Dipartimento di Ingegneria Meccanica e Aerospaziale

Corso di Laurea magistrale in

INGEGNERIA AEROSPAZIALE

Tesi di laurea magistrale

Experimental analysis on UAV systems monitoring through optical fibre-based sensors

Relatore:
Prof. Matteo Davide Lorenzo Dalla Vedova

Candidato:
Giuseppe LABIANCA

Correlatori:
Ing. Alessandro Aimasso
Ing. Matteo Bertone

Anno accademico 2023-2024

Contents

List of Figures

List of Tables

Abstract

In the last years, multiple engineering sectors have shown an increased use of optical fibre-based sensors, thanks to their inherent advantages that have encouraged experimental studies about them. Regarding aerospace applications, they are really useful for their immunity to electromagnetic disturbances, low weight and low cables' size, high sensitivity, chemical inertia and electrical passivity.
The main purpose of this thesis is to verify the optical fibre sensors capability to measure physical parameters that define aerospace systems, offering a greater quantity of data. The overall information detected could be useful to develop an innovative technique for systems monitoring, like making data sensors fusion and enhancing or substituting electronic device measures. Moreover, optical information could be used for control logic development too. This is possible because the optical fibre sensors, due to their physical advantages listed before, can be placed even in hostile environments where the application of the common sensors is complex. Thus, the activity described in this thesis want to demonstrate the effectiveness of this technology as a valuable support to the already existing solutions for aerospace systems diagnostic and prognostic.
The experimentation was conducted on an Unmanned Aerial Vehicle (UAV) model, employing the optical fibre sensors *Fibre Bragg Gratings* (FBG), with which it's possible to measure the mechanical strain and temperature. So, through a series of laboratory tests and a flight test, the potential reasonable benefits of these optical sensors have been highlighted, for monitoring the activity of safety-critical systems.

Chapter 1

General overview and main purpose

The main purpose of this experimental work conducted in these months was to verify and certify that optical fibres can be used for monitoring systems and subsystems of a drone, related to diagnostic and prognostic strategies inside the development of the *virtual sensing* technology (also called soft sensing) along the whole life cycle. A virtual sensing system uses information collected from various sensors and process parameters to calculate an estimate of the physical value required, offering a feasible and economical alternative to cheap or impractical physical measurement instruments employed nowadays.
Indeed, the main advantages of a sensor crafted with an optical fibre in front of common electrical sensors can be summarised with the following:

- Lower weight
- Lower costs of production
- Lower size
- Insensitivity to electromagnetic interference

These are key physical properties of the aerospace field for which the decrease in costs and weights is fundamental both for the business and research, as well as a minimum size that allows an enhanced distribution of the subsystems, but also the third characteristic is essential since the aerospace environment is often harsh and overflowing with interference; so, checked that optical fibres represent a very cheap and light solution, it was necessary to demonstrate that they can perform a measure of different physical parameters with high accuracy, sensitivity and reliability, that are three main values for optimum monitoring of a system.

1.1 Work-flow and test campaign planned

First of all, the drone utilized for the experimental tests was a remote-controlled UAV (Unmanned aerial vehicle) provided by the ICARUS PoliTO student team, named *Anubi*, chosen because of its simplicity and similarity with a common aircraft; it has electrical propulsion, with 6 m of wingspan and 20 kg of MTOW. The model is equipped with a modular structure in reinforced polymer with carbon fibre; this allows the integration of FBG sensors during the rolling sections of the structure.

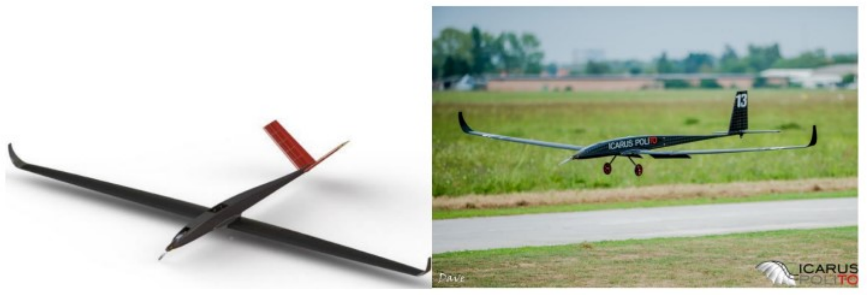


Figure 1.1: UAV Anubi [12]

It can be schematized into three main parts:

- Structural
- Flight controls
- Propulsive
- Landing gear

So, once the main systems were defined, it was necessary to assume which physical parameters were worth to be evaluated with the new sensors.

Main physical parameters To perform satisfying monitoring of the systems, it's useful to take into account two fundamental physical parameters, *temperature* and *mechanical strain* which can be detected using a fibre optical sensor; this is done to elaborate a virtual sensing logic capable of measuring the wing and landing gear's deformation subjected to static and dynamic loads (weight-load, wind shears, manoeuvres, landing phase) and the temperature of devices which warm-up so potentially can cause dangerous situations due to blazes propagation.

Optical fibre type To test the response to temperature and mechanical strain, *FBG (Fibre Bragg Grating)* sensors were employed for their properties to react to these physical variations, modifying its optical characteristics, e.g. varying the refraction index through thermal expansion and structural stress.
So, the steps of this activity described before make up the base of the following test campaign and can be summarised here:

1) Identification of the physical parameters: the temperature T and mechanic strain ϵ

2) State of art preliminary analyses

3) How to use the fibre:

   - Temperature: sensors developing for the environment (probes) and devices (stuck fibres)
   - Strain: from literature

4) Propulsive application: thermal monitoring of battery and ESC for fault diagnosis

5) Structural application:

   - Wing: FEM validation of mechanic strain ($\Delta\epsilon$)
   - Tail: static and flight tests
   - Landing gear: accurate monitoring of a critical element subjected to a stress pulse load and flight information ON/OFF

For this experimental process, some theoretical notes about optical fibres from literature and previous analysis done here in the Politecnico di Torino were taken into account to define a 'state of art' of this technology and its applications, to establish what was improvable from them through some tests campaign; indeed, these works have already defined a reliable and accurate methodology to employ these sensors, regarding samples for laboratory thermal and mechanic tests, a model aircraft for flight and static tests and a model for data transmission.

The test campaign executed has followed a meticulous schedule based on the above-mentioned applications planned which mainly regarded the comparison between the data obtained with the traditional sensors and FBGs developed during these tests, in particular:

1) Thermal calibration of FBG sensors crafted for thermal applications

2) FBG thermal response analysis on the UAV's propulsion system;

3) Static tests on the drone in the laboratory to define the appropriate FBGs acquisition frequency;

4) A flight test performed with resulting analysis and conclusions about the structural and environmental thermal monitoring;

Thus, the whole work methodology can be resumed here:

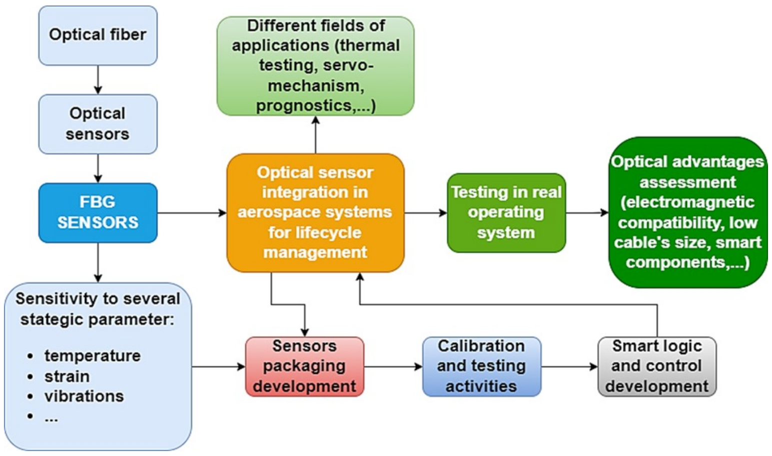

Optical fiber
Optical sensors
FBG SENSORS
Sensitivity to several stategic parameter:
• temperature
• strain
• vibrations
• ...
Different fields of applications (thermal testing, servo-mechanism, prognostics,...)
Optical sensor integration in aerospace systems for lifecycle management
Testing in real operating system
Optical advantages assessment (electromagnetic compatibility, low cable's size, smart components,...)
Sensors packaging development
Calibration and testing activities
Smart logic and control development

Figure 1.2: Overall approach to the experimental work [3]

Target summary and expected results This work-flow just represents the fundamentals of what was expected from this experimental activity, which is taking advantage of the optical fibres' remarkable qualities, in particular, the reaction to thermal and mechanical stresses and other physical properties superior to the traditional electrical sensors that make them suitable for aerospace application to obtain a new generation of sensors capable of performing an accurate, reliable and sensible measuring of key physical parameters of the airplane systems and environment that surrounds it during the flight phases. An almost impeccable correlation is expected between common sensors and these new ones, to demonstrate that FBGs are conform to be employed as a substitution or in redundancy, reducing costs, weights and measuring errors.

Chapter 2

Optical fibres overview

2.1 Description of the optical fibres

Optical fibre is a very wide topic covering its use in the world of sensing as distinct from the one in telecommunications which most people are now familiar with. Fibre optic sensing has always been a topic closely related to developments in the telecommunication industry, and for a large part has fed off the many exciting developments stemming from the commercial development of fibre technology for the telecommunication industry, following rising demand from the population for quicker and more efficient communication. This growth interest is largely due to the wide variety of ways in which light can be used to measure real physical parameters such as strain, temperature, rotation, pressure, acceleration, and chemicals. In the case of light, the parameters that can be manipulated include the phase, wavelength, intensity or polarization, or a combination of these, which results in a huge diversity of possible approaches for the design of fibre optic sensors.
These systems are considered for aerospace applications due to many advantages that can lead to solutions with the potential to outperform their conventional counterparts. These advantages are:

- Use of a lightweight and flexible harness that implies a decrease of mass;
- Potential to embed in composite structures which are increasingly being used on the airplanes;
- High signal-to-noise ratio for high measurement accuracy, indeed, optical fibres have low losses compared with traditional electronics;
- Efficient multiplexing for high sensor capacity, low power requirements per sensor;
- Insensitivity to electromagnetic interference thanks to coating and to its structure (instead of electronic instruments by which are afflicted) and use of a passive sensor free from sparking or electrostatic discharge.
- Low-cost solution, since fabrication and maintenance costs are meanly lower than common electronic devices

- Inert to chemical reactants due to the materials that make it unresponsive to aggressive chemical reactants;
- Resistance to harsh and high-temperature environments which are common in aerospace missions; due to glass-made fibres and coating material that put up with very high temperatures;
- Zero risk of sparking caused by a short circuit and consequently of blaze propagation.

Nevertheless, it's important to mention some critical aspects too:

- Requirement of conversion of an optical signal to an electronic one to make the on-board devices able to manipulate these data;
- Low resistance due to the thickness and the materials used that make the fibre very brittle and fragile, so that the bending moments become severe but this can be fixed enhancing the coating's properties;
- Application of complex instruments such as the *Interrogator* which generates a light beam that goes through the fibre, adding weight and costs;
- Installation obstacles on structural components or other types of devices due to their low resistance.

Just to conclude this description, it's noteworthy to talk about two different phenomena: the *attenuation* and the *dispersion.*

- The attenuation of the signal is the progressive decrease of the power along the length of the optical fibre. The output power is lower than that in input: it depends on the injected power, inherent properties of the fibre, contaminants inside, Rayleigh scattering, the attenuation per unit of length and the length of the fibre. Usually, the optical fibre presents an attenuation of about 0.2-0.22 dB/km at 1500 nm of wavelength.
- The dispersion can be described as a different journey that various component of the signal makes and the consequence is a deformation of the original signal and a lower speed of transmission. There are mainly two categories of these losses: Intermodal and Intra-modal ones. The *inter modal* losses concern multi-mode optical fibre. This happens because in this case the rays propagate with different modes and each one gets a reflection with a different angulation. This means that each mode has a proper optical path to reach the end of the fibre. So, the time of propagation will be different and, consequently, the final signal will be deformed; for this reason, multi-mode optical fibre isn't employed for long distances. *Intra-model* losses, instead, concern the different speeds of the various spectral components of the signal, typical for single-mode optical fibre, caused by two components. The first one is the *chromatic dispersion*: it is produced because of different wavelengths of the spectral that have different speeds, mainly due to two factors: first of all, the material dispersion, so the variation of the material refraction index related to the wavelength. The second

contribution to intra-model losses is the polarization mode dispersions. It means that in this case, wavelengths that normally have the same velocity, instead present different speeds, due to some imperfections of the material which compose the core. Obviously, this phenomenon originates a deformation of the light signal and then a loss is generated.

2.1.1 Fibre's structure

Optical fibre is a cylindrical glass material and it's composed of multiple concentric layers (2.1): the core, cladding, and coating.

- The *core* is realised with glass or polymeric materials and it's the inner layer of the fibre, in which the light signal is transmitted, with a thickness usually of 50 μm.
- The *cladding* is realised to ensure the proper operability of the entire fibre, with a thickness of 125 μm.
- The *coating* is the external layer with the task of protecting the fibre from extreme achings that can become destructive due to low fibre's bending resistance.

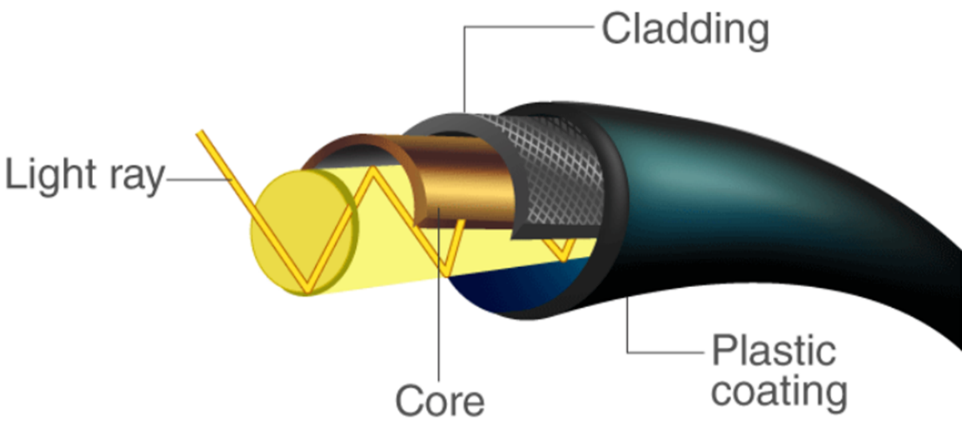


Figure 2.1: Main structure of the fibre [1]

2.1.2 Physical principle

It has been mentioned that a light beam is sent through the fibre in accordance with the *Snall's law* displayed in the picture below:

$$n_1 sen(\theta_1) = n_2 sen(\theta_2) \tag{2.1}$$

with:

- n_1 as the refraction index of material 1;
- n_2 as the refraction index of material 2;
- θ_1 as the incidence angle;

- θ_2 as the refraction angle.

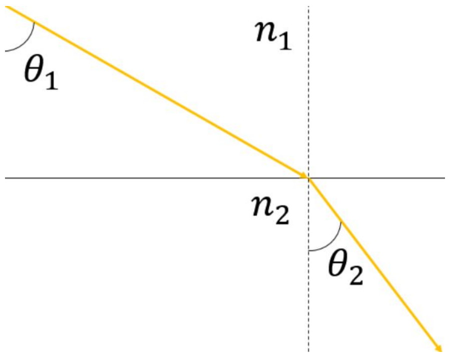


Figure 2.2: Snell's law representation [13]

This law describes how the refraction of a light beam works when it passes through two materials with different refraction indexes, where the refraction index of a material is a dimensionless physical quantity that refers to the ratio of the light speed into the vacuum and that in the considered material. For this reason, the light beam will change direction from θ_1 to θ_2 and for certain values of these indexes there may be a *total reflection.*
Now, if we consider the optical fibre as described before and we adopt Snell's law, n_1 and θ_1 are, respectively, the refraction index and the incidence angle of the light beam in the core at the interface with the cladding, in which we have n_1 and the refraction angle θ_2. However, the difference between the refractive indices enables the core–cladding interface to effectively act as a mirror such that a series of internal reflections transmits the light from one end of the fibre to the other.
There are some principles of light transmission through the optical fibre that are significant for fibre-optic chemical sensor function and design:

- *Critical angle*: if light strikes the cladding at an angle greater than the critical angle θ_c, the light is internally reflected at the core–cladding interface (figure 2.3). If light strikes the cladding at an angle less than the critical angle, as shown in figure 2.3, it is both partly reflected and refracted. The critical angle is defined by the ratio between the cladding and the core refractive indices:

$$\theta_1 = arcsen\left(\frac{n_2}{n_1}\right) \tag{2.2}$$

  and if θ_2 is higher than the critical value $\pi/2$, inside the core there will be the total reflection phenomenon.

- *Acceptance Cone.* In order to get high light transmission, it should propagate through the fibre by a series of total internal reflections. This transmission is achieved if the angles of the light entering the fibre are within the acceptance cone as shown in figure 2.3. The acceptance cone size depends on the refractive indices of the core and the cladding and also on the refractive index of air n_0.

- *Numerical Aperture.* The acceptance cone's width determines the efficiency of light collection of the fibre and can also be described in terms of the numerical aperture

$NA = \sqrt{n_1^2 - n_2^2}$, where a high NA indicates a wide acceptance cone and better light-gathering capabilities of the fibre. and a typical NA value for high-quality glass fibre is 0.55.

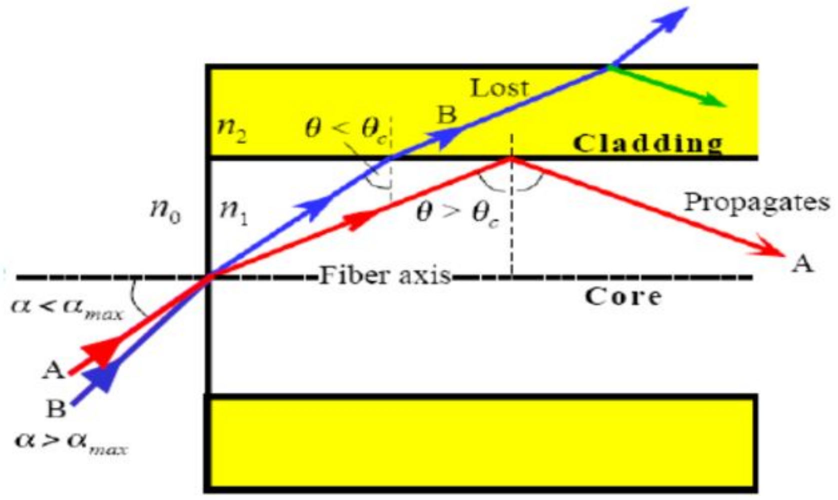


Figure 2.3: Scheme of the optical fibres' working [5]

Furthermore, it's possible to define maximum angle of the *acceptance cone* as:

$$\alpha_{max} = arcsin\left(\frac{NA}{n_0}\right) \tag{2.3}$$

2.1.3 Optical fibre types and configurations

Optical fibres are usually made of glass in a process in which a glass preform is warmed up and then the fibre is taken out from the melted hot glass. In most fibre-optic chemical sensors, glass fibres are used to transmit light in the visible and near-infrared regions of the optical spectrum ($400 < \lambda < 700$ nm). Otherwise, when the light employed is in the UV region, quartz (pure silica) fibres are used as the core material and doped silica (with a lower refractive index) is used as the cladding material. Optical fibres made of silver halide or chalcogenides are used to transmit light in the infrared region of the spectrum ($\lambda > 700$ nm). Plastic fibres are also used for fibre-optic chemical sensors; these fibres are very flexible and cheap but their optical characteristics are inferior to those of glass fibres and their heat tolerance is lower.
Optical fibres are also manufactured in many different configurations and sizes. For some fibre-optic chemical sensor applications, single optical fibres have diameters ranging from 50 to 500 μm while for other fibre-optic chemical sensor applications, fibre-optic bundles comprising thousands of identical single fibres (each with a diameter of a few micrometres) are employed to improve light transmission. In coherent fibre bundles, the position of each fibre on one end is identical to its position on the other end this peculiar configuration allows each individual fibre to transmit light in an addressable manner from one end of the bundle to the other; these bundles are used for imaging and sensing simultaneously. This difference is shown below in the figure:

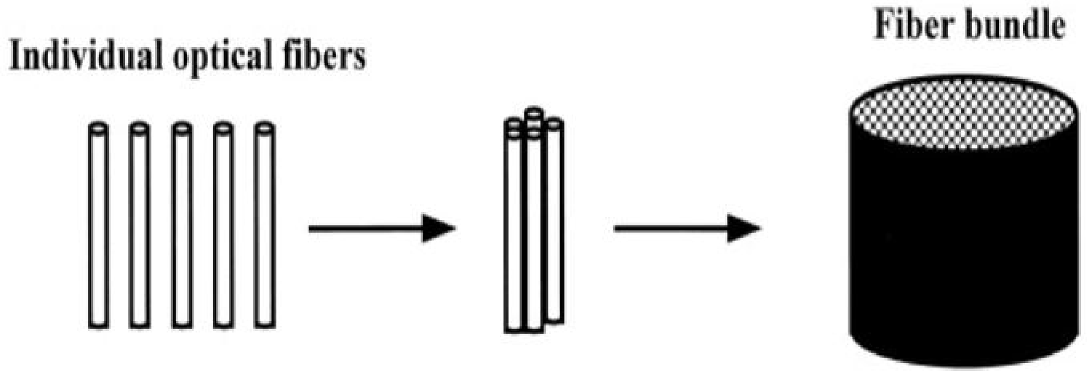


Figure 2.4: Individual and bundled fibers [14]

Another distinction is related to core's diameter, indeed there are two technologies:

- *Single-mode*: core's diameter is almost 10 μm, while the cladding arrives at 125 μm. With a very long period of use (about 20 years) and low losses, this type is employed for communication technologies but also in our case of study and allows the propagation of the mode *m=0* in the axial direction.

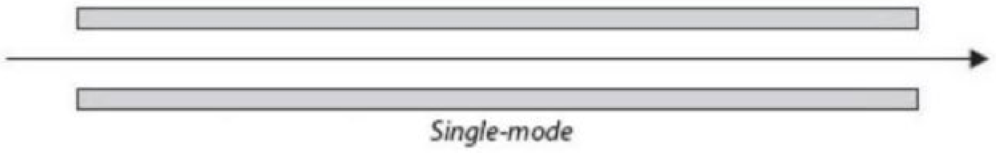


Figure 2.5: Single-mode propagation [2]

- *Multi-mode*: now the core has a diameter of about 50 μm, while that of the cladding is the same as the previous one, with the task of improving the transmission power, but the losses are higher. For this reason, multi-mode fibre is only used for transmitting signals at short distances, where the fibre can conduct several modes of light radiation.

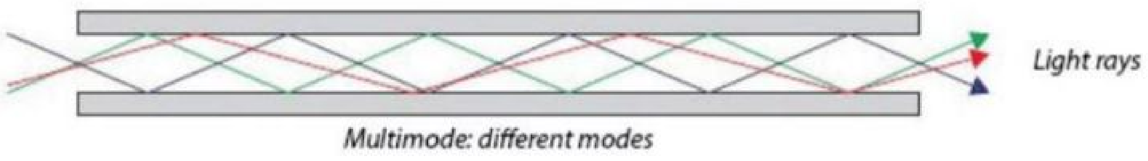


Figure 2.6: Multi-mode propagation [2]

2.1.4 Fibre optic sensors

There are mainly 3 types of sensors that involve this technology:

- *Interferometric*: optical fibres have been investigated at various sensor areas owing to many unique features described before that greatly improve the performance of interferometric systems. This technology is based on the interference of two or more light beams emitted from the same light source propagating through air or different means with different optical paths and arriving simultaneously at a point in space or on the surface of an object. The optical path difference (OPD) due to the perturbation introduced in the sensor influences the phase difference which is directly encoded into the interference fringe patterns in the acquired spectrum. Therefore, by measuring the changes in the interference spectrum, it's possible to obtain information about the changes in optical paths in an optical measurement system;

- *Distributed*: they offer the possibility of monitoring variations of one-dimensional structural physical fields along the entire optical fibre using different phenomena of scattering, such as Rayleigh, Brillouin or Raman. An important advantage regarding distributed sensing is that it only requires a single connection cable to communicate the acquired data to the reading unit in opposite to the large number of otherwise required connecting cables when using discrete sensors. This characteristic makes these sensors more cost-effective and at the same time opens a wide range of important applications such as the continuous (in space and time) monitoring of large civil engineering structures.

Now focusing on the distributed sensors, many types belong to this category, including the FBG sensors too, which are listed here:

- FBG: these are the sensors employed in the measurements treated during the realization of this thesis. The presence of Bragg grating is the peculiarity of these sensors, composed of fibres displaced inside the core and in an orthogonal direction compared to the optical fibre axis; anyway, they are described more accurately further on.

- OTDR (Optical Time Domain Reflectometry) which uses *Rayleigh scattering* that is an interaction of photons with particles with a diameter less large than 1/15 times of light wavelength: this is a method to detect changes in the structural strain from local reflection induced by an optical fibre sensitive to micro bending. A laser diode launches very short pulses into the fibre that acts as a reflector while it is bending. The reflected light is detected by a photodetector coupled with fast sampling electronics. The OTDR sensor is suitable for distributed sensing but also requires relatively expensive instruments and a DAQ system with a high data sampling rate;

- ROTDR (Raman Optical Time Domain Reflectometry): uses the Raman backscattering signal of an optical pulse to obtain environmental information along the sensing fibre, with the pulse width limiting spatial resolution to the meter level in current systems;

- BOTDR (Brillouin Optical Time Domain Reflectometry): it can detect the strain and the temperature information, localised by the return time of the probe pulse, along the fibre based on the spontaneous Brillouin scattering process, which is caused by non-linearities inside the material related to acoustic photons. So, photons can be scattered shifting their frequency, so that the reflected ones have a lower frequency than incident ones, therefore the difference between these two frequencies is the emitted photons frequency. Parameters of the BOTDR system, including the spatial resolution, the signal-to-noise ratio, the measurement speed, and the sensing range, have a mutually restrictive relationship.

2.1.5 FBG sensors

It's a single-mode fibre exposed laterally to periodic light beam impulses whose only a specific wavelength doesn't pass through the Bragg grating, called Bragg's wavelength, so that it will be reflected, except for the rest, so it acts like a filter. The variation of the physical parameter measured by the sensor is related to the deformation of the Bragg grating and consequently to the variation of Bragg's wavelength that can be calculated from the relative frequency:

$$\lambda = 2n_{eff} \cdot \Lambda \tag{2.4}$$

where n_{eff} is the refraction index of the material in which light flows and is subjected to a modulation so that $n_{eff} = n_i + \Delta n$ and Λ is the Bragg period, identified as the distance between those orthogonal fibres showed in figure 2.7

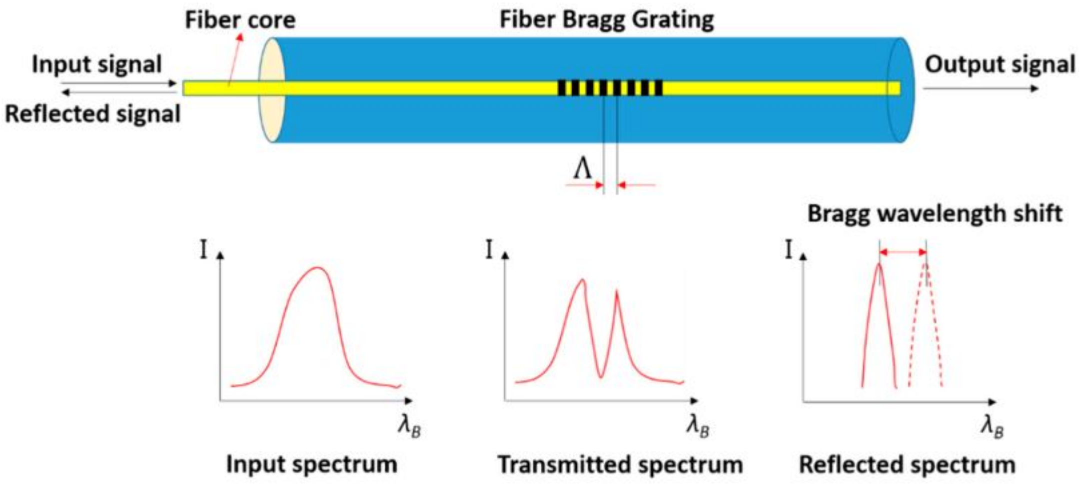


Figure 2.7: Schematic diagram of a fiber Bragg grating (FBG) sensor [7]

2.1.6 Grating structure

The structure of the FBG can vary via the refractive index or the grating period. The grating period can be uniform or graded, and either localised or distributed in a superstructure. The refractive index has two primary characteristics, the refractive index profile, and the offset. Typically, the refractive index profile can be uniform or apodized, and the refractive index offset is positive or zero. There are six common structures for FBGs:

- uniform positive-only index change;

- Gaussian apodized
- raised-cosine apodized
- chirped
- discrete phase shift
- superstructure

2.1.7 Manufacture

Fibre Bragg gratings are created by "inscribing" or "writing" systematic (periodic or aperiodic) variations of refractive index into the core of a special type of optical fibre using an intense ultraviolet (UV) source such as a UV laser, employing two methods, *holographic method* and the *phase masking* but generally the latter is most commonly used. It's based on a diffractive optical instrument which can modulate the UV radiation's period; the fibre is germanium-doped silica due to its photosensitive nature, which means that the refractive index of the core varies when it's exposed to UV light, to induce a permanent refractive index change.

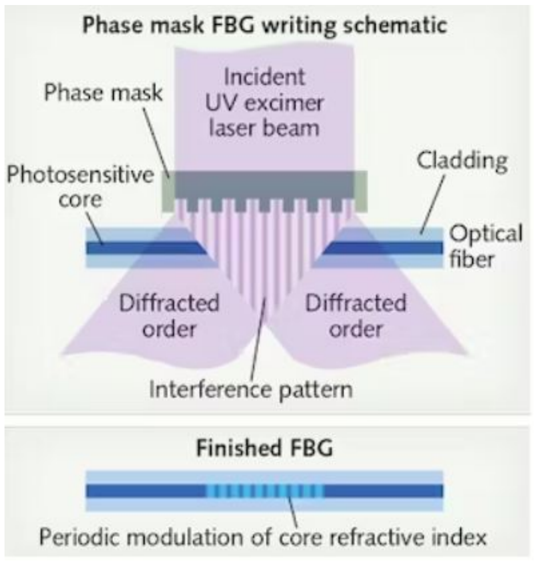


Figure 2.8: Fabrication process of FBG sensors [8]

An excimer laser is employed as the incident lase (a combination of a noble gas and a reactive one); when it is diffracted by the mask, it will generate a succession of high-power and low-power zones, defining the FBG's period which is the distance between the high-intensity zones, half of the phase mask's one. The only disadvantage is that with a mask it's possible to create a FBG sensor with only one Bragg's wavelength, that is the one represented in the graphs of the next chapters.
The holographic method consists of a UV laser split into two beams which interfere with each other creating a periodic intensity distribution along the interference pattern. The refractive index of the photosensitive fibre changes according to the intensity of light that

it is exposed to. This method allows for quick and easy changes to the Bragg wavelength, which is directly related to the interference period and a function of the incident angle of the laser light. This technique can be improved by introducing a prism that varies the angle between the two different UV beams that then interfere, thus enhancing the stability of the fabrication process.

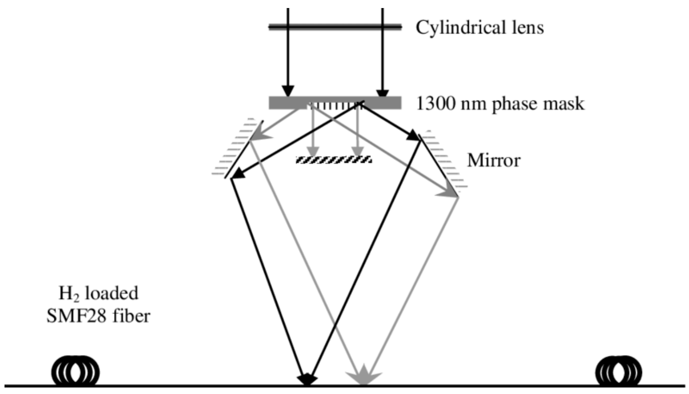


Figure 2.9: The holographic method [10]

2.1.8 FBG encapsulation

Due to the fragility of the optical fibre above-mentioned, it is hard to directly employ FBGs in aerospace systems without any protection, so it is necessary to develop encapsulation techniques for bare FBG strain sensors. Two kinds of encapsulation techniques are developed for FBG strain sensors and one for FBG temperature sensors:

- Capillary encapsulation: the metal holder ring is used to keep the FBG deformation consistent with the concrete structures and the stretched optical fibre is ready for the temperature compensation connector.
- Slice base encapsulation: FBG encapsulated in a slice base by glue, which can be conveniently used on the surface of mental and concrete structures.

Due to that, the encapsulation layers will change the sensitivity of the original FBGs, the encapsulated FBG sensors must be calibrated before they are applied in practical structures.

2.1.9 How a FBG sensor can measure physical quantities

In the initial description of optical fibres, we discussed the many advantages they offer in terms of their behaviour to environmental conditions such as insensitivity to electromagnetic interference, immunity to chemical reactants and resistance to harsh and high-temperature environments, which make it possible for this technology to make measures of strain, temperature, pressure or humidity through the variation of the *Bragg's wavelength*. This is feasible because the *refraction index* changes as a result of, for example, a thermal alteration and through the fibre's thermal expansion variation of the *grating period*, both parameters related to the Bragg's wavelength, as well as mechanical stress that causes elongation or a shortening of the fibre. Also, the humidity can affect that quantity since vapour molecules may enter inside the fibre, changing the refraction index. However, through simple mathematical relations, it is possible to explain the influence of mechanical and thermal stresses on the FBG sensor, starting from the equation 2.4:

$$\frac{\Delta\lambda}{\lambda} = \frac{\Delta n_{eff}\Lambda}{n_{eff}\Lambda} \tag{2.5}$$

A quasi-linear relation between the axial deformation of FBG and Bragg's wavelength can be found for both small mechanical and thermal stresses with an approximation through a first-order Taylor expansion, thus writing:

$$\frac{\Delta\lambda_B}{\lambda_B} = \frac{\Delta n_{eff}\Lambda}{n_{eff}\Lambda} = \left(1 + \frac{1}{n_{eff}}\frac{\delta n_{eff}}{\delta\epsilon}\right)\Delta\epsilon = (1+p_E)\Delta\epsilon \tag{2.6}$$

$$\frac{\Delta\lambda_B}{\lambda_B} = \frac{\Delta n_{eff}\Lambda}{n_{eff}\Lambda} = \left(\frac{1}{\Lambda}\frac{\delta\Lambda}{\delta T} + \frac{1}{n_{eff}}\frac{\delta n_{eff}}{\delta T}\right)\Delta T = (\alpha_\Lambda + \alpha_n)\Delta T \tag{2.7}$$

Combining these two equations it's possible to obtain the relation between Bragg's wavelength, strain and temperature:

$$\Delta\lambda_B = \lambda_B(1+p_E)\Delta\epsilon + \lambda_B(\alpha_\Lambda + \alpha_n)\Delta T \tag{2.8}$$

or in the most common way that is:

$$\Delta\lambda_B = \lambda_B(k_\epsilon\Delta\epsilon + k_T\Delta T) \tag{2.9}$$

where:

- λ_B is the *nominal Bragg's wavelength*, that is the FBG output in normal conditions;
- p_E is the *strain optic coefficient* of the fibre that represents the variation of the refractive index due to axial deformations, depending on the fibre's material;
- α_Λ is the *thermal expansion coefficient* that quantifies the elongation or the shortening of the fibre due to temperature variations and it also depends on the fibre's material and if the fibre is fixed on a support, the right value is that of the support's material;
- α_n is the fibre's *thermo-optic coefficient.* It is intrinsic to the fibre's material since it does not vary depending on the material of any support to which the fibre may be glued.

Chapter 3

State of art of FBGs development

In this chapter, the latest works and results achieved are presented to give a wide overview, pointing out the exploitable basis of the further test campaign in terms of data acquisition systems for FBG sensors and previous tests for thermal and structural applications, so making up a knowledge of a series of data about FGBs' response, which has guided the choices of what could have been improved and how to organise and prepare the experimental tests.

3.1 Data acquisition system

For the Anubi drone, a 4G/5G wireless connection was developed to transmit and store data obtained with FBG sensors, passing through a middleware that is a C/C++ Linux application which receives sensor data and then sends them to the cloud database by using a 4G/5G connection [12]; this acquisition system model can be represented with an elementary scheme here:

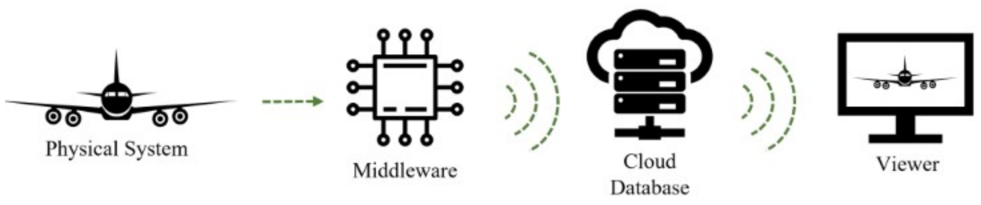


Figure 3.1: Transmitting system on Anubi [12]

The *SmartScan Interrogator* is the instrument used to get data from FBG sensors during this flight test, shown in figure 3.2, that is the device that interacts with the FBGs by sending a light beam and consequently receiving and acquiring their responses in terms of Bragg wavelength values; its functioning is shown here with the whole transmitting network aboard:

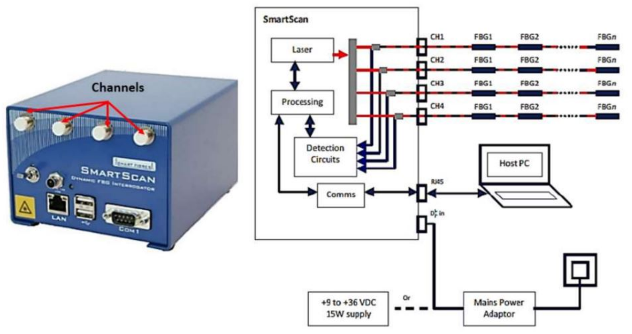


Figure 3.2: Functioning system of the interrogator [11]

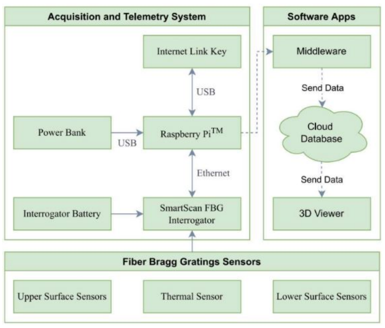
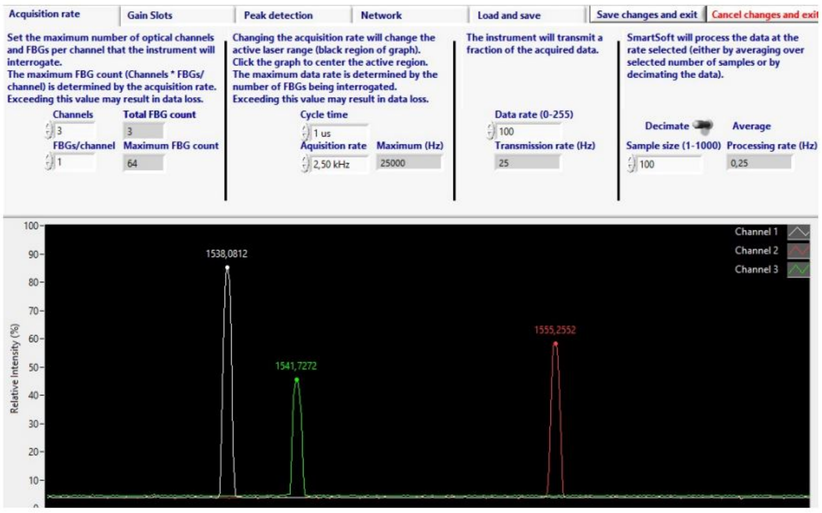


Figure 3.3: Transmitting sequence of FBG data [11]

To transmit and collect these data, the interrogator via an Ethernet cable with *Raspberry Pi Model B+* system on chip (SOC), both supplied by a battery. This SOC runs a *Middleware* which gets data from the interrogator, transmitting them to a cloud database with a certain frequency that is much lower than that one the interrogator can reach itself. Indeed, it's this transmitting frequency that will be the problem for the post-processing as reported further on in the last chapter. However, the interrogator can work contemporary with four independent channels, reading wavelengths from *1528 to 1568 nm*, where a maximum of 16 FBG sensors could be detected for each channel and reaching a maximum frequency of acquisition of *25 kHz*.
Through the software *SmartSoftSSI* installed on a computer, the digital signals coming from the cloud database are elaborated, displaying wavelength trends and saving numerical values in a file *.csv*; it is the only interface between the user and the interrogator and subsequently, the fibres were connected to it.



Set the maximum number of optical channels and FBGs per channel that the instrument will interrogate.
The maximum FBG count (Channels * FBGs/channel) is determined by the acquisition rate.
Exceeding this value may result in data loss.

Channels: 3
Total FBG count: 3
FBGs/channel: 1
Maximum FBG count: 64

Changing the acquisition rate will change the active laser range (black region of graph).
Click the graph to center the active region.
The maximum data rate is determined by the number of FBGs being interrogated.
Exceeding this value may result in data loss.

Cycle time: 1 us
Aquisition rate: 2,50 kHz
Maximum (Hz): 25000

The instrument will transmit a fraction of the acquired data.

Data rate (0-255): 100
Transmission rate (Hz): 25

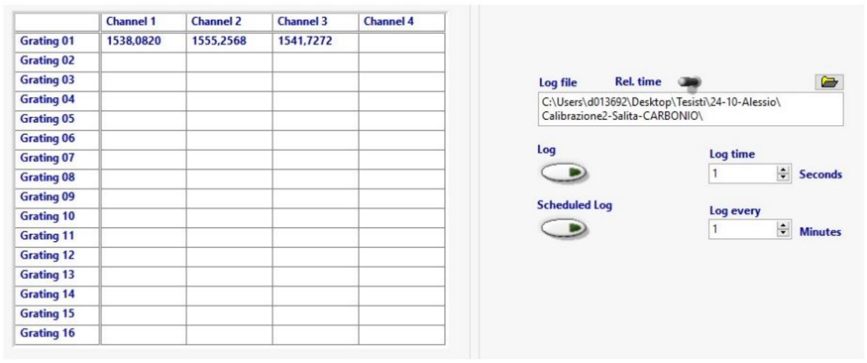
SmartSoft will process the data at the rate selected (either by averaging over selected number of samples or by decimating the data).

Decimate Average
Sample size (1-1000): 100
Processing rate (Hz): 0,25

Figure 3.4: Settings and graphic visualisation

The user can modify several settings in the window *Instrument* to define how the interrogator has to work. For example, the number of channels and FBGs per channel or the chosen numbers of the Bragg's wavelength that should be visualised, graphically in the

figure (3.4) and numerically in the *Basic acquisition* window (figure 3.5). This window is very important because the acquisition cannot start if the user doesn't create a file that will contain the FBG's data, so this step can be done in this window; moreover, it presents the commands *Log* and *Scheduled log* where different times can be set: the first one is the command which starts a single acquisition of the chosen duration, taking into account also the settings decided in the Instrument set up; the second one starts a schedule where a single log is activated after each set period, with the previously decided duration.
Each acquisition elaborates a *.log* type of file, which can be read as a text document; in these files, there are all the acquisitions for each FBG per channel, with the chosen acquisition frequency selected in the figure 3.4.



	Channel 1	Channel 2	Channel 3	Channel 4
Grating 01	1538,0820	1555,2568	1541,7272	
Grating 02				
Grating 03				
Grating 04				
Grating 05				
Grating 06				
Grating 07				
Grating 08				
Grating 09				
Grating 10				
Grating 11				
Grating 12				
Grating 13				
Grating 14				
Grating 15				
Grating 16				

Log file
Rel. time
C:\Users\d013692\Desktop\Tesisti\24-10-Alessio\Calibrazione2-Salita-CARBONIO\
Log
Log time: 1 Seconds
Scheduled Log
Log every: 1 Minutes

Figure 3.5: Channels visualisation

3.2 Studies on FBGs monitoring capabilities

In this section, a series of previous studies and analyses concerning the FBGs' response to structural and thermal stresses were taken into account to better understand which are the results obtained to define a database useful for further tests and a work procedure for managing these sensors. Thermal tests in climatic chambers on different materials, crafting methods and strain tests are reported hereafter, which provided first coefficients k_t and k_ϵ values but above all the demonstration of the response's independence to thermal and strain variations.

3.2.1 The cross-sensitivity problem

As reported before, the Bragg grating reacts both to strain and temperature variations, but also humidity. This implies that when the sensor is applied for structural measurement, the reflected wavelength λ_B by grating will change following a certain strain imposed by a load but also if a temperature variation occurs since there is a linear correlation between them as seen with the equation 2.9; this is the main criticality of FBG, which is named *cross-sensitivity*. Nevertheless, a technique to decouple the different contributions was tested and described in the book *Aerospace Science and Engineering: III Aerospace PhD-Days* [3], presenting as the simplest strategy three different steps, which has already been applied with great successful results:

1) An optical sensor is thermally calibrated;

2) After it has been placed on the component for mechanical measures, it is flanked by a second sensor that is not mechanically stressed, providing only the temperature value (it could be electronic or optic);

3) Data from the first sensor can be filtered by the thermal contribution.

These are the main results:

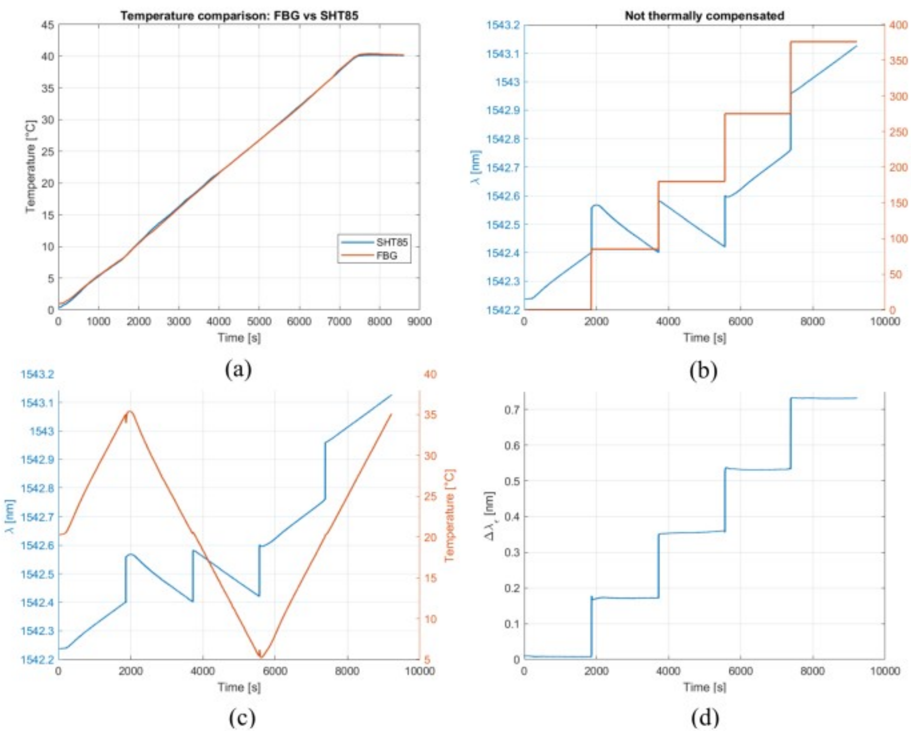
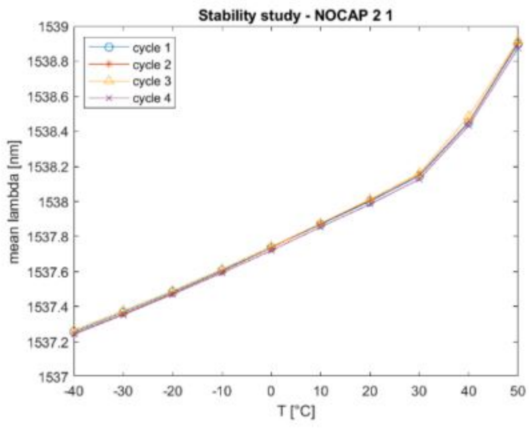


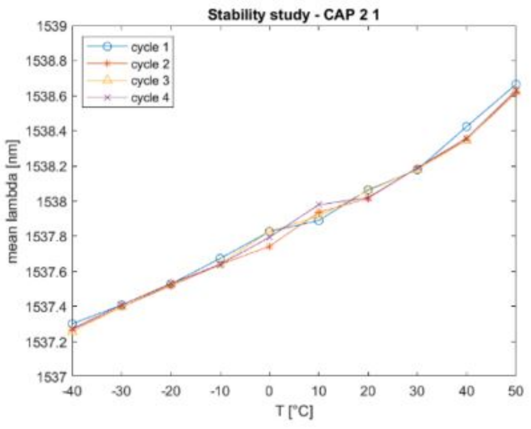
Figure 3.6: Comparison between temperature sensing of FBG and electronic sensor (a), FBG output when mechanical loads and thermal variation contextually act on the sensor (b,c) and after thermal compensation (d) [3]

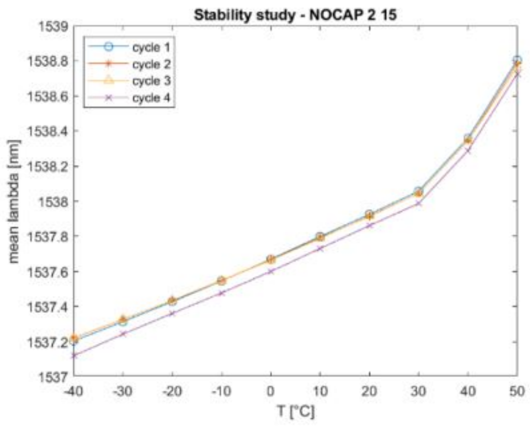
The most important conclusion of this work is that, from the results obtained, it was verified that FBG sensors can measure mechanical strain after the wavelength trend is filtered out by environmental conditions.

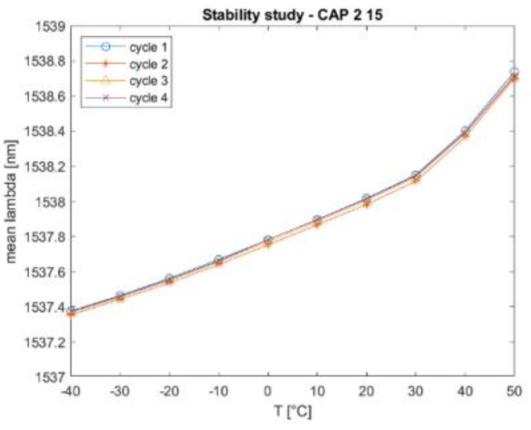
3.2.2 Thermal monitoring

Environmental temperature Concerning the temperature monitoring with FBGs, there was a development of an Outside Air Temperature (OAT) sensor, whose performances were optimised for reliability, accuracy and durability, linked to the necessity for reliable and precise temperature measurements under harsh operational conditions in aircraft applications. Conventionally, these measurements are accomplished using thermocouples, which, despite their reliability, face several limitations such as susceptibility to electromagnetic noise, physical size constraints, and limited multiplexing capabilities, while FBGs are a solution to these problems for their reduced size or immunity to electromagnetic interference [9]. These sensors were crafted embedding the fibres within custom-made aluminium channels, with different thicknesses, realized in $AlSi_{10}Mg$ with Additive Manufacturing Laser Powder Technique; for the bonding approach, two distinct methodologies were adopted, that is sensors secured with glue (silicone resin MasterSil 151©) and others glue-less to consider the effect of mechanical locking (achieved with heat shrink polymers) compared to the classical glueing approach for tubes where this was not fitting due to reduced space for air evacuation. For these sensors, several tests took place in a climatic chamber, executing 10 consecutive cycles from -40 °C to 50 °C, spaced out by a gradient of 10 °C every 30 minutes, reporting how wavelength has changed based on the temperature inside the chamber to define a stability study of the two types of bonding. The first results are about each eight sensors secured with the glue, showing the stability during the first four cycles:









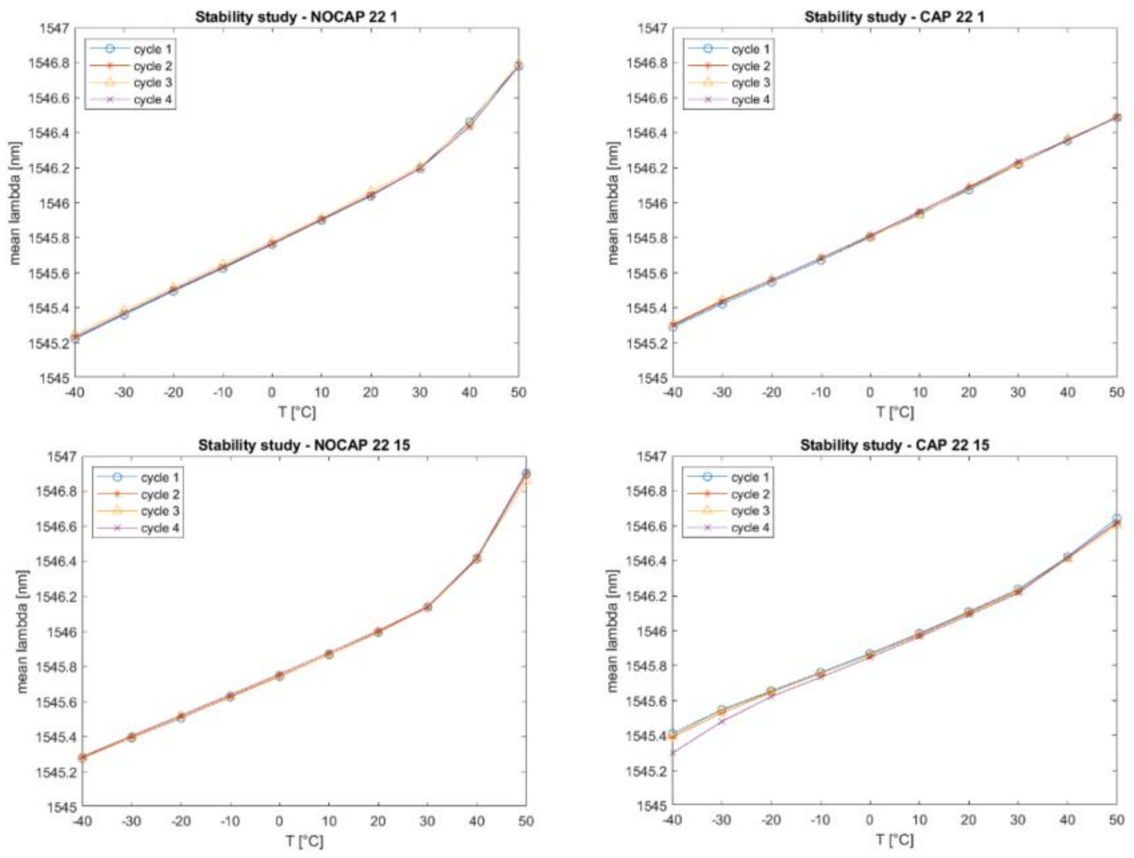


Figure 3.7: Stability study for glue-secured sensors [9]

It's evident that the resin affected the sensors' behaviour, underlined by a deviation from linearity above 25 °C. This so-called 'knee' is caused by a variation of viscosity at high temperatures so that the Bragg's grating is contracting or expanding due to the glue's elasticity in addition to the temperature effects. So, the most important conclusion is that the stability of the fibre's response is inevitably conditioned by the adhesive's viscosity, undermining the reliability of the temperature measurement for OAT applications. In particular, these results indicate that it is preferable to avoid using silicone resin to bond FBG sensors on the metal channels, which also increases the thermal coefficient from the theoretical one that is $k_T = 0.00972\ pm/^{o}C$ as listed in the table below:

NOCAP 2 1	0.01258
CAP 2 1	0.01300
NOCAP 2 15	0.01216
CAP 2 15	0.01083
NOCAP 22 1	0.01365
CAP 22 1	0.01301
NOCAP 22 15	0.01221
CAP 22 15	0.01189

Table 3.1: Thermal coefficient values for glue-secured specimens [9]

Now the focus is on the *mechanical constrained specimens result*, obtained with the same thermal cycle inside the climatic chamber already seen for the previous case:

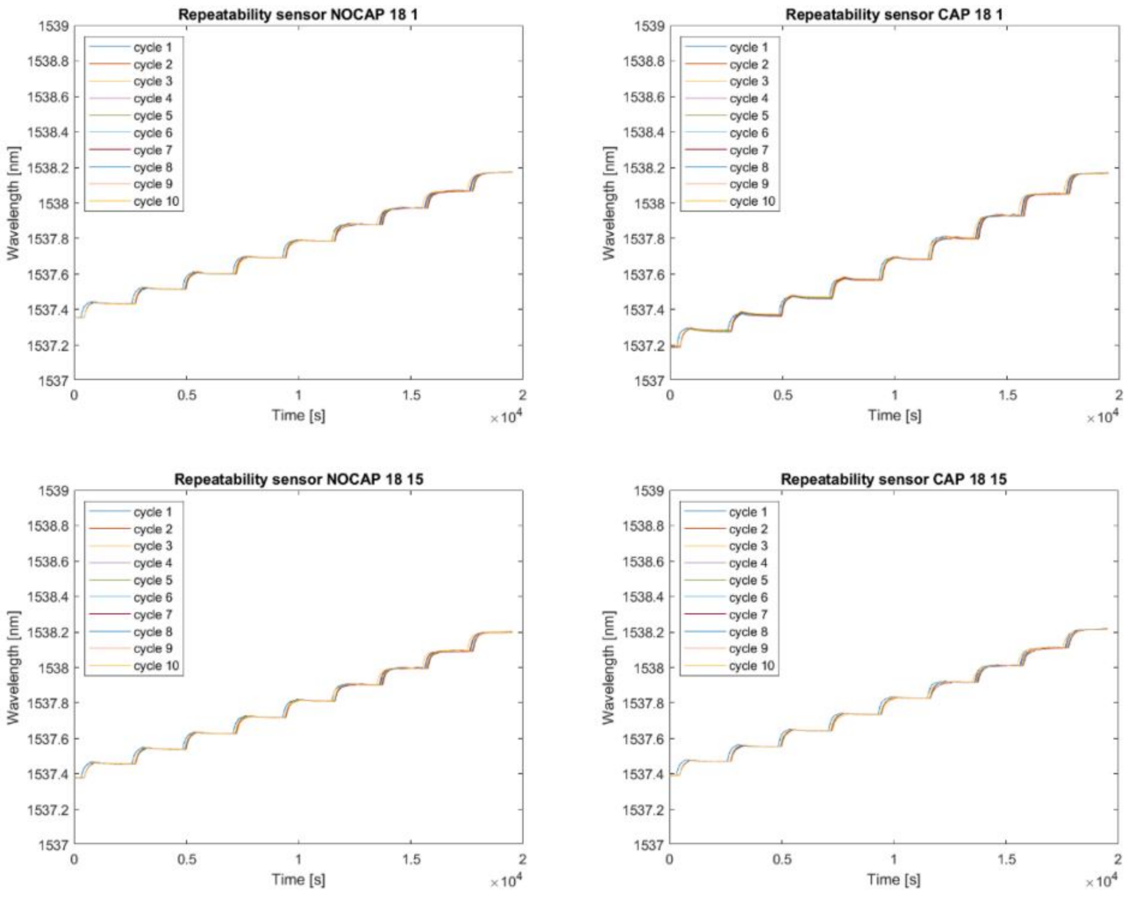


Figure 3.8: Repeatability study for mechanical secured sensors [9]

From the data obtained, one can infer the high repeatability that characterizes these sensors, thus constituting an important first advantage over glue-secured sensors. Focusing the attention on the calibration lines, the mean value of the wavelength for each step is calculated to investigate its correlation with the corresponding mean temperature which represents a linear relationship according to the equation valid in the absence of strain:

$$\Delta\lambda = k_T \Delta T \tag{3.1}$$

Graphic results are shown here, with the corresponding coefficient k_T reported in this table:

NOCAP 18 1	0.00908
CAP 18	0.01091
NOCAP 18 15	0.00910
CAP 18 15	0.00913

Table 3.2: Thermal coefficient values for mechanical secured specimens [9]

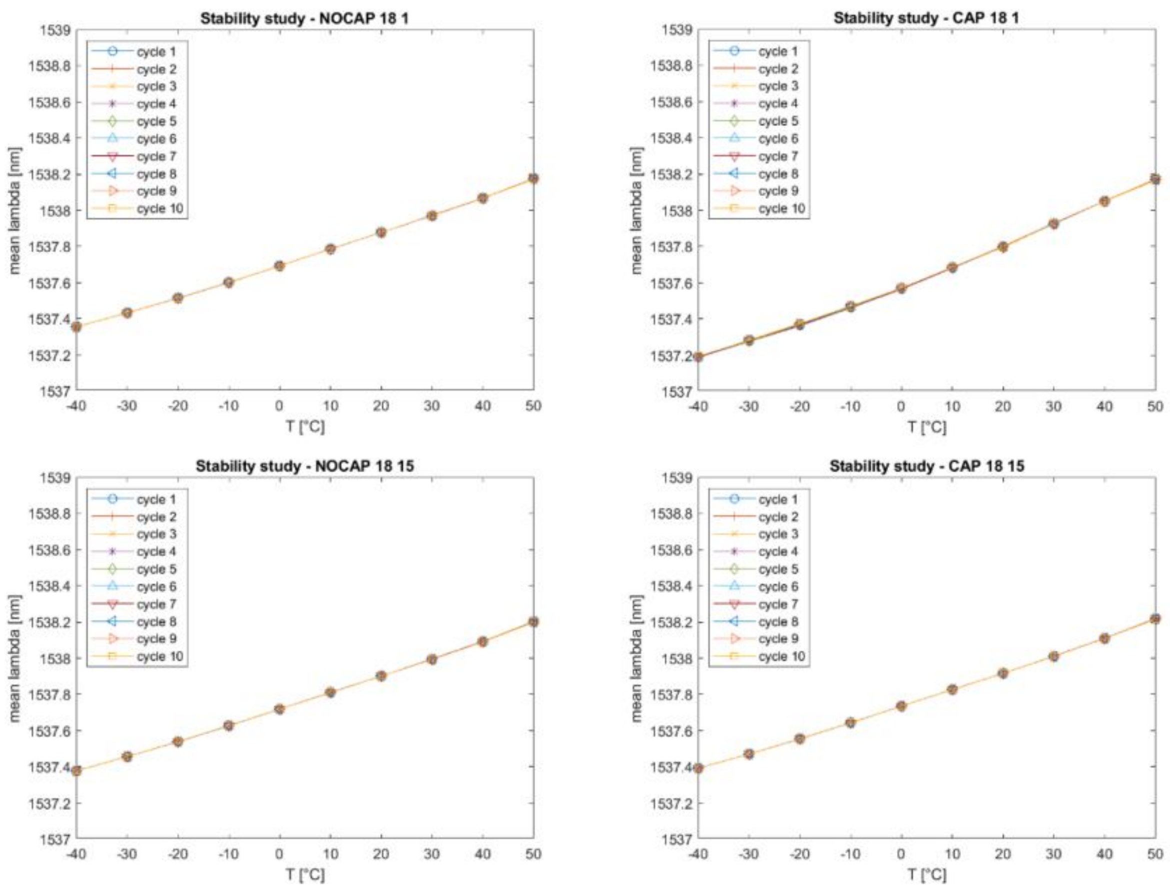


Figure 3.9: Stability study for mechanical secured sensors [9]

These graphs demonstrate that the linear correlation between wavelength and temperature is part of these sensors as expected, in addition to satisfactory stability over the cycles, both for the temperature inside the climate chamber and the fibre's response. The linear fit's proportionality coefficient k_T calculated for each FBG sensor is very similar to the theoretical value, with an admissible deviation of 10 %.

Bounding procedures for thermal applications Furthermore, during this previous test campaign, different bounding techniques were compared and the effects of fibre pre-tensioned were analysed on a composite and aluminium plate, which are the most common materials on an aircraft [6]. The results showed a possible influence on the correlation coefficients but with minimal difference in the final accuracy.

The pre-tensioned resined FBG on aluminium has resulted as the best possible configuration:

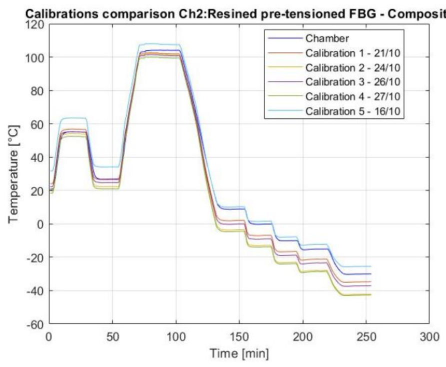


Figure 3.10: Calibration of pre-tensioned FBG on aluminium plate [6]

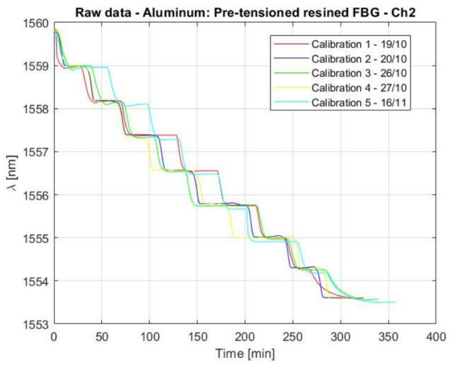


Figure 3.11: Verification of repeatability of pre-tensioned FBG [6]

It offers great repeatability and accuracy performances; indeed, a $RMSE = 2.6465$ was calculated for this case, while the Kapton-fixed FBG and the not tensioned glued FBG on composite have an optimal accuracy ($RMSE = 3.3292$ and $RMSE = 3.3373$ respectively) but not an equally good repeatability. At last, the glued not tensioned FBG on aluminium showed both good repeatability and accuracy after the end of the glue settling, but at the same time, this fixing method was not reliable since a sort of break occurred at very low temperatures, after a few thermal cycles. Anyway, overall, it was assumed that all these specimens meanly had better repeatability than the specimens on aluminium, since, the λ values at each step were very similar throughout the various calibrations as shown here:

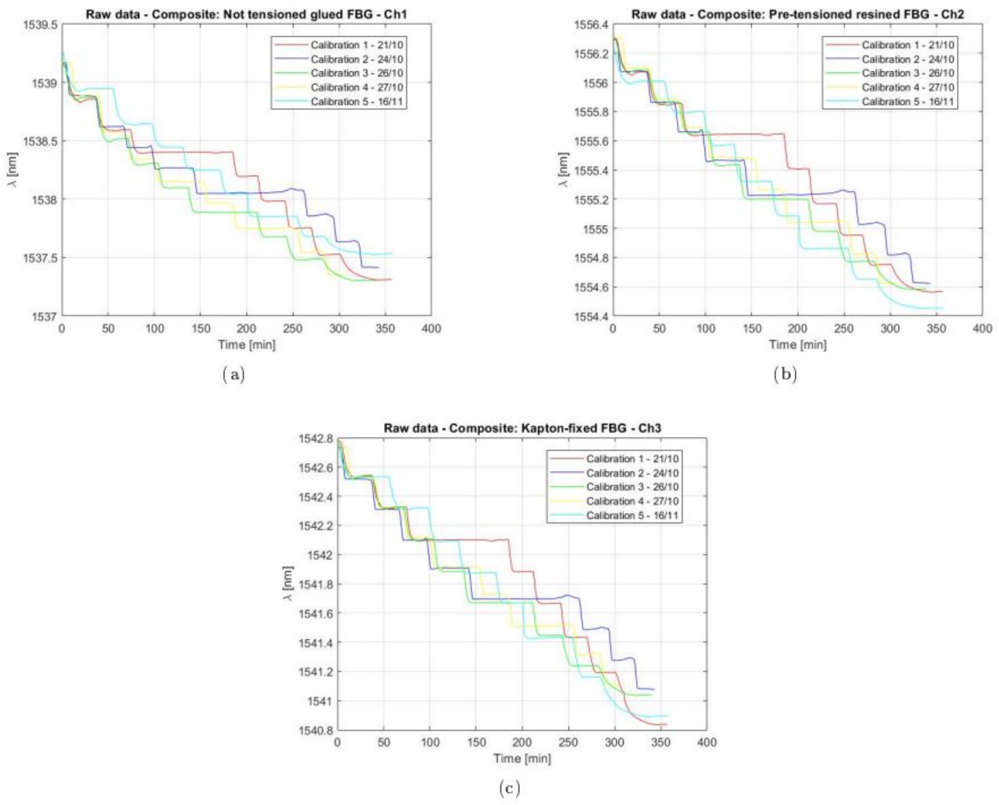


Figure 3.12: Raw data on composite plate [6]

3.2.3 Structure monitoring

Regarding this topic, the optical fibre transmission lines along with FBG sensors can offer multiple advantages compared to conventional ones since in traditional structural health monitoring systems, different types of sensors are used: strain gauges, piezoelectric sensors, ultrasonic sensors etc. These commonly employed sensors require an electrical transmission line to be connected to the monitoring system, setting up the main downside of the traditional monitoring system. Moreover, some of the main disadvantages are represented by a considerable amount of weight due to the materials used (i.e. copper wires), vulnerability to electromagnetic interference, relatively low bandwidth, low corrosion resistance and all the potential problems related to electrical wiring.
About the application of FBG sensors for structural monitoring, the work *Analysis of FBG Sensors Performances When Integrated Using Different Methods for Health and Structural Monitoring in Aerospace Applications* [4] was really useful to start from; indeed, the linear relationship between applied load and wavelength λ was verified and the linear coefficient was calculated for both pre-loaded and not pre-loaded fibres and both traction (gradient $k = 0.002238$ taken as mean) and compression ($k = -0.002303$) state:

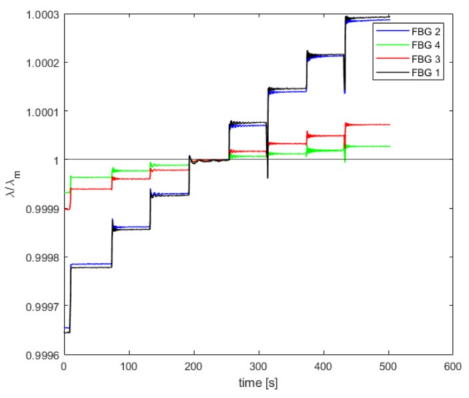


Figure 3.13: Traction test raw data [4]

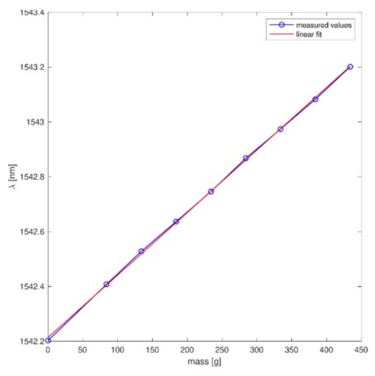


Figure 3.14: Traction test linear fit [4]

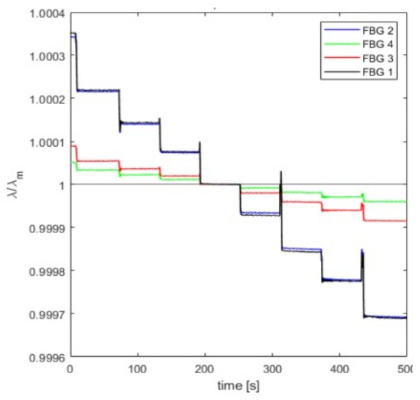


Figure 3.15: Compression test raw data [4]

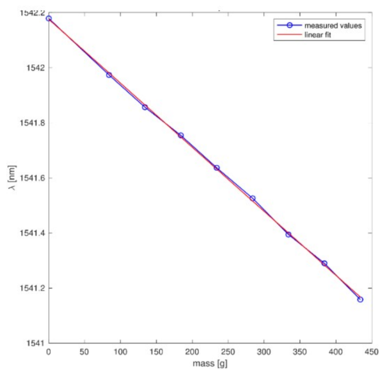


Figure 3.16: Compression test linear fit [4]

Chapter 4

Calibration of temperature sensors

4.1 Introduction

Regarding the purpose in engaging FBG sensors on UAVs to monitor its systems and subsystems, it was useful to do some tests on the behaviour under heat stress, divided into two parts: thermal cycles in a climatic chamber to define a satisfying calibration and tests on the Anubi drone. Since we have to stick these sensors on some devices as the battery that supplies the electrical motor of the Anubi drone, it was essential to think of suitable packaging for it to reduce its risk of braking during a flight but this can entail a different linear coefficient k_T. So, this first part was set to verify that this coefficient doesn't change between sensors with and without packaging, demonstrating that an additional calibration of the sensor can be avoided, standing for a relevant advantage of this technology; this would make it possible to decrease potentially the setting and maintenance time that is crucial for aerospace costs and service quality.

4.2 Experimental setup - solution 1

The packaging was realised with double aluminium layers (3x3 cm) joined with double-sided tape, the latter square shape cut in the centre where the sensor is placed so that it could stretch with heat and vary its temperature linear coefficient, reminding the equation 2.9. This solution was pondered to protect it and to carry out its detection in a faster and more accurate way as compared to a previous pipe realized in additive, because of the low thermal inertia and high thermal conductivity of aluminium; this packaging was stabilized with hot glue and a red heat-shrinkable sheath so that it could have remained flat while air was blown in the climatic chamber.

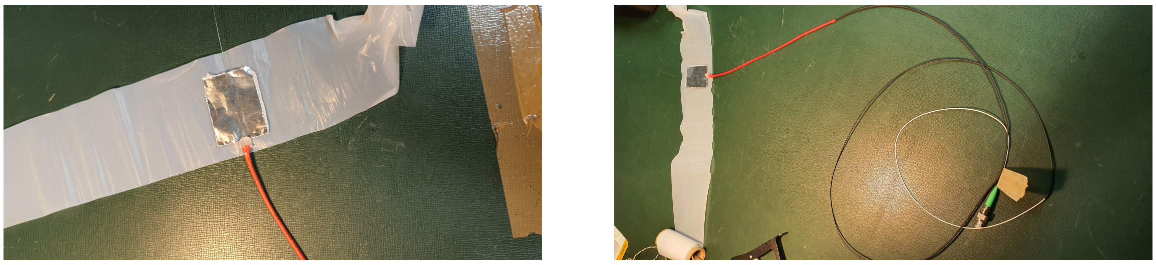


Figure 4.1: Packaging of FBG sensor for temperature tests in the climatic chamber

The climatic chamber employed is the KK-50 CHLT built by 'Beger Laboratory Equipment' which allows an accurate simulation of the environmental conditions with the following characteristics:

Description	Value
Weight	125 kg
Volume	50 L
Temperature range	[-40; +180 °C]
Sensibility	0.1 C°
Hotting speed	5.3 °C/min
Cooling speed	2.1 °C/min

Table 4.1: Climatic chamber's physical characteristics

Its working cycle and structure are displayed here:

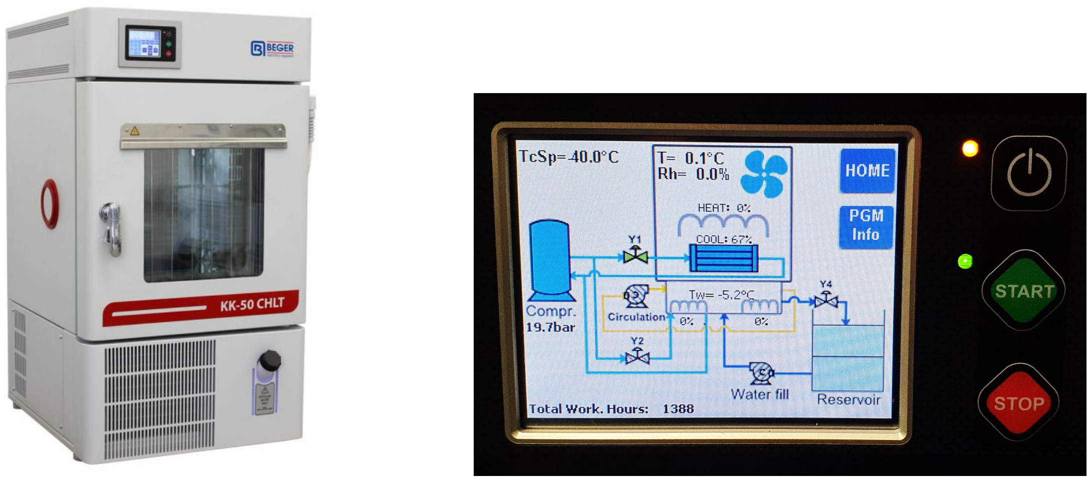


Figure 4.2: Climatic chamber KK-50 CHLT

The climatic chamber is calibrated to a temperature gradient varying from -40 °C to +50 °C; the experimental procedure comprises 9 stages, beginning from the lowest temperature and incrementally reaching the highest, characterized by a stable temperature that persists for thirty minutes. Upon conclusion of this time frame, the succeeding stage starts, elevating the temperature by 10 °C; this cycle has been repeated 2 times. These extremes of temperature were set to not melt the glue and to do a sufficient statistic number of

heat tests, acquiring at 2.5 Hz; resuming, the chamber decreased its temperature to -40 °C and then increased it to 50 °C.
Along with the chamber and its probe, the user can exploit its proper software *KK-Tool*, to collect and read the temperature measured by the chamber's sensor. An auxiliary display has been used to monitor the chamber's temperature working (figure 4.3 The climatic chamber must be connected to the computer to collect the data, generating a file *.DAT*, that can be read as a text file, updated with a new temperature measure each time a user-set timer expires, that is if a sixty seconds timer is set, the chamber's software would update the file *.DAT* every sixty seconds.
Data acquired from the two FBG sensors were obtained with the SmartSoftSSI software and displayed on monitors as below:

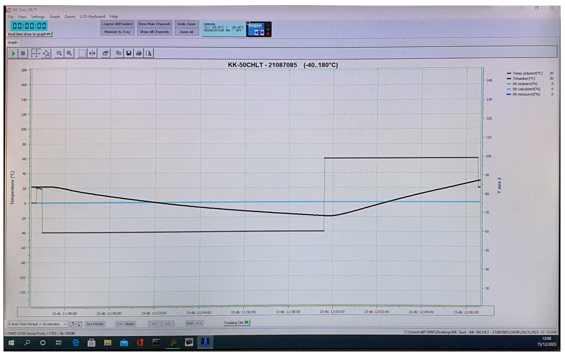


Figure 4.3: Temperature's trend on the chamber software

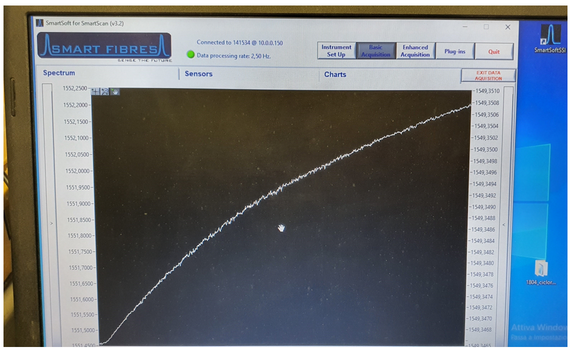


Figure 4.4: Wavelength's trend on SmartScan

From these figures, it already is evident a good similarity between the two trends, underlining the goodness of the correlation and measuring of this FBG sensor but not completely satisfying; indeed, with a look at the second figure, it's clear a change of slope, caused by a not perfect detection of change of the temperature; anyway, more specific analyses were conducted in Matlab to explain better what happened.

4.2.1 Post-processing

Data coming from FBG sensors were analysed in Matlab with *CC_Analysis.m* code which let us display the linear regression lines to assure that the calibration was acceptable in such a way that the same coefficient k_T can be used for further applications but also what happened with the acquisition of temperature data saved on KK-Tool software to verify if the cycles imposed were fulfilled correctly by the climatic chamber.

4.2.2 Results and comments

So, eventually, it was possible to extract the linear regression line with data imported and processed with Matlab, calculating also the R-factor with the *Curve Fitter* tool. that is a reasonable parameter to estimate the pertinence of this calibration. The main graphs are shown here:

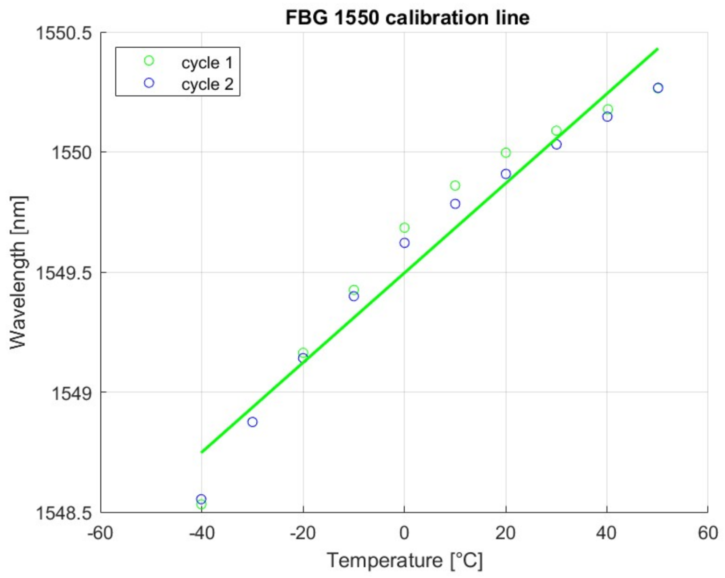
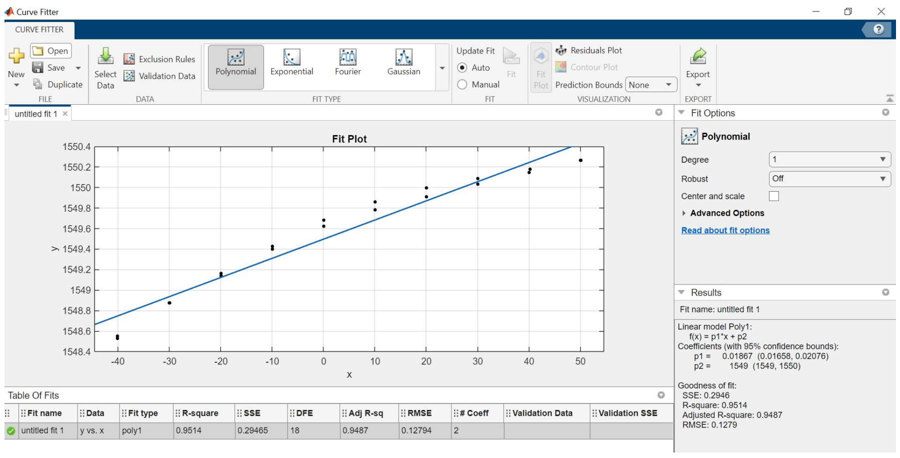


Figure 4.5: Linear regression line of 1st calibration



Polynomial

Degree: 1

Robust: Off

Center and scale: ☐

Advanced Options

Read about fit options

Results

Fit name: untitled fit 1

```
Linear model Poly1:
   f(x) = p1*x + p2
Coefficients (with 95% confidence bounds):
     p1 =    0.01867  (0.01658, 0.02076)
     p2 =       1549  (1549, 1550)

Goodness of fit:
  SSE: 0.2946
  R-square: 0.9514
  Adjusted R-square: 0.9487
  RMSE: 0.1279
```

Table Of Fits

	Fit name	Data	Fit type	R-square	SSE	DFE	Adj R-sq	RMSE	# Coeff	Validation Data	Validation SSE
✓	untitled fit 1	y vs. x	poly1	0.9514	0.29465	18	0.9487	0.12794	2		

Figure 4.6: Curve fitting of the calibration

It's easy to spot that the calibration of the sensor covered with aluminium (1550 is the basic λ_B) wasn't adequate because there is a variation in the slope between -10 °C and 30 °C, indeed the gap compared to the linear regression line is ample and it's evident with the value of the R-factor that is 0.9514, not sufficient for the aerospace sector.
With this result, a new calibration of a potential packaged sensor was mandatory since the response would not have been linear in further tests, besides the fact that the calculated coefficient is quite different from the theoretical one. After an analysis, it was assumed that this effect was due to the necking of the coating around the fibre core which changed its temperature sensibility; this is most likely caused by the change in coating deformation compared to the core, due to the different coefficients of thermal expansion which made polymethyl methacrylate assuming a plastic behaviour, while the glass fibre was still straining elastically, changing accordingly the whole sensor deformation. Another minor unwanted effect was the imperfect flatness of the packaging during the blowing of the air

inside the climatic chamber that moved it up and down, thus changing the accuracy of the measure; this is an important aspect regarding the sticking of the sensor on some devices placed on the UAV so that it must be placed correctly to not modify temperature data acquisition, in particular during the flight when vibrations and g-forces reach significant values, leading to a motion of the sensor.
However, the most important conclusion regarded necking and how to avoid it to improve the sensor's sensibility, finding a structural solution on the fibre but also the packaging that covers it.

4.3 Experimental setup - solution 2

As a possible solution to the necking problem, the coating of the fibre wrapped around the core was removed where the sensor is located along the fibre and a part of the double-sided tape was cut obtaining a U-shape. This was done to make the sensor freer when there is a variation of the temperature but enough to not extend too much, otherwise, it would have detected also a mechanical deformation as seen in the equation 2.9, conditioning the wavelength value. Moreover, the glue used in the joint between the packaging and sheath was removed to test higher temperature; indeed, in this second attempt, the higher temperature reached was 60 °C, which could have melted the glue. The rest of the experimental setup was the same and also in this case a look at the trends realised on SmartSoftSSI was useful:

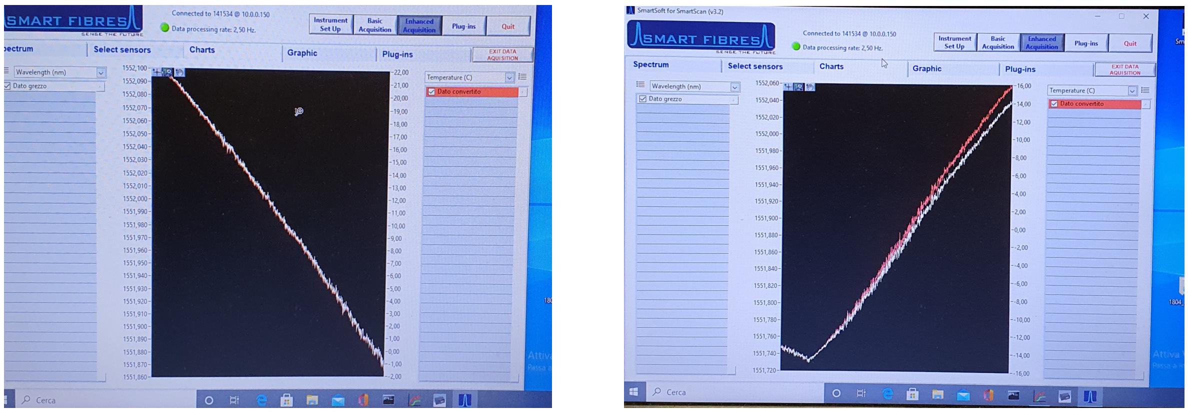


Figure 4.7: Wavelength trends of FBG sensor during cooling and heating

In this case, there wasn't a considerable variation in the slope, letting suppose that the calibration was satisfying and that the measuring between the free and packaged sensor was the same without gaps of temperature values, but this is more evident and accurate with the next graphics.

4.3.1 Results and comments

As done before, *CC_Analysis.m* code was employed to analyze the temperature data coming from the climatic chamber and interrogator to obtain the calibration line of the FBG sensor (now with a Bragg wavelength of 1552 nm). So, first of all, it's now displayed the calibration line:

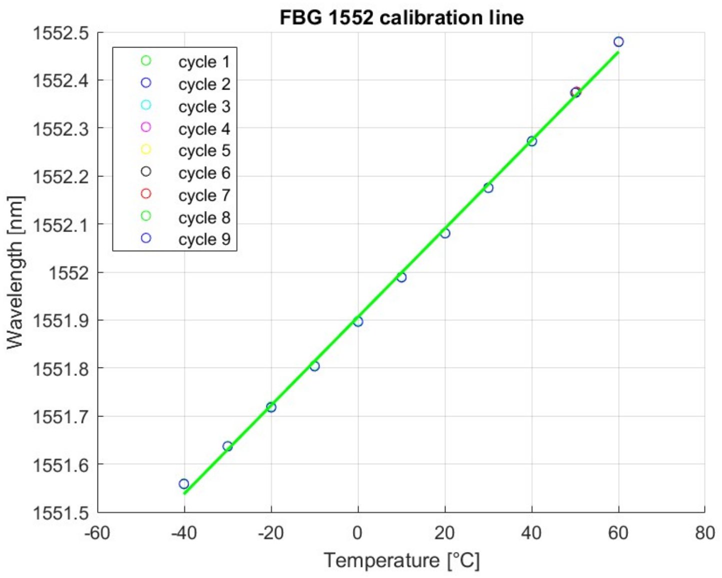
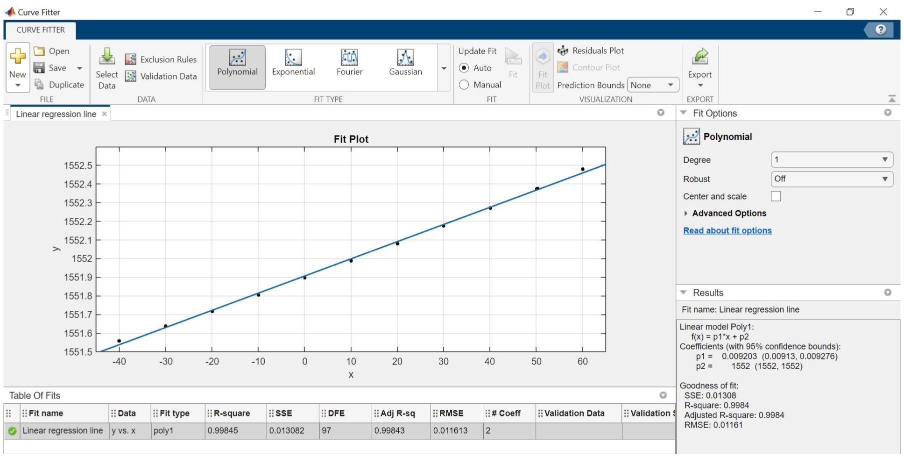


Figure 4.8: Linear regression line of the 2nd calibration

The result obtained is very remarkable because it's clear that there is a suitable correlation between wavelengths and the temperature of the chamber. Indeed, there are no more types of non-linearity as the knee seen before, showing instead points distributed almost perfectly along a line, generating in this way a linear regression line that is supposed to have a good R-factor; this aspect has been verified with the *Curve Fitter* tool:



Results

Fit name: Linear regression line

```
Linear model Poly1:
    f(x) = p1*x + p2
Coefficients (with 95% confidence bounds):
     p1 =   0.009203  (0.00913, 0.009276)
     p2 =       1552  (1552, 1552)

Goodness of fit:
  SSE: 0.01308
  R-square: 0.9984
  Adjusted R-square: 0.9984
  RMSE: 0.01161
```

Table Of Fits

Fit name	Data	Fit type	R-square	SSE	DFE	Adj R-sq	RMSE	# Coeff	Validation Data	Validation
✓ Linear regression line	y vs. x	poly1	0.99845	0.013082	97	0.99843	0.011613	2		

Figure 4.9: Curve fitting of FBG 1552

So, $R = 0.9984$ and $RMSE = 0.01161$ were obtained for this linear regression line, which are very considerable values, especially for the aerospace sector, so it could be assumed that this calibration was satisfying for the main purpose of the work. According to this result, two important parameters were found out to convert later wavelengths detected into temperature data:

- $\lambda_0 = 1551.90\ nm$
- $k_T = 0.0092\ pm/^oC$

This k_T stands with the theoretical value $k_T = 0.00972\ pm/^oC$; this confirms that this calibration was done correctly but above all that these two thermal coefficients are very similar, indicating that this packaged FBG sensor can be employed for monitoring a kind of thermal device. However, other tests were done in the climatic chamber to get enough data to affirm that statistically, the calibration of the packaged sensor is valuable for the application, with the same FBG sensor and climatic chamber and varying the temperature between -40 °C and 60 °C, with a step of 10 °C as displayed here with the linear regression line:

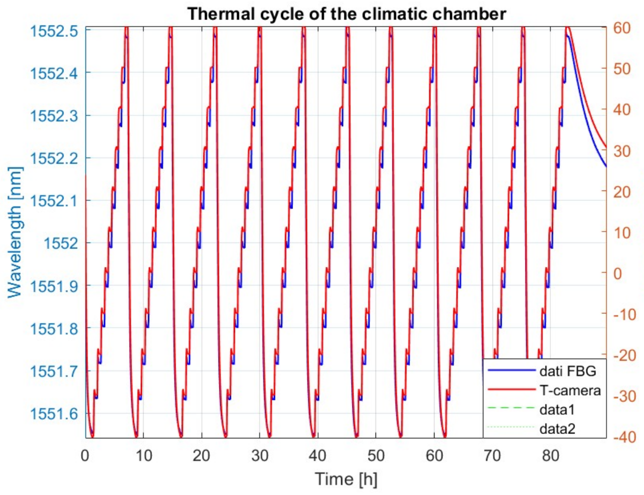


Figure 4.10: Comparison between wavelength and temperature

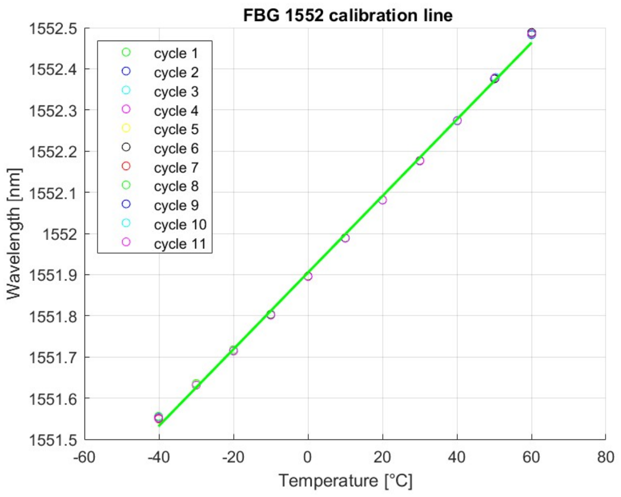


Figure 4.11: Linear regression line

As well as before, the main coefficients that are tabulated here:

Coefficient	Value
λ_0	1551.905 nm
k_T	0.009307 pm/°C
R-factor	0.9985
RMSE	0.01136

Table 4.2: Data of the 3rd calibration

These data are very similar to the previous ones, underlining the goodness of this new layout of the sensor, both for the thermal coefficient that is comparable with the one of the nacked sensor and for the high property of the linear relation between wavelength and temperature of the climatic chamber.

4.3.2 Comparison between temperature data

With data achieved in these tests, the wavelengths detected by the packaged FBG sensor were transformed into temperatures with the equation:

$$T_i = \frac{\lambda_i - \lambda_0}{k_T} \tag{4.1}$$

So, once this calibration was completed, data coming from the probe inside the climatic chamber and FBG sensor were compared defining manually a cycle that could have replied the usual temperature conditions during the a flight test in summer; moreover, the RMSE (root-mean-square error) was calculated as:

$$RMSE = \sqrt{\frac{\sum_{i=0}^{N}(T_{chamber}(i) - T_{FBG}(i))^2}{N}} \tag{4.2}$$

which result is $RMSE = 0.1013$ for the 2nd calibration, leading to a remarkable result as shown in this figure, with data synchronized at UTC:

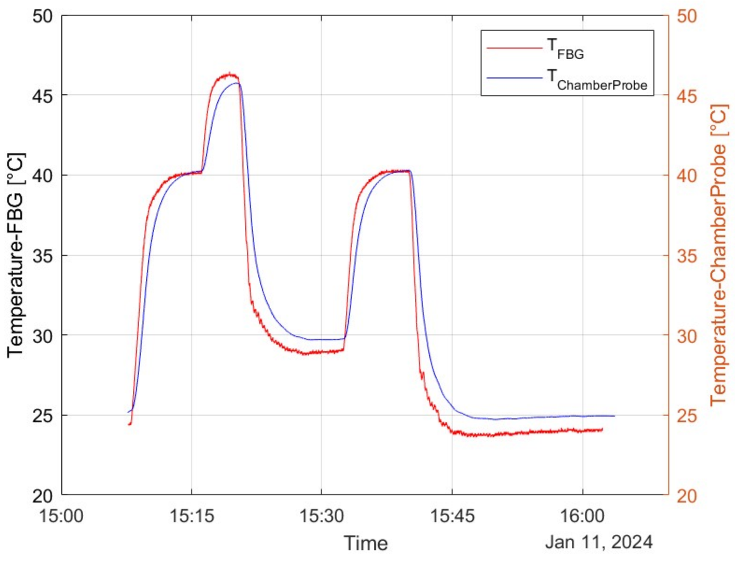


Figure 4.12: Temperature comparison between the chamber probe (blue line) and FBG sensor (red line) of the 2nd calibration

There is a considerable worthy correlation between the two trends, although some differences are present at 25 °C and 30 °C, making suppose that the calibration would have been improved around those temperatures; also the different but slight location of the probe and sensor and the margin of error of 1 °C concerning the probe affected that gap of bit less than 1 °C between the two lines, that is eventually acceptable considering these factors.

Another comparison was done also for the 3rd calibration, with $RMSE = 0.0435$ obtained in this case, which is a result much better than the previous one that confirms the elevated R-factor calculated and so the effectiveness of this calibration, that can be demonstrated with this comparison:

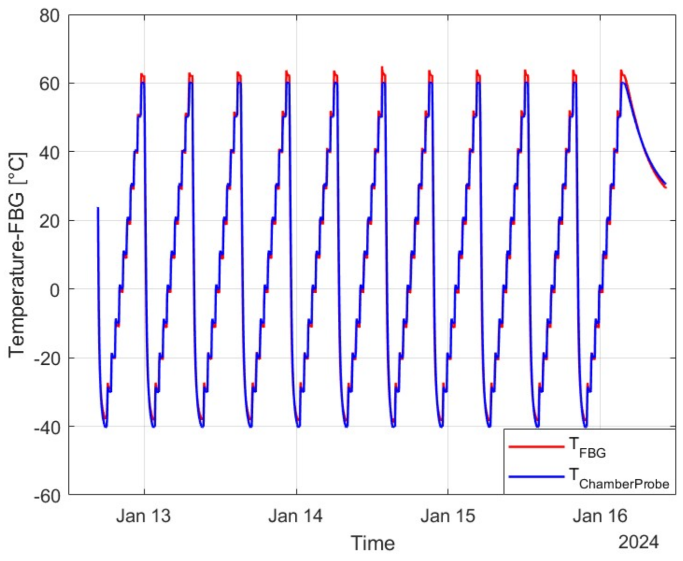


Figure 4.13: Temperature comparison between the chamber probe (blue line) and FBG sensor (red line) of the 3rd calibration

The trends between the two detected temperatures are very similar, above all where the temperature drops down from 60 °C to -40 °C for the beginning of another cycle, so when there aren't any surges of temperature but just a trend of the 1st order that characterizes a thermal probe; while mainly at the outer of the cycles there are differences of 1.5/2 °C caused by the dissimilarity of the two thermal inertia which nevertheless is more evident when bumps of 10 °C are imposed at the climatic chamber, so leading to a great response also in short transients.

This result is very promising for this experimental work because it's demonstrated that thermal FBG sensors (packaged) can be used to measure and accordingly monitor the thermal condition of some devices that make up the propulsion system of the UAV Anubi, which is described and tested in the next chapter.

4.4 Calibration of a sensor without packaging

With the good results obtained from the previous calibration, it was also carried out for the sensor with $\lambda_B = 1530\ nm$ that was used to monitor the Anubi's propulsion system analyzed in the next chapter. This sensor has no cover made of aluminium or other material but is completely exposed to the external environment, so it can be shown that the use or non-use of a sensor cover does not change its behaviour. Calibration in the thermal chamber took place through three cycles, each of which involved steps of 20 °C starting from 10 °C up to 50 °C, to simulate the temperature ranges that may occur during a flight test as shown in this figure:

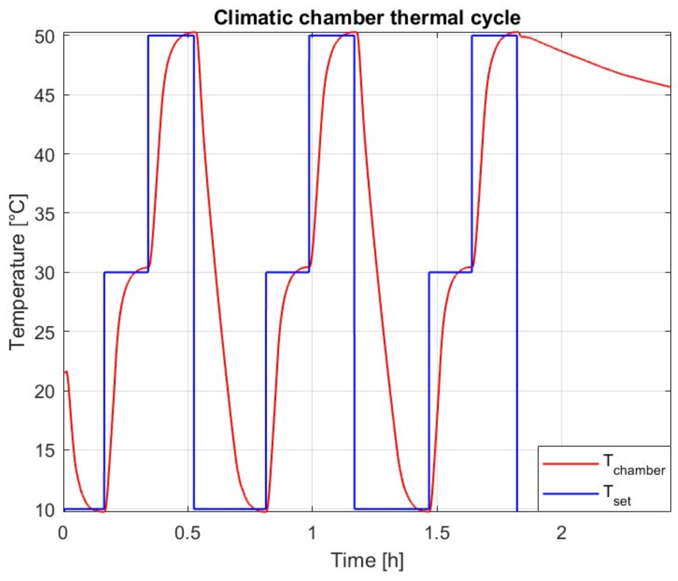


Figure 4.14: Thermal cycle for the calibration of the sensor 1530

4.4.1 Results and comments

As was the case with the 1552 sensor, the calibration line was created through the *CC_Analysis.m* code, through the *CurveFitter* tool the correlation factor R was calculated to get a clear idea of the thermal linearity of this sensor and with the code *Lambda2temp.m* the conversion from wavelength to temperature was made to compare it with that detected by the probe in the thermal chamber, so the results are now displayed:

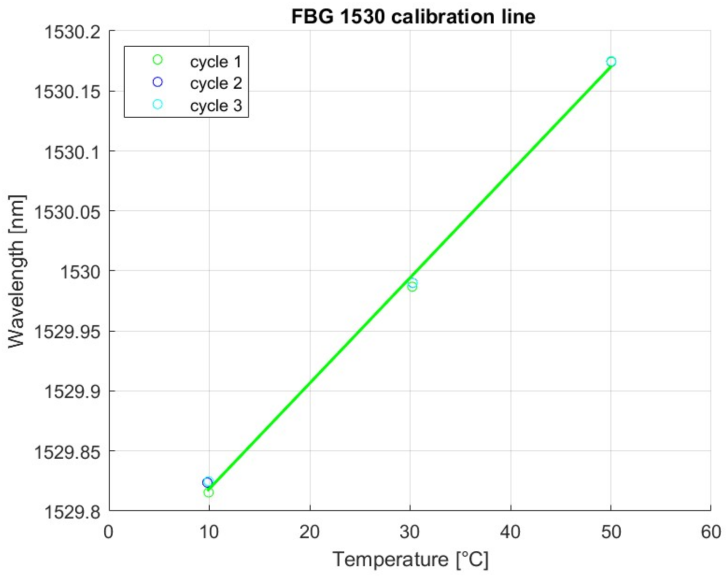
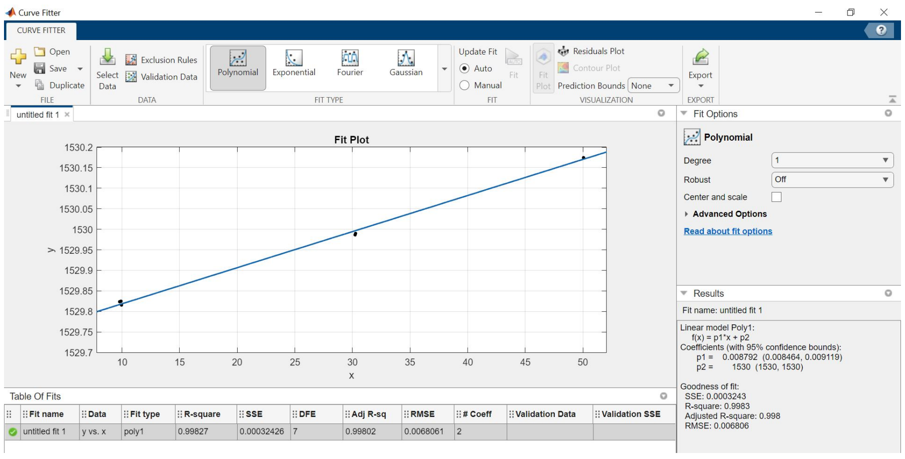


Figure 4.15: Linear regression line of sensor 1530



Results

Fit name: untitled fit 1

```
Linear model Poly1:
    f(x) = p1*x + p2
Coefficients (with 95% confidence bounds):
      p1 =    0.008792  (0.008464, 0.009119)
      p2 =        1530  (1530, 1530)

Goodness of fit:
  SSE: 0.0003243
  R-square: 0.9983
  Adjusted R-square: 0.998
  RMSE: 0.006806
```

Table Of Fits

	Fit name	Data	Fit type	R-square	SSE	DFE	Adj R-sq	RMSE	# Coeff	Validation Data	Validation SSE
✓	untitled fit 1	y vs. x	poly1	0.99827	0.00032426	7	0.99802	0.0068061	2		

Figure 4.16: Calculation of R-factor

As can be easily guessed, there is a good linear relationship between wavelength and temperature in the climate chamber, confirmed by the excellent R-factor of 0.998 which confirms the good quality of this calibration too, together with the thermal coefficient obtained $k_T = 0.0088\ pm/^oC$, which differs from the theoretical value for just 9%.
The last check concerns the comparison between the temperatures detected by the probe and FBG sensor:

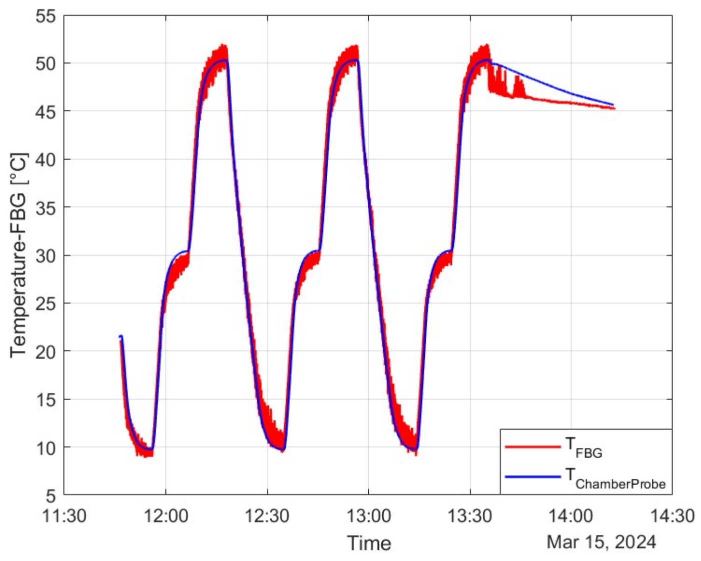


Figure 4.17: Temperature comparison between the chamber probe and FBG sensor 1530

It is clear that the comparison is very positive, net of those series of oscillations caused by the fact that the sensor was not fixed on some structure but was completely free to vibrate as a result of the escape of airflow inside the chamber; anyway, the trends are very similar each other, thus this sensor has been used for the following temperature tests on Anubi treated in the next chapter.

Chapter 5

Temperature monitoring on the propulsion system

5.1 Introduction

Once the calibration part was completed, this chapter analyses the thermal response of the sensors constructed; in fact, after the considerable calibration conducted, these sensors were applied to critical components of the propulsion system of the Anubi drone for thermal monitoring. After a preliminary analysis of the propulsion system, the power supply battery and the *Electronic Speed Control* (ESC) were the devices chosen for this test as thermally they are the most critical, so for them, it was crucial to verify the applicability of FBGs in terms of aerospace systems diagnostics. Three different tests took place, where the data from FBG sensors were acquired with the identical system described in Chapter 5 and then post-processed with MatLab, thus obtaining representative graphs of the wavelengths, then converted to the battery and ESC's temperature data to evaluate the thermal trend.

5.2 Description of the propulsion system

The propulsion system of Anubis is essentially composed of 3 main components:

- *AXI 2826/10 GOLD LINE V3* electrical motor: a brushless motor with neodymium magnets and a rotating case, designed to transfer extremely high torque allowing to rotate large diameter and pitch propellers without the need for a gearbox;

- *FullPower PRO 85A SBEC* Electronic Speed Control (ESC): is an electronic circuit that controls and regulates the speed of an electric motor through the current flowing in the motor windings. It may also provide reversing of the motor and dynamic braking;

- *WELLPOWER LIPO ULTIMA 5200 MAH / 11,1 VOLT 3S 35/70C CH8* supply battery pack: abbreviated as LiPo, it's a rechargeable battery of lithium-ion technology using a polymer electrolyte instead of a liquid electrolyte, providing the specific energy to the ESC, higher than other lithium battery types and is used in applications where weight is a critical feature, as in this case.

The whole functioning system can be shown in the following figure:

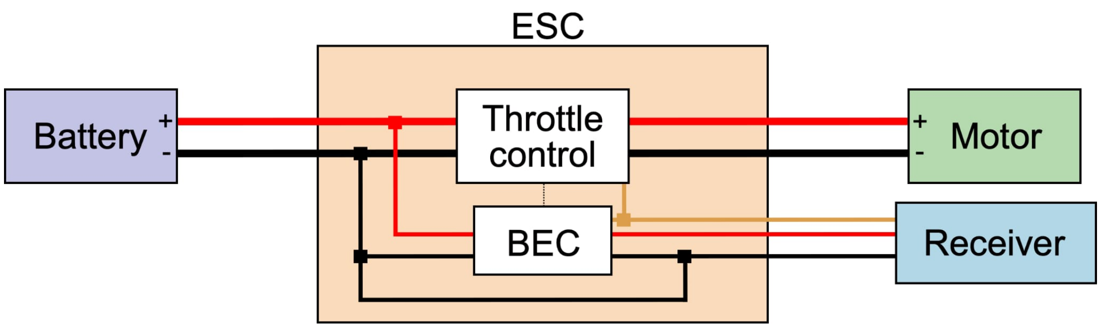


Figure 5.1: Electronic scheme of Anubi's propulsion system

5.3 Experimental set-up

These tests were conducted on the ground with Anubi switched on, by giving it a series of commands as if it had to perform manoeuvres in the air, from take-off to landing in order to impose a certain thermal cycle on the battery and ESC similar to that of a flight. The fibre *1552* was stuck on the battery with a piece of paper adhesive tape, while the fibre *1530* with a Kapton seal:

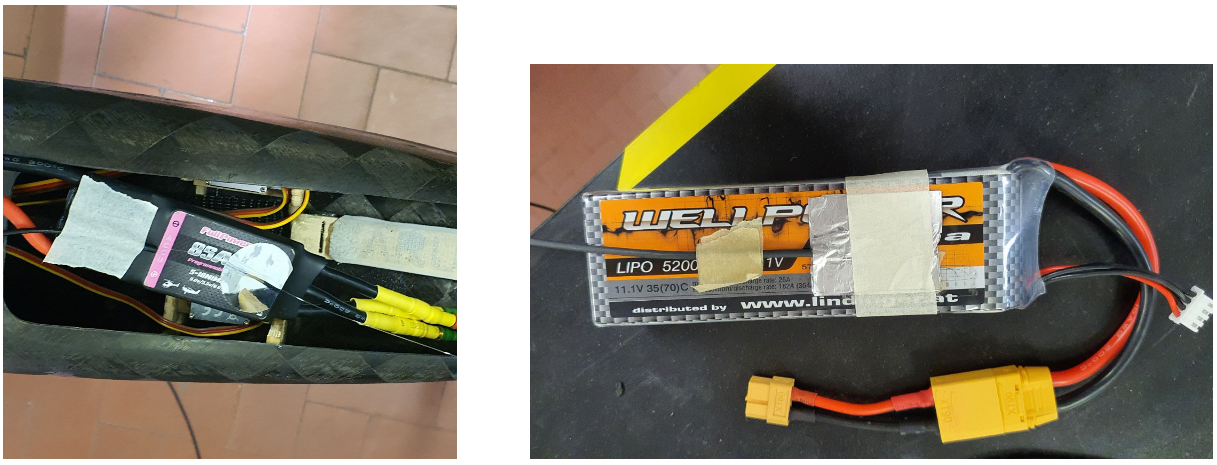


Figure 5.2: ESC and battery set-up

After placing the ESC and battery on top of each other at the front of the fuselage with their respective sensors, both fibres were connected as usual to the interrogator, the latter wired to the PC with an Ethernet cable for visualisation and data acquisition via the software SmartSoft and setting an acquisition frequency of 2 Hz for the first two tests and 25 Hz for the third one; in particular, ESC's fibre was connected to Channel 1 (white line), while the battery's fibre to Channel 2 (red line) as displayed here:

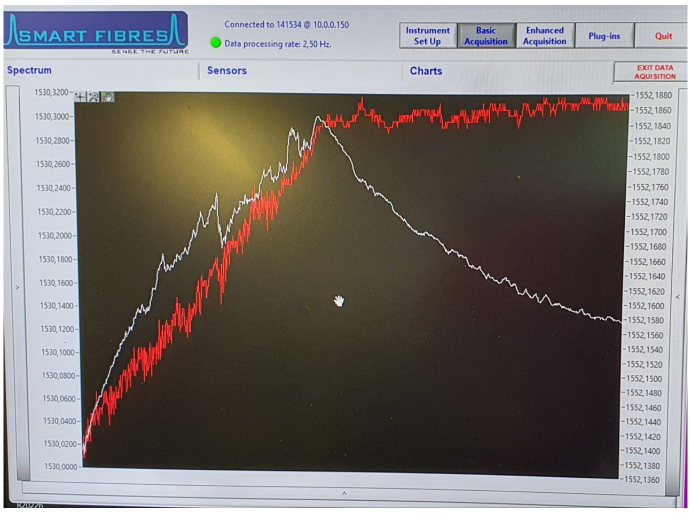
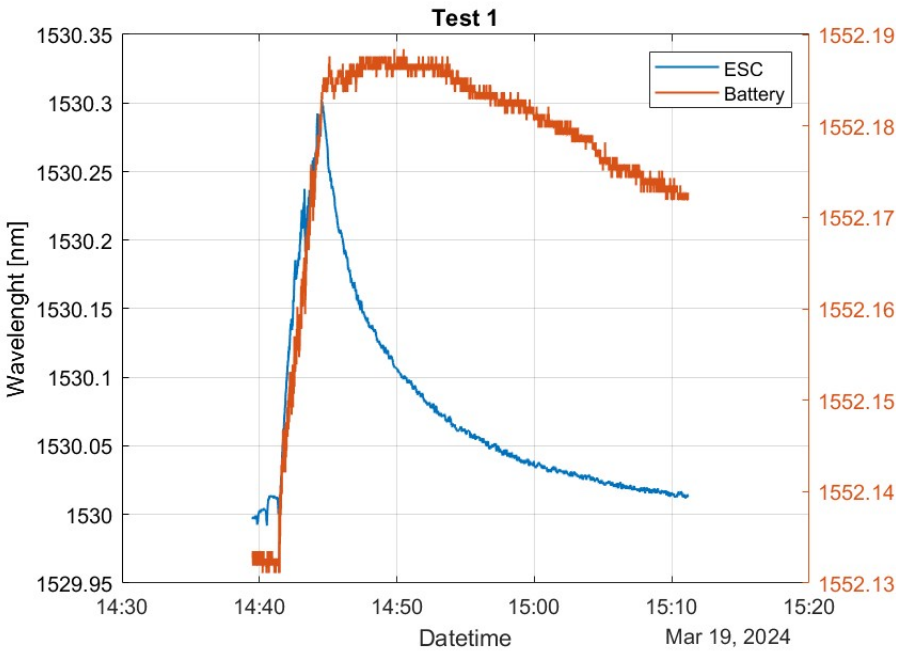


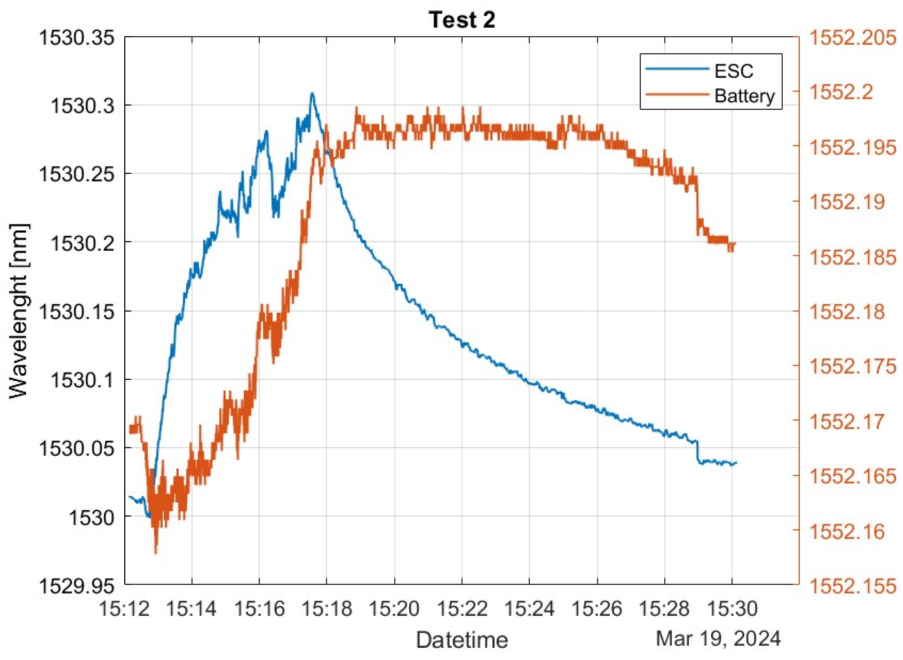
Figure 5.3: Visualization of FBG data on the software

5.4 Results and comments

The data collected were imported as usual on MatLab and later analysed with the code *ImportDati_ Temp.m*, showing both the λ trends of battery and ESC but also, through the conversion with the equation 4.1 the temperature trends to make a more detailed evaluation of the thermal response.

We now proceed by showing the graphs obtained of the previously mentioned quantities, followed by comments on the results. The first 3 graphs concern wavelength trends over time during the execution of dummy manoeuvres while the drone was fixed to the ground:





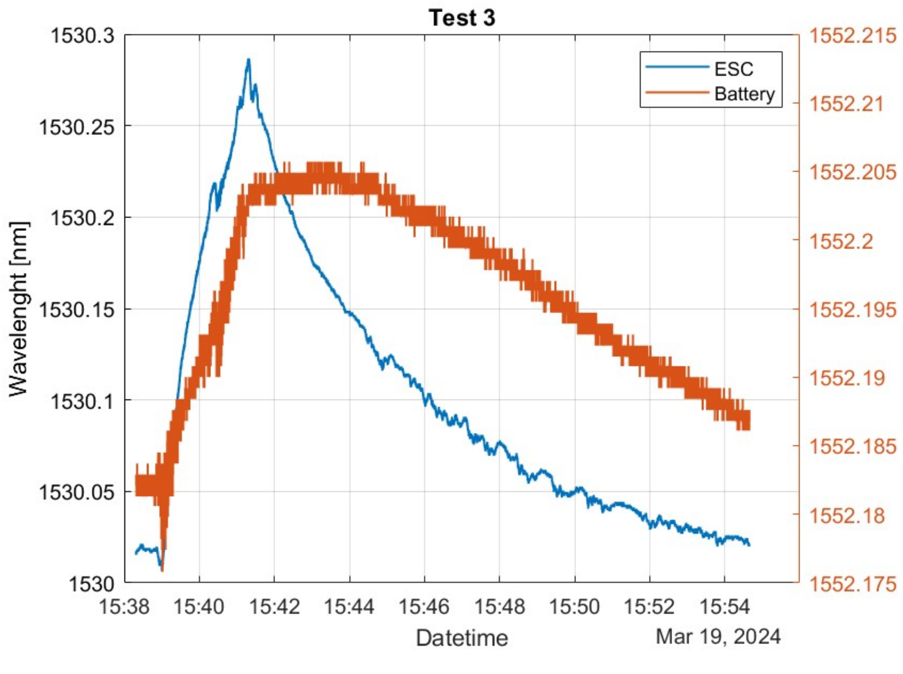
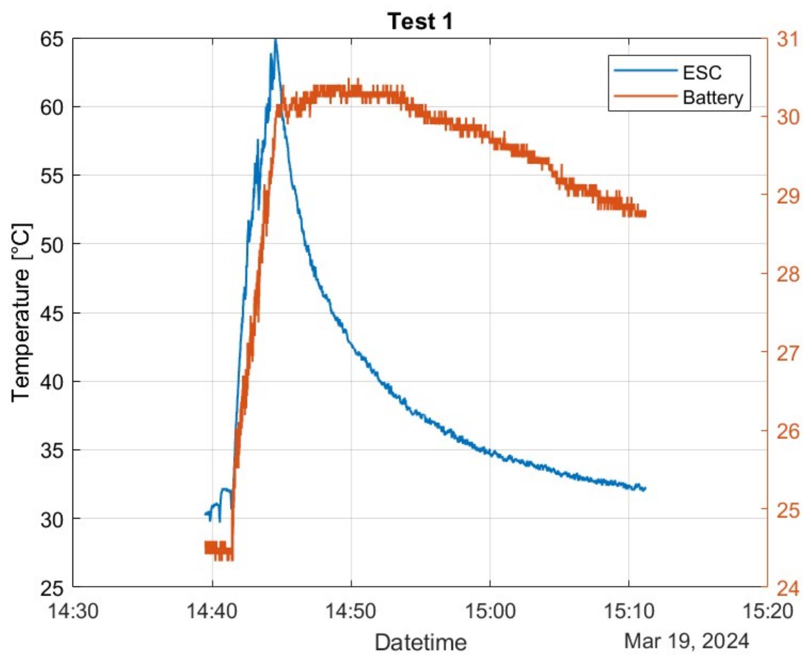
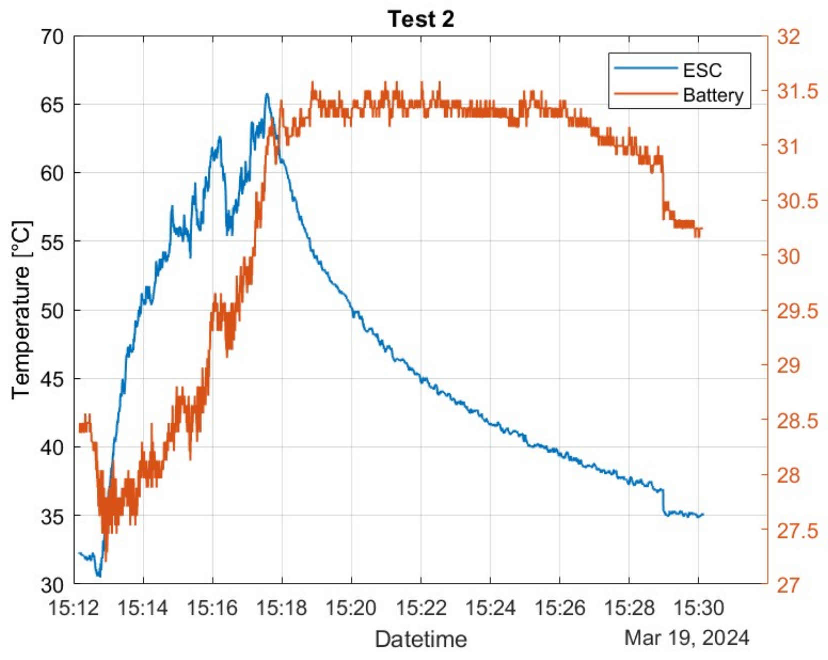


Figure 5.4: Temperature trends

What can already be seen is an exponential evolution of λ trends for both devices, typical of a thermal system as already observed in the calibration of FBG thermal sensors, thus already constituting a first indication that the thermal response may be satisfactory. Indeed, in the early stages concerning take-off where there is an increase in the power required by the electric motor so both the ESC and the battery begin to heat up and consequently, the wavelength also has to increase and this is what happens as shown in the three graphs. The opposite happens in the descent and landing phases, during which a decrease in power in order to reduce the speed of the drone is obviously followed by a cooling of the two components, which leads, as can be seen, to a reduction in λ. However, to get a better idea of the thermal response, the graphs with the temperature trends measured by FBG sensors are shown here:





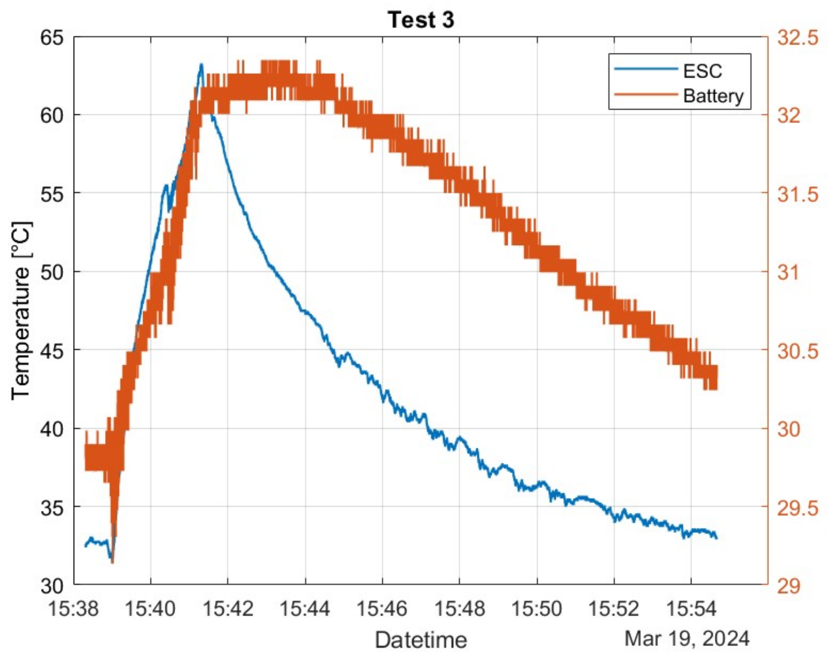


Figure 5.5: Temperature trends

First of all, the linear relationship between wavelength and temperature is clear, as the trends have not changed since the previous graphs, demonstrating once again the practicality of using these FBG sensors in monitoring this important physical quantity in system monitoring. This is proved also, as mentioned before, and again evident here, with the temperature that evolves exponentially, just like a first-order system such as a traditional temperature probe.
It can be seen that the initial temperature of the battery increased with each test, which is certainly a symptom that they were repeated within a few minutes and that the battery, therefore, did not have a chance to cool down, but also of the fact that as it discharged, it warmed up more; in addition, it is also important to note the different cooling rates between the two components, caused evidently by the different thermal inertia dependent on the material of which they are made but especially by the substantial difference in mass (366g of the battery against 57 g of the ESC), thus a greater difficulty for the battery to cool quickly, indicating a real temperature trend recorded by the sensor.
Moreover, apart from small oscillations caused by sensor vibrations induced by the airflow generated by the propeller, more emphasised in the third test because data were acquired with 25 Hz, the long oscillations in the second test are noteworthy because they highlight the response speed of the two sensors, characterised by the low thermal inertia of aluminium and Kapton, as well as the optical fibre; it can therefore be said that these sensors have a good response speed, which is definitely a key aspect if it wants to carry out an appreciable diagnostic analysis of a component subject to heating and therefore potentially hazardous. However, it's important to underline also that since the ESC and battery are positioned on top of each other, mutual heat exchange between them must also be considered, which leads to greater oscillations especially in the battery's transient when the electric motor is switched off as it is in contact with the air.
Ultimately, the temperatures shown in the graphs are very similar to those that have

been reached on other occasions for both devices before these sensors were developed, as a final confirmation of the goodness of their thermal monitoring capability. Therefore, all of these are very important findings concerning the main objective of this thesis, as fibre-optic sensors have been shown to offer a thermal response characterised by valuable speed, sensitivity and accuracy, providing a solid basis for further future developments in aerospace applications.

Chapter 6

Static tests

6.1 Introduction

In these last two chapters, the focus is on structural monitoring, to demonstrate that FBG sensors can also be used to measure mechanical deformation. First of all, to see the real oscillations recorded by these sensors, it is necessary to set a requirement on the acquisition frequency to be applied to the *Raspberry Pi* connected to the interrogator, although there is a limit in flight, while on the ground this limit can be overcome using the SmartSoftSSI software. So, two tests were conducted to achieve this purpose: in the first test, some weights were applied on the right-half wing where FBGs are implemented, to simulate a vertical linear acceleration while, in the second one, we just moved the drone over a rung to test also the deformation of the landing gear. As mentioned before, the idea was to find the right acquisition frequency of the interrogator for further flight by seeing the main oscillation frequencies of FBG sensors, whose values were evaluated with a frequency analysis built on *Fast Fourier Transform*, so that a satisfactory correlation can be obtained between IMU (the Inertial Measurement Unit) and FBG's, claiming that these sensors can also measure mechanical deformation of UAV's main structures as the wing and the landing gear.

6.2 Test 1

6.2.1 Experimental set-up

The experimental setup involved these elements:

- UAV Anubi;
- Interrogator;
- 12 FBG sensors;
- IMU;
- Battery;

- Notebook with *SmartSoftSSI* software;

First of all, IMU unit was connected to the battery to be supplied. This unit is composed of GPS (in this case not used), an altitude sensor and a gyroscopic sensor, so it's possible to obtain data about the linear accelerations. Then, the interrogator was placed inside the fuselage, supplied by another dedicated battery, while the FBG sensors were connected to it. As reported before, the interrogator is employed to obtain data from FBG sensors, with a dedicated communication channel for each sensor. It operates sending a laser beam to the sensor and detecting the reflected wavelength and sending these results as the FBG's output to the notebook and managed with SmartSoftSSI. 12 of these FBG sensors are employed to measure accelerations along the z-axis and one for the temperature and they are installed along the half-right wing.

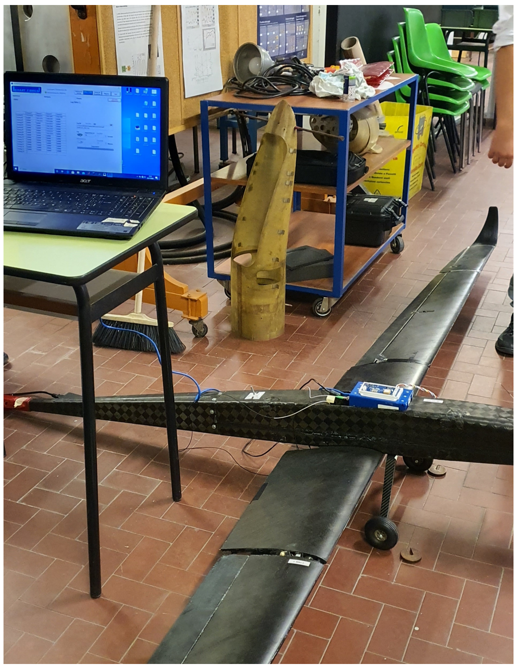


Figure 6.1: Experimental set-up

6.2.2 Data acquisition

For this test, the acquisition frequency was increased to 2500 Hz to get a higher number of data, while IMU acquired with 200 Hz. In this case, the acquisition lasted 132 seconds and throughout it, the half-right wing was loaded with some weights to simulate an aerodynamic load on it so that the upper face was in traction and the lower one was compressed, while the FBG sensors were in charge of recording the deformation of the half-wing. With SmartSoftSSI software, it was possible to manage data arriving from optical fibres containing the values of the reflected wavelengths for each FBG. Due to this frequency acquisition, 330520 data for each FBG sensor were collected in 132 seconds, while 36816 were the data about IMU component, so in this case, results from IMU were sub-league in comparison with FBG's. Further on, all data were bundled in files *.txt* type and then imported into Matlab to generate codes to process these data and evaluate them with plots.

6.2.3 Post-processing

Talking about Matlab codes, *Teststatico_1.m* was used to import FBG data in Matlab, further on grouped in a matrix called *Anubi1*, with temporal instants of acquisition and all wavelengths values of the FBG sensors, while with *ElaborazioneDati_IMU.m* IMU data were imported, showing in *structs* format data about altitude, GPS, time or linear and gyro accelerations for example and with *AirTelemetry.m* function these data have been processed to obtain results about Anubi telemetry that were later plotted with *Parametri_IMU.m*. *Teststatico_1.m* code was used once again to generate plots about the trends of wavelength values as functions of time, listed from first to 12th and to import them on a Simulink model named *Import_dati.slx* with the matrix *Signal*, while in the scope the signal is shown in the next figure with a yellow line (sensor 12 in this case):

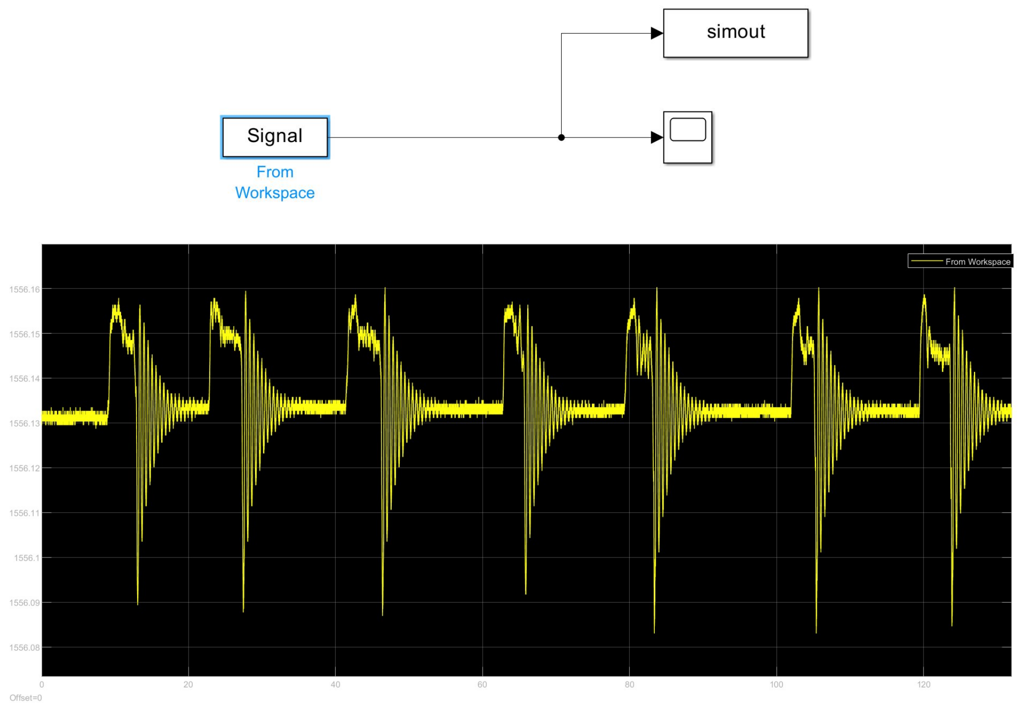


Figure 6.2: Simulink model Import_dati.slx and scope result

This step was necessary to elaborate a *Structure with time* variable in output to evaluate it in the frequency domain using the *FFT Analyzer* tool for frequency analyses (opened by typing in the Command Window "powerFFT"), useful to determine FBG signal's frequencies which is composed of. This analysis is required to become aware of which main frequencies typify the FBG response and compare it with the IMU response so that it was feasible to assure which sampling frequency is better to fix.
For this analysis, every transient of each signal was sampled, as shown in figure 6.3 just to refer to an example:

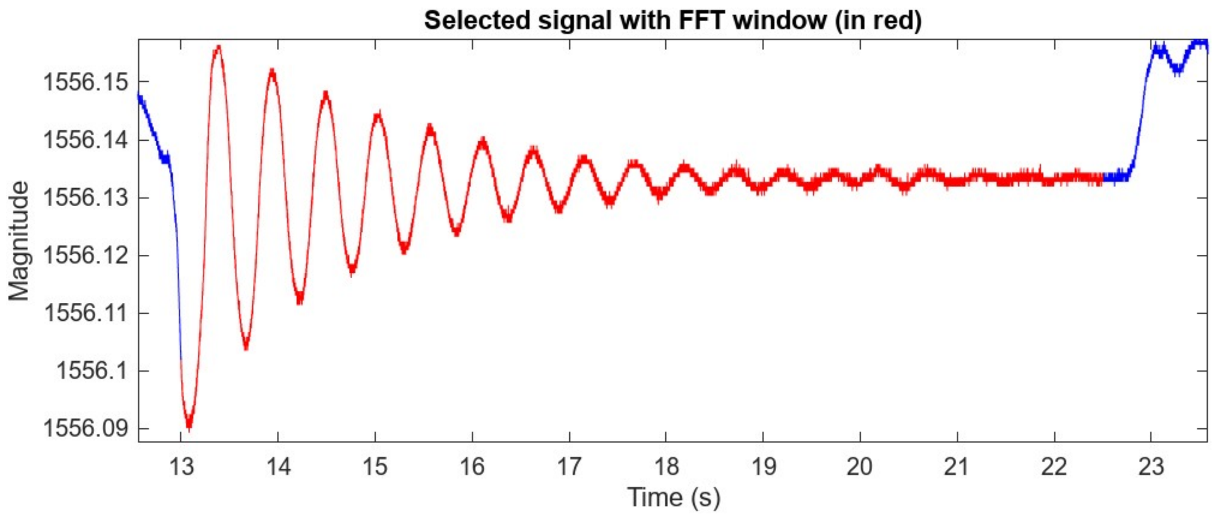


Figure 6.3: Transient sampled of sensor 12

It's something that reminds a 2nd order system, an MCK system for example where in this case there are 2 peaks for a second, so it can be expected a value of almost 2 Hz as the main frequency of this transient but this result is shown further on.

6.2.4 Results and comments

In this section results are shown as listed now:

- IMU telemetry;
- FBG sensors' wavelengths in the time domain;
- Frequency analysis of every transient for each FBG sensor

First of all, IMU telemetry is reported with a list of physical parameters such as temperature and pressure but in this case, the focus is just on linear accelerations because, as told in the previous chapter, the purpose is to obtain a linear correlation between wavelengths and deformation ϵ_z that is linked to linear acceleration on that axis. So now there are the results deriving from IMU about linear accelerations, where it has to be mentioned that the temporal axis has been translated toward the left to match the final times of IMU and FBG's data since IMU was started almost 30 seconds early:

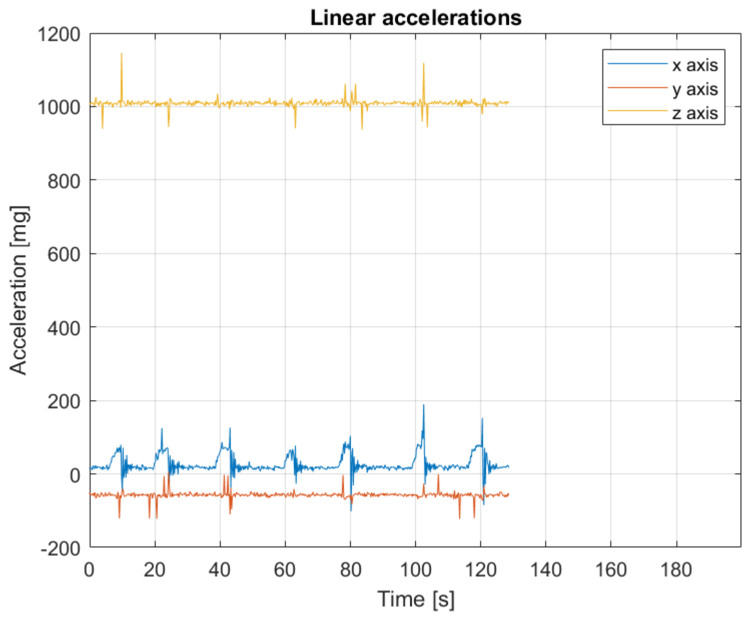


Figure 6.4: Linear accelerations recorded by IMU

It seems that the real stress, or the oscillations barely, took place along *x axis* and not along the z-axis but this is due to the different position of IMU compared to the positions of FBG sensors; indeed, as shown in figure 6.1, IMU is placed above the interrogator whereas fibres are installed inside the right-half wing. This setup causes IMU to have a variance compared to the x-axis so that when the load is deployed, IMU has a roll around that axis and this is why oscillations of blue trend are more evident than those of yellow trend. However, since loads were applied along the z-axis, y and x accelerations are approximately zero in steady state, while z acceleration is around *1000 mg* because of *g* acceleration, concluding that results from IMU are logical.
Considering now trends in FBG sensors, the moments when loads were applied are evident, with transients that remind 2nd order systems under-dumped, similar between them for each sensor.

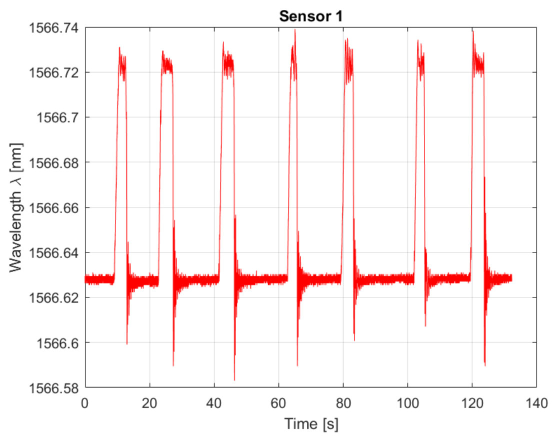


Figure 6.5: Trend about sensor 1 in static test 1

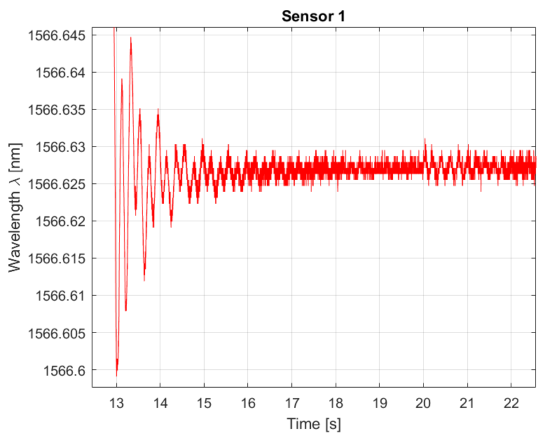


Figure 6.6: Focus on the first transient of sensor 1

Indeed, regarding frequency analysis, every transient of a sensor has peaks at frequencies that are similar but also between sensors, proving coherence in acquiring data from the sensors. These transients regard the moment just next weights are applied. So, taking just a transient of this sensor (sensor 1 in this case), peaks at about 2 Hz and 5 Hz are evident so in the first instance it's logical to think that the interrogator has to sample data in flight at least 10 Hz to see oscillations at those frequencies, as predicted by *Nyquist-Shannon* law.

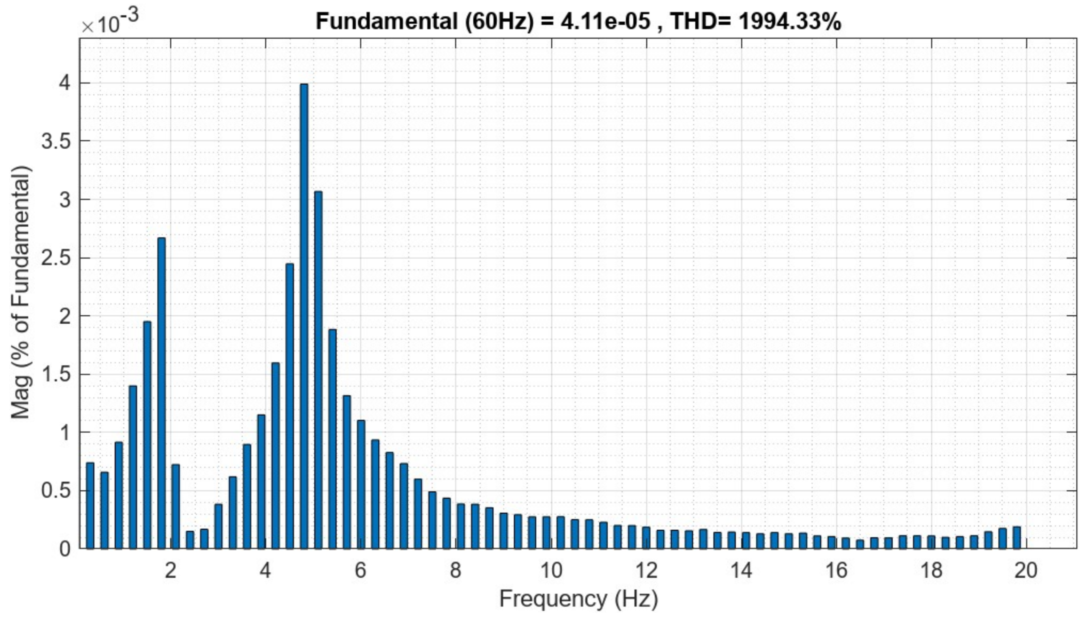


Figure 6.7: Frequency analysis of the first transient of sensor 1

This result was obtained for the whole sensors and this means that, since oscillations caused by a certain load along the z-axis on the right-half wing have normal modes at 2 Hz and 5 Hz, so in flight, the acquisition frequency of *Raspberry Pi* connected to the interrogator has to be at least 10 Hz.

6.3 Test 2

6.3.1 Experimental set-up and post-processing

The whole experimental setup it's the same as the previous one except for the addition of two disks used as steps for the landing gear shown in figure 6.1, so all the connections are the same for both FBG sensors and IMU unit.
In this case, the acquisition lasted almost 455 seconds with a frequency of 2500 Hz set by SmartFibres software for FBG sensors and 200 Hz for IMU so that data collected by the first ones are 1138313 and 90817 coming from the inertial platform. As before, data are grouped in files *.txt* and then imported in Matlab.
Furthermore, the post-processing for this test is similar to the first one, except for the importing of FBG data that this time was made with *Teststatico_2.m* code used also for plotting those data, while *ElaborazioneDati_IMU.m* and *Parametri_IMU.m* were

employed for importing and plotting IMU data again. The same Simulink model was also used and data in output were analysed on FFT Analyzer to find the frequencies of the oscillation modes, sampling all transients of each sensor.

6.3.2 Results and comments

Regarding the graphic in figure 6.8, since IMU is placed between the two steps, it's clear that accelerations along the z-axis are more evident compared to the previous case; indeed, the 'load' applied in this test was just a movement up and down from the steps, so IMU detects mainly vertical acceleration that is visible on yellow trends while the other trends, about x and y axis, are almost steady around initial value.

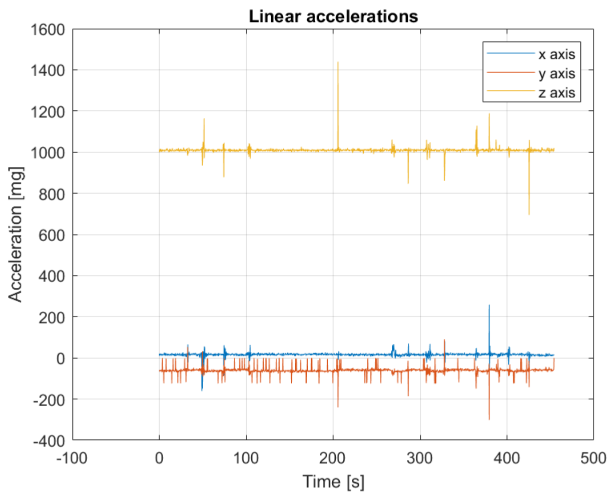


Figure 6.8: Linear accelerations recorded by IMU in test 2

Looking at FBG sensors trends, now in particular sensor 1, the moments when Anubi was dropped down are evident seeing the transient shown in the figure zoomed below:

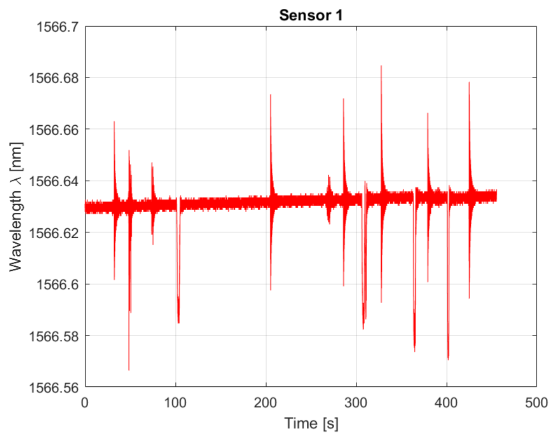


Figure 6.9: Trend about sensor 1 in static test 2

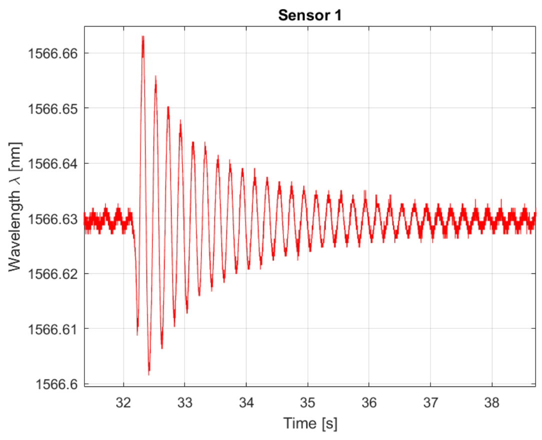


Figure 6.10: Focus on the first transient of sensor 1

this transient, like the next one with similar magnitude, responds like a 2nd system that is explainable by the fact that tyres act like springs in an MCK system, where these transients have one oscillation mode at 5 Hz, the same value that characterizes also transients of the other sensors.

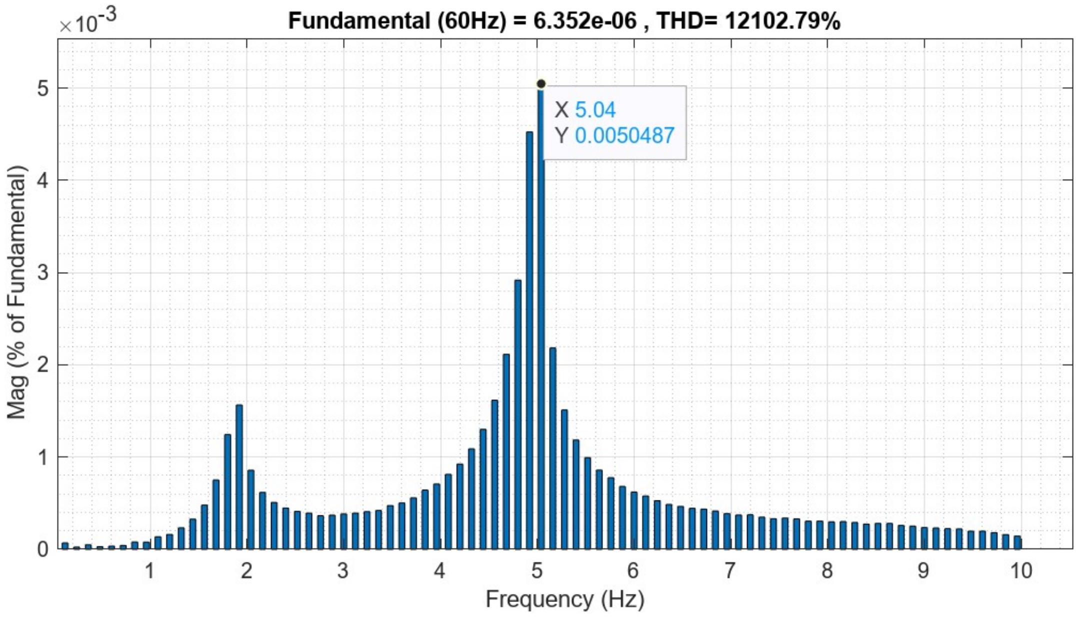


Figure 6.11: Frequency analysis of transient about step down from disks

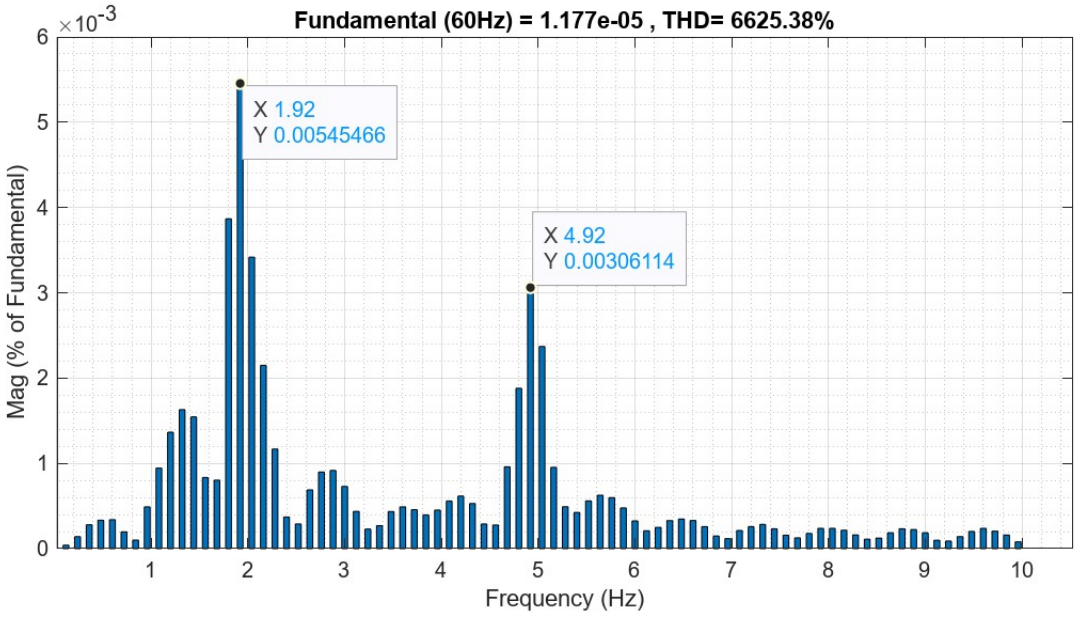


Figure 6.12: Frequency analysis of transient about step up from disks

This result supports the previous conclusion that the acquisition frequency of the Raspberry Pi would have been raised to 10 Hz in flight. This counts also for the transients with lower magnitude (when Anubi stepped up the disks) that have oscillation modes around 1.5 Hz and 5 Hz.

Chapter 7

Structural monitoring

7.1 Introduction

To complete the analysis of FBG sensors in terms of system monitoring, we now turn to the evaluation of the response following mechanical stresses in a flight test, trying to understand whether the λ_B wavelength undergoes any kind of change, thus if these sensors provide a response. To do this, a flight test took place to acquire the first set of data with FBG sensors installed on the UAV Anubi, comparing them with IMU ones concerning the linear accelerations, in particular along **z axis**, to find a linear correlation between these two physical quantities. This is done because, as above-mentioned, the wavelength λ is related to the linear deformation ϵ of the half-wing that is in turn linked to the acceleration along the z-axis, as reported in the initial equation 2.9.

FBG sensors were affixed to the four-fibre structure: two on the wing, between the side rail and the lining, placed one on the back and the other on the belly and have in total 11 sensors FBG useful to evaluate the aerodynamic loads on the wing (3 sensors in series on the belly and eight sensors in series on the back); a fibre on the back of the fuselage which, as we shall see, acts as an environmental temperature detector and one last fibre on the landing gear to evaluate its behaviour.

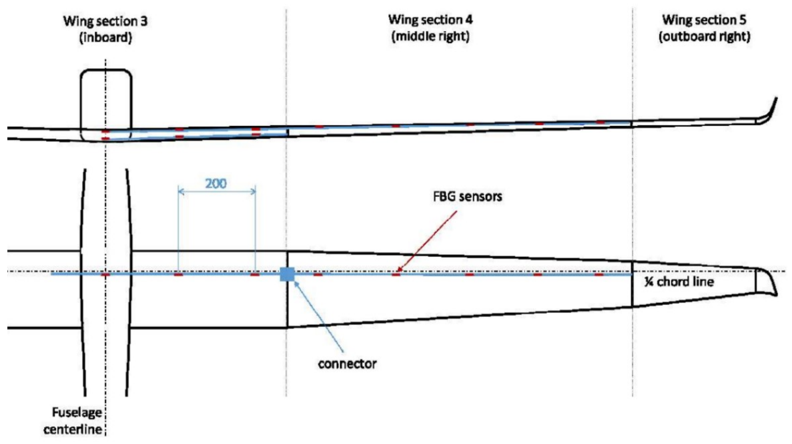


Figure 7.1: FBG sensors placement on Anubi

All the equipment is shown and classified here below:

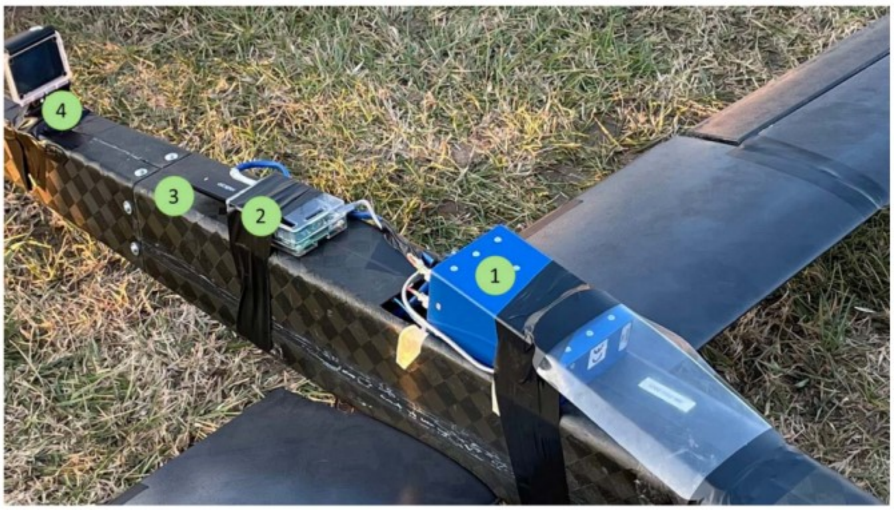


Figure 7.2: Distribution of equipment on Anubi

Component	**Mass [g]**
SmartScan FBG interrogator (1)	1100
Raspberry Pi^{TM} (2) Model B+	45
Interrogator battery	419
USB power bank	360
FBG sensors	<1
Wiring and connections	30
TOTAL	**1955**

Table 7.1: Total equipment list and weight on Anubi

The data obtained in this flight were imported and elaborated on Matlab and from these, a series of physical quantities were valued, such as temperature, pressure, gyroscopic parameters, linear accelerations and more.
For this thesis, only linear acceleration along the z-axis was considered, both from IMU and FBG's, because this is the one mainly related to strain of the half-wing, so as if it is subjected only to bending even though generally this structure is affected by a bending-twisting that is neglected in this study. After all, the purpose is just to demonstrate that with FBG sensors it's possible to measure an accurate local deformation with the acceleration information which the optical fibre is affected of.
To obtain a reasonable result, it was necessary to adopt a *low pass* filter on IMU results to compare in a better way data from IMU and FBG's to calculate the *Pearson correlation coefficient (R)* that is s the most common way of measuring a linear correlation that measures the strength and direction of the relationship between two variables, to demonstrate that what it's measured with a normal sensor, can be done with FBG sensors too; in other words, if for example, the temperature sensor measures 30 °C in a certain point, that value has to be detected by the optical fibre so that it's rational to assume that

FBG sensors can be employed in redundancy with electrical sensors. But this R-factor is related to the number of data coming from IMU and FBG sensors, the latter in turn depending on transmitting frequency from Anubi in flight to the database, so in this part, it was also evaluated if this frequency applied for the flight was reasonable or not.

At the same time, there are also data acquired with IMU unit, GPS, altitude sensor and power sensor all implemented in a motherboard, saved in file *.txt*, containing information about speed, angular rate, linear accelerations, battery voltage, altitude but above all, data coming from the temperature sensor will be analysed.

7.2 Data post-processing

As said before, data about acceleration along the z-axis will be compared with FBG sensors ones. The latter are imported in Matlab with some codes realised now listed:

- *ElaborazioneDati_IMU.m* to import data from the motherboard (GPS, ALT, IMU and PWR);
- *Air_Telemetry.m* to elaborate the previous data mentioned;
- *ParametriVolo.m* to plot IMU data;
- *ElaborazioneDati_FBG.m* to import, elaborate and plot FBG sensors data;
- *ImportDataSL.m* to import acceleration data get with IMU in Simulink, to compare IMU and FBG data and to create data vectors for the Curve Fitter tool.

7.3 Results, comments and conclusions about the structural monitoring

FBG data The first result to show is about the trends of wavelengths read by FBG sensors that regard accelerations along the z-axis for the two flight tests, displayed here, normalized concerning wavelength of sensor 1:

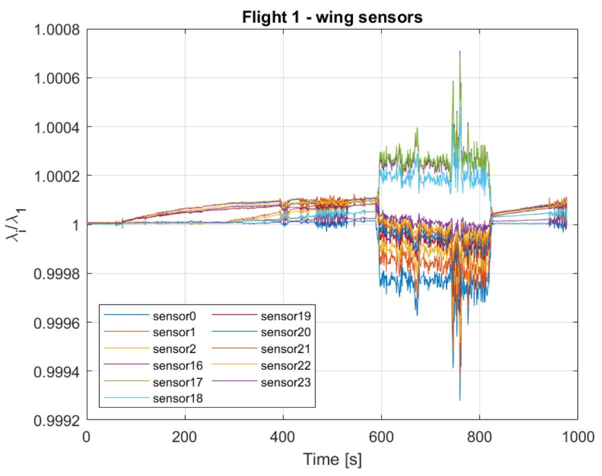


Figure 7.3: Wing FBG data in flight test 1

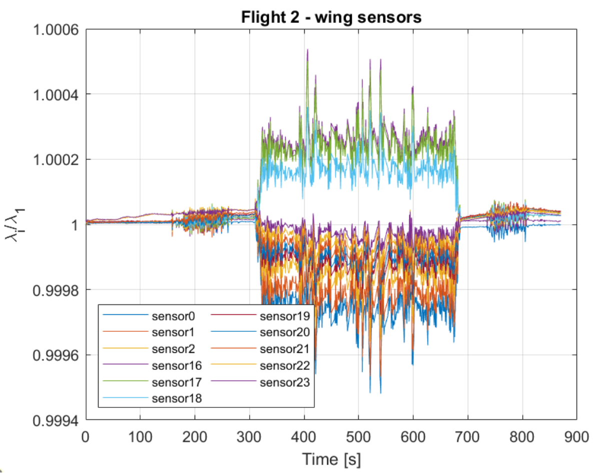


Figure 7.4: Wing FBG data in flight test 2

It's possible to divide the trends into four main phases spotted by oscillations: starting fibres, transportation of Anubi to the track, manoeuvre phases and landing. In particular, from 0 to 172 seconds there is the starting fibres phase, where the rising value of wavelengths is related to the warming up of fibres since they are responsible for temperature as seen in equation 2.9, then a phase to 216 seconds in which Anubi was transferred from the canopy to the track, indeed sensors detected oscillations for this movement. It's useful to take on that all the sensors except for 16, 17 and 18 are installed on the top of the wing and this is highlighted from a ratio greater than 1 due to half wing weight that affects longitudinal balance during the standing phase. Later on, after the take-off phase, a series of manoeuvres occurred, characterized by copious oscillations of wavelengths, with a ratio, also in this case, greater than 1 for sensors 16, 17 and 18 because lift acts on it so that part is subject to traction, while the rest of ratios are minor than 1 due to compression of the wing.

IMU data Now results coming from IMU about linear accelerations are shown below:

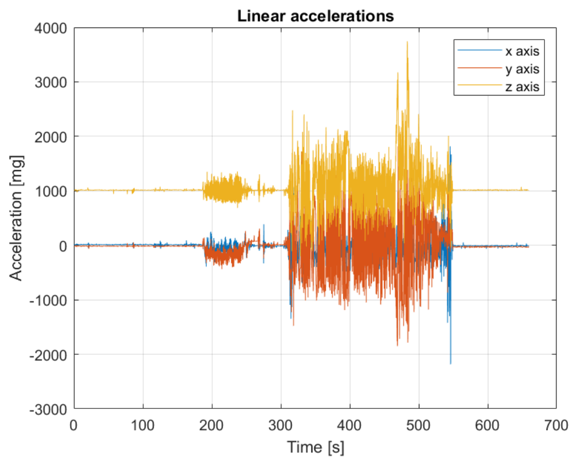


Figure 7.5: Linear accelerations in flight test 1

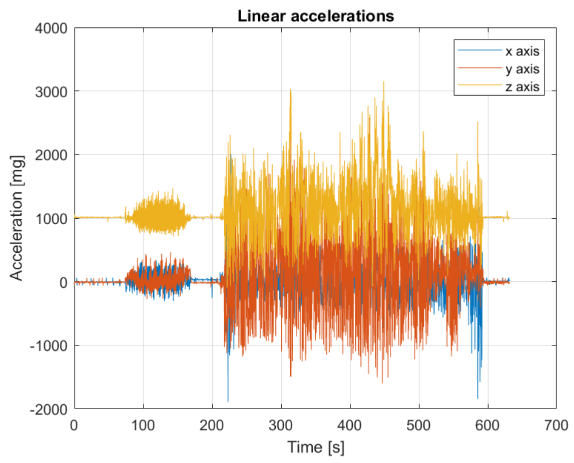


Figure 7.6: Linear accelerations in flight test 2

For both flight tests, longitudinal acceleration has a steady value of around 1000 mg (g-force) and this is due to the only weight of the half-wing, while the others are around 0 mg because there are any particular accelerations forced along those axes. Again, the phases above-mentioned are evident in these data too, indeed there are oscillations during the rolling phase and in flight when some manoeuvres are imposed.

Comparison between FBG and IMU data So, once that data from IMU and FBG's are visualised (sensor 18 was considered in this case since results are equal for the sensors placed on the bottom of the wing), they were compared in these plots to find a certain conformity between them.

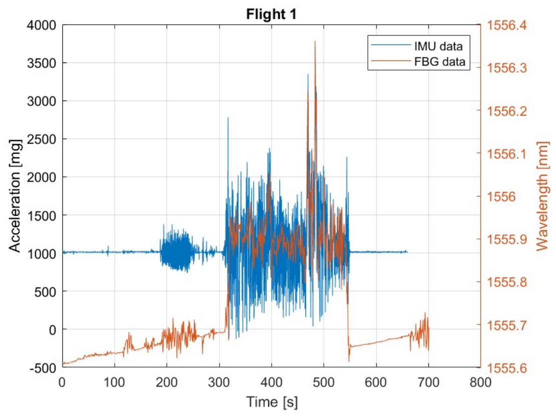


Figure 7.7: Trend of IMU and FBG data in the flight test 1

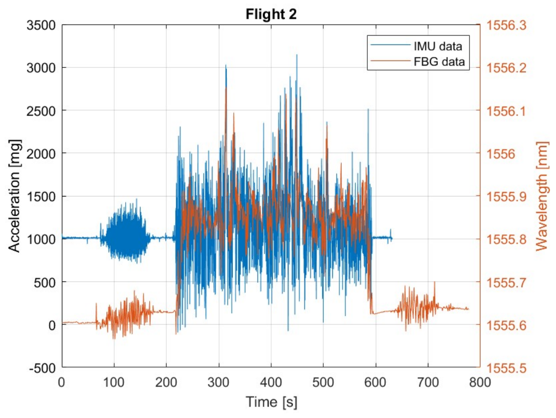


Figure 7.8: Trend of IMU and FBG data in the flight test 2

At first sight, oscillations detected by the IMU unit have a higher magnitude than FBG's ones so it was necessary to apply a low-pass filter realised with Simulink and named *Filtro_passabasso.slx* to evaluate in a better way data conformity. A low-pass filter (LPF) is a circuit that only passes signals below its cutoff frequency while attenuating all signals above it. It is the complement of a high-pass filter, which only passes signals above its cutoff frequency and attenuates all signals below it. For this filter, it was imposed the *Stopband edge frequency* at 1.52 Hz, which was the acquisition frequency in flight imposed to the SOC, so that frequencies higher than that value were cut off, while the lower ones were passed and displayed in the scope as shown below together with the Simulink scheme for the flight 1, where Acc_3 is acceleration along z-axis:

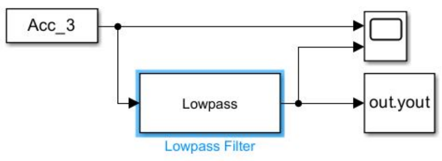


Figure 7.9: Lowpass filter Simulink model

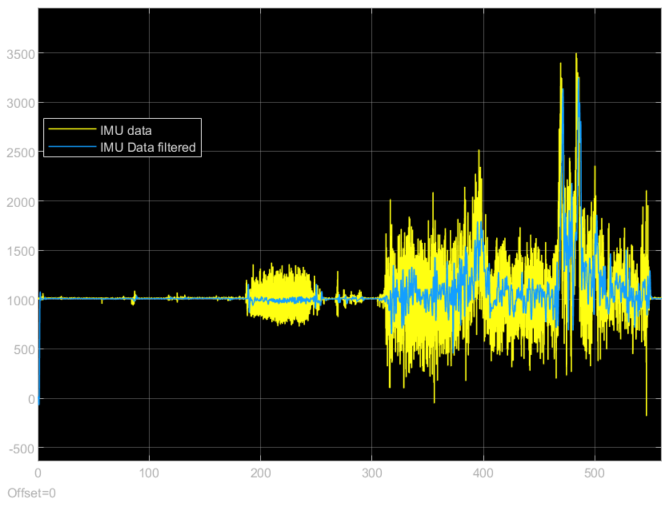


Figure 7.10: IMU signal filtered

As foreseeable, oscillations have now lower magnitudes and are more comparable with wavelengths; this is displayed in the next figures for both flights, taking for example data from sensor 18:

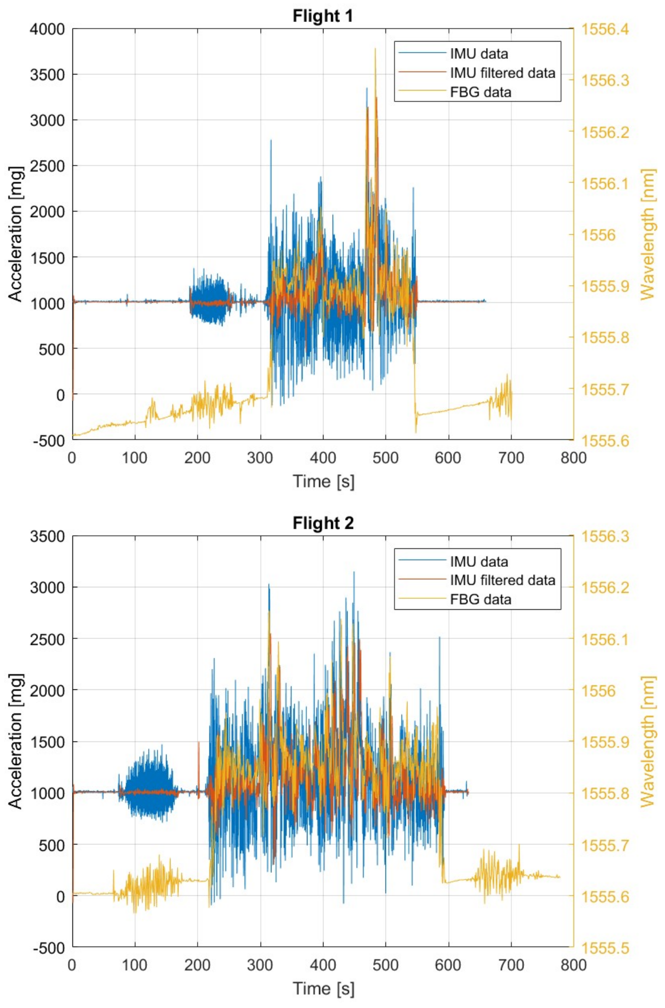


Figure 7.11: Trend of IMU, FBG and IMU filtered data in the flight tests

The first conclusion is that trends are similar, indicating a good conformity between them, so this let think that FBG sensors can somehow record data as IMU does; anyway, focusing around some oscillations, it's evident that there is a not passable disparity as shown here:

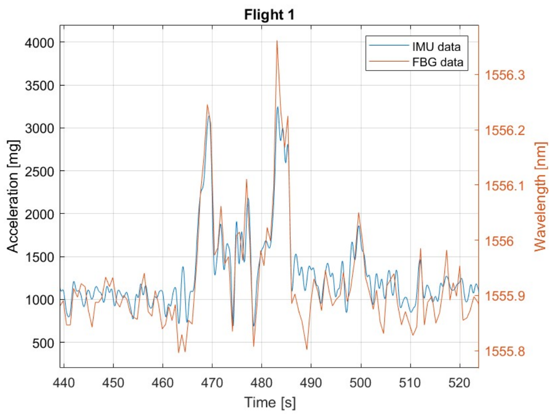


Figure 7.12: Focus on the main oscillations in the flight test 1

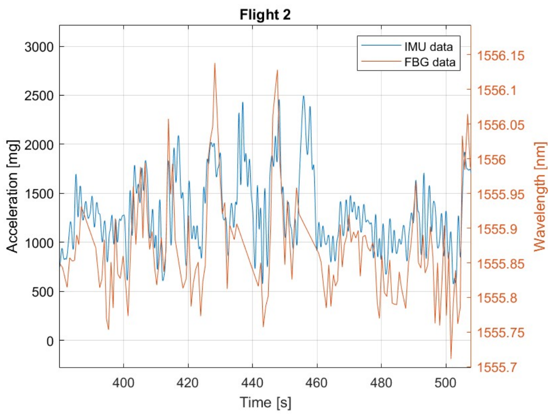


Figure 7.13: Focus on the main oscillations in the flight test 2

This is explainable for two reasons: 4G connection was lost due to low signal and the wide difference value of acquisition frequency, so it was predictable that there wouldn't have been a good linear correlation between these data. Anyway, at this point, the idea was to find this linear correlation evaluating at the same time the correlation factor R; obviously, this wasn't possible covering the whole flight so the focus was on the phase of manoeuvres. Before doing this analysis on Matlab tool *Curve Fitter*, it was necessary to increase the vector dimension of FBG data with *interp1* Matlab function, to plot them on the y-axis with IMU ones on the x-axis. After a first analysis for flight 2 considering only the manoeuvres phase, it's clear a very low R factor, that's not acceptable for aerospace standards, so other analyses were done shrinking the window of study around the two greatest picks. The result is more satisfying but it's not enough; indeed, in the aerospace sector, the R factor has to be almost *0.95* while the higher value obtained is *0.91*, which means 68 % of standard deviation. After all, this was predictable since the acquisition frequency imposed on the interrogator was too low, especially compared to the IMU one so it wasn't feasible to get data about high-frequency dynamics with FBG sensors; so this R factor values it's proof of inconsistency between IMU and FBG's in certain little phases shown in figures 7.12 and 7.13. This can be improved by going to increase the acquisition frequency to the value of 10 Hz obtained from the static tests in the previous chapter, so that the actual oscillations recorded by the FBG sensors can be appreciated and thus be comparable with those obtained from the IMU. In any case, it was still possible to obtain a mechanical response, which is an important first step in the development of these sensors, since, net of the filtering of the IMU data and a not-too-high R factor, the wavelength oscillates with a behaviour similar to that assumed by the vertical acceleration, thus making it assumed that the path is the right one.

7.4 Landing gear structural monitoring

In this part, data from the sensor placed on the undercarriage are shown here to make it structural monitoring as well, so as to broaden the scope of application on UAV systems. Just as with the wing data, the data were processed on MatLab, going to plot the wavelength trend during the flight phases, as shown here:

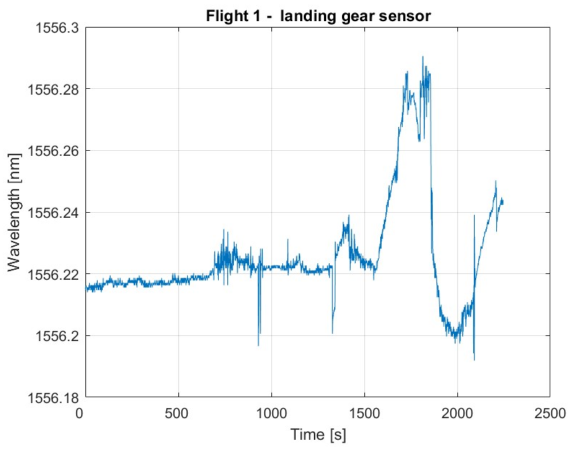


Figure 7.14: Landing gear FBG data in flight test 1

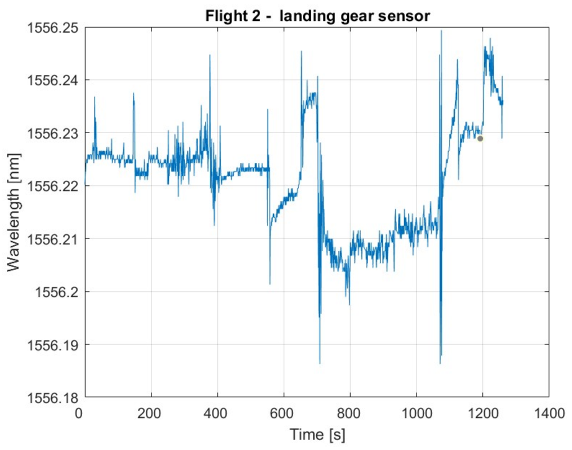


Figure 7.15: Landing gear FBG data in flight test 2

Concerning the landing gear data of flight 1, low peaks of oscillations are evident for those phases in which the drone executed a manoeuvre, around 750 s and 1300 s, while the landing phase is highlighted by a prominent peak around 1700 s which is the consequence of the soft impact at the soil when the landing gear undergoes a certain elastic deformation that implies a variation of the grating's length inside the fibre and thus of the wavelength ($\Delta\epsilon \longrightarrow \Delta\lambda$); after some seconds in which the landing gear has accumulated mechanic energy during the elastic deformation, releases it returning in few seconds in its dynamic equilibrium, denoted by a decrease of the wavelength following this back-deformation. However, in this flight but also the second one, those quasi-instantaneous but prevalent peaks are caused by numerical electronic problems of the processor connected with the interrogator which shadow some data, so the plotting process is in trouble with these missing data, elaborating those fake peaks which due not to some mechanical vibrations. So, this aspect has to lead to a better connection system which may reduce or eliminate these numerical problems that could falsify results about mechanical vibrations and so the following structural assessments.

From the results of the second flight, it can be seen that the landing was more tenuous, as the wavelength did not change much, indicating precisely that the mechanical deformation was less than in the first landing, thus concluding that these sensors are capable of offering some response to some type of mechanical stress applied to the UAV structures, in this case, the landing gear. This still provides an encouraging basis for the future development of these sensors in the context of structural monitoring of a critical system such as precisely the landing gear.

7.5 Thermal environmental monitoring: results and comments

In the introduction, an analysis of temperature data concerning environmental monitoring was foretold; so in this section thermal data coming from IMU and FBG sensors are displayed:

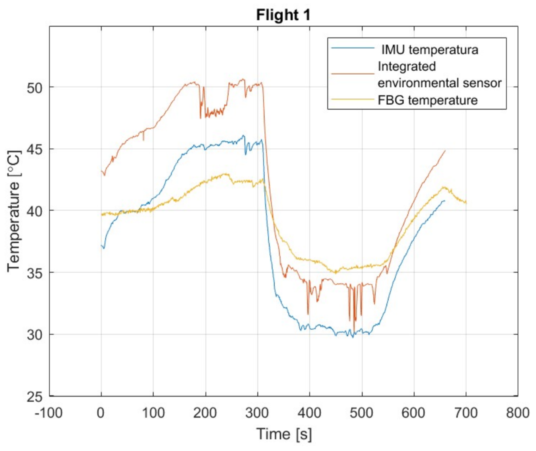


Figure 7.16: Temperature data of the flight 1

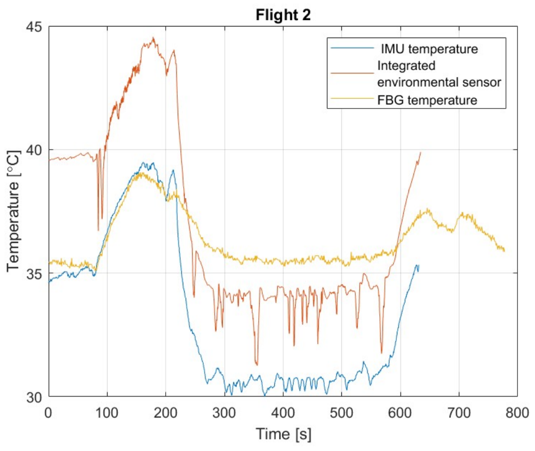


Figure 7.17: Temperature data of the flight 2

An offset of at least 5 °C can easily be inferred from these trends, presumably due to the different placement of the three sensors, although the trends are nevertheless roughly similar, except for noticeable fluctuations for the integrated environmental sensor caused by electromagnetic disturbances, which is one aspect that nonetheless highlights the insensitivity of the FBG sensor to these disturbances since the trend of the latter does not show noticeable oscillations. It is also worth saying that these are still correct temperature variations, as it increases throughout the takeoff and in-flight manoeuvring since the drone has been moved from a cool environment and exposed directly under the sun with an ambient temperature of about 25 °C, in addition to the fact that the various components heat up due to the increased power required. In contrast, the temperature begins to drop during landing and moving back to a cool environment.
Anyway, these differences between values still cannot be acceptable if safe thermal monitoring of systems is to be carried out, and that is why subsequent tests have been carried out in order to make thermal sensors capable of servicing this function with satisfactory results.

Conclusions

At the end of this experimental thesis work, it is appropriate to draw conclusions from the results obtained, recalling the aim of this work to demonstrate that these new fibre-optic sensors may be able to perform both thermal and structural monitoring of the Anubi drone's main systems, providing a starting point for subsequent research activities on FBG sensors.

In the first instance, it is appropriate to draw attention to the excellent response obtained with the experimentation of thermal sensors. In terms of thermal calibration, high values of the correlation factor R were obtained (in both cases 0.998), thus indicating a very satisfactory linear relationship, even considering that one sensor was provided with packaging as protection, while the second was not. Concerning the direct application of the propulsion system of the Anubi drone, as demonstrated in the respective graphs shown, remarkable temperature trends were obtained, that explicate the real physic behaviour of the components, thus constituting a rather reliable and sensitive thermal response of the sensors. Eventually, it can be assured that the thermal measurements were quite satisfactory, suggesting that these types of sensors can pursue further more painstaking development in the area of diagnostics of aerospace systems subjected to heating and thus, in this respect, can be used as an additional database to complement the traditional sensors currently in use for the thermal monitoring and so higher safety standards reachable.

The same cannot be said, however, for the results obtained from the analysis done for the sensors placed on Anubis' wing during the flight test, which were used to measure its mechanical deformation. As it turns out, there is certainly a response of the FBG sensors as a result of the accelerations induced by the manoeuvres carried out in flight that still leaves a point on which to imprint future work. However, it still did not leave us satisfied given the poor correlation obtained between the acceleration recorded by the IMU and the wavelengths of the FBG sensors. Nevertheless, an operational requirement regarding the frequency of acquisition by the interrogator, higher than 10 Hz, for subsequent flight tests was defined through the execution of the two static tests performed on the ground, since this turned out to be the main problem of the lack of good correlation between the two data sets. However, the results obtained are not to be dismissed out of hand, as there was a wavelength behaviour that was very similar to that recorded by the vertical accelerations, thus suggesting that there is a response to mechanical deformation by FBGs, but that it must be improved to obtain more accurate results.

Therefore, from the point of view of system structural monitoring, it can be said that there are improvements to be made to try to obtain a linear relationship between the mechanical deformation induced on the wing and the wavelength recorded by the interrogator, to

show that again, this new sensor technology can be employed in the aerospace field for diagnostics of the most critical structural elements and offer an additional amount of data to be added to those collected with the technologies already in use.
Resuming, with this important basis from which to build, the goal for the next experimental applications is first of all to carry out another flight test with the correct acquisition frequency, in order to obtain more encouraging results regarding the measurement of mechanical deformation. On thermal monitoring, the desire is definitely to carry out further tests, so that it can be verified that these sensors are able to offer high accuracy, sensitivity, reliability and repeatability, so as to find applications possibly even on aeroplanes.

Appendices

List of input files, codes and files created

Calibration of temperature sensors

- Acquisizioni_C.Climatica: climatic chamber data;
- Acquisizioni_Interrogatore: FBG sensors data;
- CC_Analysis: to import data from the climatic chamber and SmartSoftSSI software and to elaborate graphs of the temperature inside the chamber and linear regression line of the sensor;
- Lambda2temp: to convert the wavelengths recorded by the FBGs into temperature data and compare them with data obtained with the probe inside the climatic chamber;
- Retta_di_taratura_FBG_1552: Matlab data containing the coefficients of the linear regression line for the FBG 1552;
- Retta_di_taratura_FBG_1530: Matlab data containing the coefficients of the linear regression line for the FBG 1530;
- Folders *Acquisizione_*: they contains all input data and resulting graphs.

Temperature monitoring on the propulsion system

- Folder *Raw data*: FBG sensors data
- ImportDati_Temp: to import data from SmartSoftSSI and elaborate graphs of the wavelengths;
- Lambda2temp: to convert the wavelengths recorded by the FBGs into temperature data
- Folder *Grafici_ ottenuti*: it contains the graphs elaborated.

Static tests

- Folder *Dati_IMU*: IMU data;
- Folder *Dati_grezziFBG*: FBG data;
- ElaborazioneDati_IMU: to import IMU data;
- Air_Telemetry: function to put AirTelemetry data into a data structure;
- Parametri_IMU: to generate graphs about IMU data;
- Teststatico_1: to import FBG data of test 1 and generate graphs of the time trends;
- Teststatico_2: to import FBG data of test 2 and generate graphs of the time trends;
- Import_dati: Simulink model to convert FBG data into a *Structure with time* variable, necessary to use the FFT Analyzer;
- Folder *Risultati IMU*: contains the graphs showing IMU data;
- Folder *Risultati FBG*: contains the graphs showing FBG data.

Structural monitoring

- Folders *Volo1_PhotoNext* and *Volo2_PhotoNext*: IMU telemetry data and FBG data
- ElaborazioneDati_IMU: to import IMU data;
- Air_Telemetry: function to put AirTelemetry data into a data structure;
- ElaborazioneDati_FBG: to import FBG data and generate graphs of wavelength trends during the flight tests;
- ParametriVolo1: to generate graphs about IMU data and comparison with FBG data (ParametriVolo2 for flight 2)
- Filtropassabasso: Simulink model to filter IMU data;
- ImportDatiSL: to move IMU linear acceleration data to Simulink and show graphs of the filtered data compared with FBG ones;
- Folders *Risultati volo 1* and *Risultati volo 2*: they contains all the resultant graphs of the flight tests.

Matlab codes

CC_Analysis.m

```
%% DATA ACQUISITION FOR THE THERMAL CALIBRATION


%% CLIMATIC CHAMBER FILE ACQUISITION


% uploading files from folder
cartella_acquisizione=input("Inserire il numero
   identificativo della catella,\n" + ...
     "presente all'interno del percorso - C:\Users\Alessandro\
        Desktop\DOCUMENTI\SCUOLA\UNIVERSITA\Ricerca\Thales
        Alenia Space\Thales misure,\n" + ...
     "in cui sono contenuti i file (Es: 01 indica la cartella
        Acquisizione_01):","s");
fprintf('\n');
cartella_destinazione="C:\Users\User\Desktop\Test camera
   climatica 14_12\Acquisizione_"+cartella_acquisizione;

disp("C:\Users\User\Desktop\Test camera climatica\
   Acquisizione_"+cartella_acquisizione+"\Acquisizioni_C.
   Climatica")
fprintf('\n');

percorso_dati =cartella_destinazione+"\Acquisizioni_C.
   Climatica\"; %folder where the smart soft files are saved
   'directory\'

% I get the list of files in the folder
disp('ELENCO FILE .DAT')
fprintf('\n');

% Saves a list of all .log file names
file_list = dir(percorso_dati);% is a 5x1 structure that
   contains name, folder, data, bytes, isdir, datenum

% I check that the files are in order,
%in theory it should already be so
k_pc = 2;%num "ghost file" in the folder (depends on pc)
k = length(file_list)-k_pc; % num total of .DAT files found
   in the folder
```

```
for j=1:k+k_pc
    for i = j:k+k_pc
        if(file_list(j).datenum > file_list(i).datenum)
            tmp = file_list(j);
            file_list(j) = file_list(i);
            file_list(i) = tmp;
        end
    end
end
clear i j

fprintf('- Sono stati trovati e ordinati %d file .DAT.\n',k)
fprintf('\n\n');



%Building structure in which I save and merge data read from
   various files


    dati_CT_originali = struct( 'Time', [], 'T_set', [], '
       T_camera',[]);



%Union of found .DAT files that will be saved in
   original_CT_data


    for n = 1:k

     %Read settings automatically generated files from Import
         Data

        opts = delimitedTextImportOptions("NumVariables", 30)
           ;
        opts.DataLines = [2, Inf];
        opts.Delimiter = "\t";
        opts.VariableNames = ["MS_DATE_TIME", "Tset", "Tramp
           ", "Tk", "Tsl", "RhSet", "Rhc", "Rh", "Ztv", "Tv",
            "Pkmp", "Pgk", "Phk", "Pgv", "Phv", "V", "CN", "
           RUN", "K1", "Y1", "Y2", "Y3", "Y4", "LV_C", "LV_H
           ", "LV_L", "Wf", "Cp", "AL", "VarName30"];
        opts.VariableTypes = ["double", "double", "double", "
           double", "double", "double", "double", "double", "
```

```
      double", "double", "double", "double", "double", "
      double", "double", "double", "double", "double", "
      double", "double", "double", "double", "double", "
      double", "double", "double", "double", "double", "
      double", "string"];
   opts = setvaropts(opts, 30, "WhitespaceRule", "
      preserve");
   opts = setvaropts(opts, 30, "EmptyFieldRule", "auto")
      ;
   opts.ExtraColumnsRule = "ignore";
   opts.EmptyLineRule = "read";

%Reading the file
   nome_file = percorso_dati+file_list(n).name;
   dati_camera = readtable(nome_file, opts);

   Time = posixtime(datetime(datetime(dati_camera.
      MS_DATE_TIME,'ConvertFrom','excel'),'InputFormat',
      'yyyy-MM-dd HH:mm:ss.SSSS'));%saves the
      acquisition instants column by converting it to
      epoch from data Exel format
   T_camera = dati_camera.Tk;
   T_set = dati_camera.Tset;



  %I gradually merge the data from the current file with
      the data from the files already read
  % dati_CT_originali(n).Data = datetime(file_list(n).
     datenum,'ConvertFrom','datenum'); %Data creazione
     file
   dati_CT_originali(1).Time=cat(1,dati_CT_originali(1).
      Time,Time);
   dati_CT_originali(1).T_camera=cat(1,dati_CT_originali
      (1).T_camera,T_camera);
   dati_CT_originali(1).T_set=cat(1,dati_CT_originali(1)
      .T_set,T_set);
  % end

end


%Initial data request
```

```
ok=0;
while ok==0
    disp("INSERIRE DATI RELATIVI AI CICLI CONTENUTI
       NELLA CARTELLA: Acquisizione_"+
       cartella_acquisizione);
    fprintf('\n');

    T_max = input("inserisci la temperatura MASSIMA
       impostata per il ciclo, espressa in celsius:
       ");
    T_min = input("inserisci la temperatura MINIMA
       impostata per il ciclo, espressa in celsius:
       ");
    T_step=input("inserisci il SALTO DI TEMPERATURA
       PROGRESSIVO impostato per il ciclo, espresso
       in celsius e in valore assoluto: "); %salto
       progressivo di temperatura impostato nella
       camera termica
    cd= input("definisci il tipo di ciclo\n(1=
       crescente, 2=decrescente, 3=crescente-
       decrescente, 4=decrescente-crescente):");

    %initial data check

    tot_step=abs(T_max-T_min)/T_step;
    if ne(floor(tot_step)*T_step,abs(T_max-T_min)) ||
        T_max < T_min || cd<1 || cd>4 || ne(floor(cd)
       -cd,0)
        disp('ATTENZIONE: DATI INSERITI NON VALIDI')
    else
        ok=1;
    end

end


if cd==1
    T_start=T_min;
    T_stop=T_max;
end
if cd==2
    T_start=T_max;
    T_stop=T_min;
    T_step=-T_step;
end
```

```
if cd==3
    T_start=T_min;
    T_stop=T_min;
    tot_step=tot_step*2;
end
if cd==4
    T_start=T_max;
    T_stop=T_max;
    tot_step=tot_step*2;
    T_step=-T_step;
end


 tot_step = tot_step+1; %es: (50-20)/10=3  => step
    (20,30,40,50)

%dati_camera = struct('Time',Time,'T_camera',T_camera
   , 'T_set', T_set);

%CYCLE THAT SELECTS ONLY THE DATA FROM THE INSTANTS
   WHEN THE
%STATIONARITY WITHIN THE ERROR BAND CHOSEN IN THE
   FIRST
%USEFUL INTERVAL PRECEDING A TRANSIENT.
%THE DATA SO SELECTED ARE USED TO CALIBRATE THE FBG(
   ELIMINATES THE INFORMATION CONTNEUTED IN THE
   TRANSIENTS) SENSORS.

%Initialization of counters and flags
ciclo = 1;  %control variable to keep track of which
   loop we are consoidering during iterations
c_step = 1;%step counter for each cycle
find_StartTime = 0;%flag active when the stente
   preceding a T_set change is found
start_ciclo=0;%flag active when the start of a cycle
   has been detected
start_step=0;%flag active when within a cilco we are
   in a stationary step
inversione=0;

flag_errore=0;
while flag_errore==0
    errore_ammesso=input("Inserire il valore della
       soglia d'errore ammessa:");
    if errore_ammesso>=0 && flag_errore<=1
```

```
        flag_errore=1;
    else
        fprintf("VALORE IMMESSO NON VALIDO\n");
    end
end

dati_CT_ciclo = struct('Name',{},'StartTime',{},'
   EndTime',{},'Time',[],'T_set',[],'T_camera',[],'
   t_Start_Step',[],'t_End_Step',[], 'Mean_T_Step',
   []);



%Part that identifies and saves the start and end
   instants of each
%cycle and of each step per cycle.

for i = 1:(length(dati_CT_originali(1).Time)-1)

     %I find and save: the BEGINNING OF THE CYCLE (
        coincident with the beginning of the
     %first step)and the end of the first step

    if ((cd==3 && dati_CT_originali(1).T_set(i)==
       T_max) || (cd==4 && dati_CT_originali(1).T_set
       (i)==T_min)) && inversione==0
        T_step=-T_step;
        inversione=1;
    end




    if start_ciclo==0 && dati_CT_originali(1).T_set(i
       )==T_start && dati_CT_originali(1).T_set(i) +
       T_step == dati_CT_originali(1).T_set(i+1)%is
       entered T_set(i)+T_step==T_set(i+1) and not
       another comparison operator because there are
       cases where it makes different amplitude jumps


        j=i;
        while find_StartTime==0 && j>=1
            if  (dati_CT_originali(1).T_camera(j) <=
               (T_start + errore_ammesso) &&
                dati_CT_originali(1).T_camera(j) >= (
                T_start - errore_ammesso)) %If T is
                within an allowable temperature range.
                    find_StartTime=1;
                    dati_CT_ciclo(ciclo).Name= "
                      Ciclo_"+num2str(ciclo);
                    dati_CT_ciclo(ciclo).t_End_Step(
                      c_step)=dati_CT_originali(1).
                      Time(j);
              end
              j=j-1;
          end
          find_StartTime=0;

          while find_StartTime==0 && j>=1

                if dati_CT_originali(1).T_camera(j) >
                  (T_start + errore_ammesso) ||
                  dati_CT_originali(1).T_camera(j) <
                  (T_start - errore_ammesso)
                   %If the measured temperature is
                     outside the
                   %band identified by the set T and
                     the
                   %sesnability of the instrument,
                     then:
                   find_StartTime=1;
                   dati_CT_ciclo(ciclo).StartTime=
                     dati_CT_originali(1).Time(j+1);
                   dati_CT_ciclo(ciclo).t_Start_Step(
                     c_step)=dati_CT_originali(1).
                     Time(j+1);
                end

                j=j-1;
          end
          start_ciclo=1;
          start_step=1;
          c_step = c_step+1;
          find_StartTime = 0;
          clear j;
      end
```

```
%I find and save: The end and beginning of each
   INTERMEDIATE STEP.

if c_step<tot_step && start_step==1 &&
   dati_CT_originali(1).T_set(i) + T_step ==
   dati_CT_originali(1).T_set(i+1) && ne(
   dati_CT_originali(1).T_set(i), T_start)
   %if the set T is not the cycle start T and
   %at the next instant varies, then:

   j=i;
   while find_StartTime==0 && j>=1
       if dati_CT_originali(1).T_camera(j) <=(
          dati_CT_originali(1).T_set(i) +
          errore_ammesso) &&  dati_CT_originali
          (1).T_camera(j) >= (dati_CT_originali
          (1).T_set(i) - errore_ammesso)

             find_StartTime=1;
             dati_CT_ciclo(ciclo).t_End_Step(
                c_step)=dati_CT_originali(1).
                Time(i);
       end
       j=j-1;
   end
   find_StartTime=0;


   while find_StartTime==0 && j>=1

          if dati_CT_originali(1).T_camera(j) >
             (dati_CT_originali(1).T_set(i) +
             errore_ammesso) ||
             dati_CT_originali(1).T_camera(j) <
             (dati_CT_originali(1).T_set(i) -
             errore_ammesso)
              %If the measured temperature is
                 outside the
              %band identified by the set T and
                 the
              %sensitivity of the instrument,
                 then:
              find_StartTime=1;
              dati_CT_ciclo(ciclo).t_Start_Step(
                 c_step)=dati_CT_originali(1).
```

```
                    Time(j+1);
            end

            j=j-1;

        end

        find_StartTime=0;
        clear j;
        c_step = c_step+1;
      end


       %I find and save end and start STEP FINAL
          CYCLE.

    if c_step>=tot_step && start_step==1 && ne(
       dati_CT_originali(1).T_set(i),
       dati_CT_originali(1).T_set(i+1)) &&
       dati_CT_originali(1).T_set(i)==T_stop


        j=i;
        while find_StartTime==0 && j>=1
            if (dati_CT_originali(1).T_camera(j) <= (
               dati_CT_originali(1).T_set(i) +
               errore_ammesso) &&  dati_CT_originali
               (1).T_camera(j) >= (dati_CT_originali
               (1).T_set(i) - errore_ammesso))

                    find_StartTime=1;
                    dati_CT_ciclo(ciclo).t_End_Step(
                       c_step)=dati_CT_originali(1).
                       Time(j);
                    dati_CT_ciclo(ciclo).EndTime=
                       dati_CT_originali(1).Time(j);
            end
            j=j-1;
        end
        find_StartTime=0;

        while find_StartTime==0 && j>=1

              if dati_CT_originali(1).T_camera(j) >
                 (dati_CT_originali(1).T_set(i) +
                  errore_ammesso) ||
                  dati_CT_originali(1).T_camera(j) <
                  (dati_CT_originali(1).T_set(i) -
                  errore_ammesso)
                   %If the measured temperature is
                      outside the
                   %band identified by the set T and
                      the
                   %sensitivity of the instrument,
                      then:
                   find_StartTime=1;
                   dati_CT_ciclo(ciclo).t_Start_Step(
                      c_step)=dati_CT_originali(1).
                      Time(j+1);
               end

               j=j-1;
        end

        find_StartTime=0;
        clear j;

        ciclo = ciclo+1;
        c_step=1;
        start_ciclo=0;
        start_step=0;
        inversione=0;
    end

end


%Part that saves to data_CT_cycle the list of data
   associated with each
%identified cycle
for i = 1:(ciclo-1)
    dati_CT_ciclo(i).Time=dati_CT_originali.Time(find
       (dati_CT_originali(1).Time==dati_CT_ciclo(i).
       StartTime):find(dati_CT_originali(1).Time==
       dati_CT_ciclo(i).EndTime));
    dati_CT_ciclo(i).T_camera=dati_CT_originali.
       T_camera(find(dati_CT_originali(1).Time==
       dati_CT_ciclo(i).StartTime):find(
       dati_CT_originali(1).Time==dati_CT_ciclo(i).
       EndTime));
```

```
            dati_CT_ciclo(i).T_set=dati_CT_originali.T_set(
              find(dati_CT_originali(1).Time==dati_CT_ciclo(
              i).StartTime):find(dati_CT_originali(1).Time==
              dati_CT_ciclo(i).EndTime));

            for c_step=1:tot_step
                dati_CT_ciclo(i).Mean_T_Step(c_step)=mean(
                  dati_CT_ciclo(i).T_camera(find(
                  dati_CT_ciclo(i).Time==dati_CT_ciclo(i).
                  t_Start_Step(c_step)):find(dati_CT_ciclo(i
                  ).Time==dati_CT_ciclo(i).t_End_Step(c_step
                  ))));

            end
        end

%PART THAT SAVES ALL DATA AND GRAPHS PRODUCED
%Normalizes the time axis with respect to the initial instant
    and sets the units of
%measurement to the hours

l=length(dati_CT_originali(1).Time);
sottraendo=ones(l,1);
graph_time=(dati_CT_originali(1).Time-sottraendo*
   dati_CT_originali(1).Time(1))/3600;


%Saving the graph generated from the input data(
   original_CT_data)
plot(graph_time,dati_CT_originali(1).T_camera, 'r', '
   LineWidth',1); % old dati_CT_originali(1).Time
hold on;
grid on;
title("Acquisizione "+cartella_acquisizione+" Camera Termica
   ");
xlabel("Tempo [h]");
ylabel("Temperatura [ C ]");
plot(graph_time,dati_CT_originali(1).T_set, 'b', 'LineWidth'
   ,1);
legend('T_camera','T_set', 'Location','Best');
ylim([min(dati_CT_originali(1).T_camera),max(
   dati_CT_originali(1).T_camera)]);
xlim([min(graph_time),max(graph_time)]);

hold off;
```

```
saveas(gcf,fullfile(cartella_destinazione,"
   Grafico_Acquisizione_"+cartella_acquisizione+"_CT_completa
   .jpg"),'jpg');


%Saving the data collected for each individual cycle analyzed
for i=1:(ciclo-1)
     l=length(dati_CT_ciclo(i).Time);
    sottraendo=ones(l,1);
    graph_time_c=(dati_CT_ciclo(i).Time-sottraendo*
       dati_CT_ciclo(i).Time(1))/3600;

    mkdir(fullfile(cartella_destinazione, "Ciclo_"+num2str(i)
       ));
    cartella_destinazione_c=fullfile(cartella_destinazione,"
       Ciclo_"+num2str(i));
    date_str=datestr(datetime(dati_CT_ciclo(i).Time(1), '
       ConvertFrom', 'posixtime'));
    date_str(3)='_';
    date_str(7)='_';
    date_str(12)='_';
    date_str(15)='_';
    date_str(18)='_';
    date_str(20)='_';
    nome_file_dati_ciclo=date_str + "
       Dati_CameraTermica_Ciclo_n " + num2str(i)+"_(E="+
       num2str(errore_ammesso)+").mat" ;
    dati_CT_ciclo_singolo=dati_CT_ciclo(i);
    save (fullfile(cartella_destinazione_c,
       nome_file_dati_ciclo) , 'dati_CT_ciclo_singolo') ;
    disp("- dati Ciclo n  "+num2str(i)+" acquisiti con
       successo");
    fprintf('\n\n\n');

     date_str(3)=' ';
    date_str(7)=' ';
    date_str(12)=' ';
    date_str(15)=' ';
    date_str(18)=' ';
    date_str(20)=' ';

    figure;
    plot(graph_time_c,dati_CT_ciclo(i).T_camera,'r', '
       LineWidth', 1);
```

```
hold on ;
plot(graph_time_c,dati_CT_ciclo(i).T_set,'b', 'LineWidth'
   , 1);
grid on;
title(date_str+"grafico Camera Termica Ciclo n  "+num2str
   (i));
text(dati_CT_ciclo(i).t_End_Step(1),T_min-10,"Errore
   ammesso="+num2str(errore_ammesso)+" C ");
xlabel('Tempo[h]');
ylabel('Temperatura[ C ]');
legend('T_camera','T_set', 'Location','Best');
ylim([min(dati_CT_ciclo(i).T_camera),max(dati_CT_ciclo(i)
   .T_camera)]);
xlim([min(graph_time_c),max(graph_time_c)]);


date_str(3)='_';
date_str(7)='_';
date_str(12)='_';
date_str(15)='_';
date_str(18)='_';
date_str(20)='_';

for c_step=1:tot_step

    x1=plot((dati_CT_ciclo(i).t_Start_Step(c_step)-
       dati_CT_ciclo(i).StartTime)/3600*ones(size(
       dati_CT_ciclo(i).T_camera)), dati_CT_ciclo(i).
       T_camera,'g', 'LineWidth',0.5);
    x2=plot((dati_CT_ciclo(i).t_End_Step(c_step)-
       dati_CT_ciclo(i).StartTime)/3600*ones(size(
       dati_CT_ciclo(i).T_camera)), dati_CT_ciclo(i).
       T_camera,'g', 'LineWidth',0.5);
    f=size(dati_CT_ciclo(i).T_camera);
    %a1=area([dati_CT_ciclo(i).t_Start_Step(c_step),
       dati_CT_ciclo(i).t_End_Step(c_step) ], [
       dati_CT_ciclo(i).T_camera(f(1)), dati_CT_ciclo(i).
       T_camera(f(1))], dati_CT_ciclo(i).T_camera(1), '
       FaceColor', 'green', 'FaceAlpha', 0.3);
    a1=area([(dati_CT_ciclo(i).t_Start_Step(c_step)-
       dati_CT_ciclo(i).StartTime)/3600,(dati_CT_ciclo(i)
       .t_End_Step(c_step)-dati_CT_ciclo(i).StartTime)
       /3600 ], [dati_CT_ciclo(i).T_camera(f(1)),
       dati_CT_ciclo(i).T_camera(f(1))], dati_CT_ciclo(i)
       .T_camera(1), 'FaceColor', 'green', 'FaceAlpha',
            0.3);
         set(x1, 'HandleVisibility', 'off');
         set(x2, 'HandleVisibility', 'off');
         set(a1, 'HandleVisibility', 'off');

     end
     nome_file_graf_ciclo=date_str+"Grafico Camera Termica
        Ciclo n  "+num2str(i)+"_(E="+num2str(errore_ammesso)
        +")";
     saveas(gcf,fullfile(cartella_destinazione_c,
        nome_file_graf_ciclo)+".pdf",'pdf');
     saveas(gcf, fullfile(cartella_destinazione_c,
        nome_file_graf_ciclo)+".jpg",'jpg');
     hold off;

end


disp('- dati termici acquisiti, elaborati e salvati con
   successo')




%% FBGs DATA ACQUISITION AND ANALYSIS.
percorso_dati =cartella_destinazione+"\
   Acquisizioni_Interrogatore\"; %folder where the smart soft
    files are saved 'directory\'

s = input("inserisci il numero di sensori FBG: ");

% I get the list of files in the folder
disp('ELENCO FILE FBG')

% Saves a list of all .log file names
file_list = dir(percorso_dati);%    una struttura 5x1 che
   contiene nome, cartella, data, bytes, isdir, datenum

% Now I need to check that the data are already put in
   ascending order (in
% theory it should already be so)
k_pc = 2;%num "ghost file" in the folder (depends on pc)
k = length(file_list)-k_pc; % num totale dei file .log
   trovati nella cartella
for j=1:k+k_pc
```

```
    for i = j:k+k_pc
        if(file_list(j).datenum > file_list(i).datenum)
            tmp = file_list(j);
            file_list(j) = file_list(i);
            file_list(i) = tmp;
        end
    end
end
clear i j

fprintf('- Sono stati trovati e ordinati %d file .log.\n',k)




% Data loading

disp('CARICAMENTO FILE FBG')


dati_FBG_originali = struct( 'Time', [], 'FBG_1',[],'FBG_2'
   ,[],'FBG_3',[],'FBG_4',[]);
ok=0;
for n=1:k


    clear opts;
  opts = delimitedTextImportOptions("NumVariables", 2);

    % Specify range and delimiter
    opts.DataLines = [1, 1];
    opts.Delimiter = "=";

    % Specify column names and types
    opts.VariableNames = ["Var1", "VarName2"];
    opts.SelectedVariableNames = "VarName2";
    opts.VariableTypes = ["string", "double"];

    % Specify file level properties
    opts.ExtraColumnsRule = "ignore";
    opts.EmptyLineRule = "read";

    % Specify variable properties
    opts = setvaropts(opts, "Var1", "WhitespaceRule", "
       preserve");
```

```
opts = setvaropts(opts, "Var1", "EmptyFieldRule", "auto")
   ;
opts = setvaropts(opts, "VarName2", "TrimNonNumeric",
   true);
opts = setvaropts(opts, "VarName2", "DecimalSeparator",
   ",");
opts = setvaropts(opts, "VarName2", "ThousandsSeparator",
    ".");


nome_file=percorso_dati+file_list(n).name;
t_sync_FBG=table2array(readtable(nome_file, opts));

fprintf("-caricamento dati da file n %d\n",n);

clear opts;

    %Data reading
    opts = delimitedTextImportOptions("NumVariables", 5);

    % Specify range and delimiter
    opts.DataLines = [6, Inf];
    opts.Delimiter = "\t";

    % Specify column names and types
    opts.VariableNames = ["StartTimeUTC1686232646077080",
        "VarName2", "VarName3", "VarName4", "VarName5"];
    opts.VariableTypes = ["double", "double", "double", "
       double", "double"];

    % Specify file level properties
    opts.ExtraColumnsRule = "ignore";
    opts.EmptyLineRule = "read";

    % Specify variable properties
    opts = setvaropts(opts, ["
       StartTimeUTC1686232646077080", "VarName2", "
       VarName3", "VarName4", "VarName5"], "
       TrimNonNumeric", true);
    opts = setvaropts(opts, ["
       StartTimeUTC1686232646077080", "VarName2", "
       VarName3", "VarName4", "VarName5"], "
       DecimalSeparator", ",");
    opts = setvaropts(opts, ["
       StartTimeUTC1686232646077080", "VarName2", "
       VarName3", "VarName4", "VarName5"], "
       ThousandsSeparator", ".");



    dati_FBG = readtable(nome_file,opts);
    for z=1:length(dati_FBG.VarName2)
        if dati_FBG.VarName2(z)==0
            dati_FBG.VarName2(z)=1529.75;
        end
    end
    fuso_orario=hours(1);%the interrogator encodes posix
       according to UTC time zone while the climate
       chamber according to the time zone of the pc to
       which it is connected

    Time=dati_FBG.StartTimeUTC1686232646077080;
    for i=1:length(Time)
        Time(i)=posixtime(datetime((Time(i)+t_sync_FBG(1)
           ), 'ConvertFrom', 'posix')+fuso_orario);
    end

    if ok==0
        lambda_FBG(1)=round(dati_FBG.VarName2(1));
        lambda_FBG(2)=round(dati_FBG.VarName3(1));
        lambda_FBG(3)=round(dati_FBG.VarName4(1));
        lambda_FBG(4)=round(dati_FBG.VarName5(1));
        ok=1;
    end




     %I gradually merge the data from the current file
        with the data from the files already read by
        matching the lamdas

     dati_FBG_originali(1).Time=cat(1,dati_FBG_originali
        (1).Time,Time);
     FBGs_a=fieldnames(dati_FBG_originali);
     FBGs_b=fieldnames(dati_FBG);
     for i=2:s+1
         for j=2:s+1
             if lambda_FBG(j-1)==round(dati_FBG.(FBGs_b{i
                })(1))
```

```
                        dati_FBG_originali(1).(FBGs_a{j})=cat(1,
                            dati_FBG_originali(1).(FBGs_a{j}),
                           dati_FBG.(FBGs_b{i}));
                    end
                end
            end




end

for d=2:s+1
    l=length(dati_FBG_originali(1).Time);
    sottraendo=ones(l,1);
    graph_time_fbg=(dati_FBG_originali(1).Time-sottraendo*
       dati_FBG_originali(1).Time(1))/3600;

    figure;
    plot(graph_time_fbg,dati_FBG_originali(1).(FBGs_a{d}), 'b
       ', 'LineWidth',1); %dati_FBG_originali(1).Time
    hold on;
    grid on;
    title("Acquisizione "+cartella_acquisizione+" FBG "+
       num2str(lambda_FBG(d-1)));
    xlabel("Tempo [h]");
    ylabel("Lunghezza d'onda [nm]");
%      x1=plot(dati_CT_ciclo(1).StartTime*ones(size(
   dati_FBG_originali(1).(FBGs_a{d}))),dati_FBG_originali(1)
   .(FBGs_a{d}),'g');
%      x2=plot(dati_CT_ciclo(length(dati_CT_ciclo)).EndTime*
   ones(size(dati_FBG_originali(1).(FBGs_a{d}))),
   dati_FBG_originali(1).(FBGs_a{d}),'g');
    x1=plot((dati_CT_ciclo(1).StartTime-dati_FBG_originali(1)
       .Time(1))/3600*ones(size(dati_FBG_originali(1).(FBGs_a
       {d}))),dati_FBG_originali(1).(FBGs_a{d}),'g');
    x2=plot((dati_CT_ciclo(length(dati_CT_ciclo)).EndTime-
       dati_FBG_originali(1).Time(1))/3600*ones(size(
       dati_FBG_originali(1).(FBGs_a{d}))),dati_FBG_originali
       (1).(FBGs_a{d}),'g');
    set(x1, 'HandleVisibility', 'off');
    set(x2, 'HandleVisibility', 'off');
    xlim([min(graph_time_fbg),max(graph_time_fbg)]);
    ylim([min(dati_FBG_originali(1).(FBGs_a{d})),max(
       dati_FBG_originali(1).(FBGs_a{d}))]);
```

```
hold off;
saveas(gcf,fullfile(cartella_destinazione,"
   Grafico_Acquisizione_"+cartella_acquisizione+"_FBG_"+
   num2str(lambda_FBG(d-1))+"_completa.jpg"),'jpg');

 %I create and save an overall graph comparing T and
    lamda
figure;

yyaxis left;
plot(graph_time_fbg,dati_FBG_originali(1).(FBGs_a{d}), 'b
   ', 'LineWidth',1);
ylim([min(dati_FBG_originali(1).(FBGs_a{d})),max(
   dati_FBG_originali(1).(FBGs_a{d}))]);
ylabel("Lunghezza d'onda [nm]");

hold on;
yyaxis right;
plot(graph_time,dati_CT_originali(1).T_camera, 'r', '
   LineWidth',1);
grid on;
title("Acquisizione "+cartella_acquisizione+" FBG "+
   num2str(lambda_FBG(d-1))+" confronto");
xlabel("Tempo [h]");
legend('dati FBG','T-camera','Location','Best');


set(x1, 'HandleVisibility', 'off');
set(x2, 'HandleVisibility', 'off');
x1=plot((dati_CT_ciclo(1).StartTime-dati_FBG_originali(1)
   .Time(1))/3600*ones(size(dati_FBG_originali(1).(FBGs_a
   {d}))),dati_FBG_originali(1).(FBGs_a{d}),'g');
x2=plot((dati_CT_ciclo(length(dati_CT_ciclo)).EndTime-
   dati_FBG_originali(1).Time(1))/3600*ones(size(
   dati_FBG_originali(1).(FBGs_a{d}))),dati_FBG_originali
   (1).(FBGs_a{d}),'g');
ylabel('Temperatura[ C ]');
ylim([min(dati_CT_originali(1).T_camera),max(
   dati_CT_originali(1).T_camera)]);
xlim([min(graph_time_fbg),max(graph_time_fbg)]);
hold off;
saveas(gcf,fullfile(cartella_destinazione,"
   Grafico_Confronto_"+cartella_acquisizione+"_FBG_"+
   num2str(lambda_FBG(d-1))+"_completa.jpg"),'jpg');
```

```
end



disp("-dati dei sensori FBG acquisiti con successo\n");

% FBG DATA ANALYSIS


  dati_FBG_ciclo = struct('Name',{},'StartTime',{},'EndTime'
     ,{},'Time',[],'t_Start_Step',[],'t_End_Step',[],'FBG_1'
     ,[],'Mean_FBG_1_Step',[],'FBG_2' , [],'Mean_FBG_2_Step'
     ,[],'FBG_3',[],'Mean_FBG_3_Step',[],'FBG_4',[],'
     Mean_FBG_4_Step', []);


disp('ANALISI DATI FBG')
start_ciclo=0;
start_step=0;
c_step=1;
n=1;


%PART THAT TAKES THE DATA FOR THE INTERVALS FOUND FROM THE
   SECTION
%OF THE THERMAL CHAMBER
    for i=2:length(dati_FBG_originali(1).Time)


        %I look for the instants of time at which a sampling
           was done.
        %of FBGs closest to the characteristic instants
           identified by the
        %thermal camera and save them

        if n<ciclo
            while dati_FBG_originali(1).Time(i)>dati_CT_ciclo
               (n).EndTime && start_ciclo==0
                n=n+1;
            end
```

```
if(dati_FBG_originali(1).Time(i)>=dati_CT_ciclo(n
   ).StartTime) && (dati_FBG_originali(1).Time(i)
   <=dati_CT_ciclo(n).t_End_Step(1)) &&
   start_ciclo==0

    if abs((dati_FBG_originali(1).Time(i) -
       dati_CT_ciclo(n).StartTime))<=abs((
       dati_CT_ciclo(n).StartTime-
       dati_FBG_originali(1).Time(i-1)))
        dati_FBG_ciclo(n).StartTime=
           dati_FBG_originali(1).Time(i);
    else
        dati_FBG_ciclo(n).StartTime=
           dati_FBG_originali(1).Time(i-1);

    end
    start_ciclo=1;
    dati_FBG_ciclo(n).Name="Ciclo_"+n;

end

if(dati_FBG_originali(1).Time(i)>=dati_CT_ciclo(n
   ).t_Start_Step(c_step)) && start_ciclo==1 &&
   start_step==0


     if abs((dati_FBG_originali(1).Time(i)-
        dati_CT_ciclo(n).t_Start_Step(c_step)))<=
        abs((dati_CT_ciclo(n).t_Start_Step(c_step)
        -dati_FBG_originali(1).Time(i-1)))
            dati_FBG_ciclo(n).t_Start_Step(c_step)
               =dati_FBG_originali(1).Time(i);
    else
        dati_FBG_ciclo(n).t_Start_Step(c_step)=
           dati_FBG_originali(1).Time(i-1);
    end
    start_step=1;

end
if(dati_FBG_originali(1).Time(i)>=dati_CT_ciclo(n
   ).t_End_Step(c_step)) && start_ciclo==1 &&
   start_step==1

        if (dati_FBG_originali(1).Time(i)-
           dati_CT_ciclo(n).t_End_Step(c_step))<=(
                     dati_CT_ciclo(n).t_End_Step(c_step)-
                     dati_FBG_originali(1).Time(i-1))
                      dati_FBG_ciclo(n).t_End_Step(c_step)=
                         dati_FBG_originali(1).Time(i);
                  else
                      dati_FBG_ciclo(n).t_End_Step(c_step)=
                         dati_FBG_originali(1).Time(i-1);
                  end
                  start_step=0;
                  c_step=c_step+1;
           end
           if(dati_FBG_originali(1).Time(i)>=dati_CT_ciclo(n
              ).EndTime) && start_ciclo==1 && start_step==0

                  if abs((dati_FBG_originali(1).Time(i)-
                     dati_CT_ciclo(n).EndTime))<=abs((
                     dati_CT_ciclo(n).EndTime-
                     dati_FBG_originali(1).Time(i-1)))
                      dati_FBG_ciclo(n).EndTime=
                         dati_FBG_originali(1).Time(i);
                  else
                      dati_FBG_ciclo(n).EndTime=
                         dati_FBG_originali(1).Time(i-1);
                  end
                   start_ciclo=0;
                   c_step=1;
                   n=n+1;

           end
       end

   end

   if start_ciclo==1
       disp("LE ACQUISIZIONI DEGLI FBG TERMINANO PRIMA DI
          QUELLE DELLA CAMERA CLIMATICA");
   end


%% COMPLETES THE FILLING OF THE FIELDS OF dati_FBG_ciclo


 FBGs_a=fieldnames(dati_FBG_ciclo);
 for i = 1:(ciclo-1)
    dati_FBG_ciclo(i).Time=dati_FBG_originali.Time(find(
     dati_FBG_originali(1).Time==dati_FBG_ciclo(i).
     StartTime):find(dati_FBG_originali(1).Time==
     dati_FBG_ciclo(i).EndTime));

   for d=2:s+1
       j=d*2+3; %To select only the odd numbers from 7 to 13
       dati_FBG_ciclo(i).(FBGs_a{j})=dati_FBG_originali.(
          FBGs_a{j})(find(dati_FBG_originali(1).Time==
          dati_FBG_ciclo(i).StartTime):find(
          dati_FBG_originali(1).Time==dati_FBG_ciclo(i).
          EndTime));

       for c_step=1:tot_step
           dati_FBG_ciclo(i).(FBGs_a{j+1})(c_step)=mean(
              dati_FBG_ciclo(i).(FBGs_a{j})(find(
              dati_FBG_ciclo(i).Time==dati_FBG_ciclo(i).
              t_Start_Step(c_step)):find(dati_FBG_ciclo(i).
              Time==dati_FBG_ciclo(i).t_End_Step(c_step))));

       end
   end
end



% Data saving



for i=1:(ciclo-1)

   l=length(dati_FBG_ciclo(i).Time);
   sottraendo=ones(l,1);
   graph_time_fbg_c=(dati_FBG_ciclo(i).Time-sottraendo*
      dati_FBG_ciclo(i).Time(1))/3600;

   l=length(dati_CT_ciclo(i).Time);
   sottraendo=ones(l,1);
   graph_time_c=(dati_CT_ciclo(i).Time-sottraendo*
      dati_CT_ciclo(i).Time(1))/3600;

   %mkdir(fullfile(cartella_destinazione, "Ciclo_"+num2str(i
      )));
   cartella_destinazione_c=fullfile(cartella_destinazione,"
      Ciclo_"+num2str(i));
```

```
date_str=datestr(datetime(dati_FBG_ciclo(i).Time(1), '
   ConvertFrom', 'posixtime'));
date_str(3)='_';
date_str(7)='_';
date_str(12)='_';
date_str(15)='_';
date_str(18)='_';
date_str(20)='_';
nome_file_dati_ciclo=date_str + "
   Dati_Interrogatore_Ciclo_n " + num2str(i)+"_(E="+
   num2str(errore_ammesso)+").mat" ;
dati_FBG_ciclo_singolo=dati_FBG_ciclo(i);
save (fullfile(cartella_destinazione_c,
   nome_file_dati_ciclo) , 'dati_FBG_ciclo_singolo') ;
disp("- dati Ciclo n "+num2str(i)+" acquisiti con
   successo");




for d=2:s+1


      date_str(3)=' ';
      date_str(7)=' ';
      date_str(12)=' ';
      date_str(15)=' ';
      date_str(18)=' ';
      date_str(20)=' ';


    j=d*2+3;
    figure;

    yyaxis left;
    plot(graph_time_fbg_c,dati_FBG_ciclo(i).(FBGs_a{j}),'
       b', 'LineWidth', 1.5);
    ylabel("Lunghezza d'onda[nm]");
    ylim([min(dati_FBG_ciclo(i).(FBGs_a{j})),max(
       dati_FBG_ciclo(i).(FBGs_a{j}))]);

    hold on ;
    yyaxis right;
    plot(graph_time_c,dati_CT_ciclo(i).T_set,'k', '
       LineWidth', 1);
```

```
plot(graph_time_c,dati_CT_ciclo(i).T_camera,'r', '
   LineWidth', 1);
grid on;
title(date_str+"grafico Completo Ciclo n  "+num2str(i
   )+" FBG "+num2str(lambda_FBG(d-1)));
text(dati_CT_ciclo(i).t_End_Step(1),T_min-10,"Errore
   ammesso="+num2str(errore_ammesso)+" C ");
xlabel('Tempo[h]');
ylabel('Temperatura[ C ]');
ylim([min(dati_CT_ciclo(i).T_camera),max(
   dati_CT_ciclo(i).T_camera)]);
xlim([min(graph_time_fbg_c),max(graph_time_fbg_c)]);
legend('Lambda','T-set','T-camera','Location','Best')
   ;


date_str(3)='_';
date_str(7)='_';
date_str(12)='_';
date_str(15)='_';
date_str(18)='_';
date_str(20)='_';

for c_step=1:tot_step

     x1=plot((dati_CT_ciclo(i).t_Start_Step(c_step)-
        dati_CT_ciclo(i).StartTime)/3600*ones(size(
        dati_FBG_ciclo(i).Time)), dati_FBG_ciclo(i).
        Time,'g', 'LineWidth',0.5);
    x2=plot((dati_CT_ciclo(i).t_End_Step(c_step)-
       dati_CT_ciclo(i).StartTime)/3600*ones(size(
       dati_FBG_ciclo(i).Time)), dati_FBG_ciclo(i).
       Time,'g', 'LineWidth',0.5);
    f=size(dati_CT_ciclo(i).T_camera);
    a1=area([(dati_CT_ciclo(i).t_Start_Step(c_step)-
       dati_CT_ciclo(i).StartTime)/3600,(
       dati_CT_ciclo(i).t_End_Step(c_step)-
       dati_CT_ciclo(i).StartTime)/3600 ], [max(
       dati_CT_ciclo(i).T_camera), max(dati_CT_ciclo(
       i).T_camera)], min(dati_CT_ciclo(i).T_camera),
        'FaceColor', 'green', 'FaceAlpha', 0.3);
    set(x1, 'HandleVisibility', 'off');
    set(x2, 'HandleVisibility', 'off');
    set(a1, 'HandleVisibility', 'off');
```

```
end
nome_file_graf_ciclo=date_str+"Grafico Completo Ciclo
     n  "+num2str(i)+" FBG "+num2str(lambda_FBG(d-1))
   +"_(E="+num2str(errore_ammesso)+")";
saveas(gcf,fullfile(cartella_destinazione_c,
   nome_file_graf_ciclo)+".pdf",'pdf');
saveas(gcf, fullfile(cartella_destinazione_c,
   nome_file_graf_ciclo)+".jpg",'jpg');
hold off;


date_str(3)=' ';
date_str(7)=' ';
date_str(12)=' ';
date_str(15)=' ';
date_str(18)=' ';
date_str(20)=' ';


%I save the graphs with only the wavelength
figure;

plot(graph_time_fbg_c,dati_FBG_ciclo(i).(FBGs_a{j}),'
   b', 'LineWidth', 1.5);
ylabel("Lunghezza d'onda[nm]");
ylim([min(dati_FBG_ciclo(i).(FBGs_a{j})),max(
   dati_FBG_ciclo(i).(FBGs_a{j}))]);

hold on ;

grid on;
title(date_str+"grafico Interrogatore Ciclo n  "+
   num2str(i)+" FBG "+num2str(lambda_FBG(d-1)));
text(dati_FBG_ciclo(i).t_End_Step(1),min(
   dati_FBG_ciclo(i).(FBGs_a{j})-0.2),"Errore ammesso
   ="+num2str(errore_ammesso)+" C ");
xlabel('Tempo[h]');
xlim([min(graph_time_fbg_c),max(graph_time_fbg_c)]);


 date_str(3)='_';
date_str(7)='_';
date_str(12)='_';
date_str(15)='_';
date_str(18)='_';
```

```
        date_str(20)='_';

        for c_step=1:tot_step

           x1=plot((dati_FBG_ciclo(i).t_Start_Step(c_step)-
              dati_FBG_ciclo(i).StartTime)/3600*ones(size(
              dati_FBG_ciclo(i).Time)), dati_FBG_ciclo(i).(
              FBGs_a{j}),'g', 'LineWidth',0.5);
            x2=plot((dati_FBG_ciclo(i).t_End_Step(c_step)-
               dati_FBG_ciclo(i).StartTime)/3600*ones(size(
               dati_FBG_ciclo(i).Time)), dati_FBG_ciclo(i).(
               FBGs_a{j}),'g', 'LineWidth',0.5);
            f=size(dati_FBG_ciclo(i).(FBGs_a{j}));
            a1=area([(dati_FBG_ciclo(i).t_Start_Step(c_step)-
               dati_FBG_ciclo(i).StartTime)/3600,(
               dati_FBG_ciclo(i).t_End_Step(c_step)-
               dati_FBG_ciclo(i).StartTime)/3600 ], [max(
               dati_FBG_ciclo(i).(FBGs_a{j})), max(
               dati_FBG_ciclo(i).(FBGs_a{j}))], min(
               dati_FBG_ciclo(i).(FBGs_a{j})), 'FaceColor', '
               green', 'FaceAlpha', 0.3);
            set(x1, 'HandleVisibility', 'off');
            set(x2, 'HandleVisibility', 'off');
            set(a1, 'HandleVisibility', 'off');

        end
        nome_file_graf_ciclo=date_str+"Grafico Interrogatore
           Ciclo n "+num2str(i)+" FBG "+num2str(lambda_FBG(d
           -1))+"_(E="+num2str(errore_ammesso)+")";
        saveas(gcf,fullfile(cartella_destinazione_c,
           nome_file_graf_ciclo)+".pdf",'pdf');
        saveas(gcf, fullfile(cartella_destinazione_c,
           nome_file_graf_ciclo)+".jpg",'jpg');
        hold off;


    end
end


%% CALIBRATION
colori = ['r', 'g', 'b', 'c', 'm', 'y', 'k'];
retta_taratura = struct('Name',{},'Coefficienti',[],'Lambda_d
   ',[]);
u = input("scrivere 1 se si vuole INCLUDERE IL PRIMO CICLO, 0
    per ESCLUDERLO: ");
disp("INIZIO CALCOLO PER TARATURA SENSORI FBG");
if u==1
for n=2:s+1 %I perform the calibration for each FBG
    disp("-Taratura FBG_"+num2str(lambda_FBG(n-1)));
    retta_taratura(n-1).Lambda_d=lambda_FBG(n-1);
    figure;
    hold on;
    x=[];
    y=[];
    tmpx=[];
    tmpy=[];
    nome_graf="Retta_di_taratura_FBG_"+num2str(lambda_FBG(n
       -1))+"_(Acquisizione_"+num2str(cartella_acquisizione)
       +")";
    for i=1:(ciclo-1)
        j=n*2+3;
        tmpx=dati_CT_ciclo(i).Mean_T_Step;
        tmpy=dati_FBG_ciclo(i).(FBGs_a{j+1});
        x=cat(2,x,tmpx);
        y=cat(2,y,tmpy);
        colore = colori(mod(i, length(colori)) + 1); %
           Selezione del colore dal tavolozza dei colori
        p=plot(tmpx,tmpy,'o','MarkerSize',5,'Color',colore);


    end

    retta_taratura(n-1).Coefficienti=polyfit(x,y,1);
    y_r=polyval(retta_taratura(n-1).Coefficienti,x);
    retta_taratura(n-1).Name="FBG_"+num2str(lambda_FBG(n-1))
       +"(A_"+num2str(cartella_acquisizione)+")";
    plot(x,y_r,'g','LineWidth',1.5);
    title("Retta di taratura FBG "+num2str(lambda_FBG(n-1))+"
        (Acquisizione "+num2str(cartella_acquisizione)+")");
    xlabel('Temperatura [ C ]');
    ylabel("Lunghezza d'onda [nm]");

    for c=1:ciclo-1
    legendLabels{c} = "cycle "+num2str(c);
    end

    legend (legendLabels,'Location','NorthWest');
    hold off;
    retta_taratura_singolo=retta_taratura(n-1);
    save(fullfile(cartella_destinazione,nome_graf+".mat"), '
       retta_taratura_singolo');
    saveas(gcf,fullfile(cartella_destinazione,nome_graf)+".
       pdf",'pdf');
    saveas(gcf, fullfile(cartella_destinazione,nome_graf)+".
       jpg",'jpg');
end

nome_graf="Retta_di_taratura_FBGs_(Acquiaizione_"+num2str(
   cartella_acquisizione)+")";
save(fullfile(cartella_destinazione,nome_graf+".mat"), '
   retta_taratura');

disp("TUTTI I DATI ANALIZZATI CON SUCCESSO");
disp("FINE PROGRAMMA");

else
    for n=2:s+1 %I perform the calibration for each FBG
    disp("-Taratura FBG_"+num2str(lambda_FBG(n-1)));
    retta_taratura(n-1).Lambda_d=lambda_FBG(n-1);
    figure;
    hold on;
    x=[];
    y=[];
    tmpx=[];
    tmpy=[];
    nome_graf="Retta_di_taratura_noprimo_FBG_"+num2str(
       lambda_FBG(n-1))+"_(Acquisizione_"+num2str(
       cartella_acquisizione)+")";
    for i=1:(ciclo-1)
        j=n*2+3;
        tmpx=dati_CT_ciclo(i).Mean_T_Step;
        tmpy=dati_FBG_ciclo(i).(FBGs_a{j+1});
        x=cat(2,x,tmpx);
        y=cat(2,y,tmpy);
        colore = colori(mod(i, length(colori)) + 1); % Color
           selection from the color palette
        p=plot(tmpx,tmpy,'o','MarkerSize',5,'Color',colore);


    end

    retta_taratura(n-1).Coefficienti=polyfit(x,y,1);
    y_r=polyval(retta_taratura(n-1).Coefficienti,x);
    retta_taratura(n-1).Name="FBG_"+num2str(lambda_FBG(n-1))
```

```
        +"(A_"+num2str(cartella_acquisizione)+")";
    plot(x,y_r,'g','LineWidth',1.5);
    title("Retta di taratura FBG "+num2str(lambda_FBG(n-1))+"
         (Acquisizione "+num2str(cartella_acquisizione)+")");
    xlabel('Temperatura [ C ]');
    ylabel("Lunghezza d'onda [nm]");

    for c=1:ciclo-2
    legendLabels2{c} = "cycle "+num2str(c);
    end

    legend (legendLabels2,'Location','NorthWest');
    hold off;
    retta_taratura_singolo=retta_taratura(n-1);
    save(fullfile(cartella_destinazione,nome_graf+".mat"), '
        retta_taratura_singolo');
    saveas(gcf,fullfile(cartella_destinazione,nome_graf)+".
        pdf",'pdf');
    saveas(gcf, fullfile(cartella_destinazione,nome_graf)+".
        jpg",'jpg');
end

nome_graf="Retta_di_taratura_FBGs_(Acquiaizione_"+num2str(
    cartella_acquisizione)+")";
save(fullfile(cartella_destinazione,nome_graf+".mat"), '
    retta_taratura');



disp("TUTTI I DATI ANALIZZATI CON SUCCESSO");
disp("FINE PROGRAMMA");
end
%% TARATURA NEW
%
% colori = ['r', 'g', 'b', 'c', 'm', 'y', 'k'];
% retta_taratura = struct('Name',{},'Coefficienti',[],'
    Lambda_d',[]);
% disp("INIZIO CALCOLO PER TARATURA SENSORI FBG");
% dev_std = struct('Lambda_b',{},'Dev_step',[],'Mean_dev',[])
    ;
%
% for n=2:s+1 %eseguo la taratura per ogni fbg
%     disp("-Taratura FBG_"+num2str(lambda_FBG(n-1)));
%     dev_std(n-1).Lambda_b=lambda_FBG(n-1);
%     retta_taratura(n-1).Lambda_d=lambda_FBG(n-1);
```

```
%
%     %Plotto tutti i punti ottenuti per ogni temperatura
%     figure;
%     hold on;
%     x=[];
%     y=[];
%     tmpx=[];
%     tmpy=[];
%     tmpd=[];
%     x_mean=[];
%     y_mean=[];
%     nome_graf="Retta_di_taratura_FBG_"+num2str(lambda_FBG(n
   -1))+"_(Acquisizione_"+num2str(cartella_acquisizione)+",E
   ="+num2str(errore_ammesso)+")";
%
%
%     for i=6:(ciclo-1)
%         j=n*2+3;
%         tmpx=dati_CT_ciclo(i).Mean_T_Step;
%         tmpy=dati_FBG_ciclo(i).(FBGs_a{j+1});
%         x=cat(2,x,tmpx);
%         y=cat(2,y,tmpy);
%         colore = colori(mod(i, length(colori)) + 1); %
   Selezione del colore dal tavolozza dei colori
%         plot(tmpx,tmpy,'o','MarkerSize',5,'Color',colore);
%         plot(tmpx,tmpy,'-','MarkerSize',5,'Color',colore);
%     end
%
%     %effettuo la taratura
%     retta_taratura(n-1).Coefficienti=polyfit(x,y,1);
%     y_r=polyval(retta_taratura(n-1).Coefficienti,x);
%     retta_taratura(n-1).Name="FBG_"+num2str(lambda_FBG(n-1)
   )+"(A_"+num2str(cartella_acquisizione)+")";
%     plot(x,y_r,'r','LineWidth',2);
%     title("Retta di taratura FBG "+num2str(lambda_FBG(n-1))
   +" (Acquisizione "+num2str(cartella_acquisizione)+",E="+
   num2str(errore_ammesso)+")");
%     xlabel('Temperatura [ C ]');
%     ylabel("Lunghezza d'onda [nm]");
%
%      %salvo i grafici e i dati
%     retta_taratura_singolo=retta_taratura(n-1);
%
%     savefig(fullfile(cartella_destinazione,'Grafico_'+
   nome_graf)+'.fig');
```

```
%     cartella_pdf=fullfile(cartella_destinazione,"Copie_PDF
   ");
%     cartella_jpg=fullfile(cartella_destinazione,"Copie_JPG
   ");
%     %mkdir(cartella_pdf);
%     %mkdir(cartella_jpg);
% %     saveas(gcf,fullfile(cartella_pdf,'Grafico_'+nome_graf
   )+".pdf",'pdf');
% %     saveas(gcf, fullfile(cartella_jpg,'Grafico_'+
   nome_graf)+".jpg",'jpg');
%
%     %save(fullfile(cartella_destinazione,nome_graf+".mat"),
    'retta_taratura_singolo');
%     hold off;
%
%     %calcolo la deviazione standard per ogni step e anche
   la media della
%     %deviazione (per ogni sensore)
%     for j=1:c_step
%         for i=6:(ciclo-1)
%             tmpd(i)=dati_FBG_ciclo(i).(FBGs_a{n*2+3+1})(j);
%         end
%         dev_std(n-1).Dev_step(j)=std(tmpd);
%     end
%     dev_std(n-1).Mean_dev=mean(dev_std(n-1).Dev_step);
% end
%
%
% % save(fullfile(cartella_destinazione,"
   Dati_Retta_di_taratura_FBGs_(Acquisizione_"+num2str(
   cartella_acquisizione)+",E="+num2str(errore_ammesso)+")
   "+".mat"), 'retta_taratura');
%
% %produco il grafico per visualizzare le deviazioni standard
% figure;
% hold on;
% grid on;
% grid minor;
%
% if cd==1 || cd==2
%     x=T_start:T_step:T_stop;
% else
%     x=T_start:T_step:(T_start+T_step*((tot_step-1)/2+1));
%
%     x=cat(1,x,(T_start+T_step*((tot_step-1)/2+1)):-T_step:
   T_stop)
% end
% name=[];
% for i=1:s
%     colore = colori(mod(i, length(colori)) + 1); %
   Selezione del colore dal tavolozza dei colori
%      x2=plot(x,dev_std(i).Dev_step, 'o','MarkerSize',5,'
   Color',colore);
%     x1= plot(x,dev_std(i).Dev_step, '-','MarkerSize',5,'
   Color',colore);
%     plot(x, dev_std(i).Mean_dev*ones(1,length(x)), '-', '
   MarkerSize', 5, 'Color', colore);
%     name= cat(1,name,num2str(dev_std(i).Lambda_b(1)));
%     name= cat(1,name,'Mean');
%     set(x1, 'HandleVisibility', 'off');
%     %set(x2, 'HandleVisibility', 'off');
%
% end
% title("Deviazione standard FBGs (Acquisizione "+num2str(
   cartella_acquisizione)+",E="+num2str(errore_ammesso)+")");
% legend(name,'Location', 'Best');
% xlabel('Temperatura[$^oC$]');
% ylabel('Deviazione standard[nm]');
% xlim([min(x),max(x)]);
% xticks(x);
%
% %salvo
% nome_graf="Deviazione_standard_FBGs_(Acquisizione_"+num2str
   (cartella_acquisizione)+",E="+num2str(errore_ammesso)+")";
% % savefig(fullfile(cartella_destinazione,'Grafico_'+
   nome_graf)+'.fig');
% % saveas(gcf,fullfile(cartella_pdf,'Grafico_'+nome_graf)+".
   pdf",'pdf');
% % saveas(gcf, fullfile(cartella_jpg,'Grafico_'+nome_graf)
   +".jpg",'jpg');
% % save(fullfile(cartella_destinazione,"Dati_"+nome_graf+".
   mat"), 'dev_std');
% hold off;
% disp("TUTTI I DATI ANALIZZATI CON SUCCESSO");
% disp("FINE PROGRAMMA");
```

Lambda2temp.m

```
% DATA COLLECTION

wavelength_FBG = dati_FBG_originali.FBG_1;
lambda_0 = retta_taratura_singolo.Coefficienti(2);  % prendo
   il termine noto della retta di taratura
k_T = retta_taratura_singolo.Coefficienti(1);    % prendo il
   coefficente lineare della retta di taratura


% LINEAR RELATION

%T_0 = (wavelength_FBG(1)-lambda_0)/k_T;

T_FBG = (wavelength_FBG-lambda_0)/k_T;


% RMSE CALCULATION

vq = linspace(0,1e4,length(dati_FBG_originali.FBG_1));
T_cam_new = interp1(dati_CT_originali.T_camera,vq,'linear','
   extrap');

N = length(T_FBG);

for i=1:length(T_FBG)
    RMSE = sqrt(sum((T_cam_new(i)-T_FBG(i)).^2)/N);
    i=i+1;
end


% CONVERTION IN TIME EPOCH

Time_epochFBG = datetime(dati_FBG_originali.Time*1e3,'
   ConvertFrom','epochtime','Epoch','1970-01-01','
   TicksPerSecond',1000);


Time_epochCC = datetime(dati_CT_originali.Time*1e3,'
   ConvertFrom','epochtime','Epoch','1970-01-01','
   TicksPerSecond',1000);


% GRAPHS
```

```
figure()

plot(Time_epochFBG,T_FBG,'r',Time_epochCC,dati_CT_originali.
   T_camera,'b',Linewidth=1.2)

xlabel('Time')

ylabel('Temperature-FBG [ C ]')

grid on

legend('T_{FBG}','T_{ChamberProbe}')

% hold on
%
% yyaxis right
%
% plot(Time_epochCC,dati_CT_originali.T_camera,'b')
%
% legend('T_{FBG}','T_{ChamberProbe}')
%
% ylabel('Temperature-ChamberProbe [ C ]')
```

ImportDati_Temp.m

```
%% THE FIRST 5 LINES OF EACH .TXT FILE WERE DELETED TO
   FACILITATE IMPORTING



%% Import data from text file
% Script for importing data from the following text file:
%
%    filename: C:\Users\User\Desktop\Test batteria-ESC\Raw
   data\20240319-1539_prova_propulsione.log
%
% Auto-generated by MATLAB on 20-Mar-2024 09:44:32

%% Set up the Import Options and import the data
opts = delimitedTextImportOptions("NumVariables", 3);

% Specify range and delimiter
opts.DataLines = [1, Inf];
opts.Delimiter = "\t";
```

```
% Specify column names and types
opts.VariableNames = ["VarName1", "VarName2", "VarName3"];
opts.VariableTypes = ["double", "double", "double"];

% Specify file level properties
opts.ExtraColumnsRule = "ignore";
opts.EmptyLineRule = "read";

% Specify variable properties
opts = setvaropts(opts, ["VarName1", "VarName2", "VarName3"],
    "TrimNonNumeric", true);
opts = setvaropts(opts, ["VarName1", "VarName2", "VarName3"],
    "DecimalSeparator", ",");
opts = setvaropts(opts, ["VarName1", "VarName2", "VarName3"],
    "ThousandsSeparator", ".");

% Import the data
%I create tables in which to enter data and transform them
   into matrices
A = readtable("C:\Users\User\Desktop\Test batteria-ESC\Raw
   data\20240319-1539_prova_propulsione.log", opts);

B = readtable("C:\Users\User\Desktop\Test batteria-ESC\Raw
   data\20240319-1612_prova_propulsione2.log", opts);

C = readtable("C:\Users\User\Desktop\Test batteria-ESC\Raw
   data\20240319-1638_prova_propulsione3.log", opts);

A = table2array(A);

B = table2array(B);

C = table2array(C);

%I load the coefficients of the two calibration straight
   lines.

load("Retta_di_taratura_FBG_1530_(Acquisizione_03).mat")

taratura1530 = retta_taratura_singolo;

load("Retta_di_taratura_FBG_1552_(Acquisizione_04).mat")

taratura1552 = retta_taratura_singolo;
```

```
%% Clear temporary variables
clear opts

% TIME CONVERTION

%%THESE STARTIMEutc COME FROM THE FIRST LINE OF EACH FILE .
   LOG
StartimeUTC1 = 1710859170.245015;
StartimeUTC2 = 1710861129.442897;
StartimeUTC3 = 1710862698.948571;

Time_epochFBG1 = datetime((A(:,1)+StartimeUTC1)*1e3,'
   ConvertFrom','epochtime','Epoch','1970-01-01','
   TicksPerSecond',1000);
Time_epochFBG2 = datetime((B(:,1)+StartimeUTC2)*1e3,'
   ConvertFrom','epochtime','Epoch','1970-01-01','
   TicksPerSecond',1000);
Time_epochFBG3 = datetime((C(:,1)+StartimeUTC3)*1e3,'
   ConvertFrom','epochtime','Epoch','1970-01-01','
   TicksPerSecond',1000);

% WAVELENGTH GRAPHS

figure(1)
plot(Time_epochFBG1,A(:,2),LineWidth=1);

xlabel ('Datetime')

ylabel ('Wavelenght [nm]')

hold on

yyaxis right

plot(Time_epochFBG1,A(:,3),LineWidth=1)

legend('ESC','Battery')

grid on

title ('Test 1')

figure(2)
plot(Time_epochFBG2,B(:,2),LineWidth=1);
```

```
xlabel ('Time')

ylabel ('Wavelenght [nm]')

hold on

yyaxis right

plot(Time_epochFBG2,B(:,3),LineWidth=1)

legend('ESC','Battery')

grid on

title ('Test 2')

figure(3)
plot(Time_epochFBG3,C(:,2),LineWidth=1);

xlabel ('Time')

ylabel ('Wavelenght [nm]')

hold on

yyaxis right

plot(Time_epochFBG3,C(:,3),LineWidth=1)

legend('ESC','Battery')

grid on

title ('Test 3')


% FBG DATA CONVERSION INTO TEMPERATURE

lambda_0_ESC = taratura1530.Coefficienti(2);  % I take the
   known term of the calibration line
k_T_ESC = taratura1530.Coefficienti(1);

lambda_0_BAT = taratura1552.Coefficienti(2);  % I take the
   known term of the calibration line
k_T_BAT = taratura1552.Coefficienti(1);
```

```
T_ESC1 = (A(:,2)-lambda_0_ESC)/k_T_ESC;
T_ESC2 = (B(:,2)-lambda_0_ESC)/k_T_ESC;
T_ESC3 = (C(:,2)-lambda_0_ESC)/k_T_ESC;

T_BAT1 = (A(:,3)-lambda_0_BAT)/k_T_BAT;
T_BAT2 = (B(:,3)-lambda_0_BAT)/k_T_BAT;
T_BAT3 = (C(:,3)-lambda_0_BAT)/k_T_BAT;

% TEMPERATURE GRAPHS

figure(4)
plot(Time_epochFBG1,T_ESC1,LineWidth=1)

xlabel ('Time')

ylabel ('Temperature [ C ]')

hold on

yyaxis right

plot(Time_epochFBG1,T_BAT1,LineWidth=1)

legend('ESC','Battery')

grid on

title ('Test 1')



figure(5)
plot(Time_epochFBG2,T_ESC2,LineWidth=1)

xlabel ('Time')

ylabel ('Temperature [ C ]')

hold on

yyaxis right

plot(Time_epochFBG2,T_BAT2,LineWidth=1)
```

```
legend('ESC','Battery')

grid on

title ('Test 2')


figure(6)
plot(Time_epochFBG3,T_ESC3,LineWidth=1)

xlabel ('Time')

ylabel ('Temperature [ C ]')

hold on

yyaxis right

plot(Time_epochFBG3,T_BAT3,LineWidth=1)

legend('ESC','Battery')

grid on

title ('Test 3')
```

ElaborazioneDati_IMU.m

```
% ---FLIGHT 1---
Folder = "C:\Users\User\Desktop\TestVolo\PhotoNext\
   Volo1_PhotoNext";
[GPS,ALT,IMU,PWR] = Air_Telemetry("C:\Users\User\Desktop\
   TestVolo\PhotoNext\Volo1_PhotoNext");


%%
% ---FLIGHT 2---
Folder = "C:\Users\User\Desktop\TestVolo\Volo2_PhotoNext";
[GPS,ALT,IMU,PWR] = Air_Telemetry("C:\Users\User\Desktop\
   TestVolo\PhotoNext\Volo2_PhotoNext");
```

```
%%
% ---STATIC TEST 1 ---
Folder = "C:\Users\User\Desktop\Test_Statici_Photonext\
   Dati_IMU\LOG_1";
[GPS,ALT,IMU,PWR] = Air_Telemetry("C:\Users\User\Desktop\
   Test_Statici_Photonext\LOG_1");

%%
% ---STATIC TEST 2 ---
Folder = "C:\Users\User\Desktop\Test_Statici_Photonext\
   Dati_IMU\LOG_2";
[GPS,ALT,IMU,PWR] = Air_Telemetry("C:\Users\User\Desktop\
   Test_Statici_Photonext\LOG_2");
```

Air_Telemetry.m

```
function [GPS,ALT,IMU,PWR] = Air_Telemetry(Folder)
%Function to put AirTelemetry data into data structure after
   a minimum
%number of GPS satellites in locked
%Just use this [GPS,ALT,IMU,PWR] = Air_Telemetry("Folder_path
   "); to import data into workspace

%User defined parameters
minSAT = 0;             %6 for the flight test, 0 for the
   static tests
                        %minimum number of satellites to
                           start importing the log
                        %This prevents the plane teleporting
                           from Africa to
                        %the runway when the GPS has a fix

                        %You can vary this parameter if you
                           need other data
                        %before GPS fix


%Calibration parameters
ACC_OFFSET = [0 0 0];
GYRO_OFFSET = [0 0 0];

I_CAL = 91.186./65565;
%I_CAL = 0.002778708152215;
V_BAT_CAL = 1./2091.15;
```

```
V_PV_CAL = 0.008002;
V_PV_OFF = 62.9342;

%Program start

ALT_PATH = strcat(Folder,"/ALT.txt");
GPS_PATH = strcat(Folder,"/GPS.txt");
IMU_PATH = strcat(Folder,"/IMU.txt");
PWR_PATH = strcat(Folder,"/PWR.txt");

%Get data from altimeter file
analysis_file = fopen(ALT_PATH);
FileContents = textscan(analysis_file, '%s', 'delimiter', '\n
   '); % Reads each line of the file and saves it in a cell
StringArray = string(FileContents{:});
                             % Turns each line in the cell in
    a string
j = 1;
for i=1:length(StringArray)
    format = '%c,%u,%f,%f';
    tempstr = StringArray(i);
    tempALT = sscanf(tempstr,format);
    if(length(tempALT) == 4)

        ALT_SD(j,:) = tempALT;
         j = j + 1;
    end
    clear tempALT;
end

%Get data from GPS file
analysis_file = fopen(GPS_PATH);
FileContents = textscan(analysis_file, '%s', 'delimiter', '\n
   '); % Reads each line of the file and saves it in a cell
StringArray = string(FileContents{:});
                             % Turns each line in the cell in
    a string
j = 1;
for i=1:length(StringArray)
    format = '%c,%u,%d:%d:%f,%lf,%lf,%f,%f,%d,%f,%f';
    tempstr = StringArray(i);
    tempGPS = sscanf(tempstr,format);
    if(length(tempGPS) == 12)

        GPS_SD(j,:) = tempGPS;
```

```
            j = j + 1;
        end
        clear tempALT;
    end

    %Import IMU RAW data
    IMU_RAW = fopen(IMU_PATH);
    buf = fread(IMU_RAW);
    buf = cast(buf,'uint8');
    for i=1:(length(buf)/33)
      row = (i*33)-33;
        IMURAW.acc(i,1) = bitshift(cast(buf(row+3),'int16'),8);
        IMURAW.acc(i,1) = IMURAW.acc(i,1) + cast(buf(row+2),'
           int16');

        IMURAW.acc(i,2) = bitshift(cast(buf(row+6),'int16'),8);
        IMURAW.acc(i,2) = IMURAW.acc(i,2) + cast(buf(row+5),'
           int16');

        IMURAW.acc(i,3) = bitshift(cast(buf(row+9),'int16'),8);
        IMURAW.acc(i,3) = IMURAW.acc(i,3) + cast(buf(row+8),'
           int16');

        IMURAW.gyro(i,1) = bitshift(cast(buf(row+12),'int16'),8);
        IMURAW.gyro(i,1) = IMURAW.gyro(i,1) + cast(buf(row+11),'
           int16');

        IMURAW.gyro(i,2) = bitshift(cast(buf(row+15),'int16'),8);
        IMURAW.gyro(i,2) = IMURAW.gyro(i,2) + cast(buf(row+14),'
           int16');

        IMURAW.gyro(i,3) = bitshift(cast(buf(row+18),'int16'),8);
        IMURAW.gyro(i,3) = IMURAW.gyro(i,3) + cast(buf(row+17),'
           int16');

        IMURAW.time(i) = cast(buf(row+20),'int32');
        IMURAW.time(i) = IMURAW.time(i) + bitshift(cast(buf(row
           +21),'int32'),8);
        IMURAW.time(i) = IMURAW.time(i) + bitshift(cast(buf(row
           +22),'int32'),16);
        IMURAW.time(i) = IMURAW.time(i) + bitshift(cast(buf(row
           +23),'int32'),24);

        IMURAW.timestamp(i) = cast(buf(row+25),'uint32');
        IMURAW.timestamp(i) = IMURAW.timestamp(i) + bitshift(cast
            (buf(row+26),'uint32'),8);
        IMURAW.timestamp(i) = IMURAW.timestamp(i) + bitshift(cast
            (buf(row+27),'uint32'),16);
        IMURAW.timestamp(i) = IMURAW.timestamp(i) + bitshift(cast
            (buf(row+28),'uint32'),24);

        IMURAW.temperature(i) = bitshift(cast(buf(row+31),'int16'
            ),8);
        IMURAW.temperature(i) = IMURAW.temperature(i) + cast(buf(
            row+30),'int16');

end

clear buf


%Import Power RAW data
PWR_RAW = fopen(PWR_PATH);
buf = fread(PWR_RAW);
buf = cast(buf,'uint8');
for i=1:(length(buf)/16)
    row = (i*16)-16;
        PWRRAW.V_BAT(i) = bitshift(cast(buf(row+3),'uint16'),8);
        PWRRAW.V_BAT(i) = PWRRAW.V_BAT(i) + cast(buf(row+2),'
            uint16');

        PWRRAW.V_PV(i) = bitshift(cast(buf(row+6),'uint16'),8);
        PWRRAW.V_PV(i) = PWRRAW.V_PV(i) + cast(buf(row+5),'uint16
            ');

        PWRRAW.I(i) = bitshift(cast(buf(row+9),'uint16'),8);
        PWRRAW.I(i) = PWRRAW.I(i) + cast(buf(row+8),'uint16');

        PWRRAW.timestamp(i) = cast(buf(row+11),'uint32');
        PWRRAW.timestamp(i) = PWRRAW.timestamp(i) + bitshift(cast
            (buf(row+12),'uint32'),8);
        PWRRAW.timestamp(i) = PWRRAW.timestamp(i) + bitshift(cast
            (buf(row+13),'uint32'),16);
        PWRRAW.timestamp(i) = PWRRAW.timestamp(i) + bitshift(cast
            (buf(row+14),'uint32'),24);
end


%IMU_SD = importdata(IMU_PATH,',');
%PWR_SD = importdata(PWR_PATH,',');
```

```
i = 1;
while GPS_SD(i,10) < minSAT
    i = i+1;
end

GPS_START = i;
startTimestamp = GPS_SD(i,2);

i = 1;
while ALT_SD(i,2) < startTimestamp
    i = i+1;
end
ALT_START = i;

i = 1;
while PWRRAW.timestamp(i) < startTimestamp
    i = i+1;
end
PWR_START = i;

i = 1;
while IMURAW.timestamp(i) < startTimestamp
    i = i+1;
end
IMU_START = i;

PWR.timestamp = PWRRAW.timestamp(PWR_START:end) -
   startTimestamp;
PWR.V_BAT = cast(PWRRAW.V_BAT(PWR_START:end),'double').*
   V_BAT_CAL;                    %Battery voltage in volts
PWR.V_PV = cast(PWRRAW.V_PV(PWR_START:end),'double').*
   V_PV_CAL;                       %RFU
PWR.V_PV = PWR.V_PV - V_PV_OFF;
PWR.I = cast(PWRRAW.I(PWR_START:end),'double').*I_CAL;
                              %Motor current in amperes

ALT.buffer = ALT_SD(ALT_START:end,1);
                                                 %circular
   buffer ID
ALT.timestamp = ALT_SD(ALT_START:end,2) - startTimestamp;
                           %uC timer in us
ALT.temperature = ALT_SD(ALT_START:end,3);
                                             %altimeter
   temperature in C
```

```
ALT.pressure = ALT_SD(ALT_START:end,4);
                                                   %pressure
   reading in mbar

IMU.timestamp = IMURAW.timestamp(IMU_START:end) -
   startTimestamp;                      %uC timer in us
IMU.time = cast(IMURAW.time(IMU_START:end),'double') .* 25;
                              %IMU timer in us
IMU.acc = (cast(IMURAW.acc(IMU_START:end,:),'double') -
   ACC_OFFSET) .* 0.244;        %acceleration in mg
IMU.gyro = (cast(IMURAW.gyro(IMU_START:end,:),'double') -
   GYRO_OFFSET) .* 17.5;      %angular velocity in mdps
IMU.temperature = (cast(IMURAW.temperature(IMU_START:end),'
   double')./256) + 25;      %IMU temperature in C

IMU.timestamp = cast(IMU.timestamp,'double');


GPS.buffer = GPS_SD(GPS_START:end,1);
                                                     %circular
   buffer ID
GPS.timestamp = GPS_SD(GPS_START:end,2) - startTimestamp;
                                %uC timer in us

%time of day calculation
time.hours = GPS_SD(GPS_START:end,3);
time.minutes = GPS_SD(GPS_START:end,4);
time.seconds = GPS_SD(GPS_START:end,5);
GPS.time = time;                                              %
   GPS time of day

GPS.latitude = GPS_SD(GPS_START:end,6);                       %
   latitude in decimal degrees
GPS.longitude = GPS_SD(GPS_START:end,7);                      %
   longitude in decimal degrees
GPS.heading = GPS_SD(GPS_START:end,8);                        %
   heading in decimal degrees
GPS.speed = GPS_SD(GPS_START:end,9);                          %
   speed in km/h
GPS.satellites = GPS_SD(GPS_START:end,10);                    %
   number of locked satellites
GPS.hdop = GPS_SD(GPS_START:end,11);                          %
   dilution of precision
GPS.altitude = GPS_SD(GPS_START:end,12);                      %
   GPS altitude in m
```

ParametriVolo.m

```
%% GRAPHS FOR ALL IMU PARAMETERS

%% ALT

figure1 = figure();


InitialTime = 2614;              %2614
time = [];

c = 1;

while c <= length(ALT.timestamp)
    diffTime = (ALT.timestamp(c) - InitialTime)/1000000;
    time = [time, diffTime];
    c = c + 1;

end
%%

plot(time,ALT.temperature)

title("Temperature")

xlabel("Time [s]")

ylabel("Temperature [\circC]")

grid on
% saveas(figure1,"TemperaturaALT.fig")
% saveas(figure1,"TemperaturaALT.png")

figure2 = figure();

plot(time,ALT.pressure)

title("Pressure")

xlabel("Time [s]")

ylabel("Pressure [mbar]")
```

```matlab
% saveas(figure2,"PressioneALT.fig")
% saveas(figure2,"PressioneALT.png")
grid on
%% PWR

figure3 = figure();


InitialTime = 1332;
time = [];

c = 1;

while c <= length(PWR.timestamp)
    diffTime = (PWR.timestamp(c) - InitialTime)/1000000;
    time = [time, diffTime];
    c = c + 1;

end

plot(time,PWR.V_BAT)

title("Battery voltage")

xlabel("Time [s]")

ylabel("Voltage [V]")

grid on
% saveas(figure3,"TensioneBatteria.fig")
% saveas(figure3,"TensioneBatteria.png")

figure4 = figure();

plot(time,PWR.I)

title("Motor electrical current")

xlabel("Time [s]")

ylabel("Electrical current [A]")

grid on
% saveas(figure4,"CorrenteMotore.fig")
% saveas(figure4,"CorrenteMotore.png")
```

```
figure5 = figure();

plot(time,PWR.V_PV)

xlabel("Time [s]")

ylabel("Voltage [V]")

% saveas(figure5,"TensioneBoh.fig")
% saveas(figure5,"TensioneBoh.png")
grid on
%% GPS

figure6 = figure();


InitialTime = 0;
time = [];

c = 1;

while c <= length(GPS.timestamp)
    diffTime = (GPS.timestamp(c) - InitialTime)/1000000;
    time = [time, diffTime];
    c = c + 1;

end

plot(time,GPS.latitude)

title("Latitudine")

xlabel("Tempo [s]")

ylabel("Latitudine [decimal degrees]")

grid on
% saveas(figure6,"LatitudineGPS.fig")
% saveas(figure6,"LatitudineGPS.png")


figure7 = figure();

plot(time,GPS.longitude)
```

```
title("Longitudine")

xlabel("Tempo [s]")

ylabel("Longitudine [decimal degrees]")

grid on
% saveas(figure7,"LongitudineGPS.fig")
% saveas(figure7,"LongitudineGPS.png")

figure8 = figure();

plot(time,GPS.heading)

title("Heading")

xlabel("Tempo [s]")

ylabel("Heading [decimal degrees]")

grid on
% saveas(figure8,"HeadingGPS.fig")
% saveas(figure8,"HeadingGPS.png")

figure9 = figure();

plot(time,GPS.speed)

title("Velocit ")

xlabel("Tempo [s]")

ylabel("Velocit  [km/h]")

grid on
% saveas(figure9,"Velocit GPS.fig")
% saveas(figure9,"Velocit GPS.png")

figure10 = figure();

plot(time,GPS.altitude)

title("Altitudine")
```

```
xlabel("Tempo [s]")

ylabel("Altitudine [m]")

grid on
% saveas(figure10,"AltitudineGPS.fig")
% saveas(figure10,"AltitudineGPS.png")

%% IMU

figure11 = figure();


InitialTime = 4691;
time = [];

c = 1;

while c <= length(IMU.timestamp)
    diffTime = (IMU.timestamp(c) - InitialTime)/1000000;
    time = [time, diffTime];
    c = c + 1;

end

plot(time,IMU.temperature)

title("Temperature IMU")

xlabel("Time [s]")

ylabel("Temperature [\circC]")

% saveas(figure11,"TemperaturaIMU.fig")
% saveas(figure11,"TemperaturaIMU.png")

grid on
%%
figure12 = figure();

colonna1 = [];

c = 1;

while c <= length(IMU.acc)
```

```
    colonna1 = [colonna1, IMU.acc(c,1)];
    c = c + 1;
end

colonna2 = [];

c = 1;

while c <= length(IMU.acc)
    colonna2 = [colonna2, IMU.acc(c,2)];
    c = c + 1;
end

colonna3 = [];

c = 1;

while c <= length(IMU.acc)
    colonna3 = [colonna3, IMU.acc(c,3)];
    c = c + 1;
end

plot(time(1:45:end), colonna1(1:45:end))

hold on

plot(time(1:45:end), colonna2(1:45:end))   %-55.4920 volo 2

hold on

plot(time(1:45:end), colonna3(1:45:end))

title("Linear accelerations")

xlabel("Time [s]")

ylabel("Acceleration [mg]")

legend('x axis','y axis','z axis');

grid on
% saveas(figure12,"accIMU.fig")
% saveas(figure12,"accIMU.png")
```

```
%%
figure13 = figure();

colonna11 = [];

c = 1;

while c <= length(IMU.gyro)
    colonna11 = [colonna11, IMU.gyro(c,1)];
    c = c + 1;
end

colonna22 = [];

c = 1;

while c <= length(IMU.gyro)
    colonna22 = [colonna22, IMU.gyro(c,2)];
    c = c + 1;
end

colonna33 = [];

c = 1;

while c <= length(IMU.gyro)
    colonna33 = [colonna33, IMU.gyro(c,3)];
    c = c + 1;
end

plot(time(1:45:end)-55.4920, colonna11(1:45:end))

hold on

plot(time(1:45:end)-55.4920, colonna22(1:45:end))

hold on

plot(time(1:45:end)-55.4920, colonna33(1:45:end))

title("Velocit  angolare")

xlabel("Tempo [s]")

ylabel("Velocit  angolare [mdps]")
```

```
% saveas(figure13,"gyroIMU.fig")
% saveas(figure13,"gyroIMU.png")



%% Wing-acceleration overlap z

opts = delimitedTextImportOptions("NumVariables", 3);

% Specify range and delimiter
opts.DataLines = [2, Inf];
opts.Delimiter = ",";

% Specify column names and types
opts.VariableNames = ["curr_time", "index", "wavelength"];
opts.VariableTypes = ["double", "double", "double"];

% Specify file level properties
opts.ExtraColumnsRule = "ignore";
opts.EmptyLineRule = "read";

% Specify variable properties
opts = setvaropts(opts, ["curr_time", "index", "wavelength"],
    "TrimNonNumeric", true);
opts = setvaropts(opts, ["curr_time", "index", "wavelength"],
    "ThousandsSeparator", ",");

% Import the data
Interrogatorefibre = readtable("C:\Users\User\Desktop\
   TestVolo\PhotoNext\Volo1_PhotoNext", opts);



%%
figure15 = figure();

InitialTime3 = 4691;
colonna3 = [];
time = [];
c = 1;

while c <= length(IMU.acc)
    colonna3 = [colonna3, IMU.acc(c,3)];
    diffTime = (IMU.timestamp(c) - InitialTime3)/1000000;
    time = [time, diffTime];
    c = c + 1;
end

%1689607590422 GPS activation of the flight 2, line 8946

time18 = [];
wavelength18 = [];
c = 8946;
InitialTime = 1689607590422;

while c <= length(Interrogatorefibre.index)
    if Interrogatorefibre.index(c) == 18
        diffTime = (Interrogatorefibre.curr_time(c) -
           InitialTime)/1000;
        time18 = [time18, diffTime];
        wavelength18 = [wavelength18, Interrogatorefibre.
           wavelength(c)];
        c = c + 1;

    else c = c + 1;


    end
end



plot(time(1:45:end), colonna3(1:45:end))

title("Flight test 1")
xlabel("Time [s]")
ylabel("Acceleration [mg]")

yyaxis right

plot(time18, wavelength18)

ylabel("Wavelength [nm]")

grid on
legend('Acceleration z axis','FBG sensor')

% saveas(figure15,"SovrapposizioneAlaAccZ(3).fig")
% saveas(figure15,"SovrapposizioneAlaAccZ(3).png")
```

```
%% Electric current and temperature

figure16 = figure();


InitialTimeT = 2614;
timeT = [];

c = 1;

while c <= length(ALT.timestamp)
    diffTime = (ALT.timestamp(c) - InitialTimeT)/1000000;
    timeT = [timeT, diffTime];
    c = c + 1;

end


InitialTimeI = 1332;
timeI = [];
c = 1;

while c <= length(PWR.timestamp)
    diffTime = (PWR.timestamp(c) - InitialTimeI)/1000000;
    timeI = [timeI, diffTime];
    c = c + 1;

end


plot(timeT,ALT.temperature)

xlabel("Tempo [s]")
ylabel("Temperatura [\circC]")

yyaxis right

plot(timeI,PWR.I)

ylabel("Corrente [A]")

lgd = legend("sensore di temperatura","corrente motore");
```

```
% saveas(figure16,"SovrapposizioneCorrT.fig")
% saveas(figure16,"SovrapposizioneCorrT.png")
```

ElaborazioneDati_FBG.m

```
%% FBG DATA IMPORT

%% Set up the Import Options and import the data
opts = delimitedTextImportOptions("NumVariables", 3);

% Specify range and delimiter
opts.DataLines = [2, Inf];
opts.Delimiter = ",";

% Specify column names and types
opts.VariableNames = ["curr_time", "index", "wavelength"];
opts.VariableTypes = ["double", "double", "double"];

% Specify file level properties
opts.ExtraColumnsRule = "ignore";
opts.EmptyLineRule = "read";

% Specify variable properties
opts = setvaropts(opts, ["curr_time", "index", "wavelength"],
    "TrimNonNumeric", true);
opts = setvaropts(opts, ["curr_time", "index", "wavelength"],
    "ThousandsSeparator", ",");

% Import the data
Interrogatorefibre = readtable("C:\Users\User\Desktop\Volo
   luglio 2023\volo2.txt", opts);

%Interrogatorefibre = importdata("C:\Users\User\Desktop\
   TestVolo\volo1.txt");


%% !!! For the flight test 2, select the file "volo2.txt"
   from the folder TestVolo !!!



%% Clear temporary variables
clear opts
```

```
%% Plotting standardized ensemble wing sensors

figure2 = figure();

InitialTime =  1689603317840;  %1699270113.870225;

%1689603317840; %timestamp number 1800 (I remove the first
   1300s where nothing happens) for the flight 1
%1689607497808; %timestamp number 560 (I remove the first 400
   s where nothing happens) for the flight 2

time0 = [];
wavelength0 = [];
Wavelength0 = [];

time1_FBG = [];
wavelength1 = [];
Wavelength1 = [];

time2 = [];
wavelength2 = [];
Wavelength2 = [];

time16 = [];
wavelength16 = [];
Wavelength16 = [];

time17 = [];
wavelength17 = [];
Wavelength17 = [];

time18 = [];
wavelength18 = [];
Wavelength18 = [];

time19 = [];
wavelength19 = [];
Wavelength19 = [];

time20 = [];
wavelength20 = [];
Wavelength20 = [];
```

```
time21 = [];
wavelength21 = [];
Wavelength21 = [];

time22 = [];
wavelength22 = [];
Wavelength22 = [];

time23 = [];
wavelength23 = [];
Wavelength23 = [];

c = 1;

while c <= length(Interrogatorefibre.index)
    if Interrogatorefibre.index(c) == 0
        if Interrogatorefibre.curr_time(c) - InitialTime >= 0
            diffTime = (Interrogatorefibre.curr_time(c) -
               InitialTime)/1000;
            time0 = [time0, diffTime];
            wavelength0 = [wavelength0, Interrogatorefibre.
               wavelength(c)];
            Wavelength0 = wavelength0/Interrogatorefibre.
               wavelength(1);
            c = c + 1;
        else c = c + 1;

        end

    elseif Interrogatorefibre.index(c) == 1
        if Interrogatorefibre.curr_time(c) - InitialTime >= 0
            diffTime = (Interrogatorefibre.curr_time(c) -
               InitialTime)/1000;
            time1_FBG = [time1_FBG, diffTime];
            wavelength1 = [wavelength1, Interrogatorefibre.
               wavelength(c)];
            Wavelength1 = wavelength1/Interrogatorefibre.
               wavelength(2);
            c = c + 1;
        else c = c + 1;

        end

    elseif Interrogatorefibre.index(c) == 2
        if Interrogatorefibre.curr_time(c) - InitialTime >= 0
```

```
            diffTime = (Interrogatorefibre.curr_time(c) -
               InitialTime)/1000;
            time2 = [time2, diffTime];
            wavelength2 = [wavelength2, Interrogatorefibre.
               wavelength(c)];
            Wavelength2 = wavelength2/Interrogatorefibre.
               wavelength(3);
            c = c + 1;
        else c = c + 1;

        end

    elseif Interrogatorefibre.index(c) == 16
        if Interrogatorefibre.curr_time(c) - InitialTime >= 0
            diffTime = (Interrogatorefibre.curr_time(c) -
               InitialTime)/1000;
            time16 = [time16, diffTime];
            %time16 = [time16, Interrogatorefibre.curr_time(c
               )];
            %% Comment on the previous line and select the
               latter if you want real time
            wavelength16 = [wavelength16, Interrogatorefibre.
               wavelength(c)];
            Wavelength16 = wavelength16/Interrogatorefibre.
               wavelength(4);
            c = c + 1;
        else c = c + 1;

        end

    elseif Interrogatorefibre.index(c) == 17
        if Interrogatorefibre.curr_time(c) - InitialTime >= 0
            diffTime = (Interrogatorefibre.curr_time(c) -
               InitialTime)/1000;
            time17 = [time17, diffTime];
            wavelength17 = [wavelength17, Interrogatorefibre.
               wavelength(c)];
            Wavelength17 = wavelength17/Interrogatorefibre.
               wavelength(5);
            c = c + 1;
        else c = c + 1;

        end
```

```
elseif Interrogatorefibre.index(c) == 18
    if Interrogatorefibre.curr_time(c) - InitialTime >= 0
        diffTime = (Interrogatorefibre.curr_time(c) -
           InitialTime)/1000;
        time18 = [time18, diffTime];
        %time18 = [time18, Interrogatorefibre.curr_time(c
           )];
        wavelength18 = [wavelength18, Interrogatorefibre.
           wavelength(c)];
        Wavelength18 = wavelength18/Interrogatorefibre.
           wavelength(6);
        c = c + 1;
    else c = c + 1;

    end

elseif Interrogatorefibre.index(c) == 19
    if Interrogatorefibre.curr_time(c) - InitialTime >= 0
        diffTime = (Interrogatorefibre.curr_time(c) -
           InitialTime)/1000;
        time19 = [time19, diffTime];
        wavelength19 = [wavelength19, Interrogatorefibre.
           wavelength(c)];
        Wavelength19 = wavelength19/Interrogatorefibre.
           wavelength(7);
        c = c + 1;
    else c = c + 1;

    end

elseif Interrogatorefibre.index(c) == 20
    if Interrogatorefibre.curr_time(c) - InitialTime >= 0
        diffTime = (Interrogatorefibre.curr_time(c) -
           InitialTime)/1000;
        time20 = [time20, diffTime];
        wavelength20 = [wavelength20, Interrogatorefibre.
           wavelength(c)];
        Wavelength20 = wavelength20/Interrogatorefibre.
           wavelength(8);
        c = c + 1;
    else c = c + 1;

    end

elseif Interrogatorefibre.index(c) == 21
```

```
        if Interrogatorefibre.curr_time(c) - InitialTime >= 0
            diffTime = (Interrogatorefibre.curr_time(c) -
               InitialTime)/1000;
            time21 = [time21, diffTime];
            wavelength21 = [wavelength21, Interrogatorefibre.
               wavelength(c)];
            Wavelength21 = wavelength21/Interrogatorefibre.
               wavelength(9);
            c = c + 1;
        else c = c + 1;

        end

    elseif Interrogatorefibre.index(c) == 22
        if Interrogatorefibre.curr_time(c) - InitialTime >= 0
            diffTime = (Interrogatorefibre.curr_time(c) -
               InitialTime)/1000;
            time22 = [time22, diffTime];
            wavelength22 = [wavelength22, Interrogatorefibre.
               wavelength(c)];
            Wavelength22 = wavelength22/Interrogatorefibre.
               wavelength(10);
            c = c + 1;
        else c = c + 1;

        end

    elseif Interrogatorefibre.index(c) == 23
        if Interrogatorefibre.curr_time(c) - InitialTime >= 0
            diffTime = (Interrogatorefibre.curr_time(c) -
               InitialTime)/1000;
            time23 = [time23, diffTime];
            wavelength23 = [wavelength23, Interrogatorefibre.
               wavelength(c)];
            Wavelength23 = wavelength23/Interrogatorefibre.
               wavelength(11);
            c = c + 1;
        else c = c + 1;

        end

    else c = c + 1;


    end
```

```
end

%% PLOTTING DATA

plot(time0,Wavelength0)

hold on

plot(time1_FBG,Wavelength1)

hold on

plot(time2,Wavelength2)

hold on

plot(time16,Wavelength16)

hold on

plot(time17,Wavelength17)

hold on

plot(time18,Wavelength18)

hold on

plot(time19,Wavelength19)

hold on

plot(time20,Wavelength20)

hold on

plot(time21,Wavelength21)

hold on

plot(time22,Wavelength22)

hold on

plot(time23,Wavelength23)
```

```
title("FBG sensors")

xlabel("Time [s]")

ylabel("\lambda_i/\lambda_1")

grid on

lgd = legend("sensor0","sensor1","sensor2","sensor16","
   sensor17","sensor18","sensor19","sensor20","sensor21","
   sensor22","sensor23");
lgd.Location = "southwest";
lgd.NumColumns = 2;
lgd.FontSize = 6;

saveas(figure2,"SensoriAla2.fig")
saveas(figure2,"SensoreAla2.png")
```

ImportDataSL.m

```
%% Collecting IMU acceleration data to be imported in
   Simulink

Acc_1 =  [IMU.timestamp'*(1e-6) IMU.acc(:,1)]; %input
   variable of Simulink

Acc_2 =  [IMU.timestamp'*(1e-6) IMU.acc(:,2)];

Acc_3 =  [IMU.timestamp'*(1e-6) IMU.acc(:,3)];

%%  FLIGHT 1

clc

close all

load('Volo_1.mat')  % LOADING IMU AND FBG DATA SAVED INTO
   THIS VARIABLE .mat TO ACCELERATE THE PROCEDURE

%% TIME CONVERTION INTO THE REAL DATA IN WHICH TEST TOOK
```

```
   PLACE

T_out = datetime(out.tout*1e3-2000 + InitialTimeGPS,'
   ConvertFrom','epochtime','Epoch','1970-01-01','
   TicksPerSecond',1000);  !! IMU GPS TIME !!

%InitialTimeGPS for the flight 2

T18 = datetime(time18,'ConvertFrom','epochtime','Epoch','
   1970-01-01','TicksPerSecond',1000);  % !! I CONSIDERED THE
    SENSOR 18  BUT IT CAN BE ANY OTHER !!

%plot(T1,IMU.acc(:,3),T_out,out.yout)

%% out.tout and out.yout are the variables coming from
   Simulink filter and saved in Volo_1.mat

%%% THESE FOLLOWING 4 LINES ARE ONLY USED TO REDUCE THE FOCUS
    ON THE MAIN OSCILLATIONS

% T_out(1:3000) = [];
% out.yout(1:3000) = [];
%
% T_out(2500:end) = [];
% out.yout(2500:end) = [];

%%%%%%%%%%%%%

%% PLOTTING FILTERED DATA AFTER THE USE OF THE LOW-PASS
   FILTER

figure(1)

plot(T_out,out.yout) %only filtered data

xlabel('Date')

ylabel('Acceleration z [mg]')

hold on

yyaxis right

%%% THESE FOLLOWING 4 LINES ARE ONLY USED TO REDUCE THE FOCUS
    ON THE MAIN OSCILLATIONS
```

```
% T18(1:830) = [];
% wavelength18(1:830) = [];
%
% T18(315:end) = [];
% wavelength18(315:end) = [];

%%%%%%%%%%


plot(T18,wavelength18)

legend('IMU data','IMU data filtered','FBG data')

acc_z_filtr = out.yout;

ylabel('\lambda [nm]','Color','k')

grid on

%%   FLIGHT  2

% clc
%
% clear

close all

load('Volo_2.mat') % LOADING IMU AND FBG DATA SAVED INTO THIS
    VARIABLE .mat TO ACCELERATE THE PROCEDURE

%% TIME CONVERTION INTO THE REAL DATA IN WHICH TEST TOOK
   PLACE

T_out = datetime(out.tout*1e3 + InitialTimeGPS,'ConvertFrom',
   'epochtime','Epoch','1970-01-01','TicksPerSecond',1000);
   !! IMU GPS TIME !!

T18 = datetime(time18,'ConvertFrom','epochtime','Epoch','
   1970-01-01','TicksPerSecond',1000);  !! I CONSIDERED THE
   SENSOR 18  BUT IT CAN BE ANY OTHER !!

% plot(T1,IMU.acc(:,3),T_out,out.yout,LineWidth=1.2) %
   originale
```

```
%%% THESE FOLLOWING 4 LINES ARE ONLY USED TO REDUCE THE FOCUS
    ON THE MAIN OSCILLATIONS

% T_out(1:3100) = [];
% out.yout(1:3100) = [];
%
% T_out(250:end) = [];
% out.yout(250:end) = [];
%%%%

figure(1)

plot(T_out,out.yout) %ONLY FILTERED DATA

ylabel('Accelerazione lungo z [mg]')

hold on

yyaxis right

%%% THESE FOLLOWING 4 LINES ARE ONLY USED TO REDUCE THE FOCUS
    ON THE MAIN OSCILLATIONS

% T18(1:540) = [];
% wavelength18(1:540) = [];
%
% T18(35:end) = [];
% wavelength18(35:end) = [];

%%%%%%

plot(T18,wavelength18)

%legend('Dato IMU','Dato IMU filtrato','Dato FBG')
legend('Dato IMU filtrato','Dato FBG')

acc_z_filtr = out.yout;

ylabel('\lambda [nm]')



%%   FLIGHT 1

%% In the following lines, I make the two vectors (of IMU
   acceleration z and the wavelength from sensor 18) the same
    size with the function 'interp1' so that I can select
   them in the CurveFitter to calculate the R factor

close all

v = wavelength18;

%v = acc_z_filtr;

xq = linspace(14,300,2499); %% THE INTERVAL DEPENDS ON THE
   VECTOR LENGTHS AND THE R-FACTOR OBTAINED AFTER THE
   CALCULATION ON CURVEFITTER

lambda_acc_z = interp1(v,xq,'linear','extrap');

%IMU_ridotto = interp1(v,xq,'linear','extrap');

figure(2)

plot(T_out,acc_z_filtr)

xlabel('Time [s]')

ylabel('Acceleration [mg]')

hold on

yyaxis right

plot(T_out,lambda_acc_z)

legend('Dato IMU filtrato','Dato FBG')

ylabel('Wavelength [nm]')

grid on


%%  FLIGHT 2

%% In the following lines, I make the two vectors (of IMU
   acceleration z and the wavelength from sensor 18) the same
    size with the function 'interp1' so that I can select
   them in the CurveFitter to calculate the R factor
```

```
close all

v = wavelength18;

%v = acc_z_filtr;

xq = linspace(1.35,34.55,249);  %% THE INTERVAL DEPENDS ON
   THE VECTOR LENGTHS AND THE R-FACTOR OBTAINED AFTER THE
   CALCULATION ON CURVEFITTER

lambda_acc_z = interp1(v,xq,'linear','extrap');

%IMU_ridotto = interp1(v,xq,'linear','extrap');

figure(2)

plot(T_out,acc_z_filtr)

xlabel('Time [s]')

ylabel('Acceleration [mg]')

hold on

yyaxis right

plot(T_out,lambda_acc_z)

legend('Dato IMU filtrato','Dato FBG')

ylabel('Wavelength [nm]')

grid on
```

TestStatico_1.mat

```
%% Import data from text file
% Script for importing data from the following text file:
%
%    filename: C:\Users\User\Desktop\Test_Statici_Photonext\
   Anubi\Anubi_1.txt
%
% Auto-generated by MATLAB on 06-Nov-2023 17:37:51
```

```
%% Set up the Import Options and import the data
opts = delimitedTextImportOptions("NumVariables", 61);

% Specify range and delimiter
opts.DataLines = [1, Inf];
opts.Delimiter = "\t";

% Specify column names and types
opts.VariableNames = ["VarName1", "VarName2", "Var3", "Var4",
    "Var5", "Var6", "Var7", "Var8", "Var9", "Var10", "Var11",
    "Var12", "Var13", "Var14", "Var15", "Var16", "VarName17",
    "VarName18", "VarName19", "Var20", "Var21", "Var22", "
   Var23", "Var24", "Var25", "Var26", "Var27", "Var28", "
   Var29", "Var30", "Var31", "VarName32", "VarName33", "
   VarName34", "VarName35", "VarName36", "VarName37", "
   VarName38", "VarName39", "Var40", "Var41", "Var42", "Var43
   ", "Var44", "Var45", "Var46", "VarName47", "Var48", "Var49
   ", "Var50", "Var51", "Var52", "Var53", "Var54", "Var55", "
   Var56", "Var57", "Var58", "Var59", "Var60", "Var61"];
opts.SelectedVariableNames = ["VarName1", "VarName2", "
   VarName17", "VarName18", "VarName19", "VarName32", "
   VarName33", "VarName34", "VarName35", "VarName36", "
   VarName37", "VarName38", "VarName39", "VarName47"];
opts.VariableTypes = ["double", "double", "string", "string",
    "string", "string", "string", "string", "string", "string
   ", "string", "string", "string", "string", "string", "
   string", "double", "double", "double", "string", "string",
    "string", "string", "string", "string", "string", "string
   ", "string", "string", "string", "string", "double", "
   double", "double", "double", "double", "double", "double",
    "double", "string", "string", "string", "string", "string
   ", "string", "string", "double", "string", "string", "
   string", "string", "string", "string", "string", "string",
    "string", "string", "string", "string", "string", "string
   "];

% Specify file level properties
opts.ExtraColumnsRule = "ignore";
opts.EmptyLineRule = "read";

% Specify variable properties
opts = setvaropts(opts, ["Var3", "Var4", "Var5", "Var6", "
   Var7", "Var8", "Var9", "Var10", "Var11", "Var12", "Var13",
    "Var14", "Var15", "Var16", "Var20", "Var21", "Var22", "
   Var23", "Var24", "Var25", "Var26", "Var27", "Var28", "
   Var29", "Var30", "Var31", "Var40", "Var41", "Var42", "
   Var43", "Var44", "Var45", "Var46", "Var48", "Var49", "
   Var50", "Var51", "Var52", "Var53", "Var54", "Var55", "
   Var56", "Var57", "Var58", "Var59", "Var60", "Var61"], "
   WhitespaceRule", "preserve");
opts = setvaropts(opts, ["Var3", "Var4", "Var5", "Var6", "
   Var7", "Var8", "Var9", "Var10", "Var11", "Var12", "Var13",
    "Var14", "Var15", "Var16", "Var20", "Var21", "Var22", "
   Var23", "Var24", "Var25", "Var26", "Var27", "Var28", "
   Var29", "Var30", "Var31", "Var40", "Var41", "Var42", "
   Var43", "Var44", "Var45", "Var46", "Var48", "Var49", "
   Var50", "Var51", "Var52", "Var53", "Var54", "Var55", "
   Var56", "Var57", "Var58", "Var59", "Var60", "Var61"], "
   EmptyFieldRule", "auto");
opts = setvaropts(opts, ["VarName1", "VarName2", "VarName17",
    "VarName18", "VarName19", "VarName32", "VarName33", "
   VarName34", "VarName35", "VarName36", "VarName37", "
   VarName38", "VarName39", "VarName47"], "TrimNonNumeric",
   true);
opts = setvaropts(opts, ["VarName1", "VarName2", "VarName17",
    "VarName18", "VarName19", "VarName32", "VarName33", "
   VarName34", "VarName35", "VarName36", "VarName37", "
   VarName38", "VarName39", "VarName47"], "DecimalSeparator",
    ",");
opts = setvaropts(opts, ["VarName1", "VarName2", "VarName17",
    "VarName18", "VarName19", "VarName32", "VarName33", "
   VarName34", "VarName35", "VarName36", "VarName37", "
   VarName38", "VarName39", "VarName47"], "ThousandsSeparator
   ", ".");

% Import the data
Anubi1 = readtable("C:\Users\User\Desktop\
   Test_Statici_Photonext\Anubi\Anubi_1.txt", opts);


%% Clear temporary variables
clear opts


%% Plots


figure1 = figure();
```

```
time = Anubi1(:,1);

sensore_temp = Anubi1(:,2);

%plot(time,sensore_temp,'r',Linewidth=0.5)

title('Temperatura FBG')

xlabel('Tempo [s]')

ylabel("Lunghezza d'onda \lambda [nm]")

%saveas(figure1,"Sensore_temp.fig")

%%
figure2 = figure();

time = Anubi1(:,1);

sensore_1 = Anubi1(:,3);

plot(time,sensore_1,'r',Linewidth=0.5)

title('Sensor 1')

xlabel('Time [s]')

ylabel("Wavelength \lambda [nm]")

%saveas(figure2,"Sensore_1.fig")

grid on
%%

figure3 = figure();

time = Anubi1(:,1);

sensore_2 = Anubi1(:,4);

%plot(time,sensore_2,'r',Linewidth=0.5)

title('Sensore 2')

xlabel('Tempo [s]')
```

```
ylabel("Lunghezza d'onda \lambda [nm]")

%saveas(figure3,"Sensore_2.fig")


figure4 = figure();

time = Anubi1(:,1);

sensore_3 = Anubi1(:,5);

%plot(time,sensore_3,'r',Linewidth=0.5)

title('Sensore 3')

xlabel('Tempo [s]')

ylabel("Lunghezza d'onda \lambda [nm]")

%saveas(figure4,"Sensore_3.fig")


figure5 = figure();

time = Anubi1(:,1);

sensore_4 = Anubi1(:,6);

%plot(time,sensore_4,'r',Linewidth=0.5)

title('Sensore 4')

xlabel('Tempo [s]')

ylabel("Lunghezza d'onda \lambda [nm]")

%saveas(figure5,"Sensore_4.fig")


figure6 = figure();

time = Anubi1(:,1);

sensore_5 = Anubi1(:,7);
```

```
%plot(time,sensore_5,'r',Linewidth=0.5)
title('Sensore 5')
xlabel('Tempo [s]')
ylabel("Lunghezza d'onda \lambda [nm]")
%saveas(figure6,"Sensore_5.fig")

figure7 = figure();
time = Anubi1(:,1);
sensore_6 = Anubi1(:,8);
%plot(time,sensore_6,'r',Linewidth=0.5)
title('Sensore 6')
xlabel('Tempo [s]')
ylabel("Lunghezza d'onda \lambda [nm]")
%saveas(figure7,"Sensore_6.fig")

figure8 = figure();
time = Anubi1(:,1);
sensore_7 = Anubi1(:,9);
%plot(time,sensore_7,'r',Linewidth=0.5)
title('Sensore 7')
xlabel('Tempo [s]')
ylabel("Lunghezza d'onda \lambda [nm]")
%saveas(figure8,"Sensore_7.fig")
```

```
figure9 = figure();

time = Anubi1(:,1);

sensore_8 = Anubi1(:,10);

%plot(time,sensore_8,'r',Linewidth=0.5)

title('Sensore 8')

xlabel('Tempo [s]')

ylabel("Lunghezza d'onda \lambda [nm]")

%saveas(figure9,"Sensore_8.fig")


figure10 = figure();

time = Anubi1(:,1);

sensore_9 = Anubi1(:,11);

%plot(time,sensore_9,'r',Linewidth=0.5)

title('Sensore 9')

xlabel('Tempo [s]')

ylabel("Lunghezza d'onda \lambda [nm]")

%saveas(figure10,"Sensore_9.fig")


figure11 = figure();

time = Anubi1(:,1);

sensore_10 = Anubi1(:,12);

%plot(time,sensore_10,'r',Linewidth=0.5)
```

```
title('Sensore 10')

xlabel('Tempo [s]')

ylabel("Lunghezza d'onda \lambda [nm]")

%saveas(figure11,"Sensore_10.fig")


figure12 = figure();

time = Anubi1(:,1);

sensore_11 = Anubi1(:,13);

%plot(time,sensore_11,'r',Linewidth=0.5)

title('Sensore 11')

xlabel('Tempo [s]')

ylabel("Lunghezza d'onda \lambda [nm]")

%saveas(figure12,"Sensore_11.fig")


figure13 = figure();

time = Anubi1(:,1);

sensore_12 = Anubi1(:,14);

%plot(time,sensore_12,'r',Linewidth=0.5)

title('Sensore 12')

xlabel('Tempo [s]')

ylabel("Lunghezza d'onda \lambda [nm]")

%saveas(figure13,"Sensore_12.fig")

%% I CREATE THE INPUT VARIABLE OF THE SIMULINK SCHEME
```

```
clc
Signal = [time sensore_1];
% Signal = [IMU.timestamp'*1e-6 IMU.acc(:,3)];
```

TestStatico_2.mat

```
%% Import data from text file
% Script for importing data from the following text file:
%
%    filename: C:\Users\User\Desktop\Test_Statici_Photonext\
   Anubi\Anubi_2.txt
%
% Auto-generated by MATLAB on 07-Nov-2023 10:10:33

%% Set up the Import Options and import the data
opts = delimitedTextImportOptions("NumVariables", 61);

% Specify range and delimiter
opts.DataLines = [1, Inf];
opts.Delimiter = "\t";

% Specify column names and types
opts.VariableNames = ["VarName1", "VarName2", "Var3", "Var4",
    "Var5", "Var6", "Var7", "Var8", "Var9", "Var10", "Var11",
    "Var12", "Var13", "Var14", "Var15", "Var16", "VarName17",
    "VarName18", "VarName19", "Var20", "Var21", "Var22", "
   Var23", "Var24", "Var25", "Var26", "Var27", "Var28", "
   Var29", "Var30", "Var31", "VarName32", "VarName33", "
   VarName34", "VarName35", "VarName36", "VarName37", "
   VarName38", "VarName39", "Var40", "Var41", "Var42", "Var43
   ", "Var44", "Var45", "Var46", "VarName47", "Var48", "Var49
   ", "Var50", "Var51", "Var52", "Var53", "Var54", "Var55", "
   Var56", "Var57", "Var58", "Var59", "Var60", "Var61"];
opts.SelectedVariableNames = ["VarName1", "VarName2", "
   VarName17", "VarName18", "VarName19", "VarName32", "
   VarName33", "VarName34", "VarName35", "VarName36", "
   VarName37", "VarName38", "VarName39", "VarName47"];
opts.VariableTypes = ["double", "double", "string", "string",
    "string", "string", "string", "string", "string", "string
   ", "string", "string", "string", "string", "string", "
   string", "double", "double", "double", "string", "string",
    "string", "string", "string", "string", "string", "string
   ", "string", "string", "string", "string", "double", "
   double", "double", "double", "double", "double", "double",
    "double", "string", "string", "string", "string", "string
   ", "string", "string", "double", "string", "string", "
   string", "string", "string", "string", "string", "string",
    "string", "string", "string", "string", "string", "string
   "];

% Specify file level properties
opts.ExtraColumnsRule = "ignore";
opts.EmptyLineRule = "read";

% Specify variable properties
opts = setvaropts(opts, ["Var3", "Var4", "Var5", "Var6", "
   Var7", "Var8", "Var9", "Var10", "Var11", "Var12", "Var13",
    "Var14", "Var15", "Var16", "Var20", "Var21", "Var22", "
   Var23", "Var24", "Var25", "Var26", "Var27", "Var28", "
   Var29", "Var30", "Var31", "Var40", "Var41", "Var42", "
   Var43", "Var44", "Var45", "Var46", "Var48", "Var49", "
   Var50", "Var51", "Var52", "Var53", "Var54", "Var55", "
   Var56", "Var57", "Var58", "Var59", "Var60", "Var61"], "
   WhitespaceRule", "preserve");
opts = setvaropts(opts, ["Var3", "Var4", "Var5", "Var6", "
   Var7", "Var8", "Var9", "Var10", "Var11", "Var12", "Var13",
    "Var14", "Var15", "Var16", "Var20", "Var21", "Var22", "
   Var23", "Var24", "Var25", "Var26", "Var27", "Var28", "
   Var29", "Var30", "Var31", "Var40", "Var41", "Var42", "
   Var43", "Var44", "Var45", "Var46", "Var48", "Var49", "
   Var50", "Var51", "Var52", "Var53", "Var54", "Var55", "
   Var56", "Var57", "Var58", "Var59", "Var60", "Var61"], "
   EmptyFieldRule", "auto");
opts = setvaropts(opts, ["VarName1", "VarName2", "VarName17",
    "VarName18", "VarName19", "VarName32", "VarName33", "
   VarName34", "VarName35", "VarName36", "VarName37", "
   VarName38", "VarName39", "VarName47"], "TrimNonNumeric",
   true);
opts = setvaropts(opts, ["VarName1", "VarName2", "VarName17",
    "VarName18", "VarName19", "VarName32", "VarName33", "
   VarName34", "VarName35", "VarName36", "VarName37", "
   VarName38", "VarName39", "VarName47"], "DecimalSeparator",
    ",");
opts = setvaropts(opts, ["VarName1", "VarName2", "VarName17",
    "VarName18", "VarName19", "VarName32", "VarName33", "
   VarName34", "VarName35", "VarName36", "VarName37", "
   VarName38", "VarName39", "VarName47"], "ThousandsSeparator
   ", ".");
```

```
% Import the data
Anubi2 = readtable("C:\Users\User\Desktop\
   Test_Statici_Photonext\Anubi\Anubi_2.txt", opts);


%% Clear temporary variables
clear opts

%% Plots

figure1 = figure();

time = Anubi2(:,1);

sensore_temp = Anubi2(:,2);

%plot(time,sensore_temp,'r',Linewidth=0.5)

title('Temperatura FBG')

xlabel('Tempo [s]')

ylabel("Lunghezza d'onda \lambda [nm]")

%saveas(figure1,"Sensore_temp.fig")

%%
figure2 = figure();

time = Anubi2(:,1);

sensore_1 = Anubi2(:,3);

plot(time,sensore_1,'r',Linewidth=0.5)

title('Sensor 1')

xlabel('Time [s]')

ylabel("Wavelength \lambda [nm]")

%saveas(figure2,"Sensore_1.fig")

grid on
```

```
%%
figure3 = figure();

time = Anubi2(:,1);

sensore_2 = Anubi2(:,4);

%plot(time,sensore_2,'r',Linewidth=0.5)

title('Sensore 2')

xlabel('Tempo [s]')

ylabel("Lunghezza d'onda \lambda [nm]")

%saveas(figure3,"Sensore_2.fig")


figure4 = figure();

time = Anubi2(:,1);

sensore_3 = Anubi2(:,5);

%plot(time,sensore_3,'r',Linewidth=0.5)

title('Sensore 3')

xlabel('Tempo [s]')

ylabel("Lunghezza d'onda \lambda [nm]")

%saveas(figure4,"Sensore_3.fig")


figure5 = figure();

time = Anubi2(:,1);

sensore_4 = Anubi2(:,6);

%plot(time,sensore_4,'r',Linewidth=0.5)

title('Sensore 4')
```

```
xlabel('Tempo [s]')

ylabel("Lunghezza d'onda \lambda [nm]")

%saveas(figure5,"Sensore_4.fig")


figure6 = figure();

time = Anubi2(:,1);

sensore_5 = Anubi2(:,7);

%plot(time,sensore_5,'r',Linewidth=0.5)

title('Sensore 5')

xlabel('Tempo [s]')

ylabel("Lunghezza d'onda \lambda [nm]")

%saveas(figure6,"Sensore_5.fig")


figure7 = figure();

time = Anubi2(:,1);

sensore_6 = Anubi2(:,8);

%plot(time,sensore_6,'r',Linewidth=0.5)

title('Sensore 6')

xlabel('Tempo [s]')

ylabel("Lunghezza d'onda \lambda [nm]")

%saveas(figure7,"Sensore_6.fig")


figure8 = figure();

time = Anubi2(:,1);
```

```
sensore_7 = Anubi2(:,9);

%plot(time,sensore_7,'r',Linewidth=0.5)

title('Sensore 7')

xlabel('Tempo [s]')

ylabel("Lunghezza d'onda \lambda [nm]")

%saveas(figure8,"Sensore_7.fig")


figure9 = figure();

time = Anubi2(:,1);

sensore_8 = Anubi2(:,10);

%plot(time,sensore_8,'r',Linewidth=0.5)

title('Sensore 8')

xlabel('Tempo [s]')

ylabel("Lunghezza d'onda \lambda [nm]")

%saveas(figure9,"Sensore_8.fig")



figure10 = figure();

time = Anubi2(:,1);

sensore_9 = Anubi2(:,11);

%plot(time,sensore_9,'r',Linewidth=0.5)

title('Sensore 9')

xlabel('Tempo [s]')

ylabel("Lunghezza d'onda \lambda [nm]")
```

```
%saveas(figure10,"Sensore_9.fig")


figure11 = figure();

time = Anubi2(:,1);

sensore_10 = Anubi2(:,12);

%plot(time,sensore_10,'r',Linewidth=0.5)

title('Sensore 10')

xlabel('Tempo [s]')

ylabel("Lunghezza d'onda \lambda [nm]")

%saveas(figure11,"Sensore_10.fig")


figure12 = figure();

time = Anubi2(:,1);

sensore_11 = Anubi2(:,13);

%plot(time,sensore_11,'r',Linewidth=0.5)

title('Sensore 11')

xlabel('Tempo [s]')

ylabel("Lunghezza d'onda \lambda [nm]")

%saveas(figure12,"Sensore_11.fig")


figure13 = figure();

time = Anubi2(:,1);

sensore_12 = Anubi2(:,14);
```

```
%plot(time,sensore_12,'r',Linewidth=0.5)

title('Sensore 12')

xlabel('Tempo [s]')

ylabel("Lunghezza d'onda \lambda [nm]")

%saveas(figure13,"Sensore_12.fig")

%% I CREATE THE INPUT VARIABLE OF THE SIMULINK SCHEME
clc
Signal = [time sensore_1];
%Segnale = [IMU.timestamp'*1e-6 IMU.acc(:,3)];
```

Bibliography

[1] Fibre optic cables, drishtiias.com.

[2] Alessandro Aimasso. *Study and analysis of FBG sensors for aerospace applications.* PhD thesis, Politecnico di Torino, 2020.

[3] Alessandro Aimasso. Optical fiber sensor fusion for aerospace systems lifecycle management. Materials Research Forum LLC, 2023.

[4] Alessandro Aimasso, Matteo DL Dalla Vedova, and Paolo Maggiore. Analysis of fbg sensors performances when integrated using different methods for health and structural monitoring in aerospace applications. In *2022 6th International Conference on System Reliability and Safety (ICSRS)*, pages 138–144. IEEE, 2022.

[5] Alessandro Aimasso, Carlo Giovanni Ferro, Matteo Bertone, Matteo DL Dalla Vedova, and Paolo Maggiore. Fiber bragg grating sensor networks enhance the in situ real-time monitoring capabilities of mli thermal blankets for space applications. *Micromachines*, 14(5):926, 2023.

[6] Alessio Carlucci. *Effects of boundary conditions on FBG thermal calibration for aerospace applications.* PhD thesis, Politecnico di Torino, 2022.

[7] Dongdong Chen, Linsheng Huo, Hongnan Li, and Gangbing Song. A fiber bragg grating (fbg)-enabled smart washer for bolt pre-load measurement: design, analysis, calibration, and experimental validation. *Sensors*, 18(8):2586, 2018.

[8] R Delmdahl and K Buchwald. Optics fabrication: Fiber bragg grating fabrication system is automated. *Laser Focus World*, 2017.

[9] Carlo Giovanni Ferro, Alessandro Aimasso, Matteo Bertone, Nicolò Sanzo, Matteo Davide Lorenzo Dalla Vedova, and Paolo Maggiore. Experimental development and evaluation of a fiber bragg grating-based outside air temperature sensor for aircraft applications. *Transportation Engineering*, 14:100214, 2023.

[10] Mohamed Mabrouk. *BRAGG GRATING AND FABRY-PEROT FILTERS IN DATA TRANSMISSION BY LASER BEAMS.* PhD thesis, 07 2009.

[11] Antonio Costantino Marceddu, Alessandro Aimasso, Antonio Scaldaferri, Paolo Maggiore, Bartolomeo Montrucchio, and Matteo DL Dalla Vedova. Creation of a support software for the development of a system for sending and visualizing fbg sensor data

for aerospace application. In *2023 IEEE 10th International Workshop on Metrology for AeroSpace (MetroAeroSpace)*, pages 487–491. IEEE, 2023.

[12] Antonio Costantino Marceddu, Gaetano Quattrocchi, Alessandro Aimasso, Edoardo Giusto, Leonardo Baldo, Mohammad Ghazi Vakili, Matteo Davide Lorenzo Dalla Vedova, Bartolomeo Montrucchio, and Paolo Maggiore. Air-to-ground transmission and near real-time visualization of fbg sensor data via cloud database. *IEEE Sensors Journal*, 23(2):1613–1622, 2022.

[13] pothny3 (https://math.stackexchange.com/users/477375/pothny3). Simple question related to snell39;s law. Mathematics Stack Exchange. URL:https://math.stackexchange.com/q/2416385 (version: 2017-09-04).

[14] David R Walt, Israel Biran, and Tarun K Mandal. Fiber-optic chemical sensors. 2003.