – Measured current consumption of the entire device (see Table 2)
Vcc – 3.3V required to supply all the circuitry
Idevice – calculated current consumption without using any regulator.

This means that by using the step-down converter the power wasted is only 40% when compared to low dropout regulator.

4.4 Conclusions on hardware

This chapter has been dedicated to the most important hardware components that are used in the design of the sensor node. It should also be pointed out that all sensors and peripheral devices are chosen in such a way that they offer various sleep modes of operation. This, in combination with the sleeping modes of the main microcontroller make a device that is low power and runs at fully capacity only when needed. The rest of the time it just sleeps in order to save energy.

5. Software Design and architecture

5.1 Introduction on software architecture

It is important to understand that in today's software projects code modularity and reusability has the greatest importance of all time. Because of how much the hardware has evolved, speed simply is not the only parameter that evaluates one's piece of code but rather modularity, reusability and expandability.

To try to satisfy these characteristics there are many different paths to follow. For example having fixed and well-known design patterns into your software represents a huge leap forward into making your piece of code more understandable and modular. There are many popular software design patterns today but most of them apply strictly to object oriented high-lever programming languages. Nevertheless there are still some design patterns and OO-language specifics that can indeed be implemented in pure C language and used in embedded software [10].

5.2 Design patterns

Design patterns are not finished designs that can be converted into running code. They are mainly a description or a template for how to solve the problem. They provide a general solution to most common problems in software. The whole challenge becomes how to reduce your problem into a well-known problem that has already been solved by the community and then apply a design pattern on that known problem.

The sensor node has at its heart an 8/16-bit microcontroller, therefore the programming language for that is embedded C and/or assembly language. This means that certain object oriented languages specifics cannot be used. However with careful programming and well written code, to a certain degree some very important OO-languages properties can be obtained.

Let us take, for example, the concept of public and private to functions. The C compiler does not know about abstraction or encapsulation so, we have to enforce these by imposing some set of rules that each programmer must respect [8]. The first rule for achieving a public/private type of function behavior in C is to place the declaration of public functions and variables in a separate file from those which are private to a module. Another rule is to only allow the use of headers from public declarations in the *#include* directive. And, a third rule, which is a bit difficult to respect in a large software project is to embed the type of the function in its name. This is also called as the Hungarian Notation [11]. Also, "*private*" and "*public*" words can be added to header, source and function names. This way it becomes very clear when you are violating private and public modifiers when importing the header files and using the functions.

5.3 Software architecture on the embedded device

Taking all these issues into account one design pattern very much used in the software engineering filed today can be easily adapted to our need. It is the observer pattern [13]. This design pattern is used in system where there is a subject (in our case coordinator) which has several observers registered to it. Each observer must satisfy a well-defined form and must have some pre-defined functions. In our case an actual observer is a concrete sensor attached to the device and all sensors, no matter what their functionality is, must implement an abstract interface that has functions to configure and control the sensor. The coordinator then holds a list of these registered observers (devices) to it and when it wants to control them it just iterates through the list and call one of those generic functions. It is not the task of the coordinator to know each particular detail of each attached device but rather, it is the responsibility of the device to implement and use a well-defined interface that is to be used and known by the coordinator. All of the concrete sensor objects can be added and removed to and from the list of sensors dynamically, without code recompilation. This is a clear advantage over standard solutions because, if something bad happens to a sensor it can be simply removed from the list, without requiring to reprogram the microcontroller.

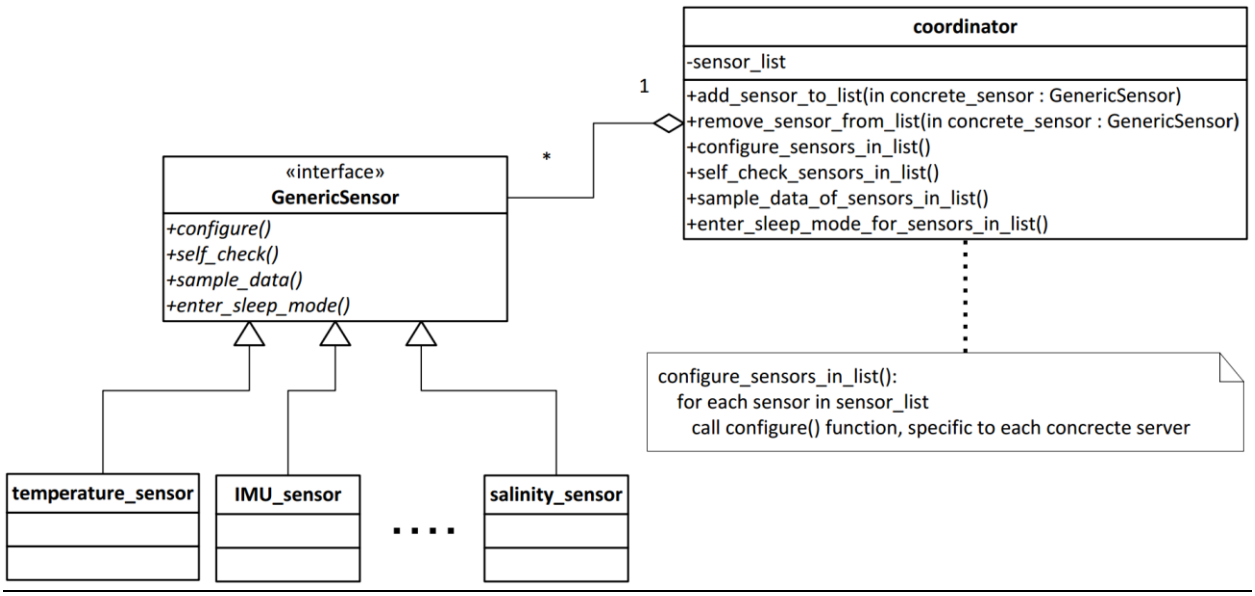


Fig. 5.1 – UML diagram for the observer design pattern

The figure above (*Fig. 5.1*) shows a UML diagram of how observer pattern is to be implemented on the embedded device but not all the implementation details are included. This pattern is mainly applied in object oriented programming languages but there is a way to implement it in C [12] and, therefore it can be implemented in embedded C and the code for it can run on the microcontroller.

The above example is only an illustration, in real case implementation, several sensor lists may exist, each of them categorizing sensors based on their usage and on the interval at which they are sampled.

It is important to mention that the configuration parameters transmitted to a concrete sensors are not directly transmitted by the coordinator. They are read from header files specific to each sensor and/or from a configuration file found on a non-volatile memory storage system (each sensor node has a micro SD card with FAT file support).

Even in the case that the coordinator issues a command to sample data for all the sensors, the actual data might not be transmitted back to the coordinator but, written to a file on the storage device or sent via the wireless medium. Basically, the code for the coordinator is written in the main source code file of the project and the files specific for sensor are written in header files and configuration files found on the storage media.

This kind of code organization leads to better debugging and in the future, with a proper documentation it will be very easy to add new functionalities to the device.

To prove the feasibility of such an architecture a sample code has been written and tested. (See attached digital content) The current sample code has been tested on an x86 machine and implementing it on the microcontroller is one of the future tasks. Other alternatives for implementing this will also be treated in a next chapter

In comparison to other embedded devices used today in practice as seen from the perspective of a real time operating system in our device, even though a fully working real time operating system will not be used, particularities of a RTOS are to be implemented. For example we would like to have the concept of "concurrent" tasks, non-blocking functions, some memory allocation capabilities (when reading and writing to and from external flash) and, most important, separation of driver code and application code. Where and how these particularities are important in this current system will be described in a later chapter.

5.4 DUNE

DUNE Unified Navigational Environment is a runtime environment for software found onboard of vehicles. It is developed by the Underwater Systems and Technology Laboratory (LSTS) located in Porto. It offers an architecture and operating system independent software framework for navigation and control of specific unmanned vehicles such as UAVs. It basically represents a composition of tasks specifically designed to be loosely coupled, to work independent, concurrent and to communicate with each other. The framework is based on POSIX threads, therefore it requires an underlying operating system that supports that. Using DUNE for the embedded node is out of the question (we have an 8/16-bit microcontroller not a UNIX capable device) however, DUNE will run on the UAVs designed to communicate with the sensor node and therefore a certain communication protocol is required between the two. The sensor node must communicate with any device running DUNE [14][15].

In order to make this possible a new kind of DUNE component (task) is to be coded. This component is used for communicating with the sensor node and to act as an intermediate between the sensor node and the end-user software, NEPTUS. Any device running DUNE that wishes to communicate with the floating sensor node must have a running task specifically for this.

Each DUNE task communicates with another DUNE task inside the same environment or to another device running DUNE using the message passing concept. All tasks must comply with a well-defined communication protocol and on the message bus only messages that correspond to that format exist. The communication protocol used in DUNE is called IMC and will be described in a later chapter.

What is important to mention here is the DUNE driver component that is to be implemented and used to communicate with the sensor node. It serves 2 main purposes: First of all it takes care of the communication between the actual sensor node and the device running DUNE. The communication is external to the device and it is done wireless. The device running DUNE is also using the wireless communication module described in a previous chapter, therefore the first job of the DUNE component is to have a driver that is used to interface the wireless communication module. The second job of the DUNE component is to read and

write data and commands to and from the sensor node and to relay the data to NEPTUS. It is basically like an interface for the NEPTUS to communicate with the sensor node.

5.5 NEPTUS

Neptus is a Command and Control software used to monitor and control unmanned systems. It is written in java making it cross-platform compatible. Neptus is used as an interface between devices running DUNE and the end-user. It communicates with such devices using the IMC protocol (which will be described later), making it compatible and interoperable with any type of device running DUNE. A key advantage of proceeding this way is that at the end you have a unified way of communicating with heterogeneous classes of autonomous vehicles and sensors.

Neptus is a framework that has the characteristic of being adaptable and flexible. Plugins can be created for particular needs and can easily be integrated with the main program source. Plugins can even be added as already compiled java .jar files. This way a convenient way of hiding code and intellectual property is achieved.

5.6 Conclusions on software

In conclusion the software part of the project is split into 3 main categories:

1. The software that runs on the sensor node, responsible for interfacing the sensors and measuring data.
2. The software that runs on the UAV under DUNE that is to communicate with the sensor node.
3. NEPTUS Application used to monitor the sensor from the end-user perspective.

The first two in the list are to communicate using a very well defined protocol. Since all UAV communicate with the sensor node using the wireless medium a lot of issues arise here: Data loss? What happens when a UAV is in reach of 2 or more sensor nodes, with which one does

it communicate? And what happens when you send a fleet of UAVs, for example quadrocopters to pick-up several floating sensor nodes from the ocean? One single sensor node could communicate with 2 or more UAVs.

These protocols used to address these issues will be described in the next chapter.

Viewed from a higher level an overall architecture of the system is presented in the next figure *(Fig. 5.2)*. What has been described in the previous paragraph (the DUNE task) is represented by the green block in the figure.

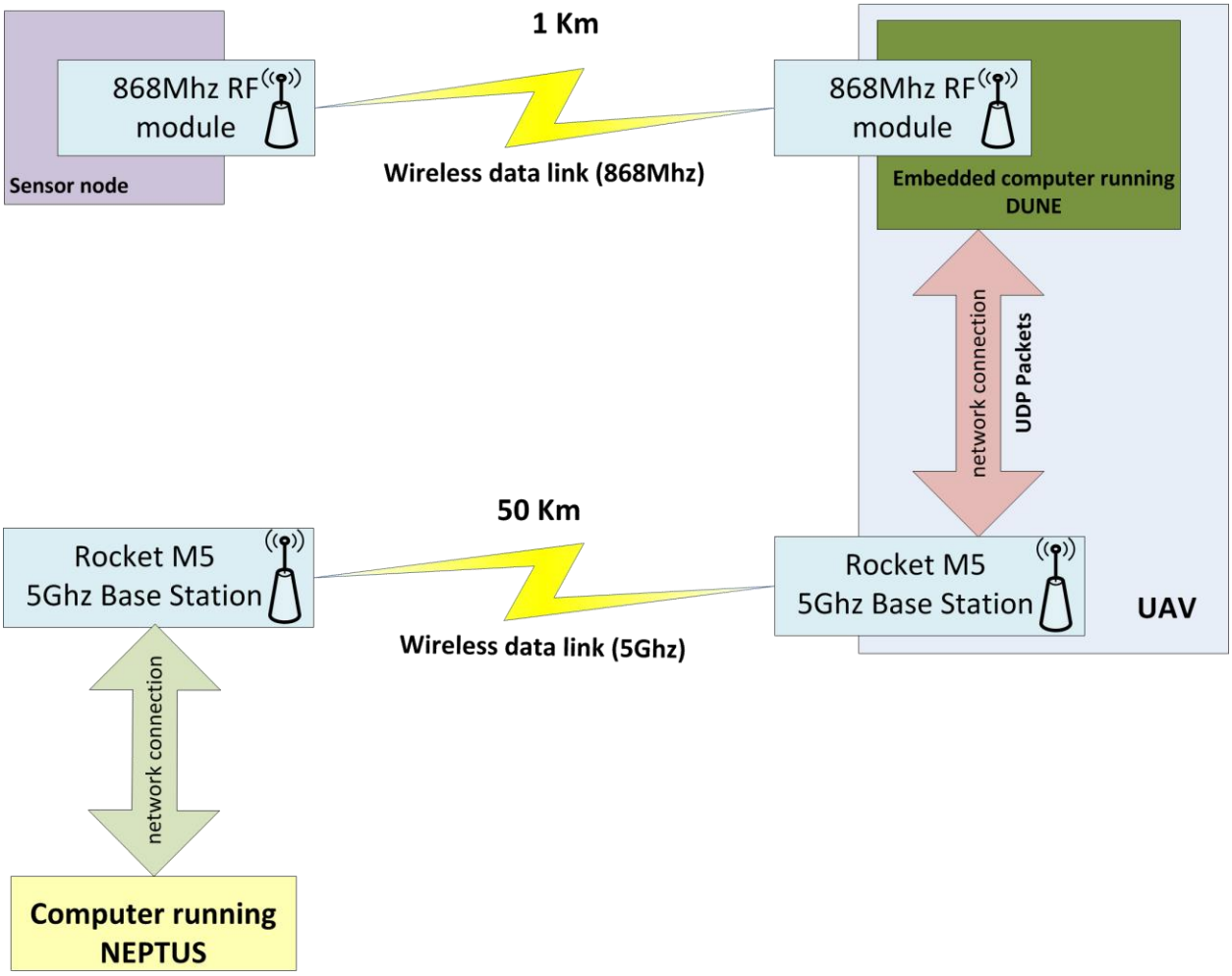


Fig. 5.2 – Communication flow between the sensor node, the UAV running DUNE and a computer running NEPTUS

6 Communication protocols

6.1 Introduction on communication protocols

A communication protocol is defined as a system of digital rules for exchanging message within or between computing nodes. Any kind of software or hardware application must use clearly defined communication protocols. These protocols must define a syntax, must have a clear semantic and some sort of synchronization mechanisms.

In this current project there are 2 main areas in which communication protocols have relevance. One is between the sensor node and the UAV and the other is between the UAV and NEPTUS application.

6.2 IMC

IMC stands for Inter-Module Communication [13]. It is a message oriented protocol with main usage in networked vehicle and sensor operations. It defines a common message set that all systems understand and use for communication between nodes in the same network, particularly, in our case DUNE tasks. The whole IMC standard is fully documented and designed inside a single XML file which, in turn can be translated using XSLT (Extensible Stylesheet Language Transformations) into different language bindings.

Each IMC message has a defined structure. It is composed of a header, a payload and a footer. An example of a message format is given in *Fig.6.1.* The standard defines a limited number of predefined messages made unique by their ID. All messages must have a fixed header and footer and the payload is the only thing that differs from one message to another. Of course, the standard also allows the creation of custom IMC messages but these, in turn must satisfy certain rules and must submit to a naming convention for their message ID. Some of the predefined messages include: Revolutions per Minute, Voltage, Current, GPS Fix,Acceleration, Magnetic Field, Distance, Temperature, Pressure, Depth, Depth Offset, Sound Speed, Water Density, Conductivity, Salinity, Wind Speed, etc. [16][17].

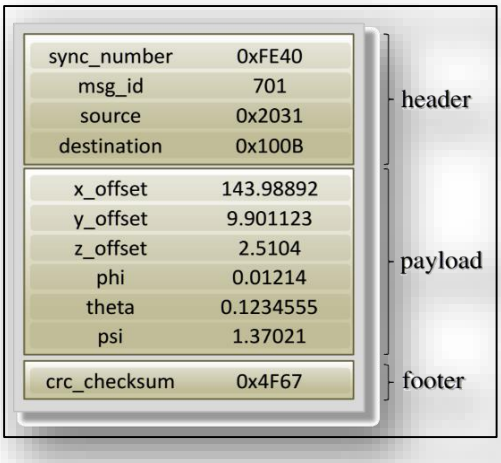


Fig. 6.1 – IMC message example

Having this in mind an overview on the system (from the communication perspective) can be seen in the following figure (*Fig.6.3*). The IMC message bus has various messages dispatched to it among which are messages coming from the sensor node driver (marked in the figure).

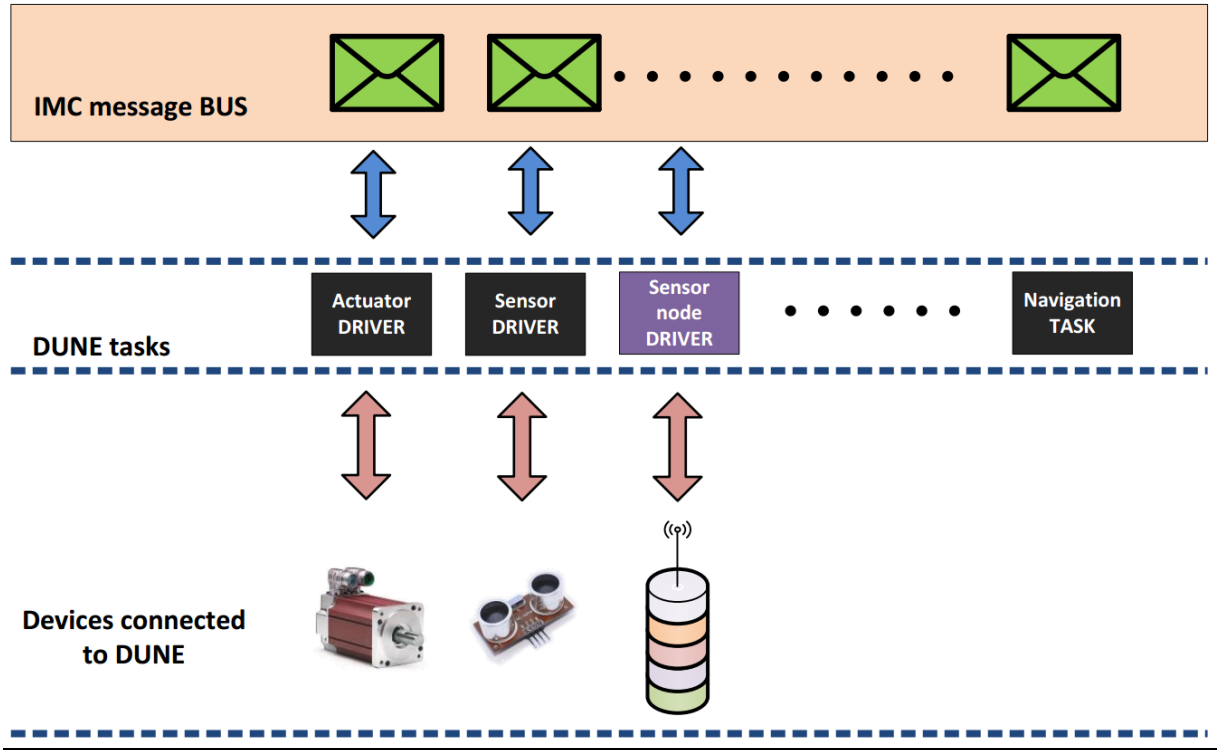


Fig. 6.3 – IMC message bus and DUNE tasks configuration example

In essence, the messages that are found on the bus can come from any source and, for the consumer tasks of the message that are transmitted by the sensor node driver it is irrelevant that the actual data comes from somewhere far away through wireless communication. That processing and parsing of data is strictly handled by the driver itself and is completely transparent for the other running tasks.

With these having been specified it is now clearer why the overall design is loosely coupled and a high degree of modularity is one of the system's strong points.

6.3 DUNE and Sensor Node protocol

The exchange of messages between the two comprise of 2 main message types: data messages and control messages.

Data messages can be from the sensor node to the UAV consisting of currently or previously monitored sensor reading. From the UAV to the sensor node data can represent configuration files to be written on the flash memory (SD card) of the sensor node, for example for modifying measuring intervals and sensor configuration parameters.

Control messages are exchanged between the two before and after any data exchange and most during sensor deployment and pick-up from/to the ocean surface. For example, during pick-up operations, the 2 devices need to know their exact position, orientation and heading in real time. Control messages may also include diagnostics, power-on or power-off commands, etc.

Having this in mind 2 approaches at defining such protocols have been considered:

6.3.1 First option for the protocol (IMC based)

In order to maintain a certain level of consistency, the IMC message protocol described earlier is used as a basis for a new protocol. Each message (data or command) will have a structure similar to an IMC message structure. Although it is not pure IMC protocol the standard is used as a basis for all message formats.

Considering this approach, one advantage arises: There is no need to rethink the new protocol details, since you are using something that has been used before and proved to work and with proper documentation.

The code listing provided in the next figure *(Fig.6.4)* shows how a basic IMC message-type can be implemented on the embedded device. It is a simple temperature message. Note the header and footer data structures that will be the same for all messages and keep their fixed format. This way a basic interpretation of any message can be done without actually reading the payload data and filtering/validation can be done based only on message ID source and destination.

```
01. typedef struct IMC_header_type
02. {
03.  uint16_t sync;    //marks the beginning of a packet
04.  uint16_t mgid;    //Message Identification Number - The identification number of the message contained in the packet
05.  uint16_t size;    //Message size (size)  - The size of the message data in the packet.
06.  uint64_t timestamp; //in the standard IMC this is fp64_t but AVR compilers don't have 64bit floats so we use integers
07.  uint16_t src;    //Source Address - The Source IMC system ID.
08.  uint16_t dst;    //Destination Address - The Destination IMC system ID
09. } IMC_header;
10.
11. typedef struct IMC_node_temperature_type
12. {
13.  IMC_header header;
14.  double data;
15.  IMC_footer footer;
16. } IMC_temperature;
17.
18.
19. typedef struct IMC_footer_type
20. {
21.  uint16_t  crc16;  //Check Sum (CRC-16-IBM)
22. } IMC_footer;
```

Fig. 6.4 – IMC-like structure definition listing

The messages to and from the sensor node(s) and UAV(s) are broadcasted. Each device within range can pick-up and read the message. But the message filtering is not done on the transport or network layer but in the application layer based on the header of these messages. This can be helpful in the sense that new filtering rules can be easily applied to messages without any driver/hardware reconfiguration.

Implementing this solution in practice and making it work raises several challenges. First of data types such as *float* and *double* for the AVR Compiler are the same. They offer the same precision. At the same time, for any capable device running dune, *float* and *double* have different precision. Also memory organization (endianess) is also an issue. The next figure shows how the data is stored in memory on the AVR architecture (left) and on the right we have the data received in DUNE. The highlighted numbers show the difference between big and little endian. Although this is not at all a big issue another approach at implementing the protocol was considered.

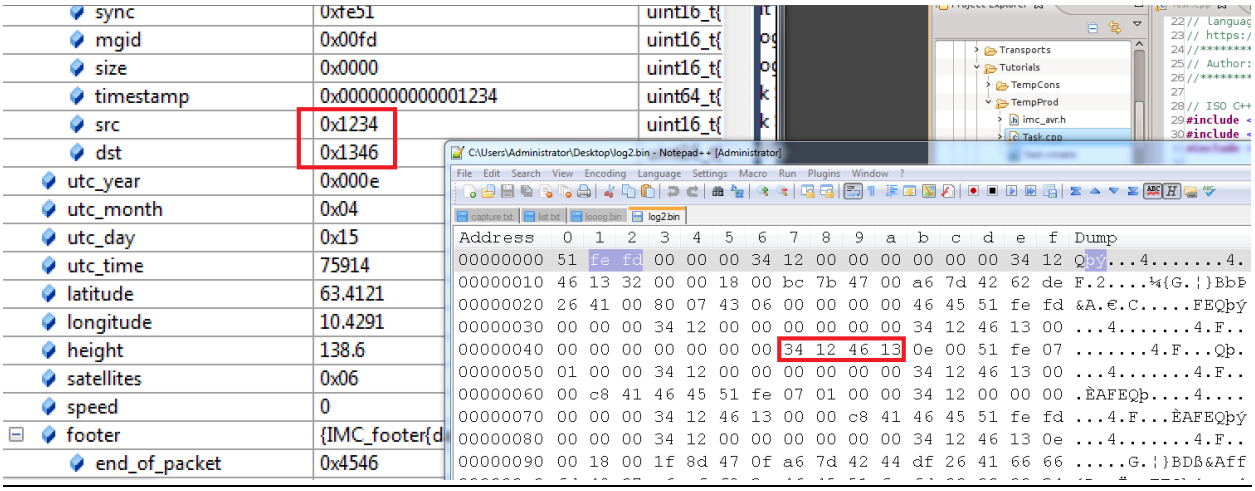


Fig. 6.5 – Endianness issues

6.3.2 Second option for the protocol (IMC-NMEA based)

The second option, which is easier to implement is based on the NMEA GPS protocol [18]. Basically every message is a sentence transmitted as a string of characters, numbers, separators and a CRC checksum field. Every such sentence has the same basic fields as the IMC data fields.

Description and examples:

All messages have the following format:

$msgid,source,destination,data*CRC

- msgid – a message identifier (ex: TEMP, GPSFIX, GYRO, etc.)
- source – a 4 digit base 10 number
- destination – a 4 digit base 10 number
- data - variable for each message type
- CRC – checksum of the package (hex)

IDs lower than 1000 are reserved and considered a broadcast address

Messages from the UAV->Sensor node:

Source is always a UAV ID, UAVs don't communicate using this protocol with each other.

Destination, is the ID of the sensor node to which the data is addressed. It is a number between 2000 and 9999

Messages from the Sensor node -> UAV:

Source is an ID unique for each sensor node.

IDs between 1000 and 1999 are reserved for UAVs

Destination is always a UAV ID, sensor nodes do not communicate with each other.

Examples:

msgid=TEMP

data is a floating point number, 3.2 digits format

example:
$TEMP,2001,0001,025.75*23

msgid=GPSFIX

data is represented by multiple fields

latitude is a floating point number, 4.7 digits format longitude is a floating point number, 4.7 digits format
altitude is a floating point number, 6.2 digits format

example:
$GPSFIX,source,dest,latitude,longitude,height*56
$GPSFIX,2002,0003,0063.4120751,0010.4292460,000148.39*56

In the proposed prototype this second option has been used.

7. Application Modes and Software States

7.1 Introduction

Generally, in engineering (automata theory and computer science) a state of a computer program is a technical term used to described the stored information at a given instant in time. More in particular, a state of an embedded device has, along with the information about the state of the data, the output and input signals of the actual circuit [15].

This chapter is mainly focused on the states in which the embedded device (sensor node) can be and will not focus on the software states of the UAVs that interact with these sensor nodes.

7.2 Basic states description

The next figure (*Fig.7.1*) shows the basic states in which the sensor node can be at any particular moment time. State transitions are not named in the figure but will be described in detail.

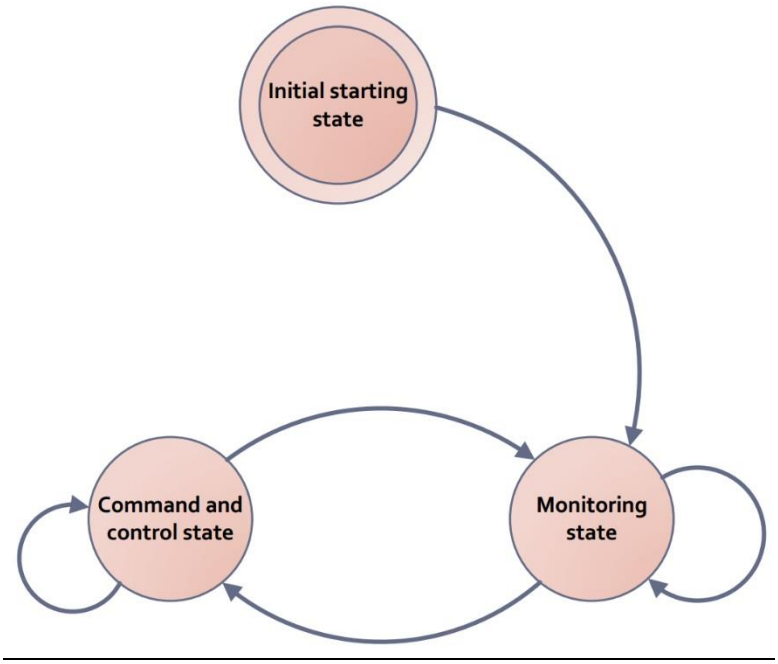


Fig. 7.1 – Basic software states of the embedded device

As anticipated, the system has a well-defined starting state to which it cannot switch back to once it has been through all the processing that is involved in that state (although certain functionalities of this initial state can be re-executed under certain commands and conditions but this will be treated later). This initial state is mainly responsible for the starting up of the system, loading of the drivers, initializing the very important filesystem, configuring the microcontroller frequencies, allocating the necessary memory for the peripherals used and, performing an initial self-check and calibration of the sensors and other peripherals attached to the microcontroller. As mentioned in a previous chapter, most of the configurations used in the system are located in files on the microSD card. If anything fails during this stage there is no point for the device to proceed to the next states and the system enters in a "problem" state or it hangs. This is not represented in the figure. This is why this initial state must be separated from the rest. Basically the ways in which the device can enter the "problem" state are not deterministic and this is out of the scope of the current discussion. We can just assume this state represents any way in which the device is not functioning, including hardware failures, power failures or even physical damage.

The monitoring state is the state in which the device will be most of the time. During this state it constantly samples data from the sensors and stores it for later transmission to the UAV.

7.3 Example scenario for state switching and transitions

Even though a real time operating system will not be fully implemented on the device some OS-like features will be used. One such example can include concurrent "tasks". The device transits from the monitoring state to command and control state only when the UAVs "tells" it to. For this to happen there has to be some sort of polling or interrupt-based mechanism for the node to "know" when it is in range of the UAV and vice versa. Therefore, in parallel with the polling of the sensor task there is another task (interrupt) that runs constantly to check and see if there is a UAV within range. If so, the device enters the command and control state and waits for commands from the UAV. The commands can be as simple as an identification request or as complex as reconfiguring the device parameters or large data exchange. Nevertheless, no matter what kind of commands it receives and executes, or for how long it stays in the command and control states, there has to be this particular and separate working state of the device.

For the device to conserve as much power as possible a good way of detecting when a UAV is in range of it is needed. For this purpose the UAV is the one that continually polls for signals coming from the device and the sensor node, in turns, send basic ping messages at a predefined time interval. The next figure (*Fig.7.2*) shows a scenario in which the sensor node changes states through basic interaction with the UAV.

The timing diagram corresponding to a) represents the moments in which the sensor node samples data from the sensors and writes that to the storage medium.

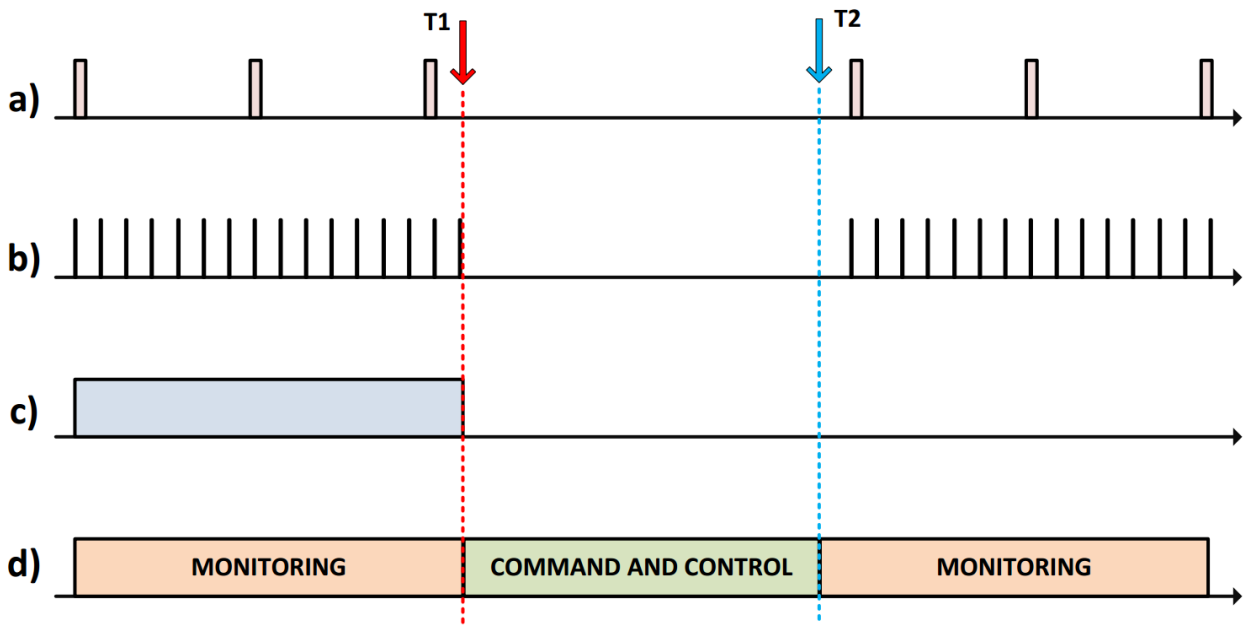


Fig. 7.2 – Timing diagram showing UAV and sensor node basic interaction

The task that sends ping messages for the UAV to be intercepted if in range is marked by b). It should be mentioned that a) and b) in the figure are tasks (or interrupts) running concurrently on the sensor node during the monitoring state. The intervals between the tasks in the figure are only to provide a rough estimate of how the system works. The period between sampling activities – timeline a) can be tens of minutes or even hours and the period between ping messages intended for the UAV are only a few seconds but there may exist times of day in which the node does not send these messages. For example, in order to save power an offline scheduling policy for this time intervals can be made. We can assume that during nighttime the sensors will not be picked up and no UAV will fly above them so, just by taking this into account out of 24 hours in which the device can send these messages we can reduce them to 12, 8 or even 6. Many other variations for these schedules can exist: only send these messages during certain week days or only during odd/even hours, etc.

The diagram c) represents a task belonging to the UAV responsible for polling of the sensor node. Diagram d) represents the states in which the sensor node is.

The scenario described in the figure is as follows: The sensor node is floating on the ocean surface while constantly monitoring data – task a) and at the same time sending messages in the clear, to be picked up by any passing UAV – task b). At the moment T1 the UAV is in range of the device. The UAV stops polling for the node because it found it – task c). The

node also knows that it was found, it stops sending ping messages and sampling data and it enters command and control mode. Between T1 and T2 the sensor node receives and executes commands from the UAV and is in the command and control state. At the end of this time interval T2, the sensor node and UAV have finished exchanging data, the sensor node goes back to the monitoring state and the UAV flies away, this time without polling any more for any new device since it has finished its mission.

Since power consumption is a big issue, the sensor nodes sends ping messages, waits for a response and then goes to sleep. This is done at a rate that is given by configuration files found on the device, usually a few seconds as mentioned before. The time interval between these messages must be carefully chosen, taking into account the minimum speed at which a fixed-wing UAV can fly and the typical range of the wireless module onboard the sensor node. For example if the minimum speed for the fixed-wing UAV is 13m/s and the range of the wireless module is, for example, 1.3km (radius) resulting in a 2.6km diameter range, the UAV would be in range of the sensor for about 200 seconds. This means that the period at which the sensor node sends its ping messages should be at least once every 100 seconds for the device to be detected by the UAV. But these figures are pure theoretical, many parameters such as weather and atmospheric conditions might affect the communication range and in practice the frequency at which the ping messages are sent would be much higher, taking into account that you also need to exchange data between the two and not only detect the floating device.

7.4 Dynamic workflow example

The monitoring state of the embedded device is its main working state and requires a more detailed description. The static representation of how it works was presented in a previous chapter, the observer design pattern is used for that. The dynamic workflow of the device has not yet been presented.

The next flowchart (*Fig.7.2*) partially shows the dynamic workflow of the software on the sensor node, starting from the initial state and proceeding to the monitoring state.

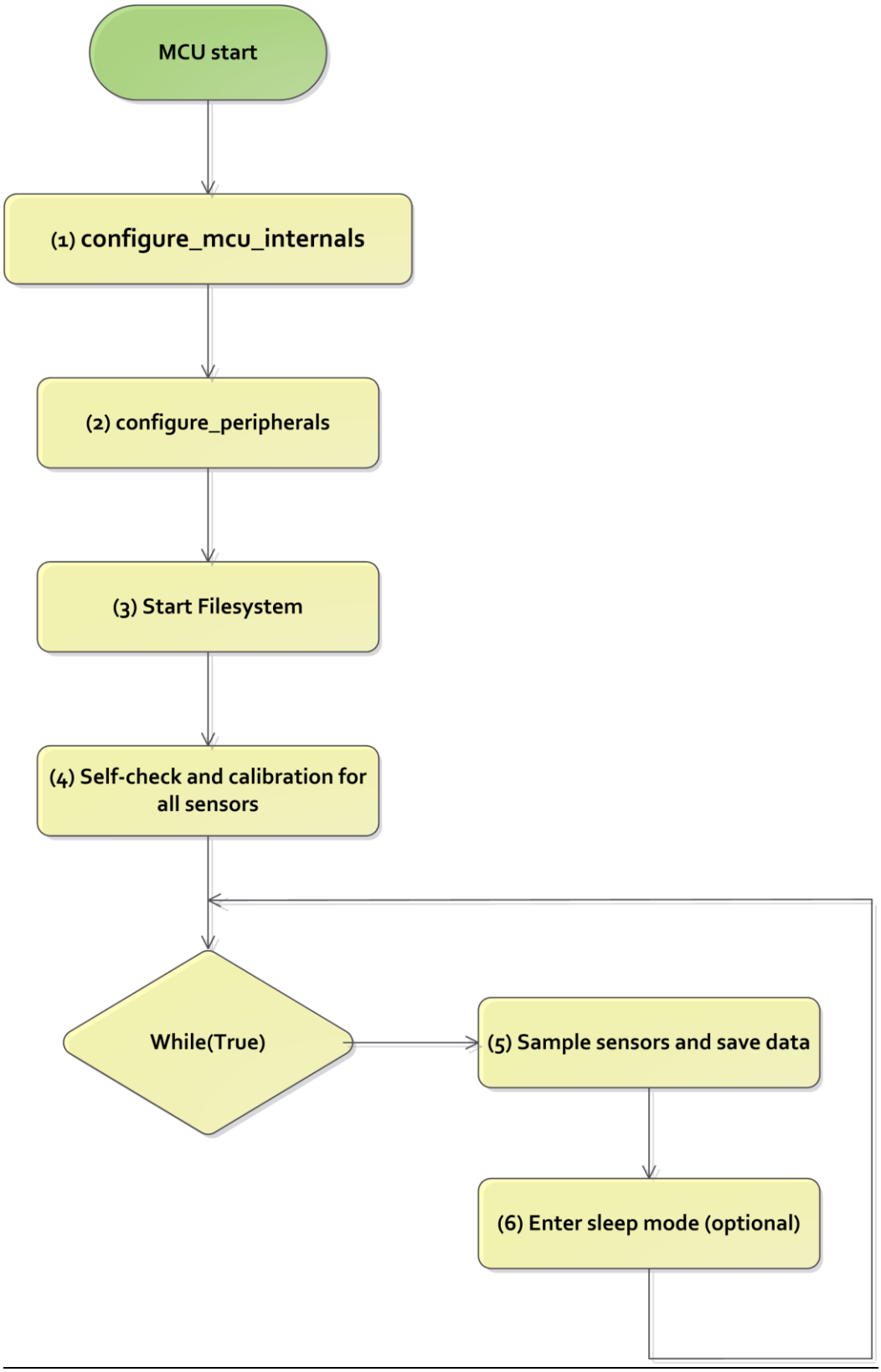


Fig. 7.3 – Basic flowchart partially showing the software workflow on the sensor node

The next few paragraphs describe what happens in each step represented in the previous flowchart, *Fig 7.3*:

(1) Configure MCU Parameters. It is the first set of operations carried out by the microcontroller. It configures the clock systems, basic drivers for peripheral communication, clock and calendar type functionalities, enables and configures interrupt routines and prepares basic boot-strap of the soon to be started tasks.

(2) Configure MCU core devices. In this step the microcontroller loads the drivers for its base core devices attached to it. No sensor node will be without a GPS module, a micro SD card for storage and wireless communication. These peripherals need to be configured and started and that is done in this step.

(3) Configure and start all connected sensors. All the sensors that are to be used in the sensor node need to be configured. During this step the MCU load and configures the drivers for the peripheral communication

(4) Self-check and calibration for all sensors. In this step all the connected sensors to the device have their initial run and sample their first few measurements. For example, for a temperature sensor a self-check function might be measuring the temperature and comparing it to a certain interval in which that temperature should be. If the temperature is not between a reasonable range, it means that something is wrong with the sensor.

(5) Sample sensors and save data. It is the most executed step of all. Here all functional and active the sensors that are connected to the device are sampled and their data is stored on the storage medium, the SD card and/or sent to the UAV

(6) Enter sleep mode. After the sensors have been sampled the device is to enter sleep mode and wake up after a certain pre-defined time. This step is optional, depending on the usage scenario of the sensor node

In practice when the sensor is first started it might be on the ground, even before loading it onto the UAV and it will be dropped in the ocean already started. In this case, step (4) and the example given in that with the temperature of the ocean would apply for the ambient temperature of the location where the node is first started. From the command and control state the execution flow can jump into most of the steps described above. For example, the UAV can give the command to re-initialize the system.

The previous flowchart and descriptions only show and describe the normal workflow of the program. In reality the device is programmed to also handle several interrupts. These interrupts are presented below:

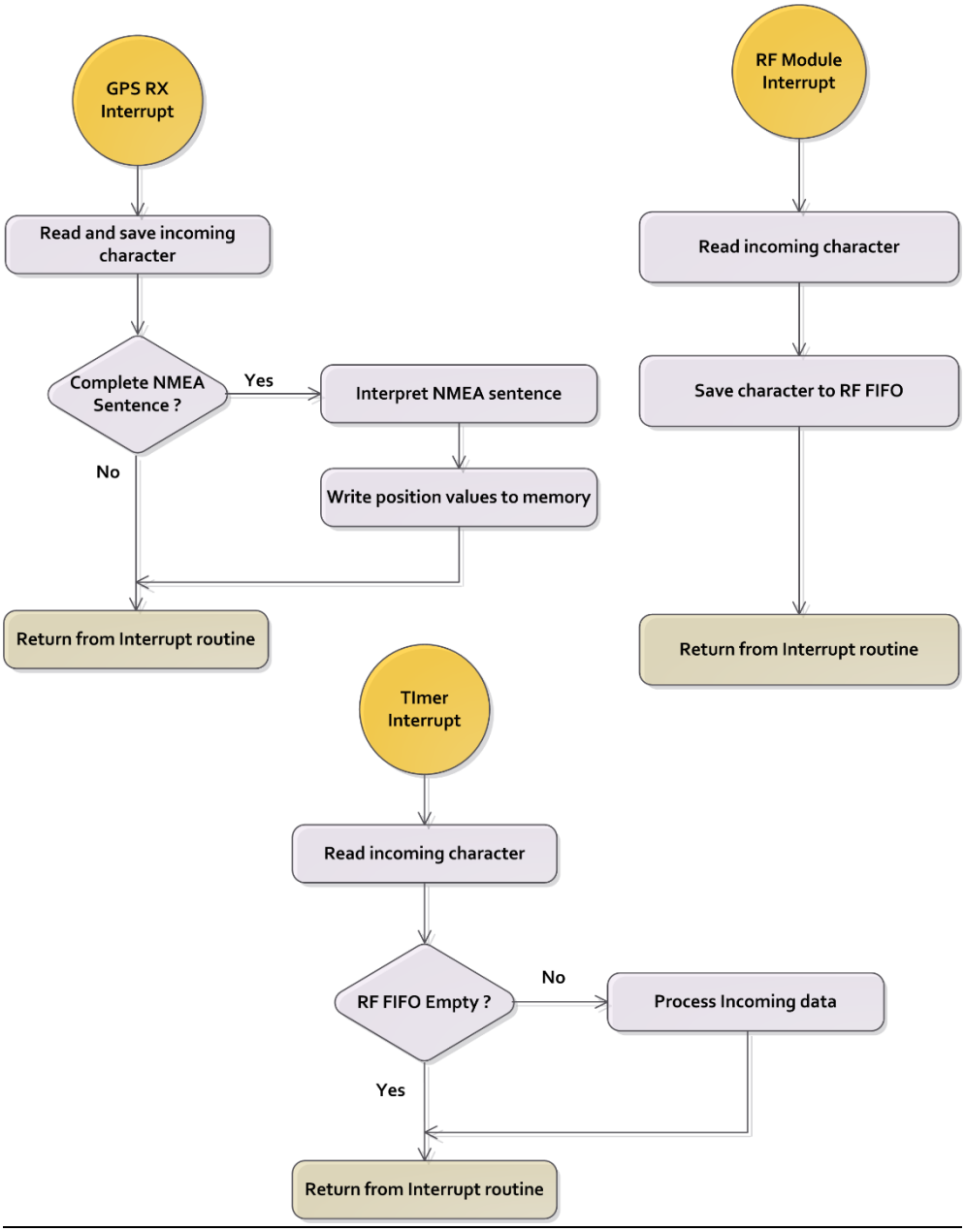


Fig.7.4 – Interrupts and their dataflow in the embedded device

To better illustrate the dataflow of the whole system (including interrupts) a rough scenario is presented in the figure below:

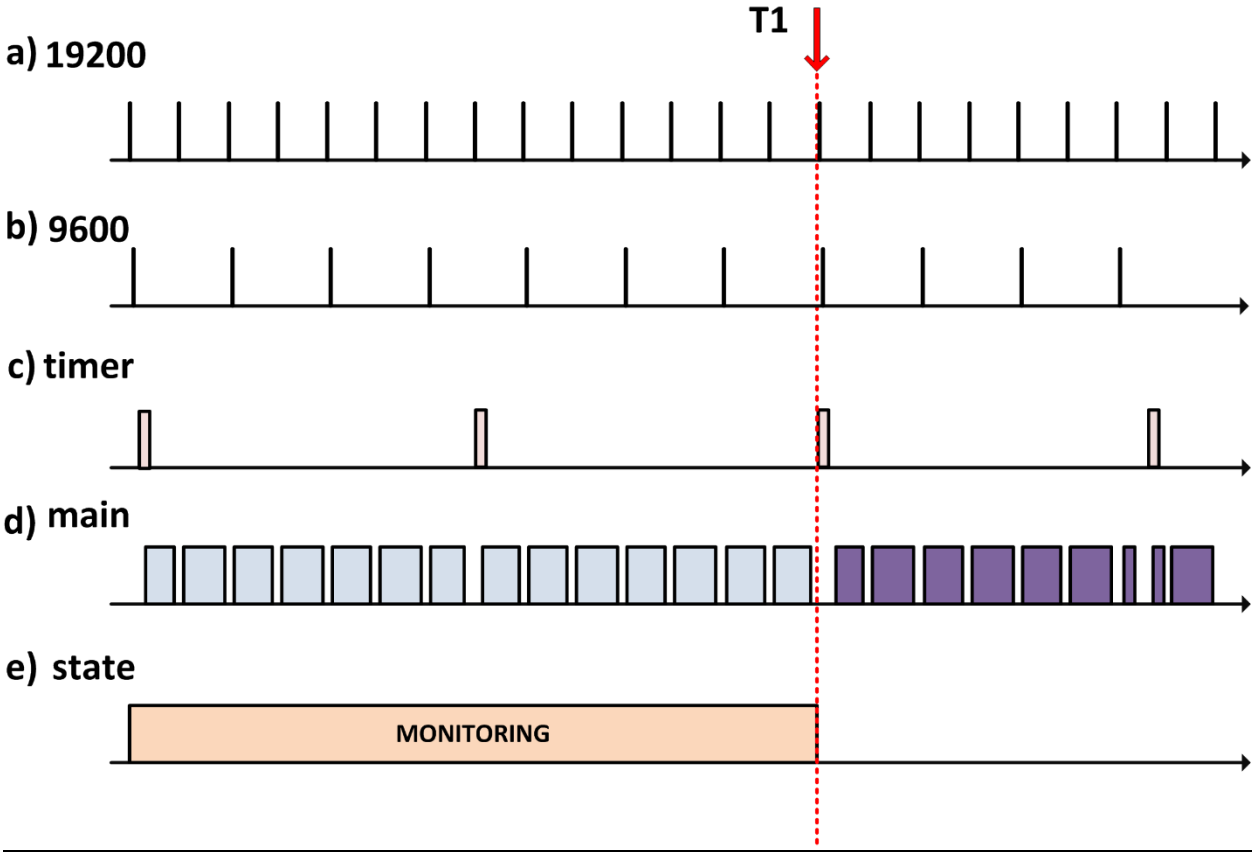


Fig.7.5 – Timeline diagram of the program flow

The timelines a), b) and c) are the interrupts presented earlier. The period at which they have been represented has been scaled in order to better illustrate their interaction. Timeline d) is the main program flow and it can be seen how it is interrupted by any of a), b) or c).

An example scenario is as follows: When T1 occurs the bytes received by radio module - interrupt a), are enough to be decoded and interpreted into a command. The command in this case is to stop monitoring. After the end of the interrupt routine the device has stopped its monitoring state (note d) and c) in the figure) but the main program loop still continues to run and interrupt are continued to be served and processed.

8. Tests, Results and Discussions

8.1 Introduction

This chapter is entirely dedicated to describing the tests that have been carried out and making some conclusions based on the results obtained.

Tests are split into 2 main categories: indoor/lab and outdoor tests. Throughout all the development phase small tests have been carried out (for example proper writing to the SD card, correct functionality of the software on the embedded device, data integrity on the I2C bus, etc.) These tests have all been part of the development and are not described in detail.

The overall system tests and proof-of-concept demonstration shall be treated in this chapter.

8.2 Field Tests

8.2.1 Radio module range test

The main purpose of the first field test was to see and measure the range of the radio modules. The test setup was composed of the two designed versions of the PCB, one of them was using the linear converter for the power supply and the other one was using the step-down converter.

Test scenario: One node of communication was placed in a fix, predefined place and not moved throughout the entire experiment. It was programmed to send data packages via the wireless radio interface (Radiocrafts RC-1280).

The second sensor node has been fitted with a GPS module. It was programmed to receive any incoming data from the wireless module, and write that data received along with a timestamp and GPS coordinates on the SD card. Basically, this test has checked the functionality of the SD card, radio module, GPS module, voltage regulators and several GPIO pins used to control some status LEDs.

The next picture shows the 2 devices tested and the way in which they have been set up:

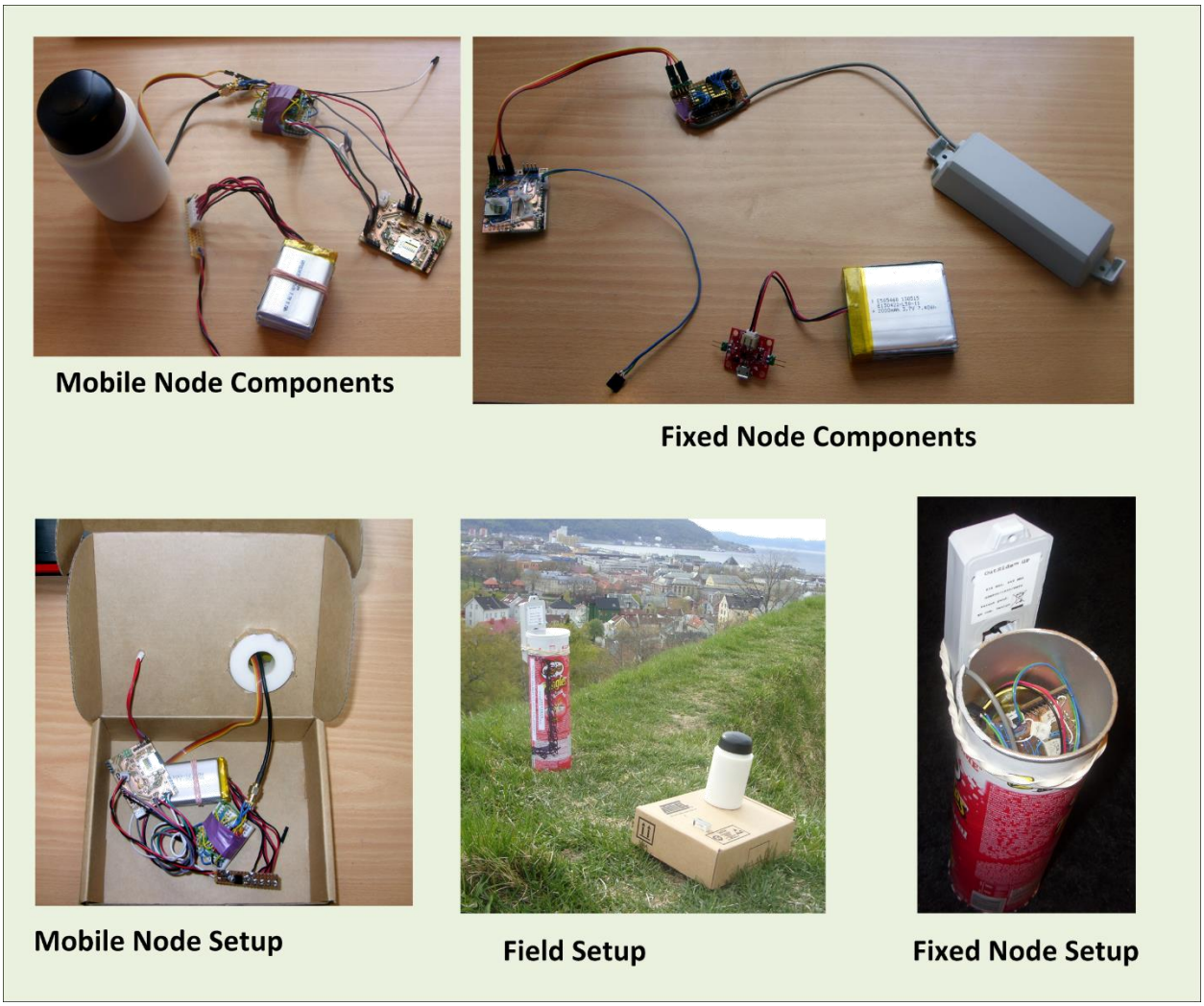


Fig.8.1 – Radio module range test setup

Test scenario:

After placing in a predefined fix position the sensor node programed to send data. The second, mobile sensor has been moved around the area in approximately 1km radius from the fixed sensor.

The next image shows the trace of the moving sensor. It is drawn using the Google maps API [19]. The red line trace represents the location at which the mobile sensor received data from

the fixed sensor. Distance measurement was done using the built-in tools provided by Google maps

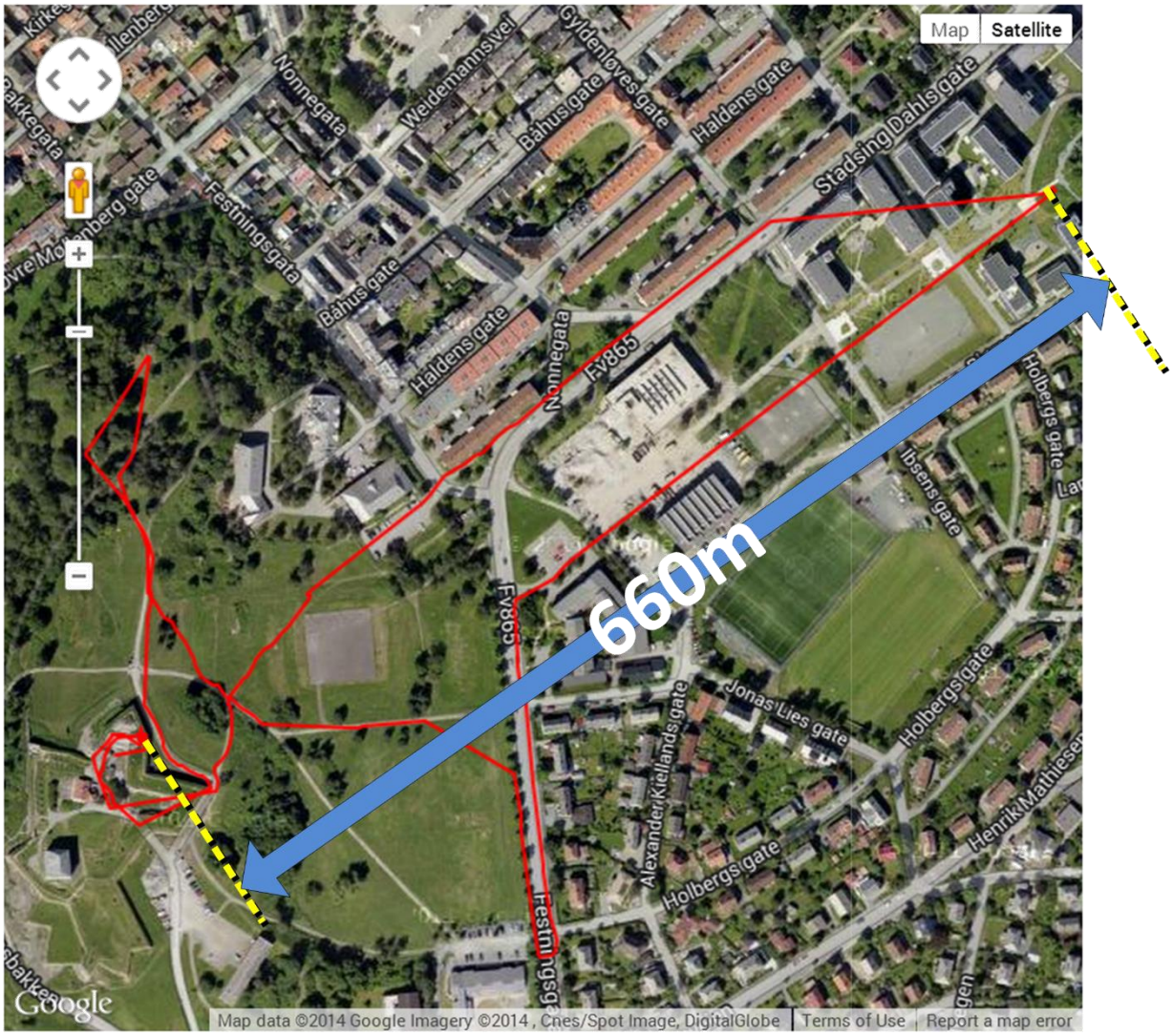


Fig.8.2 – Tested mobile sensor GPS trace

Test Results:

The results of the test were very promising: The maximum distance at which the 2 modules communicated was 660m. This leads to a potential area equivalent to a circle with the diameter of 1.3km. Although the test conditions were not ideal, this result led to believe that a valid communication between a UAV and a sensor node in the water at that distance can be successfully achieved. A better test scenario would have involved a different location, far away from the city and any radio interference that are present in a city.

8.2.2 Complete system tests on land

The next tests are perform in order to check the feasibility of the entire communication flow presented in chapter 5. A key difference from the previous test is that one more module has been added to these tests: a UAV. The UAV used is an X8-flying wing and its payload include a pandaboard and several mechanical/electrical additions (see appendix C) which was programed and set up to start DUNE software framework which was presented in a previous chapter.

The pandaboard is running Linux and a DUNE task has been written to read the incoming data from the serial port parse it and pack it into IMC messages. After that, the received messages were sent to DUNE via a 5 GHz link.

Test scenario and setup:

To be able to test the system flying a UAV was required. The test location was Breivika, Agdenes, Norway. The sensor node has been placed on land. The UAV was flying above it to collect its data and to forward it to a base station. The next image shows a snapshot of the test procedure, its main components and interactions.

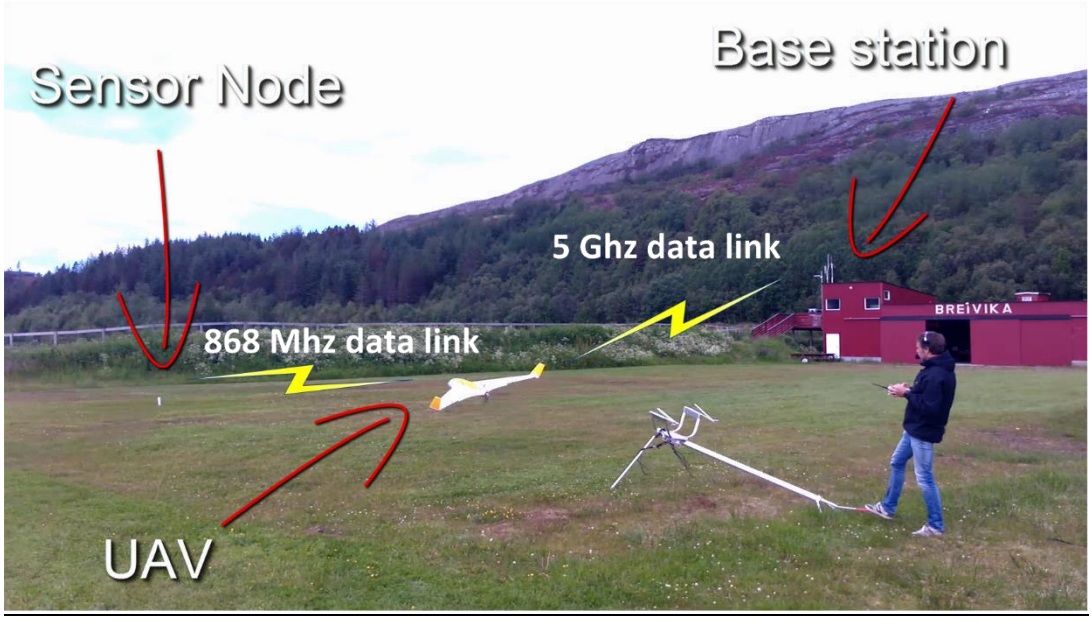


Fig.8.3 – Filed test setup on land and component interaction

In this test scenario the sensor node has been programed to sample data, send it to the UAV and log it to the SD card as well. No sleep modes have been used. The data sent by the sensor node is composed of temperature read and its GPS coordinates.

The pandaboard in the UAV payload is running a DUNE task that reads the data coming from the sensor node, logs it and at the same time, sends some data back to the sensor node and also forwards the received sensor data via UDP connection the 5 Hz link to the base station.

The base station is a computer running DUNE software which is also connected to a ROCKET M5 5GHz Hi Power 2x2 MIMO AirMax TDMA BaseStation

Test Results:

The flight took 14 minutes and throughout the entire time there was a valid data link between the sensor node, the UAV and the base station. The next image shows the flight path of the UAV. The sensor node has been placed in the center of the red circles represented on the map. The maximum distance at which the UAV was from the sensor node was 100m on the horizontal axis and 100m in altitude.

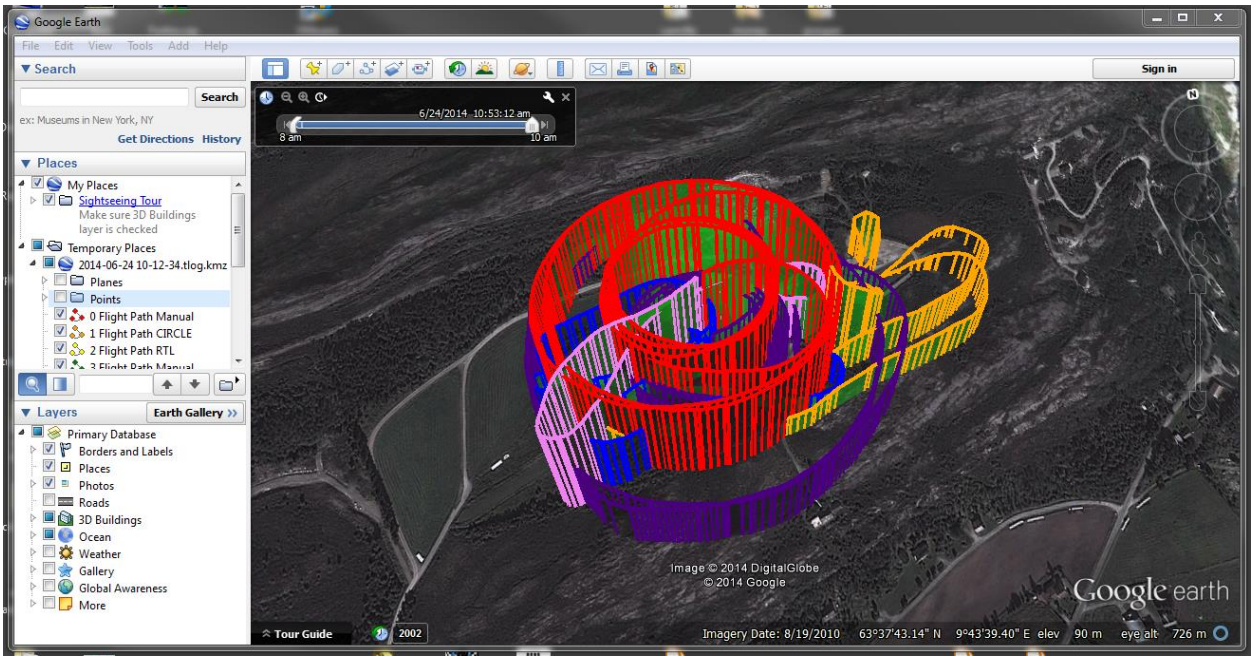


Fig.8.4 – Flight trajectory of the UAV in the land test (generated with Google earth)

The number of messages (GPS position and Temperature) that have been sent from the sensor node to the UAV was about 438. All these messages had a checksum field attached and there was only one checksum fail during the entire flight. This means that the packages lost due to invalid transmission from the sensor node to the UAV was minimum.

A different situation was in the messages sent from the UAV back to the sensor node. The data packets that are sent from the UAV to the sensor node are supposed to be commands and configuration files. In this test scenario the data (commands) received was not interpreted, it was just logged on the SD Card.

The results were quite surprising. In the next image we have a snipped of the data that was supposed to be received (left) and on the right we have the data that was actually received. Since no CRC field was added to these data packages even the corrupt ones have been logged.

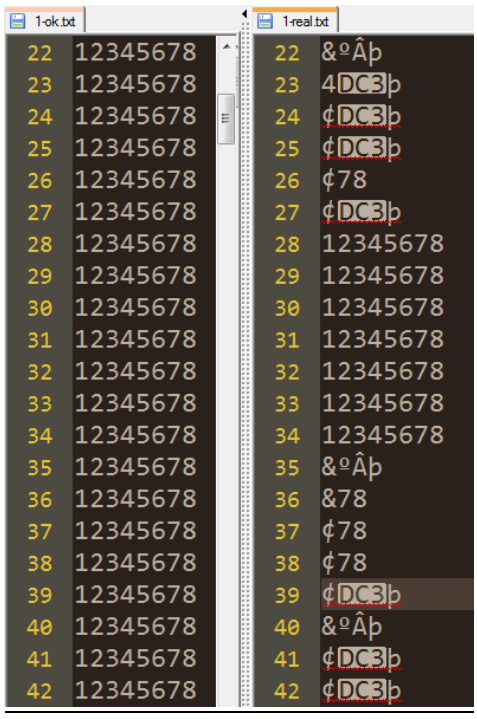


Fig.8.5 – Flight trajectory of the UAV in the first test

Computing the validity of the entire log file results show that only 13.38% of the received "12345678" packets were valid, the rest was corrupt.

The link between the UAV and the base station is more stable and robust. The messages sent from the UAV (pandaboard) to the base station (laptop computer) were IMC messages packaged in UDP network frames. The next image is a screen capture taken from the field

tests which displays the sensor node on the map (identified as “ais-1” vehicle) and its incoming data (temperature) based on the received network messages from the UAV.

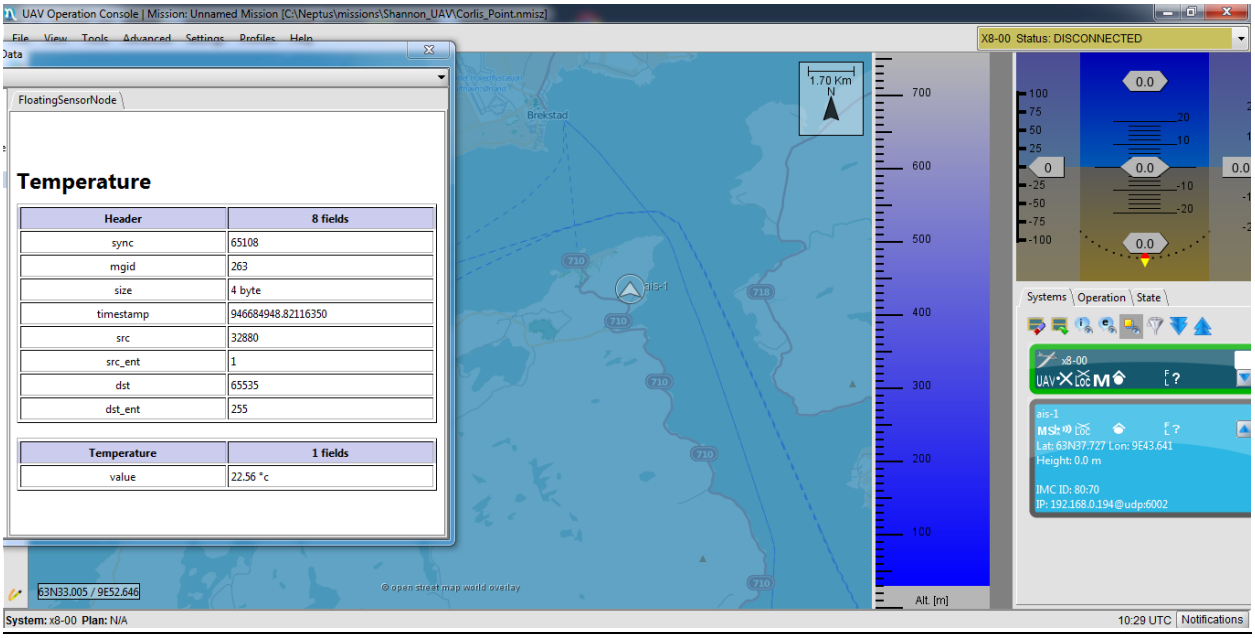


Fig.8.6 – Screen capture of Neptus showing the sensor node on the ground

The results of this test is that a valid link between the sensor node, the UAV and NEPTUS command interface has been successfully made. Some package losses are to be expected and have been present throughout the test time but the basic functionality has been proven.

8.2.3. Complete system tests on water

Test scenario:

The second test carried out was with the sensor node floating in the water. It has been placed next to the shore in relatively shallow waters and anchored to the ground using rope

Same as before, the UAV has flown above the sensor node and collected its data.

The next figure *(Fig.8.7)*. shows the test environment and the interaction between the UAV, sensor node in water and the base station.

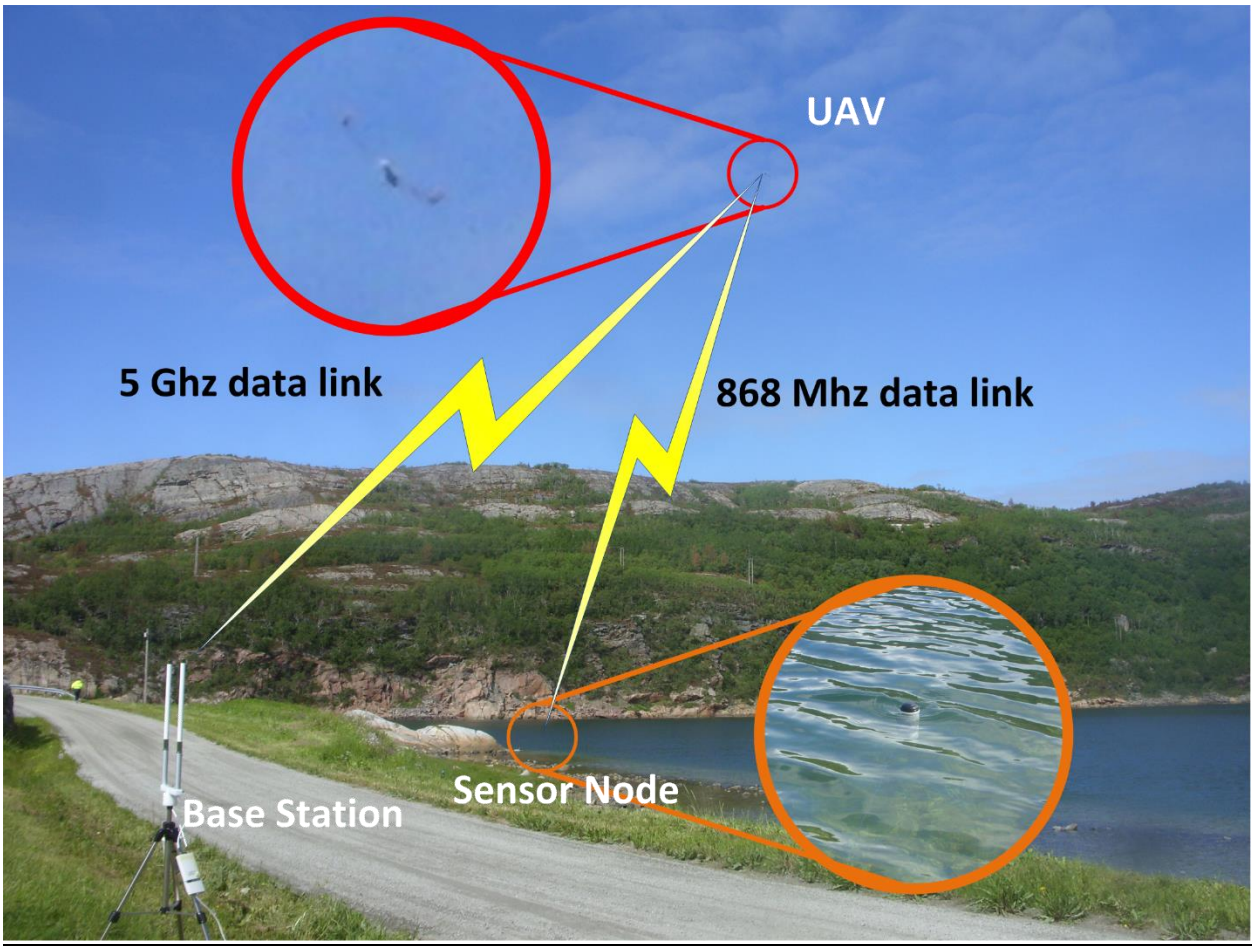


Fig.8.7 – Filed test setup on water and component interaction

Test Results:

The flight test took 12 minutes during which a number of 376 data packages representing sensor readings were sent from the sensor node to the UAV. None of these had a CRC fail.

Same as before, the string "12345678" was sent from the UAV to the sensor node and this time the data integrity was lower than in the first case. Only 6.4% of the received data packets were correct.

The flight path is presented in the next figure (Fig.8.8):

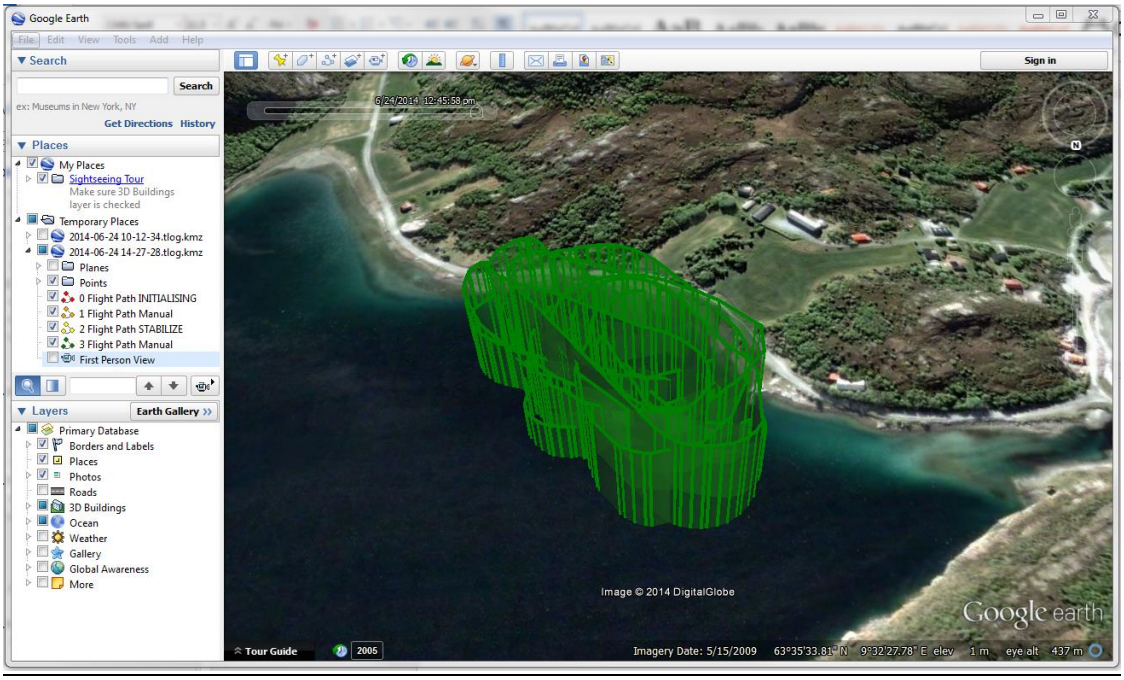


Fig.8.8 – Flight trajectory of the UAV in the water test (generated with Google earth)

The maximum distance at which the UAV was situated from the sensor node was 264m (measured using Google maps) and the height was 100m. At these distances at there was still a valid data link.

The next screen capture *(Fig8.9)* shows NETPUS command interface showing the sensor node on the map and its incoming temperature readings:

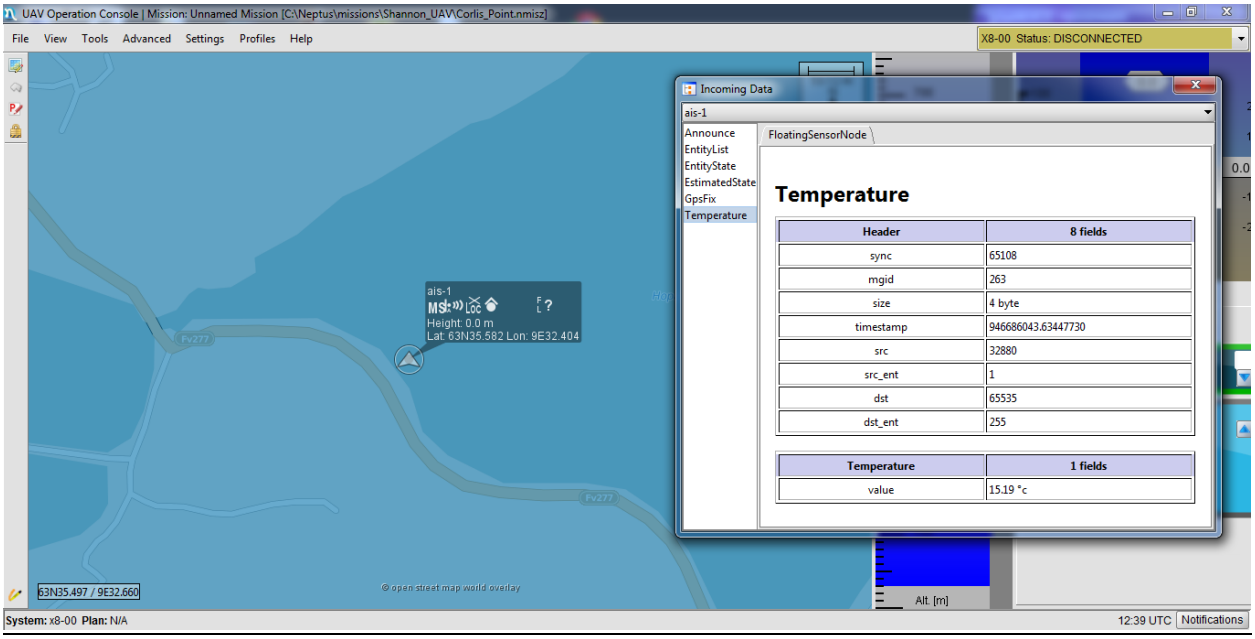


Fig.8.9 – Screen capture of Neptus showing the sensor node in the water

During the flight time the sensor has been moved from ground and placed in water. The next graph *(Fig.8.10)* shows the plotted altitude of the sensor node during the flight time.

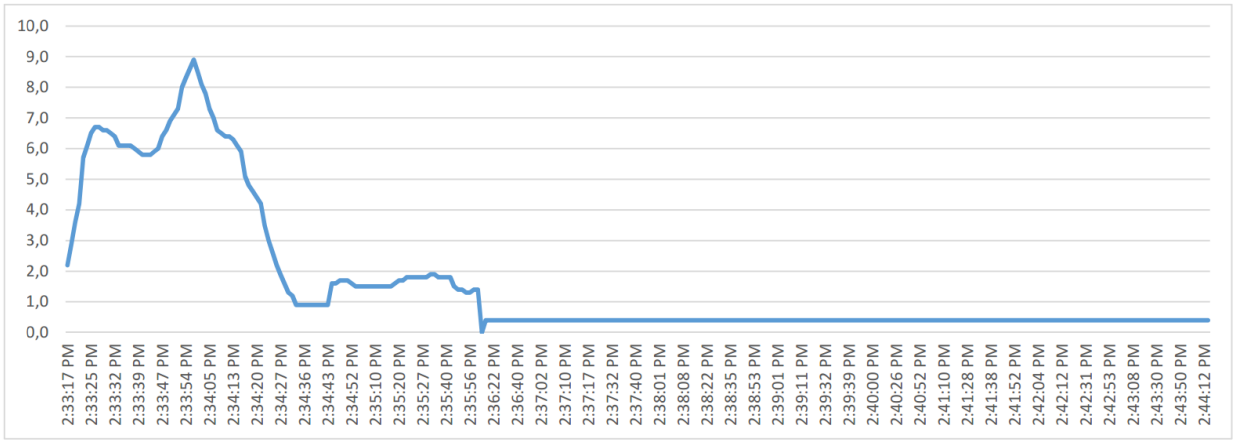


Fig.8.10 – Measured GPS altitude of the sensor node during the test

8.4 Conclusions on tests

All the numbers and statistics presented as results in this chapter are measured data during field/lab tests. Some are taken from the log files on the pandaboard and some are from the SD card inside the sensor node.

These results of the tests presented earlier were encouraging:

- The tested range of the radio devices used in the sensor node proved to be sufficient allowing for a robust sensor node to UAV communication and not so robust from the UAV back to the sensor node
- The proposed physical case solution proved to float in the water and be directionally stable, while still achieving its functionality.
- Meaningful data was sent across the system
- The software written on the embedded device and on the pandaboard have achieved their role in interfacing the embedded device
- NEPTUS command interface can be used with ease for displaying senor readings

Videos taken from the project demonstration can be seen here:

https://www.youtube.com/watch?v=7_dGRSydzB4

https://www.youtube.com/watch?v=d3XSWpmu6tk

7. Future Work

Since the system is designed to be modular, both from the software and hardware perspective there are many opportunities for adding extra functionalities to the current design.

7.1 Immediate future goals

7.1.1 A better choice for the hardware modules

As presented in the previous chapters the current hardware has been tested and proved to work but during the development and testing stage several issues have arisen with the chosen hardware modules.

For instance the Radiocrafts module, RC-1280 is not perfectly 3.3V compatible as the datasheet states. This means that a level shifter had to be used to interface it. Because of this, the complexity of the hardware has slightly increased. This is not a very bad radio module and the fact that it has been proved to work does not necessarily impose that it should be changed. At least, it should be avoided in other future related project. A possible choice for a new radio module can be the first or second option presented in Table 1 in Chapter 4.3.2.

Also, the GPS module has its limitations. Because the system is designed to be low power. All components should have and support shutdown functionalities. Most GPS module on the market have a Shutdown pin that can be used to completely power off the device. The current chosen module does not have this, limiting to some degree the extent at which optimum power efficiency can be achieved. Also the size of the device is not particularly optimum either. To fit the module in the sensor node some hardware modifications had to be done to it in order to have the external antenna connected to it.

7.1.2 Main board and software improvements

There is some discussion also related to the choice of the hardware platform in use. Even though the microcontroller chosen for the sensor node is more than capable of doing what is supposed to do improvements can be made.

Taking into account the designed software architecture (Chapter 5.3) and the proposed implementation of it, several choices exist at this point. The code that runs on the microcontroller is written in ANSI C language. This is not an object oriented programming language and implementing designs patterns in such languages is not that common, but still practiced. The compiler used (Atmel Studio 6.1) supports C++ codebase therefore the code can be rewritten in C++ or the project setup can be made so that it supports both languages. This is also possible with some work and time invested.

So the basic options are:

a) Implement the proposed hardware architecture as it is in the current project setup
b) Rewrite and merge C and C++ code so that the project supports object oriented characteristics.
c) Chose a different hardware platform altogether.

The third option might be taken into account in situations where the desired goal is the application, not the hardware itself. There are a number of hardware platforms that could have been used for this current project, small, low-power and most importantly providing a multitude of predefined libraries. One of such platform is the mbed LPC1768 [20]. It supports all the needed peripherals as the currently used platform but the time it takes to build an application is much lower. It also offers a built in OS making it more adaptable

The choices have been presented, none of these are bad and the time it would take to adopt these changes is not that long.

7.2 Long-term future goals

7.2.1 Acoustic fish telemetry module

One of the research topics currently under work at the Norwegian University of Science and Technology involves using an acoustic fish telemetry device for monitoring fish behavior and ecology. Currently these devices are placed in the water and long wires are used to connect them to a base station. Having such devices connected to a sensor node via the RS-485 interface would be a great improvement. The need for wires is gone and the data can become easily accessible by using UAVs to sample it. Having this module integrated with the sensor node also means that the acoustic fish telemetry module can be placed in more remote areas and with less impact to the ecosystem

7.2.2 Energy harvesting

From the beginning of the project it has been mentioned that the system is to be designed as efficiently as possible from a power consumption point of view. But even with all the low power design principles and sleep modes of the devices and hardware there the reality that at a certain point the battery of the device will be drained. In this case adding energy harvesting capability would overcome this.

Harvesting wave energy is not something new, it has been done before many times but almost all the present solutions for wave energy harvesting are done at a very large scale and with attaching something heavy on the bottom of the sea [21][22][23]. These constrains limit us very much to the number of options we have for energy harvesting.

Fig.7.1 – ECO 200 [24]

However there are certain ways in which wave energy harvesting can be done at this small scale. First of all there are solutions that use the piezoelectric effect. This basically converts mechanical strain into electric current and voltage. Since the sensor is floating on the ocean, at every moment in time, it will move. The wise idea would be to use this movement and generate electricity. One solution for achieving this is to use a miniaturized piezoelectric energy harvester [24] (*Fig. 7.1*)

Another option for wave energy harvesting would be to use a linear faraday generator [25][26]. This is a fairly new concept, it is not currently very popular and patents are still being issued for it. The basic principle of functionality is that what would normally be a spinning coil and some permanent magnets to generate electricity would be placed not in a circle but linearly. The rotary movement of the axis would just be replaced by a linear movement up and down in line with the waves. A basic design prototype is presented in the following figure (*Fig.7.2*)

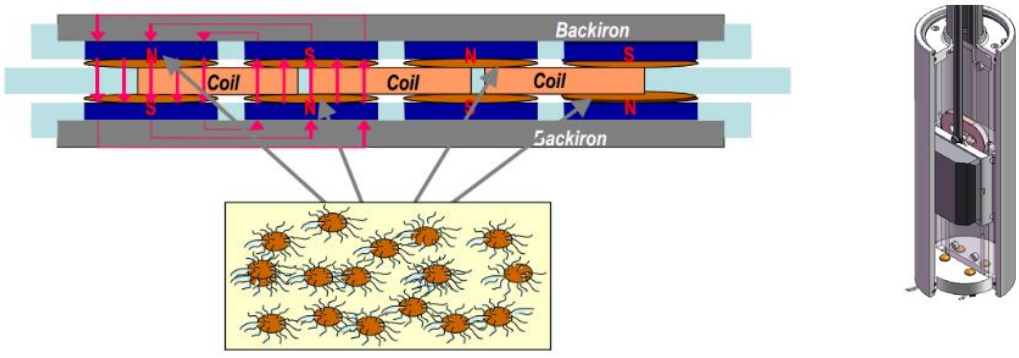


Fig. 7.2 –Design of Linear Faraday Generator with Ferro Fluid Nano Bearings (left) CIIIS Linear Generator prototype (right) [25][26]

5.2.1 Data compression algorithms

Transmitting large amounts of data over a wireless medium in a system that is designed to be low-power is not always a good idea. A simple way of reducing power consumption in this situation is to do some local processing (compressing) on the data. For this purpose a popular conversion library written in C can be used: zlib[27].

Let's say that the range of the wireless communication module fitted on the UAV and on the sensor node has the range of 1.3km (d1). Ideally this means that the UAV can be in range with the sensor node on a surface equivalent to a circle with the diameter of 2.6km (d2). But the range of the module is more like a sphere and loses range with height. If the UAV flies very high it might not reach it.

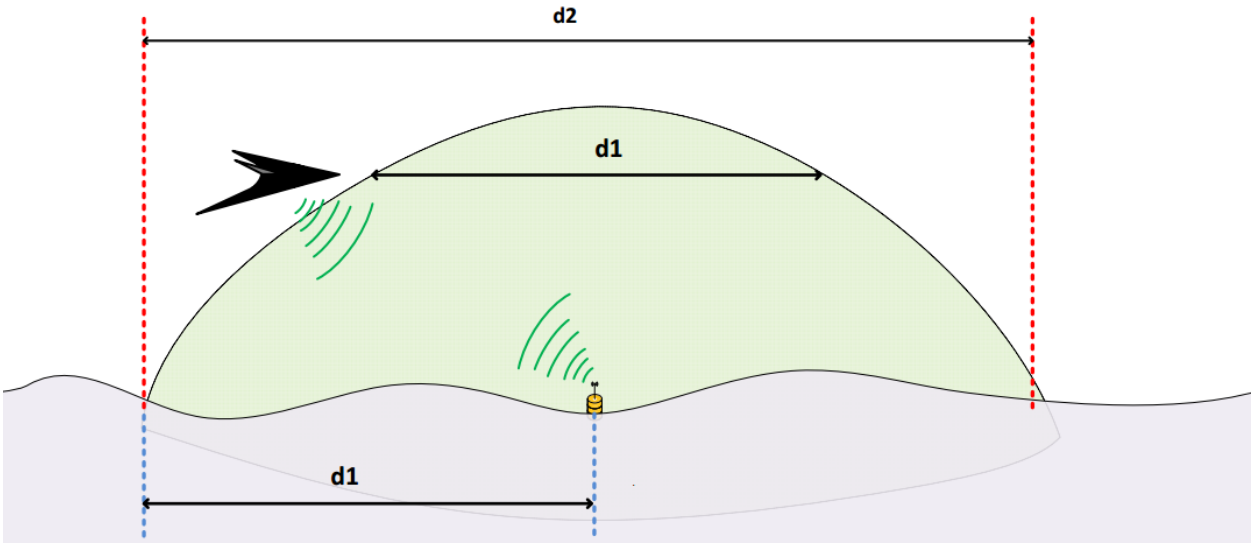


Fig. 7.3 – Example scenario regarding the range of the wireless communication modules

Taking a reasonable value into consideration we can say that the UAV is in range with the node for about 1.3km (d1 in the figure). If the minimum speed at which a fixed-wing UAV can fly is 13m/s that means that the two would be in range of each other for about 100 seconds. If the maximum speed of the wireless module is about 4.8kb/s, in the best case we could have only 480kb of data transmitted. This would certainly not be enough to contain data sampled over a few months, for example.

Here is where zlib comes in. Typical zlib compression ratios are on the order of 2:1 to 5:1. This means that in theory with some local processing both on the sensor node (for compressing) and on the UAV (for decompressing) a lot more data (up to five times) can be transmitted wirelessly if we compress it. In this case UAV would have to fly over the sensor node fewer times and the node can be in sleep mode for longer periods of time, resulting in an overall reduction of power consumption.

8. Conclusions

To sum up everything that has been presented, this project is addressing an ever increasing problem in the world in a way that has not been done previously: scalable and easy to maintain large scale ocean monitoring. It incorporates research done in various fields, electronics and communications, computer science, high level design principles and even physics and chemistry.

The tests carried out and the prototype built lead us to conclude that the approach and the idea for the project so far is good and that if continuing with the same approach, the project can bring value to the scientific community.

The software used is open source, making it free, easy to expand, upgrade and debug, having a large community of developers that constantly support it.

Finally, because of the current existing working prototype and the proof-of-concept tests carried out it is advisable to continue on with this project and develop it further.

References

- [1] ARGO motivation and concept: http://www.argo.ucsd.edu/index.html Accessed:14.12.2013
- [2] How oil spills affect marine life. http://oceanservice.noaa.gov/facts/oilimpacts.html Accessed:14.12.2013
- [3] Underwater Temperature Recorder http://star-oddi.com/products/34/subsea-temperature-recorder/default.aspx Accessed:02.05.2014
- [4] Submersible Acoustic Transmitter - http://star-oddi.com/products/44/low-frequency-acoustic-sounder-transmitter/default.aspx Accessed:15.05.2014
- [5] Real-Time Deep-Ocean Tsunami Measuring, Monitoring, and Reporting System: The NOAA DART II Description and Disclosure - Christian Meinig, Scott E. Stalin, Alex I. Nakamura NOAA, Pacific Marine Environmental Laboratory (PMEL), Hugh B. Milburn Oceanographic Engineer
  [6] MATAKI - An open, low-cost, wirelessly-enabled tracking platform http://mataki.org/ Accessed:14.12.2013
- [7] Puck antenna manufacturer: http://www.globalsources.com Accessed:14.12.2013
- [8] Measuring battery voltage http://www.microbuilder.eu/Tutorials/Fundamentals/MeasuringBatteryVoltage.aspx Accessed:14.12.2013
- [9] Design techniques for extending li-ion battery life. http://www.digikey.com/us/en/techzone/power/resources/articles/design-techniques-extending-li-ion-battery-life.html Accessed:14.12.2013
- [10] Adam Petersen, Patterns in C https://leanpub.com/patternsinc Accessed:14.12.2013
- [11] Hungarian notation - http://msdn.microsoft.com/en-us/library/aa260976(v=vs.60).aspx Accessed:10.02.2014
- [12] Object-oriented C is simple http://www.embedded.com/electronics-blogs/object-oriented-c/4397794/Object-oriented-C-is-simple- Accessed:14.12.2013
- [13] Eric Freeman, Elisabeth Robson, Bert Bates, Kathy Sierra, Head First Design Patterns, O'Reilly Media, October 2004, pages 44-52

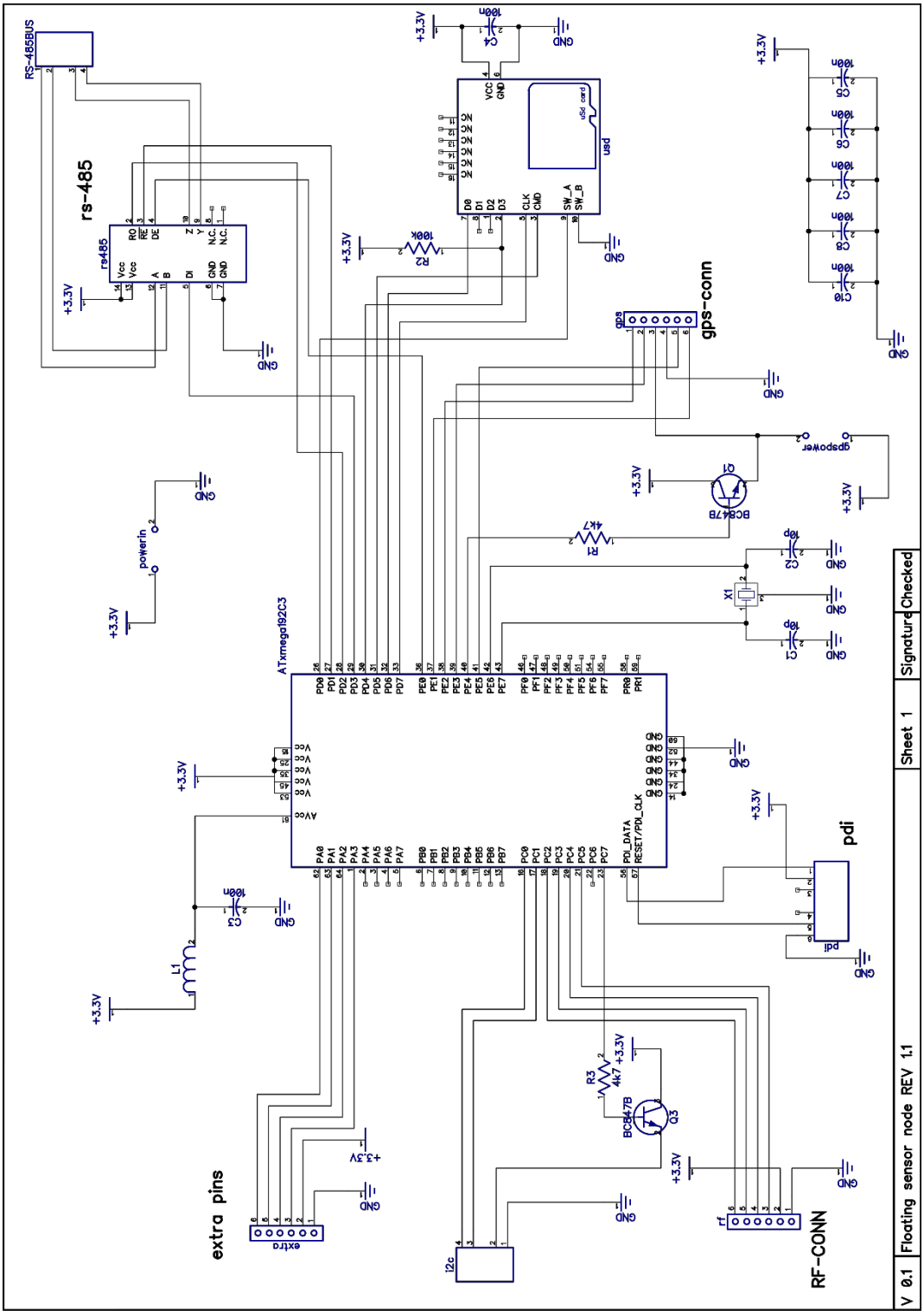
- [14] DUNE: Uniform Navigational Environment http://lsts.pt/software/dune Accessed:14.12.2013
- [15] Experiments with Deliberative Planning on Autonomous Underwater Vehicles - Jose Pinto, Joao Sousa, Frederic Py and Kanna Rajan
- [16] Implementation of a Control Architecture for Networked Vehicle Systems - Jose Pinto, Pedro Calado, Jose Braga, Paulo Dias Ricardo Martins, Eduardo Marques, J.B. Sousa - Department of Electrical and Computer Engineering, University of Porto, Portugal – 4200-465.
- [17] IMC Inter-Module Communications Protocol for networked vehicles and sensors https://www.lsts.pt/imc/doc/master/ Accessed:14.12.2013
- [18] NMEA Protocol - http://www.gpsinformation.org/dale/nmea.htm Accessed:23.02.2014
- [19] Google Maps API - https://developers.google.com/maps/ Accessed:10.01.2014
- [20] MBED http://mbed.org/platforms/mbed-LPC1768/ Accessed:18.04.2014
- [21] Commercially-viable energy from the movement of lakes http://www.renewableenergymagazine.com/article/commerciallyviable-energy-from-the-movement-of-lakes Accessed:20.12.2013
- [22] Harvesting the Ocean: A New Approach to Wave Energy Conversion http://www.worldchanging.com/archives/009241.html Accessed:20.12.2013
- [23] Wave Power Prototype Produces Electricity http://www.digikey.com/us/en/techzone/energy-harvesting/resources/articles/Wave-power-prototype-produces-electricity.html Accessed:20.12.2013
- [24] Miniaturized Energy converter for motion energy harvesting http://www.enocean.com/en/enocean_modules/eco-200/ Accessed:20.12.2013
- [25] SG2™ -Developing Technology to Generate Unlimited Green Energy from Ocean Waves - S C Chao El Segundo, California November, 2007
- [26] CIIIS presentation on linear faraday generator technology
- [27] ZLIB - A Massively Spiffy Yet Delicately Unobtrusive Compression Library http://www.zlib.net/ Accessed:18.12.2013

- [28] “Direct Gravity Estimation and Compensation in Strapdown INS Applications “3rd International Conference on Sensing Technology, Ehad Akeila, Zoran Salcic and Akshya Swain Nov. 30 – Dec. 3, 2008, Tainan, Taiwan
- [29] Tilt Sensing Using a Three-Axis Accelerometer - Freescale Semiconductor, Document Number: AN3461, Rev. 6, 03/2013

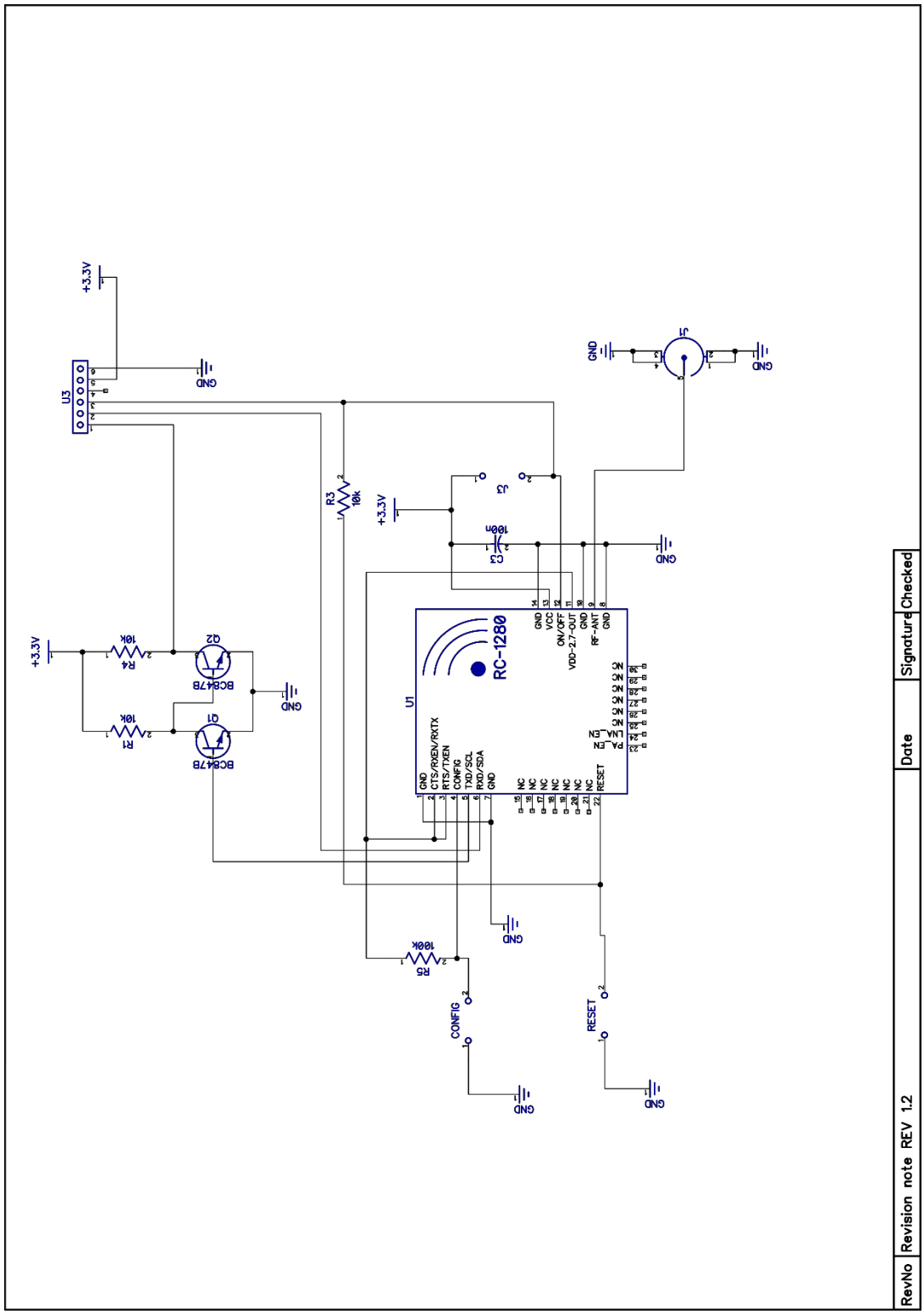
Appendices

Appendix A

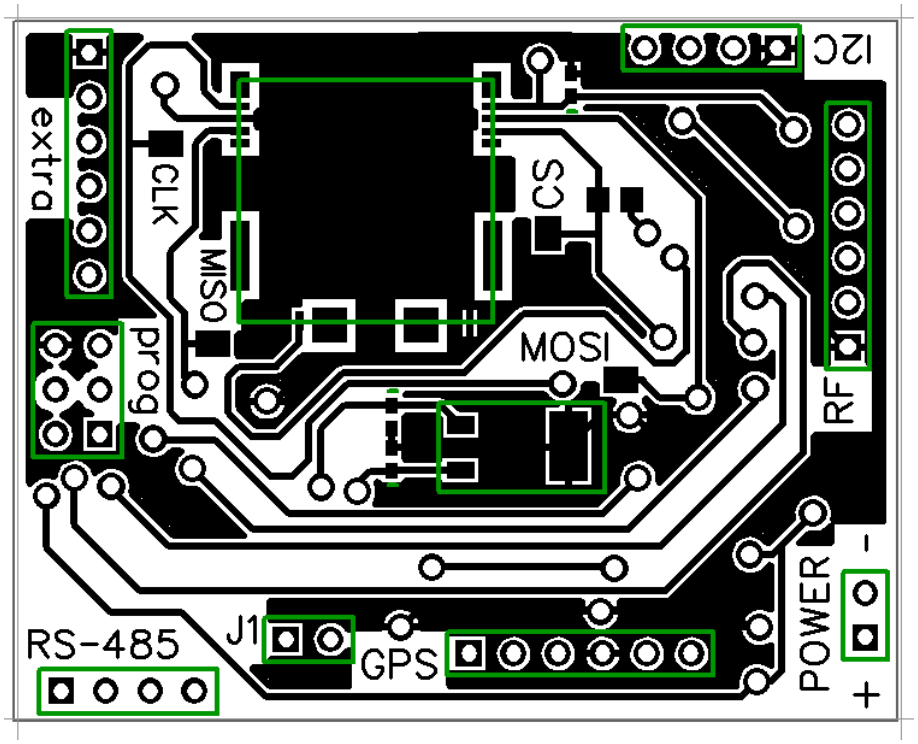
Main Board schematic

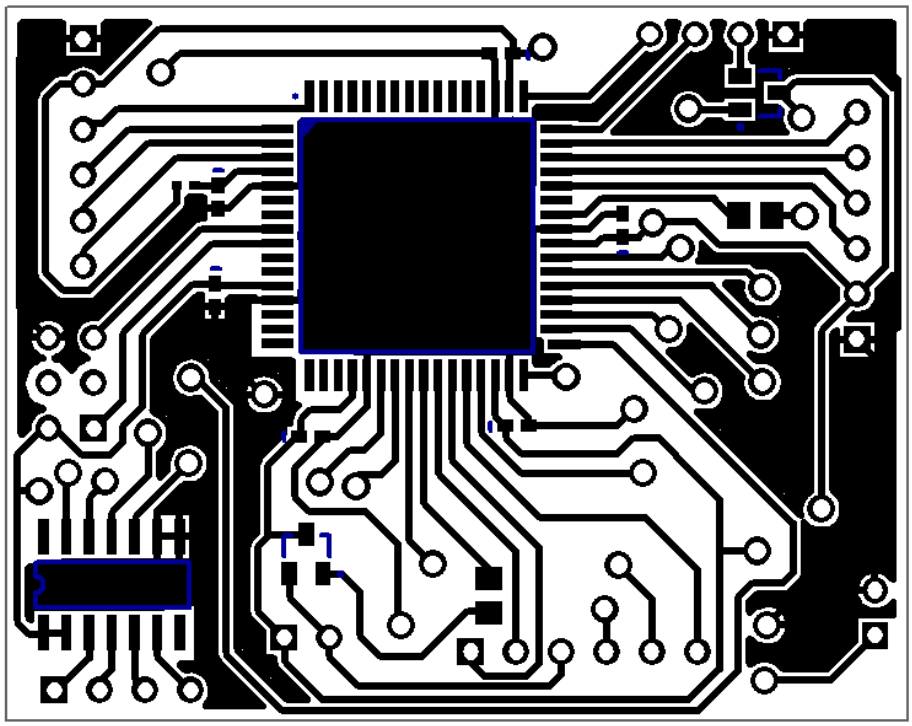


Radio module schematic

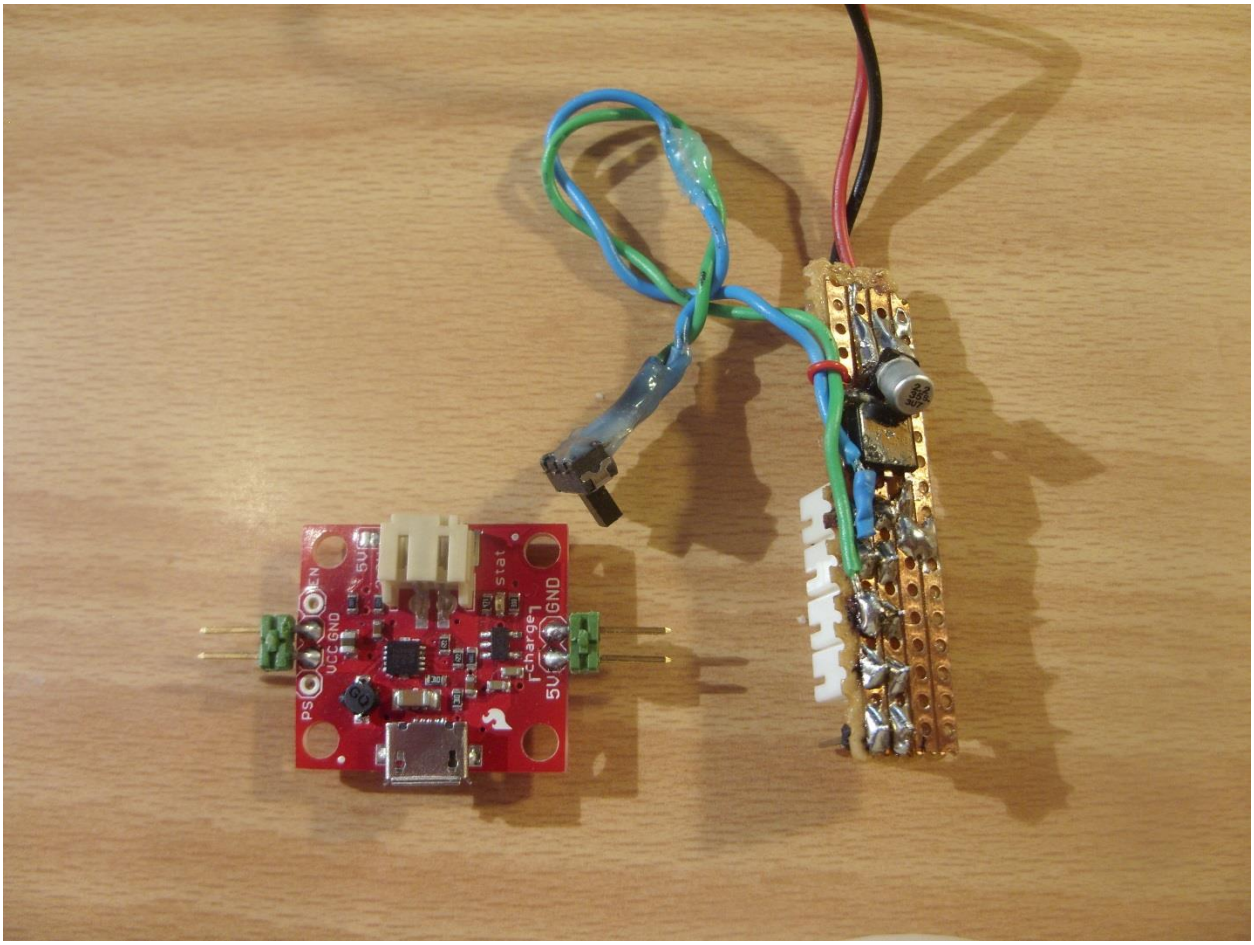


Printed circuit board layout top and bottom



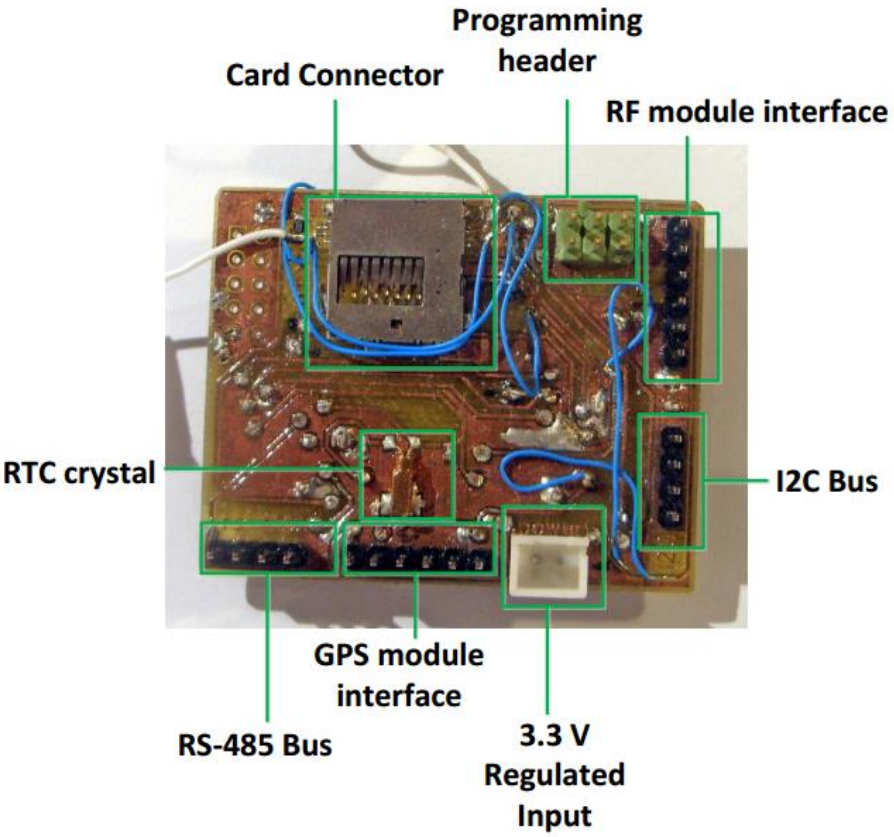


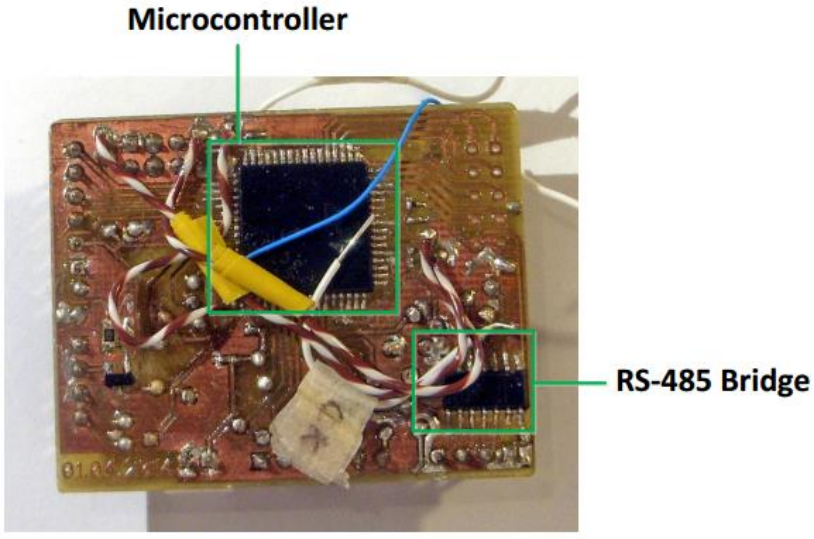
Voltage Regulators (DC-DC and linear)



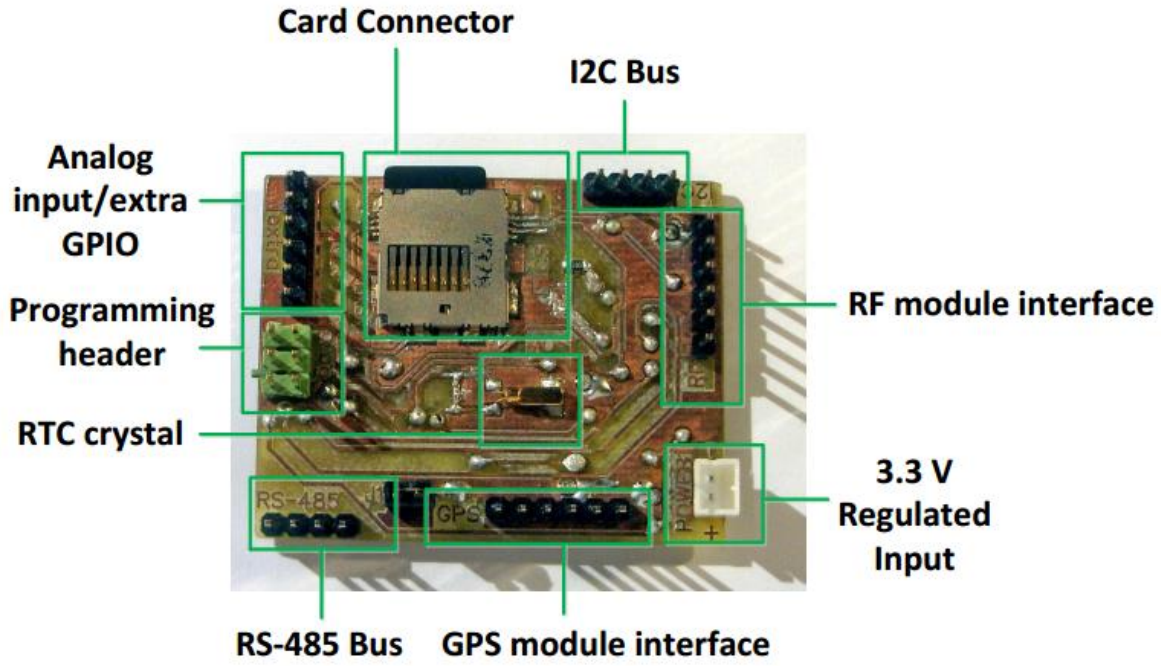
Appendix B

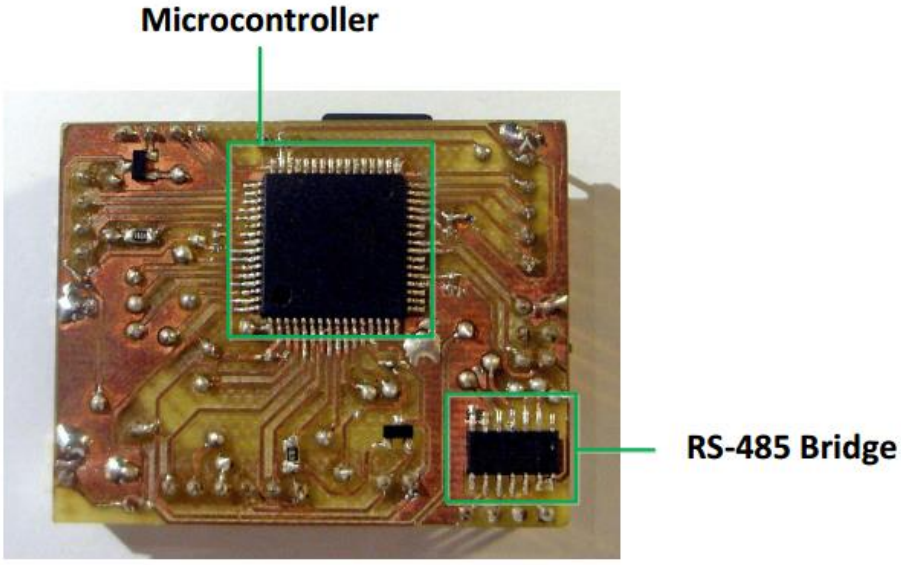
Printed Circuit Board - version 1





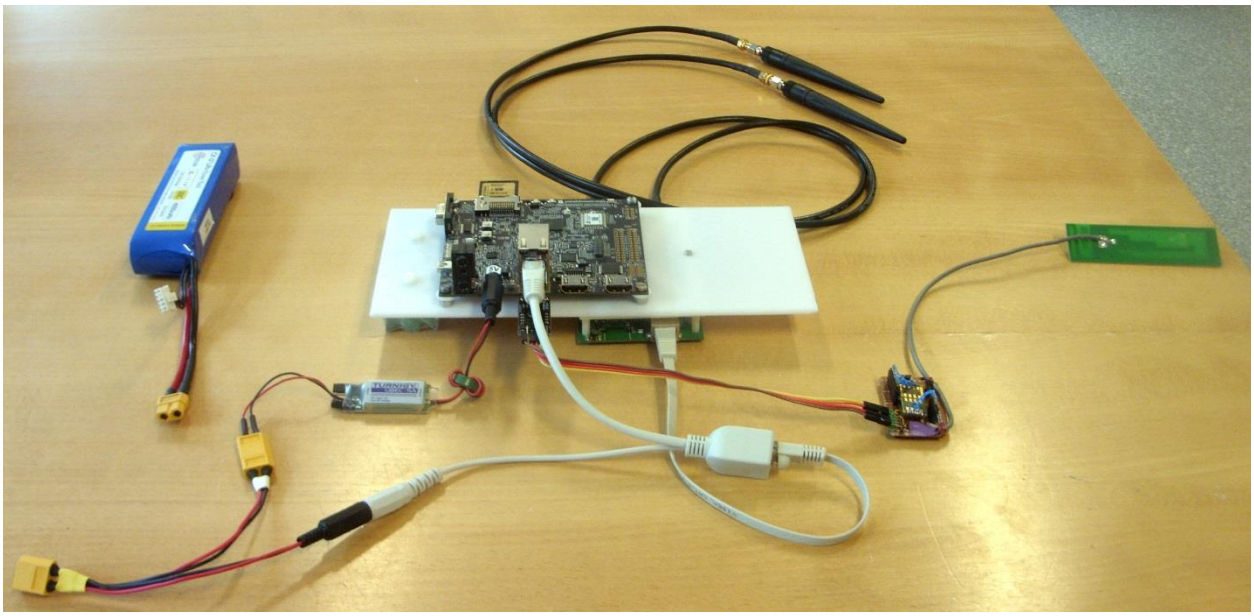
Printed Circuit Board - version 2





Appendix C

UAV Payload



During the flight tests the UAV was equipped with the following components: 11.3V battery pack, 5V voltage regulator, Pandaboard, Power over Ethernet adapter, Radiocrafts module, usb-uart converter, 868Mhz Antenna, ROCKET M5 5GHz Hi Power 2x2 MIMO AirMax TDMA BaseStation, 5Ghz antennas