Doctoral Dissertation
Doctoral Program in Electrical, Electronics and Communications Engineering
(32th Cycle)

Spectrophotometric monitoring system for continuous heavy metal detection in water environment

Felice Catania

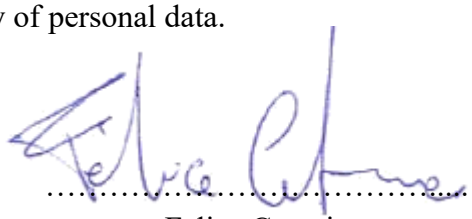
* * * * * *

Supervisor
Prof. S. Ferrero

Doctoral Examination Committee:
Prof. M. Bibuli, CNR-INM U.O.S. di Genova
Prof. G. Firpo, Università degli Studi di Genova

Politecnico di Torino
2019

I hereby declare that, the contents and organisation of this dissertation constitute my own original work and does not compromise in any way the rights of third parties, including those relating to the security of personal data.



Felice Catania
Turin, November 28, 2019

Summary

Water environmental monitoring is an important key to control and take care of human life and environment health. Water quality influences fluvial and marine wildlife, but also the neighbouring environment as well. It is necessary that water monitoring instruments become within the reach of local authorities and control units, in order to increase the amount of data available on water pollution and facilitate sharing. The pre analytical phase, sample collection, transport and preservation, has a not inconsiderable impact on the total uncertainty of the result of the analysis. It should also be considered that, the set of procedures and operations that occur between the time the sample is taken, and the performance of an analysis is one of the most delicate phases of the entire analytical procedure. The analytical results, in fact, must make it possible to establish the characteristics of the analyzed matrix in the conditions in which it occurs at the moment in which the sample is taken.

The answer will be an instrument, designed with lab on chip concept, with a modular structure. The device innovation relies on total automation of the analysis, from the sampling to production of results, in particular automatic filtration, dilution, concentration and preparation of the sample previous the measurement. Precision, reliability, fastness, robustness. The innovation regards also the reduced volumes of sample and reagents, low cost technology, high versatility, in terms of cost and technology. The system, which is the object of my project, will be able to completely automate the sample preparation phase and, using the lab on chip technology, greatly minimize the time and cost of sample preparation.

A microfluidic system, consist of glass syringes, micro pumps, and degassing system, allows to acquire the necessary quantity of water sample, filtering the sample through cascade of filters, up to nanoporosity possibly dilute or concentrate the sample, mix with any reagents.

Moreover, the implementation of this miniaturized laboratory on board of an Autonomous Underwater Vehicle, for marine water monitoring in the vicinity of oil and gas platform will be discussed.

Acknowledgment

My deepest gratitude goes to Prof. Sergio Ferrero, who has monitored me in these years of research, who has always supported me and has shown a helpfulness and kindness beyond his simple task as supervisor. I would like to thank the entire working group of the Department of Applied Sciences and Technology, with a focus on Prof. Luciano Scaltrito and Prof. Fabrizio Pirri.

Special thanks go to Dr. Monica Periolatto, who took care of the optimization of chemical methods, also coordinating the work of Dr. Augustin Miranda Fabbri and Dr. Debora Cocco, who contributed to method optimization and tests. Dr. Matteo Manichino's activity was very important for the realization of firmware and software, so I want to thank him.

I would also thank Prof. Andrea Lamberti and Prof. Matteo Cocuzza for their contribution to my research activities, and Dr. Valentina Bertana for sharing this experience with me and for her efforts in my research activities.

All this work would not have been possible without Microla Optoelectronics SRL, so my sincere thanks go to CEO Giorgio Damosso, who gave me the opportunity to carry out my research activity, showing willingness and friendship. The most important contribution to the realization of this project was provided to me by Ing. Andrea Piscitelli, who, in addition to his skills and his work for the design and realization of the mechanical and fluid dynamic part, also granted me four years of friendship. Invaluable work has also been done by Ing. Massimiliano Messere, who worked on design and implementation of electronics and firmware. I would like to tank Andrea Sulla and Andrea Quinci, who also contributed to this work. I would also like to thank Alessio, Andrea, Elena, Francesco, Gianluca and Monica.

Special thanks go to Ing Paolo Sirianni, mainly for nine years of great friendship, also for four years of team working.

Finally, I want to thank my family, who have been by my side over the years, supporting my choices and providing more valuable support than any other.

The research was sponsored by the Ministry of Economic Development, and in particular by the “Direzione generale per la sicurezza anche ambientale delle attività minerarie ed energetiche – Ufficio nazionale minerario per gli idrocarburi e le georisorse” DGS-UNMIG, and involved the work of CEMAS ELETTRA SRL for the realization of the mechanical parts, and the work of Chemifin Srl for chemical methods.

A mia madre e alla
sua inesauribile forza
di volontà

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Importance of live continuous monitoring for water environment

Man-made settlements, pollution and climate change have long been generating problems that they can directly influence the quality of supply sources. Continuously increasing contamination levels are detected in water bodies due to fertilizers, heavy metals, pesticides, etc. (Figure 1)

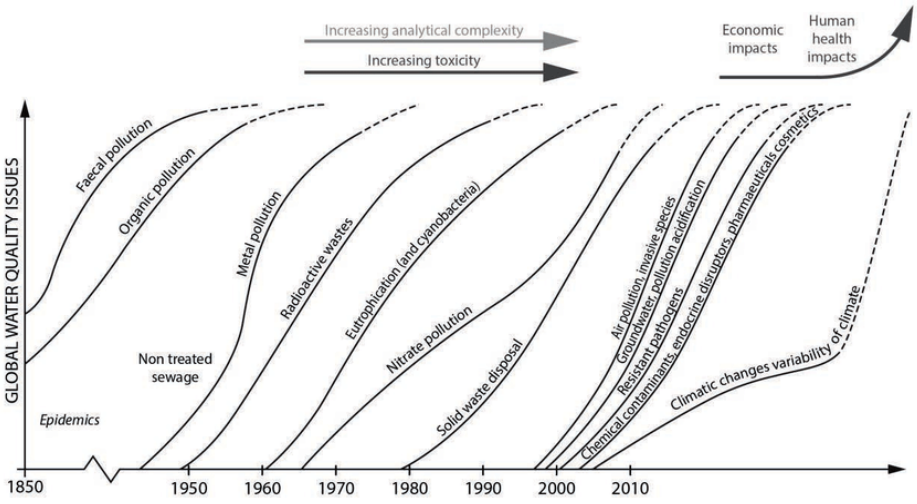


Figure 1. Trends in water quality in the evolution of water quality problems in industrialized countries. Reprinted from Tundisi et al. (2015). [1]

Several studies carried out over the last few years show that water resources will be scarce by more than 5 billion people in the next decade. Furthermore, the pollution of surface water makes the availability of the waters even more critical, mainly in the arid areas [2]. The quality and therefore the availability of water is directly influenced and influences the socio-economic life of the human being, therefore a stringent evaluation and monitoring of the water ecosystem is essential, accompanied by an increasingly stringent regulation by public authorities. With the recent issue of some regulatory measures (917/2017 / R / IDR - Quality regulation), ARERA[1] has defined a discipline of the technical quality of the integrated water service by adopting an asymmetrical approach e innovative that, starting from the conditions found in different contexts, guarantees the identification of correct stimuli and effective in promoting benefits for the benefit of service users, in a monitoring framework continuous and gradual implementation. In it are set quality objectives that the managers will have to reach quickly and in any case established on the basis of the starting quality level and therefore it is the same National Authority to request the application of innovative technologies in order to "be able to achieve significant service level objectives and in particular the quality of the water distributed quickly. " Furthermore, 20 years after its entry into force, the European Council Drinking Water Directive (Directive 98/83 / EC) is currently subject to revision, through a proposal aimed at reinforcing the law water of good quality, touching on the themes of accessibility, consumer protection, health and safety quality of life, up to the circular economy and sustainable development. The list with the limits relating to microbiological and chemical parameters has been expanded with respect to the current one with the addition of additional microbiological parameters as well as new and emerging substances, including endocrine disruptors and perfluoroalkyl substances (PFAS). For several parameters already present in the list the limit values have been reduced, while for lead and chromium it is provided for a transitional period of 10 years, at the end of which the current limit will be halved. A further very important aspect concerns the more restrictive levels compared to the current ones required for chlorites and chlorates, by-products of the disinfection of drinking water. The new limits proposed for chlorite and chlorate are not constantly guaranteed today in most of them Italian aqueducts and according to the results of a survey carried out during the year 2018 by Utilitalia (the Federation that brings together companies operating in the public

[1] Autorità di Regolazione per Energia Reti e Ambiente

services of water, the environment and electricity and Gas), in 60% of cases they are not currently effectively monitored.

1.2 Metrology for water pollutants. State of the art

The methods of analysing the quality of water intended for human consumption must be such as to guarantee results reliable and comparable. The analysis methods must meet the minimum performance characteristics defined by the standards[3][4].

A list, not exhaustive, of the pollutants that can be detected in the samples is reported in Table 1.

Table 1. Pollutant vs. detection methods

POLLUTANT	DETECTION
Heavy Metals	ICP-OS Spectrophotometry on complexed form
Non metals	Spectrophotometry on treated sample
Surfactants	Spectrophotometry on treated sample
BTEX	GC-FID, with previous concentration of the sample
Microplastics	Visual sorting + FTIR

In Italy, monitoring frequency of the quality of water is established by Legislative Decree 31/2001 [5] which combines the number of checks with the water volumes delivered during the year. The standard establishes a minimum frequency of control, in relation to the volume of water distributed.

There are two types of checks to be performed:

1. Routine check:
   The routine check aims at providing information on organoleptic and microbiological quality at regular intervals of the waters supplied for human consumption as well as information on the effectiveness of any water treatments drinking water (in particular disinfection), to ascertain

whether or not the water intended for human consumption responds to relevant parameter values set by this decree. At least the following must be checked routinely parameters:
-Aluminium and Ammonium
- Color, Conductivity and Turbidity
- Clostridium perfringens (including spores)
- Escherichia coli (E.coli)
- Hydrogen ion concentration
- Iron
- Nitrite
- Coliform bacteria at 37 ° C
- Residual disinfectant (if used)

2. Verification check:
   The verification check aims to provide the information necessary to ascertain whether all parameter values are contained in the decree they are respected. Table 2 describes the minimum frequencies for the various types of controls in relation to the volume of water distributed. Minimum sampling frequency and analysis for water intended for human consumption provided by a network of distribution, from tanks, or used in food businesses.

Table 2. Minimal check frequency related to water volume

Water Volume (m^3)	**Routine check (samples per year)**	**Verification check (Samples per year)**
≤ 100	-	-
> 100 ≤ 1000	4	1
> 1000 ≤ 10000	4	1 +1 every 3300 m^3/g
> 10000 ≤ 100000	4 +3 every 1000 m^3/g	3 +1 every 10000 m^3/g or 10000 fractions
> 100000	4 +3 every 1000 m^3/g and 1000 fraction	10 +1 every 25000 m^3/g and 10000 fractions

Controls are distributed throughout the year with a greater frequency for routine ones. By way of example, for an aqueduct that supplies around 1,000 cubic meters of water every day and which it takes about 5,000 inhabitants, the standard identifies a minimum frequency of 4 routine checks per year and 1 verification check per year.

Collection and sampling

Sampling is the procedure by which a fraction of a given substance, material or product is taken to provide a representative sample of the object of control in its entirety for the execution of tests or calibrations.

Elements analysis

For each element the recommended methods to perform the analysis are indicated. L'Istituto Superiore di Sanità, through these recommended methods, reconstructs the state of the art of best practices. Later, for each an element among those that will be analysed and monitored in the project, the control methods are suggested by the Istituto Superiore di Sanità.

Chromium (VI) methods [6], [7] is reported by way of example:

- *Method for atomic absorption spectrometry with electrothermal atomization*

  In general, the level of chromium detected in surface waters does not exceed 10 μg / L (rarely reaches 25 μg / L) mainly due to the low solubility of the trivalent form. The method is applied to the dissolved chromium fraction and to the one transferable at pH <2 present in the waters destined for human consumption. It is applicable in the concentration range between 2 - 20 μg / L. However, this range varies depending on the experimental conditions, as the characteristics of the instrument used, the quality and state of wear of the graphite oven and finally the quality and quantity of sample analysed influence instrumental sensitivity and limit of detection reachable. The method is based on the direct determination of chromium by atomic absorption spectrometry with electrothermal atomization (ETA - AAS). The previously acidified water sample comes injected, manually with micro-pipette or automatically by an autosampler, into the oven of graphite and subjected to the following thermal cycle: drying, incineration and atomization. They do not come reported interference due to the species most commonly found in the waters. The eventual "Matrix effects" must be removed using a suitable modifier or using the method of the note additions. The activation of the background corrector during the

instrumental measurement normally allows to suppress all non-specific absorptions produced by the matrix. The limit of detection of zinc by this method it is <2 μg / L;

- *Emission spectroscopic method with an inductive plasma source*
  In addition to chromium, this method also applies to aluminium, boron, cadmium, iron, manganese, nickel, lead, copper, sodium and vanadium. The sample and calibration solutions are appropriately sprayed, and the aerosol is transported in the plasma, where, following excitation phenomena, an emission spectrum is generated composed of characteristic lines of the elements present. These lines, after being separated by a system of dispersion, are detected simultaneously or sequentially by a photomultiplier or a solid-state detector that produces an electrical signal of intensity proportional to the intensity of the lines of issue. The concentration of analyte present in the sample is determined by comparison with one reference solution of known concentration. Background interference can be both negative and positive and are caused by one or more components of the matrix that, directly or indirectly, cause one of the following phenomena: variation of source temperature (with consequent variation of the "continuous" emitted by the source), bands of molecular emission, "diffuse light", ion-electron recombination. The extent of these interferences varies with the operating conditions adopted and only in the case of "diffuse light", some instrumental solutions (holographic lattice, double monochromator, etc.) can reduce significantly the problem. The measurement of the emission signal and the background, necessary for the correction of interference, is commonly performed on the sides of the analytical line. Knowing the type of substance interfering, its concentration and assuming that it is constant in the different samples is possible carry out the correction of this interference also by using the simulated matrix method. The Line interference occurs due to overlap (partial or total) between the analytical line and a line of another element. Correction of these interferences is not always possible and still involves a worsening of analytical precision. For these reasons it is advisable to use another analytical line, if available, free from interference. For this purpose, it is necessary to verify that, for the element under examination, the concentrations determined at the various spectral lines show a relative deviation not more than 10%. Otherwise discard the anomalous data. Both the accuracy and the

precision of the method are ≤ 10% of the parameter value for all the specified analytes.

1.3 Metrology for water pollutants. Economics

Surface water, ground water and drinkable water are currently monitored in all countries on a very large scale, driving both economic and political critical choices[8].

Water quality is the basis of all the economic and social systems. The pollution of marine, river and all surface waters is impacting worldwide, and has reached an important resonance in the last few decades. The disturbance of ecosystems mainly affects flora and fauna, altering the life of many organisms and leading to the extinction of others. It is clear that, these aspects have a striking impact on the life of the human being and on the large-scale economy.[9]

As shown in Figure 2, there are three Macro-Area in water quality monitoring:

- Drinkable water;
- Wastewater;
- Laboratory analysis.

Figure 2. Three different business cases. Drinkable water, wastewater and laboratory analysis.

Public authorities (but also private companies) must periodically control the contamination status of the ground waters and surface waters to enforce pollution reduction and environmental law compliance. Figure 3 presents European monitoring network from 1965 to 2004.

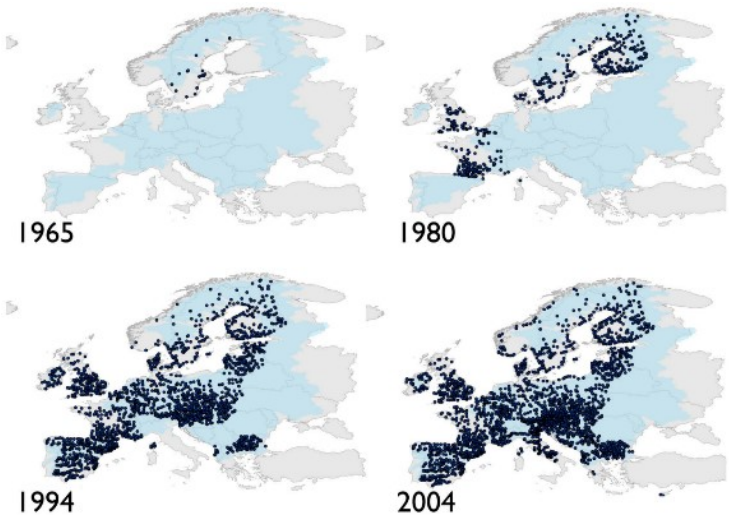


Figure 3. Evolution of the European water quality monitoring network (based on information from http://www.eea.europa.eu/themes/water/). Snapshots for 4 years in the time period 1965–2005.[8]

Most of the controls are performed by manually sampling the waters and by sending the samples to an authorized laboratory for the analysis. This means:

- High costs for sampling activities
- Long response times from the time of sampling to the analytical results
- Low sampling frequency and consequent low monitoring data resolution

A system, as shown in Figure 4, installed on site, equipped with a sampling pump, once configured and positioned, can send analytical data directly to the customer with no human intervention. This would mean:

- No costs for sampling activities. The only costs would be the first installation and the periodic maintenance for reagents refill and battery replacement.
- Very short response times (few minutes) from the time of sampling to the analytical results
- The possibility to achieve high sampling frequency and a consequent strict monitoring of the evolution of the site status.

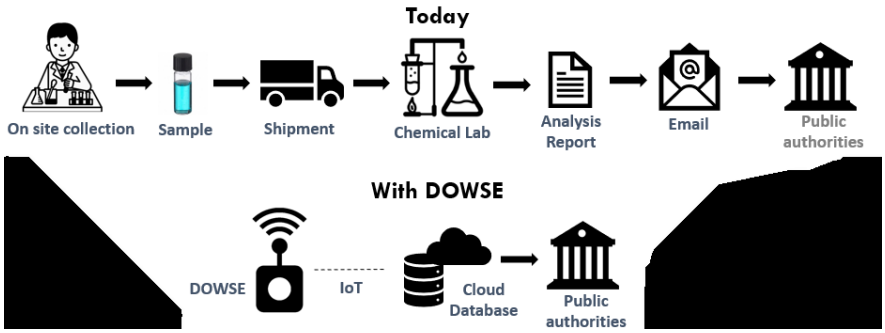


Figure 4. Conceptual comparison between actual analysis chain and a possible analysis chain with a smart monitoring sensor (DOWSE is a prototypal monitoring device developed by Microla Optoelectronics SRL and Chemifin SRL).

Table 3. Comparison between a piezometer monitoring over a 5 years period assuming to sample once per month and once per week. Source Chemifin SRL./Microla Optoelectronics SRL.

	Sampling Monthly		**Sampling Weekly**	
	TODAY	***TOMORROW***	***TODAY***	***TOMORROW***
Personnel	24.000 €	1.250 €	104.000 €	15.000 €
Instrumentation	3.000 €	30.000 €	3.000 €	30.000 €
Consumables	60 €	60 €	260 €	260 €
Purge water	6.000 €	600 €	26.000 €	2.600 €
Lab analysis*	1.800 €	0 €	7.800 €	0 €
Total	34.860 €	31.910 €	141.060 €	47.860 €
Samples	60	60	260	260
Cost per sample	581 €	532 € ***-8%***	543 €	184 € ***-66%***

Only in Italy there are over 3.000 public monitoring sites[2] for groundwater and surface waters.

Most of these are periodically monitored by public authorities to verify the pollution trends.

[2] European Environment Agency

Moreover, in Italy there are at least 4.000[3] contaminated sites under monitoring and they require at least 2 piezometers each. Therefore, there are over 8.000 piezometers for contaminated sites monitoring.

We can therefore assume an Italian potential market of 11.000 sites as a minimum (the actual figure might be much higher considering that only the province of Milan has registered over 5.000 piezometers[4]), as shown in Figure 5. According to Table 3, targeting only 5% of the Italian market would mean a potential of over 16 million €.

Figure 5. Italian surface water monitoring sites (green) and groundwater monitoring sites (red)[5].

[3] Camera dei Deputati, source ISPRA
[4] Città Metropolitana di Milano, Catasto pozzi e piezometri
[5] European Environment Agency

In Europe there are over 20.000 public monitoring sites[6] for groundwater and surface waters. Most of these are periodically monitored by public authorities to verify the pollution trends. Considering the ratio between public sites and piezometers for contaminated sites monitoring in Italy, we can assume a European potential market of over 60.000 sites (Figure 6). Targeting only 5% of the European market would mean a potential of 90 million €.

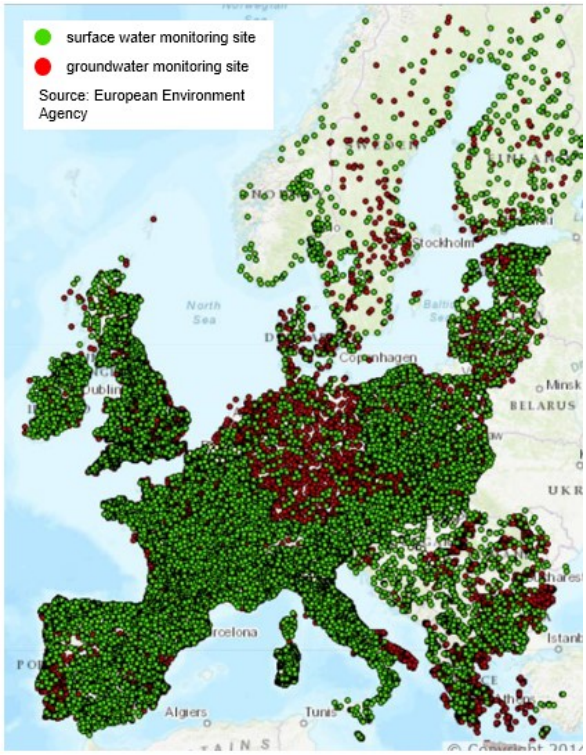


Figure 6. European surface water monitoring sites (green) and groundwater monitoring sites (red).[6]

Worldwide the market must be estimated, however the need for clean water is always increasing, especially in developing countries. USA, India, China, south-east Asia and Africa are only some of the other main geographies that we could target.

[6] European Environment Agency

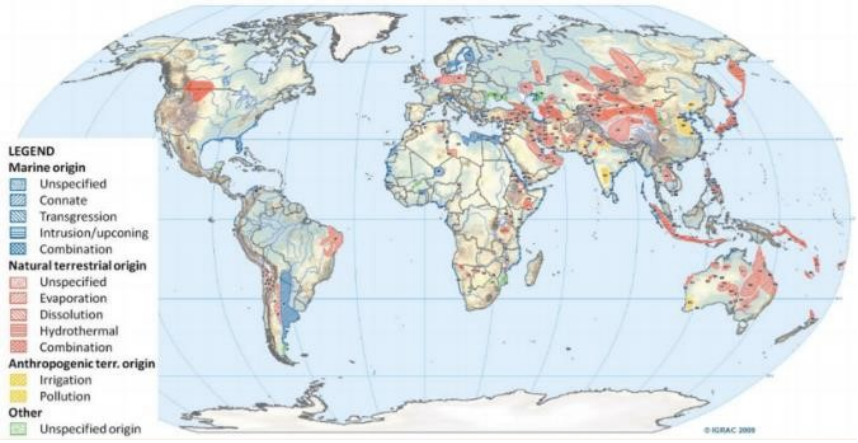


Figure 7. World distribution of major reported problems of arsenic content in groundwater at concentrations higher than 50 µg/L (Smedley, 2008; Margat and van der Gun, 2012

Industrial wastewater must comply with the environmental law and its composition must be within the required limits. In several industries Public authorities require the installation of an autosampler that collects daily samples so that they can eventually perform audits and analyze the samples. This is obviously a reactive control approach that implies:

- The need of an inspection on site performed by an auditor
- The analysis of samples weeks later
- A late reaction in case of non-conformity

With the installation of an automatic monitoring device, there would be a close monitoring of all wastewater industries without the need to send public auditors sampling waters in private companies. Also, this system would be great also for the companies themselves that could get real time data on the performance of their wastewater treatment plant without the need of an internal lab.

Potential industries for this application are:

- Steel mills
- Surface treatment plants
- Cement factories
- Pulp and paper factories
- Waste treatment factories
- Landfills
- Chemical processing plants

And any industry that has a potential hazardous discharge of wastewater

Only in Lombardy there are 1.800 companies under A.I.A.[7] (government authorization for emissions on the environment). In Italy there are over 68.000[8] manufacturing companies with more than 10 employees. It is more than reasonable to assume that at least 10% of them has a wastewater discharge. Therefore, we can estimate that at least 6.000 industries in Italy could be interested in our product. According to Table 3, addressing only 5% of this market means a potential business of 10 million €. Regarding chemical laboratories business case, today 50-60% of total costs in a chemical lab is labour. Chemical analysis is still a labour-intensive activity where automation could greatly increase the profitability of the activity and the competitiveness of the companies that embrace such process.

The most labour-intensive activity is sample filtration and preparation. An on-line device would focus on these activities to make them fully automatic. There are already some players on the market focusing on lab automation, but with different solutions and technologies.

Moreover, a patent evaluation was performed, in order to estimate the technological and economic value of a potential patent and to identify the patent landscape, using Patent Inspiration website[10].

The research filter was settled in a temporal interval until the end of 2017, considering the shadow window, and the query was "analy* AND water AND (autom* OR auton*)" in title or abstract.

In Figure 8 word cloud is represented as first output. The most recurrent words are analysis, chemical reagent, but also alarm, safety, detector, demonstrating the importance of water monitoring as a form of first control. The words

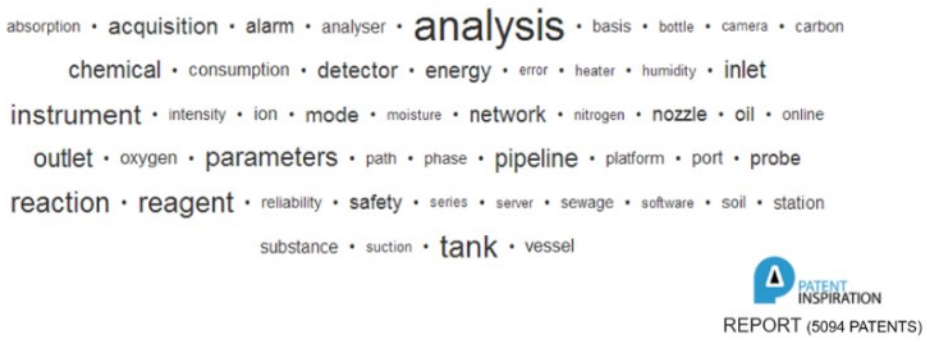


Figure 8. Word cloud

[7] ARPA
[8] ISTAT

As shown in Figure 9, the interest in this field is growing, reaching more than 1200 patents in the last two years.

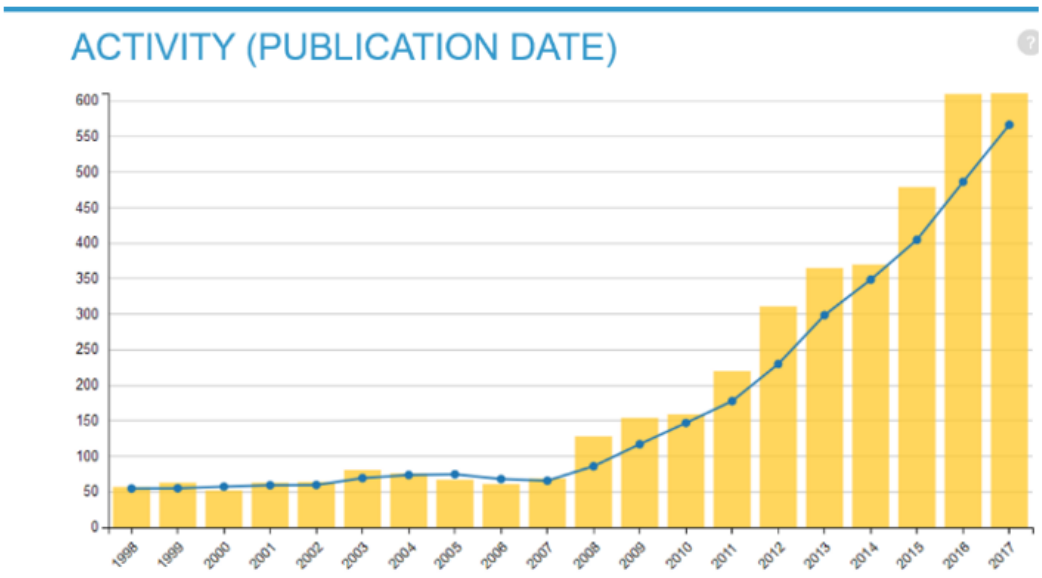


Figure 9. Patents publication activity from 1998 to 2017

An applicants evaluation, as reported in Figure 10, shown that Top 4 applicants own the majority of patents, so big players, such as Hitachi and Toshiba, are interested in this field.

Applicant	Patents
TOSHIBA KK	141
HITACHI LTD	138
HITACHI HIGH TECH CORP	97
SHIMADZU CORP	73
UNIV SICHUAN	29
STATE GRID CORP CHINA	24
MITSUBISHI HEAVY IND LTD	24
UNIV ZHEJIANG	23
OLYMPUS CORP	22
FUJI ELECTRIC CO LTD	21
TOSHIBA MEDICAL SYS CORP	20
PETROCHINA CO LTD	18

Figure 10. Applicants chart for number of patents. Big players are involved in this field.

1.4 Scope of the work

Water environmental monitoring is an important key to control and take care of human life and environment health. When water quality is poor, it affects not only aquatic life but the surrounding ecosystem as well [3], [11]–[17].

It is necessary that water monitoring instruments become within the reach of local authorities and control units, in order to increase the amount of data available on water pollution and facilitate shearing [8], [18]–[25].

The pre-analytical phase, sample collection, transport and preservation, has a considerable impact on the total uncertainty of the result of the analysis.

It should also be considered that the set of procedures and operations that occur between the time the sample is taken, and the performance of an analysis is one of the most delicate phases of the entire analytical procedure.[3], [4], [9], [26]–[29]

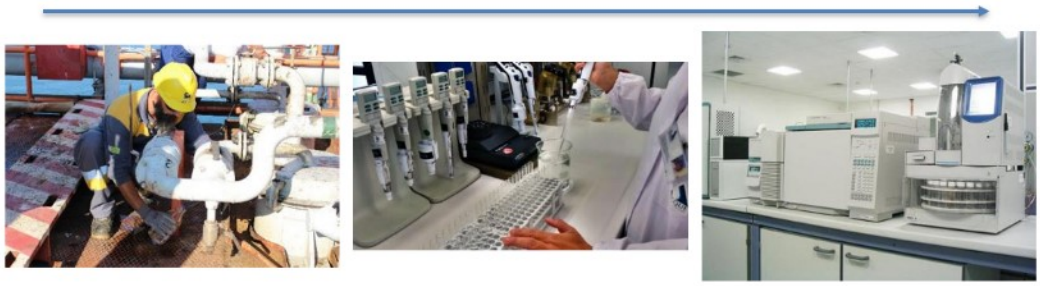


Figure 11. Present lab analysis chain. a. Manual sampling. b. Manual sample preparation. c. Expensive equipment for lab analysis.

Scope of this dissemination is to develop methods and procedures optimized for on-site miniaturized and portable devices for water quality monitoring. Methods and procedures must follow the ISO/TC (Technical Committee) 147 WATER QUALITY [30], which scope is *"Standardization in the field of water quality, including definition of terms, sampling of waters, measurement and reporting of water characteristics"*[9]. Related to ISO/TC 147 and its SCs there are 322 published ISO standards, 34 ISO standards under development, 43 Participating members and 48 Observing members. The final device must follow the ISO 15839:2003 *"Water quality -- On-line sensors/analyzing equipment for water -- Specifications and performance tests"*[7], which defines on-line sensor/analyzing equipment for water

[9] From International Organization for Standardization definition

quality measurements, defines terminology describing the performance characteristics of on-line sensors/analysing equipment and specifies the test procedures (for laboratory and field) to be used to evaluate the performance characteristics of on-line sensors/analysing equipment [31].

1.5 Thesis Outline

In Chapter 2, a description of measurement techniques for detection of dissolved heavy metals carried out, including the definition of spectrophotometry and its application in pollutants detection for water samples.

Chapter 3 addresses the methodology followed to achieve the scale down process, from laboratory methods to Lab-on-chip device. The chemical methods optimization and the Lab-on-chip design are presented in detail, a bench prototype realization is descripted, some early tests results are reported.

The realization and characterization of an embodiment of the lab-on-chip system is discussed and defined in Chapter 4. Moreover, on field deployment and measurement results are discussed in detail.

Chapter 5 shows the validation procedure, reporting test off-shore and inland results.

Finally, a summary is presented, with the contributions from this research and the opportunities for the future. Special emphasis is given to the next steps to be taken towards a better performance and the parallel generation of an increased number of substances.

Chapter 2

Spectrophotometric detection of dissolved heavy metals

2.1 Spectrophotometry: Overview and applications

An electromagnetic radiation, confirmed experimentally by Heinrich Hertz, is a wave of the electromagnetic field, that propagates, even in a vacuum, as the simultaneous propagation in the space of the oscillations of an electric field and a field magnetic [32].

The electromagnetic radiation is characterized by:

Frequency	ν	the number of vibrations per unit of time *Hertz (Hz)*
Period	T	is the time required to perform an oscillation complete (or to cover a space equal to wavelength)• the period is the inverse of the frequency (T = 1 / ν) *second*
Wavelength	λ	is the distance between two adjacent points in phase (e.g. between two consecutive highs) *m, μm, nm, Å*

Velocity of propagation	c	it depends on the medium in which the radiation is propagated in the vacuum is about 300 000 *km / s*

The frequency is a constant quantity for each radiation and in the visible field it characterizes the color of the light. Frequency and wavelength are inversely proportional:

$$\lambda = \frac{c}{\nu} \quad (1)$$

An electromagnetic radiation consists of 'discrete packets' of energy, called photons, whose energy depends on the frequency, according to the equation:

$$E = h * \nu \quad (2)$$

where h is the Planck constant: h = 6.63 * 10^{-34} J *s

The energy of a photon is sometimes also expressed in electron-volts (1eV = 1.6 x 10^{-19} J).

There are therefore various types of electromagnetic radiation, which differ in their wavelength (and of consequently due to their frequency and energy); they are summarized in the electromagnetic radiation spectrum (Figure 12).

Visible radiation represents only a small part of the electromagnetic spectrum. To the different visible radiations, which differ for their wavelength (and consequently for their frequency and energy) correspond to the different colors.

A radiation of a single color obtained by dispersion, characterized by a very precise length wave is called monochromatic. More precisely, we speak of a monochromatic light beam when it consists of radiations of only one frequency and wavelength. We speak instead of a polychromatic light beam when it consists of radiations of different frequency and length waves.

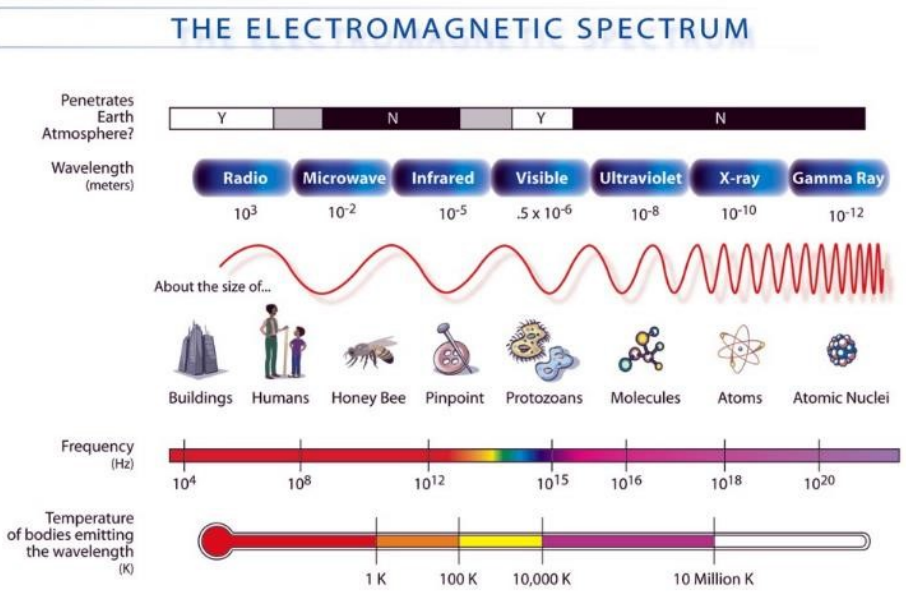


Figure 12. Electromagnetic Spectrum. , NASA [CC][10]

The spectrochemical analysis methods are based on the analysis of the spectrum of substances, which can be of emission or absorption:

- an emission spectrum is obtained when analyzing a beam of light emitted, under appropriate conditions, from a substance;
- an absorption spectrum is obtained when analyzing a light beam after it has passed through one substance.

For the same substance the emission and absorption spectrum are the same as the positive and the negative of a photograph, in the sense that a radiation present in the emission spectrum will be missing in that of absorption.

Spectrophotometric analysis consists of measuring electromagnetic radiation to obtain chemical analyzes; it can provide both qualitative and quantitative information. In fact, each substance absorbs or emits radiations of a well-defined wavelength:

- the spectrum analysis allows then to identify the nature of the substance under examination;

[10] https://mynasadata.larc.nasa.gov/sites/default/files/inline-images/EM_Spectrum3-new.jpg)

- the measurement of the intensity of the emitted or absorbed radiation allows to go back to the quantity of substance analyzed.

2.2 Scientific and Technical aspects in water heavy metals detection

Metals dissolved in aqueous solution are very common contaminants, but the identification of which presents various difficulties, due to interferences and minimum concentrations. This means that the methods of individuation must be highly selective and resolute. [33].

The most common techniques [34] are:

1. Spectroscopic techniques, as graphite furnace atomic absorption spectroscopy (GF-AAS) and inductively coupled plasma mass spectroscopy (ICP-MS);
2. Neutron activation analysis (NAA);
3. Potentiometry with ion selective electrodes (ISE).

The advantage of the mentioned methods certainly lies in the precision, accuracy and repeatability, but all of them involve process steps requiring laboratory equipment that cannot be transported or installed on site.

A technique that makes it possible to identify the presence of metal-type contaminants in different types of aqueous samples is the UV / VIS spectroscopy, which, despite a cost-effectiveness, maintains resolution, sensitivity and limit of detection comparable with more complex and expensive techniques like the inductively coupled plasma [35].

The UV/VIS spectroscopy deals with the transitions between different electronic states of the molecule. Spectra of absorption consist of many lines very close together, to appear a continuous, that is, one band. The 'fine structure' due to rotational and vibrational transitions is generally not detectable, if not in the case of electronic spectra of rarefied gases performed with high resolution spectrographs.

To perform qualitative analyzes, continuous spectrum, polychromatic rays are used. A monochromator divides the polychromatic beam in several monochromatic beams. Monochromatic beams pass through the sample. The sample absorbs different radiations with different intensity. Therefore, reporting the values

recorded in a wavelength-absorption graph, we obtain the spectrum of absorption of the tested substance, as shown in Figure 13.

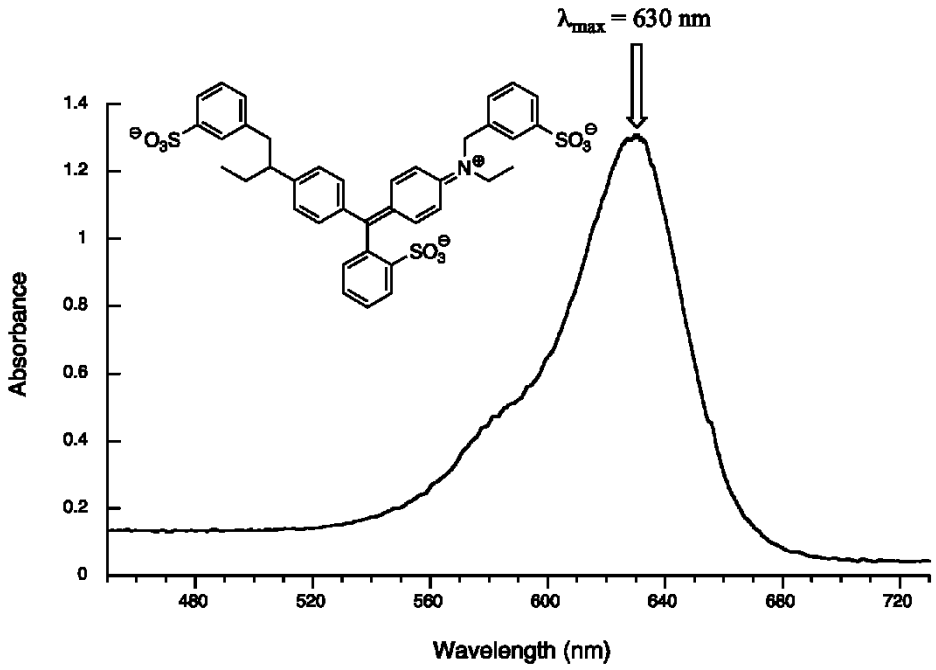


Figure 13. Food colorant Blue #1 absorption spectrum. Source https://chem.libretexts.org

Because of each substance has its absorption spectrum, the examination of these spectra allows to identify a substance, by direct comparison with known samples or through spectral databases or by check the degree of purity. The techniques that best lend themselves to qualitative, especially organic, analyzes are the infrared spectroscopy, in which each substance has numerous well-separated characteristic bands, above all nuclear magnetic resonance, which provides series of peaks directly connected to the structure of the molecule.

To perform quantitative analyzes, monochromatic beams are used, consisting of radiations of only one frequency. A quantitative analysis is possible thanks to sample absorption is depends directly to sample concentration.

Two quantities are used in quantitative analyzes, transmittance and absorbance. Using photodetectors, devices that can measure the intensity of luminous flux, two important physical quantities can be detected(Figure 14):

- I_0: intensity of the luminous flux at the entrance of the cuvette which contains the sample

- I: intensity of the luminous flux at the outlet of the cuvette which contains the sample.

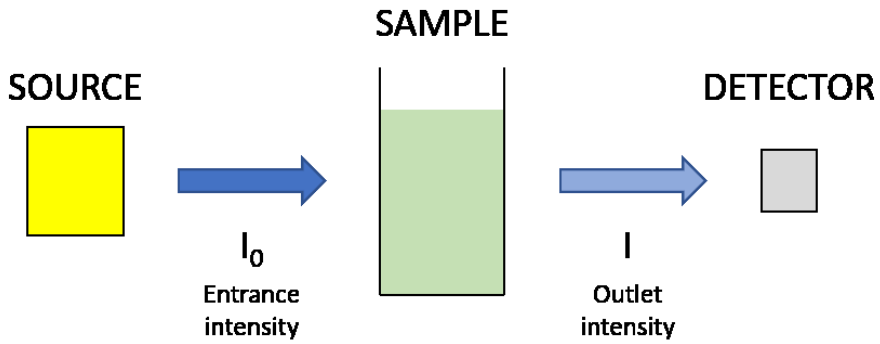


Figure 14. Quantitative analysis representation. Light intensity is partially absorbed by sample.

Transmittance is the fraction between I and I_0.

$$T = \frac{I}{I_0} \tag{3}$$

Absorbance is the negative logarithm of T

$$A = -\log_{10} T \tag{4}$$

According to Lambert-Beer law

$$A = \varepsilon * b * C \tag{5}$$

Where

A = Absorbance

ε = Molar attenuation coefficient (mol^{-1} L cm^{-1})

b = Optical path (cm)

C = Molar concentration (mol/L).

So, Lambert-Beer Law describes a direct proportionality between Absorbance and molar concentration. This equation represents a straight line passing through the origin of the axes and in which m = ε * b is the angular coefficient.

As the solute concentration increases there are notable deviations with consequent inadequate reliability of the analytical data. A high concentration means more particles in the sample, which means more collisions. Because of the formation of different aggregates in the solution, the measured absorption doesn't increase linearly with concentration. Lambert-Beer law will be valid for low concentration values. It should also be remembered that, as the concentration increases, there is an increase in the refractive index and therefore greater dispersion of the beam when crossing the solution itself. Another condition of validity of the Lambert-Beer law is that the luminous radiations, which cross the solution examined, must be monochromatic. [36]

The quantitative analysis of a contaminant in an aqueous sample involves the following steps:

- *Sample preparation*
  The analysis can be conducted directly on the solution of the substance only if it presents the maximum absorption in the wavelength range of the instrument. For this reason, appropriate chemical reactions are used between the test substance and appropriate reagents that lead to formation of compounds with absorption maxima in the required wavelength range;
- *Zero-setting and calibration*
  The zero-setting and calibration of an instrument is based on the definitions of transmittance and absorbance. For a solution with infinite concentration the instrument must have T = 0 and infinite absorbance.
  For a solution with no concentration the instrument must have T = 1 and A = 0. Interposing a perfectly opaque screen on the path of the light rays the instrument must measures T = 0 on the transmittance scale; for the most tools, this operation is automatic and therefore it is not necessary to execute it (otherwise there will be a device capable of imposing the condition T = 0);
- *Zero-setting against the Blank sample*
  The calibration at zero concentration A = 0 is instead carried out with the so-called 'zeroing against the Blank sample'. Reflection, refraction and absorption by the cell walls, the solvent and all the reagents added to form the colored compound occur during measuring phase. The

absorbance measured would therefore be affected by numerous factors not related to concentration of the test substance, thus leading to errors in the determination of the concentration of the latter. Before measuring the absorbance of the sample under examination, the absorbance is zeroed introducing the "blank sample", that is a cell identical to that of the sample and which contains a solution the most possible similar to that of the sample but in which only the test substance is absent. By not zeroing against blank sample you will lose the direct proportionality between A and concentration, that is, a straight line passing through the origin will not be obtained in graph C-A;

- *Concentration determination*
  Most instruments are equipped with photoelectric cells that produce an electrical signal dependent on light intensity. The electrical signal is then processed electronically, until obtaining an analogue or digital reading of A and / or T. Once the absorbance is obtained, with the usual zeroing against blank sample, the determination of the concentration can be obtained following different methods, remembering that absorbance and concentration are directly proportional according to equation (5).
  The *calibration curve method* is the most used method. Several samples, at least three, containing the test compound are prepared at different known concentrations and their absorbance is measured. Therefore, there will be a series of concentration values associated with the respective values of absorbance. Figure 15 shows how, reporting these values in a Cartesian graph, the “Calibration curve” is obtained. To calculate C_x, A_x is measured and the problem is solved graphically.

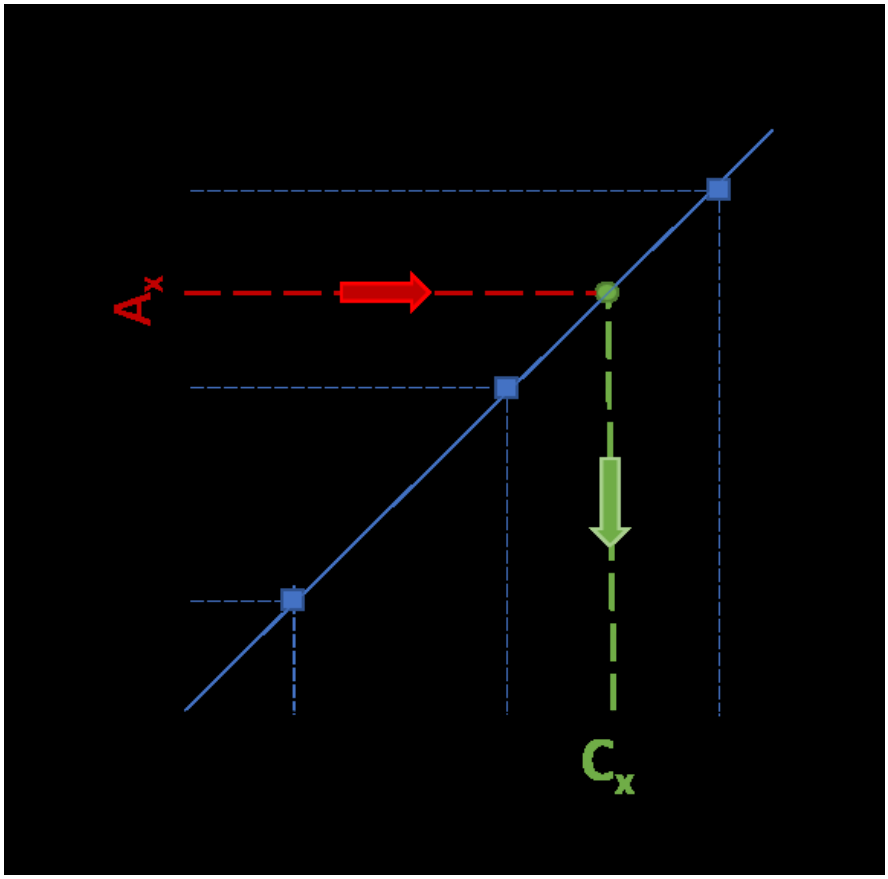


Figure 15. This graph represents a "Calibration curve" and the process flow for concentration determination.

2.3 Spectrophotometric instrumentation

The instruments used in spectrophotometric analysis are spectrophotometers and colorimeters. The essential difference between these two types of instruments consists in the fact that in the colorimeters there is one greater bandwidth. A double-beam spectrophotometer can reach bandwidths lower than 1 nm, thus using a light monochromatic, important condition for the respect of the law of Lambert-Beer and essential for useful spectra for qualitative purposes. These bandwidth differences depend on the fact that different monochromators are used: moles colorimeters optical or interferential filters are used, while in spectrophotometers prisms or gratings are used diffraction, associated with slit systems. As regards UV-visible spectrophotometers, of which an example is shown by Figure 16, the "single-beam" and the "double-beam" are the most common types. Single beam systems are used

without problems for quantitative analysis; double beam spectrophotometers are more complex and expensive, but allow quantitative analyses with great resolution than single beam ones.

Figure 16. Lambda 1050+ UV/VIs Spectrophotometer by PerkinElmer.

From a conceptual point of view, a spectrophotometer follows the principle scheme shown by Figure 17.

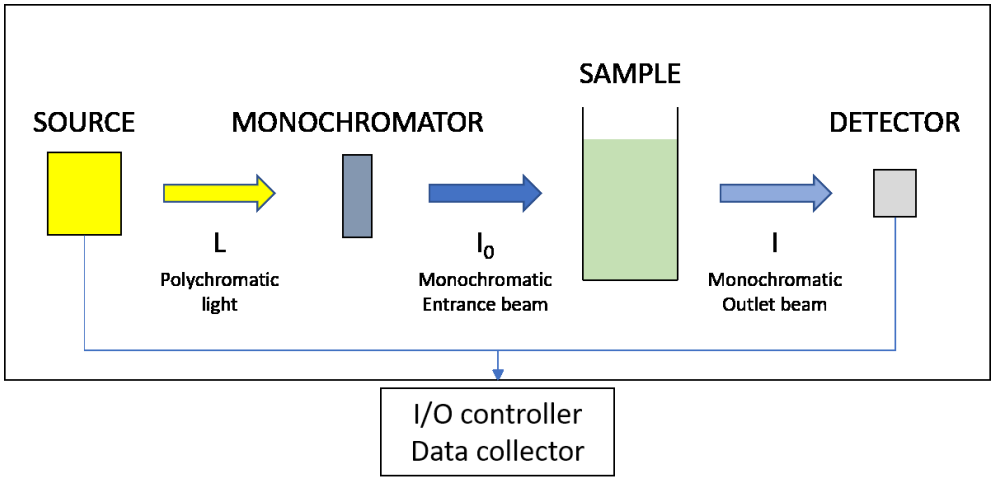


Figure 17. Scheme which shows the components of a classical spectrophotometer.

The source must emit polychromatic radiation, consequently containing all the wavelengths of the field required. Incandescent lamps (tungsten filament, quartz-iodine lamps or tungsten-halogen lamps) are used for the visible region. For the UV region, gas- discharge lamps are used (deuterium or hydrogen); they consist of a

quartz ampoule containing the rarefied gas, in which it is activated, between two electrodes, an electric discharge with the consequent emission of radiations with continuous spectrum. The UV-visible spectrophotometers will therefore have within them two different lamps, which come appropriately interchanged by the internal mechanism. An optical system, comprising a slit, lens and mirrors, is placed after the source, in order to avoid diffusing light into the instrument and to collimate the beam.

The monochromator is one of the critical components of the instrument. There are two types of monochromators. The first one is based on filters, optical or interferential kind, which block a part of the light and allow only the part to pass desired. A second one is based on a dispersing element, usually a prism or a grid, which separate the various components of the radiation and allow the subsequent selection of the desired band.

Optical filters contain appropriate substances that absorb most of the visible radiation except for an interval of wavelengths chosen, called band. Even if more filters are combined, pass bands remain in the order of 50nm and always at the expense of a weakening of the radius even for the required λ. They are used only in colorimeters.

Interferential filters are based on a typically electromagnetic phenomenon, the interference, that causes strengthening or weakening between two depending on they are in phase each other or not. They are more efficient than filters based on absorption, allowing pass bands of the amplitude of 20nm, in the visible range. Interferential filters are however more expensive and are used in the best colorimeters. The monochromators based on dispersant elements are those used in high-end spectrophotometers. They have a polychromatic beam engraved on an object, a prism or a grid, capable of divert the different radiations with different angles. The outgoing radiation will be the one that passes through the exit slit.

The cuvettes are the container of the sample to be examined. They must have a very precise optical path through the sample, depending on the concentration of the sample itself. Quartz cuvettes (SiO_2) are used in the UV range, glass or quartz cuvettes are used in the VIS range. In IR range, cuvettes made in in NaCl, KBr, CaF2 are necessary. More recently, with the advent of particularly sensitive devices, it has been possible thin optical fibers coming out of the instrument, going on to analyze very low volumes of samples (for example parts of cells).

Detectors are transducers capable of generating an electrical signal that depends on the energy of the radiation. This electrical signal, proportional to the light intensity, is transferred to an indicator. Since these are the parts of the instrument which performs the actual measurement, they have a crucial importance, as regards

both the sensitivity and the accuracy of the spectrophotometer. In UV-visible photovoltaic and photoconductive cells, phototubes and photomultipliers, photodiodes can be used. Photovoltaic and photoconductive cells are based on semiconductors that generate a voltage directly proportional to the intensity of the incident radiation. They are not very sensitive and do not cover all UV-visible range. However, they are resistant and inexpensive. For this reason, they are used in colorimeters or simple low-cost photometers. Phototubes and photomultipliers are based on the photoelectric effect, which consists of the emission of electrons from a material when it is hit by luminous radiation. The number of electrons emitted is proportional to the intensity of the incident radiation. Finally, photodiodes are microscopic chips made of silicon or germanium, which generate a voltage if invested from light radiation. They have lower sensitivity than photomultipliers, but they can be included in large numbers on a single silicon chip, lending itself so effectively to the construction of spectrophotometers with diode array.

Chapter 3

Scale down the laboratory method: From Lab to Lab-on-chip

3.1 Chemical methods optimization for miniaturization

The purpose of this chapter is to present the optimization of the methods and the technologies described in the previous chapter, in order to obtain a portable device, energy-efficient, cost-effective and capable of performing an automatic measure chain, without the intervention of any operator.

In literature, some examples of in-situ monitoring can be found. Lambrou et al. [18] reported about a low cost sensor node for drinking water and detection algorithm. However, this kind of sensor is developed for in-pipe application. Also Hangan et al. [37] described water monitoring system developing, showing a usage scenario. Encinas et al. [38] presented a monitoring system for water quality and its integration on an IoT (Internet of Things) prospective.

However, these devices were platforms which integrate commercial sensors, they were developed for a static installation, so they were not portable, and they were not developed on a Lab-on-chip approach, so they didn't replicate laboratory conditions. The device, developed in this thesis, were designed in order to communicate with an Autonomous Underwater vehicle, in order to perform dynamic campaigns to monitoring a large area in proximity to sites of interest. The

optical sensor, which perform quantitative measure of dissolved heavy metals, replicate laboratory device, improving energy and cost effectiveness, and significantly reducing dimension, in order to be installed in an AUV payload.

Regarding chemical methods, Cr(VI), Zinc, Nickel and Copper detection methods and their optimization will be presented.

3.1.1 Cr(VI) detection method

Part of the work described in this chapter has been previously published in my paper "*Cr(VI) in Water: Continuous, on Site Spectrophotometric Determination Laboratory test preliminary to microfluidic device prototyping"* [26].

Among the metals dissolved in aqueous solutions, hexavalent chromium is of great importance. Several studies have reported its toxicity in aqueous ecosystems, even compared to Chromium (III) [17] [39]. This type of dissolved metal can be quantified by spectrophotometric techniques, maintaining sensitivity, limit of quantification and competitive selectivity with respect to other techniques, such as ICP, which involve higher costs and more complex procedures [35]. Diphenylcarbazide (DFC) method is the most common colorimetric method for Cr(VI) detecting in aqueous samples, in the concentration range [0.1-1.0] ppm (parts-per-million or mg/L) [40]. The chemical reaction between DFC and Chromates in acid environment, as shown in Figure 18, generates a purple compound.

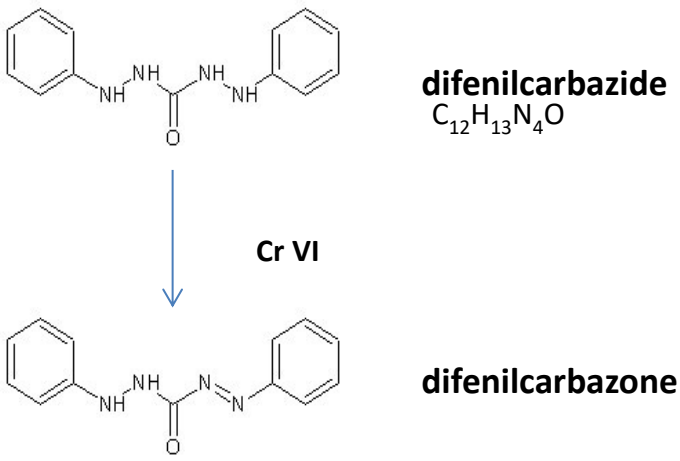


Figure 18. DFC is transparent while in the form complexed is purple

Figure 19 shows how the color intensity of the complex compound increases directly with the hexavalent chromium concentration.

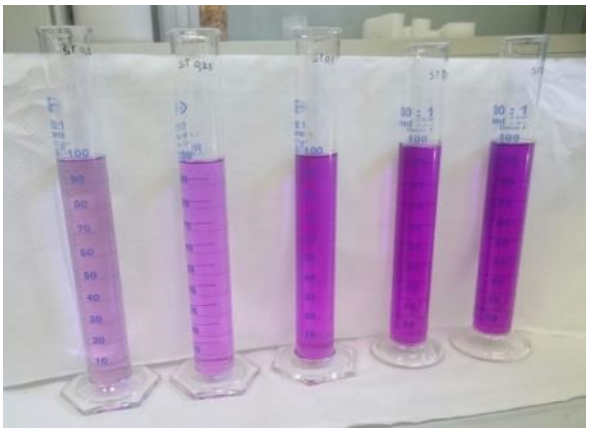


Figure 19. Increasing concentration of Cr(VI) solutions, treated with DFC, before spectrophotometer measurement. From the left: 0.1 - 0.25 - 0.5 - 0.75 – 1 ppm

The complexed compound has an absorbance maximum peak in the proximity of 540 nm (Figure 20).

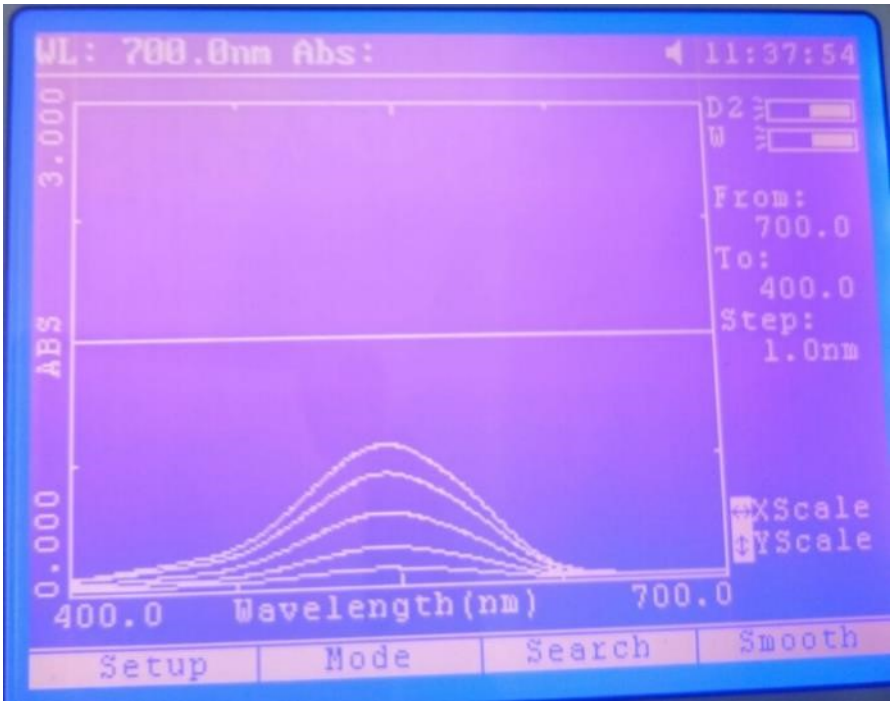


Figure 20. Spectra acquired for solutions with different Cr(VI) concentration. From the bottom: 0.1 - 0.25 - 0.5 - 0.75 - 1 ppm

The first experimental tests aimed at testing the efficacy and reproducibility of the diphenylcarbazide (DFC) method reported in the literature. The starting materials were Cr(VI) standard solution in water (ST147), with a concentration of 1000mg / L, 1-5 DFC (Sigma Aldrich), Acetone (Sigma Aldrich), Sulfuric acid 98% (Sigma Aldrich), diluted samples of Cr(VI). The DFC solution was obtained dissolving 0.25g of DFC in 50mL of acetone.

The diluted sulfuric acid 1:1 v/v, was obtained mixing 10mL of concentrated sulfuric acid in 10mL of distilled water. In order to achieve complexed samples, with a range of concentrations from 0.1 to 1 mg/L, 1 mL of diluted H_2SO_4 and 2 mL of DFC solution were added. In addition, a "blank sample" is prepared to be inserted into the spectrophotometer as a reference, adding 1mL of diluted sulfuric acid and 2mL of DFC solution to 100 mL of distilled water (which therefore does not contain chromium ions). The various samples are mixed and left to rest for 10 minutes. Then, the absorbance was read from the 6850 spectrophotometer (Janway), using optical glass cuvettes with an optical path of 1cm. Regular spectra were obtained, without jagged or secondary peaks. The spectrum was acquired from 400 to 700 nm, and the absorbances were recorded. Table 4 shows the numerical results.

Table 4. Absorbance measured for the different Cr(VI) samples.

Concentration *[mg/L]*	**Absorbance** ***max***	**Absorbance** *540nm*	**Absorbance** *536nm*	**Absorbance** *532nm*
0,1	0,100	0,100	0,099	0,097
0,25	0,224	0,223	0,221	0,216
0,5	0,427	0,426	0,421	0,411
0,75	0,660	0,659	0,651	0,637
1	0,833	0,831	0,821	0,802

The coefficient of determination of these data, as shown by the graph in Figure 21, were proximal to 1, index of a good verification of the Lambert and Beer law.

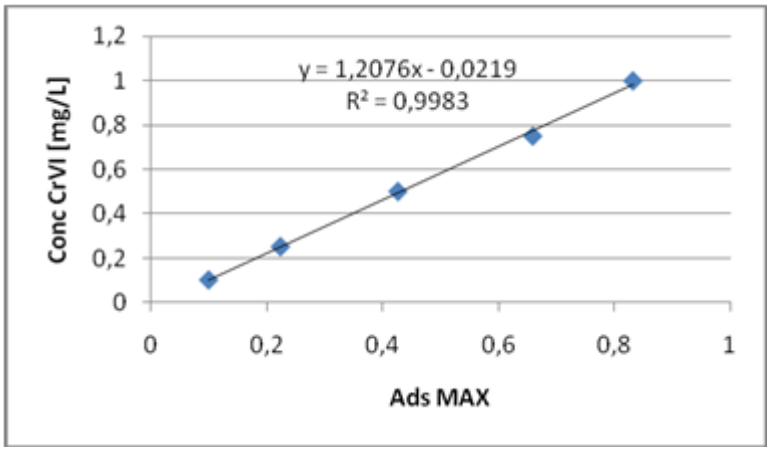


Figure 21. Calibration curve obtained with the maximum absorbances at different concentrations at 540nm.

The optimization of DFC Method continued with exhaustive measures to determine the influence of the various measurement parameters on the measurement itself. The parameters of interest for method optimization according to the restrictions imposed by miniaturization were:

- **Measure linearity** using a wavelength that does not correspond to the absorbance peak of the solution, dictated by technical requirements related to the choice of the laser diode to be used. The **laser diode** was chosen, instead of lamp or monochromator, because of its energy-efficiency, its low price and its portability;
- **Rinsing influence**, in order to highlight the effect of rinses conducted in different ways on consecutive measures using different concentration samples, to evaluate the influence of the automation of the rinse phase in automatic measurement cycle;
- Influence of the **quantity of reagents in the sample**, to assess the dose tolerance in the dosing phase in the automatic measurement cycle;
- Influence of **time and of the mixing mode**, to evaluate whether, in the presence of manual mixing (in our case) that facilitates the diffusion of the DFC in solution, waiting times for color development can be reduced;
- Influence of the **solvent in which the DFC is dissolved**, as acetone, indicated, as a solvent in the standard reference recipe, can create

compatibility problems with some materials in which it must be contained or constitute the microfluidic system in which it flows. In fact, with its use, materials with enough resistance to this solvent would be required, which however have a greater cost and present difficulties in use;

- **Aging and stability over time** of the DFC solution, analysis aimed at assessing how long the DFC solution can be stored without losing its effectiveness in the complexation of Cr (VI) ions.

According to standard method, previous described, solutions of Cr (VI) were prepared, starting from the standard 1000 ppm, with known concentrations equal to: 0.1 ppm, 0.25 ppm, 0.5 ppm and 1 ppm. The samples, of 100 mL volume, were treated according to the standard recipe and analyzed by spectrophotometer, recording the absorbance values at two wavelengths of interest, 540 nm, which is the absorbance peak, and 520 nm, wavelength chosen for laser diode. Results are reported in Table 5 and Figure 22.

Table 5. Absorbance at @540 nm and @520 nm

Concentration *[mg/L]*	**Absorbance** *540nm*	**Absorbance** *520nm*
0,1	0,1	0,087
0,25	0,2	0,167
0,5	0,39	0,325
0,75	0,559	0,468
1	0,767	0,646

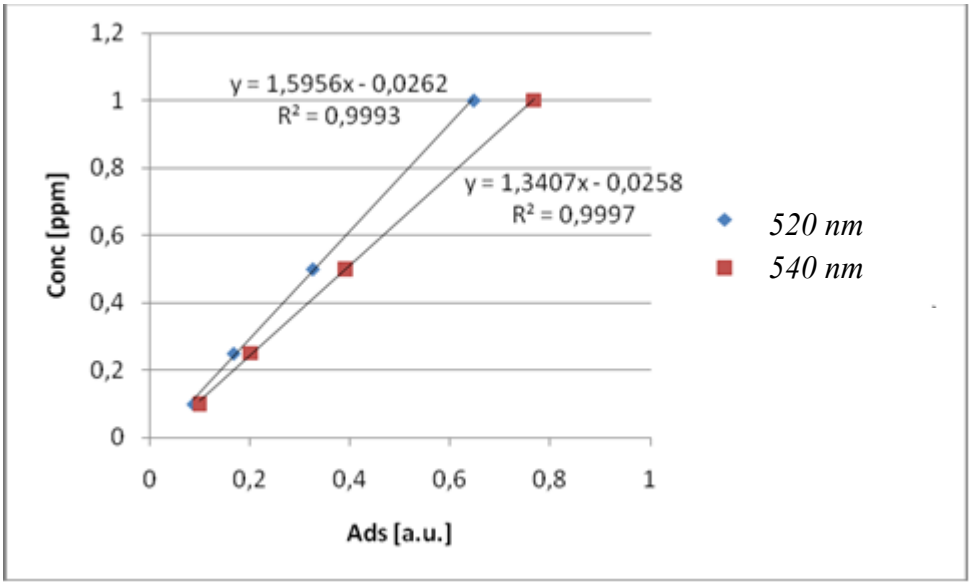


Figure 22. Calibration curve at peak absorbance (@540 nm) and chosen wavelength (@520 nm)

The linearity and the validity of the Lambert and Beer law are also maintained by reading the absorbance value at 520 nm.

Further optimizations were imposed by needs of miniaturization on the prototype: involved volumes should be reduced, highly concentrated acidic media avoided and process time limited. Moreover, while in the lab the cuvettes are carefully washed between following analysis, in a continuous setting it's not so simple; the best rinsing protocol was assessed to limit, as much as possible, the error due to contamination of the sample with the previous one. Two standard solutions have been prepared at 0.25 ppm and 1 ppm.

On both, the measurement was made using the perfectly clean and dry cuvette. The test methods were:

- Consecutive measure of the solution with the same concentration (0.25 – flush -0.25; 1- flush -1)
- Consecutive measure of solutions with different concentrations. The most concentrated solution was inoculated in the cuvette, then the cuvette was emptied and rinsed. Finally, the least concentrated solution was loaded, and the measurement was made. This highlights the influence of more concentrated residues in the cuvette, which can increase the read value on the less concentrated solution.

From the results obtained it is noted that if you work with solutions with the same concentration, rinse, however, dilutes the sample, leading to a slight error on the measure, very limited if you do not rinse. However, the extent of the error, computed as absolute percentage error, made on the read absorption is, in the worst case, 1.61% (1.79% considering the error on the concentration calculated with the calibration line) , which may be considered negligible. If, on the other hand, a measurement of a higher concentration solution (1 ppm) is preceded by the most diluted one (0.25 ppm) the effect of rinsing is different: without rinsing, it reads a greater concentration due to the residue with higher concentration in the rinse. The mistake made in this case is 15% on the absorption read. Rinsing, however, actually reduces this error, bringing it back to values similar to the previous cases (solutions with the same concentration). The error made is, in fact, in the worst case, 1.07%.

Therefore, it can be concluded that to measure solutions with different concentrations, it is better to provide a rinse between measures, but it is not necessary to repeat this operation several times. Regarding the possibility to replace acetone with e less aggressive solvent, it was decided to investigate the possibility of replacing acetone with another less aggressive solvent with respect to the plastic components to be used in the prototype. Literature [13] refers to the use of methanol as a solvent for DFC, so it is decided to test both methanol (MeOH) and ethanol (EtOH), with the same concentration used for acetone. The results, reported in Table 6 taking as reference the absorbance value read for the acetone, show that both alcohols provide very similar absorbance values. However, it was decided to exclude ethanol because it showed difficulties in dissolving the DFC: in fact, a vigorous and prolonged mechanical agitation was necessary, which did not however eliminate residues of undissolved DFC on the solvent bottom. Also, to dissolve the DFC in methanol it was necessary to shake the solution, while with the acetone the dissolution is complete and immediate, but in a short time (a few seconds) the complete dissolution is reached.

Table 6. DFC solvent influence on absorbance. Ref. acetone

Solvent	**Concentration** *[mg/L]*	**Absorbance** *540nm*	**Absorbance** *520nm*
Acetone	**0,25**	0.203	0.171
MeOH	**0,25**	0,203	0,171
EtOH	**0,25**	0,208	0,176

Considering the benefits for the choice of materials for the prototype, it is therefore proposed to use methanol instead of acetone, as a solvent for DFC.

The linearity was also checked making a calibration curve (Figure 23).

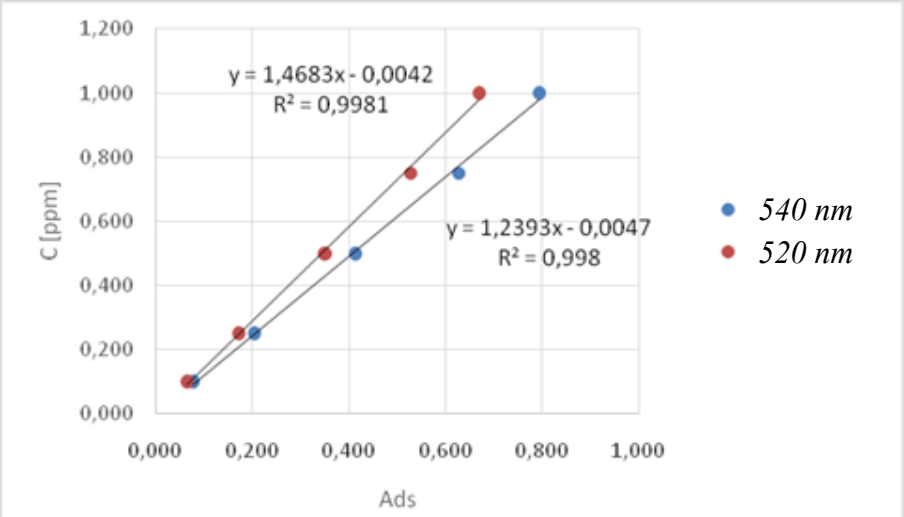


Figure 23. Calibration curve using methanol as solvent for DFC

To evaluate reagents aging, a solution of DFC in acetone, 0.5g / 100mL, was prepared and stored and used for measurements on samples with a known Cr (VI) concentration of 0.25 ppm, at different times. Table 7 shows the spectrophotometric readings at the two λ considered and it can be seen how, if the acidification of the sample is made at the time of the measurement, even after 3 days from the preparation, the DFC solution provides the same absorbance value, which there is no degradation due to aging. The measurements were carried out both on freshly prepared Cr (VI) samples and on samples previously prepared the day before, but not acidified. Different results are obtained instead in the case in which the sample of Cr (VI) is prepared, acidified with the solution of H_2SO_4 1: 1 (1 mL / 100 mL sample) and stored until the moment of measurement. The aged DFC solution is added to the time set for the measurement, but values of absorbance that are clearly lower than the reference (corresponding to time 0) are read, with an error of about 26%. Finally, on a sample with preventive acidification, 96 hours aged was tested, where the absorbance peak is completely absent, after reading and then after adding the DFC, to adding 1 mL of 1: 1 sulfuric acid solution and repeat the measure. The absorbance returned to values close to those of the reference (0.2 for λ = 540 nm and 0.167 for λ = 520 nm). Aging tests, carried out on samples with acidification at the time of measurement, were also carried out on DFC solutions in methanol,

which had given good results in terms of calibration line. Also, in this case, after 48 h of aging, there were no deviations from the reading on the reference.

Table 7. Aging test for DFC. Ref. time 0.

t [h]	Solvent	Absorbance *540nm*	Absorbance *520nm*	Acidification
0	Acetone	0,203	0,171	No
6	Acetone	0,202	0,171	No
24	Acetone	0.202	0.171	No
72	Acetone	0.213	0.179	No
96	Acetone	0.202	0.171	No
24	Acetone	0,150	0,126	Yes
48	Acetone	0,151	0,126	Yes
0	Methanol	0.204	0.172	No
24	Methanol	0.2	0.169	No
48	Methanol	0.205	0.172	No

Therefore, it can be concluded that, by dissolving the DFC both in acetone and in methanol, it is possible to store it for several days, without altering the outcome of the measurement. The parameters that, for each previous optimization, provided the best results, were combined in a method, which will be called "optimized method", and with this method the calibration line was re-built to verify any deviations from Lambert's law and Beer, due to the synergistic effect of the variations compared to the standard method. The results (Table 8 and Figure 24) show that the optimized method is well applicable for analysis, at least in the range of Cr (VI) concentrations considered.

Table 8. Values of calibration curves using optimized method

Concentration *[mg/L]*	**Absorbance** *540nm*	**Absorbance** *520nm*
0,1	0,080	0,068
0,25	0,185	0,156
0,5	0,351	0,298
0,75	0,570	0,482
1	0,715	0,605

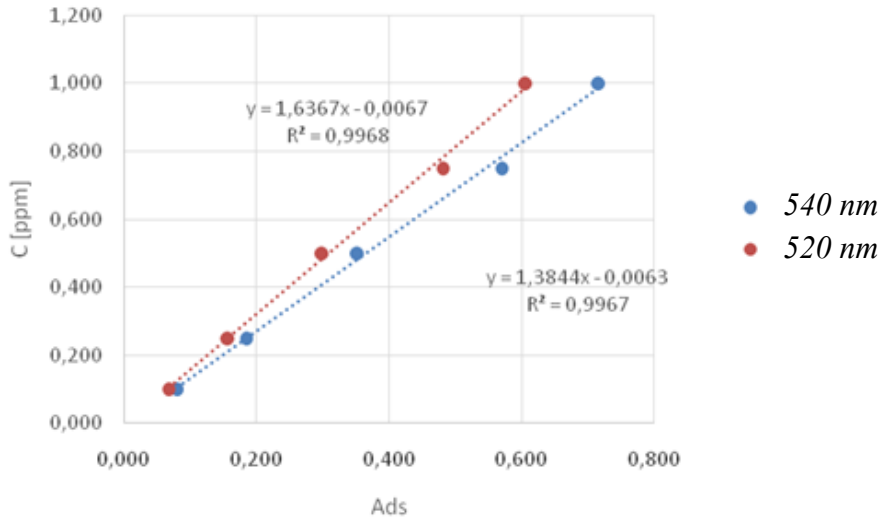


Figure 24. Calibration curve using optimized method.

It was also evaluated the tolerances of the reagents, and how they have influences on the measurement. DFC concentration must be in range between 0.1 and 0.5 g / 100 mL of methanol. For minor concentrations, the absorbance collapses; higher concentrations do not allow the DFC to completely dissolve in the solvent. Volume of DFC solution (0.2 g / 100 mL methanol) has to be in range between 2 and 5 mL / 100 mL of sample. For smaller volumes, the amount of DFC is not enough for the total complexation of the metal; larger volumes result in a measurement error. In details, 5 mL / 100 mL carry a 3% error on the standard; a volume of 10 mL / 100 mL of sample raise the error to 6.3%. The lower limit value of acid to be added to the sample was determined, equal to 0.0001mL / mL of

sample, which can be entered using the preferred concentration / volume combination. So, it can be concluded that the optimized method has a good applicability for hexavalent Chromium detection, at least in the considered concentration range.

3.1.2 Zinc detection methods

As described in previous paper [27], the same optimization method, described in the previous paragraph for Cr(VI), was implemented for Zinc detection.

Zinc and its compounds are widely used in the metallurgical industry and in the production of some scale tank components such as batteries and paints, and in the food and cosmetics industries. Its spread in marine ecosystems can be highly impacting, as it can accumulate in living organisms and end up in the food chain.[41]. Zinc is also listed as a priority substance under the Helsinki convention which protects the marine environment of the Baltic sea. [42]. Therefore, its determination in aqueous ecosystems, and in particular marine ecosystems, is highly interesting, mainly considering the toxicity of this metal and its high scientific and environmental interest. [15], [34], [43], [44].

Italian legal limits[11] are reported in Table 9

Table 9. Italian legal limits for Zinc level.

Intended use	Max concentration (mg/L)	Legislative decree	Year
Human consumption	3	D.P.R. 24 maggio 1988, n. 236 Dlgs 31/2001	1988 2001
Zootechnic	3	DM 471/99 n.293	1999
Aquaculture	2	D.L. 27 gennaio 1992, n. 131	1992
Extensive aquaculture in marine	0,05	*	2011

[11] All the parameters shown in the table have been validated in the document «Metodi di analisi delle acque per uso agricolo e zootecnico» stilato dal MINISTERO DELLE POLITICHE AGRICOLE E FORESTALI, 2011.

http://www.reterurale.it/flex/cm/pages/ServeAttachment.php/L/IT/D/3%252F8%252F9%252FD.e304126511ca17040f18/P/BLOB%3AID%3D1034/E/pdf

In literature, the most common method for colorimetric determination of Zinc (II) is based on the use of Zincon(2-carboxy-2′-hydroxy-5′-sulfoformazylbenzene) as reagent for Zinc ions complexation.[44], [45]. This method is applicable in concentration range [0.5 – 8] ppm. [46]

The method for Zinc detection foresees the formation of a blue compound, thanks to the Zinc - Zincon bond, in a basic environment (pH = 9), according to the reaction shown by the Figure 25. The analysis is based on the formation, in buffered conditions (pH 9), of the hydro-soluble complex Zn/Zincon with a clear blue color and a peak of absorbance at 618 nm. The stoichiometric ratio of chelation is 1:1; one molecule of Zincon can complex a Zn++ ion according the scheme reported in Figure 25.

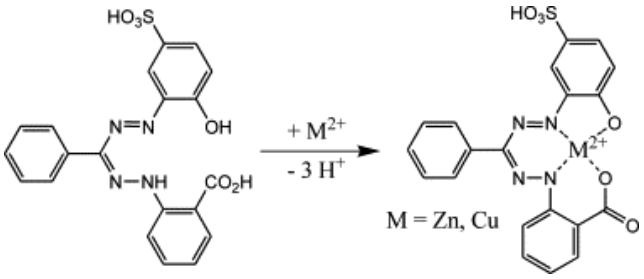


Figure 25. Zincon chemical structure in its free form (left) and metal-chelated (right).

The compound obtained from the Zn / Zincon reaction has an absorption peak at 618 nm[47]. The standard method is explained by Figure 26

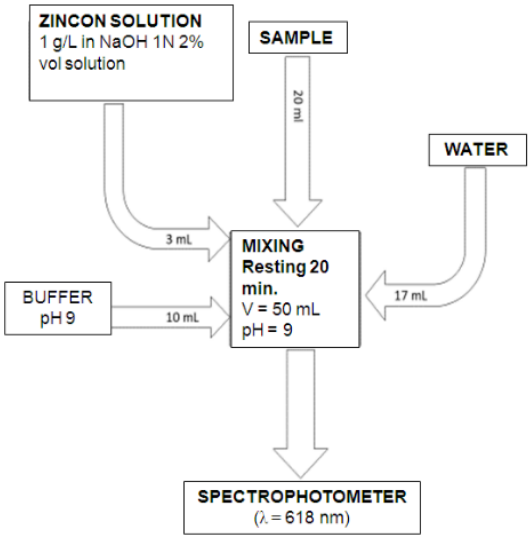


Figure 26. Standard method for Zn(II) detection using Zincon as reagent.

The standard method was validated, linearity was checked. Analytical grade general reagents were used. The concentration range of solution checked was [0.5 – 10] ppm, starting from Zn(II) standard 1000 ppm (Sigma Aldrich). Zincon solution was obtained mixing Zincon (Sigma Aldrich) in 2%vol solution of NaOH 1N (pH 12-13) in water. As buffer, boric acid and potassium chloride (Sigma Aldrich) were used. Moreover, a buffer solution (pH 8.5) was prepared using 4 g NaOH and 6.2 g H_3BO_3 (Sigma Aldrich). PH was controlled by a pH meter (Hanna Instrument). The complexed solution was analysed using a 6850 UV/Vis spectrophotometer(Janway).

As for Cr(VI), the method was optimized for automatic portable monitoring on-site.

The wavelength chosen for laser diode were 600, 618, 620, 625, 637, 638 nm. Table 10 shows the calibration curves and their linearity.

Table 10. Calibration curves at different wavelength. Ref. 618 nm

λ	Linear equation	R^2
Peak 618	y=0.1621x + 0.0289	0.9992
600	y=0.16x + 0.331 y=0.172x	0.9992
620	y=0.172x + 0.272	0.9998
625	y=0.124x + 0.0284	0.9991
637	y=0.1099x + 0.0257	0.9991
638	y=0.1083x + 0.0253	0.9991

As regards the rinsing study, solutions with different concentrations were used for consecutive measurements. The methods of rinsing investigated and compared were a double rinsing with distilled water and drying (standard method), single rinsing with distilled water without drying, rinsing with the next sample. Absorbance was then evaluated after the different methods. The errors are 15% without rinsing, 2% with distilled water, and 1% by rinsing with following sample.

The possibility to replace am high concentrated sodium hydroxide solution was investigated, because of its chemical aggressiveness.

However, alkalization is mandatory, since the color change of the solution does not take place in a neutral or acid environment (Figure 27)

Figure 27. Solution containing 1 ppm of Zn(II) complexed with Zincon. On the left the reaction occurs in buffered condition (pH 9). On the right a basic condition.

However, a less aggressive buffer solution was prepared and tested (pH 8.5). Results are shown in Figure 28

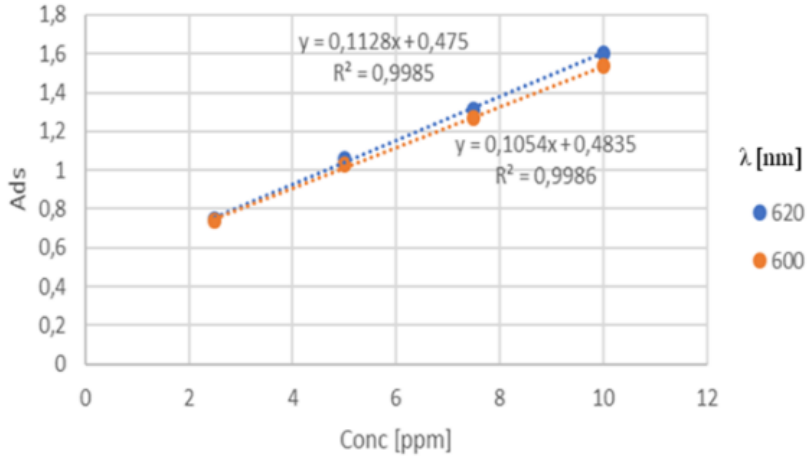


Figure 28. Calibration curves using self-prepared buffer solution.

Moreover, time of reaction, mixing phase, aging , dose tolerances and stability were tested. Figure 29 shows the obtained optimized method.

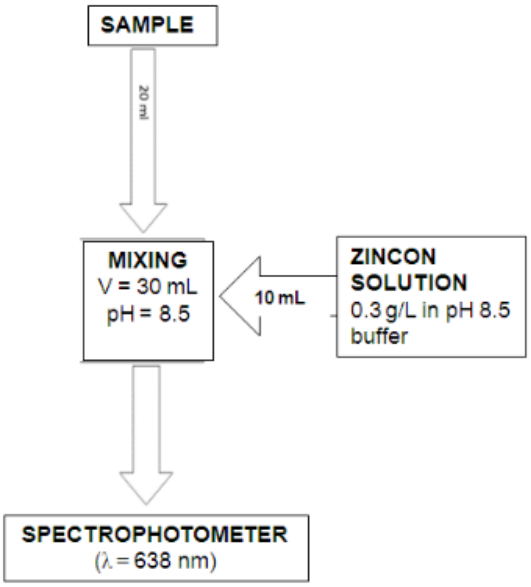


Figure 29. Final scheme of the improved method for portable devices.

3.1.3 Other heavy metals detection methods

In addition to hexavalent chromium and Zinc (II), detection of copper and nickel was investigated. The standard method were tested[40]. The aim of these tests was to identify recipes in order to use Zincon as reagent, like Zn (II) recipe. The use of a single reagent, the Zincon, for three different metals, would greatly help the implementation within a tool holder and automatic, storage, and cost saving. For copper, the optimized recipe was 20mL sample + 10mL of commercial pH4 buffer + 3mL of Zincon. Figure 30. The peak of absorbance spectrum was at 600 nm, the linearity was ensured in the range [0.1 – 10] ppm.

Figure 30. Complexed samples at pH 4, From the right 0.1 – 0.25 – 0.5 – 1 – 2 ppm

Curves of calibrations are reported for high concentrations Figure 31 and for low concentration Figure 32.

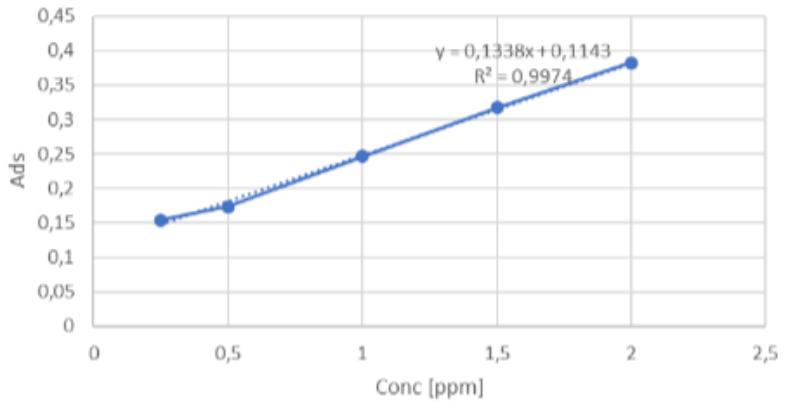


Figure 31. Curve of calibration for copper in the range 0.25 - 2 ppm. Absorbance at 600 nm.

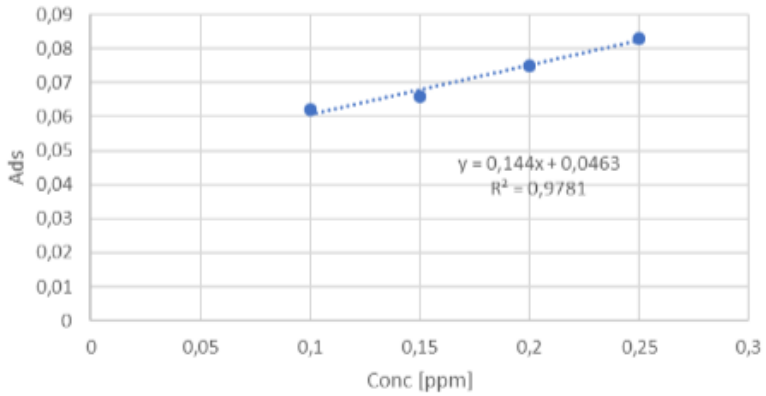


Figure 32. Curve of calibration for copper in the range 0.10 - 0.25 ppm. Absorbance at 600 nm.

For nickel, the optimized recipes were:

- Recipe 1: 20ml sample + 3ml Zincon + 7ml buffer solution
- Recipe 2: 5ml sample + 5ml Zincon B

The Zincon B solution was obtained adding 0.025 g of Zincon to 25 ml of buffer solution (pH 9). The solution was diluted to reach 100 ml with distilled water.

The absorbance peak was at 665 nm, and the linearity was tested from 0.1 ppm to 10 ppm. Figure 33.

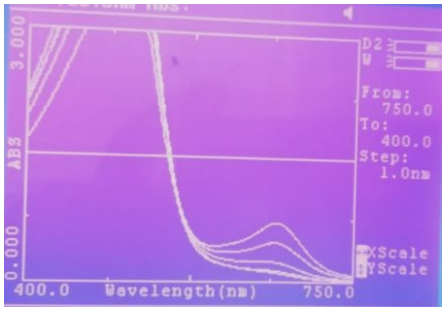


Figure 33. Absorption spectra of sample with nickel and complexed by Zincon at pH 9. From the bottom 1 - 2.5 - 5 - 10 ppm.

Results are reported in Figure 34 and Figure 35

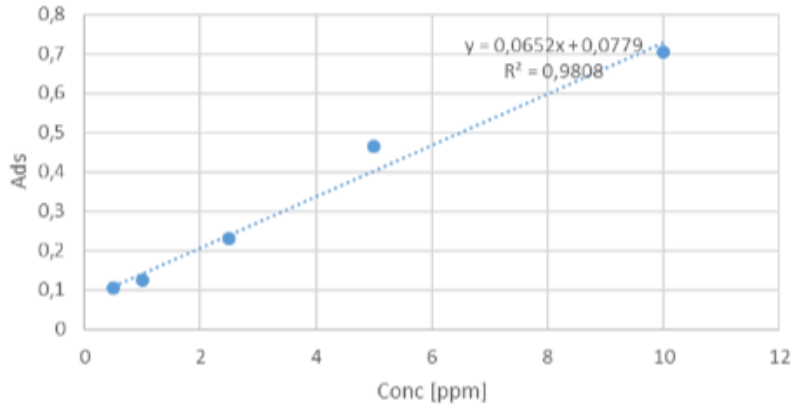


Figure 34. Curve of calibration obtained with recipe 1. Absorbance at 665 nm. This recipe has a low linearity, especially in low concentration range.

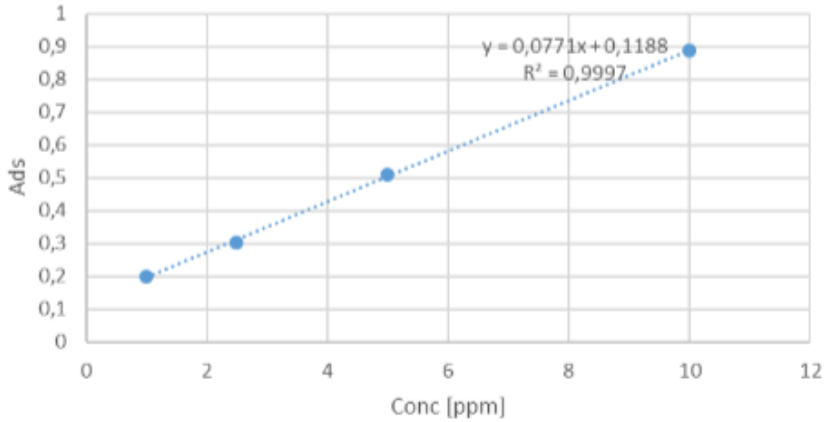


Figure 35. Curve of calibration obtained with recipe 2. Absorbance at 665 nm. Good linearity at low concentration range too.

3.2 Lab-on-chip design

The optimization of chemical methods involves a technological optimization, in order to transfer all the components of the measurement from the laboratory within a device capable of acting autonomously, that is without the intervention of the operator, having a good operating autonomy, in terms of battery life and long periods without manutention, and in order to be installed on-site.

The phases optimized were:

- Measure phase, in order to replace a spectrophotometer, which requires high space of installation, high costs, high energy consumption, operator;
- Sample preparation, in order to replace operator's manually steps;
- Data reading, collection and interpretation.

3.2.1 Optical design

The measurement phase, as described above, envisages exploiting the Lambert-Beer law, according to which the absorption of a substance is directly proportional to its concentration.

An optical design phase is therefore necessary, which involves the correct choice of the light source, the sizing of the optical system for the treatment of the laser beam and its addressing through the sample to be analyzed, the choice of the appropriate detector capable of capturing the absorbance training, the sizing of a system for conditioning the signal coming from the detector and driving the laser diode(Figure 36).

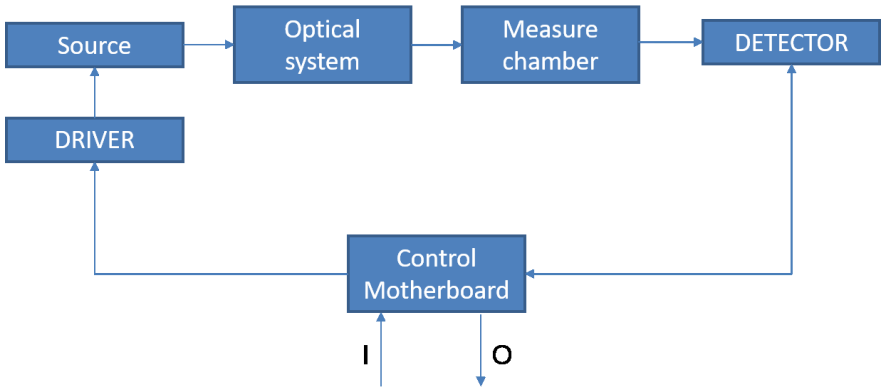


Figure 36. Logical diagram of measure phase and measure components.

The intent is to replace the current measurement method, which uses a wide spectrometer source capable of analyzing the absorption of the substance in the visible and ultraviolet range. Of all the spectrum, however, the measurement methods plan to exploit exclusively the absorption peak which corresponds to a wavelength, or in any case to a very narrow band. The idea is therefore to use a monochromatic light, corresponding to the wavelength of the absorption peak foreseen by a method, to replace the common sources. Moreover, it is decided to use a laser diode (Figure 37), as the lasers, besides being monochromatic, are coherent and therefore allow to have a greater optical power for the same power consumption, or better a lower consumption of electric power for the same power optics. The absorbance was read by a photodiode (Figure 38).

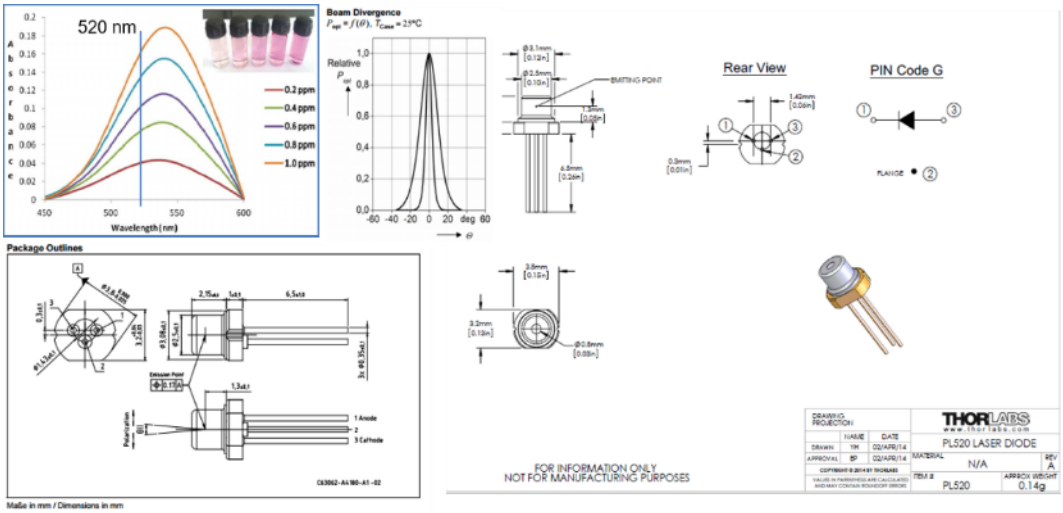


Figure 37. Example of laser diode chosen for metal detection. For hexavalent chromium detection, a wavelength was chosen (520 nm), and corresponding laser diode was identified (L520P50).

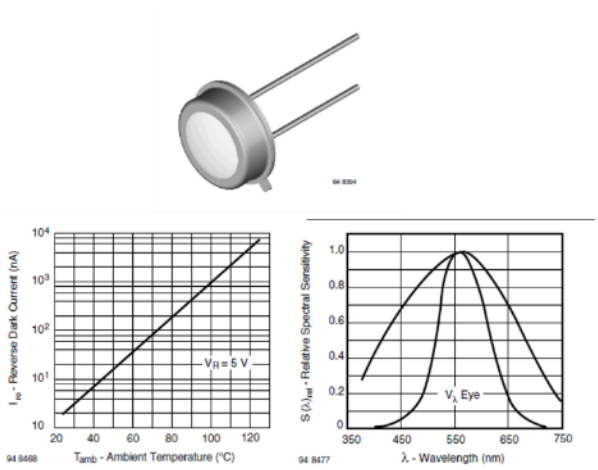


Figure 38. Example of detector chosen for absorbance quantification. For hexavalent chromium, a silicon photodiode BPW21R was chosen.

The optical path of the laser beam, through the sample, to the detector was studied by a raytracer software (Zemax). With this software, beam parameters can be simulated, and an optical system can be designed Figure 39 and Figure 40.

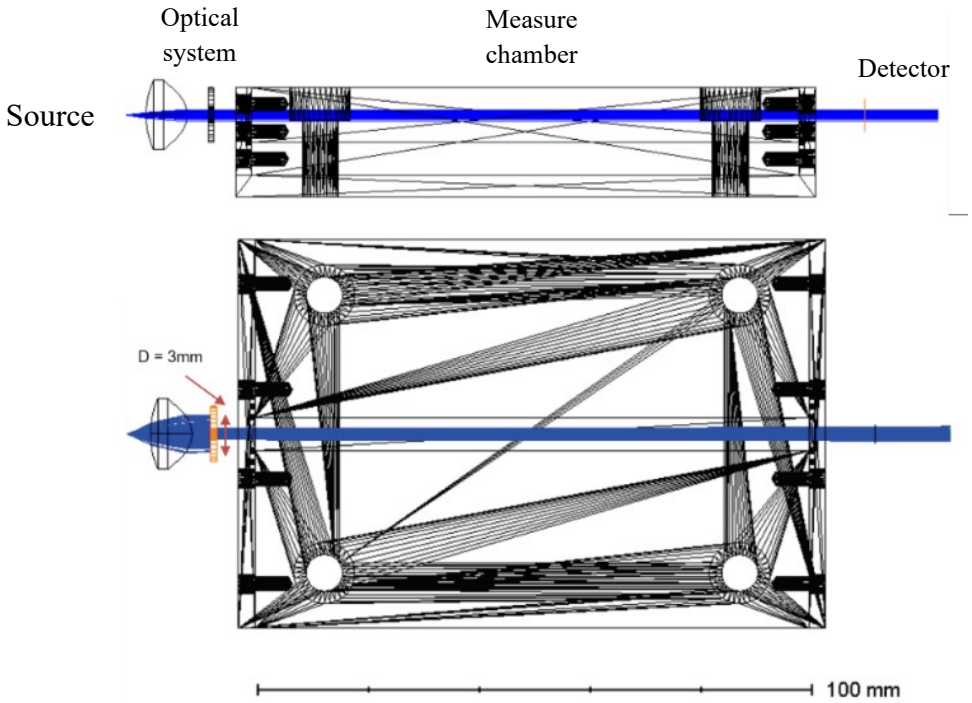


Figure 39. Optical simulation of laser beam, optical system, measure chamber and detector in Zemax

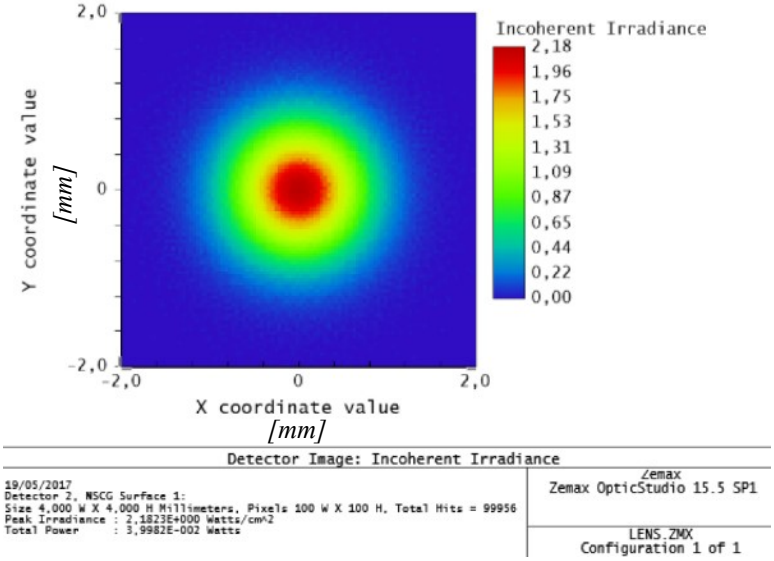


Figure 40. Power density distribution in the inlet of photodetector, simulated by Zemax

Mechanical components of the measure chamber were designed by a cad software (Solidworks), in order to identify dimensions, materials, connections and to realize mechanical drawings for mechanical workshop (Figure 41).

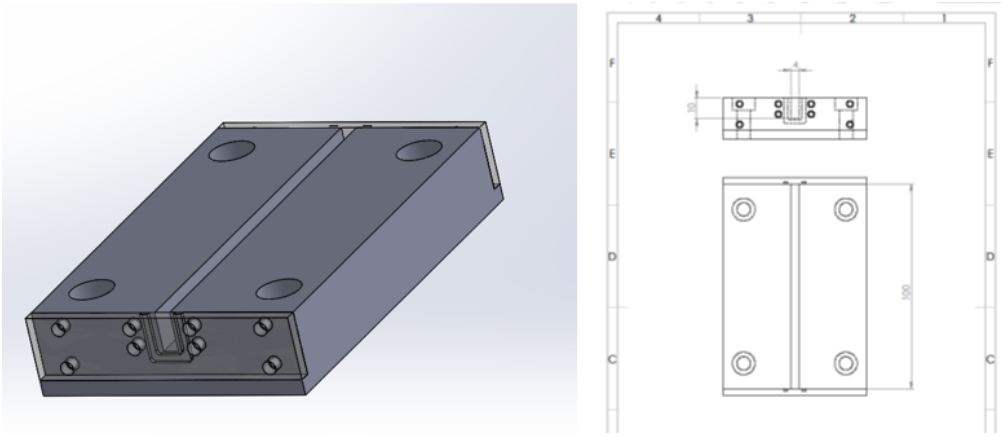


Figure 41. Measure chamber support was designed by a CAD software (SolidWorks). Dimensions in millimetres.

3.2.2 Hydraulic design

Hydraulic design was performed by an opensource software (Proficad). The optimized chemical methods were studied, the measurement parameters and

processes were converted to obtain automatic measure phases. These phases corresponding to a specific part pf the hydraulic circuit (Figure 42).

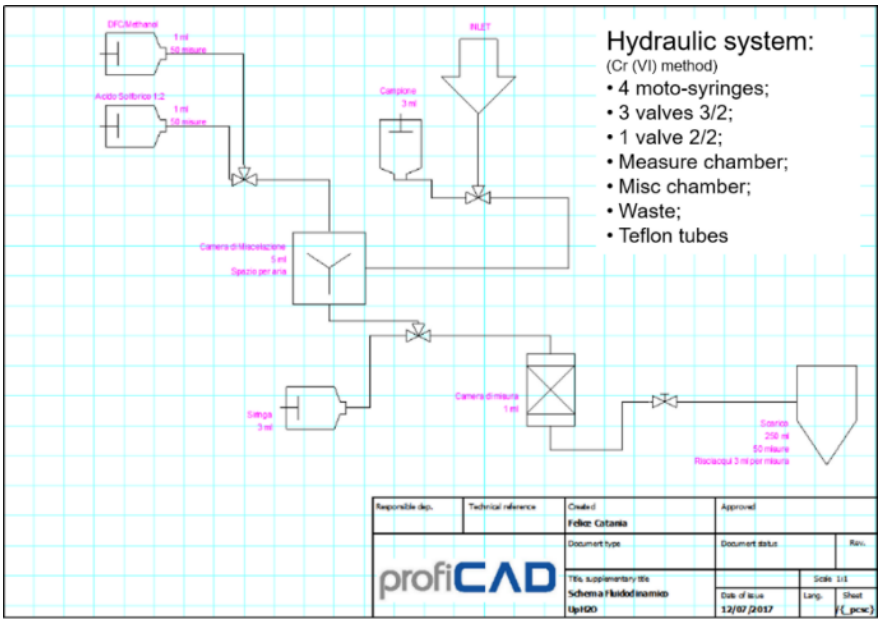


Figure 42. Hydraulic circuit designed for an automatic sampler and automatic measurement device. The components are microfluidic, in order to reduce dead volumes and contamination.

In order to minimize chemical contamination, dead volumes, and quantities of reagents and samples used, a microfluidic circuit was designed. Microfluidic components (Figure 43) were chosen, like microfluidic valves (Burket), microfluidic teflon tubes (Drawing Microfluidics), with an internal diameter of 1/32 inch and an external diameter of 1/16 inch. For reagents and sampler, glass syringes were chosen (SGH).

Figure 43. Examples of inert microfluidic components, chosen for microfluidic circuit, and therefore suitable for use with chemical compounds and minimizing dead volumes. On the left a microvalve, on the right a glass syringe .

Mixing tests were performed, in order to evaluate the feasibility of the mixing process in the syringe connected to the measurement chamber, an experimental set-up was made, right in the branch of the simulated circuit. From the experimental tests it emerged that with three cycles it is possible to obtain a mixing like that obtained in the laboratory (Figure 44). The Bubble Trap, fluidic components which, thanks to a special Teflon membrane, ensure degassing of a fluidic sample, has been tested and it was found that this element works in better conditions if the speed of the plunger is very low and, at the same time, there is a pump for vacuum. Motion tests were carried with components made of PMMA.

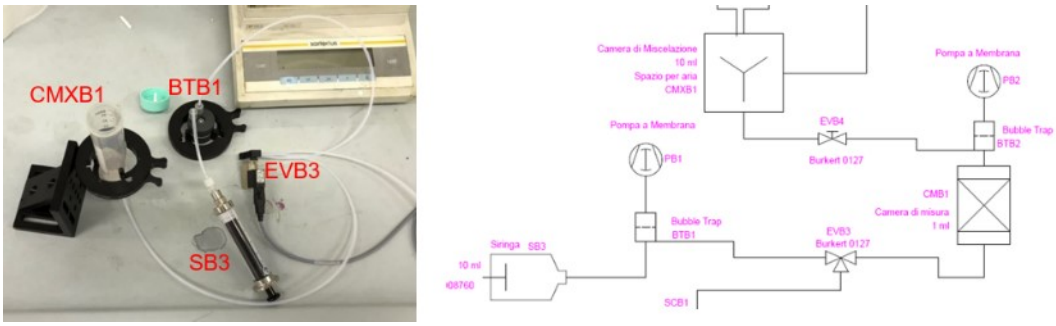


Figure 44. Setup for mixing tests. On the right the microfluidic model for mixing phase. On the left bench setup with corresponding components.

3.3 Bench prototype

. The process parameters were optimized at laboratory scale and then were tested on a bench prototype. It was composed of an automatic water sampler and an optical analyser Figure 45. As case of study, hexavalent chromium detection setup is described.

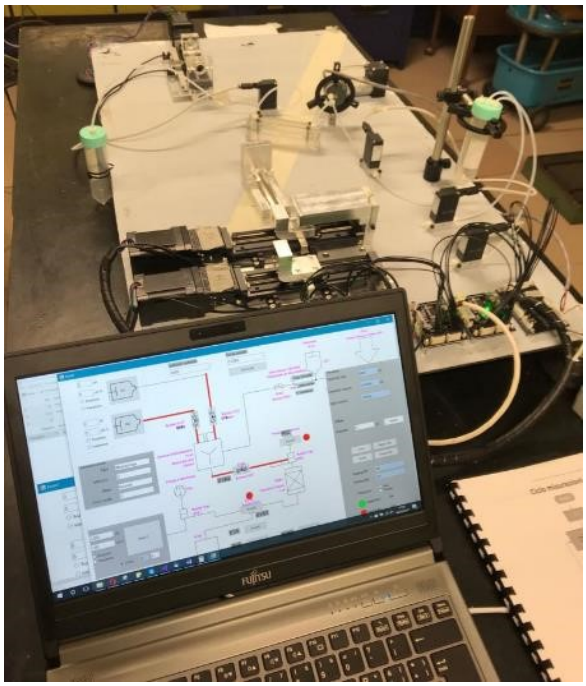


Figure 45. Bench prototype for early tests.

The optical analyzer was composed of a L520P50 laser source @520 nm (Thorlabs), collimated with an ACL 12708U aspherical lens (Thorlabs), a multichambered cuvette designed and built at Politecnico di Torino and a silicon photodiode BPW21R as detector (Vishay) (Figure 46).

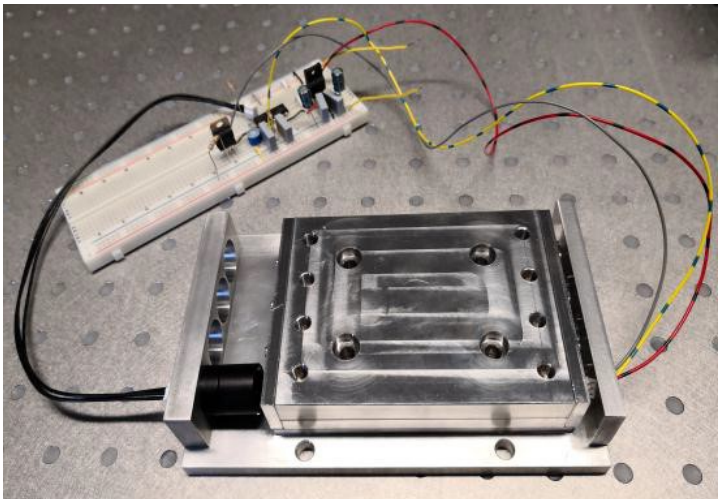


Figure 46. Optical analyzer. From the left light source, optical system, measure chamber, detector. An electronic driver for laser diode and photodetector was made on a board.

In order to control the automatic sampler, made by moto-syringes and the optical analyser, a software (APPENDIX) was made on Visual Studio (Microsoft), In collaboration with Microla Optoelectronics, features an intuitive graphic control interface (Figure 47).

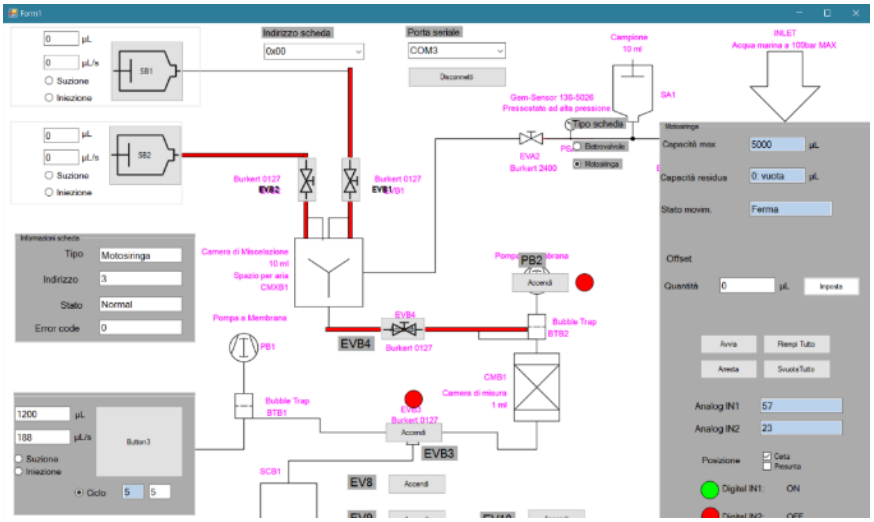


Figure 47. Graphic interface of software made to control bench prototype.

3.4 Early tests

Early tests aimed to confirm the possibility to replace standard laboratory methods and devices, with an automatic device capable of sampling, prepare the sample, measure and flush automatically. These tests were performed in Microla Optoelectronics.

First, the optical analyser was tested .

Starting materials:

- Cr VI standard solution in water (ST147), concentration: 1000mg / L Diphenylcarbazide (Sigma Aldrich,)
- Acetone (Sigma Aldrich) Sulfuric acid 98% (Sigma Aldrich)
- PL520 laser diode @ 520nm max output power 50 mW (Thorlabs)
- Silicon Photodiode BPW21R (Vishay)
- Digital multimeter 24V, resolution ± 0.001 mA (Microla)
- Power supply (Microla)
- Laser driver (Microla)

According to Lambert-Beer law, the absorbance of the samples is evaluated as Logarithm of the ratio between the value read by the photodiode using a blank sample, and the value read by the photodiode with the presence of Cr(VI) in the sample. Figure 48 shows the calibration curve obtained. Finally, a sample with a Cr(VI) concentration of 0.03 ppm was prepared and the absorbance was measured, and the measured concentration was graphically calculated.

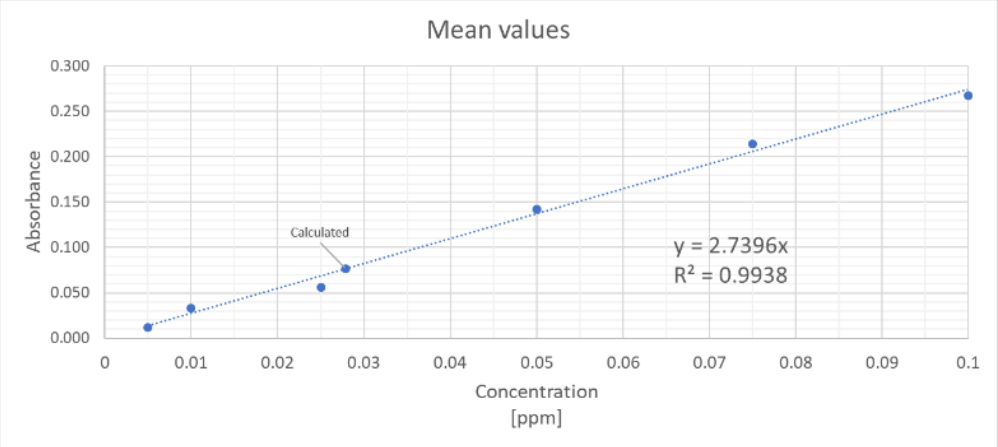


Figure 48. Calibration curve in early test for optical setup validation.

From the straight lines of trend, it can be deduced that there is a linearity between the concentration of the sample and the absorption of the laser @ 520nm read by photodiode. The linear coefficient of these line was very close to those evaluated with a spectrophotometer, the R^2 is higher than 0.990.

Table 11. Early test for optical setup. Absorbances are calculated against white value 3.163 mA.

Concentration prepared *[ppm]*	**Multimeter Values** *[mA]*	**Absorbance**	**Concentration calculated** *[ppm]*	**Error** *[ppm]*
0.1	1.710	0.267395	0.098	-0.002
0.075	1.930	0.214311	0.078	0.003
0.05	2.280	0.142107	0.052	0.002
0.025	2.784	0.055867	0.020	-0.005
0.01	2.920	0.03363	0.012	0.002
0.005	3.080	0.011911	0.004	-0.001
0.03	2.655	0.076346	0.028	-0.002

Table 11 shows the values read by the photodetector in correspondence of the prepared concentrations, the calculated absorbance value and the corresponding concentration value calculated following the calibration curve.

The last column shows the error made by the calibration curve with respect to the real values. It should be kept in mind that known concentrations have an error due to the successive dilutions with which they have been prepared.

In order to improve the precision and reliability, some mechanical modifications were implemented. In particular, an improved alignment support were manufactured and a thermal control was introduced. Further tests were performed, considering a large range of concentrations, from 0.005 ppm to 1 ppm. Mean values are reported in Table 12. A mean calibration curve was calculated. (Figure 49).

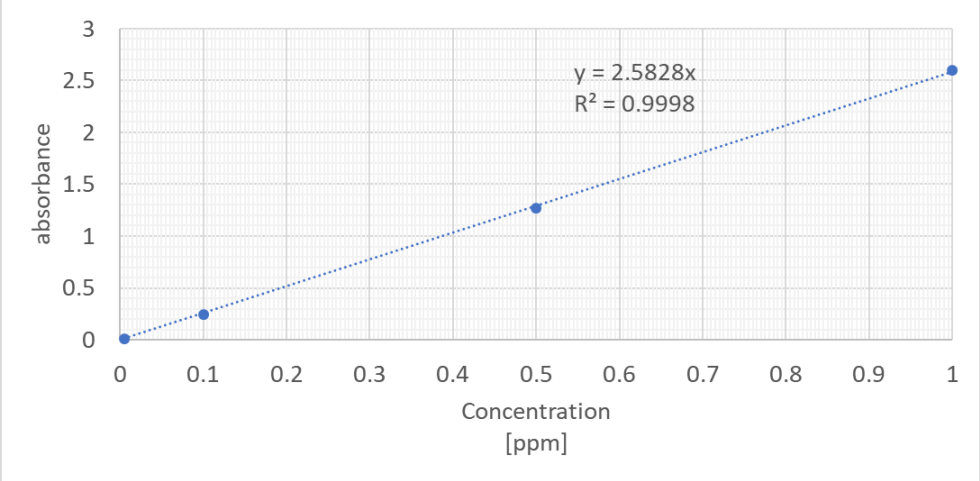


Figure 49. Calibration curve in further test. Range Cr(VI) [0.005 - 1] ppm

Table 12. Mean values of further tests performed with improved setup.

Concentration prepared *[ppm]*	**Multimeter Values** *[mA]*	**Absorbance**	**Concentration calculated** *[ppm]*	**Error** *[ppm]*
1	0.006	2.59614	1.005	0.005
0.5	0.128	1.26708	0.491	-0.009
0.1	1.343	0.24621	0.095	-0.005
0.005	2.310	0.01068	0.004	-0.001

Chapter 4

Realization and characterization of a miniaturized laboratory

4.1 Context and motivation

December 18, 2017, on the event “Energetic Activities. Security as a hub for technological innovation "organized by DGS-UNMIG[12], in collaboration with CRIET and RSE, presented the logo of the Network for offshore safety: CLYPEA [48].

Within this Network, the DGS-UNMIG[12], in collaboration with the SEADOG laboratory of Politecnico di Torino, is working on the development of a technology for environmental monitoring in the area adjacent to the off-shore hydrocarbon production platforms. As part of this project, Politecnico created an instrument capable of performing chemical analyzes on small samples of water returning any anomalies with respect to the current legislation found in the concentration of heavy materials. This instrument was called UPH$_2$O.

The UPH$_2$O sensor platform, in fact, provides the possibility of sampling with cadence at least daily (or more frequent depending on the risk analysis) water in marine environments, activating an ON / OFF procedure on the presence or absence of contaminants and preparing the collection of samples for a more accurate and quantitative analysis on the ground. this platform uses Lab-On a-Chip microfluidic

[12] Direzione Generale per la sicurezza anche ambientale delle attività minerarie ed energetiche del Ministero dello Sviluppo Economico https://unmig.mise.gov.it/index.php/it/

technologies for fluid and flow management in situ analysis of the samples. The requirements relating to sensors must include the collection and analysis of samples in situ of water and for this purpose it is housed in autonomous submarine vehicles (hereinafter Autonomous Underwater Vehicle - AUV). The expected advantages for this technological application not yet present in the state of the art consist of the possibility of:

- long-term monitoring, with multiple parameters detected simultaneously and characterized by the fact of being "where and when not otherwise possible";
- higher sampling / analysis frequency;
- wider spatial coverage around the platform;
- low cost and high cost / benefit ratio platform.

In order to verify the potential of this approach, a "Case study" was taken as first step, consisting of an implementation project for customizing the platform to detect heavy metal ions (Cr, Cu, Zn, Ni) in seawater. The UPH_2O platform therefore provides for an autonomous submarine vehicle (AUV) and a module (PAYLOAD) interfaced with the vehicle in which the measurement systems are housed (Figure 50). They provide for the collection and analysis of sea water samples autonomously.

Figure 50. Autonomous Underwater Vehicle equipped with a miniaturized laboratory, developed by Politecnico di Torino, for autonomous heavy metal detection in seawater near offshore hydrocarbon platforms.

4.2 Material and methods

4.2.1 Autonomous Underwater Vehicle

Nowadays, the use of drones in the marine environment is widespread. They are typically launched from surface vessels or fixed platforms (like oil companies) for the analysis of the surrounding marine environment. Because of their high energy autonomy, which goes from the few hours of the simplest models to days of more complex ones, they are typically used for a certain number of detections around the point from which they are launched and subsequently, due to the progressive discharging of the battery installed on board, they are recalled for loading on board on ships or on platforms.

The generic mission can therefore be divided into the following phases:

- Launching phase: in this phase the vehicle is lowered, manually or using winches in the case of large-scale topside ships, in the water surrounding the boat. The manual launch can be done from heights low enough to not damage the payload contained in the AUV;
- Starting and descending phase: at this stage, the vehicle's propulsion units are started to allow the downward descent. Some types of AUV, rather than traction adjustment, use the adjustment of the center of gravity to submerge. Moving the center of gravity forward creates a pitching moment that tends to bring the vehicle down so that, subsequently, the thrusters lead him to the required depth. Once reached the operational depth, the command is reversed to bring the motion back to horizontal;
- Operative phase: in this phase the vehicle carries out the mission for which it was built using the payload mounted on it. This phase involves the largest battery discharge cycle, forcing the vehicle to get back on board the boat to be reloaded;
- Ascent phase: at this stage the vehicle returns near the boat and then performs the descending phase in reverse. For positive vehicles[13] and when the marine environment is rather calm, as it could be in a lake, you can even think about turning off the engine and waiting for the vehicle re-emerges autonomously;

[13] A vehicle is positive if it tends to float naturally, that is if the sum of the forces in water, hydrostatic thrust and weight force, is directed upwards.

- Recharge phase: in this phase the vehicle is loaded on board of the host platform or vessel for a full charge of the battery, in order to return to operation.

There are several kinds of missions that can be carried out with an underwater drone. They range from simple filming submarines of an amateur to the complex bottom analysis carried out to check the integrity of the same [49]. Moreover, there are several kinds of AUV to those missions:

- AUV for civil activities, which include recreational purposes (free exploration), seabed surveys identification of flocks to be fished. The drones belonging to this category have in common a low level of hourly autonomy, which is around for many around the hours;
- AUV for environmental purposes, which include study of volcanic phenomena, study of marine hydrothermal currents, analysis of the leakage of chemical agents from the subsoil, mapping of marine fauna and flora, analysis of pollutants in water. The drones belonging to this category need to have greater hourly autonomy and a much heavier payload to properly carry out the proposed missions.
- AUV for defence and military purposes. They mainly have the task of performing reconnaissance and (when needed) attacking the enemy. It is intuitive to understand that they need of a wide hourly and mileage autonomy and that must be able to withstand heavy payloads due to the presence of an arsenal on board and countless sensors. It is good that they turn out invisible to SONARs, so as not to be detected and destroyed by the enemy.

Table 13 reports commercial AUVs, which are on the market, and some of their parameters.

Table 13. AUVs on the market, their parameters and applications.

Manufacturer	Commercial name	Length *[m]*	Diameter *[m]*	Max Depth *[m]*	Battery life *[h]*	Max Velocity *[m/s]*	Application
L3 Ocean-Server	Iver3 std	1.50	0.147	100	8 ÷ 14	2	Environmental
L3 Ocean-Server	Iver4 PW	2.50	0.230	300	-	2.5	Commercial Defence
EcaGroup	A27-E	4.50	0.730	300	36	3	Environmental
Kongsberg	Remus 100	1.70	0.190	100	8	2.6	All
Gabri SRL	Seastick 300	2.00	0.250	300	10	2	Environmental
Graal Tech	Folaga AUV	1.22	0.155	80	6	1	Research

For our purpose, the Gabri SRL AUV Seastick 300, was chosen (Figure 51). The vehicle used is a cylinder-like with slightly rounded heads both the parts, both posteriorly and anteriorly, which has two triple-bladed propellers on the right and left sides, two control surfaces in the front and four (each placed on one side) in the back. A cylinder coming out of the central body, which is useful to contain a GPS antenna. A hook is placed in the rear, useful for recovery in case of failure, and two hooks are placed in the central body, useful for lifting and moving it during ground operations. The vehicle is positive, it means that, as previous explained, in the event of a failure it can return to the surface autonomously. This expedient prevents to lose the vehicle into sea due to a malfunction.

Figure 51. Seastick 300 from Gabri SRL.

The drone has a double thruster configuration. This configuration makes it possible to reduce considerably the diameter, as shown in Figure 52, but the propulsive efficiency decreases compared to case with single propeller. Furthermore, the propellers could be damaged because they are no longer placed behind the body, which acts otherwise as protective element.

In addition to the system with which the mission is carried out, navigation systems are present on board:

- An inertial platform that allows tracking of both states of the vehicle and of its position: the GPS signal (global positioning system), in fact, it is not able to trace it correctly when it is immersed in water. This device is necessary to know AUV position. Inertial platform is able to provide a correct location of AUV by integrating the accelerations to which the vehicle is subject instant by instant;
- A DVL (doppler velocity logger): as stated by the manufacturer's website, DVL is responsible for measuring water velocity and serves to other tools to enable them to function properly;
- A high definition camera positioned in the front of the vehicle. It also has white LEDs facing the direction progress to control the surrounding area;
- A series of batteries designed to maintain operations during the mission.

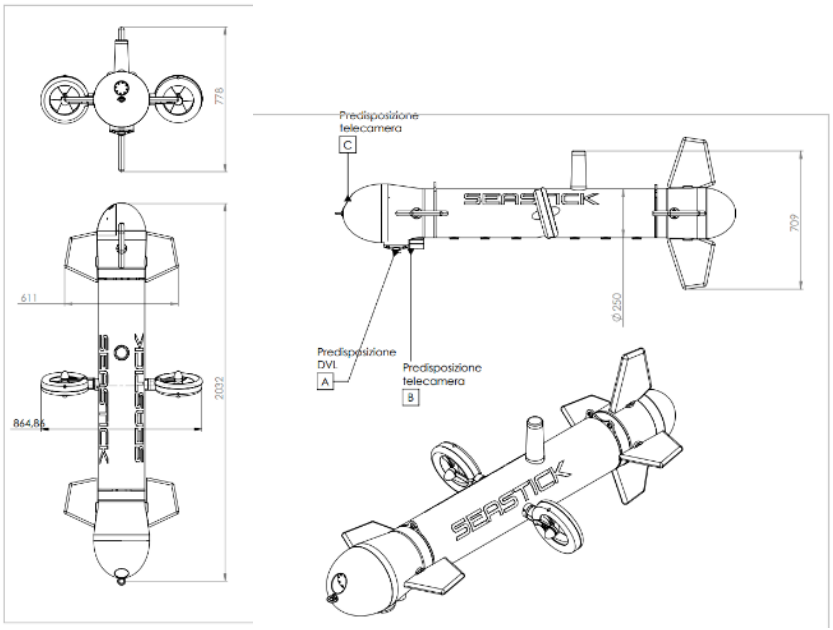


Figure 52. Engineering drawings of the submarine drone Seastick 300 by Gabri SRL.

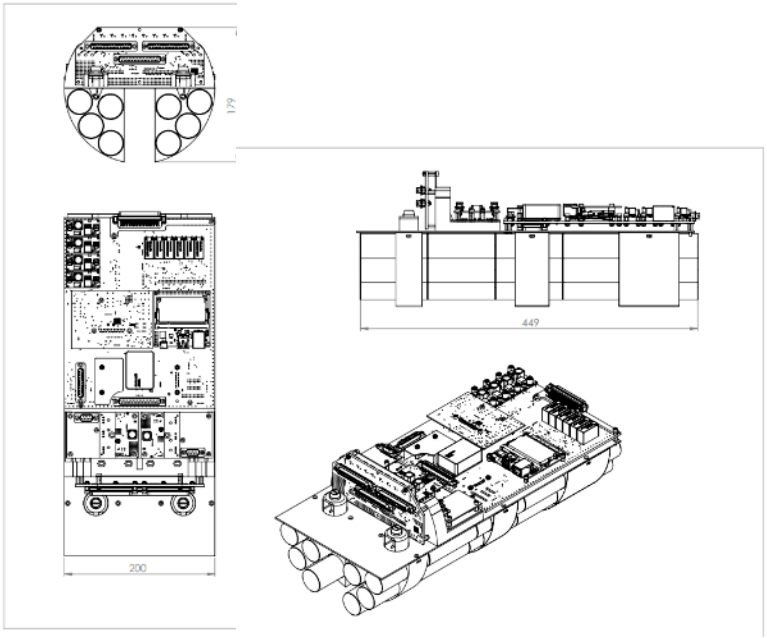


Figure 53. Engineering drawings of the stern drawer, including battery pack and electronical control devices by Gabri SRL.

All these elements are essential for carrying out the mission. Without even one of them it would be impossible to maintain contact with drone or track its position over time.

All the elements that make up the drone have been modelled using CAD software Solidworks^{TM} (Figures 54-57), and subsequently used for stability analysis during mission operations [50], a topic that will not be covered in this thesis.

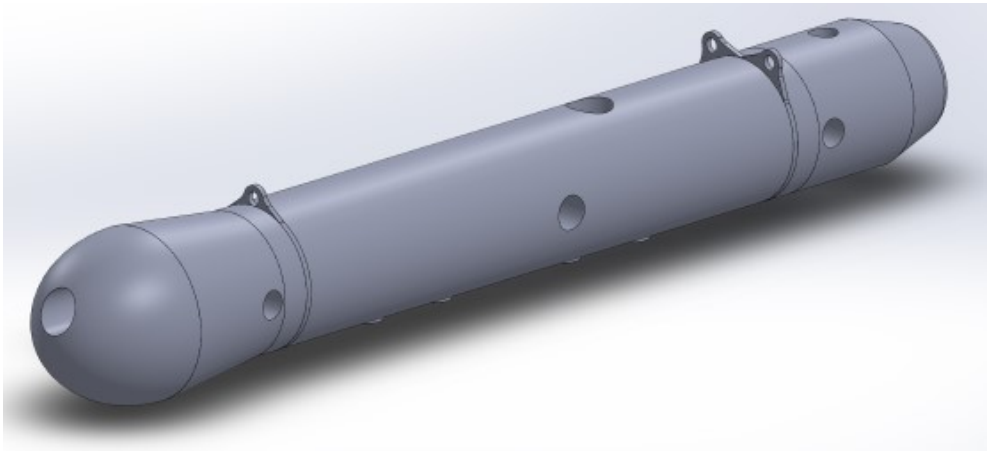


Figure 54. Main body of the AUV with frontal cone

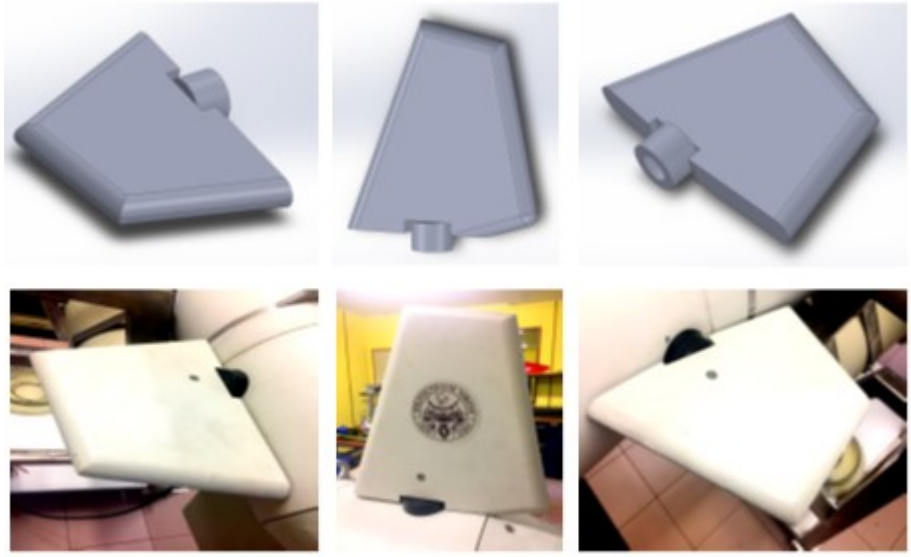


Figure 55. From the left: side wing, vertical rear wing, side front wing.

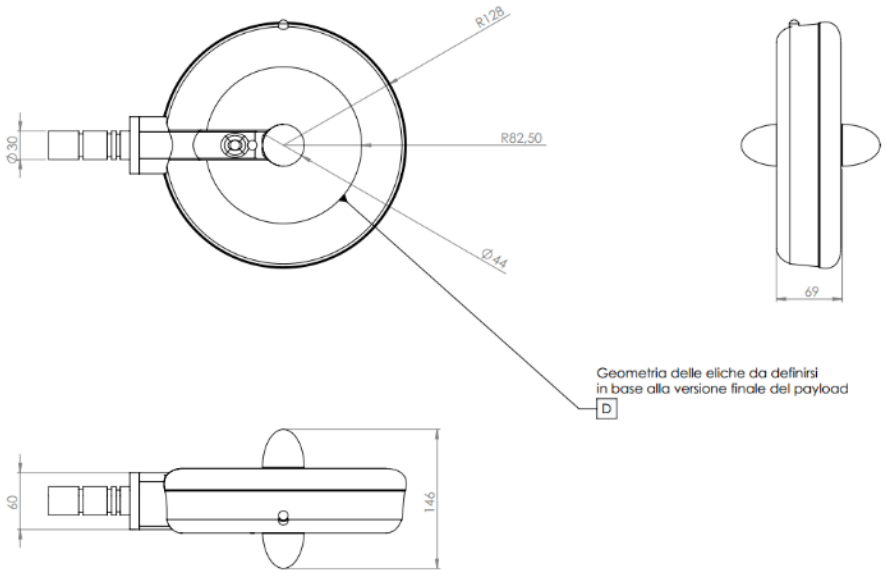


Figure 56. Engineering drawings of the propellers by Gabri SRL.

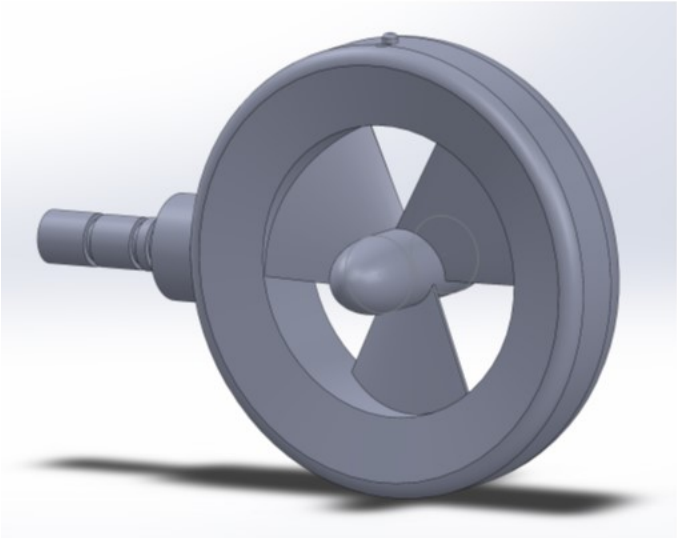


Figure 57. Model obtained by cad software of the propellers. Propellers are printed by an FDM 3D printer, in order to guarantee lightness. For mechanical resistance, a steel soul is built-in the helix.

4.2.2 Payload design and development

In order to host the miniaturized laboratory, the submarine drone requires a payload (Figure 58), which will be hooked into the underside of the aircraft through the appropriate sheet metal bars bent to support the weight of the payload.

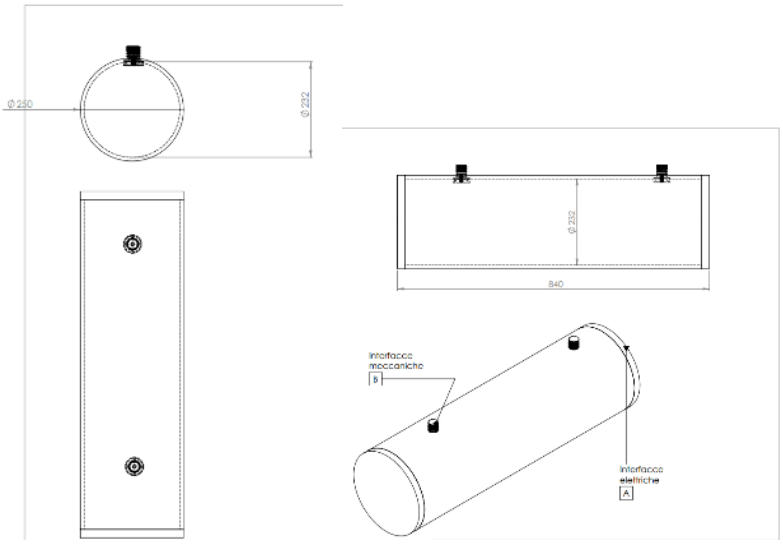


Figure 58. Engineering drawings of payload by Gabri SRL.

The payload casing is composed of a closed hollow cylinder with two caps at the ends, which fit perfectly into each other front and back holes. Although it is composed of several components, it has been realized for simplicity in a single body adding the holes for the positioning of the supports (Figure 59).

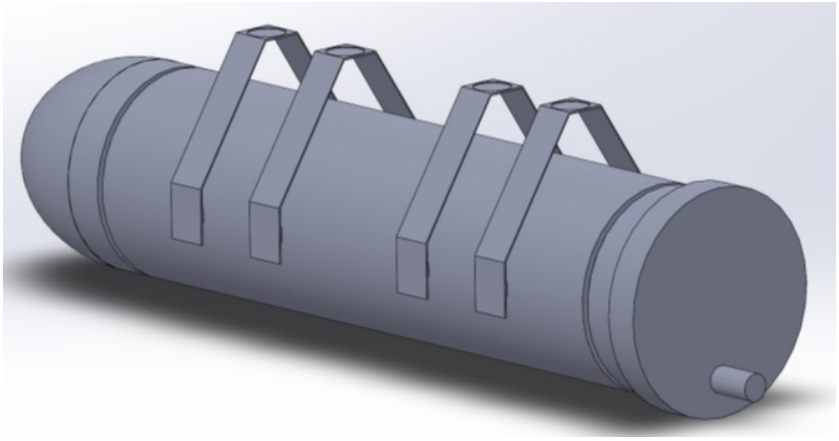


Figure 59. CAD model of the Payload with bars for anchorage.

The final assembly was made connecting the AUV model to the payload model through the bars, as shown by Figure 60.

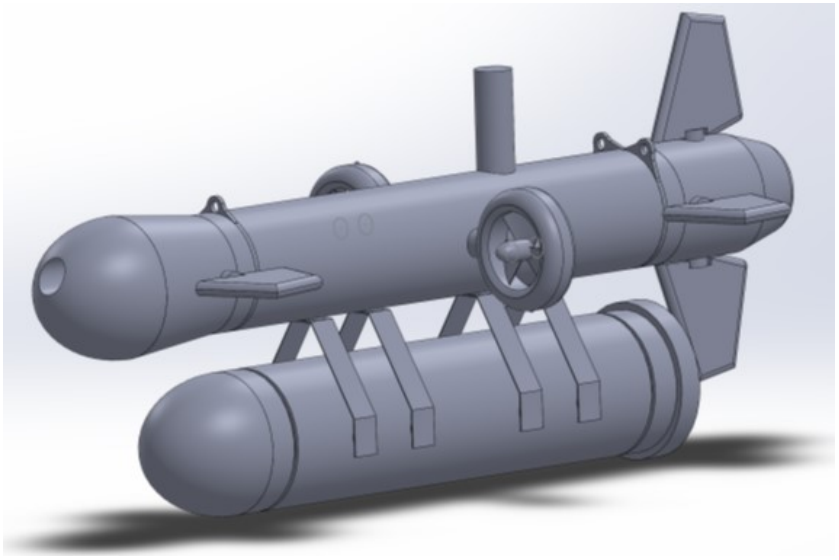


Figure 60. Final CAD model of the assembly AUV - Payload.

4.2.2.1 Hydraulic design

Within this payload, the miniaturized laboratory for dissolved metal detection must be arranged and designed in order to explain the monitoring mission, occupying as less space as possible, with as less weight as possible, sampling seawater at high depths and pressures[14], making the measurement for the recognition and quantification of the chosen dissolved metals, and storing the treated samples for an appropriate discharge at the end of the mission

In detail, following the notation represented in fluidic scheme shown in Figure 61, the mission operations are divided into several phases, corresponding to the parts of the hydraulic circuit:

- ***Suction and depressurization phase***: The sea water is initially filtered (via sintered filters) while passing outside within the payload: this allows to have spurious elements reduced to a minimum during sample analysis. From the main liquid inlet, or sea gate, the water passes to a second filter. Through a valve, called EVA1 (at high pressure, because it supports up to 50 bar), the water is placed in a 10 mL chamber, necessary sample for subsequent analyzes. Immediately afterwards there is a pressure switch that regulates the opening so that the second valve, EVA2, receives a maximum of 2 bar.;
- ***Rinsing phase:*** From here the water enters the mixing chamber, connected directly with EVB4 valve (low pressure), with the chamber measurement and with another valve (this, however, three-way) called EVB3. Of the three ways of the EVB3 valve, the first one is from which the sea water comes; the second is connected to the sampling syringe; The third is connected to the waste. Soon after having depressurized the water, the sampling syringe collects an amount of the sample which is used to perform a rinse. In this way it cleans all the tubes. Along the circuit, there are two bubble traps (one between EVB4 and the measurement chamber and one between EVB3 and the sampling syringe), consisting of a membrane, able to eliminate the bubbles that could form in the circuit.;
- ***Post-rinse discharge phase:*** Once rinsed, the EVB3 valve is switched to the side that leads to the waste, so that the rinse water, possibly

[14] At a depth of 300 m below sea level there is a pressure of 30 bar. The pressure increases by a rate of 0.1 bar per meter.

contaminated from polluting substances and reagents, it is not injected back into the sea. Subsequently rinsing, for reasons of environmental safety, is repeated a second time to further decrease the presence of elements spurious in the measurement chamber.;

- ***Blank sample measurement phase:*** After completing the double rinse, the dosage starts. The water arrives in the mixing chamber with all the valves closed, after which it is taken from the sampling syringe by opening the appropriate EVB4 and EVB3 valves (naturally after switching on of bubble traps). Once it enters the measurement chamber it is analyzed in order to obtain "blank sample". The measurement takes place through the use of laser diodes and photodiodes capable of read the current generated by it once it has crossed the chamber measure, one for each metallic substance to be identified. Water, since no further impurities are present, it is not discharged for further metal detection;
- ***Dosing, mixing and measurement phase:*** In this phase, the previous phase "blank sample measuring" is repeated with the difference that, in the mixing chamber, they come also injected the reagents through the two smallest volume syringes present upstream of the mixing chamber (in the low part pressure). Reagents are present for 100 measurements in the quantities of 10mL of methanol+DFC and 5mL of sulfuric acid. Mixing takes place through a repeated suction cycle, which occurs 5 times, so that the sample and reagents mix best of the ways. The measurement is carried out with the presence of a further control: if the measure of a metal is greater of the measurement of blank sample, the system continues to measure, in order to obtain an average value that makes it possible to avoid false positives;
- ***Post-measurement discharge phase:*** After the measuring phase, the valve EVB4 is closed and the valve EVB3 switch to side that leads to waste, where the complexed water residue is preserved. The Storage bag can hold 50 measurements.

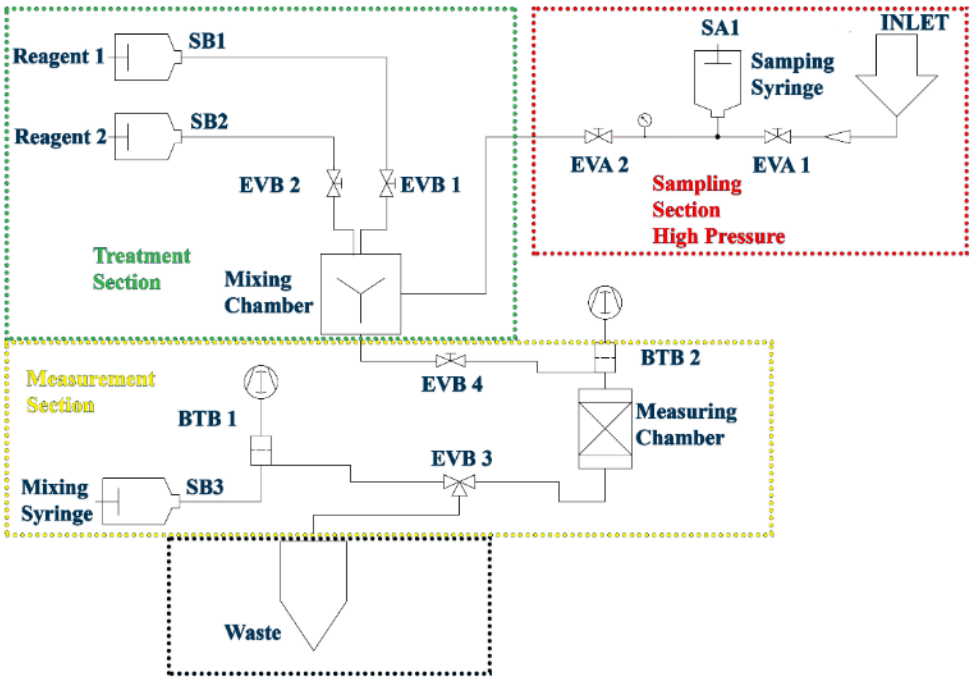


Figure 61. Fluidic scheme divided into functional sections. In the red rectangle the new high-pressure section, designed to reach 50 bars.

Compared to the previous test system (Figure 42), the fluid scheme of the system was modified, particularly in the sample branch it was decided to insert high-pressure elements to sample down to 300m. The new configuration is highlighted in red in Figure 61.

The main entrance is made up of plastic tube that, immersed in the sea, leads water from the outside to the inside of the payload via an inlet valve (Through a prefilter) able to withstand up to 100 MPa of inlet pressure (it is oversized in order to be able to operate even beyond the safety level of 300 m) and then drawn to reach the EVA1 valve (Burkert 2400) which supports from 0 Pa to 50 MPa. From this one enters the volume mixing chamber 10mL and on the other hand to the EVA2 valve, identical to EVA1(Figure 62). In the middle a pressure sensor is placed which, by measuring the pressure, is able to communicate to the system. So, depressurization to 2 bars occurred. The filter support (Figure 63) was designed and tested to support up to 100 bars.

Figure 62. From the left: Steel moto- syringe for sampling, Inlet port, EVA1 and EVA2 high pressure valves.

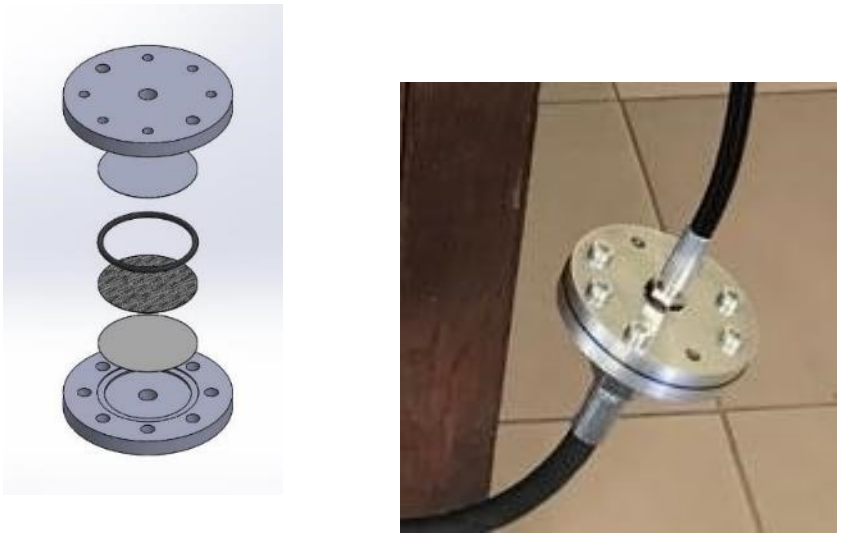


Figure 63. Filter support CAD design (left) and manufacturing (right).

Sampling and depressurization are operated by a syringe made of steel (Chemika) with a volume of 20 ml, a motorized axis (Misumi) and mechanical supports designed and manufactured by Cemas Electra srl (Figure 64).

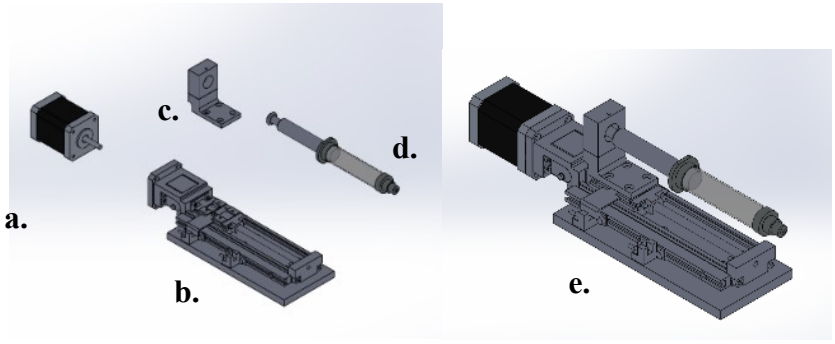


Figure 64. Moto-syringes developed for sampling, mixing, dosing and flushing. a. Stepper motor (Nema), b. actuator (Misumi), c. Mechanical support (Cemas), syringe, e. final assembly.

From EVA2 the circuit is connected directly to the mixing chamber. Once the seawater sample has arrived in the mixing chamber, there is a low pressure valve called EVB4 (Burkert 0127) able to withstand up to 2.5 bar. They are also present a 1 mL measurement chamber and a bubble trap capable of eliminating, via a pump a membrane, the bubbles present in the circuit (Figure 65).

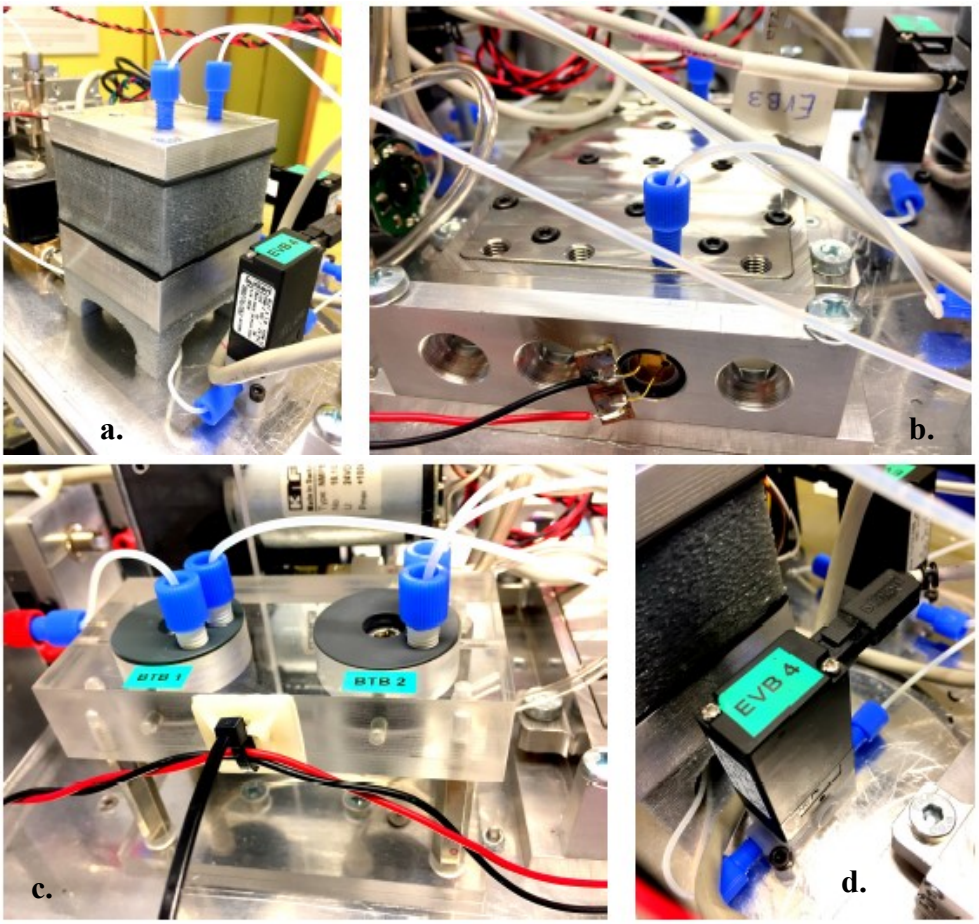


Figure 65. Fluidic elements which concur to dose, mixing and measurement phase. From the left a. Mixing chamber, b. Measurement chamber, c. Bubble traps, d. Low-pressure valve

To obtain a perfect mixing of the complexing agents and the sample volume, a mixing chamber was designed and built (Figure 66). The chamber has grafts on the upper cap for reagent and sample inlet, and its interior has a truncated-cone geometry to optimize mixing, rinsing, and minimize residual volume, which is almost zero after each measurement and rinsing, in order to minimize contamination of the next sample.

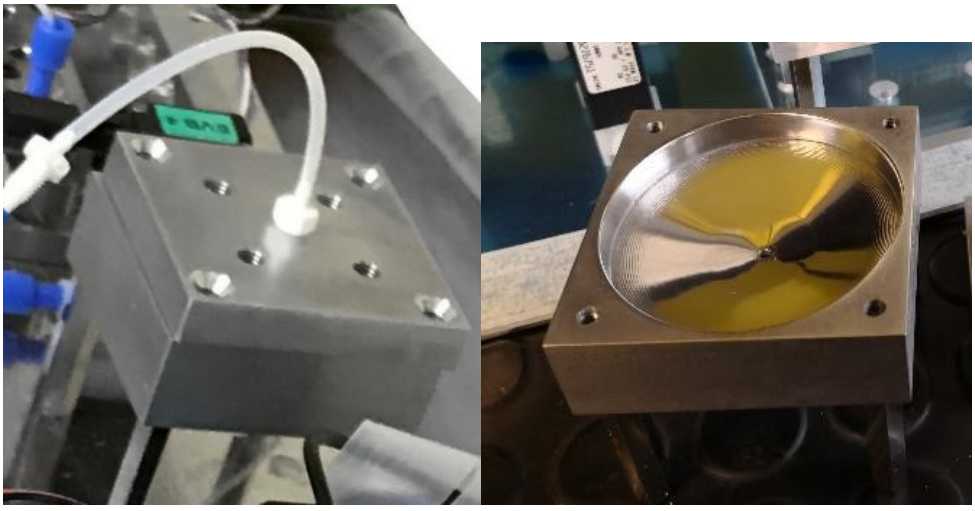
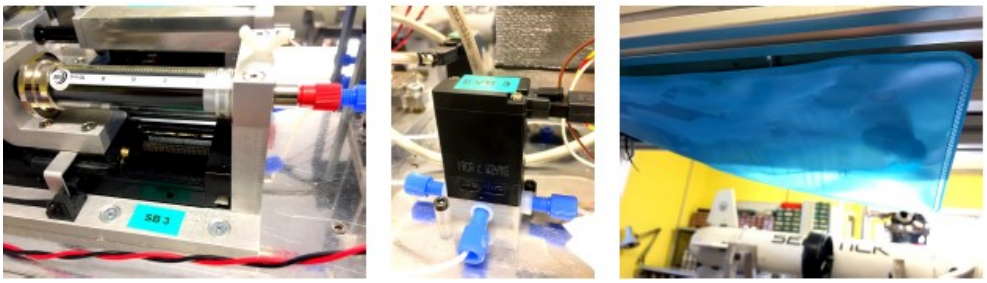


Figure 66. Mixing chamber manufactured.

The last part of the circuit consists of the syringe sampling rate, 10 mL, connected to the measurement chamber via a low pressure valve and three-way EVB3 (Burkert 0127), with a side connected in the third way to a deposit bag of waste of 250 mL (Figure 67).



a. **b.** **c.**

Figure 67. a. Mixing syringe, b. Low-pressure valve, c. waste

For the choice of syringes that will contain the reagents, glass syringes (borosilicate) with a PTFE plunger (compatible with many strong acids and organic solvents) were chosen.

In order to best size the system the following steps have been followed analysis of the syringe datasheet (volume, stroke, plunger diameter, maximum speed and maximum pressure), analysis of the stoichiometric ratios of reagents and sample, calculation of the number of theoretical samples based on the datasheet, calculation of the physical quantities of reagents (density, dynamic viscosity), pressure loss hypothesis located in the tube syringe interface and its calculation, choice of linear actuator (propeller pitch and starting torque), transformation of the loss of load into

the maximum torque allowed to the motor. Following tables show syringes design results.

Table 14. Number of samples considering 5 ml of sample + reagents

	Reagents	Number of samples
Syringe 0.5 ml	DFC + Methanol	10
	H2SO4+H2O	5
Syringe 1 ml	DFC + Methanol	20
	H2SO4+H2O	10
Syringe 2.5 ml	DFC + Methanol	50
	H2SO4+H2O	25
Syringe 5 ml	DFC + Methanol	100
	H2SO4+H2O	50
Syringe 10 ml	DFC + Methanol	200
	H2SO4+H2O	100

Table 15. Torque at maximum pressure of chosen actuator LX2001P-B1-T2042-150-OP2 (Misumi) a Helix step of 1 mm.

	Torque at p_{MAX} *[Ncm]*
Syringe 0.5 ml	4.809
Syringe 1 ml	9.619
Syringe 2.5 ml	20.453
Syringe 5 ml	38.498
Syringe 10 ml	38.515

Table 16. Torque at top speed

	Reagents	Torque at V_{MAX} *[Ncm]*
Syringe 0.5 ml	DFC + Methanol	1.200
	H2SO4+H2O	1.201
Syringe 1 ml	DFC + Methanol	1.200
	H2SO4+H2O	1.202
Syringe 2.5 ml	DFC + Methanol	1.203
	H2SO4+H2O	1.216
Syringe 5 ml	DFC + Methanol	1.210
	H2SO4+H2O	1.262
Syringe 10 ml	DFC + Methanol	1.242
	H2SO4+H2O	1.449

Once the syringes, 5 ml for sulphuric acid 1:30 in water and 10 ml for DFC+methanol, and the actuator, LX2001P-B1-T2042-150-OP2 (Misumi), have been chosen, the mechanical supports are designed and manufactured and the electronics for driving the stepper motors are designed (Figure 68). The syringe motorcycles made in this way are capable of dosing quantities of reagents in the order of tens of nanoliter.

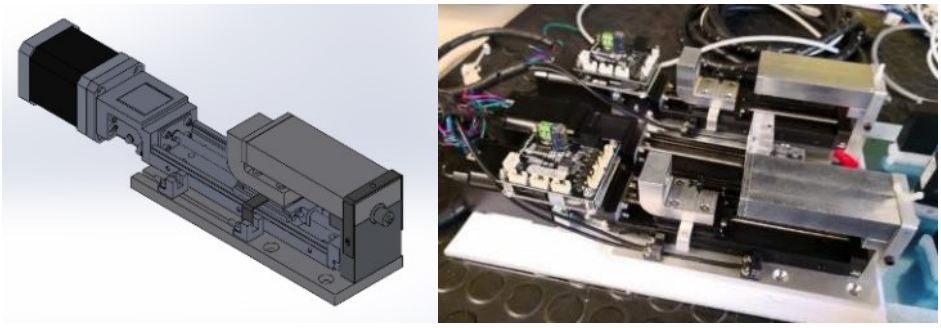


Figure 68. Final engineering CAD drawings for moto-syringes utilized to dose reagent (left) and realization (right)

The final layout of the hydraulic circuit and all the mechanical components for integration into the payload has been designed (Figure 69). All the components are housed in a 1000 mm long and 250 mm wide plate (Figure 70), inserted in the payload thanks to ad hoc guide bars (Figure 71).

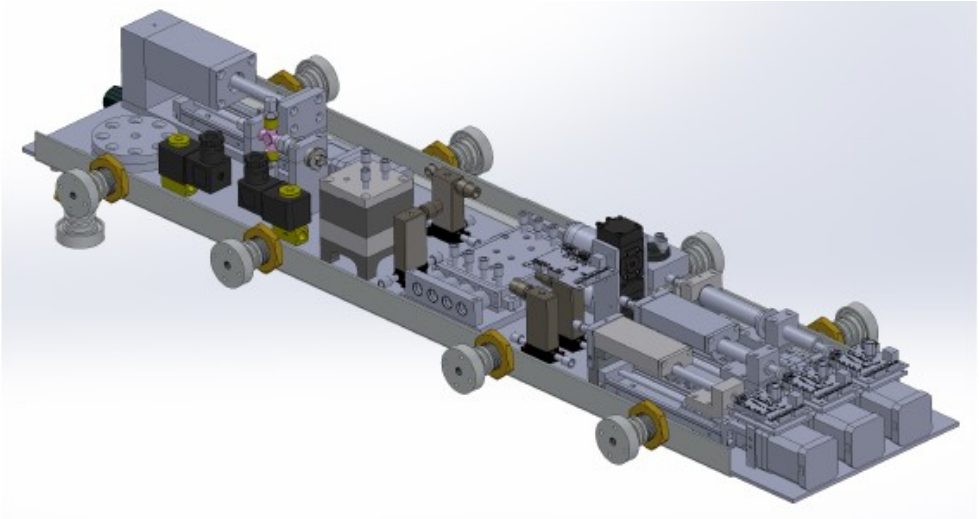


Figure 69. Final miniaturized laboratory layout designed by Solidworks.

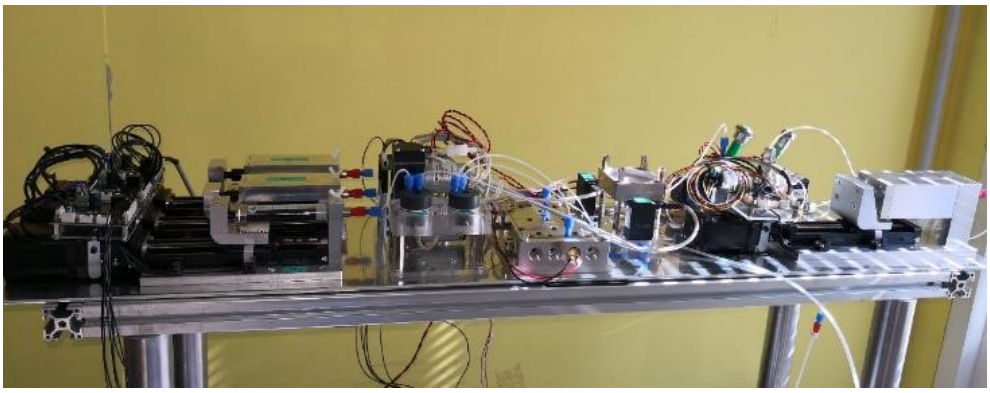


Figure 70. Final miniaturized laboratory manufactured.

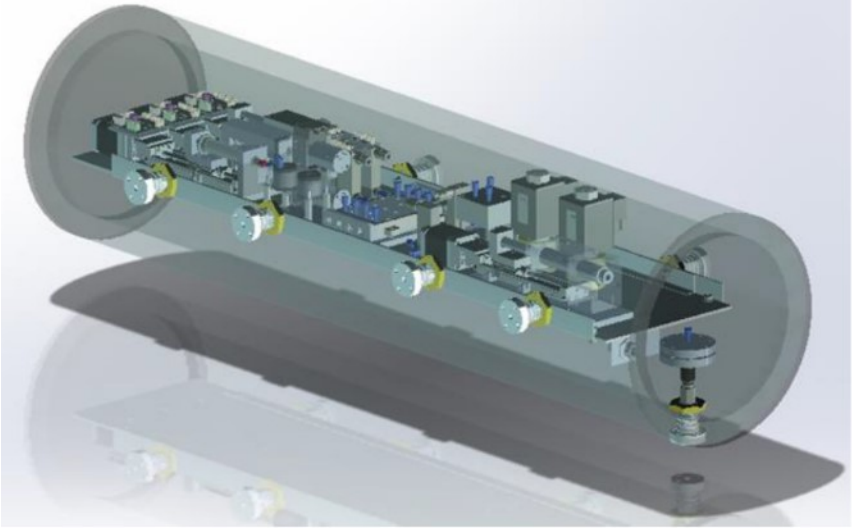


Figure 71. Miniaturized laboratory integrated within payload.

4.2.2.2 Optical design

As regards the measurement phase, the laser source - photodetector system described in the paragraph (3.2.1) has been engineered. The laser diode package chosen was TO CAN 5.6 mm for an easier integration into mechanical system. The optical system was optimized by ZEMAX™ and the lens was chosen (ACL 12708U Biaspherical lens from Thorlabs). The mechanical support for a correct laser beam treatment was designed and manufactured (Figure 72).

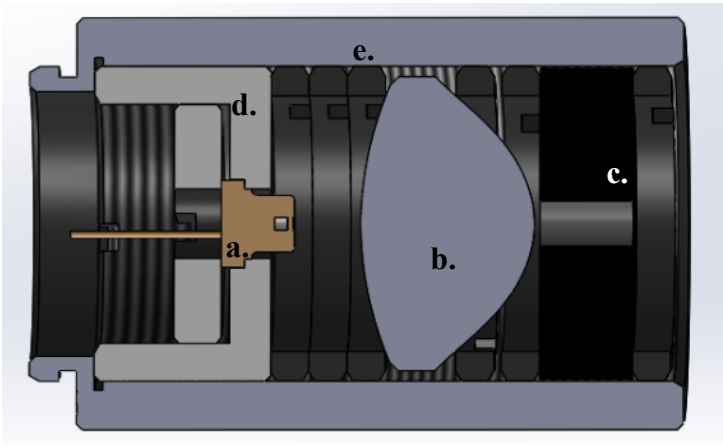


Figure 72. Optical system for laser source. a. Laser diode, b. Glass lens, c. Shutter, d. Mechanical support for laser diode, e. Mechanical support for entire optical system.

Measurement chamber and the optical system were designed and built. The measurement chamber was subdivided into 4 microchannels with a volume of 1 ml (Figure 73), creating the seats for the gaskets and optical windows, the inlet and outlet channels of the liquid for each channel, and a viton gasket compressed by a lid to guarantee the hydraulic seal (Figure 74). The materials used are AISI 316 steel, with passivation treatment to resist sea water, and glass, to maximize compatibility with chemical compounds.

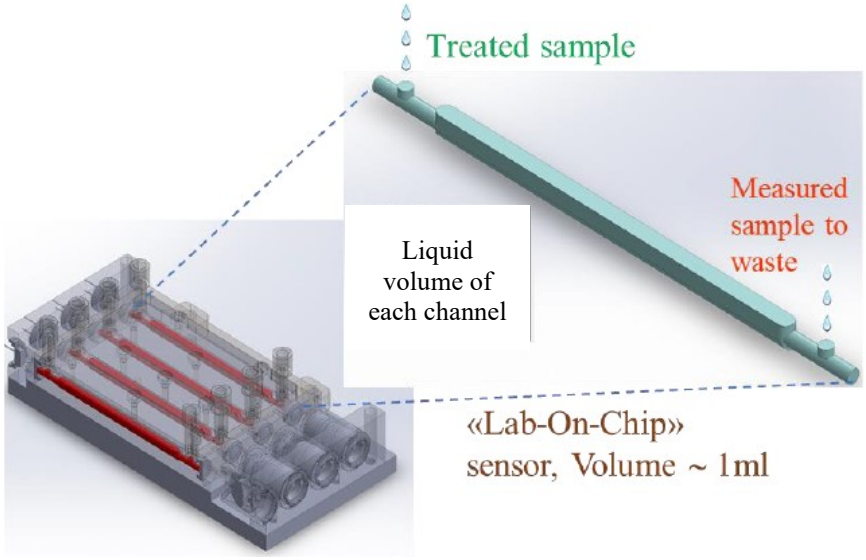


Figure 73. Measure chamber with four channels of 1ml of sample volume analyzed each.

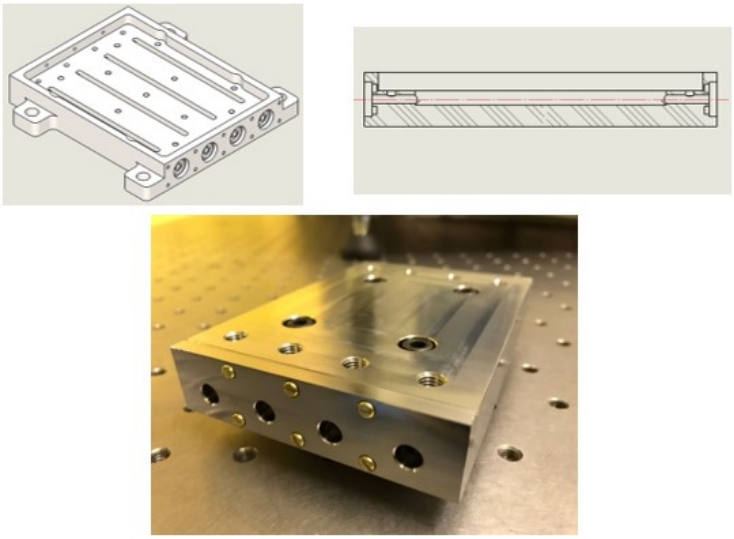


Figure 74. Engineering drawings and final manufacturing of measure chamber.

A mechanical support was also designed and built to ensure correct alignment between all the components of the optical system, namely laser source, measurement chamber, photodetector (Figure 75). To do this, a single Anticorodal aluminium base was manufactured, onto which to strip the source-holder supports, the detector-holder supports and the measuring chamber. These couplings allow very tight alignment tolerances, ensuring a measurement without optical errors.

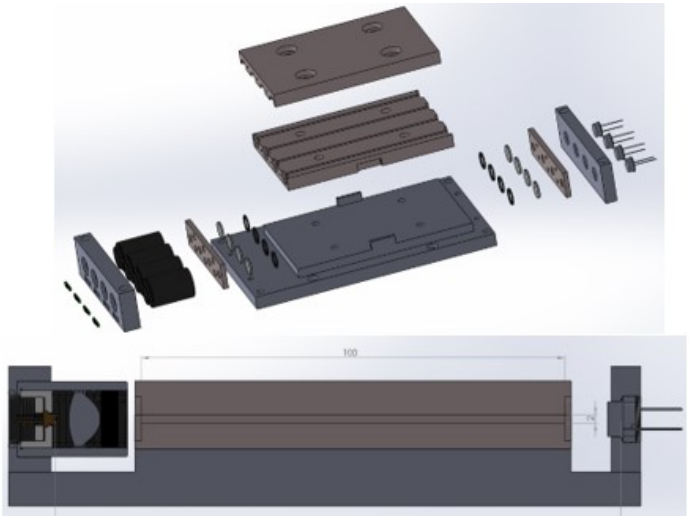


Figure 75. Explosion and section of entire measure system. Optical sources, measure chambers and photodetectors are aligned in order to perform concurrent measures in parallel for each metal.

The support was then built and installed (Figure 76). The sources have been realized with commercial anodized aluminium supports, while for the photodetectors ad hoc supports have been designed and manufactured. Alignment tests, power tests and diode stability tests were performed (Figure 77).

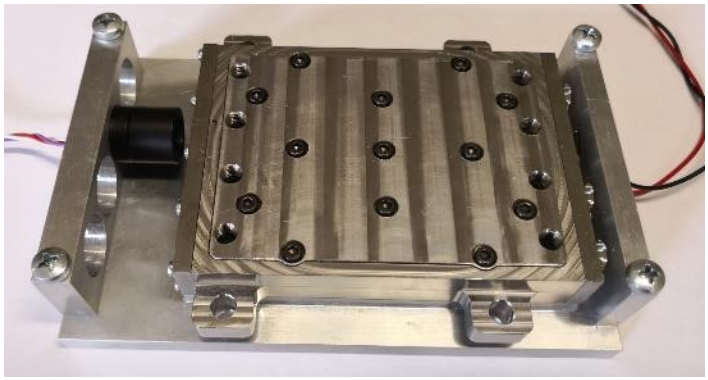


Figure 76. Final measure system manufactured.

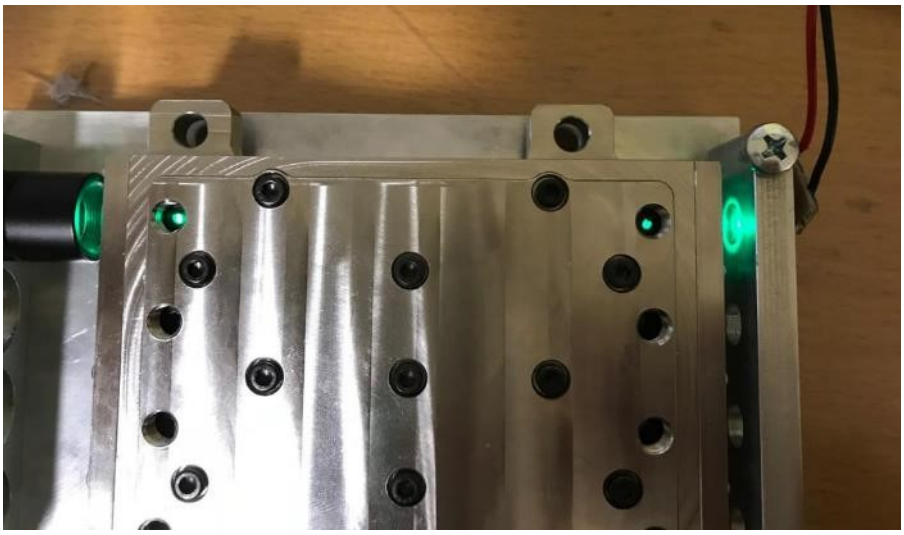


Figure 77. Alignment test for one channel of measure system.

4.2.2.3 Electrical design

Three different types of electronic boards have been designed and manufactured (Figure 78) by Microla Optoelectronics SRL.

- Motherboard, board able to perform communication with the AUV, communication with all other boards, management of firmware. An embedded PC was also integrated for software management and communication;
- Control board for solenoid valves, capable of managing also the vacuum pumps for the degassing system;
- Control board of the motor-driven syringes, which includes the stepper motor driver, and the motor-cycle management firmware, which controls initial syringe capacity and residual capacity.

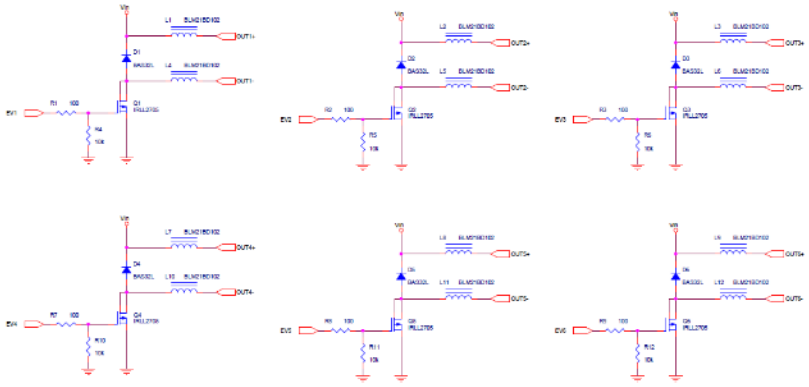


Figure 78. Circuit diagrams of electronic boards.

These boards were then designed, selecting all the components that would reduce the overall dimensions and consumption (Figure 79), considering the small payload space, and not to undermine the AUV battery pack capacity excessively and increase the number of measurements for each mission.

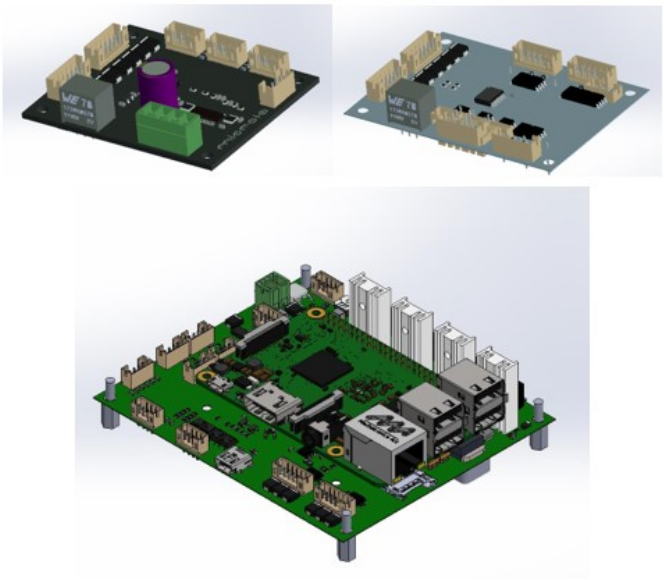


Figure 79. CAD renderings of electronic boards. From the upper-left Moto-syringes board, valve board, motherboard.

A management firmware was therefore written for each board, which provides for the integration of all the operations that make up the measurement. The boards were then manufactured and tested (Figure 80) by Microla Optoelectronics, and the

communication protocol between the motherboard and the AUV was also written and tested.

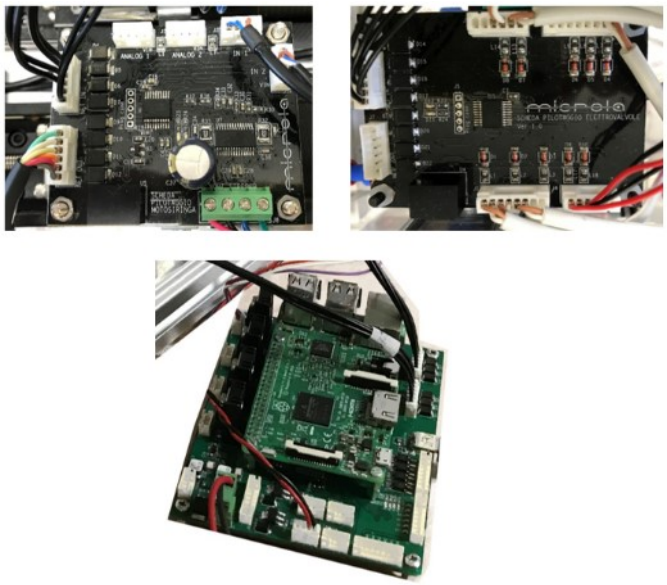


Figure 80. Final boards manufactured by Microla Optoelectronics SRL

As shown in Figure 81 ,each control board is equipped with an incoming RS422 interface and a regenerative output one (i.e. the output signal is driven by a second transceiver which generates the differential signals again in order to reach the next control device.)

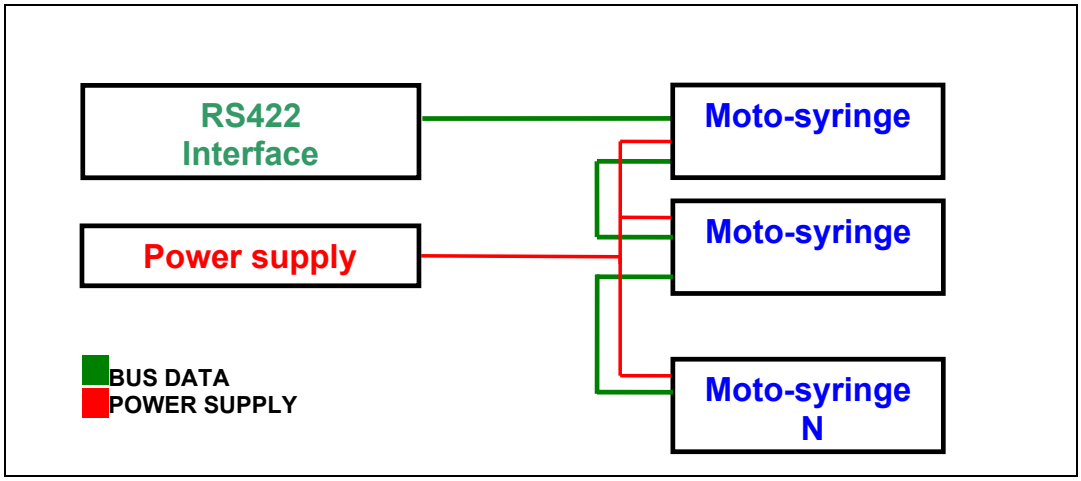


Figure 81. Logical scheme of electronic board.

On the moto-syringe board there are 8 connectors:

- J1 4pin input and analogue channel supply 1
- J2 4pin channel 1 digital input
- J3 4pin input and analogue channel 2 power supply
- J4 4pin digital channel 2 input
- J5 5pin for programming the microcontroller
- J6 6pin for RS422 input and 24V power supply
- J7 6pin for RS422 output and 24V power supply
- J8 4pin for connection to the syringe stepper motor

The pin assignment of the connectors of interest are:

J6 starting from the top pin (pin 1):

1. Signal A Rx + received from the board
2. Signal B Rx- received from the board
3. Signal Z Tx- transmitted to the board
4. Signal Y Tx+ transmitted to the board
5. 24V power supply
6. 0V power supply

J7 starting from the top pin (pin 1):

1. Signal A Rx + received from the board
2. Signal B Rx- received from the board
3. Signal Z Tx- transmitted to the board
4. Signal Y Tx+ transmitted to the board
5. 24V power supply
6. 0V power supply

J1 starting from the right (pin 1):

1. 24V power supply
2. Selectable power supply between 5V and 3.3V
3. Analog channel input 1
4. 0V

J2 starting from the right (pin 1):

1. 24V power supply
2. Selectable power supply between 5V and 3.3V

3. Analog channel input 1
4. 0V

J3 starting from the right (pin 1):
1. 24V power supply
2. Selectable power supply between 5V and 3.3V
3. Analog channel input 1
4. 0V

J4 starting from the right (pin 1):
1. 24V power supply
2. Selectable power supply between 5V and 3.3V
3. Analog channel input 1
4. 0V

J8 starting from the right (pin 1):
1. Winding 1 of the stepper motor
2. Winding 2 of the stepper motor
3. Winding 1 of the stepper motor
4. Winding 2 of the stepper motor

The communication parameters are baud rate 19200 bit/s, 8 data bits, no parity bit, 1 stop bit.

4.2.2.4 Firmware and communication system

Firmware was developed in collaboration with Microla Optoelectronics. There were two different kind of firmware, low-level firmware and high-level firmware. The low-level firmware was developed for low-level commands of each board. High-level firmware was developed to control the entire cycle, and each functional phase, and to communicate with AUV in order to start mission, report measures or errors, stop mission.

A physical address is assigned to each board, via low-level firmware, to ensure mutual communication. The board always listens on the incoming RS422 channel and upon receipt of a command this is forwarded to the outgoing RS422 port to receive the command just received on the following boards. The board in question is the only one enabled to reply and in response will send a status packet to indicate the status of its functions. The length of the command and status packages is always

the same and equal to 20 bytes. Each packet contains a byte that identifies the address of the board to which the command is addressed, in the case of a command packet, or the address of the board that responded by sending its status packet. 0xFF is an address that indicates that this packet must be received by all the boards, and only in this case, all the boards will send their status packet. Through this type of package, it is possible to command some functions of the board of interest or simply request its status without it performing any type of operation. Upon receipt of the undamaged package, then checking the correspondence of the address and the accuracy of the CRC code received with the one calculated on the board, the board will perform the requested operation and immediately send the status package. The syntax of the commands received is represented in Table 17 :

Table 17. Syntax for command communication.

*	***Type***	***Address***	***Command***	***Par1***	**…**	***Par13***	***CRCh***	***CRCl***	***0x0D***

Where:

- * Package start character.
- Type This field will always contain the letter ‘M’.
- Address This 1 byte field contains the address of the board (expressed in decimal). 0xFF for MULTI-CAST mode.
- Command This 1 byte field contains a character that identifies the type of command requested from the board. Subsequently the types of commands implemented by the board and the related parameters to be sent will be indicated.
- Par These fields take on a different value and meaning depending on the type of command sent. The type of command sent is indicated in the field called **Command**.
- CRCh + CRCl CRC16_CCIT_ZERO divided into two bytes
- 0D Carriage Return.

The high-level firmware (APPENDIX B), translates all the commands that each component of the system must perform during all the phases of the cycle, as previously discussed. This firmware, using the low-level firmware, marks the time of the various operations, activating the various components and synchronizing them together, in order to avoid errors in the cycle. The high-level firmware manages the quantities to be sampled, the pressures during the depressurization phase, the motion of the motor-syringes in the dosing and mixing phases, controls the laser diode in the measurement phase, interprets the signal coming from the detector, converting it first into absorbance and then , using the calibration curve, in measured concentration. Finally, it communicates with the submarine drone to understand when to start the cycle, when to move to a subsequent measurement point, when an error has occurred and the mission must be aborted, when the measurement cycle is finished, and the mission can be interrupted.

The AUV mission control is operated by a proprietary software of Gabri SRL (Figure 82).

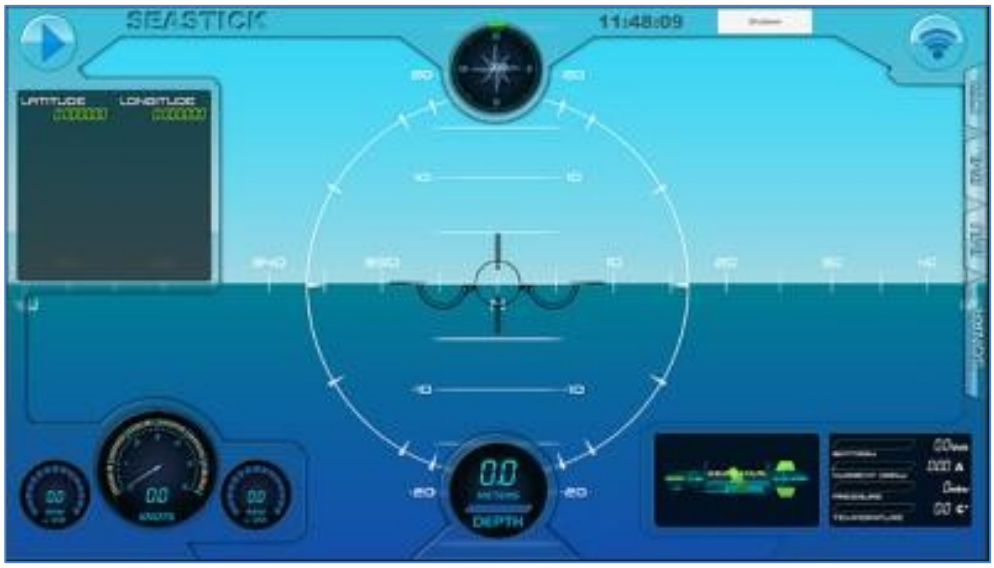


Figure 82. Seastick software interface.

4.3 Test on field

Preliminary tests have been carried out in order to validate the hydraulic system, verifying the sealings of all the components, the performance of the degassing elements and the high pressure system tightness. Moreover, tests were conducted to optimize the rinsing, mixing and dosing phases. For rinsing, it was verified that the routines implemented allowed to eliminate all the liquid from the system, except for the dead volumes, and that the contamination with the liquid of the following cycle did not occur. As for mixing, it was verified that the designed routine was enough for the complexation of the liquid with the reagents to take place correctly. Furthermore, the actual volume dosed in the mixing chamber was verified, both for the sample and for the reagents. The validation of the measurement routine was performed, which includes a first blank measurement with the not complexed sample, and a subsequent colorimetric measurement of the complexed one. In particular, the actual absence of bubbles in the measurement chamber, the correct duty cycle of the diode and the measurement protocol were investigated.

Thanks to this test it was possible to evaluate, in addition to the efficiency of the optical system and the degassing system already seen in the previous test, also the real effectiveness of the dosage and mixing routine and the electro-mechanical components involved.

First test on field[15] was carried out in order to perform a mechanical validation, an AUV- Payload communication validation, a filtering system validation and the validation of the miniaturized sensor platform (Figure 83).

Figure 83. AUV - Payload prepared for first test on field.

[15]https://unmig.mise.gov.it/index.php/it/198-notizie-stampa/2036014-primo-test-in-acqua-dell-autonomous-underwater-vehicle-auv

This test took place in Genova Sestri Ponente, at Lega Navale Italiana (Figure 84) the presence of the producers of the Seastick 300 (Gabri srl) and of the Ministry of Economic Development (Figure 85), and was the first launch of the vehicle in salt water, allowing the performance of the entire system to be evaluated while at sea, even underwater, simulating a complete mission with four sampling and measurement points.

Figure 84. First test on field was performed in a controlled water bay, placed in Genova Sestri Ponente at Lega Navale Italiana.

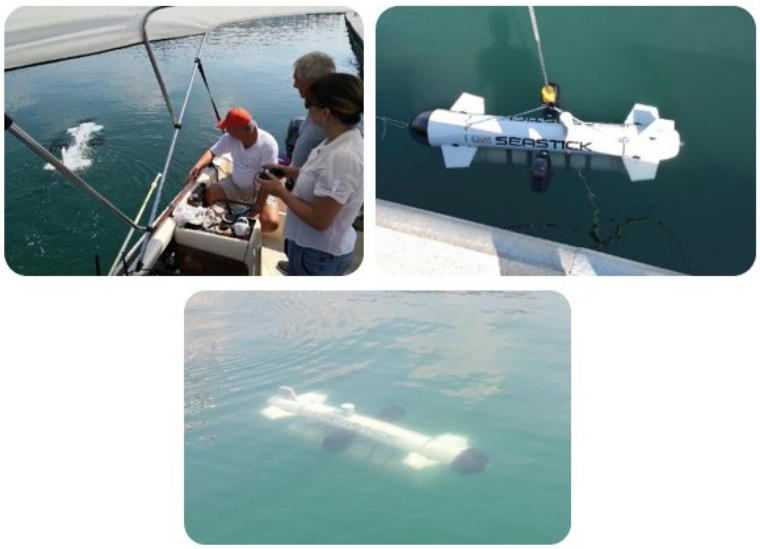


Figure 85. Scenes take from first day test.

Given the excellent results of the first on field test, the first off-shore tests will be scheduled near a gas platform off the coast of Ravenna (Figure 86).

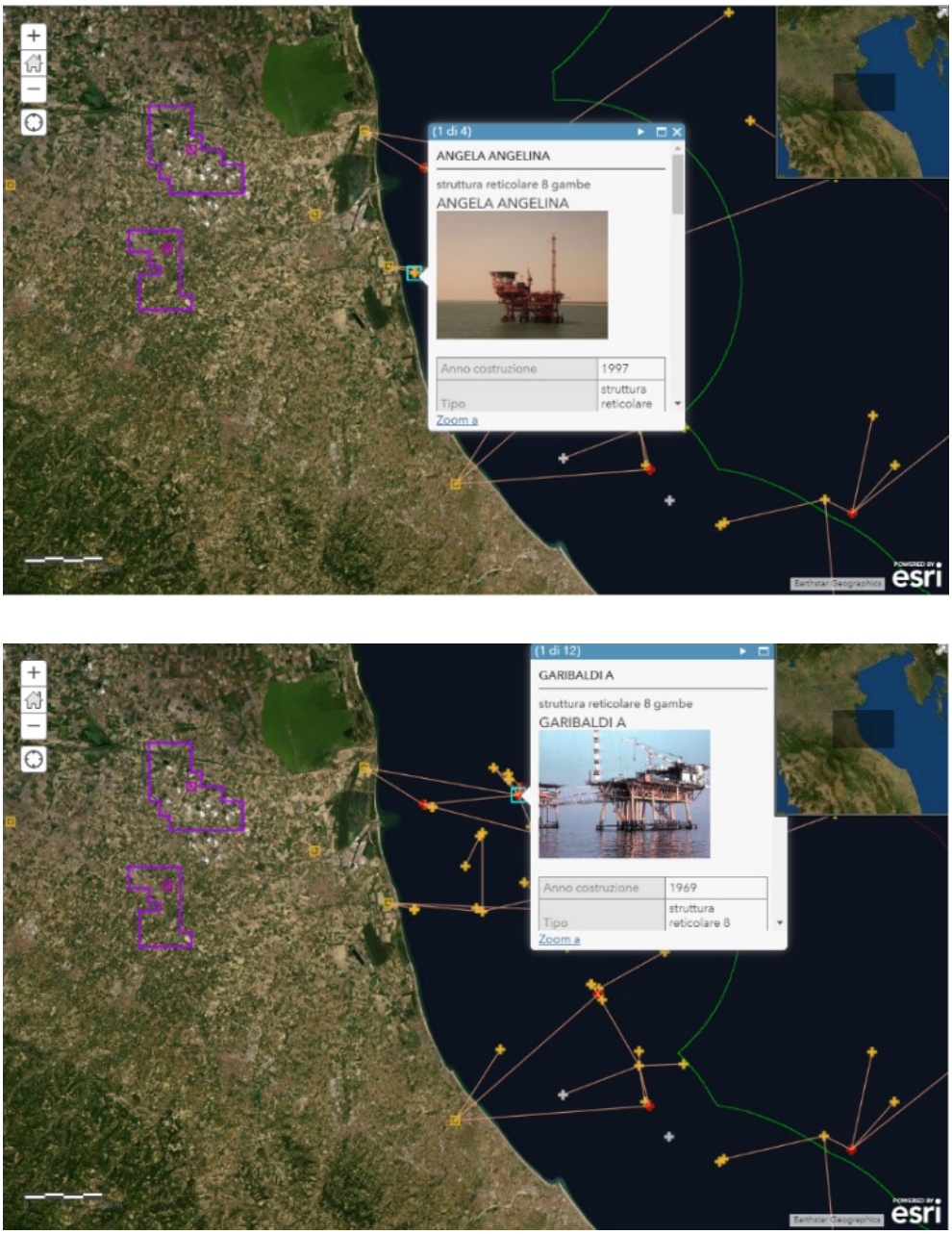


Figure 86. Two possible sites for the first off-shore tests. At the top the platform Angela Angelina, the closest to the coast, below the platform Garibaldi.

4.4 Measurement results

Finally, complete measurements were simulated with samples showing known Cr (VI) concentrations, to verify accuracy and precision of the measurement system. For each concentration, several measurement cycles were performed in order to obtain statistically valid data.

The concentrations used were:

- 0.010 ppm;
- 0.025 ppm;
- 0.100 ppm;
- 0.200 ppm;
- 0.250 ppm;
- 0.300 ppm.

In the first analysis, the stability of the blank sample measurement was investigated (Figure 87), which does not involve the dosing and mixing phase, and is therefore only sensitive to possible errors in the optical system and the degassing system.

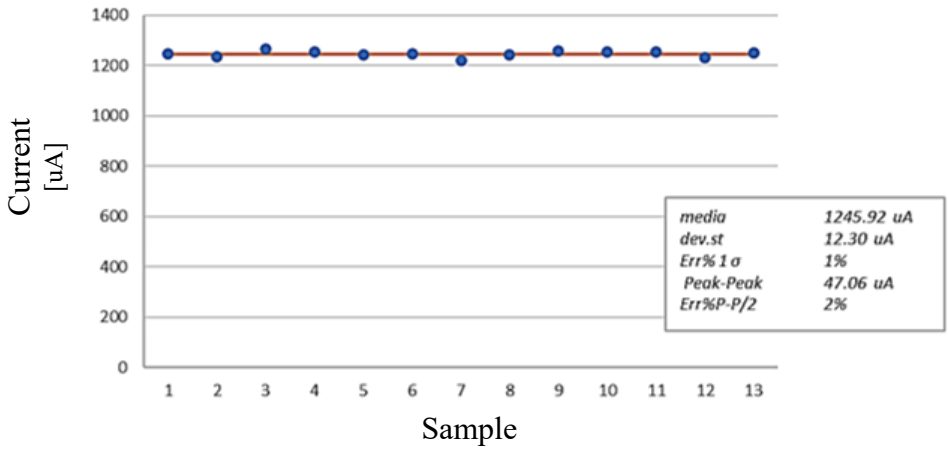


Figure 87. Blank sample measurement with complete automatic miniaturized laboratory integrated into AUV. The ordinate axis shows the current produced by photodiode, which is function of the optical power. The peak-to-peak percentage error stands at +/- 2%.

Measurement test carried good results, both as regards stability and accuracy. The tests on the measurement of white have brought excellent results, the stability of the reading and its accuracy. The measurement is sensitive to the different

temperature of the diode and to the different color of the solution with different concentrations, but the fact of repeating the measurement of the white immediately before the colorimetric measurement drastically decreases the possibility that an error occurs.

Subsequently, the quantification of the colorimetric measurement was investigated, calculating the absorbance (Figure 88), a dimensionless quantity index of the optical power absorbed as a function of the optical power, of the samples with known concentration previously listed.

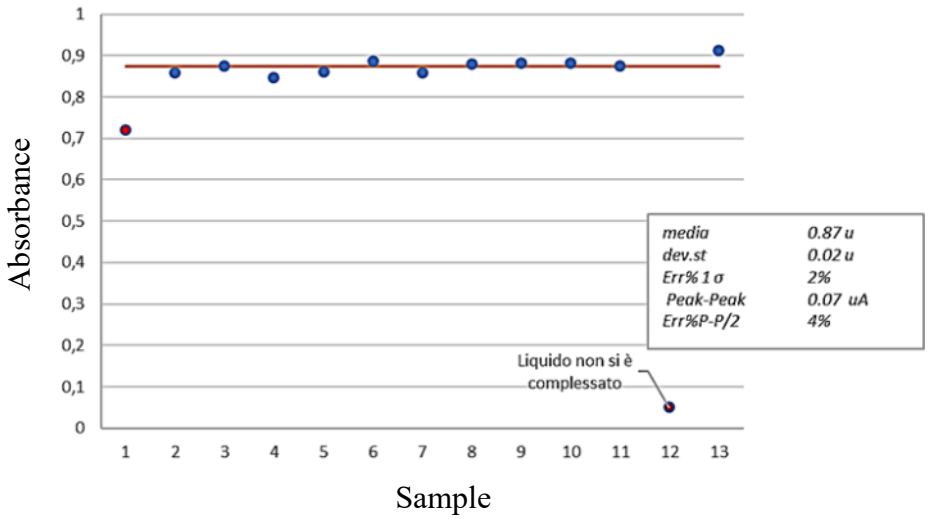


Figure 88. Repeatability test of the autonomous system, using a sample with a concentration of 0.2 ppm of Cr(VI). An "outlier" value is identified that is easily detectable and can be eliminated in post analysis.

Thanks to this test it was possible to evaluate, in addition to the efficiency of the optical system and the degassing system already observed in the previous tests, also the real effectiveness of the dosage and mixing routine and the electro-mechanical components involved. Also, in this case the test produced good results, with sporadic outlier values, indices of an error in mixing, but easily identifiable and replaceable with subsequent measurements. With the average absorbance values resulting from the different tests at different concentrations, it was possible to determine the system calibration line (Figure 89), thanks to which, being the absorbance directly proportional to the concentration according to the Lambert-Beer law, it is possible to convert the first quantity into the second, which is therefore the output of our interest.

By investigating the linearity of experimental data, it is therefore possible to evaluate the real goodness of the system. A poor linearity would in fact lead to a measurement error and would invalidate the system.

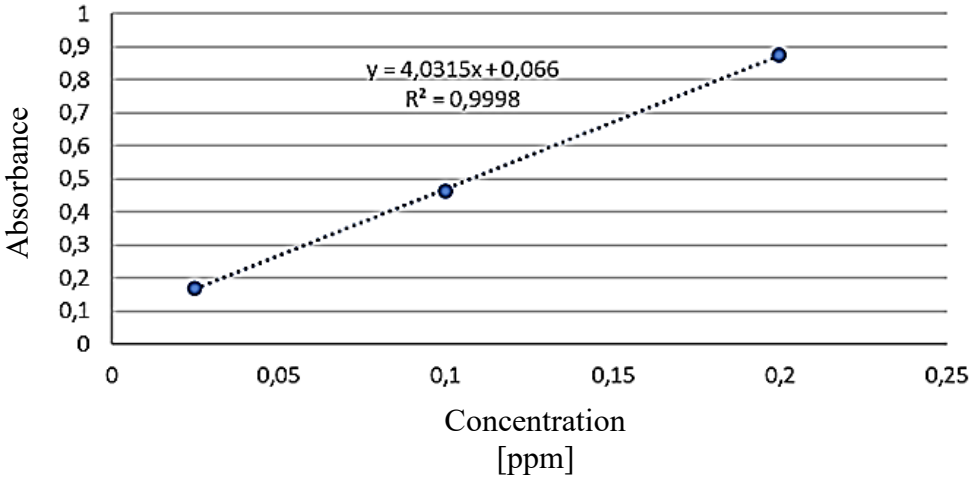


Figure 89.Calibration curve of miniaturized laboratory. The linearity is very high with an R^2 equal to 0.9998.

Calibration curves resulting from the experimental tests showed an excellent linearity, thus going to definitively validate the automatic monitoring system built in the measurement range 0.010-0.300 ppm. Thanks to the calibration curve, it is therefore possible to convert all absorbance tests into concentration measurements (Figure 90), thus ultimately evaluating the concentrations measured by the system in each test performed.

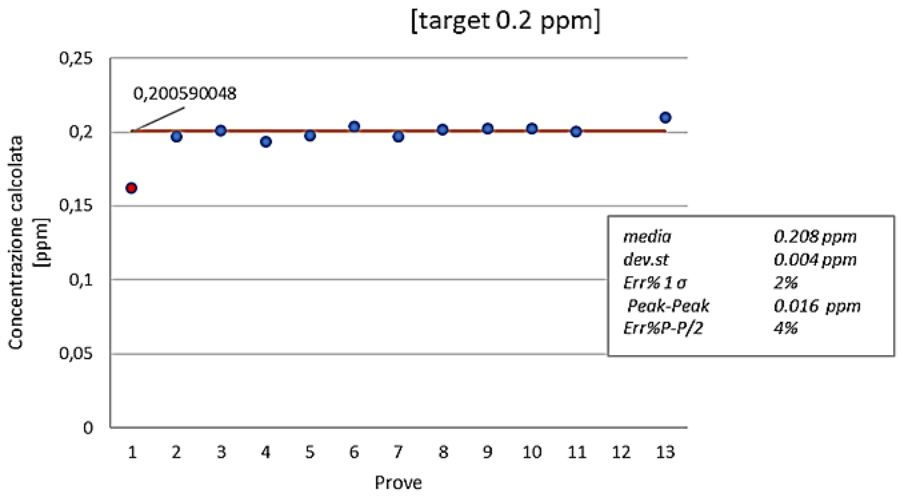


Figure 90. Concentration calculated with the absorbance values of Figure 88 and the calibration curve of Figure 89. The calculated average concentrations differ from the real datum of 0.0006 ppm, the peak to peak error stands at +/- 4%.

The average concentration values measured by the system for each set of tests, were compared to the real values of the concentrations of the samples used in the tests. It is possible to evaluate the precision of the automatic system (Table 18.).

Table 18. Average concentration measurements compared to the actual concentration of the solutions

Concentration prepared *[ppm]*	**Absorbance**	**Concentration calculated** *[ppm]*	**Absolute error** *[ppm]*	**Relative error** %
0.300	1.301	0.3062	0.0062	2.08%
0.250	1.098	0.2559	0.0059	2.35%
0.200	0.875	0.2006	0.0006	0.30%
0.100	0.464	0.0986	-0.0014	1.36%
0.025	0.170	0.0258	0.0008	3.13%
0.010	0.106	0.0098	-0.0002	1.77%

Conclusions

In this study, a new system for monitoring water from dissolved metallic pollutants has been studied. A miniaturized laboratory has been validated to work on-site in a completely autonomous way, that is, without the presence of an operator, performing the normal measurement operations that are currently performed in the laboratory, but on-field.

The main results obtained can be summarised as follow:

- Possibility of analysis of aqueous samples in a completely automated manner, in the sampling phase, in the sample preparation phase by means of reagents, in the colorimetric measurement phase and finally in the discharge or conservation phase of the complexed sample;
- Possibility of integrating the miniaturized laboratory in an autonomous submarine drone (AUV), capable of navigating up to a depth of 300 m, of sampling at predetermined and precise points, of measuring online during the phases of movement. This will allow you to map polluted areas in a timely manner, without waiting for the measurement times in the laboratory, with the possibility of correlating any anomalies to possible causes immediately;
- Possibility to use the integrated drone system to map and monitor the waters near offshore oil & gas platforms, as requested by the Ministry of Economic Development, improving the current monitoring system that presents poor timeliness and little data redundancy;
- Due to the high linearity, repeatability and stability of the measurement, the miniaturized laboratory can also be used as a device to be implanted onshore, as a monitoring tool for wastewater, groundwater and surface water, completely modifying the current water monitoring landscape both in Italy and in Europe;
- Thanks to the reduced volumes involved, it is possible to use limited quantities of reagents. This entails a high cost efficiency, which allows to considerably increase the quantity of measurements and therefore the availability of data to be catalogued and shared in cloud, fundamental feature from an IoT perspective.

The current system still has limitations, for example for each contaminant an optimized method that allows its implementation on the system is necessary, in order to significantly reduce the amount of energy and reagents needed and increase autonomy. The development of a new chemical detection method, at present, involves from six to twelve months of study for each additional metal or contaminant. Methods for Arsenic and other metals are being studying at the moment.

Future studies will concern further optimization of the device design, in order to reduce the overall dimensions and implement a greater number of methods in the same payload. Furthermore, the phases that make up the measurement cycle will be optimized, in order to minimize the cycle time, reduce consumption and increase the autonomy of the device and the number of possible measures without the intervention of the operator.

References

[1] J. G. Tundisi, T. Matsumura-Tundisi, V. S. Ciminelli, and F. A. Barbosa, "Water availability, water quality water governance: The future ahead," *IAHS-AISH Proc. Reports*, vol. 366, pp. 75–79, 2015.

[2] X. Wang, F. Zhang, and J. Ding, "Evaluation of water quality based on a machine learning algorithm and water quality index for the Ebinur Lake Watershed, China," *Sci. Rep.*, vol. 7, no. 1, pp. 1–18, 2017.

[3] Leo M.L. Nollet, *Handbook of water analysis*, SECOND EDI. Boca Raton: Taylor & Francis Group, 2007.

[4] D. Li and S. Liu, *Water Quality Monitoring and Management*. Elsevier, 2019.

[5] Gazzetta Ufficiale della Repubblica Italiana, *Decreto Legislativo 2 febbraio 2001, n. 31*. 2001, p. 27.

[6] Rapporti ISTISAN 07/31, "CROMO : METODO PER SPETTROMETRIA DI ASSORBIMENTO ATOMICO CON ATOMIZZAZIONE," 2001, pp. 255–260.

[7] Rapporti ISTISAN 07/31, "ALLUMINIO, BORO, CADMIO, CROMO, FERRO, MANGANESE, NICHEL, PIOMBO, RAME, SODIO, VANADIO: METODO SPETTROSCOPICO DI EMISSIONE CON SORGENTE A PLASMA INDUTTIVO ISS.DBA.035.REV00," 2001, pp. 322–328.

[8] L. Beck, T. Bernauer, and A. Kalbhenn, "Environmental, political, and economic determinants of water quality monitoring in Europe," *Water Resour. Res.*, vol. 46, no. 11, pp. 1–10, 2010.

[9] J. Bartram and R. Ballance, *Water Quality Monitoring*, vol. 42, no. 1–2. London: E&FN Spon, 1996.

[10] "Patent Inspiration." [Online]. Available: www.patentinspiration.com.

[11] C. Sicard *et al.*, "Tools for water quality monitoring and mapping using paper-based sensors and cell phones," *Water Res.*, vol. 70, pp. 360–369, 2015.

[12] D. Kar, P. Sur, S. K. Mandal, T. Saha, and R. K. Kole, "Assessment of heavy metal pollution in surface water," *Int. J. Environ. Sci. Technol.*, vol. 5, no. 1, pp. 119–124, 2008.

[13] K. K. Onchoke and S. A. Sasu, "Determination of Hexavalent Chromium (Cr(VI)) Concentrations via Ion Chromatography and UV-Vis Spectrophotometry in Samples Collected from Nacogdoches Wastewater Treatment Plant, East Texas (USA)," *Adv. Environ. Chem.*, vol. 2016, no. Iii, pp. 1–10, 2016.

[14] S. Ahuja, *Monitoring Water Quality: Pollution Assessment, Analysis, and Remediation*. 2013.

[15] P. K. Govil and A. K. Krishna, "Soil and Water Contamination by Potentially Hazardous Elements: A Case History From India," in *Environmental Geochemistry: Site Characterization, Data Analysis and Case Histories: Second Edition*, 2017.

[16] Y. Ouyang, "Evaluation of river water quality monitoring stations by principal component analysis," *Water Res.*, vol. 39, no. 12, pp. 2621–2635, 2005.

[17] K. R. Campbell, "Chromium Accumulation in Three Species of Central Florida Centrarchids," *Bull. Environ. Contam. Toxicol, Springer-Verlag New York Inc.*, vol. 54, pp. 185–190, 1995.

[18] T. P. Lambrou, C. C. Anastasiou, C. G. Panayiotou, and M. M. Polycarpou, "A low-cost sensor network for real-time monitoring and contamination detection in drinking water distribution systems," *IEEE Sens. J.*, vol. 14, no. 8, pp. 2765–2772, 2014.

[19] R. Yue and T. Ying, "A water quality monitoring system based on wireless sensor network & solar power supply," *2011 IEEE Int. Conf. Cyber Technol. Autom. Control. Intell. Syst. CYBER 2011*, vol. 12, no. Icese 2011, pp. 126–129, 2011.

[20] S. C. Azhar, A. Z. Aris, M. K. Yusoff, M. F. Ramli, and H. Juahir, "Classification of River Water Quality Using Multivariate Analysis," *Procedia Environ. Sci.*, vol. 30, pp. 79–84, 2015.

[21] G. Tuna, O. Arkoc, and K. Gulez, "Continuous monitoring of water quality using portable and low-cost approaches," *Int. J. Distrib. Sens. Networks*, vol. 2013, 2013.

[22] H. R, S. J, and S. R, "Techniques of trend analysis for monthly water quality data," *Water Resour. Res.*, vol. 18, no. 1, pp. 107–121, 1982.

[23] D. Šunjka and S. Lazić, "Water sampling techniques for continuous monitoring of pesticides in water," *Pestic. i Fitomedicina*, vol. 32, no. 2, pp. 85–93, 2017.

[24] M. V. Storey, B. van der Gaag, and B. P. Burns, "Advances in on-line drinking water quality monitoring and early warning systems," *Water Res.*, vol. 45, no. 2, pp. 741–747, 2011.

[25] Peng Jiang, Qingbo Huang, Jianzhong Wang, Xiaohua Dai, and Ruizhong Lin, "Research on Wireless Sensor Networks Routing Protocol for Wetland Water Environment Monitoring," in *First International Conference on Innovative Computing, Information and Control (ICICIC'06)*, 2006, pp. 251–254.

[26] F. Catania *et al.*, "Cr (VI) in Water : Continuous , on Site Spectrophotometric Determination Laboratory test preliminary to microfluidic device prototyping," *Int. J. Appl. Sci. Environ. Eng.*, vol. 1, no. 1, pp. 265–270, 2018.

[27] M. Periolatto, F. Catania, L. Scaltrito, C. F. Pirri, M. Cocuzza, and S. Ferrero,

"Continuous, on Site Spectrophotometric Determination of Zinc ions in Water, Laboratory test preliminary to microfluidic device prototyping," in *Eighth Intl. Conf. on Advances in Bio-Informatics, Bio-Technology and Environmental Engineering - ABBE 2019*, 2019, pp. 1–7.

[28] H. Van der Jagt, "Water analysis," in *Encyclopedia of Analytical Science (Second Edition)*, 2013, pp. 233–252.

[29] W. Fresenius, K. E. Quentin, W. Schneider, and others, *Water analysis; a practical guide to physico-chemical, chemical and microbiological water examination and quality assurance*. 1988.

[30] "ISO/TC 147 - Water quality." [Online]. Available: https://www.iso.org/committee/52834.html.

[31] "ISO 15839:2003." [Online]. Available: https://www.iso.org/standard/28740.html.

[32] K. Y. Tam, J. P. Larsen, B. A. Coles, and R. G. Compton, *Electromagnetism Theory and Applications*, vol. 407. 1996.

[33] M. Periolatto, F. Catania, F. Pirri, L. Scaltrito, M. Cocuzza, and S. Ferrero, "Spectrophotometric monitoring system, integrated in an autonomous underwater vehicle, for continuous heavy metal detection near offshore sites," in *Offshore Mediterranean Conference and Exhibition 2019, OMC 2019*, 2019, pp. 1–12.

[34] M. Pesavento, G. Alberti, and R. Biesuz, "Analytical methods for determination of free metal ion concentration, labile species fraction and metal complexation capacity of environmental waters: A review," *Anal. Chim. Acta*, vol. 631, no. 2, pp. 129–141, 2009.

[35] F. Ferrero, C. Tonetti, and M. Periolatto, "Adsorption of chromate and cupric ions onto chitosan-coated cotton gauze," *Carbohydr. Polym.*, vol. 110, pp. 367–373, 2014.

[36] G. Gaulitz, "Ultraviolet and Visible Spectroscopy G," Wiley-VCH Verlag GmbH & Co. KGaA, Ed. Weinheim, 2012, pp. 552–593.

[37] A. Hangan, L. Vacariu, O. Cret, and H. Hedesiu, "A prototype for the remote monitoring of water parameters," *Proc. - 19th Int. Conf. Control Syst. Comput. Sci. CSCS 2013*, pp. 634–639, 2013.

[38] C. Encinas, E. Ruiz, J. Cortez, and A. Espinoza, "Design and implementation of a distributed IoT system for the monitoring of water quality in aquaculture," *Wirel. Telecommun. Symp.*, 2017.

[39] M. Cocuzza, L. Scaltrito, S. Ferrero, S. L. Marasso, D. Perrone, and C. F. Pirri, "Innovative technologies for offshore platforms safety and environmental monitoring," *Geoing. Ambient. e Mineraria*, vol. 152, no. 3, pp. 41–50, 2017.

[40] J. Fries and H. Getrost, *Organische Reagenzien fur die Spurenanalyse*. Darmstadt, 1975.

[41] G. Nordberg, "Metals: Chemical Properties and Toxicity," in *Encyclopaedia of Occupational Health and Safety, Part IX, Chemicals*, 2011.

[42] B. Palumbo-Roe, "Zinc : the invisible threat," in *Earthwise 26, British*

Geological Survey © NERC 2010, 2010, pp. 72–73.
[43] R. Verma and P. Dwivedi, "Heavy metal water pollution- A case study," *Recent Res. Sci. Technol.*, vol. 5, no. 5, pp. 98–99, 2013.
[44] C. E. Säbel, J. M. Neureuther, and S. Siemann, "A spectrophotometric method for the determination of zinc, copper, and cobalt ions in metalloproteins using Zincon," *Anal. Biochem.*, vol. 397, no. 2, pp. 218–226, 2010.
[45] J. Ma, G. Qin, Y. Zhang, J. Sun, S. Wang, and L. Jiang, "Heavy metal removal from aqueous solutions by calcium silicate powder from waste coal fly-ash," *J. Clean. Prod.*, vol. 182, pp. 776–782, 2018.
[46] M. M. Areco, M. dos Santos Afonso, and E. Valdman, "Zinc biosorption by seaweed illustrated by the zincon colorimetric method and the Langmuir isotherm," *J. Chem. Educ.*, 2007.
[47] A. Kocyła, A. Pomorski, and A. Krężel, "Molar absorption coefficients and stability constants of Zincon metal complexes for determination of metal ions and bioinorganic applications," *J. Inorg. Biochem.*, vol. 176, no. May, pp. 53–65, 2017.
[48] DGS-UNIMIG, "Rapporto annuale 2018 - Attività dell'anno 2017," 2018.
[49] R. B. Wynn *et al.*, "Autonomous Underwater Vehicles (AUVs): Their past, present and future contributions to the advancement of marine geoscience," *Mar. Geol.*, vol. 352, pp. 451–468, 2014.
[50] Quinci Andrea, "Analisi di stabilità di un sistema AUV per il monitoraggio del Mar Mediterraneo," Politecnico di Torino, 2018.

APPENDIX A

```
Public Class Form1

    Dim VettoreRicezioneSeriale As String
    Dim ByteRicezioneSeriale(42) As Byte
    Dim VettoreTrasmissioneSeriale As String
    Dim DatiTrasmissioneSeriale(42) As Byte
    Dim VettoreStatoElettrovalvole(10)
    Dim IdSchedaDaInterrogare
    Dim ValoreCRC As Integer
    Dim IndiceRicezione As Integer
    Dim FaseCiclo As Integer
    Dim AvviaCiclo As Integer
    Dim Capacitamotos3 As Integer
    Dim NumeroCicli As Integer
    Dim StatoMovMotos3 As Integer
    Dim ControlloGMotosiringa As Integer

    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Dim Ciclo As Integer
        ControlloGMotosiringa = 0
        ComboBox1.DataSource = System.IO.Ports.SerialPort.GetPortNames
        'RIEMPE LE COMBOBOX

        For Ciclo = 0 To 255
            If Ciclo < 16 Then

ComboBoxIDscheda.Items.Add("0x0" + (Convert.ToString(Ciclo, 16)))
            Else

ComboBoxIDscheda.Items.Add("0x" + (Convert.ToString(Ciclo, 16)))
            End If
        Next

        For Ciclo = 0 To 9

ComboBoxEVtempo.Items.Add("EV" + (Convert.ToString(Ciclo + 1, 10)))
        Next
        ValoriIniziali()
        DisabilitaComandi()
        FaseCiclo = 0
        AvviaCiclo = 0

    End Sub

    Private Sub ButtonSerialConn_Click(sender As Object, e As EventArgs) Handles ButtonSerialConn.Click
        Try
            If (ButtonSerialConn.Text = "Connetti") Then
                SerialPort1.PortName = ComboBox1.Text
                SerialPort1.Open()
                ButtonSerialConn.Text = "Disconnetti"
                ValoriIniziali EV()
                AbilitaComandi()
            Else
                'Nel timerSerialClose viene verificato che non ci sia piu niente da spedire in modo tale da non mandare in errore la seriale

                TimerSerialClose.Enabled = True
                ButtonSerialConn.Text = "Connetti"
                ValoriIniziali EV()
                DisabilitaComandi()
            End If

        Catch ex As Exception
            MessageBox.Show(ex.Message)
        End Try

    End Sub

    Private Sub TimerSerialClose_Tick(sender As Object, e As EventArgs) Handles TimerSerialClose.Tick
        Dim Ciclo As Integer
        If SerialPort1.BytesToWrite = 0 Then
            SerialPort1.Close()
```

```
        For Ciclo = 0 To 19
            ByteRicezioneSeriale(Ciclo) = 0
        Next

        TimerSerialClose.Enabled = False
    End If
End Sub

Private Sub TimerTimeoutSeriale_Tick(sender As Object, e As EventArgs) Handles TimerTimeoutSeriale.Tick
    If RadioButtonEV.Checked = True Then

Me.Invoke(MethodDelegateImpostaColoreStatusLight, StatuLightEV1, Color.Yellow)

Me.Invoke(MethodDelegateImpostaColoreStatusLight, StatuLightEV2, Color.Yellow)

Me.Invoke(MethodDelegateImpostaColoreStatusLight, StatuLightEV3, Color.Yellow)

Me.Invoke(MethodDelegateImpostaColoreStatusLight, StatuLightEV4, Color.Yellow)

Me.Invoke(MethodDelegateImpostaColoreStatusLight, StatuLightEV5, Color.Yellow)

Me.Invoke(MethodDelegateImpostaColoreStatusLight, StatuLightEV6, Color.Yellow)

Me.Invoke(MethodDelegateImpostaColoreStatusLight, StatuLightEV7, Color.Yellow)

Me.Invoke(MethodDelegateImpostaColoreStatusLight, StatuLightEV8, Color.Yellow)

Me.Invoke(MethodDelegateImpostaColoreStatusLight, StatuLightEV9, Color.Yellow)

Me.Invoke(MethodDelegateImpostaColoreStatusLight, StatuLightEV10, Color.Yellow)

Me.Invoke(MethodDelegateImpostaColoreStatusLight, RectangleShape1, Color.Yellow)

Me.Invoke(MethodDelegateImpostaColoreStatusLight, RectangleShape2, Color.Yellow)

Me.Invoke(MethodDelegateImpostaColoreStatusLight, RectangleShape3, Color.Yellow)

Me.Invoke(MethodDelegateImpostaColoreStatusLight, RectangleShape4, Color.Yellow)

Me.Invoke(MethodDelegateImpostaColoreStatusLight, RectangleShape5, Color.Yellow)

Me.Invoke(MethodDelegateImpostaColoreStatusLight, RectangleShape6, Color.Yellow)

Me.Invoke(MethodDelegateImpostaColoreStatusLight, RectangleShape7, Color.Yellow)

Me.Invoke(MethodDelegateImpostaColoreStatusLight, RectangleShape8, Color.Yellow)

Me.Invoke(MethodDelegateImpostaColoreStatusLight, RectangleShape9, Color.Yellow)

Me.Invoke(MethodDelegateImpostaColoreStatusLight, RectangleShape10, Color.Yellow)
    End If

    TimerTimeoutSeriale.Enabled = False

End Sub

Private Sub TimerRichiediStato_Tick(sender As Object, e As EventArgs) Handles TimerRichiediStato.Tick
    If RadioButtonEV.Checked = True Then
        RichiediStatoEV()
    End If

    RichiediStatoMotos()

End Sub

Private Sub TrasmettiSeriale()
    TimerRichiediStato.Enabled = False
    Dim NumeroCaratterif As Integer
    NumeroCaratterif = 0
    While NumeroCaratterif <> 0
        NumeroCaratterif = SerialPort1.BytesToRead
    End While
```

```
        VettoreRicezioneSeriale = Nothing
        IndiceRicezione = 0

SerialPort1.Write(VettoreTrasmissioneSeriale)
        TimerRichiediStato.Enabled = True
    End Sub

    Private Sub TrasmettiSerialeByte()
        TimerRichiediStato.Enabled = False
        Dim NumeroCaratterif As Integer
        NumeroCaratterif = 0
        While NumeroCaratterif <> 0
            NumeroCaratterif = SerialPort1.BytesToRead
        End While
        VettoreRicezioneSeriale = Nothing
        IndiceRicezione = 0

SerialPort1.Write(DatiTrasmissioneSeriale, 0, 20)
        TimerRichiediStato.Enabled = True
    End Sub

    Private Sub SerialPort1_DataReceived(ByVal sender As Object, ByVal e As System.IO.Ports.SerialDataReceivedEventArgs) Handles SerialPort1.DataReceived
        Dim Caratteref As Integer
        Dim NumeroCaratterif As Integer
        Dim Lunghezzaf As Integer
        NumeroCaratterif = 1
        While NumeroCaratterif <> 0
            Caratteref = SerialPort1.ReadByte

ByteRicezioneSeriale(IndiceRicezione) = Convert.ToByte(Caratteref)
            IndiceRicezione = IndiceRicezione + 1
            NumeroCaratterif = SerialPort1.BytesToRead
            VettoreRicezioneSeriale = VettoreRicezioneSeriale + Chr(Caratteref)
            Lunghezzaf = Len(VettoreRicezioneSeriale)
            If ByteRicezioneSeriale(0) = &H2A And ByteRicezioneSeriale(19) = &HD And Lunghezzaf = 20 Then
                If ByteRicezioneSeriale(1) = &H45 Then 'E

'Me.Invoke(MethodDelegateModificaVisibilitaGroupbox, GroupBoxElettrovalvole, True)

'Me.Invoke(MethodDelegateModificaVisibilitaGroupbox, GroupBoxMotosiringa, False)
                    AnalisiDatiElettrovalvole()
                ElseIf ByteRicezioneSeriale(1) = &H4D Then 'M

'Me.Invoke(MethodDelegateModificaVisibilitaGroupbox, GroupBoxElettrovalvole, False)

'Me.Invoke(MethodDelegateModificaVisibilitaGroupbox, GroupBoxMotosiringa, True)
                    AnalisiDatiMotosiringa()
                End If
                IndiceRicezione = 0
            ElseIf Lunghezzaf > 20 Then
                'ByteRicezioneSeriale = Nothing
                IndiceRicezione = 0
            End If

        End While
    End Sub

    Private Sub ComboBoxIDscheda_SelectedIndexChanged(sender As Object, e As EventArgs) Handles ComboBoxIDscheda.SelectedIndexChanged
        IdSchedaDaInterrogare = ComboBoxIDscheda.SelectedIndex
    End Sub

    Private Sub CalcolaCRC()
        Dim Crcf As Short
        Dim Carf As Byte
        Dim Indice As Integer
        Crcf = 0
        For Indice = 0 To 16
            Carf = DatiTrasmissioneSeriale(Indice)
            Crcf = (&HFF And (Crcf >> 8)) Or (Crcf << 8)
            Crcf = Crcf Xor Convert.ToInt16(Carf)
            Crcf = Crcf Xor Convert.ToInt16(Crcf And &HFF) >> 4
            Crcf = Crcf Xor ((Crcf << 8) << 4)
            Crcf = Crcf Xor ((Crcf And &HFF) << 4) << 1
```

```
        Next
        ValoreCRC = Crcf
    End Sub

    Private Sub ValoriIniziali()
        TimerTimeoutSeriale.Enabled = False
        ButtonSerialConn.Text = "Connetti"
        TextBoxTipo.Text = Nothing
        TextBoxIndirizzo.Text = Nothing
        TextBoxStato.Text = Nothing
        TextBoxErrorCode.Text = Nothing
        'ATTIVA il RADIO BOTTONE ALL'APERTURA!!!!!
        RadioButtonEV.Checked = True
        RadioButtonMotos.Checked = True
        ValoriInizialiMotos()
        ValoriInizialiEV()
        ComboBoxIDscheda.SelectedIndex = 0
    End Sub

    'Private Sub TextBox1_TextChanged(sender As Object, e As EventArgs)
    '    Dim PosizioneH, PosizioneL, Posizione As Int16
    '    Dim Analog1H, Analog1L, Analog1 As Integer
    '    Dim Analog2H, Analog2L, Analog2 As Integer
    '    Dim CapacitaH, CapacitaL, Capacita As Integer

    '    If ByteRicezioneSeriale(1) = &H4D Then 'M                    'TIPO
    '
Me.Invoke(MethodDelegateScriviTextBoxString, TextBoxTipo, "Motosiringa")
    '    End If

    '
Me.Invoke(MethodDelegateScriviTextBoxByte, TextBoxIndirizzo, ByteRicezioneSeriale(2))       'INDIRIZZO

    '    If ByteRicezioneSeriale(3) = &H45 Then 'E                    'STATO
    '
Me.Invoke(MethodDelegateScriviTextBoxString, TextBoxStato, "Error")
    '    ElseIf ByteRicezioneSeriale(3) = &H4B Then 'K
    '
Me.Invoke(MethodDelegateScriviTextBoxString, TextBoxStato, "Normal")
    '    End If
    '
Me.Invoke(MethodDelegateScriviTextBoxByte, TextBoxErrorCode, ByteRicezioneSeriale(4))       'ERROR CODE

    '    'Stato POSIZIONE
    '    If ByteRicezioneSeriale(5) = &H49 Then  'I
    '
Me.Invoke(MethodDelegateCheckBoxChecked, CheckBoxCerta, False)
    '
Me.Invoke(MethodDelegateCheckBoxChecked, CheckBoxPresunta, True)
    '    ElseIf ByteRicezioneSeriale(5) = &H53 Then  'S
    '
Me.Invoke(MethodDelegateCheckBoxChecked, CheckBoxCerta, True)
    '
Me.Invoke(MethodDelegateCheckBoxChecked, CheckBoxPresunta, False)
    '    End If

    '    'Valore POSIZIONE
    '    PosizioneH = ByteRicezioneSeriale(6)
    '    PosizioneH = PosizioneH << 8
    '    PosizioneL = ByteRicezioneSeriale(7)
    '    Posizione = PosizioneH + PosizioneL
    '    If Posizione = 0 Then
    '
Me.Invoke(MethodDelegateScriviTextBoxString, TextBoxPosizioneMS, (Convert.ToString(Posizione) + ": vuota"))
    '    Else
    '
Me.Invoke(MethodDelegateScriviTextBoxString, TextBoxPosizioneMS, Posizione)
    '    End If

    '    'Stato movimentazione
    '    If ByteRicezioneSeriale(8) = &H53 Then     'S
    '
Me.Invoke(MethodDelegateScriviTextBoxString, TextBoxStatoMovimentazione, "Suzione")
```

```
    '   ElseIf ByteRicezioneSeriale(8) = &H49
Then  'I
    '
Me.Invoke(MethodDelegateScriviTextBoxS
tring,           TextBoxStatoMovimentazione,
"Iniezione")
    '   ElseIf ByteRicezioneSeriale(8) = &H46
Then  'F
    '
Me.Invoke(MethodDelegateScriviTextBoxS
tring,           TextBoxStatoMovimentazione,
"Ferma")
    '   End If

    '   'Stato digital IN 1
    '      If ByteRicezioneSeriale(9) = &H31
Then     '1
    '
Me.Invoke(MethodDelegateImpostaColoreS
tatusLight, StatusLightDI1, Color.Lime)
    '
Me.Invoke(MethodDelegateScriviLabel,
LabelDigitalIN1, "ON")
    '   ElseIf ByteRicezioneSeriale(9) = &H30
Then  '0
    '
Me.Invoke(MethodDelegateImpostaColoreS
tatusLight, StatusLightDI1, Color.Red)
    '
Me.Invoke(MethodDelegateScriviLabel,
LabelDigitalIN1, "OFF")
    '   End If

    '   'Stato digital IN 2
    '     If ByteRicezioneSeriale(10) = &H31
Then     '1
    '
Me.Invoke(MethodDelegateImpostaColoreS
tatusLight, StatusLightDI2, Color.Lime)
    '
Me.Invoke(MethodDelegateScriviLabel,
LabelDigitalIN2, "ON")
    '       ElseIf  ByteRicezioneSeriale(10)  =
&H30 Then  '0
    '
Me.Invoke(MethodDelegateImpostaColoreS
tatusLight, StatusLightDI2, Color.Red)
    '
Me.Invoke(MethodDelegateScriviLabel,
LabelDigitalIN2, "OFF")
    '   End If

    '   'Valore Analog IN 1
    '   Analog1H = ByteRicezioneSeriale(11)
    '   Analog1H = &HFFFF And (Analog1H
<< 8)
    '   Analog1L = ByteRicezioneSeriale(12)
    '   Analog1 = Analog1H + Analog1L
    '
Me.Invoke(MethodDelegateScriviTextBoxS
tring, TextBoxAnalogIN1, Analog1)

    '   'Valore Analog IN 2
    '   Analog2H = ByteRicezioneSeriale(13)
    '   Analog2H = &HFFFF And (Analog2H
<< 8)
    '   Analog2L = ByteRicezioneSeriale(14)
    '   Analog2 = Analog2H + Analog2L
    '
Me.Invoke(MethodDelegateScriviTextBoxS
tring, TextBoxAnalogIN2, Analog2)

    '   'Valore Capacità massima Siringa
    '   CapacitaH = ByteRicezioneSeriale(15)
    '   CapacitaH = &HFFFF And (CapacitaH
<< 8)
    '   CapacitaL = ByteRicezioneSeriale(16)
    '   Capacita = CapacitaH + CapacitaL
    '
Me.Invoke(MethodDelegateScriviTextBoxS
tring, TextBoxCapacitaMax, Capacita)
    'End Sub

    Private Sub ValoriIniziali EV()
        ButtonEV1.Text = ""
        ButtonEV2.Text = ""
        ButtonEV3.Text = ""
        ButtonEV4.Text = ""
        ButtonEV5.Text = ""
        ButtonEV6.Text = ""
        ButtonEV7.Text = "Spenta"
        ButtonEV8.Text = "Spenta"
        ButtonEV9.Text = "Spenta"
        ButtonEV10.Text = "Spenta"
        ButtonMultiEVSpegni.Text = "Spegni"
        ButtonMultiEVAccendi.Text          =
"Accendi"
        ButtonAvviaTimerEV.Text = "Start"
        StatuLightEV1.FillColor = Color.Red
        StatuLightEV2.FillColor = Color.Red
        RectangleShape1.FillColor = Color.Red
        RectangleShape2.FillColor = Color.Red
        RectangleShape3.FillColor = Color.Red
```

```
        RectangleShape5.FillColor =
Color.Lime
        RectangleShape9.FillColor = Color.Red
        RectangleShape10.FillColor =
Color.Red
<<<<<< = Color.Red
        StatuLightEV4.FillColor = Color.Red
        StatuLightEV5.FillColor = Color.Red
        StatuLightEV6.FillColor = Color.Red
        StatuLightEV7.FillColor = Color.Red
        StatuLightEV8.FillColor = Color.Red
        StatuLightEV9.FillColor = Color.Red
        StatuLightEV10.FillColor = Color.Red

Me.Invoke(MethodDelegateComboBoxInde
xSelect, ComboBoxEVtempo,
Convert.ToByte(0))

'Me.Invoke(MethodDelegateComboBoxInde
xSelect, ComboBoxIDscheda,
Convert.ToByte(0))

Me.Invoke(MethodDelegateComboBoxInde
xSelect, ComboBoxStatoIniziale,
Convert.ToByte(0))

Me.Invoke(MethodDelegateCheckBoxChec
ked, CheckBoxEV1, False)

Me.Invoke(MethodDelegateCheckBoxChec
ked, CheckBoxEV2, False)

Me.Invoke(MethodDelegateCheckBoxChec
ked, CheckBoxEV3, False)

Me.Invoke(MethodDelegateCheckBoxChec
ked, CheckBoxEV4, False)

Me.Invoke(MethodDelegateCheckBoxChec
ked, CheckBoxEV5, False)

Me.Invoke(MethodDelegateCheckBoxChec
ked, CheckBoxEV6, False)

Me.Invoke(MethodDelegateCheckBoxChec
ked, CheckBoxEV7, False)

Me.Invoke(MethodDelegateCheckBoxChec
ked, CheckBoxEV8, False)

Me.Invoke(MethodDelegateCheckBoxChec
ked, CheckBoxEV9, False)

Me.Invoke(MethodDelegateCheckBoxChec
ked, CheckBoxEV10, False)

Me.Invoke(MethodDelegateScriviTextBoxS
tring, TextBoxStato, Nothing)

Me.Invoke(MethodDelegateScriviTextBoxS
tring, TextBoxErrorCode, Nothing)

Me.Invoke(MethodDelegateScriviTextBoxS
tring, TextBoxIndirizzo, Nothing)

Me.Invoke(MethodDelegateScriviTextBoxS
tring, TextBoxTipo, Nothing)
    End Sub

    Private Sub ValoriInizialiMotos()

Me.Invoke(MethodDelegateImpostaColoreS
tatusLight, StatusLightDI1, Color.Red)

Me.Invoke(MethodDelegateScriviLabel,
LabelDigitalIN1, "OFF")

Me.Invoke(MethodDelegateImpostaColoreS
tatusLight, StatusLightDI2, Color.Red)

Me.Invoke(MethodDelegateScriviLabel,
LabelDigitalIN2, "OFF")

Me.Invoke(MethodDelegateCheckBoxChec
ked, CheckBoxCerta, False)

Me.Invoke(MethodDelegateCheckBoxChec
ked, CheckBoxPresunta, False)

Me.Invoke(MethodDelegateScriviTextBoxS
tring, TextBoxPosizioneMS, 0)

Me.Invoke(MethodDelegateScriviTextBoxS
tring, TextBoxStatoMovimentazione, 0)

Me.Invoke(MethodDelegateScriviTextBoxS
tring, TextBoxCapacitaMax, 0)

Me.Invoke(MethodDelegateScriviTextBoxS
tring, TextBoxAnalogIN1, 0)

Me.Invoke(MethodDelegateScriviTextBoxS
tring, TextBoxAnalogIN2, 0)
```

```
Me.Invoke(MethodDelegateScriviTextBoxString, TextBoxQuantiaMovimentazione, 0)

Me.Invoke(MethodDelegateScriviTextBoxString, TextBoxMovimentazioneVelocita, 0)


Me.Invoke(MethodDelegateScriviTextBoxString, TextBoxQuantiaMovimentazione2, 0)

Me.Invoke(MethodDelegateScriviTextBoxString, TextBoxMovimentazioneVelocita2, 0)


Me.Invoke(MethodDelegateScriviTextBoxString, TextBoxQuantiaMovimentazione3, 0)

Me.Invoke(MethodDelegateScriviTextBoxString, TextBoxMovimentazioneVelocita3, 0)


Me.Invoke(MethodDelegateScriviTextBoxString, TextBoxOffsetMS, 0)
        RadioButtonSuzione.Checked = False
        RadioButtonIniezione.Checked = False


Me.Invoke(MethodDelegateScriviTextBoxString, TextBoxStato, Nothing)

Me.Invoke(MethodDelegateScriviTextBoxString, TextBoxErrorCode, Nothing)

Me.Invoke(MethodDelegateScriviTextBoxString, TextBoxIndirizzo, Nothing)

Me.Invoke(MethodDelegateScriviTextBoxString, TextBoxTipo, Nothing)
    End Sub

    Private Sub DisabilitaComandi()
        GroupBox1.Enabled = False
        GroupBox3.Enabled = False
        GroupBox4.Enabled = False
        ComboBoxIDscheda.Enabled = False
        TimerRichiediStato.Enabled = False
        RadioButtonEV.Enabled = False
        RadioButtonMotos.Enabled = False
        GroupBoxElettrovalvole.Enabled = False
        GroupBoxMotosiringa.Enabled = False
        GroupBox2.Enabled = False
    End Sub

    Private Sub AbilitaComandi()
        GroupBox1.Enabled = True
        GroupBox3.Enabled = True
        GroupBox4.Enabled = True

        ComboBoxIDscheda.Enabled = True
        TimerRichiediStato.Enabled = True
        RadioButtonEV.Enabled = True
        RadioButtonMotos.Enabled = True
        GroupBox2.Enabled = True
        GroupBoxElettrovalvole.Enabled = True
        ValoriIniziali EV()
        GroupBoxMotosiringa.Enabled = True
        ValoriIniziali Motos()
    End Sub

    'Private Sub RadioButtonEV_CheckedChanged(sender As Object, e As EventArgs) Handles RadioButtonEV.CheckedChanged
    '  If RadioButtonEV.Checked = True Then
    '      GroupBoxElettrovalvole.Enabled = True
    '      GroupBoxMotosiringa.Enabled = False
    '      ValoriIniziali Motos()
    '  End If
    'End Sub

    'Private Sub RadioButtonMotos_CheckedChanged(sender As Object, e As EventArgs) Handles RadioButtonMotos.CheckedChanged
    '  If RadioButtonMotos.Checked = True Then
    '      GroupBoxElettrovalvole.Enabled = False
    '      GroupBoxMotosiringa.Enabled = True
    '      ValoriIniziali EV()
    '  End If
    'End Sub

    Private Sub AnalisiDatiElettrovalvole()

        If ByteRicezioneSeriale(1) = &H45 Then 'E                    'TIPO
```

```
Me.Invoke(MethodDelegateScriviTextBoxString, TextBoxTipo, "Elettrovalvole")
    End If


Me.Invoke(MethodDelegateScriviTextBoxByte, TextBoxIndirizzo, ByteRicezioneSeriale(2))    'INDIRIZZO

    If ByteRicezioneSeriale(3) = &H45 Then 'E                    'STATO

Me.Invoke(MethodDelegateScriviTextBoxString, TextBoxStato, "Error")
    ElseIf ByteRicezioneSeriale(3) = &H4B Then 'K

Me.Invoke(MethodDelegateScriviTextBoxString, TextBoxStato, "Normal")
    End If


Me.Invoke(MethodDelegateScriviTextBoxByte, TextBoxErrorCode, ByteRicezioneSeriale(4))    'ERROR CODE

    If ByteRicezioneSeriale(5) = &H41 Then

Me.Invoke(MethodDelegateImpostaColoreStatusLight, StatuLightEV1, Color.Lime)

Me.Invoke(MethodDelegateImpostaColoreStatusLight, RectangleShape2, Color.Lime)

Me.Invoke(MethodDelegateScriviButtonString, ButtonEV1, "")
    ElseIf ByteRicezioneSeriale(5) = &H53 Then

Me.Invoke(MethodDelegateImpostaColoreStatusLight, StatuLightEV1, Color.Red)

Me.Invoke(MethodDelegateImpostaColoreStatusLight, RectangleShape2, Color.Red)

Me.Invoke(MethodDelegateScriviButtonString, ButtonEV1, "")
    End If
    If ByteRicezioneSeriale(6) = &H41 Then

Me.Invoke(MethodDelegateImpostaColoreStatusLight, StatuLightEV2, Color.Lime)

Me.Invoke(MethodDelegateImpostaColoreStatusLight, RectangleShape1, Color.Lime)

Me.Invoke(MethodDelegateScriviButtonString, ButtonEV2, "")
    ElseIf ByteRicezioneSeriale(6) = &H53 Then

Me.Invoke(MethodDelegateImpostaColoreStatusLight, StatuLightEV2, Color.Red)

Me.Invoke(MethodDelegateImpostaColoreStatusLight, RectangleShape1, Color.Red)

Me.Invoke(MethodDelegateScriviButtonString, ButtonEV2, "")
    End If
    If ByteRicezioneSeriale(7) = &H41 Then

Me.Invoke(MethodDelegateImpostaColoreStatusLight, StatuLightEV3, Color.Lime)

Me.Invoke(MethodDelegateImpostaColoreStatusLight, RectangleShape3, Color.Lime)

Me.Invoke(MethodDelegateImpostaColoreStatusLight, RectangleShape5, Color.Red)

Me.Invoke(MethodDelegateImpostaColoreStatusLight, RectangleShape7, Color.Red)

Me.Invoke(MethodDelegateImpostaColoreStatusLight, RectangleShape8, Color.Red)

Me.Invoke(MethodDelegateImpostaColoreStatusLight, RectangleShape6, Color.Lime)

Me.Invoke(MethodDelegateImpostaColoreStatusLight, RectangleShape4, Color.Lime)

Me.Invoke(MethodDelegateScriviButtonString, ButtonEV3, "")
    ElseIf ByteRicezioneSeriale(7) = &H53 Then

Me.Invoke(MethodDelegateImpostaColoreStatusLight, StatuLightEV3, Color.Red)
```

```
Me.Invoke(MethodDelegateImpostaColoreS
tatusLight, RectangleShape3, Color.Red)

Me.Invoke(MethodDelegateImpostaColoreS
tatusLight, RectangleShape5, Color.Lime)

Me.Invoke(MethodDelegateImpostaColoreS
tatusLight, RectangleShape7, Color.Lime)

Me.Invoke(MethodDelegateImpostaColoreS
tatusLight, RectangleShape8, Color.Lime)

Me.Invoke(MethodDelegateImpostaColoreS
tatusLight, RectangleShape6, Color.Lime)

Me.Invoke(MethodDelegateImpostaColoreS
tatusLight, RectangleShape4, Color.Lime)

Me.Invoke(MethodDelegateScriviButtonStri
ng, ButtonEV3, "")
    End If
    If ByteRicezioneSeriale(8) = &H41
Then

Me.Invoke(MethodDelegateImpostaColoreS
tatusLight, StatuLightEV4, Color.Lime)

Me.Invoke(MethodDelegateImpostaColoreS
tatusLight, RectangleShape9, Color.Lime)

Me.Invoke(MethodDelegateImpostaColoreS
tatusLight, RectangleShape10, Color.Lime)

Me.Invoke(MethodDelegateScriviButtonStri
ng, ButtonEV4, "")
    ElseIf ByteRicezioneSeriale(8) = &H53
Then

Me.Invoke(MethodDelegateImpostaColoreS
tatusLight, StatuLightEV4, Color.Red)

Me.Invoke(MethodDelegateImpostaColoreS
tatusLight, RectangleShape9, Color.Red)

Me.Invoke(MethodDelegateImpostaColoreS
tatusLight, RectangleShape10, Color.Red)

Me.Invoke(MethodDelegateScriviButtonStri
ng, ButtonEV4, "")
    End If
    If ByteRicezioneSeriale(9) = &H41
Then

Me.Invoke(MethodDelegateImpostaColoreS
tatusLight, StatuLightEV5, Color.Lime)

Me.Invoke(MethodDelegateScriviButtonStri
ng, ButtonEV5, "")
    ElseIf ByteRicezioneSeriale(9) = &H53
Then

Me.Invoke(MethodDelegateImpostaColoreS
tatusLight, StatuLightEV5, Color.Red)

Me.Invoke(MethodDelegateScriviButtonStri
ng, ButtonEV5, "")
    End If
    If ByteRicezioneSeriale(10) = &H41
Then

Me.Invoke(MethodDelegateImpostaColoreS
tatusLight, StatuLightEV6, Color.Lime)

Me.Invoke(MethodDelegateScriviButtonStri
ng, ButtonEV6, "")
    ElseIf ByteRicezioneSeriale(10) =
&H53 Then

Me.Invoke(MethodDelegateImpostaColoreS
tatusLight, StatuLightEV6, Color.Red)

Me.Invoke(MethodDelegateScriviButtonStri
ng, ButtonEV6, "")
    End If
    If ByteRicezioneSeriale(11) = &H41
Then

Me.Invoke(MethodDelegateImpostaColoreS
tatusLight, StatuLightEV7, Color.Lime)

Me.Invoke(MethodDelegateScriviButtonStri
ng, ButtonEV7, "Spegni")
    ElseIf ByteRicezioneSeriale(11) =
&H53 Then

Me.Invoke(MethodDelegateImpostaColoreS
tatusLight, StatuLightEV7, Color.Red)

Me.Invoke(MethodDelegateScriviButtonStri
ng, ButtonEV7, "Accendi")
    End If
```

```
    If ByteRicezioneSeriale(12) = &H41 Then

Me.Invoke(MethodDelegateImpostaColoreStatusLight, StatuLightEV8, Color.Lime)

Me.Invoke(MethodDelegateScriviButtonString, ButtonEV8, "Spegni")
    ElseIf ByteRicezioneSeriale(12) = &H53 Then

Me.Invoke(MethodDelegateImpostaColoreStatusLight, StatuLightEV8, Color.Red)

Me.Invoke(MethodDelegateScriviButtonString, ButtonEV8, "Accendi")
    End If
    If ByteRicezioneSeriale(13) = &H41 Then

Me.Invoke(MethodDelegateImpostaColoreStatusLight, StatuLightEV9, Color.Lime)

Me.Invoke(MethodDelegateScriviButtonString, ButtonEV9, "Spegni")
    ElseIf ByteRicezioneSeriale(13) = &H53 Then

Me.Invoke(MethodDelegateImpostaColoreStatusLight, StatuLightEV9, Color.Red)

Me.Invoke(MethodDelegateScriviButtonString, ButtonEV9, "Accendi")
    End If
    If ByteRicezioneSeriale(14) = &H41 Then

Me.Invoke(MethodDelegateImpostaColoreStatusLight, StatuLightEV10, Color.Lime)

Me.Invoke(MethodDelegateScriviButtonString, ButtonEV10, "Spegni")
    ElseIf ByteRicezioneSeriale(14) = &H53 Then

Me.Invoke(MethodDelegateImpostaColoreStatusLight, StatuLightEV10, Color.Red)

Me.Invoke(MethodDelegateScriviButtonString, ButtonEV10, "Accendi")
    End If

    Dim indice

    For indice = 0 To 10
      VettoreStatoElettrovalvole(indice) = ByteRicezioneSeriale(indice + 5)
    Next

    TimerTimeoutSeriale.Enabled = False

  End Sub

  Private Sub AnalisiDatiMotosiringa()
    Dim PosizioneH, PosizioneL, Posizione As Int16
    Dim Analog1H, Analog1L, Analog1 As Integer
    Dim Analog2H, Analog2L, Analog2 As Integer
    Dim CapacitaH, CapacitaL, Capacita As Integer

    If ByteRicezioneSeriale(1) = &H4D Then 'M                    'TIPO

Me.Invoke(MethodDelegateScriviTextBoxString, TextBoxTipo, "Motosiringa")
    End If

    If ByteRicezioneSeriale(2) = 3 Then
      StatoMovMotos3 = ByteRicezioneSeriale(8)
    End If

Me.Invoke(MethodDelegateScriviTextBoxByte, TextBoxIndirizzo, ByteRicezioneSeriale(2))    'INDIRIZZO

    If ByteRicezioneSeriale(3) = &H45 Then 'E                    'STATO

Me.Invoke(MethodDelegateScriviTextBoxString, TextBoxStato, "Error")
    ElseIf ByteRicezioneSeriale(3) = &H4B Then 'K

Me.Invoke(MethodDelegateScriviTextBoxString, TextBoxStato, "Normal")
    End If

Me.Invoke(MethodDelegateScriviTextBoxByte, TextBoxErrorCode, ByteRicezioneSeriale(4))    'ERROR CODE
```

```
        'Stato POSIZIONE
        If ByteRicezioneSeriale(5) = &H49 Then 'I

Me.Invoke(MethodDelegateCheckBoxChecked, CheckBoxCerta, False)

Me.Invoke(MethodDelegateCheckBoxChecked, CheckBoxPresunta, True)
        ElseIf ByteRicezioneSeriale(5) = &H53 Then 'S

Me.Invoke(MethodDelegateCheckBoxChecked, CheckBoxCerta, True)

Me.Invoke(MethodDelegateCheckBoxChecked, CheckBoxPresunta, False)
        End If

        'Valore POSIZIONE
        PosizioneH = ByteRicezioneSeriale(6)
        PosizioneH = PosizioneH << 8
        PosizioneL = ByteRicezioneSeriale(7)
        Posizione = PosizioneH + PosizioneL
        If Posizione = 0 Then

Me.Invoke(MethodDelegateScriviTextBoxString, TextBoxPosizioneMS, (Convert.ToString(Posizione) + ": vuota"))
        Else

Me.Invoke(MethodDelegateScriviTextBoxString, TextBoxPosizioneMS, Posizione)
        End If
        If ByteRicezioneSeriale(2) = 3 Then
           Capacitamotos3 = Posizione
        End If

        'Stato movimentazione
        If ByteRicezioneSeriale(8) = &H53 Then 'S

Me.Invoke(MethodDelegateScriviTextBoxString, TextBoxStatoMovimentazione, "Suzione")
        ElseIf ByteRicezioneSeriale(8) = &H49 Then 'I

Me.Invoke(MethodDelegateScriviTextBoxString, TextBoxStatoMovimentazione, "Iniezione")
        ElseIf ByteRicezioneSeriale(8) = &H46 Then 'F

Me.Invoke(MethodDelegateScriviTextBoxString, TextBoxStatoMovimentazione, "Ferma")
        End If

        'Stato digital IN 1
        If ByteRicezioneSeriale(9) = &H31 Then '1

Me.Invoke(MethodDelegateImpostaColoreStatusLight, StatusLightDI1, Color.Lime)

Me.Invoke(MethodDelegateScriviLabel, LabelDigitalIN1, "ON")
        ElseIf ByteRicezioneSeriale(9) = &H30 Then '0

Me.Invoke(MethodDelegateImpostaColoreStatusLight, StatusLightDI1, Color.Red)

Me.Invoke(MethodDelegateScriviLabel, LabelDigitalIN1, "OFF")
        End If

        'Stato digital IN 2
        If ByteRicezioneSeriale(10) = &H31 Then '1

Me.Invoke(MethodDelegateImpostaColoreStatusLight, StatusLightDI2, Color.Lime)

Me.Invoke(MethodDelegateScriviLabel, LabelDigitalIN2, "ON")
        ElseIf ByteRicezioneSeriale(10) = &H30 Then '0

Me.Invoke(MethodDelegateImpostaColoreStatusLight, StatusLightDI2, Color.Red)

Me.Invoke(MethodDelegateScriviLabel, LabelDigitalIN2, "OFF")
        End If

        'Valore Analog IN 1
        Analog1H = ByteRicezioneSeriale(11)
        Analog1H = &HFFFF And (Analog1H << 8)
        Analog1L = ByteRicezioneSeriale(12)
        Analog1 = Analog1H + Analog1L
```

```
Me.Invoke(MethodDelegateScriviTextBoxString, TextBoxAnalogIN1, Analog1)

        'Valore Analog IN 2
        Analog2H = ByteRicezioneSeriale(13)
        Analog2H = &HFFFF And (Analog2H << 8)
        Analog2L = ByteRicezioneSeriale(14)
        Analog2 = Analog2H + Analog2L

Me.Invoke(MethodDelegateScriviTextBoxString, TextBoxAnalogIN2, Analog2)

        'Valore Capacità massima Siringa
        CapacitaH = ByteRicezioneSeriale(15)
        CapacitaH = &HFFFF And (CapacitaH << 8)
        CapacitaL = ByteRicezioneSeriale(16)
        Capacita = CapacitaH + CapacitaL

Me.Invoke(MethodDelegateScriviTextBoxString, TextBoxCapacitaMax, Capacita)

    End Sub

    Private Sub RichiediStatoEV()
        Dim Temp As Integer
        Dim Valore, L, H As Byte

        DatiTrasmissioneSeriale(0) = &H2A
        DatiTrasmissioneSeriale(1) = &H45 'E
        DatiTrasmissioneSeriale(2) = &HA
        DatiTrasmissioneSeriale(3) = &H53 'S
        DatiTrasmissioneSeriale(4) = &H1
        DatiTrasmissioneSeriale(5) = &H53
        DatiTrasmissioneSeriale(6) = &H5B
        DatiTrasmissioneSeriale(7) = &H53
        DatiTrasmissioneSeriale(8) = &H53
        DatiTrasmissioneSeriale(9) = &H53
        DatiTrasmissioneSeriale(10) = &H53
        DatiTrasmissioneSeriale(11) = &H53
        DatiTrasmissioneSeriale(12) = &H53
        DatiTrasmissioneSeriale(13) = &H53
        DatiTrasmissioneSeriale(14) = &H53
        DatiTrasmissioneSeriale(15) = &H35
        DatiTrasmissioneSeriale(16) = &H35
        CalcolaCRC()
        Temp = ValoreCRC
        Temp = &HFF And (Temp >> 8)
        H = Temp
        Temp = ValoreCRC - H * 256
        Temp = &HFF And Temp
        L = Convert.ToByte(Temp)
        DatiTrasmissioneSeriale(17) = H
        DatiTrasmissioneSeriale(18) = L
        DatiTrasmissioneSeriale(19) = &HD
        TrasmettiSerialeByte()
        TimerTimeoutSeriale.Enabled = True
    End Sub

    Private Sub RichiediStatoMotos()
        Dim Temp As Integer
        Dim Valore, L, H As Byte

        DatiTrasmissioneSeriale(0) = &H2A
        DatiTrasmissioneSeriale(1) = &H4D 'M

        'idscedadainterrogare
        DatiTrasmissioneSeriale(2) = IdSchedaDaInterrogare

        DatiTrasmissioneSeriale(3) = &H53 'S
        DatiTrasmissioneSeriale(4) = &H1
        DatiTrasmissioneSeriale(5) = &H53
        DatiTrasmissioneSeriale(6) = &H5B
        DatiTrasmissioneSeriale(7) = &H53
        DatiTrasmissioneSeriale(8) = &H53
        DatiTrasmissioneSeriale(9) = &H53
        DatiTrasmissioneSeriale(10) = &H53
        DatiTrasmissioneSeriale(11) = &H53
        DatiTrasmissioneSeriale(12) = &H53
        DatiTrasmissioneSeriale(13) = &H53
        DatiTrasmissioneSeriale(14) = &H53
        DatiTrasmissioneSeriale(15) = &H35
        DatiTrasmissioneSeriale(16) = &H35
        CalcolaCRC()
        Temp = ValoreCRC
        Temp = &HFF And (Temp >> 8)
        H = Temp
        Temp = ValoreCRC - H * 256
        Temp = &HFF And Temp
        L = Convert.ToByte(Temp)
        DatiTrasmissioneSeriale(17) = H
        DatiTrasmissioneSeriale(18) = L
        DatiTrasmissioneSeriale(19) = &HD
        TrasmettiSerialeByte()
        TimerTimeoutSeriale.Enabled = True
    End Sub

    Private Sub ButtonEV1_Click(sender As Object, e As EventArgs) Handles ButtonEV1.Click
```

```
    If VettoreStatoElettrovalvole(0) = &H41 Then 'A
        SpegniEV(1)
    ElseIf VettoreStatoElettrovalvole(0) = &H53 Then 'S
        AccendiEV(1)
    End If

  End Sub

  Private Sub ButtonEV2_Click(sender As Object, e As EventArgs) Handles ButtonEV2.Click
    If VettoreStatoElettrovalvole(1) = &H41 Then 'A
        SpegniEV(2)
    ElseIf VettoreStatoElettrovalvole(1) = &H53 Then 'S
        AccendiEV(2)
    End If

  End Sub

  Private Sub ButtonEV3_Click(sender As Object, e As EventArgs) Handles ButtonEV3.Click
    If VettoreStatoElettrovalvole(2) = &H41 Then 'A
        SpegniEV(3)
    ElseIf VettoreStatoElettrovalvole(2) = &H53 Then 'S
        AccendiEV(3)
    End If

  End Sub

  Private Sub ButtonEV4_Click(sender As Object, e As EventArgs) Handles ButtonEV4.Click
    If VettoreStatoElettrovalvole(3) = &H41 Then 'A
        SpegniEV(4)
    ElseIf VettoreStatoElettrovalvole(3) = &H53 Then 'S
        AccendiEV(4)
    End If

  End Sub

  Private Sub ButtonEV5_Click(sender As Object, e As EventArgs) Handles ButtonEV5.Click
    If VettoreStatoElettrovalvole(7) = &H41 Then 'A
        SpegniEV(8)
    ElseIf VettoreStatoElettrovalvole(7) = &H53 Then 'S
        AccendiEV(8)
    End If

  End Sub

  Private Sub ButtonEV6_Click(sender As Object, e As EventArgs) Handles ButtonEV6.Click
    If VettoreStatoElettrovalvole(6) = &H41 Then 'A
        SpegniEV(7)
    ElseIf VettoreStatoElettrovalvole(6) = &H53 Then 'S
        AccendiEV(7)
    End If
  End Sub

  Private Sub ButtonEV7_Click(sender As Object, e As EventArgs) Handles ButtonEV7.Click
    If VettoreStatoElettrovalvole(6) = &H41 Then 'A
        SpegniEV(7)
    ElseIf VettoreStatoElettrovalvole(6) = &H53 Then 'S
        AccendiEV(7)
    End If
  End Sub

  Private Sub ButtonEV8_Click(sender As Object, e As EventArgs) Handles ButtonEV8.Click
    If VettoreStatoElettrovalvole(7) = &H41 Then 'A
        SpegniEV(8)
    ElseIf VettoreStatoElettrovalvole(7) = &H53 Then 'S
        AccendiEV(8)
    End If
  End Sub

  Private Sub ButtonEV9_Click(sender As Object, e As EventArgs) Handles ButtonEV9.Click
    If VettoreStatoElettrovalvole(8) = &H41 Then 'A
        SpegniEV(9)
```

```
    ElseIf VettoreStatoElettrovalvole(8) =
&H53 Then 'S
      AccendiEV(9)
    End If
  End Sub

  Private Sub ButtonEV10_Click(sender As
Object, e As EventArgs) Handles
ButtonEV10.Click
    If VettoreStatoElettrovalvole(9) =
&H41 Then 'A
      SpegniEV(10)
    ElseIf VettoreStatoElettrovalvole(9) =
&H53 Then 'S
      AccendiEV(10)
    End If
  End Sub

  Private Sub AccendiEV(ByVal EVnum)

    Dim Temp As Integer
    Dim L, H As Byte
    Dim Ciclo

    DatiTrasmissioneSeriale(0) = &H2A
    DatiTrasmissioneSeriale(1) = &H45 'E
    DatiTrasmissioneSeriale(2) = 10
    DatiTrasmissioneSeriale(3) = &H43 'C
    For Ciclo = 4 To 14
      DatiTrasmissioneSeriale(Ciclo) =
&H4E 'N
    Next
    DatiTrasmissioneSeriale(3 + EVnum) =
&H41
    CalcolaCRC()
    Temp = ValoreCRC
    Temp = &HFF And (Temp >> 8)
    H = Temp
    Temp = ValoreCRC - H * 256
    Temp = &HFF And Temp
    L = Convert.ToByte(Temp)
    DatiTrasmissioneSeriale(17) = H
    DatiTrasmissioneSeriale(18) = L
    DatiTrasmissioneSeriale(19) = &HD
    TrasmettiSerialeByte()
  End Sub

  Private Sub SpegniEV(ByVal EVnum)

    Dim Temp As Integer
    Dim L, H As Byte
    Dim Ciclo

    DatiTrasmissioneSeriale(0) = &H2A
    DatiTrasmissioneSeriale(1) = &H45 'E
    DatiTrasmissioneSeriale(2) = &HA
    DatiTrasmissioneSeriale(3) = &H43 'C
    For Ciclo = 4 To 14
      DatiTrasmissioneSeriale(Ciclo) =
&H4E 'N
    Next
    DatiTrasmissioneSeriale(3 + EVnum) =
&H53
    CalcolaCRC()
    Temp = ValoreCRC
    Temp = &HFF And (Temp >> 8)
    H = Temp
    Temp = ValoreCRC - H * 256
    Temp = &HFF And Temp
    L = Convert.ToByte(Temp)
    DatiTrasmissioneSeriale(17) = H
    DatiTrasmissioneSeriale(18) = L
    DatiTrasmissioneSeriale(19) = &HD
    TrasmettiSerialeByte()
  End Sub

  Private Sub
ButtonMultiEVAccendi_Click(sender As
Object, e As EventArgs) Handles
ButtonMultiEVAccendi.Click

    Dim Temp As Integer
    Dim L, H As Byte
    Dim Ciclo

    DatiTrasmissioneSeriale(0) = &H2A
    DatiTrasmissioneSeriale(1) = &H45 'E
    DatiTrasmissioneSeriale(2) = &HA
    DatiTrasmissioneSeriale(3) = &H43 'C
    For Ciclo = 4 To 14
      DatiTrasmissioneSeriale(Ciclo) =
&H4E 'N
    Next

    If CheckBoxEV1.Checked = True Then
      DatiTrasmissioneSeriale(4) = &H41
    End If
    If CheckBoxEV2.Checked = True Then
      DatiTrasmissioneSeriale(5) = &H41
    End If
    If CheckBoxEV3.Checked = True Then
      DatiTrasmissioneSeriale(6) = &H41
    End If
    If CheckBoxEV4.Checked = True Then
```

```
        DatiTrasmissioneSeriale(7) = &H41
    End If
    If CheckBoxEV5.Checked = True Then
        DatiTrasmissioneSeriale(8) = &H41
    End If
    If CheckBoxEV6.Checked = True Then
        DatiTrasmissioneSeriale(9) = &H41
    End If
    If CheckBoxEV7.Checked = True Then
        DatiTrasmissioneSeriale(10) = &H41
    End If
    If CheckBoxEV8.Checked = True Then
        DatiTrasmissioneSeriale(11) = &H41
    End If
    If CheckBoxEV9.Checked = True Then
        DatiTrasmissioneSeriale(12) = &H41
    End If
    If CheckBoxEV10.Checked = True
Then
        DatiTrasmissioneSeriale(13) = &H41
    End If

    CalcolaCRC()
    Temp = ValoreCRC
    Temp = &HFF And (Temp >> 8)
    H = Temp
    Temp = ValoreCRC - H * 256
    Temp = &HFF And Temp
    L = Convert.ToByte(Temp)
    DatiTrasmissioneSeriale(17) = H
    DatiTrasmissioneSeriale(18) = L
    DatiTrasmissioneSeriale(19) = &HD
    TrasmettiSerialeByte()
  End Sub

  Private Sub
ButtonMultiEVSpegni_Click(sender As
Object, e As EventArgs) Handles
ButtonMultiEVSpegni.Click

    Dim Temp As Integer
    Dim L, H As Byte
    Dim Ciclo

    DatiTrasmissioneSeriale(0) = &H2A
    DatiTrasmissioneSeriale(1) = &H45 'E
    DatiTrasmissioneSeriale(2) = &HA
    DatiTrasmissioneSeriale(3) = &H43 'C
    For Ciclo = 4 To 14
        DatiTrasmissioneSeriale(Ciclo) =
&H4E 'N
    Next
    If CheckBoxEV1.Checked = True Then
        DatiTrasmissioneSeriale(4) = &H53
    End If
    If CheckBoxEV2.Checked = True Then
        DatiTrasmissioneSeriale(5) = &H53
    End If
    If CheckBoxEV3.Checked = True Then
        DatiTrasmissioneSeriale(6) = &H53
    End If
    If CheckBoxEV4.Checked = True Then
        DatiTrasmissioneSeriale(7) = &H53
    End If
    If CheckBoxEV5.Checked = True Then
        DatiTrasmissioneSeriale(8) = &H53
    End If
    If CheckBoxEV6.Checked = True Then
        DatiTrasmissioneSeriale(9) = &H53
    End If
    If CheckBoxEV7.Checked = True Then
        DatiTrasmissioneSeriale(10) = &H53
    End If
    If CheckBoxEV8.Checked = True Then
        DatiTrasmissioneSeriale(11) = &H53
    End If
    If CheckBoxEV9.Checked = True Then
        DatiTrasmissioneSeriale(12) = &H53
    End If
    If CheckBoxEV10.Checked = True
Then
        DatiTrasmissioneSeriale(13) = &H53
    End If

    CalcolaCRC()
    Temp = ValoreCRC
    Temp = &HFF And (Temp >> 8)
    H = Temp
    Temp = ValoreCRC - H * 256
    Temp = &HFF And Temp
    L = Convert.ToByte(Temp)
    DatiTrasmissioneSeriale(17) = H
    DatiTrasmissioneSeriale(18) = L
    DatiTrasmissioneSeriale(19) = &HD
    TrasmettiSerialeByte()
  End Sub

  Private Sub
ButtonAvviaTimerEV_Click(sender As
Object, e As EventArgs) Handles
ButtonAvviaTimerEV.Click
    Dim Temp As Integer
    Dim L, H As Byte
    Dim Ciclo
```

```
        DatiTrasmissioneSeriale(0) = &H2A
        DatiTrasmissioneSeriale(1) = &H45 'E
        DatiTrasmissioneSeriale(2) = &HA
        DatiTrasmissioneSeriale(3) = &H54 'T
        For Ciclo = 4 To 14
            DatiTrasmissioneSeriale(Ciclo) =
&H4E 'N
        Next

        DatiTrasmissioneSeriale(4) =
ComboBoxEVtempo.SelectedIndex + 1
        If
ComboBoxStatoIniziale.SelectedIndex = 0
Then
            DatiTrasmissioneSeriale(5) = &H41
'A
        Else
            DatiTrasmissioneSeriale(5) = &H53
'S
        End If

        Temp =
Convert.ToInt32(TextBoxTempo.Text)
        H = &HFF And (Temp >> 8)
        Temp = &HFF And Temp
        L = Convert.ToByte(Temp)
        DatiTrasmissioneSeriale(6) = H
        DatiTrasmissioneSeriale(7) = L

        CalcolaCRC()
        Temp = ValoreCRC
        Temp = &HFF And (Temp >> 8)
        H = Temp
        Temp = ValoreCRC - H * 256
        Temp = &HFF And Temp
        L = Convert.ToByte(Temp)
        DatiTrasmissioneSeriale(17) = H
        DatiTrasmissioneSeriale(18) = L
        DatiTrasmissioneSeriale(19) = &HD
        TrasmettiSerialeByte()
    End Sub

    Private Sub
ButtonAvviaMovim_Click(sender As
Object, e As EventArgs) Handles
ButtonAvviaMovim.Click

        Dim Temp As Integer
        Dim L, H As Byte
        Dim Ciclo As Integer

        For Ciclo = 0 To 19
            DatiTrasmissioneSeriale(Ciclo) =
&H0
        Next
        DatiTrasmissioneSeriale(0) = &H2A
        DatiTrasmissioneSeriale(1) = &H4D 'M
        DatiTrasmissioneSeriale(2) =
IdSchedaDaInterrogare
        'Comando movimentazione
        DatiTrasmissioneSeriale(3) = &H44  'D

        'Direzione movimentazione
        If RadioButtonSuzione.Checked = True
Then
            DatiTrasmissioneSeriale(4) = &H53
'S
        End If
        If RadioButtonIniezione.Checked =
True Then
            DatiTrasmissioneSeriale(4) = &H49
'I
        End If

        'Parametro quantità movimentazione
        Temp =
Convert.ToInt32(TextBoxQuantiaMoviment
azione.Text)
        H = &HFF And (Temp >> 8)
        Temp = &HFF And Temp
        L = Convert.ToByte(Temp)
        DatiTrasmissioneSeriale(5) = H
        DatiTrasmissioneSeriale(6) = L

        'Parametro velocità movimentazione
        Temp =
Convert.ToInt32(TextBoxMovimentazioneV
elocita.Text)
        H = &HFF And (Temp >> 8)
        Temp = &HFF And Temp
        L = Convert.ToByte(Temp)
        DatiTrasmissioneSeriale(7) = H
        DatiTrasmissioneSeriale(8) = L

        'Calcolo CRC e trasmissione
        CalcolaCRC()
        Temp = ValoreCRC
        Temp = &HFF And (Temp >> 8)
        H = Temp
        Temp = ValoreCRC - H * 256
        Temp = &HFF And Temp
        L = Convert.ToByte(Temp)
        DatiTrasmissioneSeriale(17) = H
        DatiTrasmissioneSeriale(18) = L
```

```
        DatiTrasmissioneSeriale(19) = &HD
        TrasmettiSerialeByte()
    End Sub

    Private Sub ButtonArrestaMovim_Click(sender As Object, e As EventArgs) Handles ButtonArrestaMovim.Click
        Dim Temp As Integer
        Dim L, H As Byte
        Dim Ciclo As Integer

        For Ciclo = 0 To 19
            DatiTrasmissioneSeriale(Ciclo) = &H0
        Next
        DatiTrasmissioneSeriale(0) = &H2A
        DatiTrasmissioneSeriale(1) = &H4D 'M
        DatiTrasmissioneSeriale(2) = IdSchedaDaInterrogare
        'Comando movimentazione
        DatiTrasmissioneSeriale(3) = &H46 'F

        'Calcolo CRC e trasmissione
        CalcolaCRC()
        Temp = ValoreCRC
        Temp = &HFF And (Temp >> 8)
        H = Temp
        Temp = ValoreCRC - H * 256
        Temp = &HFF And Temp
        L = Convert.ToByte(Temp)
        DatiTrasmissioneSeriale(17) = H
        DatiTrasmissioneSeriale(18) = L
        DatiTrasmissioneSeriale(19) = &HD
        TrasmettiSerialeByte()
    End Sub

    Private Sub ButtonSvuotaMS_Click(sender As Object, e As EventArgs) Handles ButtonSvuotaMS.Click
        Dim Temp As Integer
        Dim L, H As Byte
        Dim Ciclo As Integer

        For Ciclo = 0 To 19
            DatiTrasmissioneSeriale(Ciclo) = &H0
        Next
        DatiTrasmissioneSeriale(0) = &H2A
        DatiTrasmissioneSeriale(1) = &H4D 'M
        DatiTrasmissioneSeriale(2) = IdSchedaDaInterrogare
        'Comando movimentazione
        DatiTrasmissioneSeriale(3) = &H56 'V

        'Parametro velocità movimentazione
        Temp = Convert.ToInt32(TextBoxMovimentazioneVelocita.Text)
        H = &HFF And (Temp >> 8)
        Temp = &HFF And Temp
        L = Convert.ToByte(Temp)
        DatiTrasmissioneSeriale(4) = H
        DatiTrasmissioneSeriale(5) = L

        'Calcolo CRC e trasmissione
        CalcolaCRC()
        Temp = ValoreCRC
        Temp = &HFF And (Temp >> 8)
        H = Temp
        Temp = ValoreCRC - H * 256
        Temp = &HFF And Temp
        L = Convert.ToByte(Temp)
        DatiTrasmissioneSeriale(17) = H
        DatiTrasmissioneSeriale(18) = L
        DatiTrasmissioneSeriale(19) = &HD
        TrasmettiSerialeByte()
    End Sub

    Private Sub ButtonRiempiMS_Click(sender As Object, e As EventArgs) Handles ButtonRiempiMS.Click
        Dim Temp As Integer
        Dim L, H As Byte
        Dim Ciclo As Integer

        For Ciclo = 0 To 19
            DatiTrasmissioneSeriale(Ciclo) = &H0
        Next
        DatiTrasmissioneSeriale(0) = &H2A
        DatiTrasmissioneSeriale(1) = &H4D 'M
        DatiTrasmissioneSeriale(2) = IdSchedaDaInterrogare
        'Comando movimentazione
        DatiTrasmissioneSeriale(3) = &H41 'A

        'Parametro velocità movimentazione
        Temp = Convert.ToInt32(TextBoxMovimentazioneVelocita.Text)
```

```
        H = &HFF And (Temp >> 8)
        Temp = &HFF And Temp
        L = Convert.ToByte(Temp)
        DatiTrasmissioneSeriale(4) = H
        DatiTrasmissioneSeriale(5) = L

        'Calcolo CRC e trasmissione
        CalcolaCRC()
        Temp = ValoreCRC
        Temp = &HFF And (Temp >> 8)
        H = Temp
        Temp = ValoreCRC - H * 256
        Temp = &HFF And Temp
        L = Convert.ToByte(Temp)
        DatiTrasmissioneSeriale(17) = H
        DatiTrasmissioneSeriale(18) = L
        DatiTrasmissioneSeriale(19) = &HD
        TrasmettiSerialeByte()
    End Sub

    Private Sub ButtonOffsetMS_Click(sender
As Object, e As EventArgs) Handles
ButtonOffsetMS.Click
        Dim Temp As Integer
        Dim L, H As Byte
        Dim Ciclo As Integer

        For Ciclo = 0 To 19
            DatiTrasmissioneSeriale(Ciclo) =
&H0
        Next
        DatiTrasmissioneSeriale(0) = &H2A
        DatiTrasmissioneSeriale(1) = &H4D 'M
        DatiTrasmissioneSeriale(2) =
IdSchedaDaInterrogare
        'Comando movimentazione
        DatiTrasmissioneSeriale(3) = &H5A  'Z

        'Parametro velocità movimentazione
        Temp =
Convert.ToInt32(TextBoxOffsetMS.Text)
        H = &HFF And (Temp >> 8)
        Temp = &HFF And Temp
        L = Convert.ToByte(Temp)
        DatiTrasmissioneSeriale(4) = H
        DatiTrasmissioneSeriale(5) = L

        'Calcolo CRC e trasmissione
        CalcolaCRC()
        Temp = ValoreCRC
        Temp = &HFF And (Temp >> 8)
        H = Temp
        Temp = ValoreCRC - H * 256
        Temp = &HFF And Temp
        L = Convert.ToByte(Temp)
        DatiTrasmissioneSeriale(17) = H
        DatiTrasmissioneSeriale(18) = L
        DatiTrasmissioneSeriale(19) = &HD
        TrasmettiSerialeByte()
    End Sub

    Private Sub ScriviTextBoxByte(ByRef
TextBox, ByVal Bytevalue)
        TextBox.Text =
Convert.ToString(Bytevalue, 16)
    End Sub
    Private Delegate Sub
DelegateScriviTextBoxByte(ByRef
TextBox, ByVal ByteValue)
    Private
MethodDelegateScriviTextBoxByte As New
DelegateScriviTextBoxByte(AddressOf
ScriviTextBoxByte)

    Private Sub ScriviTextBoxHEX(ByRef
TextBox, ByVal Stringa)
        TextBox.Text = Chr(Stringa)
    End Sub
    Private Delegate Sub
DelegateScriviTextBoxHEX(ByRef
TextBox, ByVal Stringa)
    Private
MethodDelegateScriviTextBoxHEX As New
DelegateScriviTextBoxHEX(AddressOf
ScriviTextBoxHEX)

    Private Sub ScriviTextBoxString(ByRef
TextBox, ByVal Stringa)
        TextBox.Text = Stringa
    End Sub
    Private Delegate Sub
DelegateScriviTextBoxString(ByRef
TextBox, ByVal Stringa)
    Private
MethodDelegateScriviTextBoxString As
New
DelegateScriviTextBoxString(AddressOf
ScriviTextBoxString)

    Private Sub ScriviButtonString(ByRef
Button, ByVal Stringa)
        Button.Text = Stringa
    End Sub
```

```
    Private Delegate Sub DelegateScriviButtonString(ByRef Button, ByVal Stringa)
    Private MethodDelegateScriviButtonString As New DelegateScriviButtonString(AddressOf ScriviButtonString)

    Private Sub ImpostaColoreStatusLight(ByRef StatusLight, ByRef Colore)
        StatusLight.FillColor = Colore
    End Sub
    Private Delegate Sub DelegateImpostaColoreStatusLight(ByRef TextBox, ByRef Colore)
    Private MethodDelegateImpostaColoreStatusLight As New DelegateImpostaColoreStatusLight(AddressOf ImpostaColoreStatusLight)

    Private Sub ComboBoxIndexSelect(ByRef Control As ComboBox, ByVal Index As Byte)
        Control.SelectedIndex = Index
    End Sub
    Private Delegate Sub DelegateComboBoxIndexSelect(ByRef Control As ComboBox, ByVal Index As Byte)
    Private MethodDelegateComboBoxIndexSelect As New DelegateComboBoxIndexSelect(AddressOf ComboBoxIndexSelect)

    Private Sub ModificaVisibilitaGroupbox(ByRef Control As GroupBox, ByVal Value As Boolean)
        Control.Enabled = Value
    End Sub
    Private Delegate Sub DelegateModificaVisibilitaGroupbox(ByRef Control As GroupBox, ByVal Value As Boolean)
    Private MethodDelegateModificaVisibilitaGroupbox As New DelegateModificaVisibilitaGroupbox(AddressOf ModificaVisibilitaGroupbox)

    Private Sub CheckBoxChecked(ByRef Control As CheckBox, ByVal Value As Boolean)
        Control.Checked = Value
    End Sub
    Private Delegate Sub DelegateCheckBoxChecked(ByRef Control As CheckBox, ByVal Value As Boolean)
    Private MethodDelegateCheckBoxChecked As New DelegateCheckBoxChecked(AddressOf CheckBoxChecked)

    Private Sub ScriviLabel(ByRef Control As Label, ByVal Value As String)
        Control.Text = Value
    End Sub
    Private Delegate Sub DelegateScriviLabel(ByRef Control As Label, ByVal Value As String)
    Private MethodDelegateScriviLabel As New DelegateScriviLabel(AddressOf ScriviLabel)

    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
        Dim Temp As Integer
        Dim L, H As Byte
        Dim Ciclo As Integer

        For Ciclo = 0 To 19
            DatiTrasmissioneSeriale(Ciclo) = &H0
        Next
        DatiTrasmissioneSeriale(0) = &H2A
        DatiTrasmissioneSeriale(1) = &H4D 'M
        DatiTrasmissioneSeriale(2) = 1
        'Comando movimentazione
        DatiTrasmissioneSeriale(3) = &H44  'D

        'Direzione movimentazione
        If RadioButtonSuzione.Checked = True Then
            DatiTrasmissioneSeriale(4) = &H53 'S
        End If
        If RadioButtonIniezione.Checked = True Then
            DatiTrasmissioneSeriale(4) = &H49 'I
        End If
```

```
        'Parametro quantità movimentazione
        Temp = Convert.ToInt32(TextBoxQuantiaMovimentazione.Text)
        H = &HFF And (Temp >> 8)
        Temp = &HFF And Temp
        L = Convert.ToByte(Temp)
        DatiTrasmissioneSeriale(5) = H
        DatiTrasmissioneSeriale(6) = L

        'Parametro velocità movimentazione
        Temp = Convert.ToInt32(TextBoxMovimentazioneVelocita.Text)
        H = &HFF And (Temp >> 8)
        Temp = &HFF And Temp
        L = Convert.ToByte(Temp)
        DatiTrasmissioneSeriale(7) = H
        DatiTrasmissioneSeriale(8) = L

        'Calcolo CRC e trasmissione
        CalcolaCRC()
        Temp = ValoreCRC
        Temp = &HFF And (Temp >> 8)
        H = Temp
        Temp = ValoreCRC - H * 256
        Temp = &HFF And Temp
        L = Convert.ToByte(Temp)
        DatiTrasmissioneSeriale(17) = H
        DatiTrasmissioneSeriale(18) = L
        DatiTrasmissioneSeriale(19) = &HD
        TrasmettiSerialeByte()
    End Sub

    Private Sub Button2_Click(sender As Object, e As EventArgs) Handles Button2.Click
        Dim Temp As Integer
        Dim L, H As Byte
        Dim Ciclo As Integer

        For Ciclo = 0 To 19
            DatiTrasmissioneSeriale(Ciclo) = &H0
        Next
        DatiTrasmissioneSeriale(0) = &H2A
        DatiTrasmissioneSeriale(1) = &H4D 'M
        DatiTrasmissioneSeriale(2) = 2
        'Comando movimentazione
        DatiTrasmissioneSeriale(3) = &H44  'D
        'Direzione movimentazione
        If RadioButtonSuzione2.Checked = True Then
            DatiTrasmissioneSeriale(4) = &H53 'S
        End If
        If RadioButtonIniezione2.Checked = True Then
            DatiTrasmissioneSeriale(4) = &H49 'I
        End If

        'Parametro quantità movimentazione
        Temp = Convert.ToInt32(TextBoxQuantiaMovimentazione2.Text)
        H = &HFF And (Temp >> 8)
        Temp = &HFF And Temp
        L = Convert.ToByte(Temp)
        DatiTrasmissioneSeriale(5) = H
        DatiTrasmissioneSeriale(6) = L

        'Parametro velocità movimentazione
        Temp = Convert.ToInt32(TextBoxMovimentazioneVelocita2.Text)
        H = &HFF And (Temp >> 8)
        Temp = &HFF And Temp
        L = Convert.ToByte(Temp)
        DatiTrasmissioneSeriale(7) = H
        DatiTrasmissioneSeriale(8) = L

        'Calcolo CRC e trasmissione
        CalcolaCRC()
        Temp = ValoreCRC
        Temp = &HFF And (Temp >> 8)
        H = Temp
        Temp = ValoreCRC - H * 256
        Temp = &HFF And Temp
        L = Convert.ToByte(Temp)
        DatiTrasmissioneSeriale(17) = H
        DatiTrasmissioneSeriale(18) = L
        DatiTrasmissioneSeriale(19) = &HD
        TrasmettiSerialeByte()
    End Sub

    Private Sub Button3_Click(sender As Object, e As EventArgs)
        Dim Temp As Integer
        Dim L, H As Byte
        Dim Ciclo As Integer
```

```
      For Ciclo = 0 To 19
         DatiTrasmissioneSeriale(Ciclo)      =
&H0
      Next
      DatiTrasmissioneSeriale(0) = &H2A
      DatiTrasmissioneSeriale(1) = &H4D 'M
      DatiTrasmissioneSeriale(2) = 3
      'Comando movimentazione
      DatiTrasmissioneSeriale(3) = &H44   'D

      'Direzione movimentazione
      If   RadioButtonSuzione3.Checked   =
True Then
         DatiTrasmissioneSeriale(4)  =  &H53
'S
      End If
      If   RadioButtonIniezione3.Checked   =
True Then
         DatiTrasmissioneSeriale(4)  =  &H49
'I
      End If
      If  RadioButtonCiclo.Checked  =  True
Then
         AvviaCiclo = 1

      Else

         'Parametro quantità movimentazione
         Temp                              =
Convert.ToInt32(TextBoxQuantiaMoviment
azione3.Text)
         H = &HFF And (Temp >> 8)
         Temp = &HFF And Temp
         L = Convert.ToByte(Temp)
         DatiTrasmissioneSeriale(5) = H
         DatiTrasmissioneSeriale(6) = L

         'Parametro velocità movimentazione
         Temp                              =
Convert.ToInt32(TextBoxMovimentazioneV
elocita3.Text)
         H = &HFF And (Temp >> 8)
         Temp = &HFF And Temp
         L = Convert.ToByte(Temp)
         DatiTrasmissioneSeriale(7) = H
         DatiTrasmissioneSeriale(8) = L

         'Calcolo CRC e trasmissione
         CalcolaCRC()
         Temp = ValoreCRC
         Temp = &HFF And (Temp >> 8)
         H = Temp
         Temp = ValoreCRC - H * 256
         Temp = &HFF And Temp
         L = Convert.ToByte(Temp)
         DatiTrasmissioneSeriale(17) = H
         DatiTrasmissioneSeriale(18) = L
         DatiTrasmissioneSeriale(19) = &HD
         TrasmettiSerialeByte()

      End If
   End Sub

   Private                                Sub
RadioButtonCiclo_CheckedChanged(sender
As Object, e As EventArgs) Handles
RadioButtonCiclo.CheckedChanged
      If  RadioButtonCiclo.Checked  =  True
Then
         Timer1.Enabled = True
         FaseCiclo = 0
         NumeroCicli = 0
         IdSchedaDaInterrogare = 3
         TimerRichiediStato.Enabled = True

      Else
         TimerRichiediStato.Enabled = False
      End If

   End Sub

   Private Sub Timer1_Tick(sender As
Object, e As EventArgs) Handles
Timer1.Tick
      Dim Temp As Integer
      Dim L, H As Byte
      Dim Ciclo As Integer

      If AvviaCiclo = 1 Then
         If FaseCiclo = 0 Then
            For Ciclo = 0 To 19
               DatiTrasmissioneSeriale(Ciclo)
= &H0
            Next
            DatiTrasmissioneSeriale(0)      =
&H2A
            DatiTrasmissioneSeriale(1)      =
&H4D 'M
            DatiTrasmissioneSeriale(2) = 3
            'Comando movimentazione
            DatiTrasmissioneSeriale(3)      =
&H56   'V
```

```
            'Parametro velocità
movimentazione
            Temp =
Convert.ToInt32(TextBoxMovimentazioneV
elocita3.Text)
            H = &HFF And (Temp >> 8)
            Temp = &HFF And Temp
            L = Convert.ToByte(Temp)
            DatiTrasmissioneSeriale(4) = H
            DatiTrasmissioneSeriale(5) = L

            'Calcolo CRC e trasmissione
            CalcolaCRC()
            Temp = ValoreCRC
            Temp = &HFF And (Temp >> 8)
            H = Temp
            Temp = ValoreCRC - H * 256
            Temp = &HFF And Temp
            L = Convert.ToByte(Temp)
            DatiTrasmissioneSeriale(17) = H
            DatiTrasmissioneSeriale(18) = L
            DatiTrasmissioneSeriale(19) =
&HD
            TrasmettiSerialeByte()

            FaseCiclo = 1

            Return
        End If
        'If FaseCiclo = 0 And Capacitamotos3
= 0 Then
        '   FaseCiclo = 1
        'End If
        If NumeroCicli <
Convert.ToInt32(TextBoxCicli.Text) Then

            If FaseCiclo = 1 And
StatoMovMotos3 = &H46 Then

                For Ciclo = 0 To 19

DatiTrasmissioneSeriale(Ciclo) = &H0
                Next
                DatiTrasmissioneSeriale(0) =
&H2A
                DatiTrasmissioneSeriale(1) =
&H4D 'M
                DatiTrasmissioneSeriale(2) = 3
                'Comando movimentazione
                DatiTrasmissioneSeriale(3) =
&H44  'D
                'Direzione movimentazione

                DatiTrasmissioneSeriale(4) =
&H53  'S

                'Parametro quantità
movimentazione
                Temp =
Convert.ToInt32(TextBoxQuantiaMoviment
azione3.Text)
                H = &HFF And (Temp >> 8)
                Temp = &HFF And Temp
                L = Convert.ToByte(Temp)
                DatiTrasmissioneSeriale(5) = H
                DatiTrasmissioneSeriale(6) = L

                'Parametro velocità
movimentazione
                Temp =
Convert.ToInt32(TextBoxMovimentazioneV
elocita3.Text)
                H = &HFF And (Temp >> 8)
                Temp = &HFF And Temp
                L = Convert.ToByte(Temp)
                DatiTrasmissioneSeriale(7) = H
                DatiTrasmissioneSeriale(8) = L

                'Calcolo CRC e trasmissione
                CalcolaCRC()
                Temp = ValoreCRC
                Temp = &HFF And (Temp >> 8)
                H = Temp
                Temp = ValoreCRC - H * 256
                Temp = &HFF And Temp
                L = Convert.ToByte(Temp)
                DatiTrasmissioneSeriale(17) =
H
                DatiTrasmissioneSeriale(18) = L
                DatiTrasmissioneSeriale(19) =
&HD
                TrasmettiSerialeByte()

                FaseCiclo = 2

                Return


            End If
            'If FaseCiclo = 1 And
Capacitamotos3 =
Convert.ToInt32(TextBoxQuantiaMoviment
azione3.Text) Then
```

```
            'If FaseCiclo = 1 Then
            '    FaseCiclo = 2
            'End If
            If FaseCiclo = 2 And
StatoMovMotos3 = &H46 Then
                For Ciclo = 0 To 19

DatiTrasmissioneSeriale(Ciclo) = &H0
                Next
                DatiTrasmissioneSeriale(0)    =
&H2A
                DatiTrasmissioneSeriale(1)    =
&H4D 'M
                DatiTrasmissioneSeriale(2) = 3
                'Comando movimentazione
                DatiTrasmissioneSeriale(3)    =
&H44   'D

                'Direzione movimentazione

                DatiTrasmissioneSeriale(4)    =
&H49   'I



                'Parametro quantità
movimentazione
                Temp =
Convert.ToInt32(TextBoxQuantiaMoviment
azione3.Text)
                H = &HFF And (Temp >> 8)
                Temp = &HFF And Temp
                L = Convert.ToByte(Temp)
                DatiTrasmissioneSeriale(5) = H
                DatiTrasmissioneSeriale(6) = L

                'Parametro velocità
movimentazione
                Temp =
Convert.ToInt32(TextBoxMovimentazioneV
elocita3.Text)
                H = &HFF And (Temp >> 8)
                Temp = &HFF And Temp
                L = Convert.ToByte(Temp)
                DatiTrasmissioneSeriale(7) = H
                DatiTrasmissioneSeriale(8) = L

                'Calcolo CRC e trasmissione
                CalcolaCRC()
                Temp = ValoreCRC
                Temp = &HFF And (Temp >> 8)
                H = Temp
                Temp = ValoreCRC - H * 256
                Temp = &HFF And Temp
                L = Convert.ToByte(Temp)
                DatiTrasmissioneSeriale(17)   =
H
                DatiTrasmissioneSeriale(18) = L
                DatiTrasmissioneSeriale(19)   =
&HD
                TrasmettiSerialeByte()
                NumeroCicli = NumeroCicli + 1

Me.Invoke(MethodDelegateScriviTextBoxS
tring, TextBoxNumeroCicli, NumeroCicli)
                FaseCiclo = 1



            End If
        Else
            Timer1.Enabled = False
        End If


        'controllo che la siringa abbia finito
numero microlitri richiesti
        ' se controllo precedente è vero allora
fase ciclo =2
        'altrimenti esci



    End If



End Sub

'Abilita e disabilita menu laterale
Private Sub Button4_Click(sender As
Object, e As EventArgs) Handles
Button4.Click

    If ControlloGMotosiringa = 0 Then
        GroupBoxMotosiringa.Visible    =
False
        ControlloGMotosiringa = 1
    Else
        GroupBoxMotosiringa.Visible    =
True
```

```
        ControlloGMotosiringa = 0
    End If

End Sub

Private Sub Button3_Click_1(sender As Object, e As EventArgs) Handles Button3.Click
    Dim Temp As Integer
    Dim L, H As Byte
    Dim Ciclo As Integer

    For Ciclo = 0 To 19
        DatiTrasmissioneSeriale(Ciclo) = &H0
    Next
    DatiTrasmissioneSeriale(0) = &H2A
    DatiTrasmissioneSeriale(1) = &H4D 'M
    DatiTrasmissioneSeriale(2) = 3
    'Comando movimentazione
    DatiTrasmissioneSeriale(3) = &H44  'D

    'Direzione movimentazione
    If RadioButtonSuzione3.Checked = True Then
        DatiTrasmissioneSeriale(4) = &H53 'S
    End If
    If RadioButtonIniezione3.Checked = True Then
        DatiTrasmissioneSeriale(4) = &H49 'I
    End If
    If RadioButtonCiclo.Checked = True Then
        AvviaCiclo = 1

    Else

        'Parametro quantità movimentazione
        Temp = Convert.ToInt32(TextBoxQuantiaMovimentazione3.Text)
        H = &HFF And (Temp >> 8)
        Temp = &HFF And Temp
        L = Convert.ToByte(Temp)
        DatiTrasmissioneSeriale(5) = H
        DatiTrasmissioneSeriale(6) = L

        'Parametro velocità movimentazione
        Temp = Convert.ToInt32(TextBoxMovimentazioneVelocita3.Text)
        H = &HFF And (Temp >> 8)
        Temp = &HFF And Temp
        L = Convert.ToByte(Temp)
        DatiTrasmissioneSeriale(7) = H
        DatiTrasmissioneSeriale(8) = L

        'Calcolo CRC e trasmissione
        CalcolaCRC()
        Temp = ValoreCRC
        Temp = &HFF And (Temp >> 8)
        H = Temp
        Temp = ValoreCRC - H * 256
        Temp = &HFF And Temp
        L = Convert.ToByte(Temp)
        DatiTrasmissioneSeriale(17) = H
        DatiTrasmissioneSeriale(18) = L
        DatiTrasmissioneSeriale(19) = &HD
        TrasmettiSerialeByte()

    End If
End Sub


End Class
```

APPENDIX B

```
// definisci tutti i valori tramite ArrivatoComandoUSB : numero missioni, numero misure...

#include <P24FJ256DA210.h>
#include "DS1338.h"
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <time.h>
#include <math.h>

#include "crc.h"
#include "Compiler.h"
#include "Fsconfig.h"
#include "FSDefs.h"
#include "FSIO.h"
#include "GenericTypeDefs.h"
#include "HardwareProfile.h"
#include "SD-SPI.h"


_CONFIG3
        (
   WPFP_WPFP255 &     // Write Protection Flash Page Segment Boundary (Highest Page (same as page 85))
   SOSCSEL_EC &       // Secondary Oscillator Power Mode Select (Secondary oscillator is in Default (high drive strength) Oscillator mode)
   WUTSEL_LEG &       // Voltage Regulator Wake-up Time Select (Default regulator start-up time is used)
   ALTPMP_ALPMPDIS &  // Alternate PMP Pin Mapping (EPMP pins are in default location mode)
   WPDIS_WPDIS &      // Segment Write Protection Disable (Segmented code protection is disabled)
   WPCFG_WPCFGDIS &   // Write Protect Configuration Page Select (Last page (at the top of program memory) and Flash Configuration Words are not write-protected)
   WPEND_WPENDMEM     // Segment Write Protection End Page Select (Protected code segment upper boundary is at the last page of program memory; the lower boundary is the code page specified by WPFP)
        ) ;
_CONFIG2
        (
   POSCMOD_HS &       // Primary Oscillator Select
   IOL1WAY_ON &       // IOLOCK One-Way Set Enable (The IOLOCK bit (OSCCON<6>) can be set once, provided the unlock sequence has been completed. Once set, the Peripheral Pin Select registers cannot be written to a second time.)
   OSCIOFNC_OFF &     // OSCO Pin Configuration (OSCO/CLKO/RC15 functions as CLKO (FOSC/2))
   FCKSM_CSECMD &     // Clock Switching Enabled, Fail-Safe Clock Monitor (Clock switching and Fail-Safe Clock Monitor are disabled)
   FNOSC_PRI &        // Initial Oscillator Select
   PLL96MHZ_OFF &     // 96MHz PLL Startup Select (96 MHz PLL is enabled automatically on start-up)
   PLLDIV_NODIV &     // 96MHz PLL Prescaler Select (Oscillator input is divided by 12 (48 MHz input))
   IESO_ON
        // Internal External Switchover (IESO mode (Two-Speed Start-up) is enabled)
        ) ;
_CONFIG1
        (
   WDTPS_PS32768 &    // Watchdog Timer Postscaler (1:32,768)
```

```
    FWPSA_PR128 &       // WDT Prescaler
        (Prescaler ratio of 1:128)
    WINDIS_OFF &        // Windowed WDT
        (Standard Watchdog Timer
        enabled,(Windowed-mode is
        disabled))
    FWDTEN_OFF &        // Watchdog Timer
        (Watchdog Timer is enabled)
    ICS_PGx1 &          // Emulator Pin
        Placement Select bits (Emulator
        functions are shared with
        PGEC1/PGED1)
    GWRP_OFF &          // General Segment
        Write Protect (Writes to program
        memory are allowed)
    GCP_OFF &           // General Segment
        Code Protect (Code protection is
        disabled)
    JTAGEN_OFF          // JTAG Port Enable
        (JTAG port is enabled)
        ) ;

// Definizione delle porte di ingresso ed uscita
#define                 LEDVERDE
        LATGbits.LATG12
#define                 LEDGIALLO
        LATGbits.LATG14
#define LEDROSSO        LATEbits.LATE1
#define LEDBLU          LATEbits.LATE0
#define                 OPENDRAIN1
        LATEbits.LATE4
#define                 OPENDRAIN2
        LATEbits.LATE3
#define                 OPENDRAIN3
        LATEbits.LATE2
#define                 OPENDRAIN4
        LATGbits.LATG13
#define ONTEMP          LATEbits.LATE8
#define ON_12V          LATAbits.LATA9
#define ON_5V           LATBbits.LATB3
#define                 ON_ANALOG
        LATAbits.LATA0
#define                 ON_RASPBERRY
        LATAbits.LATA10
#define AD_CS           LATCbits.LATC2
#define AD_SHDN         LATCbits.LATC1
#define AD_BUSY         PORTCbits.RC3
#define                 OPTO_OUT1
        LATGbits.LATG15
#define                 OPTO_OUT2
        LATEbits.LATE5
#define OPTO_IN1        PORTEbits.RE6
#define OPTO_IN2        PORTEbits.RE7

#define FCY     7372800L //define your
        instruction frequency, FCY =
        FOSC/2

#define CYCLES_PER_MS ((unsigned
        long)(FCY * 0.001)) //instruction
        cycles per millisecond
#define CYCLES_PER_US ((unsigned
        long)(FCY * 0.000001))
        //instruction cycles per microsecond
#define                 DELAY_MS(ms)
        __delay32(CYCLES_PER_MS *
        ((unsigned long) ms)); //__delay32
        is provided by the compiler, delay
        some # of milliseconds
#define                 DELAY_US(us)
        __delay32(CYCLES_PER_US *
        ((unsigned long) us)); //delay some
        number of microseconds

/* DEFINIZIONE INDIRIZZI      */
const int sb1 = 0x01;   // indirizzo siringa
        bassa pressione 1  ulmax 10000
const int sb2 = 0x02;   // indirizzo siringa
        bassa pressione 2  ulmax  5000
const int sb3 = 0x03;   // indirizzo siringa
        bassa pressione 3  ulmax 10000
const int sa1 = 0x05;  // indirizzo siringa alta
        pressione 1  ulmax 10000  // finale
        sarà 20000
const int indev = 0x10; // indirizzo scheda
        elettrovalvole
/* FINE DEFINIZIONE INDIRIZZI  */

/* DEFINIZIONE SPOSTAMENTI ul      */
const int ulsb1 = 200;   // microlitri siringa
        bassa pressione 1  97
const int ulsb2 = 100;   // microlitri siringa
        bassa pressione 2  48
const int ulsb3 = 0;   // microlitri siringa bassa
        pressione 3 intot/suztot
const int ulsa1ad = 17000; // microlitri siringa
        alta  pressione 1: aspirazione e
        depressurizzazione  /3
const int ulsa1bianco = 10000; // microlitri
        siringa alta  pressione 1: bianco
const int ulsa1r = 10000; // microlitri siringa
        alta  pressione 1 risciacquo: ulatt-
        ulsa1r  /3
const int ulsa1d = 4885; // microlitri siringa
```

```
        alta pressione 1 dosaggio e
        miscelazione /3
const int ulsb3d = 4000; // microlitri siringa
        bassa pressione 3 dosaggio e
        miscelazione /3
const int ulsovrappressione = 50 ;// microlitri
        siringa bassa pressione 3
        sovrappressione per misura
/* FINE DEFINIZIONE SPOSTAMENTI ul
        */

/* DEFINIZIONE VELOCITA ul/s        */
const int ulssb1 = 25; // velocita siringa bassa
        pressione 1
const int ulssb2 = 25; // velocita siringa bassa
        pressione 2
const int ulssb3 = 200; // velocita siringa
        bassa pressione 3
const int ulssa1 = 100; // velocita siringa alta
        pressione 1
const int ulssa1risc = 150; // velocita siringa
        alta pressione 1 risciacquo
/* FINE DEFINIZIONE SPOSTAMENTI
        ul/s */

/* DEFINIZIONE VARIABILI        */
const int vmisc = 4000 ;// volume di
        miscelazione
const int ritardo = 201; // ritardo di default
const int no = 100; // parametro di default per
        riempire valori ignorati in funzioni
const int nCicliRisciacquo = 3 ;// prima di
        pres : 3
const int nCicliScarico = 2 ;// prima di pres :
        2
const unsigned long tstazionario = 5000; //
        5000
const unsigned long tlaser = 1000; // on laser
        1000
const int nmisure = 3 ;
const unsigned long tmisuraoff = 15000; // off
        laser 15000
const unsigned long ttra2cicli = 600000;
const int numero_ripetizioni = 5;
const double discrepanza_misure = 5.0000;

unsigned char tmpv[4];

/* FINE DEFINIZIONE VARIABILI    */
struct motosiringa
{
   char strcmd[20]   ;
   char strstatus[20] ;
   char tipo        ;
   char ind         ;
   char stat[8]     ;
   char errc[40]    ;
   char tppos[9]    ;
   float ulatt      ;
   char statmov     ;
   float ulmax      ;
   int finecS       ;
   int finecI       ;
   float psa        ;
   char crch        ;
   char crcl        ;
   int okTR         ;// 0 Tx ed Rx sono corretti;
        1 errore nelle comunicazioni
};

struct elettrovalvole
{
   char strcmd[20]   ;
   char strstatus[20] ;
   char tipo        ;
   char ind         ;
   char stat[8]     ;
   char errc[40]    ;
   char
        cev1,cev2,cev3,cev4,cev5,cev6,cev7
        ,cev8,cev9,cev10;  // 'A' = on ; 'S' =
        off
   char
        sev1,sev2,sev3,sev4,sev5,sev6,sev7,
        sev8,sev9,sev10;  // 'A' = on ; 'S' = off
   char crch        ;
   char crcl        ;
   int okTR         ;// 0 Tx ed Rx sono corretti;
        1 errore nelle comunicazioni
};

struct motosiringa sb1stat;
struct motosiringa sb1cmd;

struct motosiringa cmdset;

struct motosiringa sb2stat;
struct motosiringa sb2cmd;

struct motosiringa sb3stat;
struct motosiringa sb3cmd;

struct motosiringa sa1stat;
struct motosiringa sa1cmd;
```

```
struct elettrovalvole ev;

// Funzioni utilizzate nel main
void    Inizializzazioni                ( void ) ;
void    Ritardo                         ( int Millisecondif ) ;
void    TemperaturaScheda               ( double* Temperaturaf ) ;
void    AlimentazioneScheda             ( double* TensioneAlimentazionef ) ;
void    TrasmettiUSB                    ( char* Messaggiof ) ;      // Trasmissione a 115200
void    TrasmettiNET1                   ( char* Messaggiof ) ;      // Trasmissione a 19200
void    TrasmettiNET2                   ( char* Messaggiof ) ;      // Trasmissione a 19200
void    TrasmettiNET1M                  ( char* Messaggiof ) ;      // Trasmissione a 19200
void    TrasmettiNET2M                  ( char* Messaggiof ) ;      // Trasmissione a 19200
void    TrasmettiRaspy                  ( char* Messaggiof ) ;      // Trasmissione a 19200
unsigned int    ADS8341         ( char Canalef ) ;
unsigned int    ADS8341_Mediato ( char Canalef ) ;
void    TestHardware            ( void ) ;
void    CorrentiLaser           ( double PercentualeLaser1, double PercentualeLaser2, double PercentualeLaser3, double PercentualeLaser4 ) ;
void    CalcolaCRC16            ( unsigned char* Vettoref, unsigned char NumeroCaratterif, unsigned char CRCF[2] ) ;
void   getstatusM       ( struct motosiringa *data , int ind)  ;
void   initStatM        ( struct motosiringa *data , int ind) ;
void    setcmdM                 (char s[3] ,int ind, struct motosiringa *data, float fmul, float fmuls , char dis, int ritardo);
void    writePosMinFile         (int inds, float ulattivo);
char    idM                     (int indirizzo);
int     stringcomp              (char* s1, char* s2, int lunghezza);
int     str2num                 (char* ss);
int     c2i                     (char c);
int     *initvece               (char *input, int *level);
void   initStatE        ( struct elettrovalvole *data , int ind) ;
void    getstatusE              ( struct elettrovalvole *data , int ind)  ;
void   setcmdE          (char s[3] ,int ind, struct elettrovalvole *data, char* vecev, int tempoms, int ritardo)  ;
void    writeMeasureinFile      (double ValvecI[3] , double vb, double assorbanza)  ;
double  maxim                   (double ValvecI[3])  ;
double minim            (double ValvecI[3]) ;
void   str2double       (char *sv)  ;
void   MisureMain       (void);
void   dtb              (float val);
void    TrasmettiRaspyData      ( char* Messaggiof ) ;
// Definizione delle variabili utilizzate nel programma
char TempoDebug, StatoDebug, SalvaDebug ;
char    VettoreRicezioneUSB[200], IndiceSerialeUSB, ArrivatoComandoUSB ;
char    VettoreRicezioneNET1[200], IndiceSerialeNET1, ArrivatoComandoNET1 ;
char    VettoreRicezioneNET2[200], IndiceSerialeNET2, ArrivatoComandoNET2 ;
char    VettoreRicezioneRaspy[200], IndiceSerialeRaspy, ArrivatoComandoRaspy ;
int counter ;
int cc = 0; // TOGLI

double latitudine, longitudine, profondita ; // ricevuti da sottomarino
int missione = 0;

double Temperatura, TensioneAlimentazione
```

```
        ;
FSFILE * pointer ;
FSFILE * pointerM ;
FSFILE * pointerMeasure ;

char Vettore[150], Ciclof ;
unsigned char Tempf, Unitaf, Decinef ;
unsigned int Valore[4] ;
double Perc1f, Perc2f, Perc3f, Perc4f ;
TEMPO DataEOra ; // togli

int main ( void )
{
        Inizializzazioni ( ) ;
   ONTEMP = 1 ;
   ON_5V = 1 ;
   ON_12V = 1 ;
   ON_ANALOG = 1 ;
   ON_RASPBERRY = 1 ;

   DELAY_MS(15000);

   // UART4 Interfaccia di comunicazione
        verso il RaspBerry alla velocità
        19200
   TRISFbits.TRISF2     = 0 ;        // Tx del
        microcontrollore verso il RaspBerry
   TRISFbits.TRISF8     = 1 ;        // Rx del
        microcontrollore per ricevere dal
        RaspBerry
   IFS5bits.U4RXIF                   = 0 ;
   IEC5bits.U4RXIE                   = 1 ;
   U4BRG
         = 23 ;
   U4MODEbits.BRGH                   = 0 ;
   U4MODEbits.UARTEN                 = 1 ;
   U4STAbits.UTXEN                   = 1 ;
   ArrivatoComandoRaspy= 0 ;

   while (1)
   {
      TestHardware ( ) ;

      if   (   (   U1STAbits.FERR    )   ||   (
         U1STAbits.OERR ) )
          {
              U1MODE            = 0 ;
            U1STA             = 0 ;
               U1MODEbits.BRGH    = 1 ;//1
            U1MODEbits.UARTEN   = 1 ;
         U1STAbits.UTXEN     = 1 ;
        ArrivatoComandoUSB  = 0 ;
         TrasmettiUSB("\nU1RX_ERROR\n
         ");

         TrasmettiUSB(VettoreRicezioneUS
         B);
          }
      if   (   (   U2STAbits.FERR    )   ||   (
         U2STAbits.OERR ) )
          {
              U2MODE            = 0 ;
            U2STA             = 0 ;
               U2MODEbits.BRGH    = 0 ;
            U2MODEbits.UARTEN   = 1 ;
         U2STAbits.UTXEN     = 1 ;
        ArrivatoComandoNET1 = 0 ;

         TrasmettiUSB("\nU2RX_ERROR\n
         ");

         TrasmettiUSB(VettoreRicezioneNE
         T1);
          }
      if   (   (   U3STAbits.FERR    )   ||   (
         U3STAbits.OERR ) )
          {
              U3MODE            = 0 ;
            U3STA             = 0 ;
               U3MODEbits.BRGH    = 0 ;
            U3MODEbits.UARTEN   = 1 ;
         U3STAbits.UTXEN     = 1 ;
        ArrivatoComandoNET2 = 0 ;

         TrasmettiUSB("\nU3RX_ERROR\n
         ");

         TrasmettiUSB(VettoreRicezioneNE
         T1);
          }
      if ( ( U4STAbits.FERR ) )
          {
              U4MODE            = 0 ;
            U4STA             = 0 ;
               U4MODEbits.BRGH    = 0 ;
            U4MODEbits.UARTEN   = 1 ;
         U4STAbits.UTXEN     = 1 ;
        ArrivatoComandoRaspy= 0 ;
                                                  /

         TrasmettiUSB("\nU4RX_ERROR_F
         ERR\n");
          }
```

```
      if ( ( U4STAbits.OERR ) )
          {
              U4MODE            = 0 ;
            U4STA             = 0 ;
              U4MODEbits.BRGH    = 0 ;
            U4MODEbits.UARTEN   = 1 ;
          U4STAbits.UTXEN     = 1 ;
          ArrivatoComandoRaspy= 0 ;

          TrasmettiUSB("\nU4RX_ERROR_
          OERR\n");
           }
//       missione_i++;




   }
}


/
//**********************************
          ************          FUNZIONI
          ****************************
          **********************

void    __attribute__((interrupt,no_auto_psv))
          _T1Interrupt        ( void )
          {
   char Vettoref[150], Indicef ;
   unsigned int Valoref[4] ;
   TempoDebug++ ;
   if ( SalvaDebug )
      {
      LEDBLU = 1 ;
      Valoref[0] = ADS8341_Mediato ( 0 ) ;
      Valoref[1] = ADS8341_Mediato ( 1 ) ;
      Valoref[2] = ADS8341_Mediato ( 2 ) ;
      Valoref[3] = ADS8341_Mediato ( 3 ) ;
      sprintf  (  Vettoref,  "\n%d;%d;%d;%d",
          Valoref[0],  Valoref[1],  Valoref[2],
          Valoref[3] ) ;
      Indicef = 0 ;
      while ( Vettoref[Indicef] != 0x00 )
         Indicef++ ;
      if  (  FSfwrite  (  Vettoref,  1,  Indicef,
          pointer ) != Indicef )
         while ( 1 ) ;
      }
   if ( LEDVERDE == 0 )
      LEDVERDE = 1 ;
   else
      LEDVERDE = 0 ;
      IFS0bits.T1IF      = 0 ;
      }

// Gestione dell'interrupt della ricezione
      seriale tramite porta USB
void
      __attribute__((__interrupt__,no_aut
      o_psv)) _U1RXInterrupt    ( void )
      {
      unsigned char Caratteref ;
   IFS0bits.U1RXIF = 0 ;
      while ( U1STAbits.URXDA == 1 )
       {
       Caratteref = U1RXREG ;
         if ( Caratteref == '*' )
               IndiceSerialeUSB = 0 ;

      VettoreRicezioneUSB[IndiceSeriale
      USB] = Caratteref ;
         if ( IndiceSerialeUSB < 198 )
               IndiceSerialeUSB++ ;

      VettoreRicezioneUSB[IndiceSeriale
      USB] = 0x00 ;
       if ( Caratteref == '+' )
               ArrivatoComandoUSB = 1
      ;
       }

      }

// Gestione dell'interrupt della ricezione
      seriale tramite porta NET1
void
      __attribute__((__interrupt__,no_aut
      o_psv)) _U2RXInterrupt    ( void )
      {
      unsigned char Caratteref ;
   IFS1bits.U2RXIF = 0 ;
      while ( U2STAbits.URXDA == 1 )
       {
       Caratteref = U2RXREG ;
         if ( Caratteref == '*' )
               IndiceSerialeNET1 = 0 ;

      VettoreRicezioneNET1[IndiceSerial
      eNET1] = Caratteref ;
         if ( IndiceSerialeNET1 < 198 )
               IndiceSerialeNET1++ ;
```

```
        VettoreRicezioneNET1[IndiceSerial
        eNET1] = 0x00 ;
         if ( Caratteref == '+' )
                ArrivatoComandoNET1   =
        1 ;

         }

        }

// Gestione dell'interrupt della ricezione
        seriale tramite porta NET2
void
        __attribute__((__interrupt__,no_aut
        o_psv)) _U3RXInterrupt    ( void )
        {
        unsigned char Caratteref ;
   IFS5bits.U3RXIF = 0 ;
        while ( U3STAbits.URXDA == 1 )
         {
         Caratteref = U3RXREG ;
           if ( Caratteref == '*' )
                IndiceSerialeNET2 = 0 ;

        VettoreRicezioneNET2[IndiceSerial
        eNET2] = Caratteref ;
           if ( IndiceSerialeNET2 < 198 )
                IndiceSerialeNET2++ ;

        VettoreRicezioneNET2[IndiceSerial
        eNET2] = 0x00 ;
         if ( Caratteref == '+' )
                ArrivatoComandoNET2   =
        1 ;
      //TrasmettiUSB(Caratteref);
         }

        }

// Gestione dell'interrupt della ricezione
        seriale tramite porta RaspBerry
void
        __attribute__((__interrupt__,no_aut
        o_psv)) _U4RXInterrupt    ( void )
        {
        unsigned char Caratteref ;
   IFS5bits.U4RXIF = 0 ;
        while ( U4STAbits.URXDA == 1 )
         {
         Caratteref = U4RXREG ;
           if ( Caratteref == '*' )
                IndiceSerialeRaspy = 0 ;

        VettoreRicezioneRaspy[IndiceSerial
        eRaspy] = Caratteref ;
           if ( IndiceSerialeRaspy< 198 )
                IndiceSerialeRaspy++ ;

        VettoreRicezioneRaspy[IndiceSerial
        eRaspy] = 0x00 ;
         if ( Caratteref == '+' )
                ArrivatoComandoRaspy   =
        1 ;
         }

        }


void    Inizializzazioni                (
        void )
        {
        // Configurazione pin programmabili
        __builtin_write_OSCCONL           (
        OSCCON & 0xBF ) ;
         // Unlock Registers

        // Configure Input Functions (Table
        10-3))
        RPINR18bits.U1RXR  = 4 ;          //
        Assegna U1RX al pin RP4     USB
   RPINR19bits.U2RXR   = 13 ;
                 // Assegna U2RX al pin
        RP13    NET 1
   RPINR17bits.U3RXR   = 7 ;
                 // Assegna U3RX al pin
        RP7     NET 2
   RPINR27bits.U4RXR   = 15 ;
                 // Assegna U4RX al pin
        RP15    RaspBerry
   RPINR20bits.SDI1R   = 19 ;
                 // Assegna SDI1 al pin
        RP19    ADS8341
   RPOR4bits.RP8R      = 18 ;
         // Porta RP7 mappata come OC1 per
        il pilotaggio del laser 1
   RPOR4bits.RP9R      = 19 ;          // Porta
        RP8 mappata come OC2 per il
        pilotaggio del laser 2
   RPOR9bits.RP18R     = 20 ;          // Porta
        RP9 mappata come OC3 per il
        pilotaggio del laser 3
   RPOR13bits.RP27R    = 21 ;          //
        Porta RP27 mappata come OC4 per il
```

```
        pilotaggio del laser 4
RPINR22bits.SDI2R   = 22 ;              //
        Assign SDI2 To Pin RP22

// Configure Output Functions (Table 10-
        4)
RPOR1bits.RP2R      = 3 ;
                    // Assegna U1TX al pin
        RP2     USB
RPOR14bits.RP28R    = 5 ;
                    // Assegna U2TX al pin
        RP28    NET1
RPOR3bits.RP6R      = 28 ;
                    // Assegna U3TX al pin
        RP6     NET2
RPOR15bits.RP30R    = 30 ;
                    // Assegna U3TX al pin
        RP30    RaspBerry
RPOR10bits.RP21R    = 8 ;
                    // Assegna SCK1OUT al
        pin RP21  ADS8341
RPOR13bits.RP26R    = 7 ;               //
        Assegna SDO1 al pin RP26
        ADS8341
RPOR1bits.RP3R                  = 10 ;
                        // Assign SDO2
        To Pin RP3
RPOR6bits.RP12R                 = 11 ;
                        //          Assign
        SCK2OUT To Pin RP12

__builtin_write_OSCCONL ( OSCCON |
        0x40 ) ;                        //  Lock
        Register


// Settaggio delle due porte optoisolate in
        uscita e due in ingresso
TRISGbits.TRISG15   = 0 ;   // Uscita
        optoisolata 1
TRISEbits.TRISE5    = 0 ;   // Uscita
        optoisolata 2
TRISEbits.TRISE6    = 1 ;   // Ingresso
        optoisolato 1
TRISEbits.TRISE7    = 1 ;   // Ingresso
        optoisolato 2

// Settaggio delle porte dei quattro led
TRISGbits.TRISG12   = 0 ;  // Led Verde
TRISGbits.TRISG14   = 0 ;  // Led Giallo
TRISEbits.TRISE0    = 0 ;  // Led Rosso
TRISEbits.TRISE1    = 0 ;  // Led Blu

// Settaggio stato iniziale dei quattro led
// LEDVERDE   = 0 ;  // Led Verde
LEDGIALLO  = 0 ;  // Led Giallo
LEDROSSO   = 0 ;  // Led Rosso
LEDBLU     = 0 ;  // Led Blu


// Settaggio delle 12 porte di espansione
        tutte in uscita
TRISBbits.TRISB15   = 0 ;  // I/O1
TRISBbits.TRISB14   = 0 ;  // I/O2
TRISBbits.TRISB13   = 0 ;  // I/O3
TRISBbits.TRISB12   = 0 ;  // I/O4
TRISFbits.TRISF12   = 0 ;  // I/O5
TRISFbits.TRISF13   = 0 ;  // I/O6
TRISDbits.TRISD4    = 0 ;  // I/O7
TRISDbits.TRISD5    = 0 ;  // I/O8
TRISDbits.TRISD6    = 0 ;  // I/O9
TRISDbits.TRISD7    = 0 ;  // I/1O
TRISFbits.TRISF0    = 0 ;  // I/11
TRISFbits.TRISF1    = 0 ;  // I/12

// Settaggio delle 4 uscite open drain
TRISEbits.TRISE4    = 0 ;   // Mos di
        uscita opedrain 1
TRISEbits.TRISE3    = 0 ;   // Mos di
        uscita opedrain 2
TRISEbits.TRISE2    = 0 ;   // Mos di
        uscita opedrain 3
TRISGbits.TRISG13   = 0 ;   // Mos di
        uscita opedrain 4
OPENDRAIN1  = 0 ;
OPENDRAIN2  = 0 ;
OPENDRAIN3  = 0 ;
OPENDRAIN4  = 0 ;

// Settaggio dei pin che abilitano gli enable
        delle alimentazioni
TRISAbits.TRISA9    = 0 ;  // Controlla
        alimentazione 12V
ON_12V          = 0 ;
TRISBbits.TRISB3    = 0 ;  // Controlla
        alimentazione 5V
ON_5V           = 0 ;
TRISAbits.TRISA0    = 0 ;  // Controlla
        alimentazione 5V analogica
ON_ANALOG       = 0 ;
TRISAbits.TRISA10   = 0 ;  // Controlla
        alimentazione per il RaspBerry
ON_RASPBERRY    = 0 ;
```

```
// Impostazione delle porte per il RTC DS1338
    TRISAbits.TRISA14       = 0 ;
    TRISAbits.TRISA15       = 0 ;
    ODCAbits.ODA14
     = 1 ;
                    // Porte poste in modalità Open Drain
    ODCAbits.ODA15
     = 1 ;
                    // Porte poste in modalità Open Drain
    LATAbits.LATA14
     = 1 ;
    LATAbits.LATA15
     = 1 ;

    // Impostazione del timer1 per il lampeggio del led verde ogni secondo ( tempo di timer di 500ms ))
    TMR1
     = 0x0000 ;
IFS0bits.T1IF     = 0 ;
IEC0bits.T1IE     = 1 ;
PR1           = 14400 ;
T1CONbits.TCKPS   = 3 ;
    T1CONbits.TON          = 1 ;
TempoDebug      = 0 ;
StatoDebug      = 1 ;
SalvaDebug      = 0 ;

// Settaggio dei 4 output compare per realizzare i 4PWM a circa 10KHz per i lpilotaggio del generatore di corrente
TRISBbits.TRISB5 = 0 ;    // Porta PWM per ilotaggio laser 3
TRISBbits.TRISB8 = 0 ;    // Porta PWM per ilotaggio laser 1
TRISBbits.TRISB9 = 0 ;    // Porta PWM per ilotaggio laser 2
TRISGbits.TRISG9 = 0 ;    // Porta PWM per ilotaggio laser 4
OC1CON1 = 0 ;
OC1CON2bits.SYNCSEL = 31 ;
OC1CON1bits.OCTSEL  = 0x07 ;
OC1CON1bits.OCM     = 6 ;
OC1R    = 0 ;
OC1RS   = 200 ;
OC2CON1 = 0 ;
OC2CON2bits.SYNCSEL = 31 ;
OC2CON1bits.OCTSEL  = 0x07 ;
OC2CON1bits.OCM     = 6 ;
OC2R    = 0 ;
OC2RS   = 200 ;
OC3CON1 = 0 ;
OC3CON2bits.SYNCSEL = 31 ;
OC3CON1bits.OCTSEL  = 0x07 ;
OC3CON1bits.OCM     = 6 ;
OC3R    = 0 ;
OC3RS   = 200 ;
OC4CON1 = 0 ;
OC4CON2bits.SYNCSEL = 31 ;
OC4CON1bits.OCTSEL  = 0x07 ;
OC4CON1bits.OCM     = 6 ;
OC4R    = 0 ;
OC4RS   = 200 ;

// UART1 Interfaccia di comunicazione che utilizza il CP2102 con velocità di 115200 baud
TRISDbits.TRISD8    = 0 ;     // Tx del microcontrollore verso USB
TRISDbits.TRISD9    = 1 ;     // Rx del microcontrollore per ricevere dalla USB
IFS0bits.U1RXIF             = 0 ;
IEC0bits.U1RXIE             = 1 ;
U1BRG
     = 3 ;
U1MODEbits.UARTEN           = 1 ;
U1STAbits.UTXEN             = 1 ;
ArrivatoComandoUSB  = 0 ;

// UART2 Interfaccia di comunicazione RS422 con la NET1 con velocità 19200
TRISBbits.TRISB4    = 0 ;     // Tx del microcontrollore verso la NET1
TRISBbits.TRISB2    = 1 ;     // Rx del microcontrollore per ricevere dalla NET1
IFS1bits.U2RXIF             = 0 ;
IEC1bits.U2RXIE             = 1 ;
U2BRG
     = 23 ;
U2MODEbits.BRGH             = 0 ;
U2MODEbits.UARTEN           = 1 ;
U2STAbits.UTXEN             = 1 ;
ArrivatoComandoNET1 = 0 ;

// UART3 Interfaccia di comunicazione RS422 con la NET2 con velocità 19200
```

```
TRISBbits.TRISB6    = 0 ;       // Tx del microcontrollore verso la NET2
TRISBbits.TRISB7    = 1 ;       // Rx del microcontrollore per ricevere dalla NET2
IFS5bits.U3RXIF                 = 0 ;
IEC5bits.U3RXIE                 = 1 ;
U3BRG
     = 23 ;
U3MODEbits.BRGH                 = 0 ;
U3MODEbits.UARTEN               = 1 ;
U3STAbits.UTXEN                 = 1 ;
ArrivatoComandoNET2 = 0 ;



    // Inizializzazione del convertitore A/D
TRISEbits.TRISE8    = 0 ;       // Settaggio uscita per alimentare il sensore di corrente
TRISEbits.TRISE9    = 1 ;       // Settaggio del pin in ingresso
    AD1CON1
     = 0x00E4 ;
    AD1CON2
     = 0x2404 ;
    AD1CSSHbits.CSSL16  = 1 ;   // Corrisponde al canale per la misura della tensione d'ingresso RC4
    AD1CSSHbits.CSSL21  = 1 ;   // Corrisponde al canale per la misura della temperatura  RE9
    AD1CON3bits.SAMC    = 31 ;
    AD1CON3bits.ADCS    = 64 ;
    AD1CHSbits.CH0SA    = 11 ;
    AD1CON1bits.ADON    = 1 ;
    ANSA
     = 0x0000 ;
    ANSB
     = 0x0000 ;
    ANSC
     = 0x0010 ;
ANSD
     = 0x0000 ;
    ANSE
     = 0x0200 ;
ANSF
     = 0x0000 ;
    ANSG
     = 0x0000 ;

    // Impostazioni dell'interfaccia SPI per il convertitore A/D ADS8341
SPI1CON1bits.PPRE   = 3 ;
SPI1CON1bits.SPRE   = 2 ;
SPI1CON1bits.SSEN   = 0 ;
SPI1CON1bits.MSTEN  = 1 ;
SPI1CON1bits.CKP    = 0 ;
SPI1CON1bits.CKE    = 1 ;
SPI1CON1bits.SMP    = 0 ;
SPI1CON1bits.MODE16 = 0 ;
SPI1STATbits.SPIROV             = 0 ;
SPI1STATbits.SPIEN  = 1 ;
TRISCbits.TRISC2    = 0 ;       // Pin CS del convertitore A/D settato come uscita
AD_CS               = 1 ;
TRISCbits.TRISC1    = 0 ;       // Pin SHDN del convertitore A/D settato come uscita
AD_SHDN             = 1 ;
TRISCbits.TRISC3    = 1 ;       // Pin BUSY del convertitore A/D settato come ingresso
    TRISGbits.TRISG7            = 0         ;
                                // Pin CLK settato come uscita porta RP21
    TRISGbits.TRISG6            = 0 ;
                                // Pin DOUT settato come uscita    porta RP26
    TRISGbits.TRISG8            = 1 ;
                                // Pin DOUT settato come ingresso porta RP19

    // Impostazioni dell'interfaccia SPI per la scheda SD
SPI2CON1bits.PPRE   = 3 ;
SPI2CON1bits.SPRE   = 7 ;
SPI2CON1bits.SSEN   = 0 ;
SPI2CON1bits.MSTEN  = 1 ;
SPI2CON1bits.CKP    = 0 ;
SPI2CON1bits.CKE    = 1 ;
SPI2CON1bits.SMP    = 0 ;
SPI2CON1bits.MODE16 = 0 ;
SPI2STATbits.SPIROV             = 0 ;
SPI2STATbits.SPIEN  = 1 ;
    TRISDbits.TRISD0            = 0         ;
    // CS della scheda MicroSD
    TRISDbits.TRISD12           = 1         ;
    // Pin in input per il detect della scheda MicroSD
    TRISDbits.TRISD11           = 0 ;       // Clock della scheda MicroSD
    TRISDbits.TRISD3            = 1 ;       //
```

```
        Input del microcontrollore della porta
        SPI dedicata alla MicroSD
        TRISDbits.TRISD10       = 0 ;     //
        Output del microcontrollore della
        porta SPI dedicata alla MicroSD
        }


void    Ritardo
                ( int Millisecondif )
        {
        int CicloA, CicloB ;
        for ( CicloA = 0 ; CicloA <
        Millisecondif ; CicloA++ )
         {
         for ( CicloB =0 ; CicloB < 1460 ;
        CicloB++ )
                {       }
         }
        }


void    TrasmettiUSB            ( char*
        Messaggiof )
   {
        unsigned char Puntatoref ;
        Puntatoref = 0 ;
        while ( Messaggiof[Puntatoref] !=
        0x00 )
         {
         while ( !U1STAbits.TRMT )
        {  }
         if ( Messaggiof[Puntatoref] == '\n' ||
        Messaggiof[Puntatoref] == '\\n' )
                U1TXREG = 0x0D ;

         else
                U1TXREG         =
        Messaggiof[Puntatoref] ;
         Puntatoref++ ;
         }
   }


void    TrasmettiNET1           ( char*
        Messaggiof )
   {
        unsigned char Puntatoref ;
        Puntatoref = 0 ;
        while ( Messaggiof[Puntatoref] !=
        0x00 )
         {
         while ( !U2STAbits.TRMT )
        {  }
         if ( Messaggiof[Puntatoref] == '\n' )
                U2TXREG = 0x0D ;

         else
                U2TXREG         =
        Messaggiof[Puntatoref] ;
         Puntatoref++ ;
         }
   }


void    TrasmettiNET2           ( char*
        Messaggiof )
   {
        unsigned char Puntatoref ;
        Puntatoref = 0 ;
        while ( Messaggiof[Puntatoref] !=
        0x00 )
         {
         while ( !U3STAbits.TRMT )
        {  }
         if ( Messaggiof[Puntatoref] == '\n' )
                U3TXREG = 0x0D ;

         else
                U3TXREG         =
        Messaggiof[Puntatoref] ;
         Puntatoref++ ;
         }
   }


void    TrasmettiNET1M          ( char*
        Messaggiof )
   {
        unsigned char Puntatoref ;
        Puntatoref = 0 ;
        while ( Puntatoref < 20  )
         {
         while ( !U2STAbits.TRMT )
        {  }

         U2TXREG                 =
        Messaggiof[Puntatoref] ;
         Puntatoref++ ;
         }
   }


void    TrasmettiNET2M          ( char*
```

```
        Messaggiof )
  {
        unsigned char Puntatoref ;
        Puntatoref = 0 ;
        while ( Puntatoref < 20  )
         {
         while ( !U3STAbits.TRMT )
        {  }
         U3TXREG                    =
        Messaggiof[Puntatoref] ;
         Puntatoref++ ;
         }
  }

void    TrasmettiRaspy          ( char*
        Messaggiof )
  {
        unsigned char Puntatoref ;
        Puntatoref = 0 ;
        while  (  Messaggiof[Puntatoref]  !=
        0x00 )
         {
         while ( !U4STAbits.TRMT )
        {  }
         if ( Messaggiof[Puntatoref] == '\n' )
                U4TXREG = 0x0D ;

         else
                U4TXREG                 =
        Messaggiof[Puntatoref] ;
         Puntatoref++ ;
         }
  }

void   TrasmettiRaspyData          ( char*
        Messaggiof )
  {
        unsigned char Puntatoref ;
        Puntatoref = 0 ;
        while ( Puntatoref < 50 )
         {
         while ( !U4STAbits.TRMT )
        {  }
         U4TXREG                    =
        Messaggiof[Puntatoref] ;
         Puntatoref++ ;
         }
  }


void   TemperaturaScheda          ( double*
        Temperaturaf )
  {
  int Valoref ;
  double Tempf ;
  Valoref = ADC1BUF1 ;
  Tempf = (double)Valoref ;
  Tempf = Tempf * 3300.0 / 1024.0 ;
  Tempf = Tempf - 500 ;
  Tempf = Tempf / 10.0 ;
  *Temperaturaf = Tempf ;
  }


void   AlimentazioneScheda        ( double*
        TensioneAlimentazionef )
  {
  int Valoref ;
  double Tempf ;
  Valoref = ADC1BUF0 ;
  Tempf = (double)Valoref ;
  Tempf = Tempf * 3.3 / 64.0 ;
  *TensioneAlimentazionef = Tempf ;
  }


unsigned int   ADS8341          ( char Canalef
        )
  {
  unsigned int Altof, Bassof, Valoref ;
  // Definizione dei canali del convertitore
        A/D esterno
  Valoref = 4 ;
  if ( Canalef == 0 )
     Valoref = 0xA7 ;
  if ( Canalef == 1 )
     Valoref = 0xE7 ;
  if ( Canalef == 2 )
     Valoref = 0xD7 ;
  if ( Canalef == 3 )
     Valoref = 0x97 ;
  if ( Valoref == 4 )
     return ( 65535 ) ;
  AD_CS = 0 ;
  SPI1BUF = Valoref ;
  while ( !SPI1STATbits.SPIRBF )
     {  }          /* check for TX shift register
        full */
  Altof = SPI1BUF ;
  SPI1BUF = 0x00 ;
  while ( !SPI1STATbits.SPIRBF )
     {  }          /* check for TX shift register
        full */
  Altof = SPI1BUF ;
```

```
  SPI1BUF = 0x00 ;
  while ( !SPI1STATbits.SPIRBF )
     { }          /* check for TX shift register
        full */
  Bassof = SPI1BUF ;
  AD_CS = 1 ;

  Altof = Altof * 256 ;
  Altof = Altof + Bassof ;
  return ( Altof ) ;
  }


unsigned int    ADS8341_Mediato     ( char
        Canalef )
  {
  unsigned int Campionif[8], Tempf ;
  unsigned long Valoref ;
  char IndiceAf, IndiceBf ;

  for ( IndiceAf = 0 ; IndiceAf < 8 ;
        IndiceAf++ )
     Campionif[IndiceAf]    =    ADS8341(
        Canalef ) ;

  for ( IndiceAf = 0 ; IndiceAf < 6 ;
        IndiceAf++ )
         {
         for ( IndiceBf = 0 ; IndiceBf < 7 ;
        IndiceBf++ )
                  {
                  if ( Campionif[IndiceBf] >
        Campionif[(IndiceBf+1)] )
                           {
                           Tempf          =
        Campionif[IndiceBf] ;

         Campionif[IndiceBf]                =
        Campionif[(IndiceBf+1)] ;

         Campionif[(IndiceBf+1)] = Tempf ;
                           }
                  }
         }

  Valoref = (unsigned long)(Campionif[2]) ;
  Valoref    =    Valoref    +    (unsigned
        long)(Campionif[3]) ;
  Valoref    =    Valoref    +    (unsigned
        long)(Campionif[4]) ;
  Valoref    =    Valoref    +    (unsigned
        long)(Campionif[5]) ;
  Valoref = Valoref / 4 ;
  Tempf   = (unsigned int)(Valoref) ;
  return ( Tempf ) ;
  }


void    TestHardware          ( void )
  {

  //TrasmettiUSB                          (
        VettoreRicezioneRaspy);
  if ( TempoDebug == 20 )
     {
     TempoDebug = 0 ;
     StatoDebug = 1 ;
     }

  if    (    ArrivatoComandoUSB    ==    1    ||
        ArrivatoComandoRaspy == 1 )
     {
        if          (          strcmp
        ("*Help+",VettoreRicezioneUSB)
        == 0 )
                 {
        StatoDebug = 1 ;
        TempoDebug = 0 ;
        }
        else        if        (        strcmp
        ("*Temp+",VettoreRicezioneUSB)
        == 0 )
                 {
        StatoDebug = 2 ;
        TempoDebug = 0 ;

        initStatE ( &ev , indev);

        initStatM ( &sb1stat , sb1);
        initStatM ( &sb1cmd  , sb1);

        initStatM ( &sb2stat , sb2);
        initStatM ( &sb2cmd  , sb2);

        initStatM ( &sb3stat , sb3);
        initStatM ( &sb3cmd  , sb3);

        initStatM ( &sa1stat , sa1);
        initStatM ( &sa1cmd  , sa1);

        setcmdE        ("set",    indev,    &ev,
        "1a,2a,3a,4a,5a,6a,7a,8a,9a,10a", no,
        ritardo);
        getstatusE( &ev , indev);
```

```
TrasmettiUSB(ev.strstatus);
setcmdM ("int", sa1, &sa1cmd,
ulsa1ad, 400, 'I', ritardo );
getstatusM( &sa1stat , sb3);
TrasmettiUSB(sa1stat.strstatus);
int pippo=1;
}
else if ( strcmp
("*Val+",VettoreRicezioneUSB) ==
0 )
{
StatoDebug = 3 ;
TempoDebug = 0 ;
}
else if ( strcmp
("*ADC+",VettoreRicezioneUSB)
== 0 )
{
StatoDebug = 4 ;
TempoDebug = 0 ;
}
else if ( strcmp
("*MOS+",VettoreRicezioneUSB)
== 0 )
{
StatoDebug = 5 ;
TempoDebug = 0 ;
}
else if ( strcmp
("*Net1+",VettoreRicezioneUSB)
== 0 )
{
StatoDebug = 6 ;
TempoDebug = 0 ;
}
else if ( strcmp
("*Net2+",VettoreRicezioneUSB)
== 0 )
{
StatoDebug = 7 ;
TempoDebug = 0 ;
}
else if ( strcmp
("*Data+",VettoreRicezioneUSB)
== 0 )
{
StatoDebug = 8 ;
TempoDebug = 0 ;
}
else if ( strcmp
("*Set+",VettoreRicezioneUSB) ==
0 )
{
StatoDebug = 9 ;
TempoDebug = 0 ;
}
else if ( strcmp
("*Pot+",VettoreRicezioneUSB) ==
0 )
{
StatoDebug = 10 ;
TempoDebug = 0 ;
}
else if ( strcmp
("*Ricarica1+",VettoreRicezioneUS
B) == 0 )
{
StatoDebug = 11 ;
TempoDebug = 0 ;
}
else if ( strcmp
("*Ricarica2+",VettoreRicezioneUS
B) == 0 )
{
StatoDebug = 12 ;
TempoDebug = 0 ;
}
else if ( strcmp
("*Smonta1+",VettoreRicezioneUS
B) == 0 )
{
StatoDebug = 13 ;
TempoDebug = 0 ;
}
else if ( strcmp
("*Smonta2+",VettoreRicezioneUS
B) == 0 )
{
StatoDebug = 14 ;
TempoDebug = 0 ;
}
else if ( strcmp
("*Leggi+",VettoreRicezioneUSB)
== 0 )
{
StatoDebug = 16 ;
TempoDebug = 0 ;
}
else if ( strcmp
("*DosMisc+",VettoreRicezioneUS
B) == 0 )
{
StatoDebug = 17 ;
TempoDebug = 0 ;
```

```
    }
  else if ( strcmp
    ("*Misura+",VettoreRicezioneUSB)
    == 0 )
          {
    StatoDebug = 18 ;
    TempoDebug = 0 ;
    }
  else if ( strcmp
    ("*Scarico+",VettoreRicezioneUSB)
    == 0 )
          {
    StatoDebug = 19 ;
    TempoDebug = 0 ;
    }
  else if ( strcmp
    ("*TestMisure+",VettoreRicezioneU
    SB) == 0 )
          {
    StatoDebug = 20 ;
    TempoDebug = 0 ;
    }
  else if ( strcmp
    ("*TestFirmware+",VettoreRicezion
    eUSB) == 0 )
          {
    StatoDebug = 21 ;
    TempoDebug = 0 ;
    }
  else if ( strcmp
    ("*TestFirmware+",VettoreRicezion
    eRaspy) == 0 )
          {
    StatoDebug = 0 ; //  21
    TempoDebug = 0 ;
    ArrivatoComandoRaspy = 0;
    TrasmettiUSB ( "\nINIZIO
     MISSIONE\n");
    missione++;


    MisureMain();


    int arrva = 0;
    while( arrva==0 )
    {

      TrasmettiRaspy("*FineMisura-");

      if(strstr(VettoreRicezioneRaspy,
    "STOPOK+") != NULL )
          arrva++;
        //TrasmettiUSB
      (VettoreRicezioneRaspy);
      }
      ArrivatoComandoRaspy = 0;

      TrasmettiUSB("\n\n*FineMisura-  :
      QUINDI *STOPOK+\n");

      }
    ArrivatoComandoUSB = 0 ;
    ArrivatoComandoRaspy = 0 ;
    }

if ( StatoDebug == 1 && cc == 0)
    {
    TrasmettiUSB ( "\n\n\nElenco comandi
      disponibili :") ;
       TrasmettiUSB ( "\n*Help+      =
      Visualizza questo messaggio") ;

    TrasmettiUSB ( "\n*Temp+      =
      Visualizza la temperatura rilevata dal
      sensore a bordo scheda") ;
    TrasmettiUSB ( "\n*Val+   = Visualizza
      la tensione di alimentazione della
      scheda e TESTA SENSORE HP") ;
    TrasmettiUSB ( "\n*ADC+       =
      Visualizza il valore adimensionato
      del converitore A/D") ;
    TrasmettiUSB ( "\n*MOS+   = Esegue
      un test sulle uscite open drain di
      potenza") ;
    TrasmettiUSB ( "\n*Net1+   = Avvia il
      test sulla NET1") ;
    TrasmettiUSB ( "\n*Net2+   = Avvia il
      test sulla NET2") ;
    TrasmettiUSB ( "\n*Data+      =
      Visualizza la data e l'ora attuale") ;
    TrasmettiUSB ( "\n*Set+     = Setta la
      data e l'ora del modulo RTC") ;
    TrasmettiUSB ( "\n*Pot+     = Setta le
      potenze dei vari laser") ;
    TrasmettiUSB ( "\n*Ricarica1+     =
      Ricarica sb1") ;
    TrasmettiUSB ( "\n*Ricarica2+     =
      Ricarica sb2") ;
    TrasmettiUSB ( "\n*Smonta1+       =
      Smonta sb1") ;
    TrasmettiUSB ( "\n*Smonta2+       =
      Smonta sb2") ;
    TrasmettiUSB ( "\n*Leggi+   = Legge e
```

```
        chiude il file di salvataggio del
        convertitore A/D") ;
      TrasmettiUSB ( "\n*DosMisc+ =
        Dosaggio e Miscelazione") ;
      TrasmettiUSB ( "\n*Misura+ =
        Misura") ;
      TrasmettiUSB ( "\n*Scarico+ =
        Scarico") ;
      TrasmettiUSB ( "\n*TestMisure+ =
        Testa la struttura FW escluso lo
        scarico") ;
      TrasmettiUSB ( "\n*TestFirmware+ =
        Testa la struttura FW completo") ;
      StatoDebug = 0 ;
//      cc++;
      }

   if ( StatoDebug == 2 )
      {
      TemperaturaScheda ( &Temperatura ) ;
      sprintf( Vettore,"\nTemperatura : %.1f
        °C ", Temperatura ) ;
      TrasmettiUSB ( Vettore ) ;
      StatoDebug = 0 ;
      }

   if ( StatoDebug == 3 )
      {

      initStatM ( &sa1stat , sa1);
      initStatM ( &sa1cmd  , sa1);
      initStatE ( &ev , indev);
      setcmdE ("set", indev, &ev,
        "1c,2c,3c,4c,5c,6c,7c,8c,9a,10c", no,
        ritardo);
      getstatusE( &ev , indev);
      int ik;
      while(ik<20)
      {

        setcmdM ("mov", sa1, &sa1cmd,
         100, 400, 'I', ritardo );
        DELAY_MS(1000);
        getstatusM( &sa1stat , sa1);
        memset(Vettore, '\0',
         sizeof(Vettore));
        sprintf ( Vettore, "\nPressione attuale
         SA1 : %.2f bar ;\nPosizione attuale
         : %.2f ul ;\n" , sa1stat.psa
         ,sa1stat.ulatt) ;
        TrasmettiUSB(Vettore);
        ik++;
      }
      StatoDebug = 0 ;

      }

   if ( StatoDebug == 4 )
      {
      for ( Ciclof = 0 ; Ciclof < 4 ; Ciclof++ )
        Valore[Ciclof] = ADS8341_Mediato
        ( Ciclof ) ;
      sprintf ( Vettore, "\nCH0 = %d ; CH1
        = %d ; CH2 = %d ; CH3 = %d",
        Valore[0], Valore[1], Valore[2],
        Valore[3] ) ;
      TrasmettiUSB ( Vettore ) ;
      StatoDebug = 0 ;
      }

   if ( StatoDebug == 5 )
      {
      OPENDRAIN1 = 1 ; OPENDRAIN2 =
        0 ; OPENDRAIN3 = 0 ;
        OPENDRAIN4 = 0 ;
      Ritardo ( 500 ) ;
      OPENDRAIN1 = 0 ; OPENDRAIN2 =
        1 ; OPENDRAIN3 = 0 ;
        OPENDRAIN4 = 0 ;
      Ritardo ( 500 ) ;
      OPENDRAIN1 = 0 ; OPENDRAIN2 =
        0 ; OPENDRAIN3 = 1 ;
        OPENDRAIN4 = 0 ;
      Ritardo ( 500 ) ;
      OPENDRAIN1 = 0 ; OPENDRAIN2 =
        0 ; OPENDRAIN3 = 0 ;
        OPENDRAIN4 = 1 ;
      Ritardo ( 500 ) ;
      OPENDRAIN1 = 0 ; OPENDRAIN2 =
        0 ; OPENDRAIN3 = 0 ;
        OPENDRAIN4 = 0 ;
      StatoDebug = 0 ;
      }

   if ( StatoDebug == 6 )
      {
      int p=1;
      TrasmettiNET1 ( "*Frase partita dalla
        NET1+" ) ;
      int c=1;
      TrasmettiUSB ( "\nPacchetto ricevuto
        sulla NET2 : " ) ;
      Ritardo ( 100 ) ;
      if ( ArrivatoComandoNET2 == 1 )
```

```
        {
        TrasmettiUSB                    (
         VettoreRicezioneNET2 ) ;
        ArrivatoComandoNET2 = 0 ;
        }
     else
        TrasmettiUSB ( "Nessun pacchetto
         ricevuto" ) ;
     StatoDebug = 0 ;
     }

  if ( StatoDebug == 7 )
     {
     TrasmettiNET2 ( "*Frase partita dalla
         NET2+" ) ;
     TrasmettiUSB ( "\nPacchetto ricevuto
         sulla NET1 : " ) ;
     Ritardo ( 100 ) ;
     if ( ArrivatoComandoNET1 == 1 )
        {
        TrasmettiUSB                    (
         VettoreRicezioneNET1 ) ;
        ArrivatoComandoNET1 = 0 ;
        }
     else
        TrasmettiUSB ( " Nessun pacchetto
         ricevuto" ) ;
     StatoDebug = 0 ;
     }

  if ( StatoDebug == 8 )
     {
     LeggiCalendario      ( &DataEOra ) ;
     sprintf ( Vettore, "\nSono le ore
         %x:%x:%x ",      DataEOra.Ore,
         DataEOra.Minuti,
         DataEOra.Secondi ) ;
     TrasmettiUSB ( Vettore ) ;
     sprintf ( Vettore, "\nLa data odierna e'
         %x.%x.%x ",      DataEOra.Data,
         DataEOra.Mese, DataEOra.Anno ) ;
     TrasmettiUSB ( Vettore ) ;
     StatoDebug = 0 ;
     }

  if ( StatoDebug == 9 )
     {
     ArrivatoComandoUSB = 0 ;
     TrasmettiUSB ( "\nIndicare il giorno (
         ex *12+ ): " ) ;
     while ( ArrivatoComandoUSB == 0 )
        {   }
     if ( ( VettoreRicezioneUSB[2] )== 0x2B
         )
        Tempf = ( VettoreRicezioneUSB[1] )
         - '0' ;
     else
        {
        Decinef = ( VettoreRicezioneUSB[1]
         ) - '0' ;
        Unitaf = ( VettoreRicezioneUSB[2] )
         - '0' ;
        Decinef = Decinef << 4 ;
        Decinef = Decinef & 0xF0 ;
        Unitaf = Unitaf & 0x0F ;
        Tempf = Decinef + Unitaf ;
        }
     DataEOra.Data = Tempf ;

     ArrivatoComandoUSB = 0 ;
     TrasmettiUSB ( "\nIndicare il mese ( ex
         *2+ ): " ) ;
     while ( ArrivatoComandoUSB == 0 )
        {   }
     if ( ( VettoreRicezioneUSB[2] )== 0x2B
         )
        Tempf = ( VettoreRicezioneUSB[1] )
         - '0' ;
     else
        {
        Decinef = ( VettoreRicezioneUSB[1]
         ) - '0' ;
        Unitaf = ( VettoreRicezioneUSB[2] )
         - '0' ;
        Decinef = Decinef << 4 ;
        Decinef = Decinef & 0xF0 ;
        Unitaf = Unitaf & 0x0F ;
        Tempf = Decinef + Unitaf ;
        }
     DataEOra.Mese = Tempf ;

     ArrivatoComandoUSB = 0 ;
     TrasmettiUSB ( "\nIndicare l'anno ( ex
         *18+ ): " ) ;
     while ( ArrivatoComandoUSB == 0 )
        {   }
     if ( ( VettoreRicezioneUSB[2] )== 0x2B
         )
        Tempf = ( VettoreRicezioneUSB[1] )
         - '0' ;
     else
        {
        Decinef = ( VettoreRicezioneUSB[1]
         ) - '0' ;
```

```
    Unitaf = ( VettoreRicezioneUSB[2] )
    - '0' ;
    Decinef = Decinef << 4 ;
    Decinef = Decinef & 0xF0 ;
    Unitaf = Unitaf & 0x0F ;
    Tempf = Decinef + Unitaf ;
    }
DataEOra.Anno = Tempf ;

ArrivatoComandoUSB = 0 ;
TrasmettiUSB ( "\nIndicare l'ora( ex
    *18+ ): " ) ;
while ( ArrivatoComandoUSB == 0 )
    {  }
if ( ( VettoreRicezioneUSB[2] )== 0x2B
    )
    Tempf = ( VettoreRicezioneUSB[1] )
    - '0' ;
else
    {
    Decinef = ( VettoreRicezioneUSB[1]
    ) - '0' ;
    Unitaf = ( VettoreRicezioneUSB[2] )
    - '0' ;
    Decinef = Decinef << 4 ;
    Decinef = Decinef & 0xF0 ;
    Unitaf = Unitaf & 0x0F ;
    Tempf = Decinef + Unitaf ;
    }
DataEOra.Ore = Tempf ;

ArrivatoComandoUSB = 0 ;
TrasmettiUSB ( "\nIndicare i minuti( ex
    *56+ ): " ) ;
while ( ArrivatoComandoUSB == 0 )
    {  }
if ( ( VettoreRicezioneUSB[2] )== 0x2B
    )
    Tempf = ( VettoreRicezioneUSB[1] )
    - '0' ;
else
    {
    Decinef = ( VettoreRicezioneUSB[1]
    ) - '0' ;
    Unitaf = ( VettoreRicezioneUSB[2] )
    - '0' ;
    Decinef = Decinef << 4 ;
    Decinef = Decinef & 0xF0 ;
    Unitaf = Unitaf & 0x0F ;
    Tempf = Decinef + Unitaf ;
    }
DataEOra.Minuti = Tempf ;

ArrivatoComandoUSB = 0 ;
TrasmettiUSB ( "\nIndicare i secondi(
    ex *46+ ): " ) ;
while ( ArrivatoComandoUSB == 0 )
    {  }
if ( ( VettoreRicezioneUSB[2] )== 0x2B
    )
    Tempf = ( VettoreRicezioneUSB[1] )
    - '0' ;
else
    {
    Decinef = ( VettoreRicezioneUSB[1]
    ) - '0' ;
    Unitaf = ( VettoreRicezioneUSB[2] )
    - '0' ;
    Decinef = Decinef << 4 ;
    Decinef = Decinef & 0xF0 ;
    Unitaf = Unitaf & 0x0F ;
    Tempf = Decinef + Unitaf ;
    }
DataEOra.Secondi = Tempf ;

CaricaCalendario ( DataEOra ) ;
Ritardo ( 100 ) ;
LeggiCalendario ( &DataEOra ) ;
sprintf ( Vettore, "\nSono le ore
    %x:%x:%x ", DataEOra.Ore,
    DataEOra.Minuti,
    DataEOra.Secondi ) ;
TrasmettiUSB ( Vettore ) ;
sprintf ( Vettore, "\nLa data odierna e'
    %x.%x.%x ", DataEOra.Data,
    DataEOra.Mese, DataEOra.Anno ) ;
TrasmettiUSB ( Vettore ) ;
StatoDebug = 0 ;
}

if ( StatoDebug == 10 )
{
ArrivatoComandoUSB = 0 ;
TrasmettiUSB ( "\nIndicare la
    percentuale di potenza del laser 1 ( ex
    *50+ ): " ) ;
while ( ArrivatoComandoUSB == 0 )
    {  }
Tempf = 1  ;
while ( ( Tempf < 100 ) && ( (
    VettoreRicezioneUSB[Tempf] )!=
    0x2B ) )
    {
    VettoreRicezioneUSB[(Tempf-1)] =
```

```
    VettoreRicezioneUSB[Tempf] ;
    VettoreRicezioneUSB[Tempf]      =
    0x00 ;
    Tempf++ ;
    }
Tempf = atoi(VettoreRicezioneUSB) ;
Perc1f = (double)(Tempf) ;

ArrivatoComandoUSB = 0 ;
TrasmettiUSB    (    "\nIndicare    la
   percentuale di potenza del laser 2 ( ex
   *50+ ): " ) ;
while ( ArrivatoComandoUSB == 0 )
   {  }
Tempf = 1  ;
while  (  ( Tempf  <  100  )  &&  (  (
   VettoreRicezioneUSB[Tempf]    )!=
   0x2B ) )
   {
   VettoreRicezioneUSB[(Tempf-1)]  =
   VettoreRicezioneUSB[Tempf] ;
   VettoreRicezioneUSB[Tempf]      =
   0x00 ;
   Tempf++ ;
   }
Tempf = atoi(VettoreRicezioneUSB) ;
Perc2f = (double)(Tempf) ;

ArrivatoComandoUSB = 0 ;
TrasmettiUSB    (    "\nIndicare    la
   percentuale di potenza del laser 3 ( ex
   *50+ ): " ) ;
while ( ArrivatoComandoUSB == 0 )
   {  }
Tempf = 1  ;
while  (  ( Tempf  <  100  )  &&  (  (
   VettoreRicezioneUSB[Tempf]    )!=
   0x2B ) )
   {
   VettoreRicezioneUSB[(Tempf-1)]  =
   VettoreRicezioneUSB[Tempf] ;
   VettoreRicezioneUSB[Tempf]      =
   0x00 ;
   Tempf++ ;
   }
Tempf = atoi(VettoreRicezioneUSB) ;
Perc3f = (double)(Tempf) ;

ArrivatoComandoUSB = 0 ;
TrasmettiUSB    (    "\nIndicare    la
   percentuale di potenza del laser 4 ( ex
   *50+ ): " ) ;
   while ( ArrivatoComandoUSB == 0 )
      {  }
   Tempf = 1  ;
   while  (  ( Tempf  <  100  )  &&  (  (
      VettoreRicezioneUSB[Tempf]    )!=
      0x2B ) )
      {
      VettoreRicezioneUSB[(Tempf-1)]  =
      VettoreRicezioneUSB[Tempf] ;
      VettoreRicezioneUSB[Tempf]      =
      0x00 ;
      Tempf++ ;
      }
   Tempf = atoi(VettoreRicezioneUSB) ;
   Perc4f = (double)(Tempf) ;

   CorrentiLaser ( Perc1f, Perc2f, Perc3f,
      Perc4f ) ;
        StatoDebug = 0 ;
   }

if ( StatoDebug == 11 )
   {
      int ulssb1r = 50;
      setcmdE   ("set", indev, &ev, "1a,4n",
       no, ritardo);
      getstatusE( &ev , indev);
      setcmdM      ("sut",  sb1,  &sb1cmd,
       ulsb1, ulssb1r, 'S', 10000  );
      getstatusM( &sb1stat , sb1);
      setcmdE   ("set", indev, &ev, "1c,4n",
       no, ritardo);
      getstatusE( &ev , indev);
      StatoDebug = 0 ;
   }

if ( StatoDebug == 12 )
   {
      int ulssb2r = 50;
      setcmdE   ("set", indev, &ev, "2a,4n",
       no, ritardo);
      getstatusE( &ev , indev);
      setcmdM      ("sut",  sb2,  &sb2cmd,
       ulsb2, ulssb2r, 'S', 10000  );
      getstatusM( &sb2stat , sb2);
      setcmdE   ("set", indev, &ev, "2c,4n",
       no, ritardo);
      getstatusE( &ev , indev);
      StatoDebug = 0 ;
   }

if ( StatoDebug == 13 )
```

```
    {
      int ulssb1r = 50;
      setcmdE ("set", indev, &ev, "1a,4n",
       no, ritardo);
      getstatusE( &ev , indev);
      getstatusM( &sb1stat , sb1);
      if ( sb1stat.ulatt ==
      ((sb1stat.ulmax)/2.0))
      {
        setcmdE ("set", indev, &ev,
      "1c,4n", no, ritardo);
        getstatusE( &ev , indev);
      }
      if ( sb1stat.ulatt >
      ((sb1stat.ulmax)/2.0))
      {
        int ulff=(int)( sb1stat.ulmax -
      sb1stat.ulatt);
        setcmdM ("mov", sb1, &sb1cmd,
      ulff, ulssb1r, 'I', 10000 );
        getstatusM( &sb1stat , sb1);
        setcmdE ("set", indev, &ev,
      "1c,4n", no, ritardo);
        getstatusE( &ev , indev);
      }
      if ( sb1stat.ulatt <=
      ((sb1stat.ulmax)/2.0))
      {
        int ulff=(int)( sb1stat.ulmax -
      sb1stat.ulatt);
        setcmdM ("mov", sb1, &sb1cmd,
      ulff, ulssb1r, 'S', 10000 );
        getstatusM( &sb1stat , sb1);
        setcmdE ("set", indev, &ev,
      "1c,4n", no, ritardo);
        getstatusE( &ev , indev);
      }
      StatoDebug = 0 ;
    }

  if ( StatoDebug == 14 )
    {
      int ulssb2r = 50;
      setcmdE ("set", indev, &ev, "2a,4n",
       no, ritardo);
      getstatusE( &ev , indev);
      getstatusM( &sb2stat , sb2);
      if ( sb2stat.ulatt ==
      ((sb2stat.ulmax)/2.0))
      {
        setcmdE ("set", indev, &ev,
      "2c,4n", no, ritardo);
        getstatusE( &ev , indev);
      }
      if ( sb2stat.ulatt >
      ((sb2stat.ulmax)/2.0))
      {
        int u2ff=(int)( sb2stat.ulmax -
      sb2stat.ulatt);
        setcmdM ("mov", sb2, &sb2cmd,
      u2ff, ulssb2r, 'I', 10000 );
        getstatusM( &sb2stat , sb2);
        setcmdE ("set", indev, &ev,
      "2c,4n", no, ritardo);
        getstatusE( &ev , indev);
      }
      if ( sb2stat.ulatt <=
      ((sb2stat.ulmax)/2.0))
      {
        int u2ff=(int)( sb2stat.ulmax -
      sb2stat.ulatt);
        setcmdM ("mov", sb2, &sb2cmd,
      u2ff, ulssb2r, 'S', 10000 );
        getstatusM( &sb2stat , sb2);
        setcmdE ("set", indev, &ev,
      "2c,4n", no, ritardo);
        getstatusE( &ev , indev);
      }
      StatoDebug = 0 ;
    }
  if ( StatoDebug == 16 )
    {
    // Inizializzazione della scheda sd
    LEDROSSO  = 0 ;
    LEDBLU    = 1 ;
    LEDGIALLO  = 1 ;
    while ( !FSInit( ) ) ;
    TrasmettiUSB ( "\nLa scheda MicroSD
      e' stata inizializzazta correttamente" )
      ;

    TrasmettiUSB ( "\nIdicare il nome del
      file dove salvare i dati ( es.
      *Prova.txt+ )" ) ;
    TrasmettiUSB ( "\nI caratteri * e +
      servono per indicare l'inizio e la fine
      stringa" ) ;
    while ( ArrivatoComandoUSB == 0 )
      { }
    Tempf = 1 ;
    while ( ( Tempf < 100 ) && ( (
      VettoreRicezioneUSB[Tempf]  )!=
      0x2B ) )
      {
```

```
      Vettore[(Tempf-1)] =
       VettoreRicezioneUSB[Tempf] ;
      Vettore[Tempf] = 0x00 ;
      Tempf++ ;
      }
   pointer = FSfopen ( Vettore, "r" ) ;

   if ( pointer == NULL )
      {
      LEDROSSO = 1 ;
      TrasmettiUSB ( "\nERRORE : non
       riesco ad aprire il file in lettura" ) ;
      }
   else
      {
      char Tempr[15];
      char Tempr2[24];
      memset(Tempr, '\0', sizeof(Tempr));
      memset(Tempr2, '\0',
       sizeof(Tempr2));
      FSfread(Tempr, 1, 16, pointer);
      TrasmettiUSB(Tempr);
      while(FSfeof(pointer) == 0)
      {
         FSfread(Tempr2, 1, 24, pointer);
         TrasmettiUSB(Tempr2);
      }

      SalvaDebug = 1;
      LEDROSSO = 0;
      LEDBLU = 0;
      LEDGIALLO = 1;
      }
   StatoDebug = 0;
   if (FSfclose(pointer))
      while (1);
   TrasmettiUSB("\nFile chiuso");
   }
if ( StatoDebug == 17 )
   {


   initStatM ( &sb1stat , sb1);
   initStatM ( &sb1cmd  , sb1);
   initStatM ( &sb2stat , sb2);
   initStatM ( &sb2cmd  , sb2);
   initStatM ( &sb3stat , sb3);
   initStatM ( &sb3cmd  , sb3);
   initStatM ( &sa1stat , sa1);
   initStatM ( &sa1cmd  , sa1);
   /*        INIZIO    DOSAGGIO    E
      MISCELAZIONE  */
   TrasmettiUSB("\nINIZIO DOSAGGIO
      E MISCELAZIONE\n");

   setcmdE ("set", indev, &ev,
      "1a,2a,10c", no, ritardo);
   getstatusE( &ev , indev);

   int t4,t5,t6;
   t4 = (ulsb1/ulssb1)*1000  + ritardo;
   t5 = (ulsb2/ulssb2)*1000  + ritardo;
   t6 = (ulsa1d/ulssa1)*1000 + ritardo;

   setcmdM ("mov", sb1, &sb1cmd,
      ulsb1, ulssa1, 'I', t4  );
   getstatusM( &sb1stat , sb1);

   setcmdM ("mov", sb2, &sb2cmd,
      ulsb2, ulssb2, 'I', t5  );
   getstatusM( &sb2stat , sb2);



   setcmdE ("set", indev, &ev,
      "1c,2c,10n", no, ritardo);
   getstatusE( &ev , indev);

   setcmdE ("set", indev, &ev,
      "3a,4a,7a,8a", no, ritardo);
   getstatusE( &ev , indev);

   int iv = 0 ;
   int sv = 0 ;
   int tmpvar = 1;
   while(iv < (nCicliRisciacquo) && sv <
      (nCicliRisciacquo+1) )
   {
      if ( tmpvar == 1 )
      {
         setcmdM   ("mov", sb3, &sb3cmd,
       vmisc, ulssb3, 'S', ritardo  );
         getstatusM( &sb3stat , sb3);
         sv++;
         tmpvar = 2 ;
      }else if (  tmpvar == 2 )
      {
         setcmdM   ("mov", sb3, &sb3cmd,
       vmisc, ulssb3, 'I', ritardo  );
         getstatusM( &sb3stat , sb3);
         iv++;
         tmpvar = 1 ;
      } else
      {
```

```
        Ritardo (ritardo);
        getstatusM( &sb3stat , sb3);
      }
    }

    int t7 = (ulsb3d/ulssb3)*1000 + ritardo;
    int ulsb3_1,ulsb3_2,ulsb3_3,ulsb3_4;
    ulsb3_1=4000;
    ulsb3_2=3000;
    ulsb3_3=3000;
    ulsb3_4=2000;
    setcmdM ("mov", sb3, &sb3cmd,
      ulsb3_1, ulssb3, 'S', ritardo );
    getstatusM( &sb3stat , sb3);
    setcmdM ("mov", sb3, &sb3cmd,
      ulsb3_2, ulssb3, 'I', ritardo );
    getstatusM( &sb3stat , sb3);
    setcmdM ("mov", sb3, &sb3cmd,
      ulsb3_3, ulssb3, 'S', ritardo );
    getstatusM( &sb3stat , sb3);
    setcmdM ("mov", sb3, &sb3cmd,
      ulsb3_4, ulssb3, 'I', ritardo );
    getstatusM( &sb3stat , sb3);

    setcmdE ("set", indev, &ev, "4a,8n",
      no, ritardo);
    getstatusE( &ev , indev);


    TrasmettiUSB("\nFINE DOSAGGIO E
      MISCELAZIONE\n");
    /*          FINE    DOSAGGIO    E
      MISCELAZIONE */


    StatoDebug = 0 ;
    }
  if ( StatoDebug == 18 )
    {

    double d1,d2,d3,d4,d1o,d2o,d3o,d4o;

    memset(Vettore, '\0', sizeof(Vettore));
    double
      ValoreI[nmisure][4],ValoreI2[nmisu
      re],valorebiancom, absorbance;
    double valc[4] ,mult, uno, risultatoI[4];
    uno = (double)1.00000;
    mult = (double)(2048.0000*2.0000);
    double bits = pow(2,15) ;
    int flagout,flagcount,i,ij;
    flagout = 1;
    flagcount = 0;
    double maxmis,minmis,dmis;
    double mmis[4];


    /*  INIZIO MISURA  */
    TrasmettiUSB("\nINIZIO
      MISURA\n");

    d1 = 85.00;// laser verde Cr: 85.00
    d2 = 75.00;
    d3 = 75.00;// laser rosso Zn :40.00
    d4 = 75.00;
    d1o = 0.00;
    d2o = 0.00;
    d3o = 0.00;
    d4o = 0.00;


  //                    unsigned    int
      Valoref[4][nmisure],MisureMedia[4
      ];// 4 canali per 4 metalli
    memset(Vettore, '\0', sizeof(Vettore));
    uno = (double)1.00000;
    mult = (double)(2048.0000*2.0000);
    flagout = 1;
    flagcount = 0;
    while (flagout == 1 && flagcount <
      numero_ripetizioni )
    {
      for ( i=0; i<nmisure ; i++ )
      {
        DELAY_MS(tmisuraoff);
        CorrentiLaser ( d1 , d2 , d3 , d4 ) ;
        DELAY_MS(tlaser);

        valc[0]                    =
      ((double)ADS8341_Mediato(    0
      ))/bits ;
        valc[1]                    =
      ((double)ADS8341_Mediato(    1
      ))/bits ;
        valc[2]                    =
      ((double)ADS8341_Mediato(    2
      ))/bits ;
        valc[3]                    =
      ((double)ADS8341_Mediato(    3
      ))/bits ;
        CorrentiLaser ( d1o , d2o , d3o ,
      d4o ) ;

        for (ij=0 ; ij<4 ; ij++)
```

```
   {
      if ( valc[ij] <= 1.0000 &&
valc[ij] > 0.0000  )
      {
         ValoreI[i][ij] = mult*( uno -
valc[ij] ) ; // Cr VI  uA
         risultatoI[ij] = mult*( uno -
valc[ij] ) ; // Cr VI  uA
      }
      else if ( valc[ij] > 1.0000 )
      {
         ValoreI[i][ij] = 0.0000 ; // Cr
VI  uA
         TrasmettiUSB("\nval > 1 \n");
      }
      else if ( valc[ij] < 0.0000 )
      {
         TrasmettiUSB("\nErrore
misura\n");
         ValoreI[i][ij] = 0.0000 ; // Cr
VI  uA
      }

   sprintf ( Vettore,
"\nConcentrazione di CrVI misurata :
\n CH0: %.2f  uA  ;\n CH1: %.2f  uA
;\n CH2: %.2f  uA  ;\n CH3: %.2f  uA
;\n"  ,  risultatoI[0],  risultatoI[1],
risultatoI[2], risultatoI[3]) ;
   TrasmettiUSB(Vettore);
}

for (ij=0 ; ij<4 ; ij++)
   {
   for ( i=0; i<nmisure ; i++ )
      {
         ValoreI2[i]=ValoreI[i][ij];
      }

   maxmis = maxim ( ValoreI2 ) ;
   minmis = minim ( ValoreI2 ) ;
   dmis = maxmis - minmis;
   mmis[ij]  =  (maxmis  +
minmis)/2.0000;
   absorbance                    =
log10(valorebiancom/mmis[ij]);
   }


   flagout = 0;
   flagcount++;
   }

   sprintf ( Vettore, "\nCorrente di CrVI
      misurata media : \n CH0: %.2f  uA
      ;\n CH1: %.2f  uA  ;\n CH2: %.2f  uA
      ;\n CH3: %.2f  uA  ;\n" , mmis[0],
      mmis[1], mmis[2], mmis[3] ) ;
   TrasmettiUSB(Vettore);
//    sprintf ( Vettore, "\nAssorbanza di CrVI
      calcolata :  %.2f  ;\n" , absorbance ) ;


   TrasmettiUSB("\nFINE MISURA\n");
   /*  FINE MISURA  */

   StatoDebug = 0 ;



   }
 if ( StatoDebug == 19 )
   {
   /*  INIZIO SCARICO POST MISURA
      */
   TrasmettiUSB("\nINIZIO  SCARICO
      POST MISURA\n");

   setcmdE  ("set",  indev,  &ev,
      "3c,4a,7a,8a", no, ritardo);
   getstatusE( &ev , indev);

   setcmdM  ("int", sb3, &sb3cmd, ulsb3,
      ulssb3, 'I', 10000  );
   getstatusM( &sb3stat , sb3);

   int iv = 0 ;
   int sv = 0 ;
   while(iv  <  (nCicliScarico)  &&  sv  <
      (nCicliScarico+1))
   {
      if ( sb3stat.finecI == 1 && ev.sev3 ==
      'S')
      {
         setcmdE  ("set",  indev,  &ev,
      "3a,4n", no, ritardo);
         getstatusE( &ev , indev);
         setcmdM  ("sut",  sb3,  &sb3cmd,
      ulsb3, ulssb3, 'S', 10000  );
         getstatusM( &sb3stat , sb3);
         sv++;
```

```
        }else if ( sb3stat.finecS == 1  &&
        ev.sev3 == 'A')
        {
          setcmdE      ("set",  indev,  &ev,
        "3c,4n", no, ritardo);
          getstatusE( &ev , indev);
          setcmdM    ("int",  sb3,  &sb3cmd,
        ulsb3, ulssb3, 'I', 10000  );
          getstatusM( &sb3stat , sb3);
          iv++;
        } else
        {
          Ritardo (1000);
          getstatusM( &sb3stat , sb3);
        }

      }

      setcmdE  ("set", indev, &ev, "4c,7c,8c",
         no, ritardo);
      getstatusE( &ev , indev);

      TrasmettiUSB("\nFINE      SCARICO
         POST MISURA\n");
      /*  FINE SCARICO POST MISURA */

      StatoDebug = 0 ;
//          TrasmettiUSB("\nFINE  Comando
         EV\n");
      }
   if ( StatoDebug == 20 || StatoDebug == 21
         )
   {
      MisureMain();
   }

}



void     CorrentiLaser              ( double
         PercentualeLaser1,           double
         PercentualeLaser2,           double
         PercentualeLaser3,           double
         PercentualeLaser4 )
   {
   unsigned int Tempffla ;
   double Percffla ;
   Percffla = PercentualeLaser1 * 2.0 ;
   if ( Percffla > 200.0 )
      Percffla = 200.0 ;
   Tempffla = (unsigned int)Percffla ;
   OC1R    = Tempffla ;

   Percffla = PercentualeLaser2 * 2.0 ;
   if ( Percffla > 200.0 )
      Percffla = 200.0 ;
   Tempffla = (unsigned int)Percffla ;
   OC2R    = Tempffla ;

   Percffla = PercentualeLaser3 * 2.0 ;
   if ( Percffla > 200.0 )
      Percffla = 200.0 ;
   Tempffla = (unsigned int)Percffla ;
   OC3R    = Tempffla ;

   Percffla = PercentualeLaser4 * 2.0 ;
   if ( Percffla > 200.0 )
      Percffla = 200.0 ;
   Tempffla = (unsigned int)Percffla ;
   OC4R    = Tempffla ;
   }

void CalcolaCRC16          ( unsigned char*
         Vettoref,        unsigned      char
         NumeroCaratterif,  unsigned      char
         CRCF[2] )
   {
   unsigned int Crcf, Carf, If ;
   Crcf = 0 ;
   for ( If = 0 ; If < NumeroCaratterif ; If++ )
      {
      Carf = *Vettoref++ ;
      Carf = Carf & 0x00FF ;
      Crcf = (unsigned char)(Crcf >>8) | (Crcf
         << 8) ;
      Crcf = Crcf ^ Carf ;
      Crcf =  Crcf  ^  ((unsigned  char)(Crcf  &
         0xFF) >> 4 ) ;
      Crcf = Crcf ^ ((Crcf << 8 ) << 4 ) ;
      Crcf = Crcf ^ ((Crcf & 0xFF) << 4 ) <<
         1 ;
      }
   If = Crcf ;
   If = If >> 8 ;
   CRCF[0] = (char)If ;  //LSB
   If = Crcf ;
   If = If & 0x00FF ;
   CRCF[1] = (char)If ;  //MSB
   }

void   getstatusM          ( struct motosiringa
         *data, int ind )
{
```

```
char tipo;
char stat[8];
char errc[40];
char tppos[9];
float ulatt;
char statmov;
float ulmax;
char crch;
char crcl;
char mess[20];
int i;
int finecS;
int finecI;
float psa;
unsigned char CRCF[2];
data->okTR = 1 ;
memset(stat, (char)(0), sizeof(stat));
memset(errc, (char)(0), sizeof(errc));
memset(tppos, (char)(0), sizeof(tppos));

for ( i = 0 ; i<20; i++)
{
   mess[i] = (char)(0);
   data->strstatus[i] = (char)(0);
   data->strcmd[i] = (char)(0);
}
mess[0] = '*'                    ;
mess[1] = 'M'                     ;
          ;
mess[2] = (char)ind                    ;
mess[3] = 'S'                     ;// Comando
       Stato

CalcolaCRC16( mess,17, CRCF );

mess[17] = CRCF[0];//MSB CRC  0x81;//
mess[18] = CRCF[1];//LSB CRC  0xAD;//
mess[19] = (char)(0x0D);

TrasmettiNET2M( mess ) ;
Ritardo ( 201 ) ;

// check pacchetto Txed Rx
char RxCRCL, RxCRCH ;

char rx[20];
while(data->okTR == 1)
{
   for ( i = 0 ; i<20; i++)
   {
      if (i<17)
         rx[i] = VettoreRicezioneNET2[i];
      else
         rx[i] = (char)(0);
   }
   RxCRCH = VettoreRicezioneNET2[17]
      ;
   RxCRCL = VettoreRicezioneNET2[18]
      ;

   unsigned char CRCFcalc[2];
   CalcolaCRC16( rx,17, CRCFcalc );
   char H,L;
   H = (char)CRCFcalc[0];
   L = (char)CRCFcalc[1];
   if (H==RxCRCH && L==RxCRCL)
      data->okTR = 0;
   else   if   (CRCFcalc[0]!=RxCRCH   ||
      CRCFcalc[1]!=RxCRCL)
   {
      //TrasmettiUSB("\nCORRUZIONE
       Tx o Rx Motosiringa");
      data->okTR = 1;
      TrasmettiNET2M( mess ) ;
      Ritardo ( 201 ) ;
   }
}
//fine check
for (i = 0; i<20 ; i++)
{

   data->strstatus[i]                    =
      VettoreRicezioneNET2[i];
}

tipo =  VettoreRicezioneNET2[1]     ;


if (VettoreRicezioneNET2[3]=='K')
{
   strcpy(stat, "Normale");

}else if (VettoreRicezioneNET2[3]=='E')
{
   strcpy(stat, "Errore");

}
int par1;
par1 =  (int)VettoreRicezioneNET2[4] ;
if (par1==0)
{
   strcpy(errc  ,  "Normale"  );
}else if (par1==1)
{
```

```
    strcpy(errc , "D: mov richiesta ne
        inizione ne suzione" ) ;
}else if (par1==2)
{
    strcpy(errc , "D: richiesta ul sup al
        max" );
}else if (par1==3)
{
    strcpy(errc , "AVD: richiesta vel nulla"
        ) ;
}else if (par1==4)
{
    strcpy(errc , "AVD: richiesta vel sup al
        max" );
}else if (par1==5)
{
    strcpy(errc , "D: max capacita
        raggiunta" );
}
if (VettoreRicezioneNET2[5]=='I')
{
    strcpy(tppos , "Presunta" ) ;
}else if (VettoreRicezioneNET2[5]=='S')
{
    strcpy(tppos , "Certa" );
}
int plsb, pmsb;
plsb = ( (int)VettoreRicezioneNET2[7] ) &
        0x00FF          ;//LSB pos
pmsb = (( (int)VettoreRicezioneNET2[6]
        )<<8 ) & 0xFF00     ;//MSB pos

float p1,p2;
p1=(float)plsb;
p2=(float)pmsb;

ulatt = p1+p2 ;

statmov = VettoreRicezioneNET2[8] ;

if ( statmov == 'S')
    i=0;
  //TrasmettiUSB ( "\nSir Suzione" ) ;
else if ( statmov == 'I')
    i=0;
  //TrasmettiUSB ( "\nSir Iniezione" ) ;
else if ( statmov == 'F')
{
    //TrasmettiUSB ( "\nSir Ferma" ) ;
    if
        (VettoreRicezioneNET2[5]=='S')//p
        osizione certa
    {
        writePosMinFile( ind , ulatt );
    }else                                   if
        (VettoreRicezioneNET2[5]=='I')//po
        sizione presunta
    {
        char* sm;
        int ritardom;
        float fmulm, fmulsm;
        char dism;

        sm      = "set"     ;
        ritardom =  101         ;// indipendente
        dal valore
        fmulm    = 0.0     ;// indipendente dal
        valore
        fmulsm   = 0.0         ;// indipendente
        dal valore
        dism     = 'S'      ;// indipendente dal
        valore

        setcmdM(sm , (char)ind , &cmdset ,
        fmulm , fmulsm , dism , ritardom );
    }
}

int cmlsb,cmmsb;
cmlsb = ( (int)VettoreRicezioneNET2[16]
        ) & 0x00FF          ;//LSB ctot
cmmsb                       =                   ((
        (int)VettoreRicezioneNET2[15]
        )<<8 ) & 0xFF00    ;//MSB ctot

float t1,t2;
t1=(float)cmlsb;
t2=(float)cmmsb;

ulmax = t1 + t2  ;

finecI = c2i ( VettoreRicezioneNET2[9] );
finecS = c2i ( VettoreRicezioneNET2[10]
        );

int prlsb,prmsb;
prlsb = ( (int)VettoreRicezioneNET2[12] )
        & 0x00FF          ;//LSB ctot
prmsb                       =                   ((
        (int)VettoreRicezioneNET2[11]
        )<<8 ) & 0xFF00    ;//MSB ctot

float pr1,pr2;
pr1=(float)prlsb;
```

```
   pr2=(float)prmsb;

   psa =( pr1 + pr2 )*(160.0000/5000.0000) +
         1.0132; // 1.0132500411216163

   crch = VettoreRicezioneNET2[17] ;
   crcl = VettoreRicezioneNET2[18] ;

   data->tipo=tipo       ;
   data->ind=idM (ind)       ;
   data->ulatt=ulatt     ;
   data->statmov=statmov  ;
   data->ulmax=ulmax     ;
   data->crch=crch       ;
   data->crcl=crcl       ;
   data->finecS=finecS    ;
   data->finecI=finecI    ;
   data->psa=psa         ;

   for(i=0;i<8;i++)
      data->stat[i]=stat[i];
//   strcpy(data->stat, stat);
   for(i=0;i<40;i++)
      data->errc[i]=errc[i];
//   strcpy(data->errc, errc);
   for(i=0;i<9;i++)
      data->tppos[i]=tppos[i];
//   strcpy(data->tppos, tppos);
}

void   initStatM             ( struct motosiringa
         *data, int ind )
{
   int i;
   for ( i = 0 ; i<20; i++)
   {
      data->strstatus[i] = (char)(0);
      data->strcmd[i] = (char)(0);
   }

   data->tipo='M';
   char indc;
   indc = idM (ind);
   data->ind=indc;
   char z;
   z = (char)(0);
   memset(data->stat, (char)(0), sizeof(data-
         >stat));
   memset(data->errc, (char)(0), sizeof(data-
         >errc));
   memset(data->tppos,              (char)(0),
         sizeof(data->tppos));
   data->ulatt=0.0;
   data->statmov=z;
   data->ulmax=0.0;
   data->crch=z;
   data->crcl=z;
   data->okTR=1;
   data->finecS=0;
   data->finecI=0;
   data->psa=0.0;
}

   // indirizzo
void   setcmdM(char*  s  ,int  ind,  struct
         motosiringa *data, float fmul, float
         fmuls , char dis, int ritardo)
{
   unsigned char CRCF[2];
   int mul = (int)fmul ;
   int muls = (int)fmuls ;
   char mess[20];
   int i;
   for ( i = 0 ; i<20; i++)
   {
      mess[i] = (char)(0);
      data->strstatus[i] = (char)(0);
      data->strcmd[i] = (char)(0);
   }
   mess[0] = '*'                    ;
   mess[1] = 'M'                     ;
   mess[2] =  (char)(ind)              ;
   data->ind=idM(ind);
   char st[3];
   char pot;
   pot=0;
   data->okTR = 1 ;
   while(pot<3)
   {
      st[pot]=s[pot];
      pot++;
   }
   if (stringcomp("mov",st,3)==1)
   {
      mess[3] = 'D'                  ;// Comando
         movimentazione
      mess[4] = dis                        ;// Par 1
         Iniezione
      mess[5] = (char)((mul>>8) & 0x00FF  )
         ;//MSB Par 2
      mess[6] = (char)(mul & 0x00FF       )
         ;//LSB Par 3
      mess[7] = (char)((muls>>8) & 0x00FF )
         ;//MSB Par 4
```

```
    mess[8] = (char)(muls & 0x00FF       )
        ;//LSB Par 5

  }
  else if (stringcomp("arr",st,3)==1)
  {
    mess[3] = 'A'                  ;// Comando
        arresto
  }
  else if (stringcomp("int",st,3)==1)
  {
    mess[3] = 'V'                  ;// Comando
        iniezione totale
    mess[4] = (char)((muls>>8) & 0x00FF )
        ;//MSB Par 1
    mess[5] = (char)(muls & 0x00FF       )
        ;//LSB Par 2
  }
  else if (stringcomp("sut",st,3)==1)
  {
    mess[3] = 'A'                  ;// Comando
        suzione totale
    mess[4] = (char)((muls>>8) & 0x00FF )
        ;//MSB Par 1
    mess[5] = (char)(muls & 0x00FF       )
        ;//LSB Par 2
  }
  else if (stringcomp("set",st,3)==1)
  {
    LEDROSSO    = 0 ;
    LEDBLU      = 1 ;
    LEDGIALLO   = 1 ;
    while ( !FSInit( ) ) ;
       //TrasmettiUSB ( "\nLa scheda
        MicroSD e' stata inizializzata
        correttamente" ) ;
    char fileM[7];
    fileM[0]= 'M' ;fileM[2]= '.' ;fileM[3]=
        't' ;fileM[4]= 'x' ;fileM[5]= 't' ;
    fileM[1]= idM(ind);
    fileM[6]= 0x00;
    pointerM = FSfopen ( fileM, "r" ) ;
    int ultoset;
    if ( pointerM == NULL )
       {
       LEDROSSO = 1 ;
       //TrasmettiUSB ( "\nERRORE : non
        riesco ad aprire il file in lettura" ) ;
       ultoset=0;
       }
    else
       {
       char Tempr[11];
       memset(Tempr, '\0', sizeof(Tempr));
       FSfread(Tempr, 1, 12, pointerM);

       int ultoset;
       ultoset = str2num (Tempr);
       //ultoset = strtol(Tempr, &ptr, 10);
       if (FSfclose(pointerM))
          while (1);
       //TrasmettiUSB("\nFile chiuso");
       LEDROSSO = 0;
       LEDBLU = 0;
       LEDGIALLO = 1;
       }

    mess[3] = 'Z'                  ;// Comando
        set posizione
    mess[4] = (char)((ultoset>>8) & 0x00FF
        )  ;//MSB Par 1
    mess[5] = (char)(ultoset & 0x00FF       )
        ;//LSB Par 2
  }

  CalcolaCRC16( mess,17, CRCF );
  mess[17] = CRCF[0];//MSB CRC  0x81;//
  mess[18] = CRCF[1];//LSB CRC  0xAD;//
  mess[19] = (char)(0x0D);

  for ( i = 0 ; i<20; i++)
     data->strcmd[i] = mess[i];

  TrasmettiNET2M( mess ) ;

  memset(mess, (char)(0), sizeof(mess));
  mess[0] = '*'                  ;
  mess[1] = 'M'                  ;
  mess[2] = (char)ind                  ;
  mess[3] = 'S'                  ;// Comando
        Stato

  CalcolaCRC16( mess,17, CRCF );

  mess[17] = CRCF[0];//MSB CRC  0x81;//
  mess[18] = CRCF[1];//LSB CRC  0xAD;//
  mess[19] = (char)(0x0D);
  Ritardo ( 1001 ) ;
  while (VettoreRicezioneNET2[8] != 'F')
  {
     TrasmettiNET2M( mess ) ;
     Ritardo ( 1001 ) ;
  }
  // check pacchetto Txed Rx
```

```
  char RxCRCL, RxCRCH ;
  char rx[20];
  while(data->okTR == 1)
  {
    for ( i = 0 ; i<20; i++)
    {
      if (i<17)
        rx[i] = VettoreRicezioneNET2[i];
      else
        rx[i] = (char)(0);
    }
    unsigned char CRCFcalc[2];
    CalcolaCRC16( rx,17, CRCFcalc );
    char H,L;
    RxCRCH = VettoreRicezioneNET2[17]
      ;
    RxCRCL = VettoreRicezioneNET2[18]
      ;

    H = (char)CRCFcalc[0];
    L = (char)CRCFcalc[1];
    if (H==RxCRCH && L==RxCRCL)
      data->okTR = 0;
    else if (CRCFcalc[0]!=RxCRCH ||
      CRCFcalc[1]!=RxCRCL)
    {
      //TrasmettiUSB("\nCORRUZIONE
        Tx o Rx MOTOSIRINGA
        COMANDO");
      data->okTR = 1;
      TrasmettiNET2M( mess ) ;
      Ritardo ( 201 ) ;
    }
  }

}

void writePosMinFile (int inds, float
      ulattivo)
{
  LEDROSSO  = 0 ;
  LEDBLU    = 0 ;
  LEDGIALLO  = 1 ;
  while ( !FSInit( ) ) ;
    //TrasmettiUSB ( "\nLa scheda
      MicroSD e' stata inizializzazta
      correttamente" ) ;
  LEDGIALLO = 0 ;
  /*Salvataggio nel file della posizione della
      siringa*/
  char fileM[7];// 6
  fileM[0]= 'M' ;fileM[2]= '.' ;fileM[3]= 't'
      ;fileM[4]= 'x' ;fileM[5]= 't' ;
  fileM[1]= idM (inds);
  fileM[6]= 0x00;

  pointerM = FSfopen ( fileM, "w" ) ;
  if ( pointerM == NULL )
  {
    LEDROSSO = 1 ;
    //TrasmettiUSB ( "\nERRORE : non
      riesco ad aprire il file in scrittura" ) ;
  }
  else
  {
    char pos[20];
    char Indiceff;
    int i;
    for (i=0;i<20;i++)
      pos[i]= 0x00;
    sprintf ( pos, "%f", ulattivo ) ;
    Indiceff = 0 ;
    while ( pos[Indiceff] != 0x00 )
      Indiceff++ ;
    if ( FSfwrite ( pos, 1, Indiceff, pointerM
      ) != Indiceff )
      while ( 1 ) ;

    if (FSfclose(pointerM))
      while (1);

  }
  /*Fine salvataggio*/

}

void initStatE ( struct elettrovalvole
      *data, int ind )
{
  int i;
  for ( i = 0 ; i<20; i++)
  {
    data->strstatus[i] = (char)(0);
    data->strcmd[i] = (char)(0);
  }

  data->tipo='E';
  char indc;
  indc = idM (ind);
  data->ind=indc;
  char z;
  z = (char)(0);

  memset(data->stat, (char)(0), sizeof(data-
```

```
        >stat));
    memset(data->errc, (char)(0), sizeof(data-
        >errc));

    data->cev1 = 'S' ;
    data->cev2 = 'S' ;
    data->cev3 = 'S' ;
    data->cev4 = 'S' ;
    data->cev5 = 'S' ;
    data->cev6 = 'S' ;
    data->cev7 = 'S' ;
    data->cev8 = 'S' ;
    data->cev9 = 'S' ;
    data->cev10= 'S' ;
    // 'A' = on ; 'S'= off
    data->sev1 = 'S' ;
    data->sev2 = 'S' ;
    data->sev3 = 'S' ;
    data->sev4 = 'S' ;
    data->sev5 = 'S' ;
    data->sev6 = 'S' ;
    data->sev7 = 'S' ;
    data->sev8 = 'S' ;
    data->sev9 = 'S' ;
    data->sev10= 'S' ;

    data->crch=z;
    data->crcl=z;
    data->okTR=1;
}
void   getstatusE          ( struct elettrovalvole
          *data, int ind )
{
    char tipo;
    char stat[8];
    char errc[40];
    char crch;
    char crcl;
    char mess[20];
    int i;
    unsigned char CRCF[2];
    data->okTR = 1 ;
    memset(stat, (char)(0), sizeof(stat));
    memset(errc, (char)(0), sizeof(errc));

    for ( i = 0 ; i<20; i++)
    {
       mess[i] = (char)(0);
       data->strstatus[i] = (char)(0);
       data->strcmd[i] = (char)(0);
    }
    mess[0] = '*'                    ;
    mess[1] = 'E'                    ;
//    char indcm;
//    indcm = idM (ind)                ;
    mess[2] = (char)ind              ;
    mess[3] = 'S'                    ;// Comando
          Stato

    CalcolaCRC16( mess,17, CRCF );

    mess[17] = CRCF[0];//MSB CRC  0x81;//
    mess[18] = CRCF[1];//LSB CRC  0xAD;//
    mess[19] = (char)(0x0D);

    TrasmettiNET1M( mess ) ;
    Ritardo ( 101 ) ;

    char mx[20];
    for (i = 0; i<20 ; i++)
    {
       mx[i]=VettoreRicezioneNET1[i];
       data->strcmd[i] = mess[i];
       data->strstatus[i]                =
          VettoreRicezioneNET1[i];
    }

    tipo =  mx[1]     ;

    char indc;
    int indi;
    indi =(int)mx[2] ;
    indc = idM ( indi );
    ind  =  indc ;
    if (mx[3]=='K')
    {
       strcpy(stat, "Normale");
       //stat  =  "Normale"  ;
    }else if (mx[3]=='E')
    {
       strcpy(stat, "Errore");
       //stat  =  "Errore"  ;
    }
    int par1;
    par1 =  (int)mx[4] ;
    if (par1==0)
    {
       strcpy(errc  ,  "Normale"  );
    }else if (par1==0x01)
    {                                          //
       strcpy(errc  ,  "EV1:Stato non permesso"
          ) ;
    }else if (par1==0x02)
    {
```

```
    strcpy(errc , "EV2:Stato non permesso"
        );
}else if (par1==0x03)
{
    strcpy(errc , "EV3:Stato non permesso"
        ) ;
}else if (par1==0x04)
{
    strcpy(errc , "EV4:Stato non permesso"
        );
}else if (par1==0x05)
{
    strcpy(errc , "EV5:Stato non permesso"
        );
}else if (par1==0x06)
{
    strcpy(errc , "EV6:Stato non permesso"
        );
}else if (par1==0x07)
{
    strcpy(errc , "EV7:Stato non permesso"
        );
}else if (par1==0x08)
{
    strcpy(errc , "EV8:Stato non permesso"
        );
}else if (par1==0x09)
{
    strcpy(errc , "EV9:Stato non permesso"
        );
}else if (par1==0x0A)
{
    strcpy(errc , "EV10:Stato non
        permesso" );
}else if (par1==0x0B)
{
    strcpy(errc , "Cmd tmp EV non
        presente" );
}else if (par1==0x0C)
{
    strcpy(errc , "Cmd tmp nullo" );
}
data->sev1  = mx[5]  ;
data->sev2  = mx[6]  ;
data->sev3  = mx[7]  ;
data->sev4  = mx[8]  ;
data->sev5  = mx[9]  ;
data->sev6  = mx[10] ;
data->sev7  = mx[11] ;
data->sev8  = mx[12] ;
data->sev9  = mx[13] ;
data->sev10 = mx[14] ;
crch = mx[17] ;
crcl = mx[18] ;

data->tipo=tipo       ;
data->ind=(char)ind       ;
data->crch=crch       ;
data->crcl=crcl       ;

for(i=0;i<8;i++)
    data->stat[i]=stat[i];

for(i=0;i<40;i++)
    data->errc[i]=errc[i];

// check pacchetto Txed Rx
char RxCRCL, RxCRCH ;
char rx[20];
while(data->okTR == 1)
{
    for ( i = 0 ; i<20; i++)
    {
        if (i<17)
            rx[i] = VettoreRicezioneNET1[i];
        else
            rx[i] = (char)(0);
    }
    unsigned char CRCFcalc[2];
    CalcolaCRC16( rx,17, CRCFcalc );
    char H,L;
    RxCRCH = mx[17] ;
    RxCRCL = mx[18] ;

    H = (char)CRCFcalc[0];
    L = (char)CRCFcalc[1];
    if (H==RxCRCH && L==RxCRCL)
        data->okTR = 0;
    else if (CRCFcalc[0]!=RxCRCH ||
        CRCFcalc[1]!=RxCRCL)
    {
        //TrasmettiUSB("\nCORRUZIONE
        Tx o Rx EV");
        data->okTR = 1;
        TrasmettiNET1M( mess ) ;
        Ritardo ( 201 ) ;
    }
}


}
```

```
void     setcmdE       (char s[3], int ind, struct
         elettrovalvole *data, char* vecev, int
         tempoms , int ritardo)
{
   unsigned char CRCF[2];
   char mess[20];
   int counternum,counteracn,l,jj,kk,virgc;
   l=0;counternum=0;counteracn=0;
   data->okTR = 1 ;
   char strin[100];
   memset(strin, 0x00, sizeof(strin));

   strcpy(strin,vecev);

   char vecnum[100];
   char vecacn[100];
   memset(vecnum, 0x00, sizeof(vecnum));
   memset(vecacn, 0x00, sizeof(vecacn));

   l=0;jj=0;virgc=0;
   while (strin[l] != 0x00)
   {
      if (strin[l] != 'a' && strin[l] != 'c' &&
          strin[l] != 'n' )
      {
         vecnum[jj]=strin[l];
         jj++;
      }
      l++;
   }
   l=0;kk=0;
   while (strin[l] != 0x00)
   {
      if ( strin[l] == 'a' || strin[l] == 'c' || strin[l]
           == 'n' )
      {
         //sprintf(vecacn[kk],"%c",strin[l]);
         vecacn[kk]=strin[l];
         kk++;
      }
      l++;
   }
   int ll=0;
   char vv[100];
   int intvecnum[100];
   memset(intvecnum,          0          ,
          sizeof(intvecnum));
   const char tok[] = ",";
   char * tmp = (char *)vecnum;
   size_t count;
   for (count=0; tmp[count]; tmp[count] ==
          tok[0] ? count++ : * tmp++)
   {       }
   tmp = (char *)vecnum;
   size_t i;
   for (i = 0, l = 0; i < count; i++) {
      l = strcspn (tmp, tok);
      if (l == 0) {
      } else {
         sprintf(vv,"%.*s.", l, tmp);
         intvecnum[ll]= str2num(vv);
         ll++;
      }
      tmp += sizeof(char) * (l + 1);
   }
   sprintf(vv,"%s.", tmp);
   intvecnum[ll]= str2num(vv);

   //ok riparti da qui: vai strutture cevX con if
          su vecacn (usa anche intvecnum)
   char
          cev1,cev2,cev3,cev4,cev5,cev6,cev7
          ,cev8,cev9,cev10; // 0 = on ; 1 = off
          cev1  = 'N';
   cev2  = 'N';
   cev3  = 'N';
   cev4  = 'N';
   cev5  = 'N';
   cev6  = 'N';
   cev7  = 'N';
   cev8  = 'N';
   cev9  = 'N';
   cev10 = 'N';
   //int
          sev1,sev2,sev3,sev4,sev5,sev6,sev7,
          sev8,sev9,sev10;  // 0 = on ; 1 = off
   int ch;
   ch=0;
   while(intvecnum[ch] != 0 )
   {
      switch (intvecnum[ch])
      {
         case 1:
            if (vecacn[ch]=='a')
               cev1 = 'A' ;
            else if (vecacn[ch]=='c')
               cev1 = 'S' ;
            else if (vecacn[ch]=='n')
               cev1 = 'N' ;
            break;
         case 2:
            if (vecacn[ch]=='a')
               cev2 = 'A' ;
            else if (vecacn[ch]=='c')
```

```
            cev2 = 'S' ;
        else if (vecacn[ch]=='n')
            cev2 = 'N' ;
        break;
    case 3:
        if (vecacn[ch]=='a')
            cev3 = 'A' ;
        else if (vecacn[ch]=='c')
            cev3 = 'S' ;
        else if (vecacn[ch]=='n')
            cev3 = 'N' ;
        break;
    case 4:
        if (vecacn[ch]=='a')
            cev4 = 'A' ;
        else if (vecacn[ch]=='c')
            cev4 = 'S' ;
        else if (vecacn[ch]=='n')
            cev4 = 'N' ;
        break;
    case 5:
        if (vecacn[ch]=='a')
            cev5 = 'A' ;
        else if (vecacn[ch]=='c')
            cev5 = 'S' ;
        else if (vecacn[ch]=='n')
            cev5 = 'N' ;
        break;
    case 6:
        if (vecacn[ch]=='a')
            cev6 = 'A' ;
        else if (vecacn[ch]=='c')
            cev6 = 'S' ;
        else if (vecacn[ch]=='n')
            cev6 = 'N' ;
        break;
    case 7:
        if (vecacn[ch]=='a')
            cev7 = 'A' ;
        else if (vecacn[ch]=='c')
            cev7 = 'S' ;
        else if (vecacn[ch]=='n')
            cev7 = 'N' ;
        break;
    case 8:
        if (vecacn[ch]=='a')
            cev8 = 'A' ;
        else if (vecacn[ch]=='c')
            cev8 = 'S' ;
        else if (vecacn[ch]=='n')
            cev8 = 'N' ;
        break;
    case 9:
        if (vecacn[ch]=='a')
            cev9 = 'A' ;
        else if (vecacn[ch]=='c')
            cev9 = 'S' ;
        else if (vecacn[ch]=='n')
            cev9 = 'N' ;
        break;
    case 10:
        if (vecacn[ch]=='a')
            cev10 = 'A' ;
        else if (vecacn[ch]=='c')
            cev10 = 'S' ;
        else if (vecacn[ch]=='n')
            cev10 = 'N' ;
        break;
    default:
        //TrasmettiUSB("Errore set EV");
        ch--;
        break;
   }
   ch++;
}
data->cev1  = cev1  ;
data->cev2  = cev2  ;
data->cev3  = cev3  ;
data->cev4  = cev4  ;
data->cev5  = cev5  ;
data->cev6  = cev6  ;
data->cev7  = cev7  ;
data->cev8  = cev8  ;
data->cev9  = cev9  ;
data->cev10 = cev10 ;



for ( i = 0 ; i<20; i++)
{
   mess[i] = (char)(0);
   data->strstatus[i] = (char)(0);
   data->strcmd[i] = (char)(0);
}
mess[0] = '*'                    ;
mess[1] = 'E'                    ;
                ;
mess[2] = (char)ind                      ;
data->ind=(char)ind ;
char st[3];
char pot;
pot=0;
while(pot<3)
{
```

```
    st[pot]=s[pot];
    pot++;
}
if (stringcomp("set",st,3)==1)
{
    mess[3] = 'C'                    ;// Comando
        apri/chiudi/niente
    mess[4]  = cev1     ;//Par 1
    mess[5]  = cev2     ;//Par 2
    mess[6]  = cev3     ;//Par 3
    mess[7]  = cev4     ;//Par 4
    mess[8]  = cev5     ;//Par 5
    mess[9]  = cev6     ;//Par 6
    mess[10] = cev7     ;//Par 7
    mess[11] = cev8     ;//Par 8
    mess[12] = cev9     ;//Par 9
    mess[13] = cev10    ;//Par 10
}
else if (stringcomp("stm",st,3)==1)
{
    ch=0;
    char cevt;
    if (vecacn[ch]=='a')
        cevt = 'A' ;
    else if (vecacn[ch]=='c')
        cevt = 'S' ;
    else if (vecacn[ch]=='n')
        cevt = 'N' ;
    mess[3] = 'T'                                ;//
        Comando A/S per unperiodo in ms
    mess[4] = (char)( intvecnum[ch]          )
        ;// Par 1 Iniezione
    mess[5] =           cevt                ;// Par
        2
    mess[6]  =  (char)((tempoms>>8)  &
        0x00FF )    ;//MSB Par 3
    mess[7]  =  (char)(tempoms  &  0x00FF
        )    ;//LSB Par 4

}
else if (stringcomp("get",st,3)==1)
{
    mess[3] = 'S'                    ;// Comando
        Stato
}


CalcolaCRC16( mess,17, CRCF );
mess[17] = CRCF[0];//MSB CRC  0x81;//
mess[18] = CRCF[1];//LSB CRC  0xAD;//
mess[19] = (char)(0x0D);

data->crch = mess[17]  ;
data->crcl = mess[18]  ;

for ( i = 0 ; i<20; i++)
   data->strcmd[i] = mess[i];

data->tipo = mess[1]              ;
TrasmettiNET1M( mess ) ;
Ritardo ( ritardo ) ;
// check pacchetto Txed Rx
char RxCRCL, RxCRCH ;
char rx[20];
while(data->okTR == 1)
{
   for ( i = 0 ; i<20; i++)
   {
      if (i<17)
         rx[i] = VettoreRicezioneNET1[i];
      else
         rx[i] = (char)(0);
   }
   unsigned char CRCFcalc[2];
   CalcolaCRC16( rx,17, CRCFcalc );
   char H,L;
   RxCRCH = VettoreRicezioneNET1[17]
      ;
   RxCRCL = VettoreRicezioneNET1[18]
      ;

   H = (char)CRCFcalc[0];
   L = (char)CRCFcalc[1];
   if (H==RxCRCH && L==RxCRCL)
      data->okTR = 0;
   else  if  (CRCFcalc[0]!=RxCRCH  ||
      CRCFcalc[1]!=RxCRCL)
   {
      //TrasmettiUSB("\nCORRUZIONE
       Tx o Rx EV");
      data->okTR = 1;
      TrasmettiNET1M( mess ) ;
      Ritardo ( ritardo ) ;
   }
}

if
      (stringcomp("get",st,3)==1)//postpro
      cessing dello status
{
   char mx[20];
   char stat[8];
   char errc[40];
   memset(stat, (char)(0), sizeof(stat));
```

```
memset(errc, (char)(0), sizeof(errc));
int ii;
for (ii = 0; ii<20 ; ii++)
{
   mx[ii]=VettoreRicezioneNET1[ii];
   data->strstatus[ii]                    =
    VettoreRicezioneNET1[ii];
}

if (mx[3]=='K')
{
   strcpy(stat, "Normale");
   //stat  =  "Normale"  ;
}else if (mx[3]=='E')
{
   strcpy(stat, "Errore");
   //stat  =  "Errore"  ;
}
int par1;
par1 =  (int)mx[4] ;
if (par1==0)
{
   strcpy(errc  ,  "Normale"  );
}else if (par1==0x01)
{                                          //
   strcpy(errc      ,      "EV1:Stato    non
    permesso" ) ;
}else if (par1==0x02)
{
   strcpy(errc      ,      "EV2:Stato    non
    permesso"  );
}else if (par1==0x03)
{
   strcpy(errc      ,      "EV3:Stato    non
    permesso" ) ;
}else if (par1==0x04)
{
   strcpy(errc      ,      "EV4:Stato    non
    permesso"  );
}else if (par1==0x05)
{
   strcpy(errc      ,      "EV5:Stato    non
    permesso"  );
}else if (par1==0x06)
{
   strcpy(errc      ,      "EV6:Stato    non
    permesso"  );
}else if (par1==0x07)
{
   strcpy(errc      ,      "EV7:Stato    non
    permesso"  );
}else if (par1==0x08)
{
   strcpy(errc      ,      "EV8:Stato    non
    permesso"  );
}else if (par1==0x09)
{
   strcpy(errc      ,      "EV9:Stato    non
    permesso"  );
}else if (par1==0x0A)
{
   strcpy(errc      ,      "EV10:Stato   non
    permesso"  );
}else if (par1==0x0B)
{
   strcpy(errc   ,   "Cmd tmp EV non
    presente"  );
}else if (par1==0x0C)
{
   strcpy(errc  ,  "Cmd tmp nullo"  );
}

data->sev1  = mx[5]  ;
data->sev2  = mx[6]  ;
data->sev3  = mx[7]  ;
data->sev4  = mx[8]  ;
data->sev5  = mx[9]  ;
data->sev6  = mx[10] ;
data->sev7  = mx[11] ;
data->sev8  = mx[12] ;
data->sev9  = mx[13] ;
data->sev10 = mx[14] ;

for(i=0;i<8;i++)
   data->stat[i]=stat[i];

for(i=0;i<40;i++)
   data->errc[i]=errc[i];

}else  //      per tutti gli altri comandi il crc
       rimane quello inviato e non di stato
{
   char mx[20];
   char stat[8];
   char errc[40];
   memset(stat, (char)(0), sizeof(stat));
   memset(errc, (char)(0), sizeof(errc));
   int ii;
   for (ii = 0; ii<20 ; ii++)
   {
      mx[ii]=VettoreRicezioneNET1[ii];
      data->strstatus[ii]                    =
       VettoreRicezioneNET1[ii];
   }
```

```
if (mx[3]=='K')
{
   strcpy(stat, "Normale");
   //stat = "Normale" ;
}else if (mx[3]=='E')
{
   strcpy(stat, "Errore");
   //stat = "Errore" ;
}
int par1;
par1 = (int)mx[4] ;
if (par1==0)
{
   strcpy(errc , "Normale" );
}else if (par1==0x01)
{                                          //
   strcpy(errc      ,      "EV1:Stato   non
    permesso" ) ;
}else if (par1==0x02)
{
   strcpy(errc      ,      "EV2:Stato   non
    permesso" );
}else if (par1==0x03)
{
   strcpy(errc      ,      "EV3:Stato   non
    permesso" ) ;
}else if (par1==0x04)
{
   strcpy(errc      ,      "EV4:Stato   non
    permesso" );
}else if (par1==0x05)
{
   strcpy(errc      ,      "EV5:Stato   non
    permesso" );
}else if (par1==0x06)
{
   strcpy(errc      ,      "EV6:Stato   non
    permesso" );
}else if (par1==0x07)
{
   strcpy(errc      ,      "EV7:Stato   non
    permesso" );
}else if (par1==0x08)
{
   strcpy(errc      ,      "EV8:Stato   non
    permesso" );
}else if (par1==0x09)
{
   strcpy(errc      ,      "EV9:Stato   non
    permesso" );
}else if (par1==0x0A)
{
   strcpy(errc      ,      "EV10:Stato   non
    permesso" );
}else if (par1==0x0B)
{
   strcpy(errc      ,      "Cmd tmp EV non
    presente" );
}else if (par1==0x0C)
{
   strcpy(errc , "Cmd tmp nullo" );
}
data->sev1  = mx[5]  ;
data->sev2  = mx[6]  ;
data->sev3  = mx[7]  ;
data->sev4  = mx[8]  ;
data->sev5  = mx[9]  ;
data->sev6  = mx[10] ;
data->sev7  = mx[11] ;
data->sev8  = mx[12] ;
data->sev9  = mx[13] ;
data->sev10 = mx[14] ;

for(i=0;i<8;i++)
   data->stat[i]=stat[i];

for(i=0;i<40;i++)
   data->errc[i]=errc[i];

   }
}
//

int  stringcomp      (char*  s1,  char*  s2,  int
      lunghezza)
{// =1 le 2 stringhe sono uguali
   int i, var, mult;
   var= 1;
   for (i=0;i<lunghezza;i++)
   {
      if (s1[i]==s2[i])
         mult=1;
      else
         mult=0;
      var=var*mult;
   }
   return var;
}

int str2num (char* ss)
{
   int c;
   c=0;
```

```
   while (ss[c] != '.')
   {
      c++;
   }

         int dec = 0;
   int i;
         for(i=0; i<c; i++){
          dec = dec * 10 + ( ss[i] - '0' );
         }

   return dec;

}

int *initvece  (char *input, int *level)
{

   char *cp = strtok(input, ",");
   if (cp == NULL) {
      // Non ci sono altri separatori
      return (int *) malloc(sizeof(int) *
         *level);
   }

   int my_index = -1;
   int n;
   if (sscanf(cp, "%d", &n) == 1) {
      my_index = *level;
      *level += 1;
   } else {
      //TrasmettiUSB("Intero non valido");
   }
   int *array = initvece(NULL, level);
   if (my_index >= 0) {
      array[my_index] = n;
   }
   return array;
}

char   idM              (int indirizzo)
{
   switch (indirizzo)
   {
      case 0:
         return '0';
         break;
      case 1:
         return '1';
         break;
      case 2:
         return '2';
         break;
      case 3:
         return '3';
         break;
      case 4:
         return '4';
         break;
      case 5:
         return '5';
         break;
      case 6:
         return '6';
         break;
      case 7:
         return '7';
         break;
      case 8:
         return '8';
         break;
      case 9:
         return '9';
         break;
      case 10:
         return '10';
         break;
      case 11:
         return '11';
         break;
      case 12:
         return '12';
         break;
      case 13:
         return '13';
         break;
      case 14:
         return '14';
         break;
      case 15:
         return '15';
         break;
      case 16:
         return '16';
         break;
      case 17:
         return '17';
         break;
      case 18:
         return '18';
         break;
      case 19:
         return '19';
         break;
```

```
      case 20:
         return '20';
         break;
      default:
         return '0';
   }


}

int   c2i            (char c)
{
   switch (c)
   {
      case '0':
         return 0;
         break;
      case '1':
         return 1;
         break;
      case '2':
         return 2;
         break;
      case '3':
         return 3;
         break;
      case '4':
         return 4;
         break;
      case '5':
         return 5;
         break;
      case '6':
         return 6;
         break;
      case '7':
         return 7;
         break;
      case '8':
         return 8;
         break;
      case '9':
         return 9;
         break;
      default:
         return 0;
   }


}

void      writeMeasureinFile         (double
          ValvecI[3] , double vb, double
          assorbanza)
{
   LEDROSSO   = 0 ;
   LEDBLU     = 0 ;
   LEDGIALLO  = 1 ;
   while ( !FSInit( ) ) ;
      //TrasmettiUSB ( "\nLa scheda
          MicroSD e' stata inizializzazta
          correttamente per il salvataggio della
          misura" ) ;
   LEDGIALLO = 0 ;
   TEMPO DataEOranow ;
  // data e ora coordinate latitudine
          longitudine profondità
   LeggiCalendario ( &DataEOranow
          ) ;
   char fileMeasure[100];
   memset(fileMeasure, 0x00,
          sizeof(fileMeasure));

   TrasmettiUSB ( "\nSCRITTURA
          MISURA IN SCHEDA\n");
   char mr[20];
   strcpy(mr,"TxMisure");
   int arrva = 0;
   while( arrva==0 )
   {

      TrasmettiRaspy(mr);
      if(strstr(VettoreRicezioneRaspy, "*R_")
          != NULL &&
          strstr(VettoreRicezioneRaspy, "_+")
          != NULL)
         arrva=1;
      //TrasmettiUSB
          (VettoreRicezioneRaspy);
   }
   ArrivatoComandoRaspy = 0;
   str2double(VettoreRicezioneRaspy);
   memset(VettoreRicezioneRaspy, 0x00,
          sizeof(VettoreRicezioneRaspy));
   //    latitudine = 43.15  ; // ricevuti da
          sottomarino
   //    longitudine = 40.91 ; // ricevuti da
          sottomarino
   //    profondita = 30.11  ; // ricevuti da
          sottomarino
   double maxmis,minmis,mmis;
   maxmis = maxim ( ValvecI ) ;
   minmis = minim ( ValvecI ) ;
   mmis = (maxmis + minmis)/2.0000;
```

```
sprintf ( fileMeasure, "%x_%x_%x-
      %x_%x_%x+Missione_%d+%.0f_
      %.0f_%.0f.txt ", DataEOranow.Data,
      DataEOranow.Mese,
      DataEOranow.Anno,
      DataEOranow.Ore,
      DataEOranow.Minuti,
      DataEOranow.Secondi , missione ,
      latitudine , longitudine , profondita )
      ;

pointerMeasure = FSfopen ( fileMeasure,
      "w" ) ;
if ( pointerMeasure == NULL )
{
   LEDROSSO = 1 ;

}
else
{
   char posm[400];
   char Indiceffm;
   int i,jj;
   // inizio allocazione e scrittura
   for (i=0;i<400;i++)
      posm[i]= 0x00;
   sprintf ( posm,
      "data:%x_%x_%x\nora:%x_%x_%x
      \nMissione_%d\nlat:%f\nlon:%f\npr
      f:%f\n", DataEOranow.Data,
      DataEOranow.Mese,
      DataEOranow.Anno,
      DataEOranow.Ore,
      DataEOranow.Minuti,
      DataEOranow.Secondi, missione ,
      latitudine , longitudine , profondita )
      ;
   Indiceffm = 0 ;
   while ( posm[Indiceffm] != 0x00 )
      Indiceffm++ ;
   if ( FSfwrite ( posm, 1, Indiceffm,
      pointerMeasure ) != Indiceffm )
      while ( 1 ) ;
   // fine allocazione e scrittura
   for (jj=0;jj<nmisure;jj++)
   {
      // inizio allocazione e scrittura
      for (i=0;i<250;i++)
         posm[i]= 0x00;
      sprintf ( posm, "\nmisura %d : %.2f
      uA ;\n",jj+1,ValvecI[jj] ) ;
      Indiceffm = 0 ;
      while ( posm[Indiceffm] != 0x00 )
         Indiceffm++ ;
      if ( FSfwrite ( posm, 1, Indiceffm,
      pointerMeasure ) != Indiceffm )
         while ( 1 ) ;
      // fine allocazione e scrittura
   }
   // inizio allocazione e scrittura
   for (i=0;i<400;i++)
      posm[i]= 0x00;
   sprintf ( posm, "\nmisura media : %.2f
      uA ;\n",mmis ) ;
   Indiceffm = 0 ;
   while ( posm[Indiceffm] != 0x00 )
      Indiceffm++ ;
   if ( FSfwrite ( posm, 1, Indiceffm,
      pointerMeasure ) != Indiceffm )
      while ( 1 ) ;
   // fine allocazione e scrittura

   // inizio allocazione e scrittura
   for (i=0;i<400;i++)
      posm[i]= 0x00;
   sprintf ( posm, "\nmisura media bianco :
      %.2f uA ;\n",vb ) ;
   Indiceffm = 0 ;
   while ( posm[Indiceffm] != 0x00 )
      Indiceffm++ ;
   if ( FSfwrite ( posm, 1, Indiceffm,
      pointerMeasure ) != Indiceffm )
      while ( 1 ) ;
   // fine allocazione e scrittura

   // inizio allocazione e scrittura
   for (i=0;i<400;i++)
      posm[i]= 0x00;
   sprintf ( posm, "\nmisura assorbanza
      media : %.2f uA ;\n",assorbanza ) ;
   Indiceffm = 0 ;
   while ( posm[Indiceffm] != 0x00 )
      Indiceffm++ ;
   if ( FSfwrite ( posm, 1, Indiceffm,
      pointerMeasure ) != Indiceffm )
      while ( 1 ) ;
   // fine allocazione e scrittura

   if (FSfclose(pointerMeasure))
      while (1);
   //TrasmettiUSB ( posm ) ;
   //TrasmettiUSB("\nFile chiuso");
   int a1,a2,a3;
```

```
a1=1;
a2=2;
a3=3;
char posm2[500];
for (i=0;i<500;i++)
   posm2[i]= 0x00;
sprintf ( posm2,
   "data:%x_%x_%x;\nora:%x_%x_%
   x;\nlat:%f;\nlon:%f;\nprf:%f;\n\nmis
   ura %d : %.2f uA ;\n\nmisura %d :
   %.2f uA ;\n\nmisura %d : %.2f uA
   ;\n\nmisura media : %.2f uA
   ;\n\nmisura media bianco : %.2f uA
   ;\n&&",
     DataEOranow.Data,
   DataEOranow.Mese,
   DataEOranow.Anno,
     DataEOranow.Ore,
   DataEOranow.Minuti,
   DataEOranow.Secondi ,
     latitudine , longitudine ,
   profondita,a1,ValvecI[0],a2,ValvecI
   [1],a3,ValvecI[2],mmis,vb ) ;

uint8_t trasmdatav[28];
int jk = 1;
i = 0;
unsigned char CRC16t[2];
trasmdatav[0]='<';
dtb(ValvecI[0]);
for (i=0;i<4;i++)
{
   trasmdatav[jk]=tmpv[i];
   jk++;
}
dtb(ValvecI[1]);
for (i=0;i<4;i++)
{
   trasmdatav[jk]=tmpv[i];
   jk++;
}
dtb(ValvecI[2]);
for (i=0;i<4;i++)
{
   trasmdatav[jk]=tmpv[i];
   jk++;
}
dtb(mmis);
for (i=0;i<4;i++)
{
   trasmdatav[jk]=tmpv[i];
   jk++;
}
dtb(vb);
for (i=0;i<4;i++)
{
   trasmdatav[jk]=tmpv[i];
   jk++;
}
dtb(assorbanza);
for (i=0;i<4;i++)
{
   trasmdatav[jk]=tmpv[i];
   jk++;
}

crc_t crc;

crc = crc_init();
crc = crc_update(crc, (unsigned char
   *)trasmdatav, 25);
crc = crc_finalize(crc);
unsigned int If;
If = crc ;
If = If >> 8 ;
CRC16t[0] = (char)If ;  //LSB
If = crc ;
If = If & 0x00FF ;
CRC16t[1] = (char)If ;  //MSB

trasmdatav[jk] = CRC16t[1];//MSB
   CRC  0x81;//
jk++;
trasmdatav[jk] = CRC16t[0];//LSB
   CRC  0xAD;//
jk++;
trasmdatav[jk]='>';
arrva=0;
while(arrva==0)
{
   TrasmettiRaspyData(trasmdatav);
   DELAY_MS(5000);
   if(strstr(VettoreRicezioneRaspy,
    "*FILEOK+") != NULL )
      arrva=1;
}
TrasmettiUSB("*FILEOK+ SI");
ArrivatoComandoRaspy = 0;
  }
  /*Fine salvataggio*/

}
double maxim (double
      ValvecI[3])
```

```
{
   int c,size,location;
   double maximum;
   size = 3;
   maximum = ValvecI[0];
   for (c = 1; c < size; c++)
   {
      if (ValvecI[c] > maximum)
      {
         maximum  = ValvecI[c];
         location = c+1;
      }
   }

   return maximum;

}
double  minim                (double ValvecI[3])
{
   int c,size,location;
   double minimum;
   size = 3;
   minimum = ValvecI[0];
   for (c = 1; c < size; c++)
   {
      if (ValvecI[c] < minimum)
      {
         minimum  = ValvecI[c];
         location = c+1;
      }
   }

   return minimum;

}

void str2double(char *sv)
{

   int c,i;
   c=0;
   i=0;
   int index=0;
   int indv[4];
   char strtot[100],newstr[100];
   while (sv[c] != '+')
   {
      strtot[c]=sv[c];
      if (sv[c]=='_')
      {
         indv[index]=c;
         index++;
      }
      c++;
   }
   int in,fin,cont,nstrc;
   double dati[3],temp;
   for (i=0;i<3;i++)
   {
      nstrc = 0;
      in = indv[i];
      fin = indv[i+1];
      for (cont = in+1 ; cont<fin;cont++ )
      {
         newstr[nstrc] = strtot[cont];
         nstrc++;
      }
      temp = atof(newstr);
      dati[i] = (double)(temp);
   }
   latitudine = dati[0] ;
   longitudine = dati[1] ;
   profondita = dati[2]  ;



}

void MisureMain(void)
{

      int p=0;
      int i=0;
      int flag=0;
      int
         ulsb3_1,ulsb3_2,ulsb3_3,ulsb3_4,uls
         b3_5;
      double absorbance;
//     while ( flag < 900 )
//     {

      initStatE ( &ev , indev);

      initStatM ( &sb1stat , sb1);
      initStatM ( &sb1cmd  , sb1);

      initStatM ( &sb2stat , sb2);
      initStatM ( &sb2cmd  , sb2);

      initStatM ( &sb3stat , sb3);
      initStatM ( &sb3cmd  , sb3);

      initStatM ( &sa1stat , sa1);
      initStatM ( &sa1cmd  , sa1);
```

```
// INIZIALIZZAZIONI PER ROUTINE TEST FW
setcmdE ("set", indev, &ev, "1c,2c,3c,4c,5c,6c,7c,8c,9a,10c", no, ritardo);
getstatusE( &ev , indev);
setcmdM ("int", sa1, &sa1cmd, ulsa1ad, 400, 'I', ritardo );
setcmdM ("int", sb3, &sb3cmd, vmisc, ulssb3, 'I', 1000 );
getstatusM( &sb3stat , sb3);
// FINE INIZIALIZZAZIONI

/* INIZIO ASPIRAZIONE E DEPRESSURIZZAZIONE */
TrasmettiUSB("\nINIZIO ASPIRAZIONE E DEPRESSURIZZAZIONE\n");

setcmdE ("set", indev, &ev, "9a,10c", no, ritardo);
getstatusE( &ev , indev);

getstatusM( &sa1stat , sa1);
memset(Vettore, '\0', sizeof(Vettore));
sprintf ( Vettore, "\nPressione attuale SA1 : %.2f bar ;\n" , sa1stat.psa ) ;
TrasmettiUSB(Vettore);

int t1 = (ulsa1ad/ulssa1)*1000 + ritardo ;
int t1breve = 500+ritardo; // 1/50*1000 + 101 = 301
//  float psa = 21.5 ;  // cancella riga



if ( sa1stat.psa >= 1.0 ) // psa diventa: sa1stat.psa ---> if ( sa1stat.psa >= 1.0 )
{
  setcmdM ("mov", sa1, &sa1cmd, ulsa1ad, ulssa1, 'S', t1 );
  getstatusM( &sa1stat , sa1);



  setcmdE ("set", indev, &ev, "9c,10n", no, ritardo);
  getstatusE( &ev , indev);
  while ( sa1stat.psa > 2.0 ) // psa diventa: sa1stat.psa ---> while ( sa1stat.psa > 2.0 )
  {
    setcmdM ("mov", sa1, &sa1cmd, 10, 10, 'S', t1breve );

    getstatusM( &sa1stat , sa1);



    memset(Vettore, '\0', sizeof(Vettore));
    sprintf ( Vettore, "\nPressione attuale SA1 : %.2f bar ;\n" , sa1stat.psa ) ;
    TrasmettiUSB(Vettore);

//    psa = psa - 5.0;  // cancella riga
  }
} else if ( sa1stat.psa < 1.0 ) // psa diventa: sa1stat.psa ---> else if ( sa1stat.psa < 1.0 )
{
  TrasmettiUSB("\nERRORE ASPIRAZIONE E DEPRESSURIZZAZIONE\n");

}
if ( sa1stat.psa < 2.0 ) // psa diventa: sa1stat.psa ---> if ( sa1stat.psa < 2.0 )
  TrasmettiUSB("\nFINE ASPIRAZIONE E DEPRESSURIZZAZIONE\n");
else
  TrasmettiUSB("\nERRORE ASPIRAZIONE E DEPRESSURIZZAZIONE\n");

/* FINE ASPIRAZIONE E DEPRESSURIZZAZIONE */

 p=1;
/* INIZIO RISCIACQUO */
TrasmettiUSB("\nINIZIO RISCIACQUO\n");
Ritardo (501);
getstatusM( &sa1stat , sa1);
int ulattsa1d=( (int)( sa1stat.ulatt ) ) - ulsa1r ;// togli
p=0;
setcmdE ("set", indev, &ev, "9n,10a",
```

```
    no, ritardo);
p=3;
getstatusE( &ev , indev);
int t2 = (ulattsa1d/ulssa1)*1000 +
    ritardo;
setcmdM ("mov", sa1, &sa1cmd,
    ulattsa1d, ulssa1risc, 'I', t2 );
getstatusM( &sa1stat , sa1);
setcmdE ("set", indev, &ev, "9n,10c",
    no, ritardo);
getstatusE( &ev , indev);
p=4;
setcmdE ("set", indev, &ev,
    "3a,4a,7a,8a", no, ritardo);
getstatusE( &ev , indev);
int iv , sv ;
iv = 0 ;
sv = 0 ;
int tmpvar = 1;
int t3 = (10000/ulssb3)*1000 + ritardo;
while(iv < (nCicliRisciacquo) && sv <
    (nCicliRisciacquo+1) )
{
   if ( tmpvar == 1 )
   {
      setcmdM ("mov", sb3, &sb3cmd,
    vmisc, ulssb3, 'S', t3 );
      getstatusM( &sb3stat , sb3);
      sv++;
      tmpvar = 2 ;
   }else if ( tmpvar == 2 )
   {
      setcmdM ("mov", sb3, &sb3cmd,
    vmisc, ulssb3, 'I', t3 );
      getstatusM( &sb3stat , sb3);
      iv++;
      tmpvar = 1 ;
   } else
   {
      Ritardo (t3);
      getstatusM( &sb3stat , sb3);
   }
}
setcmdM ("mov", sb3, &sb3cmd,
    vmisc, ulssb3, 'S', t3 );
getstatusM( &sb3stat , sb3);


TrasmettiUSB("\nFINE
    RISCIACQUO\n");
/* FINE RISCIACQUO */
```

```
p=3;
/* INIZIO SCARICO */
TrasmettiUSB("\nINIZIO
    SCARICO\n");

setcmdE ("set", indev, &ev,
    "3c,4a,7a,8a", no, ritardo);
getstatusE( &ev , indev);

setcmdM ("int", sb3, &sb3cmd, ulsb3,
    ulssb3, 'I', t3 );
getstatusM( &sb3stat , sb3);

iv = 0 ;
sv = 0 ;
while(iv < (nCicliScarico) && sv <
    (nCicliScarico+1))
{
   if ( sb3stat.finecI == 1 && ev.sev3 ==
    'S' )
   {
      setcmdE ("set", indev, &ev,
    "3a,4n", no, ritardo);
      getstatusE( &ev , indev);
      setcmdM ("mov", sb3, &sb3cmd,
    vmisc, ulssb3, 'S', t3 );
      getstatusM( &sb3stat , sb3);
      sv++;
   } else if ( sb3stat.ulatt >= 3999.00000
    && ev.sev3 == 'A')
   {
      setcmdE ("set", indev, &ev,
    "3c,4n", no, ritardo);
      getstatusE( &ev , indev);
      setcmdM ("int", sb3, &sb3cmd,
    ulsb3, ulssb3, 'I', t3 );
      getstatusM( &sb3stat , sb3);
      iv++;
   } else
   {
      Ritardo (t3);
      getstatusM( &sb3stat , sb3);
   }
}
setcmdE ("set", indev, &ev, "3n,4c",
    no, ritardo);//"3n,4c,7c,8c"
getstatusE( &ev , indev);

TrasmettiUSB("\nFINE SCARICO\n");
/* FINE SCARICO */
    /* INIZIO BIANCO */
```

```
p=1;
setcmdE ("set", indev, &ev, "9c,10a",
   no, ritardo);
getstatusE( &ev , indev);
setcmdM ("mov", sa1, &sa1cmd,
   ulsa1bianco, ulssa1, 'I', t1 );
getstatusM( &sa1stat , sa1);
setcmdE ("set", indev, &ev,
   "3a,4a,7a,8a,10c", no, ritardo);
getstatusE( &ev , indev);


ulsb3_1=4000;
ulsb3_2=3000;
ulsb3_3=3000;
ulsb3_4=2000;
ulsb3_5=1000;
setcmdM ("mov", sb3, &sb3cmd,
   ulsb3_1, ulssb3, 'S', t3 );
getstatusM( &sb3stat , sb3);
setcmdM ("mov", sb3, &sb3cmd,
   ulsb3_2, ulssb3, 'I', t3 );
getstatusM( &sb3stat , sb3);
setcmdM ("mov", sb3, &sb3cmd,
   ulsb3_3, ulssb3, 'S', t3 );
getstatusM( &sb3stat , sb3);
setcmdM ("mov", sb3, &sb3cmd,
   ulsb3_4, ulssb3, 'I', t3 );
getstatusM( &sb3stat , sb3);

/* INIZIO MISURA BIANCO */
TrasmettiUSB("\nINIZIO MISURA
   BIANCO\n");

double d1,d2,d3,d4,d1o,d2o,d3o,d4o;
d1 = 85.00;// laser verde Cr: 85.00
d2 = 75.00;
d3 = 75.00;// laser rosso Zn :40.00
d4 = 75.00;
d1o = 0.00;
d2o = 0.00;
d3o = 0.00;
d4o = 0.00;


// unsigned int
   Valoref[4][nmisure],MisureMedia[4
   ];// 4 canali per 4 metalli
memset(Vettore, '\0', sizeof(Vettore));
double
   ValoreI[nmisure],valorebiancom,
   MisureMedia;
double valc ,mult, uno, risultatoI;
uno = (double)1.00000;
mult = (double)(2048.0000);
double bits = pow(2,15) ;
int flagout,flagcount;
flagout = 1;
flagcount = 0;
double maxmis,minmis,dmis,mmis;
while (flagout == 1 && flagcount <
   numero_ripetizioni )
{
  for ( i=0; i<nmisure ; i++ )
  {
    DELAY_MS(tmisuraoff);
    CorrentiLaser ( d1 , d2 , d3 , d4 ) ;
    DELAY_MS(tlaser);
    valc =
  ((double)ADS8341_Mediato( 0
  ))/bits ;
    if ( valc <= 1.0000 && valc >
  0.0000 )
    {
      ValoreI[i] = mult*( uno - valc ) ;
  // Cr VI uA
      risultatoI = mult*( uno - valc ) ;
  // Cr VI uA
    }
    else if ( valc > 1.0000 )
    {
      ValoreI[i] = 0.0000 ; // Cr VI uA
      TrasmettiUSB("\nval > 1 \n");
    }
    else if ( valc < 0.0000 )
    {
      TrasmettiUSB("\nErrore
  misura\n");
      ValoreI[i] = 0.0000 ; // Cr VI uA
    }

    sprintf ( Vettore, "\nBianco
  misurato : %.2f uA ;\n" , risultatoI )
  ;
    TrasmettiUSB(Vettore);
    CorrentiLaser ( d1o , d2o , d3o ,
  d4o ) ;

}
maxmis = maxim ( ValoreI ) ;
minmis = minim ( ValoreI ) ;
dmis = maxmis - minmis;
mmis = (maxmis + minmis)/2.0000;
if ( dmis < discrepanza_misure )
```

```
    {
      flagout = 0;
      flagcount++;

    }else if ( dmis >= discrepanza_misure
    )
    {
      flagout = 1;
      flagcount++;

    }

}

sprintf ( Vettore, "\nBianco misurato
   medio :  %.2f  uA  ;\n" , mmis ) ;
TrasmettiUSB(Vettore);
valorebiancom = mmis ;

TrasmettiUSB("\nFINE MISURA
   BIANCO\n");
/*  FINE MISURA BIANCO */
setcmdM ("int", sb3, &sb3cmd,
   ulsb3_5, ulssb3, 'I', t3  );
getstatusM( &sb3stat , sb3);


/*  FINE BIANCO  */

p=4;
/*  INIZIO DOSAGGIO E
   MISCELAZIONE  */
TrasmettiUSB("\nINIZIO DOSAGGIO
   E MISCELAZIONE\n");

setcmdE ("set", indev, &ev,
   "1a,2a,10c", no, ritardo);
getstatusE( &ev , indev);

int t4,t5,t6;
t4 = (ulsb1/ulssb1)*1000  + ritardo;
t5 = (ulsb2/ulssb2)*1000  + ritardo;
t6 = (ulsa1d/ulssa1)*1000 + ritardo;

setcmdM ("mov", sb1, &sb1cmd,
   ulsb1, ulssa1, 'I', t4  );
getstatusM( &sb1stat , sb1);

setcmdM ("mov", sb2, &sb2cmd,
   ulsb2, ulssb2, 'I', t5  );
getstatusM( &sb2stat , sb2);

//      setcmdM ("mov", sa1, &sa1cmd,
      ulsa1d, ulssa1, 'I', t6  );
//    getstatusM( &sa1stat , sa1);

setcmdE ("set", indev, &ev,
   "1c,2c,10n", no, ritardo);
getstatusE( &ev , indev);

setcmdE ("set", indev, &ev,
   "3a,4a,7a,8a", no, ritardo);
getstatusE( &ev , indev);

iv = 0 ;
sv = 0 ;
tmpvar = 1;
while(iv < (nCicliRisciacquo) && sv <
   (nCicliRisciacquo+1) )
{
  if ( tmpvar == 1 )
  {
    setcmdM  ("mov", sb3, &sb3cmd,
   vmisc, ulssb3, 'S', t3  );
    getstatusM( &sb3stat , sb3);
    sv++;
    tmpvar = 2 ;
  }else if (  tmpvar == 2 )
  {
    setcmdM  ("mov", sb3, &sb3cmd,
   vmisc, ulssb3, 'I', t3  );
    getstatusM( &sb3stat , sb3);
    iv++;
    tmpvar = 1 ;
  } else
  {
    Ritardo (t3);
    getstatusM( &sb3stat , sb3);
  }
}

int t7 = (ulsb3d/ulssb3)*1000 + ritardo;

setcmdM ("mov", sb3, &sb3cmd,
   ulsb3_1, ulssb3, 'S', t3  );
getstatusM( &sb3stat , sb3);
setcmdM ("mov", sb3, &sb3cmd,
   ulsb3_2, ulssb3, 'I', t3  );
getstatusM( &sb3stat , sb3);
setcmdM ("mov", sb3, &sb3cmd,
   ulsb3_3, ulssb3, 'S', t3  );
getstatusM( &sb3stat , sb3);
setcmdM ("mov", sb3, &sb3cmd,
   ulsb3_4, ulssb3, 'I', t3  );
```

```
      getstatusM( &sb3stat , sb3);

      setcmdE    ("set", indev, &ev, "4a,8n",
         no, ritardo);
      getstatusE( &ev , indev);

//         setcmdM    ("mov", sb3, &sb3cmd,
         ulsovrappressione, ulssb3, 'I', ritardo
         );
//       getstatusM( &sb3stat , sb3);

      DELAY_MS(tstazionario);



      TrasmettiUSB("\nFINE DOSAGGIO E
         MISCELAZIONE\n");
      /*            FINE      DOSAGGIO      E
         MISCELAZIONE  */

         p=5;
      /*  INIZIO MISURA  */
      TrasmettiUSB("\nINIZIO
         MISURA\n");



      d1 = 85.00;// laser verde Cr: 85.00
      d2 = 75.00;
      d3 = 75.00;// laser rosso Zn :40.00
      d4 = 75.00;
      d1o = 0.00;
      d2o = 0.00;
      d3o = 0.00;
      d4o = 0.00;


//                                unsigned     int
         Valoref[4][nmisure],MisureMedia[4
         ];// 4 canali per 4 metalli
      memset(Vettore, '\0', sizeof(Vettore));
//    double ValoreI[nmisure], MisureMedia;
//      double valc ,mult, uno, risultatoI;
      uno = (double)1.00000;
      mult = (double)(2048.0000);
//      double bits = pow(2,15) ;
//      int flagout,flagcount;
      flagout = 1;
      flagcount = 0;
//      double maxmis,minmis,dmis,mmis;
      while (flagout == 1 && flagcount <
         numero_ripetizioni )
      {
for ( i=0; i<nmisure ; i++ )
{
   DELAY_MS(tmisuraoff);
   CorrentiLaser ( d1 , d2 , d3 , d4 ) ;
   DELAY_MS(tlaser);
   valc                                =
((double)ADS8341_Mediato(          0
))/bits ;
   if ( valc <= 1.0000 && valc >
0.0000  )
   {
      ValoreI[i] = mult*( uno - valc ) ;
// Cr VI  uA
      risultatoI = mult*( uno - valc ) ;
// Cr VI  uA
   }
   else if ( valc > 1.0000 )
   {
      ValoreI[i] = 0.0000 ; // Cr VI uA
      TrasmettiUSB("\nval > 1 \n");
   }
   else if ( valc < 0.0000 )
   {
      TrasmettiUSB("\nErrore
misura\n");
      ValoreI[i] = 0.0000 ; // Cr VI uA
   }

   sprintf ( Vettore, "\nCorrente di
CrVI misurata :  %.2f  uA  ;\n" ,
risultatoI ) ;
   TrasmettiUSB(Vettore);
   CorrentiLaser ( d1o , d2o , d3o ,
d4o ) ;

}
maxmis = maxim ( ValoreI ) ;
minmis = minim ( ValoreI ) ;
dmis = maxmis - minmis;
mmis = (maxmis + minmis)/2.0000;
absorbance                          =
log10(valorebiancom/mmis);
if ( dmis < discrepanza_misure   &&
absorbance >= 0.0000)
{
   flagout = 0;
   flagcount++;

}else if ( dmis >= discrepanza_misure
|| absorbance < 0.0000 )
{
   flagout = 1;
```

```
            flagcount++;

          }

        }
        sprintf ( Vettore, "\nCorrente di CrVI
          misurata media : %.2f uA ;\n" ,
          mmis ) ;
        TrasmettiUSB(Vettore);
        sprintf ( Vettore, "\nAssorbanza di CrVI
          calcolata : %.2f ;\n" , absorbance ) ;
        TrasmettiUSB(Vettore);
        writeMeasureinFile ( ValoreI,
          valorebiancom, absorbance ) ;


        setcmdM ("sut", sb3, &sb3cmd, ulsb3,
          ulssb3, 'S', t3 );
        getstatusM( &sb3stat , sb3);



        TrasmettiUSB("\nFINE MISURA\n");
        /* FINE MISURA */



        p=6;

        /* INIZIO SCARICO POST MISURA
          */
        TrasmettiUSB("\nINIZIO SCARICO
          POST MISURA\n");

        setcmdE ("set", indev, &ev,
          "3c,4a,7a,8a", no, ritardo);
        getstatusE( &ev , indev);

        setcmdM ("int", sb3, &sb3cmd, ulsb3,
          ulssb3, 'I', t3 );
        getstatusM( &sb3stat , sb3);

        iv = 0 ;
        sv = 0 ;
        while(iv < (nCicliScarico) && sv <
          (nCicliScarico+1))
        {
//          if ( sb3stat.finecI == 1 && ev.sev3
          == 'S')
          if ( ev.sev3 == 'S')
          {
            setcmdE ("set", indev, &ev,
              "3a,4n", no, ritardo);
            getstatusE( &ev , indev);
            setcmdM ("mov", sb3, &sb3cmd,
            vmisc, ulssb3, 'S', t3 );
            getstatusM( &sb3stat , sb3);
            sv++;
          }// else if ( sb3stat.ulatt >=
          3999.00000 && ev.sev3 == 'A')
          else if ( ev.sev3 == 'A')
          {
            setcmdE ("set", indev, &ev,
            "3c,4n", no, ritardo);
            getstatusE( &ev , indev);
            setcmdM ("int", sb3, &sb3cmd,
            ulsb3, ulssb3, 'I', t3 );
            getstatusM( &sb3stat , sb3);
            iv++;
          } else
          {
            Ritardo (t3);
            getstatusM( &sb3stat , sb3);
          }
        }

        setcmdE ("set", indev, &ev, "4c,7c,8c",
          no, ritardo);
        getstatusE( &ev , indev);

        TrasmettiUSB("\nFINE SCARICO
          POST MISURA\n");
        /* FINE SCARICO POST MISURA */
        p=7;



        StatoDebug = 0 ;

}

void dtb(float val){


size_t siz,siza;
siza = sizeof(double);
siz=4;
memcpy (&tmpv, &val, siz);


}
```