POLITECNICO DI TORINO

MASTER's Degree in ELECTRONIC ENGINEERING

MASTER's Degree Thesis

Design and Development of a Distributed Monitoring System for Bridges

Supervisors

Prof. ALESSIO CARULLO

Prof. ALBERTO VALLAN

Dr. MARCO CIVERA

Dr. MAURO AIMAR

Candidate

Matilde BIDONE

OCTOBER 2024

Abstract

The state of Italy's bridges is alarming: more than half were built before the 1970s, making them more than 50 years old and approaching the typical lifespan of reinforced concrete structures built after the Second World War. Inadequate maintenance has led to numerous closures and collapses, with the cost of extraordinary repairs often exceeding that of demolition and reconstruction. In addition, only 60000 of Italy's 1.5 million bridges are monitored, often using outdated visual inspection methods. In this context, this thesis aims to provide an alternative solution, in the hope that tragic events such as the collapse of the Morandi Bridge (Genoa, 2018) will never happen again.

The thesis presents the design and development of a low-cost Structural Health Monitoring (SHM) system for bridges. The system is based on the postulate that a structural change due to damage results in a more or less pronounced change of its dynamic behaviour (natural frequency, damping ratio, mode shapes).

The proposed solution consists of a wired sensor network to be applied to the structure under test for continuous and real-time monitoring. It is based on two different printed circuit boards (PCBs). The first PCB (called BridgeWatch) integrates an accelerometer, an inclinometer, and a gyroscope to capture bridge movements. In the project, four of these boards have been included, which are to be placed in strategic locations to capture motion data useful for our purposes. The second PCB (called EnvironMonitor) collects environmental data, including rainfall, wind speed, temperature, and bridge-water level. Only one of this kind of board is sufficient.

The integration of multiple sensors allows for a comprehensive understanding of both structural and environmental conditions affecting the bridge's integrity, and facilitates the identification of correlations between them.

A on-site Raspberry Pi acts as the overall coordinator: it communicates with the BridgeWatch microcontrollers via an RS485 bus and an ad-hoc software protocol, and with the EnvironMonitor microcontroller via an RS232 bus and the NMEA 0183 protocol. Once all the data has been collected, it is processed and an output file is created for subsequent data analysis. In addition, the system is powered by a 12 V battery that is recharged by photovoltaic panels, eliminating the need for an electrical outlet.

The steps taken to create the system described above began with an analysis of the requirements and a study of the state of the art to keep up to date with the technologies currently in use. All the individual sensors to be used and the hardware components necessary for the overall operation were then selected and the schematics and layouts of the PCBs were created using Altium Designer. Once the PCBs were printed, all the

components were soldered by hand and the development of the firmware began. The main program is the one that runs on the Raspberry and manages all communication and synchronisation between the boards. On the other hand, each microcontroller on each board runs a firmware created with the IDE provided by STMicroelectronics, which manages communication with the sensors present and sends data to the Raspberry on request.

The final phase was laboratory validation. Two main test were carried out. The first involved using a shaker to reproduce sinusoidal signals at different frequencies, and analysing the data from a BridgeWatch board attached to the instrument using FFT. For all frequencies used in the test, the graph obtained after applying the FFT showed a single, clear peak at the input frequency, as we expected. The second test was carried out by placing the 4 BridgeWatch boards on a model building in the Civil Engineering department. The aim was to extract the natural frequencies of the structure from the data recorded by the boards, and to check that they matched the known ones. Also in this case all the obtained data was consistent with the expected results.

Acknowledgements

First and foremost, I would like to thank my parents, without whom none of this would have been possible. Thank you for making me the person I am, for always supporting me in all my decisions and for never letting me lack anything. You have taught me to look at reality with a critical eye, to ask questions and try to give me answers, and you have always stimulated my curiosity in all areas. Thank you, Dad, for making me love maths and physics so much, and for all our adventures in the mountains, which I hope will continue for a long time to come. Thank you, Mum, for being my pillar, for listening to my doubts and always giving me the best advice, and for passing on your values. I will be forever grateful to both of you.

Thanks to my friends Matti, Matte, Ale and Tommi, you made the "Poli years" the best years of my life. Thank you for all the laughs you made me have and all the experiences we shared together, I will carry them in my heart forever. I am sure that our relationship will not change even if we go our separate ways. I wish you all the best and I am sure you will do great things.

Thanks to Ale, who stands me and supports me every day. Thank you for believing in me more than I do, for all the things you have taught me and for all the moments and adventures we have shared. Thanks for your love, you are an example to me.

Thank you Emma, I will always be grateful to have known you and to have you in my life. You are one of the few people I know I can talk to about anything and that you will always understand, and I hope to share all our future achievements with you. Your smile is infectious, never lose it.

Thanks to Fra, Anna and Olghi, rowing and life companions since the last years of high school. Thank you for accompanying and supporting me at every stage of my journey, even though we see each other very little now, I always feel your affection and closeness.

Thank you to the entire Trail Running Torino group, in you I have found a family. Thank you for allowing me to clear my mind before and after every exam by running in the hills with you, and for giving me excuses not to study during the weekends when we organised long runs. You are an amazing group.

I would like to sincerely thank also my supervisors Marco and Mauro for their guidance and constant presence throughout my project. I would also like to thank my colleagues Valeria, Christian, Marco, Samuele and Davide for sharing this project and the activities of the Alta Scuola Politecnica with me. You have made this journey even more beautiful.

Finally, I would like to thank Berica Cavi, who provided us with the cable we needed free of charge by personally bringing it to us. You were very valuable.

Table of Contents

List of Tables

List of Figures

Acronyms

SHM

Structural Health Monitoring

PCB

Printed Circuit Board

FEM

Finite Elements Model

MEMS

Micro Electro-Mechanical Systems

FFT

Fast Fourier Transform

AI

Artificial Intelligence

ADC

Analog to Digital Converter

LSB

Least Significant Bit

SMD

Surface-Mount Devices

Chapter 1

Introduction

1.1 Motivation

The motivation for this thesis stems from wanting to contribute, even if only in a small way, to the resolution of a problem that can no longer be neglected in Italy: the monitoring and securing of infrastructure and bridges in particular.
Due to Italy's rather irregular orography, the heritage of road bridges and viaducts is quite significant. However, as explained in the articles from Marina Marzulli 2023 and Huffington Post 2018, most of them were built between 1955 and 1980, during the years of economic boom, when there were no specific standards on durability, quality of materials, and planned maintenance. In addition, they were designed for significantly lower traffic volumes than today. Inadequate maintenance actions during these years have led to numerous closures and collapses, with the costs for extraordinary repairs, according to the Italian Institute of Construction Technology (ITC), often surpassing those for demolition and reconstruction (Redazione ANSA 2018). Furthermore, Sensoworks estimates that only 60000 of Italy's 1.5 million bridges are monitored, mostly using outdated methods (Redazione BitMAT 2021). Of the others, little or nothing is known, and often their competence is unknown. What emerges is an alarming picture of the current state of our infrastructure that therefore deserves attention and concrete action.

The collapse of the Morandi Bridge (43 dead), in particular, uncovered this dramatic situation, and served as a stark reminder of the potential consequences of inadequate bridge monitoring and maintenance. The proposal for a low-cost, non-invasive, and easily applicable monitoring system is intended to give a new alternative to those involved in securing bridges, in the hope that these dramatic collapse events will never happen again.

1.2 Structural Health Monitoring

Infrastructure, including bridges, buildings and dams, plays a vital role in our daily lives. However, these structures can deteriorate over time due to various factors such as ageing, corrosion, environmental conditions, traffic loads and unexpected events such as earthquakes or accidents. Structural Health Monitoring (SHM) refers to the automated monitoring of civil infrastructure and is a critical process for assessing the condition of bridges and detecting damage and performance degradation. It involves the observation of a structure over time, using periodically sampled structural response and operational environment measurements from an array of sensors, and the subsequent assessment of the current condition and future performance of the structure (Chen and Ni 2018). It has received increasing interest over the last two decades as an alternative to manual inspection. The primary goal of SHM is to identify potential issues before they become critical allowing for timely maintenance or repairs, thus enhancing safety and reducing costs. An SHM system consists of two key components:

- **Data acquisition system:** it involves selecting sensor types, their number and location, and the data acquisition/storage/transmittal hardware.
- **Data analysis and interpretation tool:** it regards the sophisticated algorithms and data analysis techniques that are applied to the collected data to assess the structural condition. Anomalies or patterns indicative of damage or distress are identified.

1.3 Aim and Background: AMBROSE group

This thesis is part of a larger project in collaboration with the Alta Scuola Politecnica *XIX cycle*, a two-year multidisciplinary honours programme offered by the Politecnico di Torino and the Politecnico di Milano. The project, called AMBROSE (A Multisensor BRidge MOnitoring SystEm), aims to develop a complete SHM system for bridges, from data acquisition to analysis and interpretation, using innovative solutions. The basic idea is to develop an automated SHM system that uses sensors to collect data on the structural health of the bridge and advanced AI tools to analyse it. Myself and the 6 colleagues I worked with were divided into 3 groups, each focused on developing a different part of the system:

- **Indirect monitoring group:** It worked to design a mobile monitoring unit — i.e., a vehicle equipped with the necessary sensors and computing power to assess the condition of multiple bridges as it drives over them.
- **Direct monitoring group:** It worked to develop a system of sensors to be mounted directly on the bridge, providing continuous monitoring.
- **Data analysis group:** It worked to develop an AI-based Anomaly Detection Software capable of automatically analysing large amounts of data collected over time and draw conclusions about the health of the bridge.

Since not all bridges can be equipped with the fixed sensor system for cost reasons, the approach we propose follows a prioritisation of interventions: the bridges at risk are identified by driving the equipped vehicle over them (indirect monitoring), which is low-cost and much less invasive and time-consuming, and then a direct monitoring system is applied only to those bridges that require continuous monitoring.

This thesis is about the **development of the direct monitoring system**. So, it focuses only on the data acquisition stage, while the data analysis one is delegated to experts in the field.

1.4 Thesis outline

In the following chapters the development of the system will be analysed in details. The discussion begins with Chap.2, which examines the state of the art in SHM systems for bridges, focusing on the different monitoring approaches, sensor network technologies, real-world implementations by leading companies in the field, and the latest advances in the field. Then, Chap.3 outlines the specific requirements for the system and provides an overview of the proposed solution. It provides a concise introduction to the design approach and sets the scene for a more detailed discussion in the other chapters. An in-depth analysis of all the sensors utilized in the monitoring system is provided in Chap.4. It details each sensor's type, specifications, and functional role within the overall system. Chap.5 analyzes the schematics and provides a comprehensive explanation of the reasoning behind each design choice. Each section of the schematic is analyzed, illustrating the components chosen and their role. It also provides an estimate of the power consumption of the system. Chap.6 describes the layout design of printed circuit boards (PCBs), providing an overview of design considerations for component placement, signal routing and power distribution. Chap.7 provides a detailed description of all the communication protocols and buses used in the monitoring system, emphasizing their critical role in ensuring effective communication between sensors, boards, and the central computer. This chapter explores each protocol's specific functions, such as data transmission, synchronization, and error checking, highlighting how these protocols ensure reliable and accurate data exchange. Chap.8 focuses on analyzing the firmware developed for both the microcontroller and the Raspberry Pi, detailing how these software components manage the system's operations. Chap.9 focuses on the cost analysis of the system, to show that the proposed solution meets the requirements of being low cost. Chap.10 is dedicated to the validation of the monitoring system in a laboratory setting, detailing the processes and tests conducted to ensure its accuracy, reliability, and robustness. It describes the specific test scenarios designed to simulate various structural conditions and environmental factors, such as vibration frequencies and temperature variations, and explains the methodologies used to validate the system's performance. Finally, Chap.11 presents the conclusions drawn from the development, implementation and testing of the Structural Health Monitoring (SHM) system. This chapter summarises the key findings and achievements of the project, highlighting how the system successfully met the original design objectives, effective communication between components and reliable data processing.

Chapter 2

State of the Art in SHM Systems for Bridges

Before starting to design a new SHM system, it is crucial to carry out a state of the art analysis of the field. This is for several reasons. Firstly, knowledge of current SHM technologies and methodologies allows informed decisions to be made when selecting sensors, data acquisition systems and analysis techniques. Secondly, by reviewing the latest advances and best practices, the use of resources, including time, budget and technical expertise, can be optimised. This helps to avoid unnecessary investment in outdated or ineffective technologies. Finally, knowledge of existing SHM systems highlights gaps in current technologies and identifies areas where innovation or improvement is needed. This can lead to the development of more effective and efficient monitoring solutions. Incorporating the latest developments and techniques into SHM can therefore improve the overall performance and accuracy of the monitoring system.

2.1 Sensor Technologies

A variety of sensors are used to acquire the physical properties of the structure. It is essential to understand the main characteristics of the sensors used in SHM systems, as this knowledge will help to select the most appropriate sensors in the structure under investigation.

Moreno-Gomez et al. 2018 provides a comprehensive overview of the leading sensor technologies currently utilized in the field of SHM systems for bridges. As outlined in the paper, the sensors employed in Structural Health Monitoring systems are categorized into three primary types: kinematic, mechanical, and environmental sensors. Kinematic sensors are used to measure the motion induced by an external force that can be, for instance, moderate or strong winds, seismic waves, traffic and human-induced vibrations. This motion can be captured by measuring the displacement, velocity or acceleration of the in-test structure. Mechanical sensors are essential for monitoring the mechanical characteristics like fatigue, force, strain, corrosion, and cracks, which is essential in ensuring the safety of civil infrastructure. Environmental sensors, on the other hand, track external conditions like temperature, humidity, and wind speed, which can have a significant impact on the longevity and performance of the structure.

Below is a summarised list of the main sensor technologies used in these categories. The following sections then provide a more in-depth understanding of each technology.

- **Kinematic sensors**
  - Acceleration
    - Capacitive
    - Piezoelectric
    - MEMS
    - Force-balance
  - Velocity
    - Doppler effect
    - Electromechanical
    - Gyroscope
  - Displacement
    - Resistive
    - Linear variable differential transformer (LVDT)
    - Global Positioning Satellites (GPS)
- **Mechanical sensors**
  - Fatigue and crack detection
    - Acoustic Emission
    - Eddy currents
    - Ultrasonic waves
    - Termography
  - Force measurement
    - Load cells
  - Strain
    - Strain gauge
    - Piezoelectric
    - Vibrating wire strain gauge
  - Corrosion
    - Radio-frequency identification (RFID) waves
    - Acoustic emission
    - Eddy currents
- **Environmental sensors**

  - Wind
    - Mechanical anemometer
    - Ultrasound-based anemometer
  - Temperature
    - Thermocouples
    - RTD

- **Novel technologies trends**
  - Vision-based sensors
  - Fiber-optic-based sensors

2.1.1 Acceleration

Capacitive

A capacitive accelerometer (shown in Fig. 2.1a) works by measuring the capacitance between two plates of a capacitor. One plate is attached to an inertial mass and can be moved, the other is fixed to the internal case of the sensor. When the sensor is subjected to vibrations, the mass inside the sensor moves, changing the distance between the capacitor plates and causing that the capacitance value changes as well. The capacitance C of a parallel plate capacitor is given by the formula:

$$C = \frac{\varepsilon_0 \varepsilon_r A}{d} \tag{2.1}$$

where:

- C is the capacitance,
- ε_0 is the permittivity of free space,
- ε_r is the relative permittivity of the dielectric material between the plates,
- A is the area of one of the plates,
- d is the distance between the plates.

When the sensor is subjected to acceleration, the distance d changes in response to acceleration, thus altering the capacitance C. This change in capacitance is proportional to the acceleration experienced by the sensor and is used to determine the magnitude and direction of the acceleration.

Piezoelectric

The core principle behind piezoelectric accelerometers is the piezoelectric effect, where certain materials (such as quartz or ceramics) produce an electric charge when subjected to mechanical stress. In a typical piezoelectric accelerometer (Fig.2.1b), a seismic mass is attached to a piezoelectric element. When the accelerometer experiences acceleration, the seismic mass applies a force to the piezoelectric material, causing it to deform and generate an electric charge proportional to the applied force. This charge is then converted into a voltage signal, which can be measured and analyzed.

Examples of the use of piezoelectric accelerometers are the monitoring system for the bell tower of San Pietro in Perugia (Ubertini et al. 2017) or that of the Gabbia Tower in Mantua (Saisi et al. 2015).

Force-balance

A force-balance, or servo, accelerometer (Fig. 2.1c) measures acceleration by using a feedback control system to maintain a seismic mass in a fixed position. It consists of three main components: a seismic mass that responds to acceleration, a displacement sensor that detects shifts in the mass's position, and an actuator that generates a counteracting force. When acceleration is applied, it tends to displace the proof mass from its equilibrium position. This displacement is detected and converted into an electrical signal, which is then used to generate a counteracting force to keep the proof mass stationary. The output signal from the accelerometer reflects the amount of displacement of the seismic mass, which is directly proportional to the external acceleration, thus providing an accurate measure of acceleration based on the force required to maintain equilibrium.

Rainieri et al. 2019 describe a SHM system applied to the Campobasso's main hospital to monitor the vibration response of the structure which make use of sixteen force-balance accelerometers.

MEMS

Since the 1990s, MEMS accelerometers (Fig. 2.1d) have revolutionised structural health monitoring due to their advantages of miniaturisation, lower cost and broader applications. These devices integrate micro-scale mechanical elements with electronic components, enabling precise measurement of structural vibration and motion. MEMS accelerometers can use a variety of transduction mechanisms, including piezoresistive, capacitive, piezoelectric and inductive, with capacitive types being the most common in commercial applications. These sensors work by detecting changes in capacitance caused by the movement of a micro-scale proof mass in response to external accelerations. This movement changes the capacitance between fixed and moving plates, generating a corresponding voltage change that can be measured and analysed. The advantages of MEMS accelerometers, such as their low power consumption and ease of integration, make them highly effective for continuous monitoring of the structural integrity of bridges, buildings and other critical infrastructure.

A list of positive MEMS applications for the SHM and dynamic identification of civil engineering constructions, including wireless options, can be found in the literature. Picozzi et al. 2009 proposed the use of MEMS devices for the SHM of a suspension bridge in Istanbul. Chiara Bedon et al. 2018 present an experimental validation of MEMS accelerometers using, as a case study, the cable-stayed bridge in Pietratagliata (Italy). Dynamic results summarised in the paper demonstrate the high capability of MEMS accelerometers, with evidence of rather stable and reliable predictions, and suggest their feasibility and potential for SHM purposes.

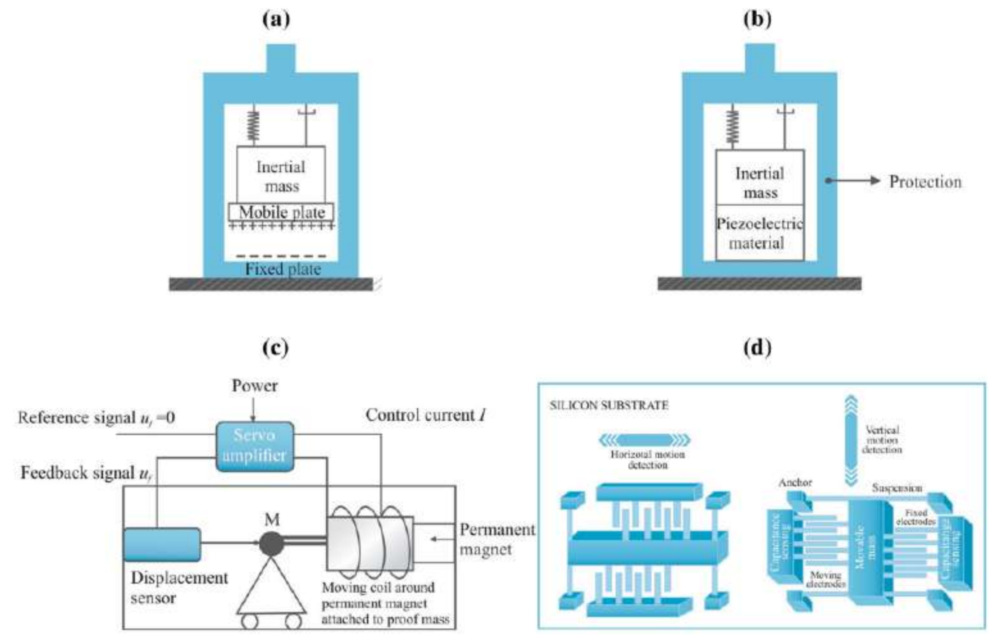


Figure 2.1: Accelerometer types: **a** Capacitive, **b** Piezoelectric, **c** Force-based, **d** MEMS. Source: Moreno-Gomez et al. 2018

2.1.2 Velocity

Doppler-effect

Doppler effect-based velocity sensors measure the velocity of an object by analyzing the frequency shift of waves reflected from the moving object. When a wave source (such as a laser or radar) emits waves towards a moving object, the frequency of the reflected waves changes depending on the object's speed relative to the sensor. This frequency shift, known as the Doppler shift, is directly proportional to the velocity of the object. The sensor detects this frequency shift and processes it to calculate the object's speed.

Yu et al. 2024 describe their study about the identification of structural properties (stiffness, mass, and damping) of a steel railway bridge located at the border of New Hampshire and Maine using laser Doppler vibrometry (LDV) for SHM.

Electromechanical

Electromechanical velocity sensors typically use a moving coil or magnet assembly where the velocity of an object induces a change in an electrical signal. When a mechanical component moves, it causes a relative motion between a coil and a magnet. This motion generates an electrical voltage proportional to the velocity of the movement. In essence, the sensor converts the kinetic energy of the moving part into an electrical signal, which can be measured and calibrated to determine the velocity.

Gyroscope

Gyroscope-based velocity sensors measure rotational velocity and angular rates rather than linear velocity. Gyroscopes operate based on the principle of conservation of angular momentum. In a typical gyroscope, a rapidly spinning rotor or a vibrating element resists changes to its orientation due to its angular momentum. As the sensor rotates, the gyroscope detects changes in orientation or angular rate by measuring the forces exerted on the spinning rotor or vibrating element. While gyroscopes are primarily used to measure rotational motion, they can be combined with other sensors, such as accelerometers, for a robust estimation of both displacements and rotations (Faulkner et al. 2020).

2.1.3 Displacement

Resistive-based transducers

Resistive-based transducers, often referred to as resistive position transducers or potentiometers, measure displacement through changes in electrical resistance. These devices consist of a resistive element, typically a wire wound into a coil or a flat resistive strip, and a wiper or slider that moves along this element. As the wiper changes its position, it changes the resistance encountered by the electric current. This change in resistance is then converted into an electrical signal representing the displacement.

GPS

GPS technology uses satellite signals to determine the precise position of objects on Earth. GPS receivers calculate their position by linking up with four satellites, each of which transmits its own position. They then measure the time it takes for the information sent by the satellites to reach them. This time delay is converted into distance measurements that allow the receiver to determine its position in three-dimensional space.

LVDT

The linear variable differential transformer is composed of a nickel-iron-based core surrounded by a primary winding and two secondary ones, which are connected in a series way. The cylindrical ferromagnetic core is attached to the object whose position is to be measured, and slides along the axis of the tube. An alternating current drives the primary and causes a voltage to be induced in each secondary proportional to the length of the core linking to the secondary. The movable ferromagnetic core within the LVDT shifts the coupling between the primary and secondary coils, causing a change in the induced voltage in the secondary coils. Since the coils are connected so that the output voltage is the difference (hence "differential") between the top secondary voltage and the bottom secondary voltage, the output is proportional to the core's displacement, allowing for extremely accurate measurements (Scuro et al. 2021).

2.1.4 Fatigue and crack detection

Eddy currents

Eddy current testing is based on the principle of electromagnetic induction. When an alternating current (AC) is passed through a coil or probe, it generates a time-varying magnetic field. This magnetic field induces circulating currents—called eddy currents—in the conductive material being tested. As the magnetic field from the probe penetrates the material, it induces eddy currents that flow in closed loops perpendicular to the direction of the magnetic field. The distribution and intensity of these eddy currents depend on the electrical and magnetic properties of the material, as well as any surface or subsurface defects present. Fatigue causes the formation and growth of microcracks or other structural changes in the material over time. These microcracks can alter the flow of eddy currents in the affected area (Papazian et al. 2007; Gasparin et al. 2010).

This technology is also used for corrosion detection since corrosion can cause a reduction in the material's thickness and create irregularities on the surface.

Acoustic Emission (AE) sensors

Acoustic Emission (AE) sensors detect the high-frequency waves generated by the rapid release of energy from material defects, such as cracks, when they propagate or grow. When a material undergoes stress, it can emit transient acoustic waves due to micro-cracking, dislocation movements, or other mechanisms. AE sensors are placed on the surface of the material or structure and are sensitive to these high-frequency acoustic waves, which are typically in the range of 20 kHz to 1 MHz. By analyzing the characteristics of these acoustic emissions, such as their amplitude, frequency, and duration, AE sensors can identify the location, type, and severity of cracks. An active AE-based sensor uses a piezoelectric transducer to convert the ultrasonic waves that travel through the material. This technology is also used for fatigue and corrosion detection (Nair and Cai 2010; Melchiorre et al. 2023).

Thermography

Thermography involves the use of thermal imaging cameras to detect variations in surface temperature that can indicate the presence of cracks. When a structure is subjected to thermal energy, such as from heating or cooling, cracks can affect the heat distribution and cause temperature anomalies. Thermographic sensors capture infrared radiation emitted by the surface and create thermal images that highlight temperature differences (Mehdi Modares and Natalie Waksmanski 2013).

Ultrasonic waves

Ultrasonic SHM is a highly effective technique for detecting localised defects in structures. It is an active method because it requires an actuator that generates high frequency ultrasonic waves that propagate through the material of the structure. Sensors then measure the response of the structure to these waves, recording data on changes in wave propagation. This approach is highly sensitive to small defects, such as cracks, voids or material degradation, which can alter the way the ultrasonic waves travel through the structure (Mutlib et al. 2015; Papazian et al. 2007).

2.1.5 Force measurement

Load cells

They are a transducer device that converts the applied force into an electrical signal; although different types of load cell can be used, the most common one is a cell composed of strain gauges. A load cell typically consists of a metal body, often referred to as the sensing element or strain gauge body, which deforms under applied load. Strain gauges are bonded to this metal body, and they measure the deformation (strain) that occurs when a force is applied. As the metal body deforms, the strain gauges experience changes in resistance due to the mechanical strain. These changes in resistance are then converted into an electrical signal using a Wheatstone bridge circuit. The resulting electrical signal is proportional to the applied force and can be calibrated to provide accurate measurements (Mehdi Modares and Natalie Waksmanski 2013).

2.1.6 Strain

Strain gauges

A strain gauge typically consists of a thin, conductive foil arranged in a grid pattern that is bonded to the surface of the material being tested. As said before, when the material deforms due to applied force, the strain gauge deforms as well, leading to a change in the length of the conductive path. This deformation alters the electrical resistance of the strain gauge. By using a Wheatstone bridge circuit, these resistance changes can be precisely measured and correlated to the amount of strain experienced by the material.

Vibrating wire strain gauges

Vibrating wire strain gauges measure strain by detecting changes in the frequency of a vibrating wire subjected to mechanical deformation. These gauges consist of a wire that is stretched and vibrated at a natural frequency. When strain is applied to the structure to which the gauge is attached, the wire experiences a change in tension, which alters its vibration frequency. The change in frequency is directly proportional to the amount of strain.

The static monitoring system for Milan Cathedral includes 12 vibrating wire extensometers, among other sensors (Gentile et al. 2019).

2.1.7 Corrosion

RFID

An RFID system used for detecting corrosion consists of two main components: the reader and the tag. The tag is a specialized circuit that includes a coil, capacitors, and resistors, and it generates a magnetic wave at a frequency typically above 10 MHz. This tag is placed on or within a test sample. The reader, positioned at a certain height above the tag, uses the principle of electromagnetic induction to generate an electrical current in the tag. This current is proportional to the strength of the magnetic field produced by the reader. When the tag is placed under the material being analyzed, any changes in the material, such as those caused by corrosion, will alter the magnetic properties of the tag. As a result, the impedance of the tag's circuit changes. The reader measures these changes in the induced current, which reflect variations in the tag's impedance. By analyzing these variations, particularly the real (resistive) and imaginary (reactive) components of the impedance, the system can detect signs of corrosion and assess the condition of the material.

The technologies examined so far were sensors for structural monitoring (static and dynamic). In the following sections, the technologies used for environmental monitoring sensors will be considered instead.

2.1.8 Wind

Mechanical anemometer

A mechanical anemometer, specifically the cup type, measures wind speed using rotating cups mounted on horizontal arms. As the wind blows, it pushes the cups, causing them to rotate around a vertical axis. The speed of rotation is proportional to the wind speed. The rotational speed of the cups is typically measured using a tachometer or a counter that records the number of rotations per unit of time and then applying a relationship to estimate the velocity. To measure wind direction, a separate device called a vane is often used in addition to the mechanical anemometer. The wind vane has a blade mounted on a horizontal axis that points in the direction of the wind. The end of the vane points in the direction from which the wind is coming. The wind vane is typically connected to a directional indicator, which provides a visual or electronic readout of the wind direction.

Ultrasound-based anemometer

An ultrasonic anemometer measures wind speed and direction using ultrasonic sound waves. It operates based on the time-of-flight principle, where the time it takes for sound waves to travel between transducers is affected by the wind speed and direction. An ultrasonic anemometer typically has three or four pairs of transducers arranged in a triangular or orthogonal configuration. Each transducer can emit and receive ultrasonic pulses. When the anemometer emits ultrasonic pulses from one transducer to another, the wind will either speed up or slow down the pulses depending on its direction relative to the sound wave. The time-of-flight (the time it takes for the pulse to travel from one transducer to another) is measured in multiple directions by the electronic circuitry. By analyzing the differences in travel times along these different paths, the anemometer can calculate not only the wind speed but also the direction of the wind.

2.1.9 Temperature

Thermocouples

Thermocouples (Fig. 2.2a) operate based on the Seebeck effect, which occurs when two wires made of different metals are joined at both ends and subjected to a temperature difference. When one junction (referred to as the hot junction) is heated while the other (the cold junction) remains at a different temperature, a continuous current flows through the thermoelectric circuit. If this circuit is interrupted in the middle, a voltage—known as the Seebeck voltage—develops across the open ends. This voltage is directly related to the temperature difference between the hot and cold junctions and depends on the specific metals used. By measuring this voltage, the temperature at the hot junction can be determined with reference to the cold junction.

RTD

RTDs (Fig. 2.2b) work based on the principle that the electrical resistance of a metal changes with temperature. Specifically, the resistance of most metals increases as the temperature rises. The most common material used in RTDs is platinum, known for its stable and predictable change in resistance with temperature. An RTD is typically constructed by winding a thin wire of platinum (or another metal) around a ceramic or glass core. This wire is then encapsulated in protective sheathing to ensure durability and protection from environmental factors. These sensors are one of the most accurate as they have a linear relationship between resistance and temperature.

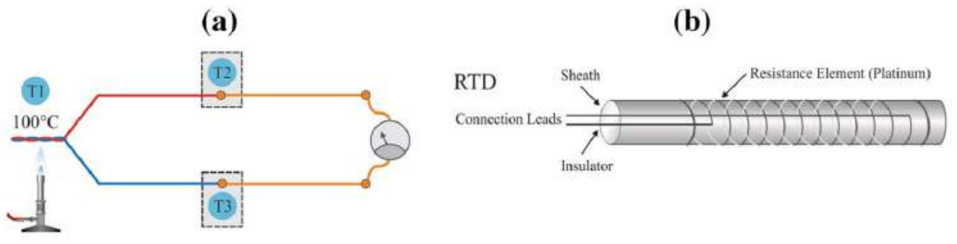


Figure 2.2: Temperature sensors: **a** Thermocouple, **b** RTD. Source: Moreno-Gomez et al. 2018

2.1.10 Novel technologies trends

Vision-based sensors

Traditional SHM systems relies on the installation of a large number of accelerometers and sensors to obtain accurate results. This process can be very expensive and time consuming. However, recent advances in computer vision and machine learning-based computational models have led to significant improvements in vision-based SHM. Unlike traditional sensors, vision-based systems are non-contact, non-destructive, and does not impose the utilization of wires, allowing for long-distance, high-precision monitoring without interfering with the structure's operation. Moreover, they can monitor multiple targets simultaneously over large areas, making them ideal for complex structures. This measurement system generally consists of the image acquisition device, the computer and an image processing software platform, which is the critical part (Indhu et al. 2022).

The structural displacement can in fact be acquired with a high speed camera at a high sample rate, which can satisfy the need of structural vibration monitoring and dynamic characteristics identification. Fukuda et al. 2010 developed a vision-based system to monitor the dynamic response of large-scale civil infrastructure which was more cost-effective.

Another application of vision-based sensors is the analysis of structural surface features, such as cracks and spalling, on steel and concrete structures. Adhikari et al. 2014 presented an approach of automated condition assessment of concrete bridges based on digital image analyses. Ho et al. 2013 developed an image-based system using cameras attached to a cable climbing robot to detect surface damage to cables using image processing and pattern recognition techniques.

Fiber-optic-based sensors

These sensors operate by utilizing the principle of light transmission through optical fibers, which are thin, flexible strands of glass or plastic that guide light along their length. The basic working principle of fiber-optic sensors involves sending a light signal through the optical fiber. As the light travels through the fiber, it interacts with the surrounding environment. Any changes in temperature, strain, or other physical properties cause a modification in the light's characteristics, such as its intensity, phase, wavelength, or polarization. These changes are then detected and analyzed to determine the corresponding physical parameter. Fiber-optic sensors offer several advantages over traditional sensors, including immunity to electromagnetic interference, high sensitivity, and the ability to function over long distances without signal loss. However, they tend to be more expensive than other options, which can be an important consideration when evaluating their use in different applications (Bremer et al. 2016; Wu et al. 2020).

2.2 Typical Approaches in SHM for Bridges

2.2.1 From visual inspections to SHM systems

Periodic visual inspections by expert observers have long been a traditional method of assessing the condition of bridges in structural health monitoring. Although this method has advantages, such as a local and visual visit and a human understanding of the damage on site, it also has several disadvantages. As well as being time consuming and not cost effective, a major drawback is the difficulty or impossibility of accessing certain parts of the bridge, particularly on complex or large structures. This limited access means that not all areas can be thoroughly inspected, potentially leaving some damage undetected. Moreover, during visual inspections, internal damage and damage initiation are very difficult to detect (M. Modares and N. Waksmanski 2013; Zinno et al. 2023).

Over the past few decades, there has been considerable progress in enhancing SHM systems to address these limitations. By using a structural health monitoring procedure, the condition of a bridge can be continuously monitored, offering reliable information and the early detection of structural defect. The early indication of problems provided by an SHM process significantly reduces maintenance costs and prevents catastrophic failure. In addition, SHM systems can operate in all weather conditions and in real time, providing a level of accuracy and reliability that surpasses traditional inspection methods. Several methodologies have been developed to implement SHM systems on bridges:

2.2.2 Local Monitoring Techniques

Local monitoring focuses on specific areas or components of a structure, providing detailed information on localised damage or stress. This approach is particularly effective in identifying and assessing damage in critical areas that are subject to fatigue or where previous damage has been detected. In bridges, for example, local monitoring techniques are often applied to areas where damage is most likely to occur, such as joints, bearings or known structural weaknesses. Technologies such as acoustic emission sensors, strain gauges and displacement sensors are commonly used. For instance, strain gauges can be used to measure the deformation in specific components, providing real-time data on how the material responds to load and environmental conditions. Similarly, AE monitoring can detect the onset of cracking or other forms of damage by capturing the stress waves emitted by the material under strain. However, all of these experimental techniques require that the vicinity of the damage is known *a priori* and their scope is limited to the areas where sensors are installed. In addition, they may not provide a complete picture of the overall structural health of the bridge. As such, local monitoring is often used in conjunction with other methods to provide a more comprehensive assessment of the bridge's condition (Zinno et al. 2023).

2.2.3 Global Monitoring Techniques

On the other hand, global monitoring methods are designed to assess overall structural integrity by analysing the behaviour of the entire system. Global methods are commonly referred to as Vibration-Based Damage Detection Methods (VBDDM) because they involve monitoring vibration responses using accelerometers to extract important data about the condition of the structure. In fact, the basic principle of the vibration-based monitoring method is that the dynamic characteristics of structural vibration, such as natural frequency, modal shape, and damping ratio, are intrinsically linked to the physical properties of the structure, such as mass, stiffness, and damping. Consequently, any changes in the structure, such as the development of cracks or loosening of connections, lead to alterations in its dynamic behavior, notably affecting frequency response functions and modal characteristics. The changes of them and their derived quantities can be used as characteristic indicators of structural damage. Salawu 1997 presents an excellent review on the use of modal frequency changes for damage diagnostics (Doebling et al. 1998).

Global methods are particularly useful for large structures where it is impractical to monitor each component individually. Vibration-based techniques have been widely investigated and adopted in civil engineering, ranging from historic buildings [2-5] to more modern long-span bridges [6-9].

Within the vibration-based methods, two branches exist:

2.2.4 Model-driven methods

In a model-based approach, a finite element analysis (FEA) is done specifically for each bridge which is calibrated with the sensor data. This model represents the expected behavior of the structure under various conditions, allowing for the identification of deviations from the norm. These deviations can then be analyzed to detect potential damage or deterioration. However, they require accurate input data and significant computational resources, making them complex and time-consuming to develop.

2.2.5 Data-driven methods

Data-based methods, in contrast, do not rely on detailed physical models of the structure. Instead, they use machine learning and statistical techniques to analyze data collected from sensors installed on the bridge. Sensor data is collected over a certain period,considered to be the structurally healthy baseline, and deviations from the baseline are assessed as structural damage. These methods can detect patterns and anomalies in the data that may indicate structural issues. Data-based methods are highly adaptable and can handle large volumes of data, making them suitable for real-time monitoring. However, they require extensive training data and may not always provide insights into the underlying causes of detected anomalies.

2.3 Wired vs Wireless Sensor Networks

One of the key features of a bridge Structural Health Monitoring system is the use of a network of sensors to collect real-time data on the structural integrity and performance of the bridge. As technology has advanced, so has the design and deployment of sensor networks. One of the key distinctions in SHM sensor networks is between wired and wireless systems. This difference has significant implications for how data is collected, transmitted and analysed, as well as the overall cost and complexity of the SHM system. Understanding the strengths and limitations of both wired and wireless sensor networks is essential to selecting the most appropriate monitoring strategy for a given bridge.

2.3.1 Wired Sensor Networks

Wired sensor networks involve the installation of physical cables connecting various sensors to a central data acquisition system. These systems offer reliable and stable data transmission with minimal interference, making them suitable for environments where continuous and high-fidelity measurements are essential. Despite their reliability, wired networks can be complex and costly to install, particularly in large-scale or hard-to-access bridge structures. An example of use of a wired sensor network is the monitoring of Hakucho Suspension Bridge, the largest suspension bridge in northeastern Japan.The bridge has permanent seismic monitoring system consisting of 27 channels of vibration sensors placed on 14 locations. They include 22 channels of uniaxial accelerometer, two uniaxial displacement sensors, and a triaxial free-field strong-motion accelerometer (Siringoringo and Fujino 2018). Another famous example is the monitoring system of the Great Belt Suspension Bridge in Denmark, considered to be one of the most advanced in the world (Friis et al. 2024).

2.3.2 Wireless Sensor Networks

Wireless sensor networks (WSNs), on the other hand, offer a more flexible and cost-effective alternative by eliminating the need for extensive cabling. These systems use wireless communication technologies, such as Wi-Fi, to transmit data from sensors to a central monitoring system. WSNs are easier to deploy and maintain because sensors can be placed in hard-to-reach locations without the constraints of cabling. However, most WSN systems for bridge structural health monitoring are battery-powered, so a major limitation in this context is the limited battery life and the need for regular, costly battery maintenance. As a result, a large number of bridges and buildings have been instrumented with wireless monitoring systems, including for example the Geumdang Bridge (South Korea) (Lynch et al. 2006) or the Gi-Lu cable-stayed bridge located in Nantou County, Taiwan (Weng et al. 2008). Jindo Bridge, a cable-stayed bridge in South Korea with a 344-m main span and two 70-m side spans, constitutes one of the largest deployment of wireless smart sensors for civil infrastructure monitoring (Jang et al. 2010).

2.4 Industry Leaders in SHM for Bridges

When developing new Structural Health Monitoring (SHM) systems for bridges, it is essential to understand the landscape of existing technologies and the companies leading the industry. Several companies have established themselves as pioneers in SHM, offering state-of-the-art solutions that set the standard for bridge monitoring. Understanding the offerings of these companies is not only about recognising the competition, it also provides valuable insight into current technology trends and gaps in the market. For any new SHM system development, it is essential to benchmark against these established solutions to ensure that the new system meets or exceeds current standards. Some of the key players in this field include:

2.4.1 Acellent Technologies

Acellent Technologies is specialized in Structural Health Monitoring within various sectors including aerospace, military, automotive, manufacturing, civil infrastructure, and energy. The company offers a comprehensive SHM solution that includes SMART Layer sensors, diagnostic hardware, and software for monitoring the integrity and condition of structures. Acellent's products are designed to detect damage, analyze its severity, and predict the remaining lifespan of assets, contributing to safety and structural integrity. It is based in Sunnyvale, California (*Acellent Technologies* n.d.).

2.4.2 OSMOS Group

OSMOS Group specializes in Structural Health Monitoring (SHM) within the civil engineering and industrial equipment sectors. The company offers continuous and real-time monitoring services using fiber optic technology to assess and ensure the structural integrity of various constructions. They also have a Mathematics Department which works closely with the structural engineers to develop data processing algorithms. OSMOS Group's solutions are utilized across a broad range of sectors, including bridges and heritage sites, as well as commercial and industrial buildings. It is based in Courbevoie, France (*OSMOS Group* n.d.).

One innovative product they offer for bridge monitoring in particular is the OSMOS WiM+D® tool. It is an automated solution that combines weighing-in-motion techniques with structural deformation analysis. Unlike traditional monitoring systems, OSMOS WiM+D® not only tracks the traffic on a bridge, but also assesses the impact of this traffic on the structural integrity of the bridge. Using advanced fibre-optic sensors, known as OSMOS Optical Strands, the system is able to detect the speed, length, direction, number of axles, weight distribution per axle and total weight of the heavy vehicle, as well as the maximum stress caused by its passage over the structure. Heavy vehicles are automatically classified by weight and direction, which can then be viewed via the SAFE Works online interface (*OSMOS Group - Weigh in motion & Deformation* n.d.).

2.4.3 Mistras Group

MISTRAS' expertise is inspection and long-term monitoring of a variety of bridge types, but their systems are also used to monitor dams, roads, and railways. Serving some of the world's most iconic bridges, the MISTRAS Group combines inspection, sensor fusion and wireless 24/7 online monitoring systems to observe the real-time condition of suspension cables, cable stays and other critical structural components in bridges and other civil structures. They are a world leader in the development and use of Acoustic Emission (AE) to detect and assist in the repair of cracks and corrosion before real damage occurs. The company is based in Princeton Junction, NJ, USA (*MISTRAS Group - Bridge Monitoring* n.d.).

2.5 Evolution of SHM

Recent advances in sensor technologies, data analysis, and communication systems have significantly improved the effectiveness of SHM systems. Some of the key developments include:

2.5.1 Machine Learning and AI in SHM

Machine learning (ML) and artificial intelligence (AI) are increasingly playing a crucial role in the advancement of Structural Health Monitoring (SHM) systems for bridges. These technologies enable the development of predictive models and diagnostic tools that can automatically analyze large volumes of sensor data to detect anomalies, assess damage, and predict the remaining useful life of bridge components. Unlike traditional SHM methods, which often require predefined rules and manual interpretation, ML algorithms can learn patterns directly from data, thus improving the accuracy and efficiency of damage detection and localization.

Recent studies have demonstrated the effectiveness of AI-based SHM systems in identifying damage and predicting maintenance needs. For example Cha et al. 2017 used a Convolutional Neural Network (CNN) architecture to classify and locate cracks in images taken from routine bridge inspections. The AI-based model demonstrated a high degree of accuracy and reliability, significantly outperforming traditional image processing techniques. Ásgrímsson et al. 2021 explore the application of Bayesian deep learning for detecting structural damage in bridges using vibration data. The study utilizes the Z-24 bridge benchmark dataset, where a sensor instrumented, three-span bridge was monitored for almost a year before being deliberately damaged in a realistic and controlled way.

2.5.2 Integration with Building Information Modeling (BIM)

Building Information Modelling (BIM) in construction is a process that enables collaboration between all the professionals working on a building throughout its life. BIM stands for Building Information Modelling and is defined as the digital representation of the physical and functional characteristics of an object, regardless of its type. It consists of a 3D model that contains all the physical, performance and functional data of the object.

However, it is not just a new format for 3D representation; it is a technology that enables the creation of an informative and collaborative model that provides useful information at every stage of the design process. This includes architectural, structural, mechanical and electrical designs, as well as detailed information on material properties, components and systems. BIM also integrates planning aspects such as construction phases, schedules, cost estimates and maintenance strategies to manage the entire project lifecycle in a consistent and efficient manner.

In this way, BIM enables real-time collaboration between all parties involved in a construction project, such as architects, engineers and contractors, leading to massive improvements in cost, safety and efficiency. As a result, it is one of the most influential construction trends today (Stannard 2021).

Bridge Information Modelling (BrIM) is a new concept derived from BIM with a focus on bridge projects. Like BIM, BrIM has been introduced to efficiently manage the exchange of information between parties involved in bridge structures (Jrade et al. 2023).

BrIM allows designers to create digital twins of bridges, with all the information and data about the bridge's components. Permanent monitoring systems can also be implemented in the models. This is possible by introducing characteristic point elements with information about the technology used and with data from sensors installed on the bridges. By embedding sensor data directly into the BIM models, real-time monitoring of structural performance becomes possible, facilitating maintenance and management strategies. This integration allows engineers to dynamically visualise and assess the health of the bridge structure, using 3D models to pinpoint areas of concern. For example, anomalies detected by sensors can be mapped directly onto the digital twin of the bridge, providing a more intuitive understanding of potential damage locations and supporting faster decision making (Gragnaniello et al. 2024; McGuire et al. 2016).

2.5.3 Indirect Monitoring Systems

Despite the many advantages of a traditional SHM system (with sensors applied directly to the structure), several challenges have hindered its widespread adoption. A major obstacle is the high cost of sensor installation and maintenance, which can be prohibitive, especially for smaller administrations with limited budgets. In addition, the vast number of bridges and viaducts in transport networks makes the design of a complete monitoring system impractical when applied at network scale.

To overcome this, the idea of indirect measurement for bridge monitoring has become an increasingly popular area of research. This idea refers to the use of the indirectly measured response of a passing vehicle to extract dynamic properties of bridges. Yang et al. 2004 first established the feasibility of this method, but since then several research papers have been published to improve the topic to its current state, especially regarding which algorithms to use. Tan et al. 2019 proposes an algorithm to extract bridge mode shapes and damping ratio.

Shokravi et al. 2020 has carried out a comprehensive analysis of the state of the art of these methods.

Chapter 3

Design of the monitoring system

3.1 Requirements

In every project, the starting point is always the client's requirements. For such a monitoring system, the requirements were told to us by the civil engineers following our project. Their requirement was for a vibration-based monitoring system specifically designed for a concrete bridge. The system needed to accurately measure accelerations to determine the bridge's natural frequencies, which could then be compared with a finite element model (FEM) of the bridge or analysed with AI software to assess potential damage. In addition, the system needed to include a meteorological station to monitor key environmental conditions and a structural temperature sensor to enable more accurate correlations and analysis. In particular, their requirements are:

1. **Sensor Types and Specifications**

    - **Accelerometers:** To detect vibrations and movements in the structure. The requirements are resumed in table 3.1.
    - **Tiltmeter:** To detect slow-varying inclinations. The requirements are resumed in table 3.2.
    - **Gyroscope:** To detect fast-varying inclinations and have more reliable data by combining these data with those of accelerometers. The requirements are resumed in table 3.3.
    - **Temperature sensor for the structure:** To monitor structural changes due to temperature.
    - **Environmental sensors:** To monitor environmental factors such as wind speed and wind direction, quantity of rain, air temperature and humidity, with requirements resumed in table 3.4. The sensor used in this case are respectively an anemometer, a rain gauge, and a termohygtrometer.

- **Water level sensor:** To monitor the height between the bridge and the water, with requirements resumed in table 3.5. The sensor used for this purpose is a radar sensor.

2. **Data Acquisition and Sampling**

   - **Appropriate Sampling Rate:** The system must be capable of capturing data at an appropriate sampling rate to accurately record dynamic behaviors. More details in Sec. 3.2.1.
   - **Real-Time Monitoring:** Instantaneous data collection and processing for immediate analysis.
   - **Data Synchronization:** All sensors should be synchronized to ensure the accuracy of multi-sensor data correlations.
   - **Data Integrity:** Implementation of error-checking protocols to prevent data loss or corruption.

3. **Environmental Requirements**

   - **Weather Resistance:** All components must be weatherproof, capable of withstanding extreme temperatures, humidity, and precipitation.
   - **Vibration Resistance:** The system should be robust against the vibrations it is designed to monitor.

4. **Power**

   - **Low Power Consumption:** The system should be energy-efficient to reduce the need for frequent maintenance.
   - **Solar Power Capability:** The system must be designed to operate with photovoltaic panels as its primary power source. This requirement is crucial to ensure the system can function in remote locations where traditional electrical connections are not available. The system should include a battery storage component to store excess energy generated during the day, providing continuous power during periods of low sunlight.

5. **Installation and Accessibility**

   - **Non-Intrusive Installation:** The system should be installed with minimal disruption to the bridge's operation.
   - **Easy Access for Maintenance:** Components should be positioned to allow easy access for inspection and maintenance.
   - **Remote Access:** The system should allow engineers to access data and diagnostics remotely.
   - **Modularity:** The design should allow for easy addition or replacement of sensors as needed.

6. **Cost Considerations**

- **Cost-Effective Solutions:** The system had to be as cost effective as possible as there is a fixed budget.

Table 3.1: Accelerometer requirements

Accelerometer	
Range	±2 g
Sample rate	$\geq$ 256 samples/s
Resolution (±2 g range)	80μg
Noise	25μg/$\sqrt{\text{Hz}}$
Bandwidth	500 Hz

Table 3.2: Tiltmeter requirements

Tiltmeter	
Range	±1°
Sample rate	1 samples/h
Resolution (±1° range)	0.000,27°(1 arcsec)

3.2 Architecture of the SHM system proposed

The idea developed is to use a wired sensor network to collect data on the structural and environmental condition of the bridge and send it to a on-site computer (specifically a Raspberry Pi) that acts as a data collector. This process the data and send it via a SIM card to a server, that can be accessed remotely and in real time for analysis using AI software tools for anomaly detection. The sensor network consists of two different boards with different purposes:

- **BridgeWatch:** responsible for structural monitoring. It is equipped with an accelerometer, inclinometer and gyroscope, as well as the input connector for an RTD probe to measure the temperature of the structure.

- **EnvironMonitor:** responsible for environmental monitoring. It has input connectors for external sensors such as rain gauge, anemometer, bridge-water height sensor, and thermo-hygrometer.

Each board also has a microcontroller to which all connected sensors send data and which communicates with the central computer.

I choose to employ a wired sensors network for my system, since the synchronization is easier and it is more reliable even in difficult weather conditions. Furthermore, it allows me to use the knowledge of communication buses that I have acquired in various courses over the past few years.

Table 3.3: Gyroscope requirements

Gyroscope	
Range	125°/s
Sample rate	1 samples/h
Resolution	0.004°/s

Table 3.4: Weather station requirements

Weather station			
	Range	Resolution	Sample rate
Rain gauge	300 mm	1 mm	
Anemometer for wind speed	50 m/s	0.5 m/s	1 sample/s
Anemometer for wind direction	360°	22.5°	1 samples/s
Hygrometer	100%	1%	2 samples/h
Thermometer	−30°/85°	1°	2 samples/h

The complete system involves a number of BridgeWatch boards, which will need to be placed in strategic locations to be able to capture accelerations useful for our purposes, and a single EnvironMonitor board. The Raspberry Pi acts as the overall coordinator: it communicates with the microcontrollers of BridgeWatch through an RS485 bus and with the microcontroller of EnvironMonitor through an RS232 bus to receive data from all the boards, it processes the data, and finally send the useful information to a server. In addition, the system is powered by a 12V battery that is recharged by solar panels, eliminating the need for an electrical outlet.

The scheme of the system operation is shown in figure 3.1. In brief:

- The BridgeWatch boards are connected in series to form the RS485 bus, with one cable used for both power supply and data transmission. The first board of the bus has the power supply cables connected directly to the battery, and the data transmission cables connected to Raspberry Pi.

- The Raspberry Pi is powered by EnvironMonitor through a USBC-USBA cable, which in turn is directly connected to the battery power supply.

- An RS232-to-USB and an RS485-to-USB converter are used to allow the communication between the two board with Raspberry Pi.

- Voltage levels are adjusted for power and data transmission using special transceivers and voltage regulators.

- The Raspberry is also equipped with a GPS module, so that the exact date and time can always be accessed. It will be crucial for placing data in the correct time frame and correlating it, as will be explained later.

Table 3.5: Water level sensor requirements

Water level sensor	
Range	7 m
Sample rate	20 samples/s
Resolution	3 cm

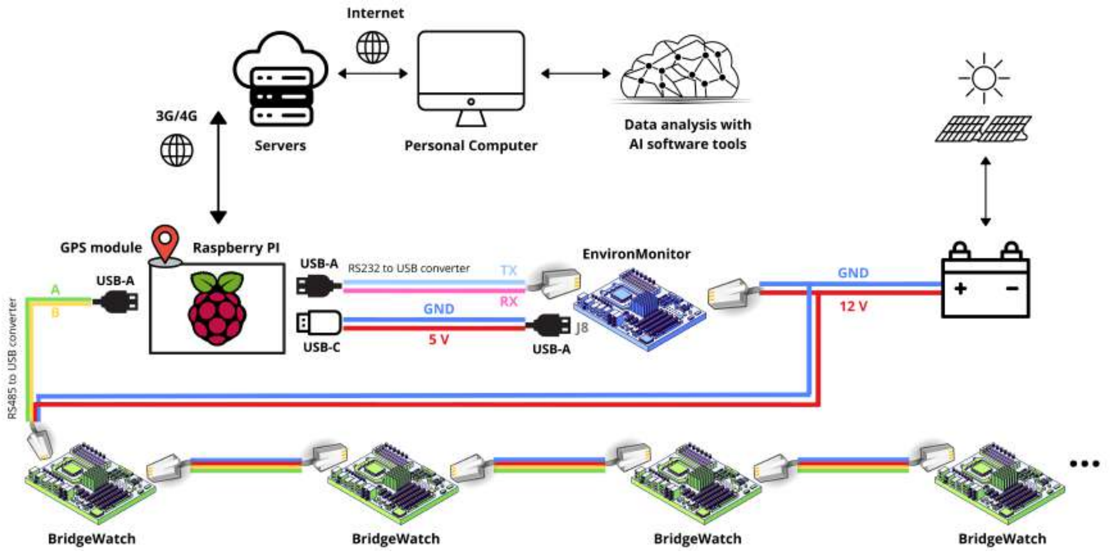


Figure 3.1: Scheme of the proposed monitoring system

3.2.1 Choice of the sampling frequency

A requirement is that the system must be specifically developed to monitor concrete bridges. These structures typically exhibit lower natural frequencies compared to other types of bridges, such as suspension or cable-stayed bridges. Consequently, the monitoring system should be capable of accurately detecting and analyzing lower-frequency vibrations and movements. Concrete bridges typically have natural frequencies in the range of 0.5 Hz to 2 Hz, depending on their size, design, and materials. The Nyquist-Shannon theorem gives us a starting point for determining the required sampling frequency.

Nyquist-Shannon theorem

The Nyquist theorem, also known as the Nyquist-Shannon sampling theorem, is a fundamental principle in signal processing and digital communication. It provides a criterion for determining the minimum sampling rate required to accurately capture and reconstruct a continuous signal without introducing aliasing. The Nyquist theorem states that a continuous signal can be completely and accurately represented by its samples if it is sampled at a rate greater than twice its highest frequency component. This rate is known as the Nyquist rate.

When a signal is sampled, it is converted from a continuous-time signal to a discrete-time signal. If the sampling rate is lower than the Nyquist rate, higher frequency components of the signal will be misrepresented or "aliased" into lower frequencies. This can cause distortions and loss of information. Aliasing occurs because the sampled data points are not sufficient to accurately reconstruct the original continuous signal. This effect causes overlapping of frequency components, resulting in artifacts in the digital representation.

The theorem relies on the concept of bandlimited signals, which means the signal contains no frequencies higher than a certain maximum frequency f_{max}. According to the Fourier transform, such signals can be perfectly reconstructed from their samples if sampled at or above the Nyquist rate.

According to the Nyquist theorem, a reasonable sampling rate would be at least 4 Hz to 10 Hz. However, to ensure a more detailed analysis and account for additional dynamics, it is often recommended to sample at a rate that is 5 to 10 times higher than the highest frequency of interest. Therefore, a sampling rate in the range of 20 Hz to 50 Hz would be appropriate for capturing the necessary data on masonry bridges. At the same time, the number of samples must be a power of 2 to perform the FFT. Since the higher the sampling frequency the better, we chose a frequency of 256 Hz, which is a compromise between high frequency resolution and a not too high number of data to send.

3.3 Preliminary cost evaluation and choice of a more affordable solution

The initial idea was to design the system so that it could really be used to monitor typical Italian bridges, i.e. 2 or 3 spans and concrete deck. In Fig.3.2 there is a schematic representation of the final system in use on the bridge (not to scale, of course), as we imagine it to be:

- 4 BridgeWatch boards are placed on each arch, two on one side and two on the other (at 1/3 and 2/3 of the arch's lenght), in order to catch up to the second mode of vibration.
- 1 BridgeWatch is placed on each pier, in order to measure rotation and inclination.
- They are all connected to the RS485 bus with a cable for both data transmission and power supply.
- A central pole holds the weather station, solar panel and a box containing the EnvironMonitor board, battery and Raspberry.
- An arm to which the radar for measuring the height of the water is also attached to the pole.

As far as attaching the BridgeWatch to the bridge is concerned, the idea is to attach it to a metal plate protected by an IP67 box and then glue the plate to the bridge.

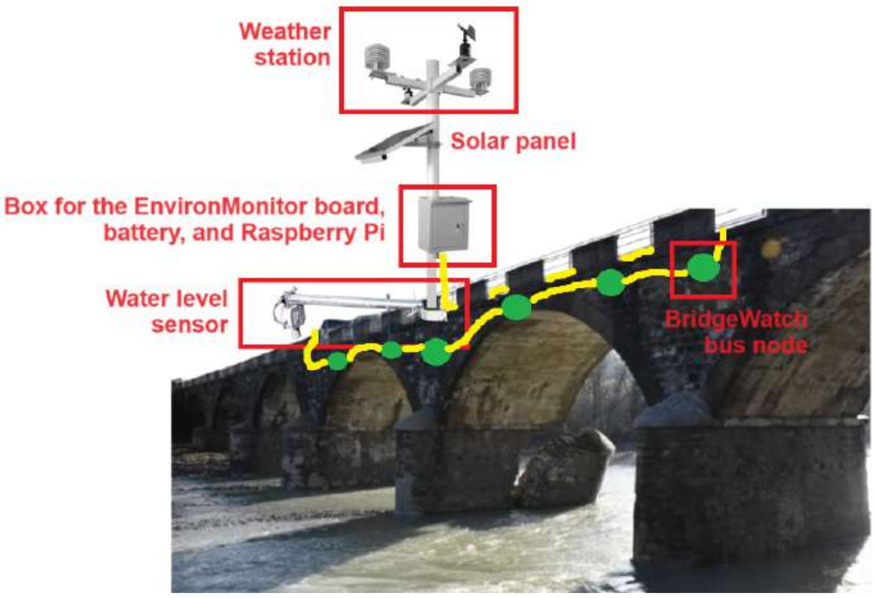


Figure 3.2: Initial idea of the complete system

However, the cost of such a project was too high to be covered by the available budget. In fact, in addition to the cost of purchasing all the sensors, there was the cost of purchasing all the necessary components and printing of the 10 BridgeWatch units, the manual cost of installing the system on the bridge, and the purchase of the pole and other mechanical supports such as the boxes and metal plates.

For this reason, we decided to design a system that was a prototype of the real one, and therefore did not need to be applied directly to the bridge, but only tested in the laboratory. In particular, it was decided not to purchase the rain gauge and the bridge water level sensor, which was the most expensive, and to reduce the number of BridgeWatch boards used from 10 to 4 (the minimum number required to perform the modal shape extraction during the laboratory test). In any case, the system was designed to include inputs for sensors that don't yet exist, but can be purchased and connected to the others in the future. The system developed can therefore be seen as a prototype, but it is ready to be used on real bridge cases. All that is needed is to increase the number of BridgeWatch boards and to connect the rain gauge and the bridge water level sensor to the EnvironMonitor board.

Chapter 4

Sensors

4.1 Accelerometer

Vibration-based sensing involves the use of accelerometers to measure the response of the studied structure under some excitation, aiming to determine its modal parameters. Since the performance of accelerometers directly impacts the quality of the collected data and the subsequent results, selecting the appropriate type of sensor is crucial.

After a thorough literature review, the choice fell on the use of MEMS technology. This is mainly due to their advantages of miniaturisation, lower costs, low power consumption and ease of integration. The latter, in particular, was crucial as it is the only type of accelerometer that could be soldered onto a PCB, allowing a single multi-sensor PCB to be realised (as we want to do in our project). This was not a must, but from the very first conception of the system, I preferred to have all the sensors integrated on a single board, so as to be able to use the communication protocols that I had studied over the years, and also to make the system nodes more compact and easier to install (better to install a board with all the sensors on it, rather than each one isolated).

The choice was then supported by many readings discussing its benefits. Cigada et al. 2007, for example, carried out a complete mechanical characterization of some MEMS sensors in comparison with other traditional accelerometers (piezoelectric and servo ones). The obtained results allow to conclude that quality/cost ratio of this kind of sensor is very high: they can be tens times cheaper than traditional accelerometers and performances can be compared. Bassoli et al. 2015 reported the results of a dynamic tests performed on a bell tower located in Ficarolo (Italy). They acquired accelerations using 11 biaxial MEMS units and managed to identify the first 8 mode shapes with good accuracy and clarity of results. They also carried out a comparison between the performance of the installed MEMS-based system and a traditional analog (piezoelectric) system and found that there were no significant differences in the modal parameters obtained.

It can be found studies about the applicability of commercially available low-cost MEMS accelerometers for structural monitoring purposes involving real situations. It can be cited the works of D'Alessando and Scudero 2021, C. Bedon et al. 2018 and Girolami et al. 2017 as examples of the successful use of low-cost MEMS accelerometers for such purposes.

The choice of MEMS accelerometers is therefore driven by the practical benefits of installation and integration, as well as their proven technical advantages, making them an ideal solution for modern SHM applications.

The selected accelerometer was the ADXL355 digital MEMS accelerometer, developed by Analog Devices and shown in figure 4.1. The ADXL355 is a low power, 3-axis accelerometer with selectable measurement ranges between ±2 g, ±4 g, and ±8 g. It integrates a 20 bits sigma-delta ADC per axis, corresponding to 3.9μg/LSB, 7.8μg/LSB and 15.6μg/LSB sensitivities, with respect to the defined measurement ranges. It also has a noise density of 22.5μg/$\sqrt{\text{Hz}}$, and a bandwidth of 1000 Hz. It requires a voltage supply from 2.25 to 3.6 V.

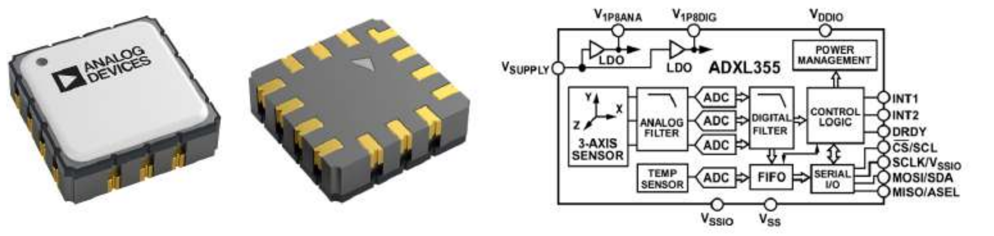


Figure 4.1: ADXL355. Source: *ADXL354/ADXL355 Low Noise, Low Drift, Low Power, 3-Axis MEMS Accelerometers Data Sheet* n.d.

It utilizes fully differential micromachined sensors to measure acceleration in the x, y, and z axes, with distinct signal paths to minimize offset drift and noise in the measurements. This enhances the accuracy and reliability of the accelerometers in vibration and structural health monitoring applications.

Three high-resolution analog-to-digital converters (ADCs) reference the analog 1.8 V supply generated by the internal low dropout (LDO) regulators, ensuring digital outputs are unaffected by supply voltage variations. Noise is further reduced by applying antialiasing filters both before and after these ADCs. The output data rates, ranging from 4 kHz to 3.9 Hz, and filter corner frequencies can be adjusted through register settings, allowing for customization based on specific application needs. It supports both SPI and I2C interfaces, and has an operating temperature range from -40 to 125°C. An important feature is that it can also function as an inclinometer, as it has the bandwidth starting at 0 Hz, so it can measure static accelerations due to gravity.

4.2 Gyroscope

The use of gyroscopes makes it possible to integrate rotations with data from the accelerometer, providing more accurate movement data. A first parameter used to 'skim' the choice of gyroscope was to find one that could be integrated on a PCB, like the accelerometer. This was because the idea from the outset was to design a PCB from scratch that would house the various structural sensors in order to have a compact and easy-to-install node. This means that MEMS technology will be used here too.

The selected gyroscope is the I3G4250D digital MEMS gyroscope, developed by ST Microelectronics and shown in figure 4.2.

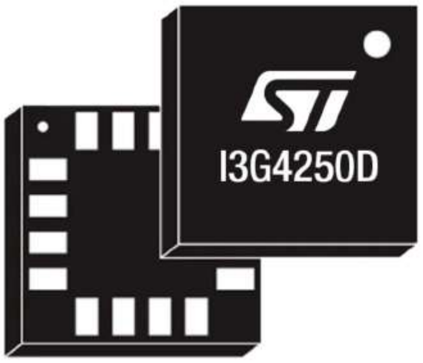


Figure 4.2: I3G4250D. Source: *I3G4250D MEMS Motion Sensor: 3-axis Digital Output Gyroscope* n.d.

The I3G4250D is a low-power, 3-axis angular rate sensor known for its exceptional stability in zero-rate conditions and consistent sensitivity across varying temperatures and over time. It comprises a sensing element and an integrated circuit (IC) interface that transmits the measured angular rate to the application via a standard SPI digital interface. Additionally, it offers compatibility with an I2C interface. The I3G4250D has a selectable full scale (±245/±500/±2000 dps) and is capable of measuring rates with a user-selectable bandwidth. It integrates a 16 bits ADC per axis, so the lowest measurement range corresponds to a sensitivity of 8.75 mdps (milli-degree per second)/digit. It requires a supply voltage from 2.4 V to 3.6 V. The block diagram is in figure 4.3. It features micromachined sensing elements that detect angular rates along x, y, and z axes. Signals from these elements are amplified, mixed, and filtered to remove noise. The ADCs digitize the signals and then digital filtering ensures accuracy and stability.

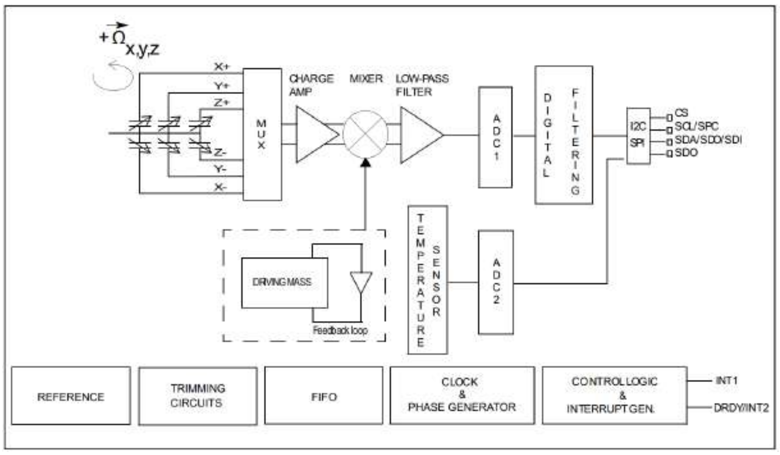


Figure 4.3: I3G4250D block diagram. Source: *I3G4250D MEMS Motion Sensor: 3-axis Digital Output Gyroscope* n.d.

4.3 Temperature sensor for the structure

A temperature sensor must be used in conjunction with structural sensors in a Structural Health Monitoring system to fully analyse the data and provide an accurate health assessment. In fact the physical properties of materials, such as strength and stiffness, can be strongly affected by temperature changes in the structure. These changes can then affect the structural responses that accelerometers monitor. The ability to differentiate between temperature-induced changes and those resulting from true structural problems is made possible by this dual-sensor technique, which increases the reliability of the SHM system in identifying and predicting potential problems.

For the system, an RTD PT100 sensor has been used. The RTD PT100 sensor is a widely used temperature sensor in SHM systems due to its accuracy and stability. It contains an internal electrical resistance whose value changes with temperature (the sensor itself is therefore also called a 'thermoresistor'). More precisely, the value of the internal resistance increases as the temperature rises: the sensor is said to have a positive temperature coefficient, indicated by the abbreviation PT (acronym of the English term 'Positive Temperature'). The number '100' in the name of the sensor indicates that the sensor has a nominal internal resistance value of 100 Ohm at a temperature of 0 °C, as specified in the IEC 751 (EN 60751) standard. There are many advantages to using a PT100 sensor, mainly due to the chemical and physical properties of platinum and the techniques used to manufacture it. These include a very wide measurable temperature range, an almost perfectly linear characteristic, high accuracy and interchangeability, stability, robustness and longevity.

RTD sensors are classified into different classes based on their accuracy and tolerance. In particular, four classes of tolerance have been defined as follows:

- Class AA: tolerance of $\pm$ (0.1 + 0.0017 $\times$ |t|) °C;
- Class A: tolerance of $\pm$ (0.15 + 0.0017 $\times$ |t|) °C;
- Class B: tolerance of $\pm$ (0.3 + 0.0017 $\times$ |t|) °C;
- Class C: tolerance of $\pm$ (0.6 + 0.0017 $\times$ |t|) °C.

The selected sensor is in Fig.4.4. It is in the form of a probe in which the sensor is integrated with the addition of electrical cables for connection, a protective insulating sheath and a connection head. It is in Class B, which offers a good balance of accuracy and cost-effectiveness. The sensor has a 4-wire connection, which completely eliminates the influence of the cable on the measurement result by compensating for possible asymmetries in the resistance of the connecting cable.

To simplify things and avoid the need to create a conditioning circuit, it was decided to use an existing chip that performs the function of measuring the resistance of the RTD and converting it into a digital temperature output. The chip in question is the MAX31865, which has an SPI interface.

Other factors that can have a significant impact on structural response and require

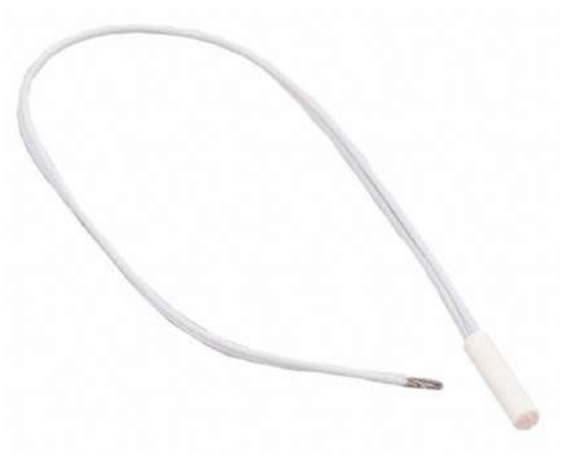


Figure 4.4: PT100 sensor. **Manufacturer:** Innovative Sensor Technology. **Manufacturer code:** P0K1.281.2K.B.150.R.S. Source: *Platinum sensor with round ceramic housing for low temperatures* n.d.

monitoring include wind and rain. Wind can induce dynamic loads and vibrations, changing the stress distribution and potentially leading to fatigue or damage over time. Rain and moisture can affect the material properties and weight distribution of a structure, causing changes in load bearing capacity and increasing the risk of corrosion or water ingress. By integrating environmental condition sensors, engineers can gain a more complete understanding of the external forces acting on the structure and how these forces interact with the inherent dynamic properties measured by accelerometers. This holistic approach improves the accuracy of health assessments and helps to make more informed decisions about maintenance and safety measures.

4.4 Anemometer

The chosen anemometer is the SKU 7911 by Davis Instrument and is represented in Fig.4.5. It includes both wind speed and wind direction sensors: the former is measured with a solid-state magnetic sensor, the latter with a wind vane and potentiometer.

The anemometer can measure speeds from 0.5 to 89 ± 1 m/s, and a direction from 0° to 360° ± 7° with 16 compass points. The resolution is 0.1 m/s for wind speed and 22.5° for wind direction. It has 4 output cables to be connected:

- Black: Wind speed contact closure to ground
- Red: Ground
- Green: Wind direction pot wiper (20KΩ potentiometer)
- Yellow: Potentiometer supply voltage

The outputs are one pulse per revolution of the wind vane, and a variable resistance 0 - 20KΩ for the direction, where 10KΩ corresponds to 180°.

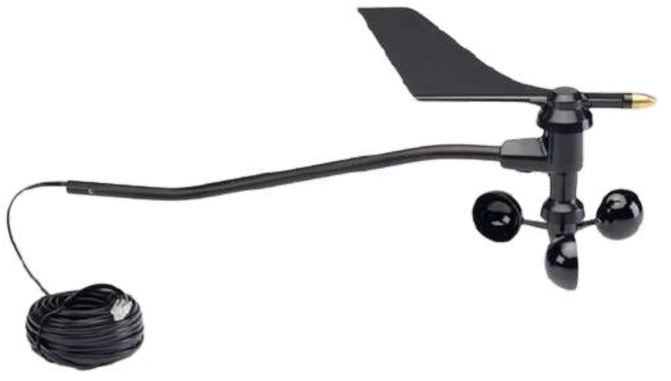


Figure 4.5: Anemometer SKU 7911. Source: *Anemometer for Weather Monitor or Wizard - SKU 7911* n.d.

4.5 Rain gauge

The selected rain gauge is the SKU 6464 from Davis Instrument, in Fig.4.6. It is designed such that rain enters the collector cone, passes through a debris-filtering screen, and collects in the tipping spoon. The spoon tips when it has collected an amount of water equal to 0.2 mm. As the spoon tips, it causes a switch closure. The rain water drains out through the screened drains in the base of the collector. The shape is aerodynamically designed to minimize rainfall catch reduction caused by high winds. The body and base of the collector are constructed of tough, UV resistant plastic. It has 4 output cable to be connected:

- Red: Switch terminal
- Green and Yellow: Switch terminal
- Black: Unused

4.6 Thermohygrometer

The selected sensor is the AM2315C, represented in Fig.4.7. This sensor features a DS18B20 temperature sensor and a MEMS capacitive humidity sensor, with an internal microcontroller that handles readings and outputs calibrated data via I2C. I has typical accuracy for the humidity of ±2%, and of ±0.3°C for temperature. It has a wide voltage support 2.2 to 5.5V DC, and excellent long-term stability.

There are 4 output cable to be connected:

- Red: Power supply
- Black: Ground

Figure 4.6: Rain gauge SKU 6464. Source: *AeroCone Rain Collector with Flat Base for Vantage Pro2 and EnviroMonitor (tipping spoon) - SKU 6464, 6464M* n.d.

- Yellow: SDA (I2C Data line)
- White: SCL (I2C Clock line)

4.7 Water level sensor

Including a river water level sensor in a SHM system is crucial since the water level of the river can have a significant impact on the structural integrity and safety of the bridge. High water levels can indicate potential flooding, which can increase the load on the bridge's foundations and supports, leading to stress and possible damage. Moreover, fluctuating water levels can affect the soil stability around the bridge's foundations, further influencing its overall stability.

The chosen sensor is VEGAPULS C 23 by VEGA, in Fig.4.8. It is a radar sensor with integrated cable for continuous level measurement of liquids. The range is 30m, with an accuracy of ±2mm. It has a current output ranging from 4 to 20mA, and therefore it will need a conditioning circuit to transform this current in voltage.

Figure 4.7: Thermohygrometer AM2315C. Source: *Data Sheet AM2315C Humidity and Temperature Module* n.d.

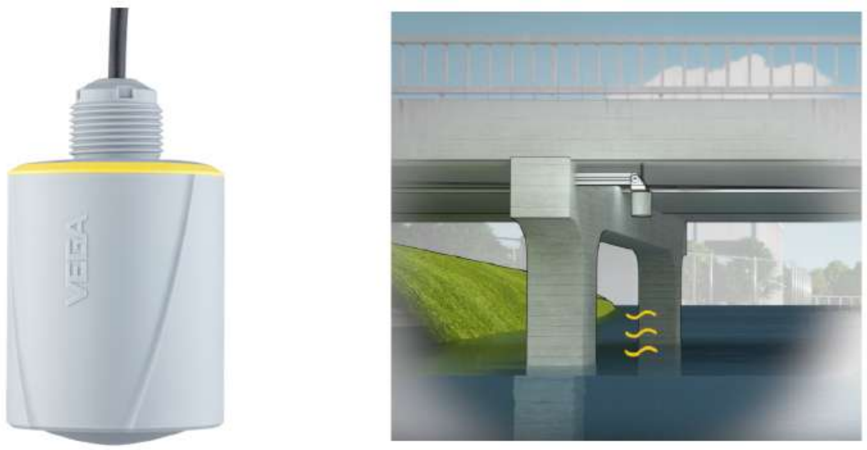


Figure 4.8: Water level sensor VEGAPULS C 23. Source: *VEGAPULS C 23 Sensore radar con cavo integrato per la misura di livello continua su liquidi* n.d.

Chapter 5

Hardware and Schematics

Schematics and layouts have been made with Altium Designer (*Altium Designer 24* n.d.), a leading software for electronic design automation (EDA).

As mentioned above, it was decided to make two different boards: one containing the structural sensors (BridgeWatch), i.e. the accelerometer and the gyroscope, plus the thermometer for temperature correction. This layout will form the nodes of the structural monitoring system. The second board, instead, acts as a weather station (EnvironMonitor), and contains all the inputs of the environmental sensors (exception made for the thermometer for structure, which, as said, is installed in every single BridgeWatch board).

The full schematics are shown in Appendix A and B, first for the BridgeWatch board and then for the EnvironMonitor. We will now analyse them in detail, piece by piece, and explain the reasons for each choice.

5.1 BridgeWatch

The full schematics of the board are shown in Appendix A. All symbols and part numbers used in the following sections refer to these schematics.

5.1.1 Power supply and RS485 communication

In figure 5.5 is reported the section of the schematic related to power supply and communication. In brief:

- J2 and J3 are the 4-wire input and output connectors. A 4-wire cable connects J3 of board *i* to J2 of board *i+1*. Pin 1 and 2 are GND and 12 V for the power supply, while pin 3 and 4 are for the two differential signals of the RS485 communication bus.
- U4 is the DC-DC converter. It leads from the battery voltage (12V) to the digital circuit voltage (3.3V), with proper protection circuit.
- U5 is the RS485 transceiver. It is used to convert the electrical level of the USART of the microcontroller (TTL) to the electrical level required by the RS485 bus.

DC-DC converter

The DC-DC converter, as just mentioned, is needed to convert the battery voltage of 12 V to the supply voltage that will be used by all the main chips on the board, which is 3.3 V. The selected component is the following:

DigiKey Part Number: 102-4243-ND
Manufacturer: CUI Inc.
Manufacturer Product Number: VX7803-500

Figure 5.1: DC-DC converter. Source: *VX7803-500 - Digikey* n.d.

It has an efficiency of 86%, and an operating temperature from -40°C to +85°C. A DC-DC converter must always have a protection circuit on its inputs. This is essential because it prevents damage to the converter and connected equipment by clamping excessive voltage levels that could result from faults or sudden changes in input conditions, and also ensures that the converter shuts down or limits the current in the event of a short circuit, preventing damage to both the converter and the load. Overall, these protections enhance the reliability, safety, and longevity of the DC-DC converter and the devices it powers. The schematic is directly inspired by the one proposed in the datasheet, shown in Fig.5.2. L1, L2, C18, C19 make the pi filter, the PTC R2 acts as the fuse, the TVS D2 acts as the MOV, and the diode D1 is a protection against the reversal of battery polarity. The values of the various components are also present in the datasheet.

A pi filter is often used at the input of a DC-DC converter to smooth out voltage variations and reduce noise from the power supply. It consists of a series inductor sandwiched between two shunt capacitors, forming a topology that resembles the Greek letter "π". This filter primarily serves to mitigate high-frequency noise and ripple voltage that might be present on the input power line. By doing so, it ensures a cleaner and more stable DC voltage is provided to the converter, enhancing its performance and efficiency. The series inductor blocks high-frequency signals, while the capacitors shunt them to ground, creating a low-pass filter effect.

A TVS diode is an electronic component designed to protect electronics from voltage spikes that occur on connected lines. The TVS diode works by diverting excess current when the induced voltage exceeds its avalanche breakdown threshold. Acting as a clamping device, it limits any overvoltages above this breakdown voltage. It resets automatically when the overvoltage subsides.

RS485 transceiver

A TTL to RS485 transceiver works by translating the voltage levels used in TTL (transistor-transistor logic) to those used in RS485. TTL values, typically found in microcontrollers

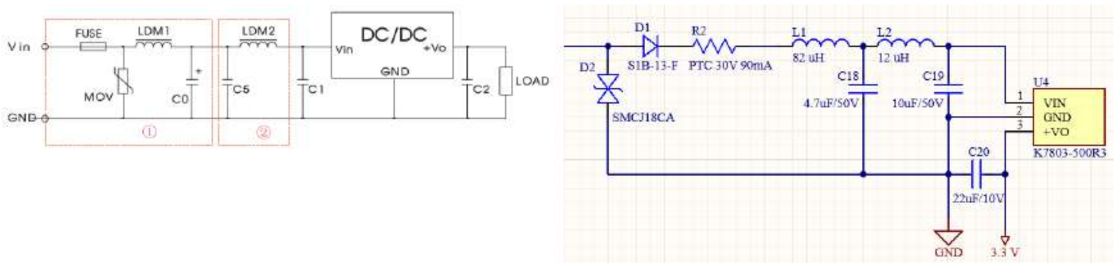


Figure 5.2: Datasheet and schematic of the DC-DC converter

and other digital logic circuits, operate at voltage levels of 0 to 5 volts (for standard TTL) or 0 to 3.3 volts (for low voltage TTL). These levels are suitable for short distance communication within electronic devices. However, for longer distances and more robust communication, RS485 is preferred. RS485 uses differential signalling with balanced wires, providing greater noise immunity and the ability to communicate over distances up to 1,200 metres. The transceiver translates the single-ended TTL signals to the differential RS485 signals and vice versa, enabling seamless communication between a microcontroller's TTL interface and an RS485 network. So, it is crucial to allow the microcontroller to send data on the RS485 bus.

The selected component is the following:

DigiKey Part Number: 296-THVD1400DRCT-ND
Manufacturer: Texas Instruments
Manufacturer Product Number: THVD1400DR

Figure 5.3: RS485 transceiver. Source: *THVD1400DR - Digikey* n.d.

As it can be seen, it has 8 pin whose functions are explained in table 5.1.

The schematic exactly follows the one in the datasheet, as it is shown in Fig.5.6. Pin 1 and 4 will be then connected to the USART Transmitter and Receiver of the microcontroller, while pin 2 and 3 (connected toghether since they are complementary) will be connected to a microcontroller pin configured as digital output. The typical application, i.e. an RS-485 bus consisting of several transceivers connected in parallel on a bus cable, is shown in Fig.5.4. To eliminate line reflections, each end of the cable is terminated with a terminating resistor, RT, whose value corresponds to the characteristic impedance, Z0, of the cable. For more details about the RS485 protocol refer to Cap.7.

Table 5.1: Transceiver pin functions

NAME		I/O	DESCRIPTION
R	1	**Digital output**	Receive data output
RE	2	**Digital input**	Receiver enable, active low
DE	3	**Digital input**	Driver enable, active high
D	4	**Digital input**	Driver data input
GND	5	**Ground**	Device ground
A	6	**Bus input/output**	Bus I/O port, A (complementary to B)
B	7	**Bus input/output**	Bus I/O port, B (complementary to A)
V_{CC}	8	**Power**	3.3-V to 5-V supply

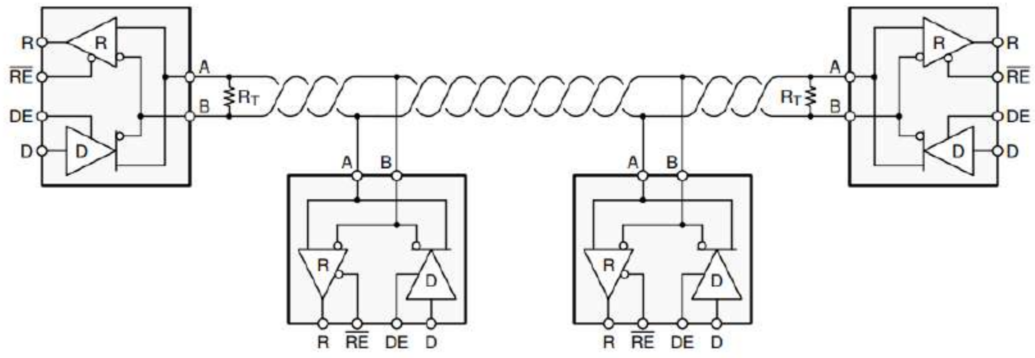


Figure 5.4: Transceiver typical application

5.1.2 Peripherals

Accelerometer

The accelerometer pin configuration and pin description are respectively in Fig. 5.7 and table 5.2.

Pin No.	**Mnemonic**	**Description**
1	CS/SCL	Chip Select for SPI (CS).
2	SCLK/V_{SSIO}	Serial Communications Clock for I^2C (SCL). Serial Communications Clock for SPI (SCLK).
3	MOSI/SDA	Master Output, Slave Input for SPI (MOSI). Serial Data for I^2C (SDA).
4	MISO/ASEL	Master Input, Slave Output for SPI (MISO). Alternate I^2C Address Select for I^2C (ASEL).
5	V_{DDIO}	Digital Interface Supply Voltage.
6	V_{SSIO}	Digital Ground.

7	RESERVED	Reserved. This pin can be connected to ground or left open.
8	V_{1P8DIG}	Digital Supply. This pin requires a decoupling capacitor. If V_{SUPPLY} connects to V_{SS}, supply the voltage to this pin externally.
9	V_{SS}	Analog Ground.
10	V_{1P8ANA}	Analog Supply. This pin requires a decoupling capacitor. If V_{SUPPLY} connects to V_{SS}, supply the voltage to this pin externally.
11	V_{SUPPLY}	Supply Voltage. When V_{SUPPLY} equals 2.25 V to 3.6 V, V_{SUPPLY} enables the internal LDO regulators to generate V_{1P8DIG} and V_{1P8ANA}. For $V_{SUPPLY} = V_{SS}$, V_{1P8DIG} and V_{1P8ANA} are externally supplied.
12	INT1	Interrupt Pin 1.
13	INT2	Interrupt Pin 2.
14	DRDY	Data Ready Pin.

Table 5.2: ADXL355 Pin Function Descriptions

It was decided to use the SPI interface for communication, so the dedicated pins (from 1 to 4) were connected to the corresponding pins on the microcontroller. On the other hand, pins from 12 to 14 are sensor's outputs and therefore have to be connected to digital inputs of the microcontroller. With regard to the part of the power supply, the accelerometer has been fitted with the various decoupling and bypass capacitors according to their datasheet, as represented in Fig. 5.8.

Gyroscope

The gyroscope's pin configuration and pin description can be found in Figure 5.9 and in the table 5.3, respectively.

Also in this case the decision was made to utilize the SPI interface for communication, leading to the connection of the dedicated pins (pins 2 to 5) to the corresponding pins on the microcontroller. Conversely, the pins 6 and 7 serve as sensor outputs and therefore need to be connected to two digital inputs of the microcontroller. Regarding the power supply section, the gyroscope has been equipped with the necessary decoupling and bypass capacitors as specified in its datasheet, as illustrated in Figure 5.10.

For more details about SPI protocol, refer to Cap. 7.

General purpose LED

It was decided to use a green surface-mounted LED, and in particular:

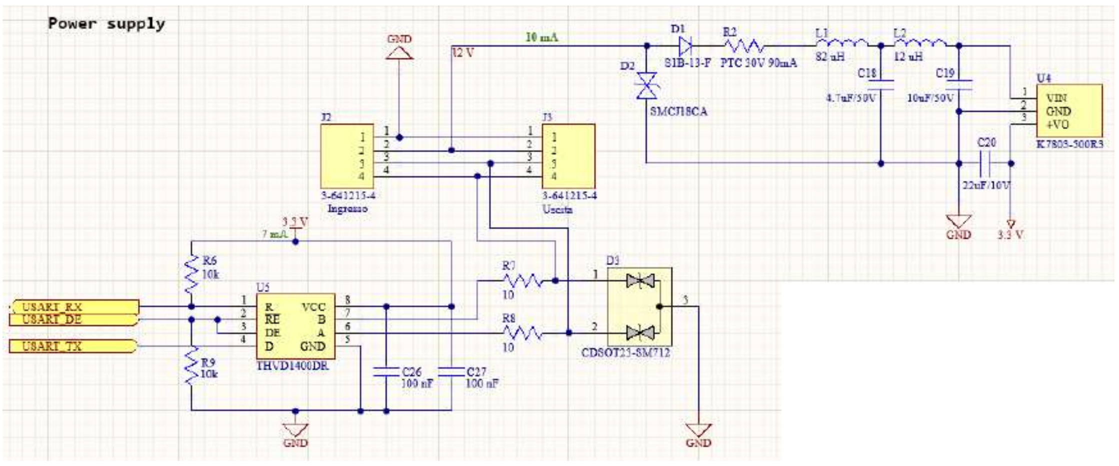


Figure 5.5: Power and RS485 schematic

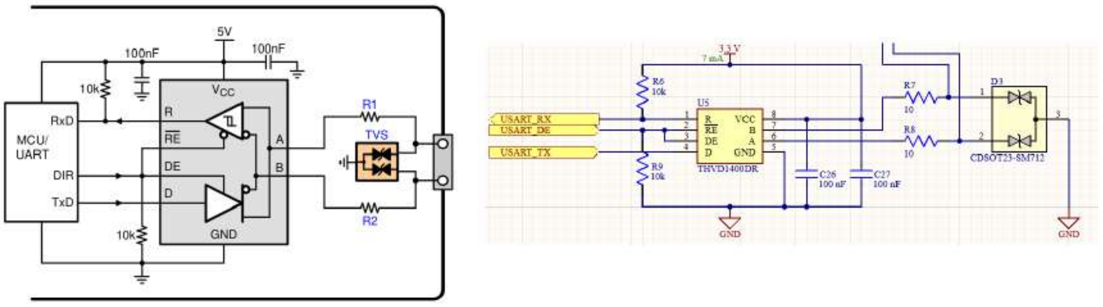


Figure 5.6: Datasheet and schematic of the RS485 transceiver

DigiKey Part Number: 732-4986-1-ND
Manufacturer: Würth Elektronik
Manufacturer Product Number: 150080VS75000

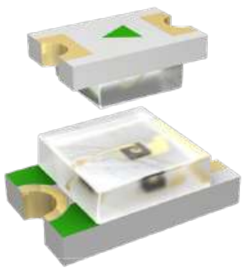


Figure 5.11: Green LED. Source: *150080VS75000 - Digikey* n.d.

The LED is in pull-up configuration: the anode is connected to an output pin on the microcontroller, which then controls the LED, while the cathode is connected to ground with a resistor between the two. R3 is necessary to limit current. The resistor value R can be calculated using Ohm's Law with the formula:

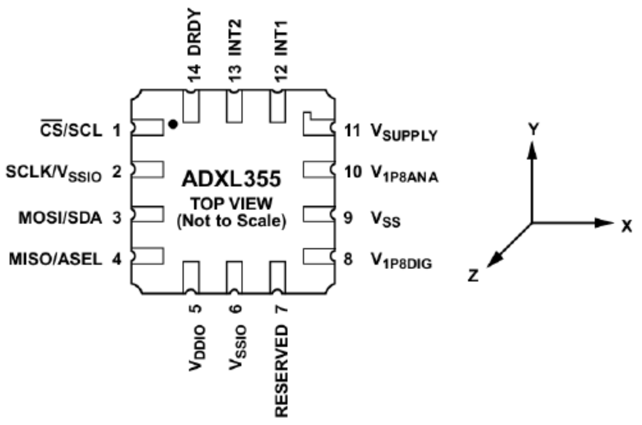


Figure 5.7: Accelerometer pin configuration. Source: *ADXL354/ADXL355 Low Noise, Low Drift, Low Power, 3-Axis MEMS Accelerometers Data Sheet* n.d.

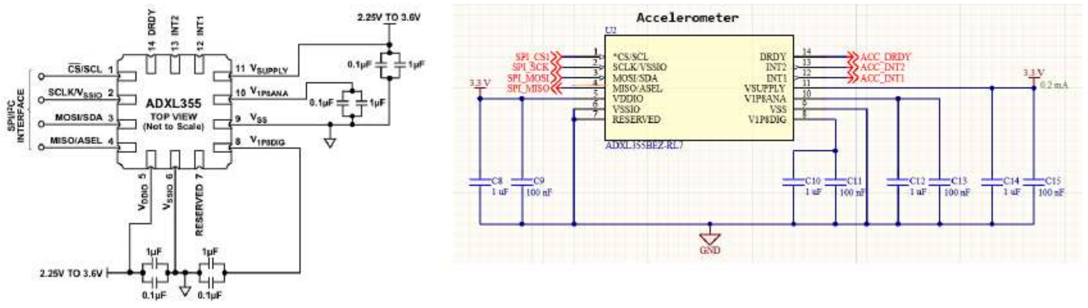


Figure 5.8: Datasheet and schematic of the accelerometer

$$R = \frac{V_{CC} - V_{LED}}{I_{LED}}$$

where:

- V_{CC} is the supply voltage, 3.3 V in our case
- I_{LED} is the desired current through the LED, 10 mA in our case
- V_{LED} is the forward voltage of the LED. This is the voltage drop across the LED when it is turned on. It can be derived from the forward current vs. forward voltage graph in the datasheet, as shown in Fig. 5.12.

RTD-to-digital converter

As said before, an RTD is a temperature sensor that works on the principle that its resistance changes with temperature. In our case, the RTD is connected in a 4-wire

Pin#	Name	Function
1	Vdd_IO	Power supply for I/O pins
2	SCL SPC	I^2C serial clock (SCL) SPI serial port clock (SPC)
3	SDA SDI SDO	I^2C serial data (SDA) SPI serial data input (SDI) 3-wire interface serial data output (SDO)
4	SDO SA0	SPI serial data output (SDO) I^2C least significant bit of the device address (SA0)
5	CS	SPI enable I^2C/SPI mode selection (1: SPI idle mode / I^2C communication enabled; 0: SPI communication mode / I^2C disabled)
6	DRDY/INT2	Data ready/FIFO interrupt
7	INT1	Programmable interrupt
8	Reserved	Connect to GND
9	Reserved	Connect to GND
10	Reserved	Connect to GND
11	Reserved	Connect to GND
12	Reserved	Connect to GND
13	GND	0 V supply
14	PLLFILT	Phase-locked loop filter (see Figure 3)
15	Reserved	Connect to Vdd
16	Vdd	Power supply

Table 5.3: I3G4250D Pin function description

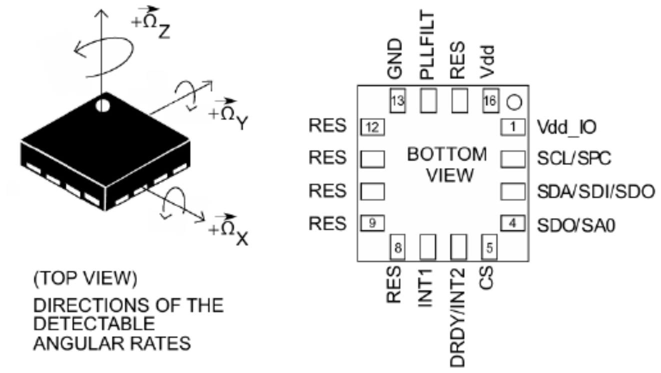


Figure 5.9: Gyroscope pin configuration. Source: *I3G4250D MEMS Motion Sensor: 3-axis Digital Output Gyroscope* n.d.

configuration to ensure accurate measurements by eliminating the effects of lead wire resistance. The resistance of the RTD changes with temperature, and the RTD-to-digital converter measures this resistance and converts it into a digital signal. The selected chip

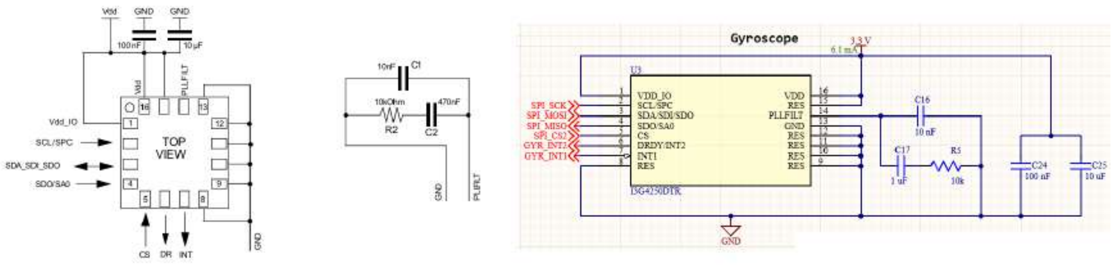


Figure 5.10: Datasheet and schematic of the gyroscope

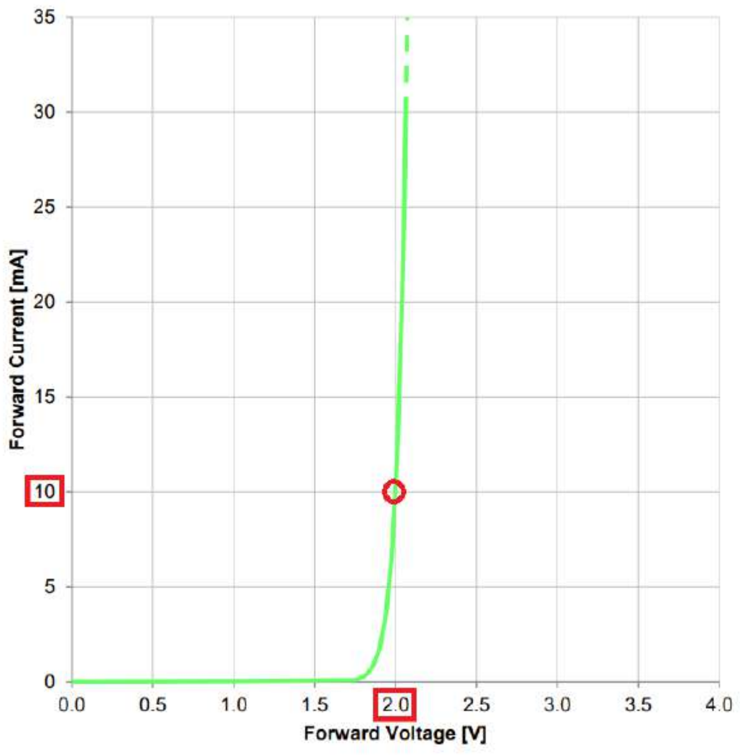


Figure 5.12: Forward current vs. forward voltage of the LED

is the following:

DigiKey Part Number: MAX31865ATP+-ND
Manufacturer: Analog Devices Inc./Maxim Integrated
Manufacturer Product Number: MAX31865ATP+

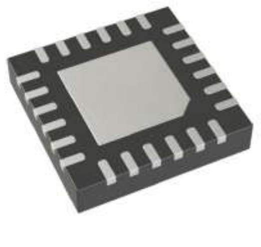


Figure 5.13: RTD-to-digital converter. Source: *MAX31865ATP+ - Digikey* n.d.

In Fig. 5.14 there is the typical application circuit, while the block diagram is in Fig. 5.15. The principle of working is the following:

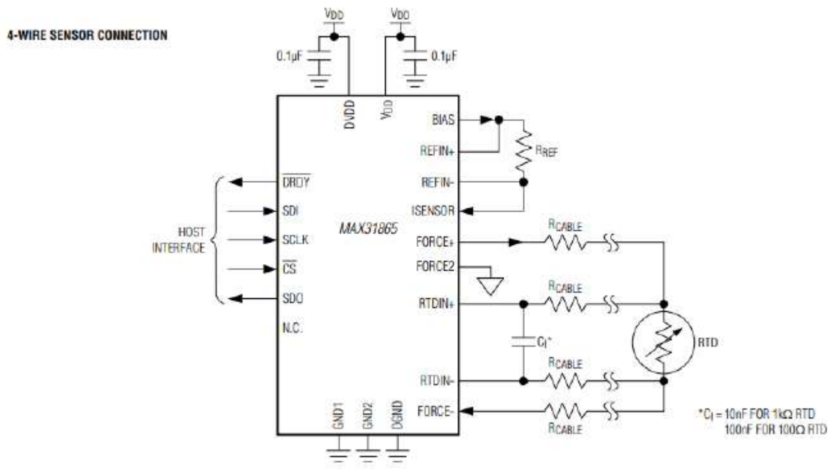


Figure 5.14: Typical application circuit of MAX31865. Source: *Datasheet MAX31865ATP+* n.d.

- The RTD and a reference resistor (R_{REF}) are connected in series.
- A bias voltage is applied to the top of the reference resistor (R_{REF}). This voltage drives the current through the series connection of R_{REF} and the RTD.
- The current flowing through the reference resistor (R_{REF}) also flows through the RTD. This is important because the current is the same for both components, ensuring that the voltage drops across them are proportional to their resistances.
- The voltage drop across R_{REF} is measured. This voltage serves as the reference voltage for the ADC (Analog-to-Digital Converter) within the MAX31865.
- The voltage drop across the RTD is applied to the differential inputs of the ADC (RTDIN+ and RTDIN-). This differential voltage is what the ADC uses to determine the RTD's resistance.

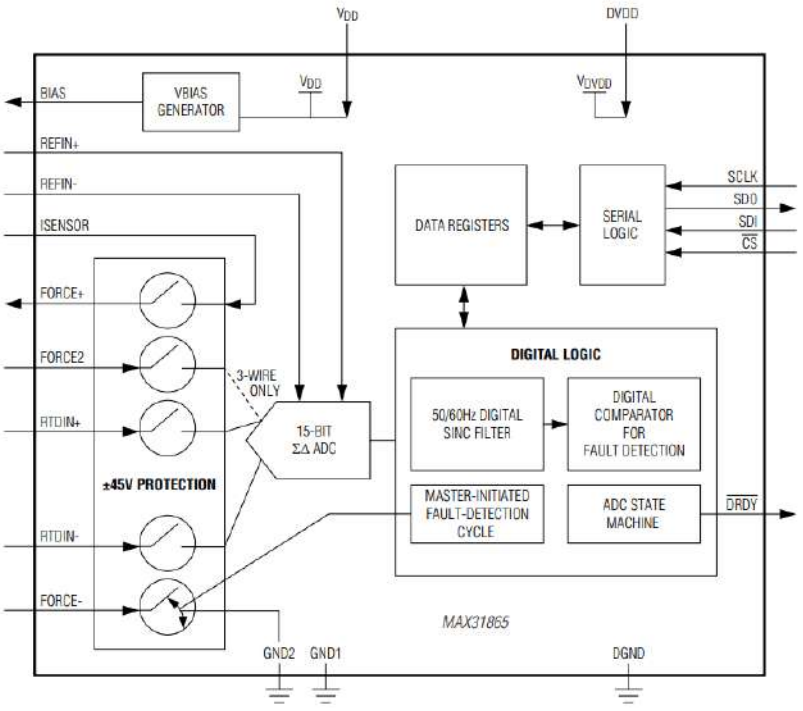


Figure 5.15: MAX31865 block diagram. Source: *Datasheet MAX31865ATP+* n.d.

- The ADC in the MAX31865 produces a digital output that represents the ratio of the RTD resistance to the reference resistance ($\mathrm{R_{REF}}$).
- Since the current through $\mathrm{R_{REF}}$ and the RTD is the same, the voltage drops across them are directly proportional to their resistances. Therefore, the ADC can determine the RTD's resistance by comparing the voltage drop across the RTD to the voltage drop across $\mathrm{R_{REF}}$.

Let I be the current flowing through both $\mathrm{R_{REF}}$ and the RTD. The voltage drop across $\mathrm{R_{REF}}$ is given by:

$$V_{\mathrm{REF}} = I \cdot R_{\mathrm{REF}}$$

Similarly, the voltage drop across the RTD is:

$$V_{\mathrm{RTD}} = I \cdot R_{\mathrm{RTD}}$$

The ADC measures these voltages and computes the ratio of the voltage drop across the RTD to the voltage drop across the reference resistor:

$$\text{Digital Output} = \frac{V_{\mathrm{RTD}}}{V_{\mathrm{REF}}} = \frac{I \cdot R_{\mathrm{RTD}}}{I \cdot R_{\mathrm{REF}}} = \frac{R_{\mathrm{RTD}}}{R_{\mathrm{REF}}}$$

Since I is the same for both resistors, it cancels out in the ratio. Thus, the digital output of the ADC is directly proportional to the ratio of the RTD resistance to the reference resistance. By knowing the precise value of R_{REF}, the MAX31865 can accurately determine the RTD's resistance, which can then be converted to a temperature reading based on the RTD's known resistance-temperature characteristics. The datasheet says that a reference resistor equal to four times the RTD's 0°C resistance is optimum for a platinum RTD. Therefore, a PT100 uses a 400 Ohm reference resistor. As it needs to be an accurate and stable resistor, it was chosen to be 0.1% accurate.

The MAX3186 includes a precision delta-sigma ADC with 15-bit resolution and communicates with a host microcontroller through an SPI interface. The SDI (Serial Data In), SDO (Serial Data Out), SCLK (Serial Clock), and CS (Chip Select) pins manage the data transfer. The DRDY (Data Ready) pin indicates when the data is ready to be read, and is therefore connected to a digital input pin on the micrcontroller.

The schematic has been made by exactly following the datasheet, as reported in Fig. 5.16. The circuit is powered by two supply voltages: V_{DD} for the digital interface and V_{DDIO} for the analog components. Each power supply pin is decoupled with 0.1µF capacitors to filter out noise and ensure stable operation.

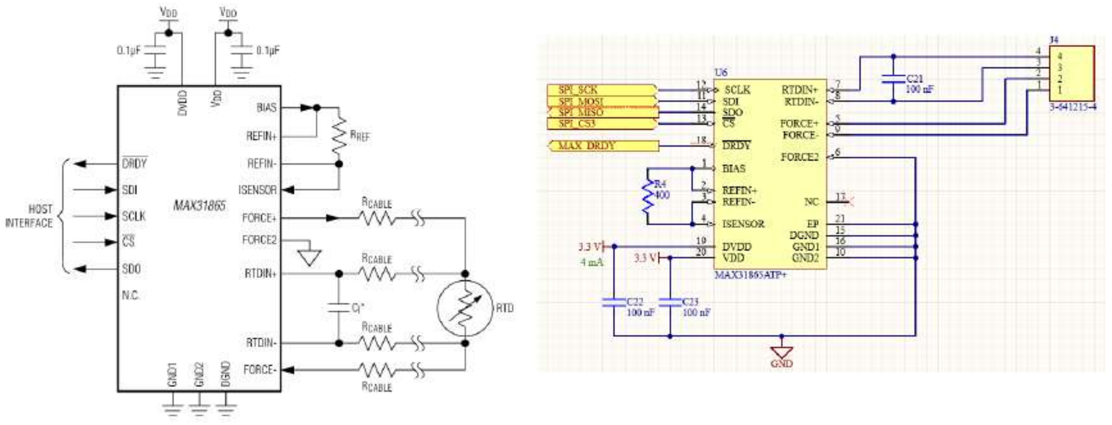


Figure 5.16: Datasheet and schematic of MAX31865

So the MAX3186 sends the measured resistance of the RTD to the microcontroller, which then has to convert it to temperature. For a PT100, the average slope between 0°C and +100°C is called alpha (α). This value depends on the impurities and their concentrations in the platinum. The resistance vs. temperature curve is reasonably linear, but has some curvature, as described by the Callendar-Van Dusen equation:

$$R(T) = R_0 \left(1 + aT + bT^2 + c(T - 100)T^3\right)$$

where:

- T = temperature (in °C)
- $R(T)$ = resistance at temperature T

- R_0 = resistance at $T = 0°\text{C}$

IEC 751 specifies $\alpha = 0.00385055$ and the following Callendar-Van Dusen coefficient values:

- $a = 3.90830 \times 10^{-3}$
- $b = -5.77500 \times 10^{-7}$
- $c = -4.18301 \times 10^{-12}$ for $-200°\text{C} < T < 0°\text{C}$, and $c = 0$ for $0°\text{C} < T < +850°\text{C}$

Since high precision in temperature measurement is not critical for our purposes, we will use the simplified approximation where the resistance-temperature relationship is assumed to be linear. Specifically, we will use the approximation $T = \alpha \cdot R$.

5.1.3 Microcontroller and JTAG

Microcontroller

The selected microcontroller is the STM32L431CBT6 by STMicroelectronics:

DigiKey Part Number: 497-17957-ND
Manufacturer: STMicroelectronics
Manufacturer Product Number: STM32L431CBT6

Figure 5.17: Microcontroller. Source: *STM32L431CBT6 - Digikey* n.d.

The STM32L431CBT6 microcontroller was chosen for the system because of its advantageous combination of high performance, low power consumption and adaptable capabilities, all of which perfectly matched the needs of our application. The STM32L431CBT6 operates at up to 80MHz, providing the processing power to handle real-time control tasks. It includes a supply voltage range of 1.71V to 3.6V for flexible power management, and its ultra-low power characteristics are essential for battery-powered or power-sensitive applications. With 64KB of RAM and 256KB of Flash memory, the microcontroller has ample space to handle data and store code. The choice of the 48-pin variant simplifies soldering and reduces board complexity. In addition, the STM32L431CBT6 has a wide range of peripherals and interfaces to enable seamless integration with sensors and other components, increasing system adaptability and efficiency. Programming is also made easy with the STM32CubeIDE (*Integrated Development Environment for STM32* n.d.), which provides an intuitive user interface and integrated configuration tools. Writing code, debugging and testing are all made easier with this development environment.

As it can be seen in Fig. 5.18, microcontroller has been equipped with its decoupling capacitors near sensitive pins exactly as indicated in the datasheet. R1 serves to indicate the microcontroller that the boot has to be done from the flash.

Oscillator

An external oscillator is connected to the microcontroller to improve synchronisation. In fact, the internal clock has an accuracy of 0.5%, and introduces too much error, leading to possible synchronisation problems and inconsistent data acquisition. On the other hand, this addition ensures that the PWM signal used by the accelerometers for sampling is highly accurate. This is critical for maintaining synchronisation between different measurement units, ensuring consistent and reliable data acquisition. The external oscillator is soldered to the dedicated Pin 3 and Pin 4 of the microcontroller.

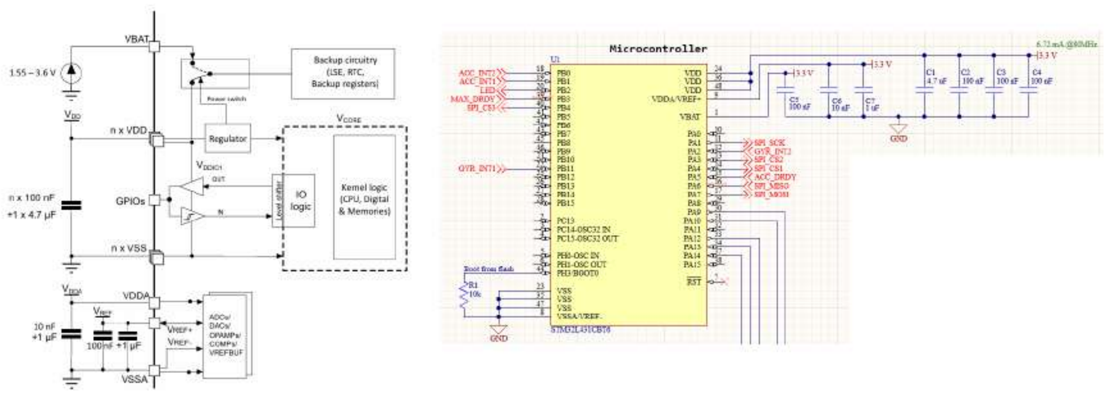


Figure 5.18: Datasheet and schematic of microcontroller

JTAG

The JTAG connector on the board serves a crucial role in debugging and programming the microcontroller. It allows for direct communication with the microcontroller for tasks such as downloading firmware, debugging code, and performing in-circuit testing. In this particular setup, a two-wire JTAG interface, utilizing only the SWDCLK (Serial Wire Debug Clock) and SWDIO (Serial Wire Debug Input/Output) lines, is employed. This simplified interface, known as Serial Wire Debug (SWD), is chosen because it provides all the necessary functionality for programming and debugging without the need for additional JTAG pins.

5.2 EnvironMonitor

The full schematics of the board are shown in Appendix B. All symbols and part numbers used in the following sections refer to these schematics.

5.2.1 Power supply

Board power supply

In figure 5.19 is reported the section of the schematic related to power supply. In brief:

- J1 is the 2-wire input connector for power supply, directly connected to the 12V battery voltage and GND.

- U3 is DC-DC converter. It leads from the battery voltage (12V) to the digital circuit voltage (3.3V), with proper protection circuit.

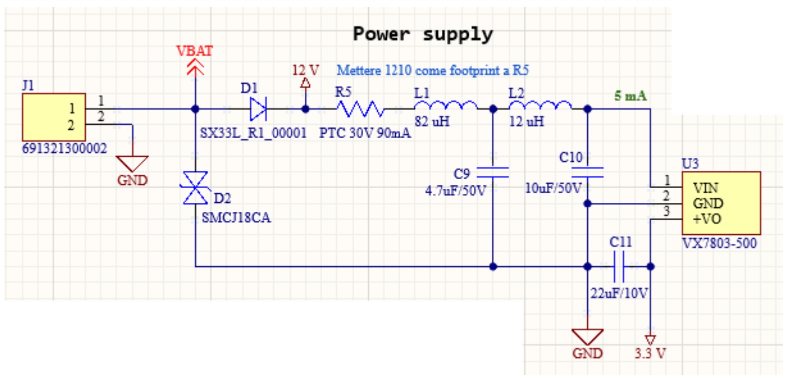


Figure 5.19: EV power schematic

The converter is the same as that used in the BridgeWatch board, so please see the previous section for details.

Raspberry power supply

This board is also responsible for generating the voltage to power the Raspberry, i.e. 5V. A DC-DC converter from 12V to 5V is therefore also used for this purpose, and the following was specifically chosen:

DigiKey Part Number: 102-6253-ND
Manufacturer: CUI Inc.
Manufacturer Product Number: PDQE15-Q24-S5-D

Figure 5.20: DC-DC converter. Source: *PDQE15-Q24-S5-D - Digikey* n.d.

It has an efficiency of 90%. We can note that the 12V input voltage is taken after it has passed the protection diode D1 against the reversal of polarity. The decoupling capacitors and the protection circuit exactly follow the datasheet, as it can be seen in Fig. 5.21. Then, the 5V voltage is connected to the dedicated pin of a USB-A connector, to which the power supply of the Raspberry will then be connected directly.

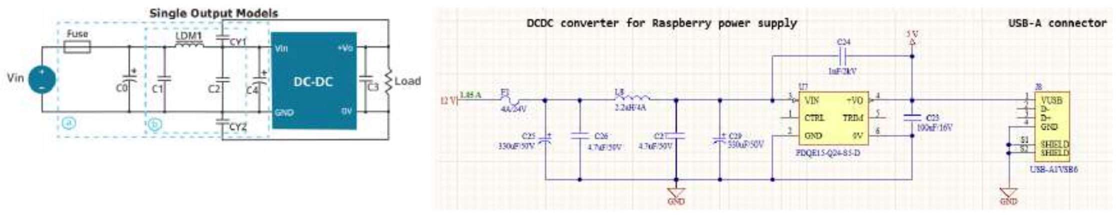


Figure 5.21: EV Raspberry power supply schematic

5.2.2 RS232 communication

Unlike the BridgeWatch board, communication between the EnvironMonitor and the Raspberry is via the RS232 protocol. The main difference is that we no longer have a bus to which several nodes are connected, but in this case it is a one-to-one communication (for more details about RS232 protocol refer to Cap. 7). As in the previous case, we now need a transceiver that converts the TTL values of the microcontroller to the values provided by the protocol. The chosen chip is the following:

DigiKey Part Number: 296-13100-1-ND
Manufacturer: Texas Instruments
Manufacturer Product Number: MAX3232IPWR

Figure 5.22: RS232 transceiver. Source: *MAX3232IPWR - Digikey* n.d.

It includes two line drivers, two line receivers, and a dual charge-pump circuit. The charge pump and four small external capacitors allow operation from a single 3-V to 5.5-V supply.

As reported in Fig. 5.23 the schematic has been made following the datasheet. Only one driver and one receiver of the two are used:

- The driver input is connected to the USART Transmitter of the microcontroller
- The driver output is connected to one pin of the dedicated connector J3
- The receiver input is connected to another pin of the dedicated connector J3
- The receiver output is connected to the USART Receiver of the microcontroller

5.2.3 Sensors' connections

A common factor for all environmental sensor inputs is the presence of a protection circuit before the signal enters the board. For every signal that arrives from the external environment, it is essential to have a protection circuit in place. External signals are often susceptible to various forms of electrical disturbances such as voltage spikes, electromagnetic

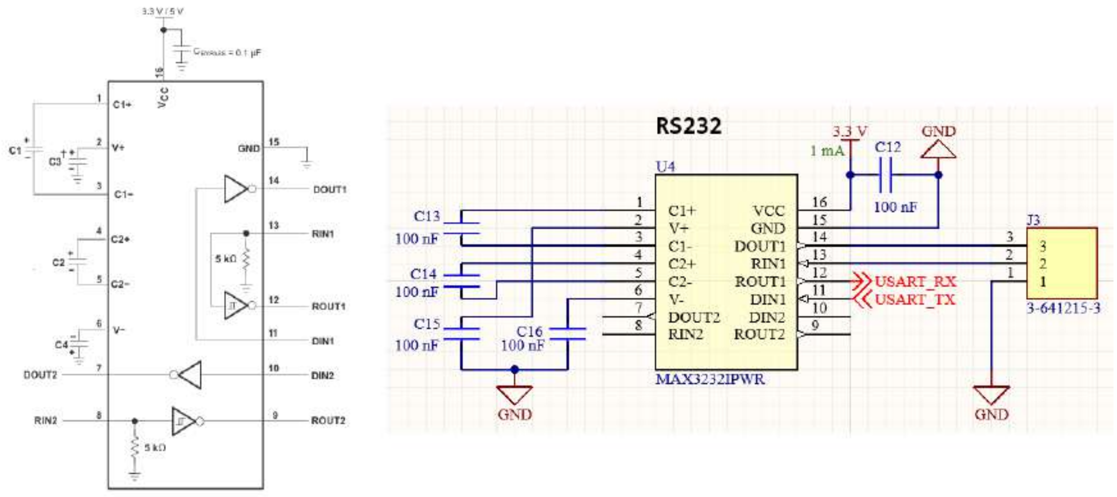


Figure 5.23: RS232 transceiver datasheet and schematic

interference (EMI), and electrostatic discharge (ESD). Without adequate protection, these disturbances can cause significant damage to sensitive electronic components and impair the overall functionality of the system. Protection circuits, which can include components such as ferrite beads, diodes, and transient voltage suppression devices, serve to filter out noise, limit voltage surges, and shield the internal circuitry from harmful external influences. Implementing these protective measures ensures the reliability, stability, and longevity of the electronic system, safeguarding it against potential damage and ensuring consistent performance even in challenging environments.

Rain gauge

The 3 wire of the rain gauge are connected in the connector J4. L3, R8 and D3 are part of the protection circuit. The output of the rain gauge is a switch that closes to ground every time the spoon tips, so a pull-up resistor is needed to maintain a clear and stable high state when the switch is open, and to allow the transition to a low state when the switch is closed.

Thermohygrometer

The 4 wire of the rain gauge are connected in the connector J6. The sensor uses the I2C interface to communicate, so simply the two signals employed by the protocol are connected to the corresponding ones on the microcontroller.

Water level sensor

The water level sensor has a current output, so it must be converted to a voltage before it can enter the ADC input and be read by the microcontroller. R14 is the resistor used to convert the current in voltage. Since the reference voltage of the ADC is equal to 2.5V, the input voltage must not be greater. For this reason the value of R14 is computed in the following way:

$$R14 \cdot I_{\max} \leq 2.5\,\mathrm{V}$$

The output current from the sensor can range from 4 to 20mA depending on the distance to the water detected. So, it becomes:

$$R14 \leq \frac{2.5}{0.02} \implies R14 \leq 125\Omega$$

A value of 120 Ω with 0.1% accuracy has been chosen.
Finally, the signal passes through the operational amplifier U2 which is in buffer configuration before it enters an Analog-to-Digital Converter (ADC). It is important for two reasons:

- Firstly, a buffer provides impedance matching, ensuring that the signal source is not loaded by the ADC's input impedance, which could otherwise distort the signal. This is particularly crucial when dealing with high-impedance sources.
- Secondly, a buffer helps to isolate the ADC from the signal source, protecting it from potential fluctuations or noise that might originate from the source. This isolation improves the overall stability and accuracy of the measurement.

Anemometer for wind speed

The 4 wire of the anemometer are connected in the connector J5. As mentioned in Chap.4, the anemometer output for wind speed is a switch that closes to ground with each turn of the vane. The schematic is therefore the same as that of the rain gauge: L6, R10, D5 are part of the protection circuit, while R11 is the pull-up resistor. Adding the capacitor C19 between the pull-up resistor and ground serves to debounce the switch and filter out noise. When the switch is toggled, it can produce multiple rapid on-off signals due to mechanical bouncing, which can cause false triggering or erratic behavior in the circuit. The capacitor helps to smooth out these transient signals by charging and discharging more slowly than the mechanical bounces, thus providing a cleaner transition from high to low states. Additionally, the capacitor acts as a low-pass filter, reducing high-frequency noise that might be coupled into the signal line.

Anemometer for wind direction

The anemometer for wind direction has an internal potentiometer which range from 0 to 20kOhm. By connecting one of the four wires of the potentiometer to the power supply and another to ground, the output voltage from the potentiometer's voltage divider will vary in proportion to the position of the wiper. If the wiper is at one end of the potentiometer, corresponding to a resistance of 0 ohms, the output voltage will be 0 volts. Conversely, if the wiper is at the other end, corresponding to the full resistance of 20 kOhms, the output voltage will be equal to the supply voltage. The output voltage then needs to go into the ADC so that it can be converted to digital and read by the microcontroller. To ensure that the voltage entering the ADC isn't greater than its reference voltage, the anemometer is supplied with the same reference voltage. L5, R9, D5 are part of the protection circuit, and the operational amplifier U6 is used as a buffer for the reasons listed above.

5.2.4 Microcontroller and JTAG

Microcontroller and JTAG are the same of BridgeWatch board, so please refer to the previous section for details. The main differences are the addition of the two pull-up resistors required by the I2C interface and the use of the ADC peripheral, which was not required before. In particular, the use of the ADC means that the supply voltage for the analogue part VDDA, which is also the reference voltage for the ADC, must be very precise and stable. For this reason, the following voltage reference is used:

DigiKey Part Number: MCP1525T-I/TTCT-ND
Manufacturer: Microchip Technology
Manufacturer Product Number: MCP1525T-I/TT

Figure 5.24: 2.5V voltage reference. Source: *MCP1525T-I/TT - Digikey* n.d.

It takes as input a voltage in the range from 2.7V and 5.5V, and outputs a voltage of 2.5V, with a tolerance of 1%. The schematic follows the datasheet, as reported in Fig.5.25.

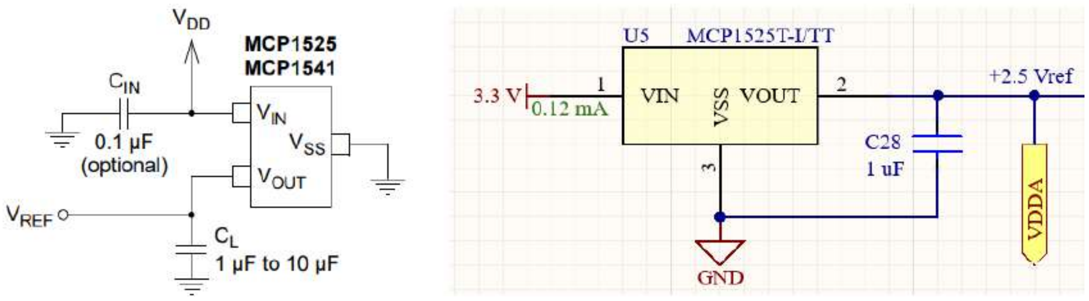


Figure 5.25: Datasheet and schematic of the voltage reference

Another important point to ensure the correct operation of the ADC is the use of an operational amplifier in buffer configuration, connected between the input signal and the input of the ADC. The buffer configuration, or voltage follower, serves two main purposes: isolating the microcontroller from potential external disturbances and preventing the input from being loaded, which could alter the signal being measured. The following chip has been used:

DigiKey Part Number: 2129-NJU77552G-TE2CT-ND
Manufacturer: Nisshinbo Micro Devices Inc.
Manufacturer Product Number: NJU77552G-TE2

Figure 5.26: Operational Amplifier. Source: *NJU77552G-TE2 - Digikey* n.d.

It is a dual rail-to-rail input/output single supply operational amplifier.

5.2.5 Batthery charge control

In order to be able to check the state of the battery at all times, it was decided to use the ADC to read this voltage as well. As the maximum battery voltage is approximately 12 V, but the ADC reference voltage is 2.5 V, it must be reduced to within this value before connecting to the peripherals. For this reason, a voltage divider was realised.

5.3 Estimated power consumption

5.3.1 BridgeWatch

To analyze the current consumption of the board, it is essential to determine the total current drawn from the battery. The first step is to identify all direct connections to the battery's 12V supply. In this case, the only direct connection is the 12V to 3.3V DC/DC converter. Therefore, the analysis should focus on this converter.

The process then involves several steps:

1. **Find the Total Output Current**: Begin by estimating the total current output of the 12V to 3.3V converter. This current represents the sum of the currents drawn by all components and devices connected to the 3.3V supply.

2. **Calculate the Output Power**: Using the total output current and the output voltage (3.3V), calculate the output power of the converter using the formula:

$$P_{\text{out}} = V_{\text{out}} \times I_{\text{out}}$$

3. **Determine the Input Power**: Next, estimate the input power required by the converter. Since some power is lost during the conversion, the input power will be higher than the output power. The input power can be calculated using the formula:

$$P_{\text{in}} = \frac{P_{\text{out}}}{\eta}$$

where η is the efficiency of the converter (typically given in the datasheet).

4. **Calculate the Input Current**: Finally, with the input power known, calculate the input current drawn from the 12V battery using:

$$I_{\text{in}} = \frac{P_{\text{in}}}{V_{\text{in}}}$$

This value represents the total current consumption from the battery due to the converter.

The main actors in determining the total output current of the converter are reported in the table 5.4. All estimates are based on the worst case scenario on the datasheet.

Component	**Estimated power consumption** $[mA]$
Microcontroller	6.72
Accelerometer	0.2
Gyroscope	6.1
LED	8
RTD to digital converter	4
Transceiver	7.3
Total	**32.32**

Table 5.4: BridgeWatch current consumption estimates

Based on the information reported on the datasheet, we can estimate the efficiency of the DC-DC converter at about 80%. With all the major current consumption contributions accounting for an average 32.32 mA at a voltage of 3.3V, the output power from the DC-DC will be

$$Pout_{DC-DC} = 32.32mA * 3.3V = 106.65mW \tag{5.1}$$

Then, the input power to the DC-DC converter will be:

$$Pin_{DC-DC} = 106.65mW/0.8V = 133.32mW \tag{5.2}$$

This, on a 12V input voltage, results in a total current consumption of the BridgeWatch board equal to:

$$I_{BW} = 133.32mW/12V = 11.11mA \tag{5.3}$$

5.3.2 EnvironMonitor

For the EnvironMonitor board, the three contributions to the total current consumption are from the 12V-3.3V DC-DC converter, the 12V-5V DC-DC converter and the water level sensor (which is powered directly by the 12V battery voltage).

For the two DC-DC converter, the reasoning to do is the same as for BridgeWatch case. First, calculate the total output current required by the converters. From this, determine the total output power. Next, using the output power and the efficiency, compute the total input power needed for the converters. Finally, derive the total input current based on the input power.

DC-DC converter 12V to 3.3V

The main actors in determining the current consumption for the converter are reported in the table 5.5. As before, all estimates are based on the worst case scenario on the datasheet.

Component	**Estimated power consumption** $[mA]$
Microcontroller	6.72
Rain gauge	0.3
Thermohygrometer	1
LED	8
Anemometer	0.3
Transceiver	1
Opamp U2	0.1
Opamp U6	0.225
Total	**37.65**

Table 5.5: Environmonitor DC-DC converter 12V to 3.3V output current consumption estimates

As for the operational amplifiers, the datasheet says that they consume 0.05mA/channel, so 0.05mA * 2 channels = 0.1 mA. The U6 opamp also has to supply current to the anemometer potentiometer. So to the initial 0.1 mA we have to add 2.5V / 20kOhm = 0.125 mA.

The output power from the DC-DC will be

$$Pout_{DC-DC} = 37.65mA * 3.3V = 124.24mW \tag{5.4}$$

And consequently the input will be:

$$Pin_{DC-DC} = 124.24mW/0.8V = 155.31mW \tag{5.5}$$

This, on a 12V input voltage, results in a total current consumption of the 12V-to-3.3V converter equal to:

$$I1_{EV} = 155.31mW/12V = 11.11mA \tag{5.6}$$

DC-DC converter 12V to 5V

Based on the information reported on the datasheet, we can estimate the efficiency of the DC-DC converter at about 88%. The output current required is only that of the Raspberry, which can be estimated as 2A. So:

$$Pout_{DC-DC} = 2A * 5V = 10W \tag{5.7}$$

Then, in ideal condition the input power should be equal to the output power of the DC-DC converter. However, with 88% efficiency, the input power to the DC-DC converter will be:

$$Pin_{DC-DC} = 10W/0.88 = 11.36W \tag{5.8}$$

This, on a 12V input voltage, results in a current consumption of

$$I2_{EV} = 11.36W/12V = 0.95A \tag{5.9}$$

Total power consumption of EnvironMonitor

The current consumption of the water-level sensor has been estimated as $I3_{bat} = 20mA$, which is the maximum current output of the sensor. So, the total power consumption of EnvironMonitor board is the sum of the three contribution:

$$I_{EV} = I1_{EV} + I2_{EV} + I3_{EV} \rightarrow I_{EV} = 5mA + 950mA + 20mA = 975mA \tag{5.10}$$

5.3.3 Total power consumption

The total current consumption is given by the sum of the current consumption of all the BridgeWatch boards and that of the EnvironMonitor board. So, for the complete system with 10 BridgeWatch boards and a single EnvironMonitor board it is:

$$I_{bat} = 10 * I_{BW} + I_{EV} \rightarrow I_{bat} = 10 * 11.11mA + 975mA = 1.09A \tag{5.11}$$

Bear in mind that this is an over-estimate.

Battery

The battery selected is a 12V, 80Ah lead car battery. It is shown in Fig.5.27. In case of bad weather or when there is no sunshine to charge the solar panels, the battery can power the system for

$$80Ah/1.09A = 73.4h \approx 73h30' \tag{5.12}$$

Figure 5.27: Battery. Source: *Battery - Amazon website* n.d.

5.4 Mechanical supports

In the context of engineering and monitoring systems, the term 'hardware' is not limited to electronic components, but also includes all the physical and mechanical components that are essential for their assembly, support and protection. Mechanical supports are essential to ensure that sensors are correctly positioned and remain stable.

The integration of electronic and mechanical components is critical. In fact, the quality and design of mounts can affect the accuracy of measurements and the reliability of the overall system. Mechanical mounts must be designed to minimise unwanted interference and vibration that could affect the data collected by the sensors. The design and choice of materials for mechanical mounts can also affect the robustness and durability of the system, particularly in harsh environments or applications that require long, maintenance-free life.

In the system developed, the main mechanical mounts are those for attaching the BridgeWatch boards to the bridge. In order to minimise the vibrations caused by the supports, we decided to mount the board on a metal plate using short, rigid spacers and glue this plate directly to the bridge using a hypoxy resin. To protect against water and moisture, an IP66 protective case is used instead. Suitable holes are drilled in the underside of the box, which is then secured to the plate covering the board with screws and silicone. The CAD representing the design just shown is in Fig.5.28.

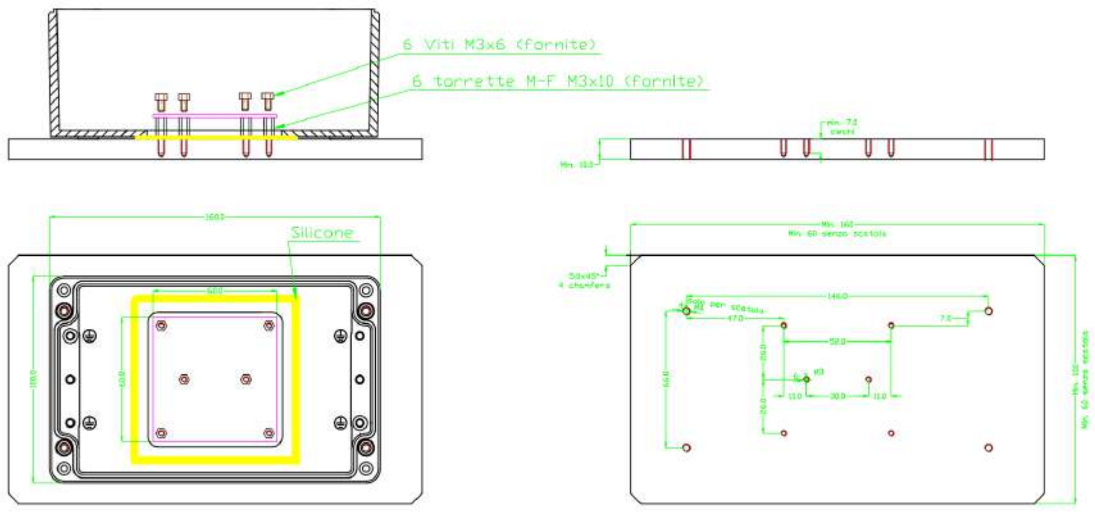


Figure 5.28: CAD of mechanical supports for BridgeWatch

The chosen case is the following. It is a die-cast aluminium 100 x 160 x 81mm enclosure, IP66 rating.

RS Part Number: 385-828
Manufacturer: Rose
Manufacturer Product Number: 01101608

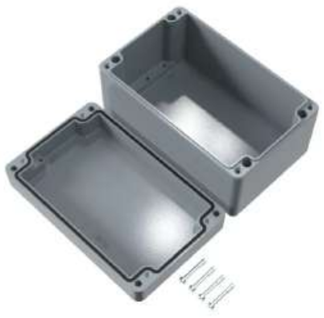


Figure 5.29: BridgeWatch protective case

Photos of the assembled system are shown in Fig.5.30 and Fig.5.31. It can also be seen that grommets have been fitted to two of the walls to allow the RS485 bus cable to connect the board to the others, while protecting the board inside from water or moisture.

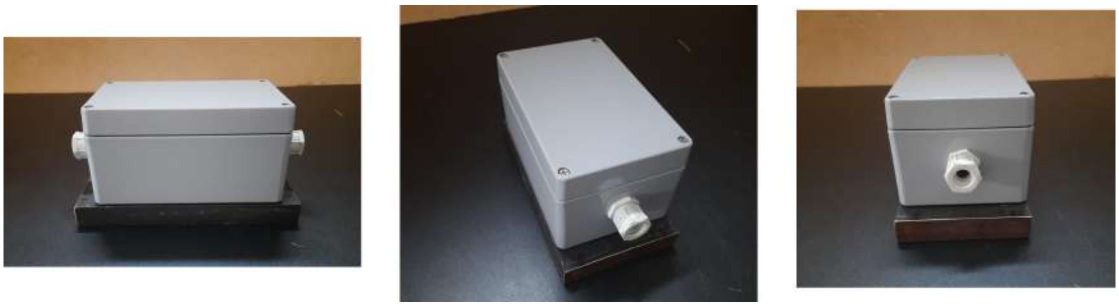


Figure 5.30: BridgeWatch case (1)

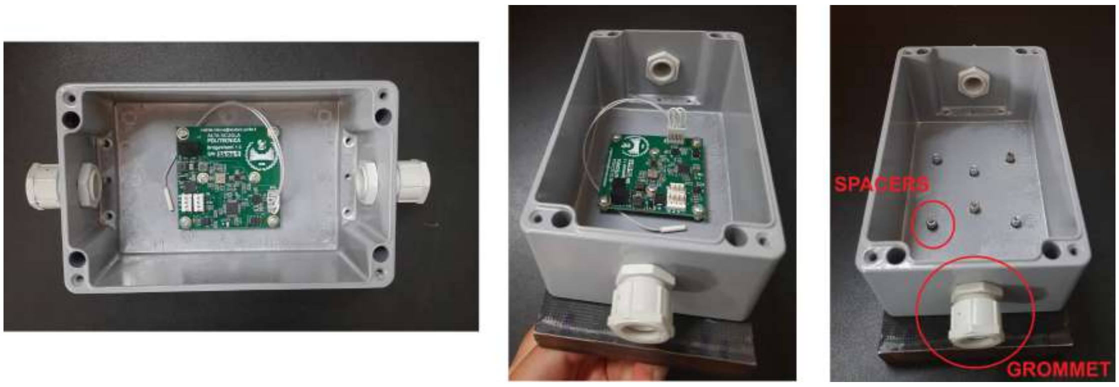


Figure 5.31: BridgeWatch case (2)

We also need a protective case for the EnvironMonitor board, battery and Raspberry Pi. This will be screwed to a pole on the bridge or somewhere else suitable. The model chosen is the following. It is 600 x 500 x 220mm.

RS Part Number: 769-6122
Manufacturer: RS PRO

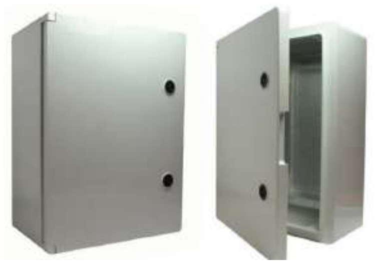


Figure 5.32: EnvironMonitor, battery, and Raspberry protective case

Another critical component of the system is the cable used to supply power and data to the BridgeWatch boards. The following cable has been selected:

Manufacturer: Berica Cavi
Manufacturer Part Number: Li2YCYv (TP)

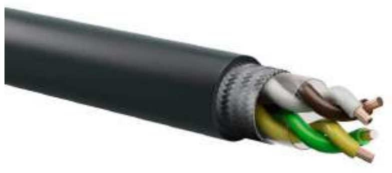


Figure 5.33: Cable for RS485 bus. Source: *Berica Cavi* n.d.

It is a cable with two twisted pairs (one for data and one for power), shielded and UV resistant.

Chapter 6

PCB development

6.1 Layouts

Once the schematic is complete, the next stage is the layout process. This involves arranging the physical placement of components on the printed circuit board (PCB) and routing the electrical connections between them. The primary objectives of the layout phase are to ensure optimum electrical performance, mechanical stability and manufacturability of the PCB. Effective layout design ensures that the circuit works as intended, and is therefore critical to the overall functionality and reliability of the final product.

It was decided to use a 2-layer PCB for both boards. It means that the boards are made up of two copper layers: one on the top and one on the bottom. In this configuration, the top layer is typically used for routing signal traces, which are the electrical paths that connect different components in the circuit. The bottom layer, on the other hand, is usually dedicated as the ground plane, a large continuous copper area that acts as a reference point for all ground connections in the circuit. It helps to reduce noise, minimise signal interference and improve overall signal integrity. This is because for every signal sent through a trace, there must be a return path to complete the electrical loop. The ground plane ensures that the return current has a low impedance path to follow and can take the shortest route back to the power source.

For simpler boards like this, adding more layers doesn't provide significant benefits. More layers are generally used in more complex designs where there are too many signals to route on just one layer, or when additional planes are needed for power distribution, signal integrity, or EMI control.

Vias are tiny conductive paths that allow signals to travel between the two layers of the board. They are small holes drilled through a PCB and plated with conductive material, typically copper. Their primary purpose is to electrically connect the top signal layer to the bottom ground plane, or to allow signals to pass between layers when necessary. For example, if a signal trace on the top layer needs to connect to the ground plane on the bottom layer, a via provides the direct path. Vias also come into play when traces need to cross each other, helping to route signals cleanly without overlap or short circuits, which is particularly important in a design limited to just two layers.

6.1.1 BridgeWatch

In Fig.6.1 there is the layout of the BridgeWatch board: the red traces are those on the signal layer, while blue traces are those on the ground plane (and so they are only draw, in reality the blue layer is not composed of individual traces but of a single continuous plane).

Component placement

To begin the layout, I first placed the drills. Next, I positioned the main components, including the accelerometer, gyroscope and microcontroller, and placed the connectors along the edges. The accelerometer and gyroscope were strategically placed in the centre, between the two central drills. This placement ensures that when the board is mounted, the central part is very rigid and mechanically coupled to the structure. As a result, the sensors are less affected by any accelerations introduced by the board itself, and it is crucial in a SHM system.

I followed this stage by placing all the other chips on the board, following a logical order according to the schematics and trying to place closely related components close together. Each component was positioned with careful consideration of ease of routing. Then I started to compact the components to get a smaller overall design. All bypass and decoupling capacitors were placed as close as possible to the pins to which they were connected, to ensure optimum filtering and noise reduction. I also revised some MCU pin assignments to better fit with the layout, to shorten connections. This was possible since many connections were on GPIOs, seamlessly exchangeable.

Bypass capacitors. Bypass capacitors are essential components used to reduce noise and regulate voltage levels in electronic circuits. They must be placed as close as possible to the power supply pins of integrated circuits (ICs), in parallel between the power supply and ground. Their primary purpose is to mitigate power supply fluctuations by acting as a local energy reserve, preventing voltage drops that can occur when the IC unexpectedly requires additional current. They also help reduce noise in the circuit by providing an alternative path for high-frequency signals to ground. In this way, bypass capacitors help to improve overall circuit performance by providing a clean, stable power source to the IC.

Routing

After the placement phase, the next step was routing. During routing, special attention was given to making the power traces slightly thicker than other traces. This is important for two main reasons. Firstly, every trace on a PCB has some inherent resistance, which can cause a voltage drop along its length. Thicker traces have less electrical resistance than thinner ones. Since power traces typically carry higher currents than signal traces, making them thicker ensures that the voltage remains stable across the board, preventing drops that could affect component performance. Secondly, thin traces carrying high current tend to heat up due to resistive losses (Joule heating). This can lead to overheating, potentially damaging the board or causing performance problems. Thicker traces help spread the current and dissipate heat more effectively, preventing hot spots and maintaining safe operating temperatures.

So, I start from the power supply components which required more attention, then shorter connection which are easier, and in conclusion the longer ones. For certain traces, it was necessary to use the ground plane to avoid crossing over other signal traces on the top layer. However, special care was taken to minimize the number of signals routed through the ground plane (Fig. 6.2). This is important to avoid obstructing the return path for currents, which helps maintain a low-impedance ground plane.

Another important point is to not route wide traces into narrow pad, to facilitate reliable soldering and ensure proper mechanical and electrical connections at the pads. Bypass capacitors have been provided with one or two vias to GND wherever there were enough space. This minimizes the inductance and resistance of the connection, thereby improving the effectiveness of the capacitors in maintaining a stable voltage supply and filtering out high-frequency noise.

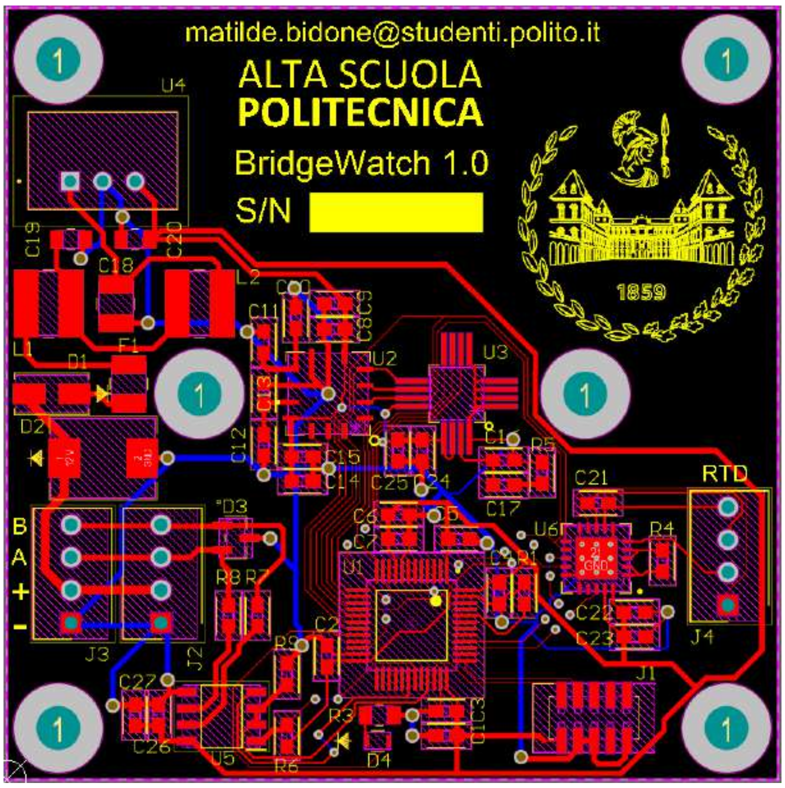


Figure 6.1: BridgeWatch layout. Dimension: 6cm x 6cm.

A nice feature of Altium Designer is its ability to display a 3D view of the PCB, allowing you to visualize the final product and make adjustments if necessary. This capability is especially useful for verifying component placement and mechanical fit within the enclosure. However, this feature is only possible if you include the .step file of each component in your library, adding the 3D model to the component footprint. In Fig.6.3 and Fig.6.4 there

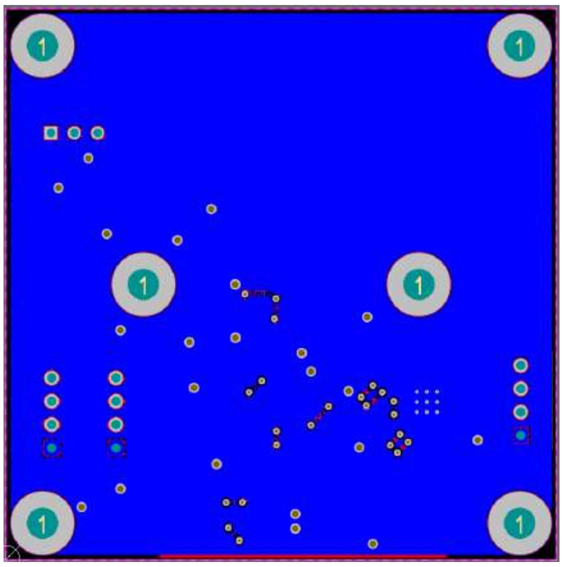


Figure 6.2: BridgeWatch ground plane

are the 3D views of BridgeWatch.

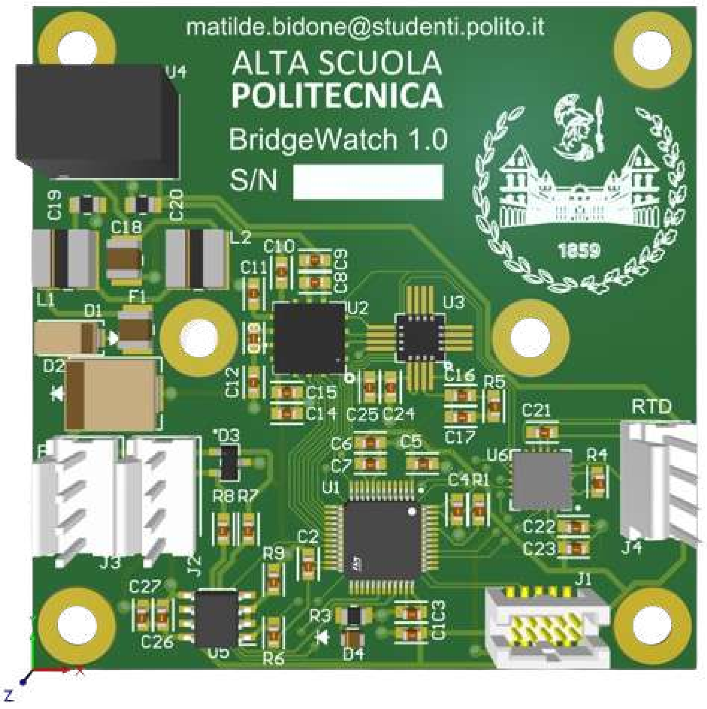


Figure 6.3: BridgeWatch 3D view

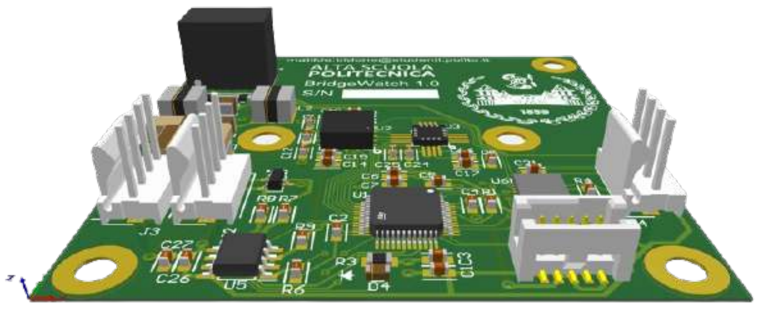


Figure 6.4: BridgeWatch 3D view (2)

Gerber files

After completing the Design Rules Check (DRC) to ensure that the layout adheres to all design specifications and constraints, the next step is to generate the Gerber files. Gerber files are a standard file format used to describe the various layers of a PCB, including the copper traces, solder masks, silkscreens, and drill data. These files provide the necessary information for PCB manufacturers to precisely fabricate the board according to the design.

Once the Gerber files are generated, they need to be uploaded to a PCB manufacturing service. These services typically have websites where you can upload your Gerber files and configure additional options for your PCB order. Some of the details you can choose include the color of the solder mask (common options are green, blue, red, black, and yellow), the color of the silkscreen, the thickness of the PCB, and the finish for the copper pads. After selecting these final details, the order can be reviewed and submitted for manufacturing.

6.1.2 EnvironMonitor

In Fig. 6.5 there is the layout of the BridgeWatch board. Again, we can see that all the environmental sensor connections have been placed on the edges for convenience. In addition, even more than in the previous case, we can see that the power supply traces are much thicker than the others, and that they are only tapered at the ends, so that they are narrower than the pads on which they arrive. In Fig. 6.6 we can see that the ground plane has no large interruptions, so the current is not impeded. Fig. 6.7 and Fig. 6.4 show two 3D views of the board.

6.2 PCB printing and component soldering

To print the PCBs, I used Multi Circuit Boards (Multi-CB), a leading European PCB manufacturer based in Germany. As mentioned at the beginning, for budget reasons I chose not to include the soldering of components by machine, but to do it myself in the

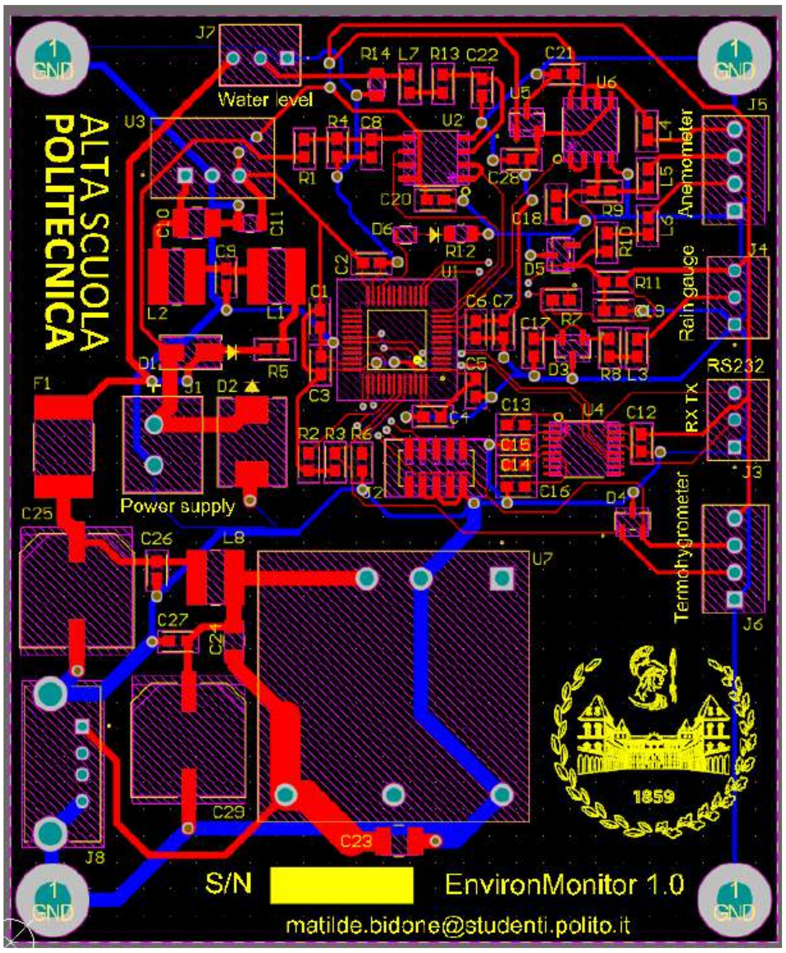


Figure 6.5: EnvironMonitor layout. Dimension: 7.2 cm x 8.7 cm.

laboratory. Fig.6.9 and Fig.6.10 show the BridgeWatch board and EnvironMonitor board respectively as they were delivered.

The next step is to solder the components. As many of the chips are very small, it was necessary to use an electron microscope to solder the pins to the correct pads without causing unwanted short circuits. All the SMD components were soldered first, and then the through holes, as they create a thickness under the board that would have made soldering the SMD components more difficult. Fig.6.11 and Fig.6.13 shows the BridgeWatch and EnvironMonitor boards with all components mounted, first from the top and then from the side. The main chips and connectors are highlighted.

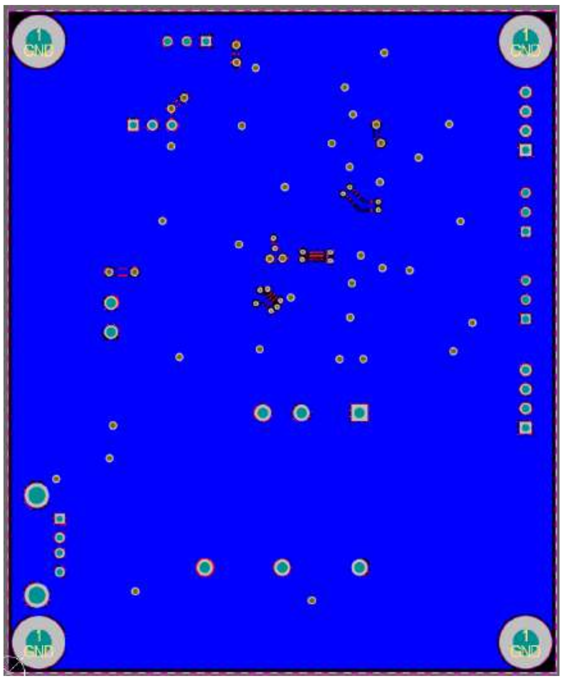


Figure 6.6: EnvironMonitor ground plane

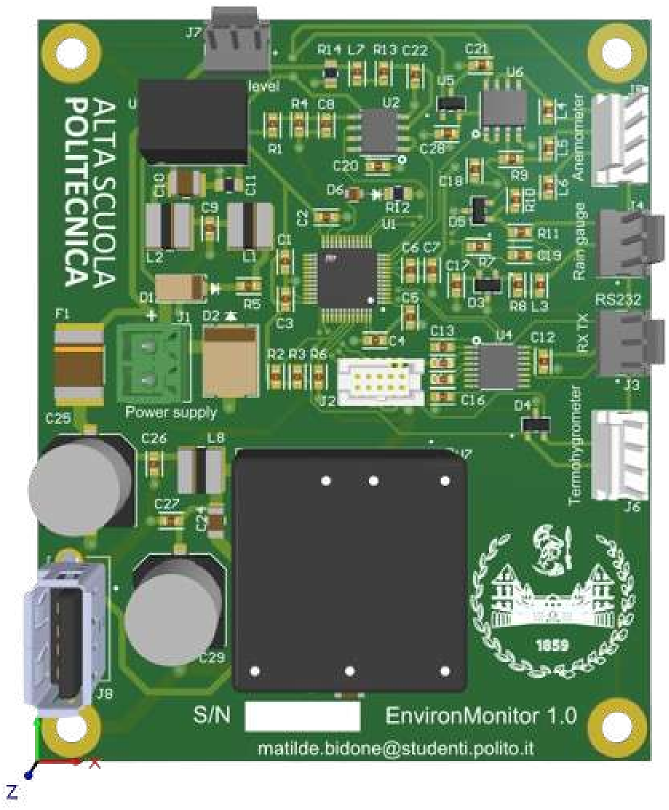


Figure 6.7: EnvironMonitor 3D view

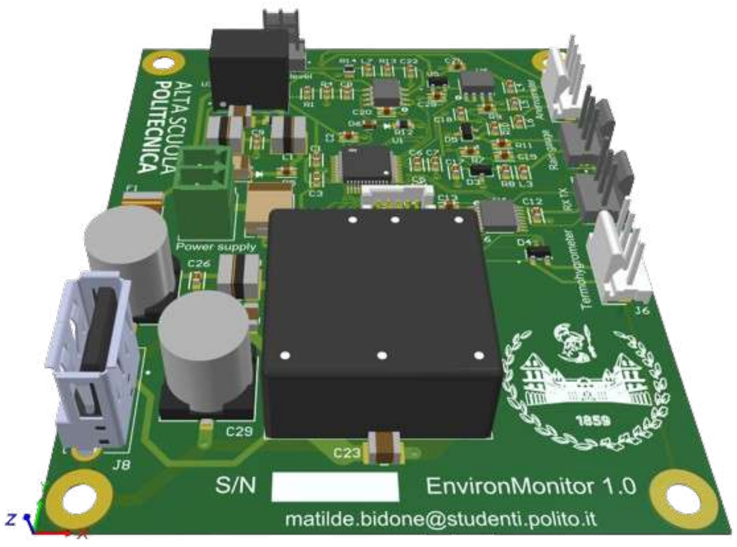


Figure 6.8: EnvironMonitor 3D view (2)

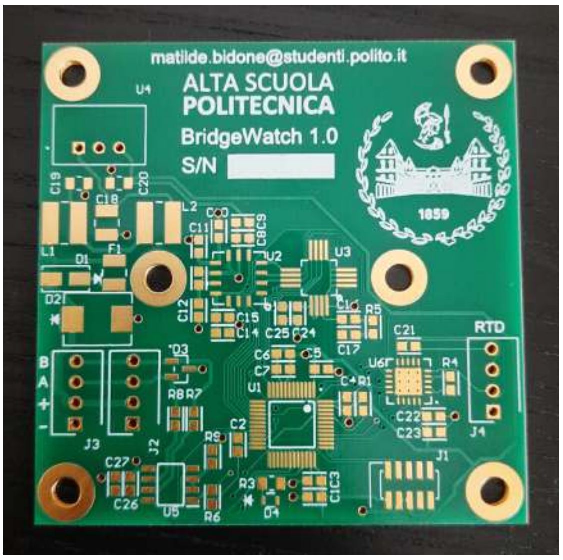


Figure 6.9: BridgeWatch board without components

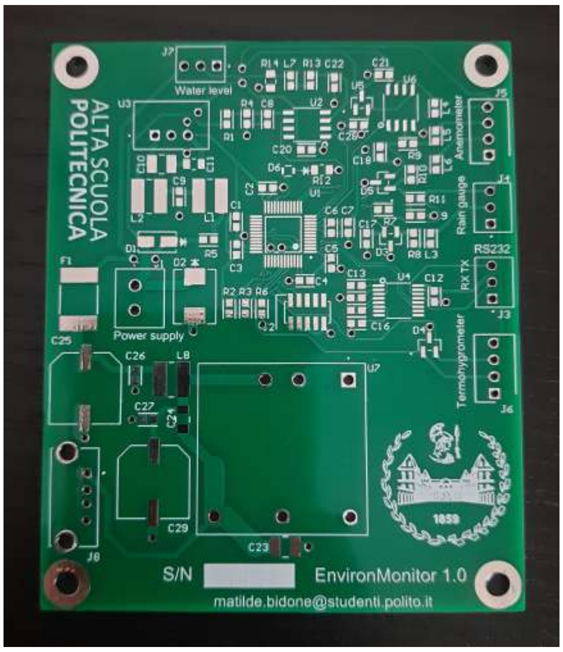


Figure 6.10: EnvironMonitor board without components

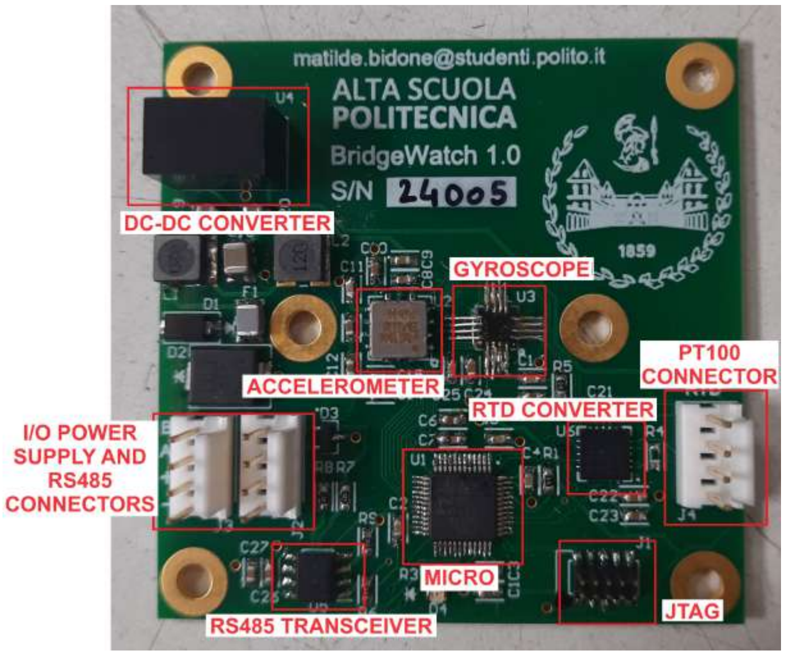


Figure 6.11: BridgeWatch board - Top view

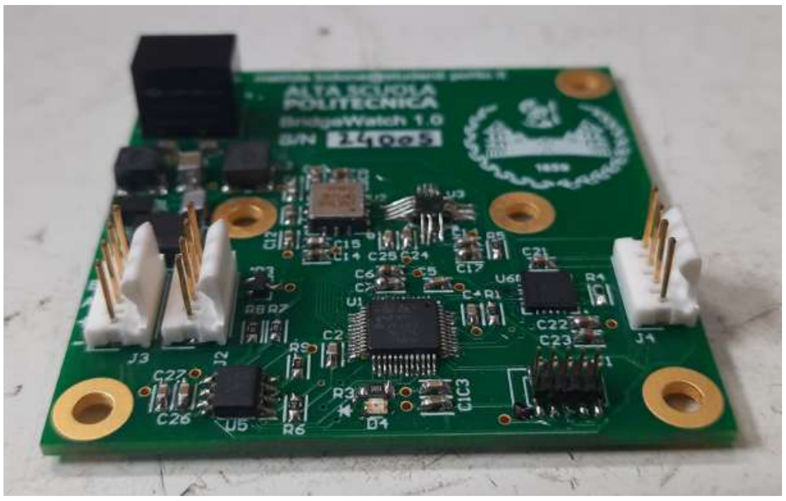


Figure 6.12: BridgeWatch board - Side view

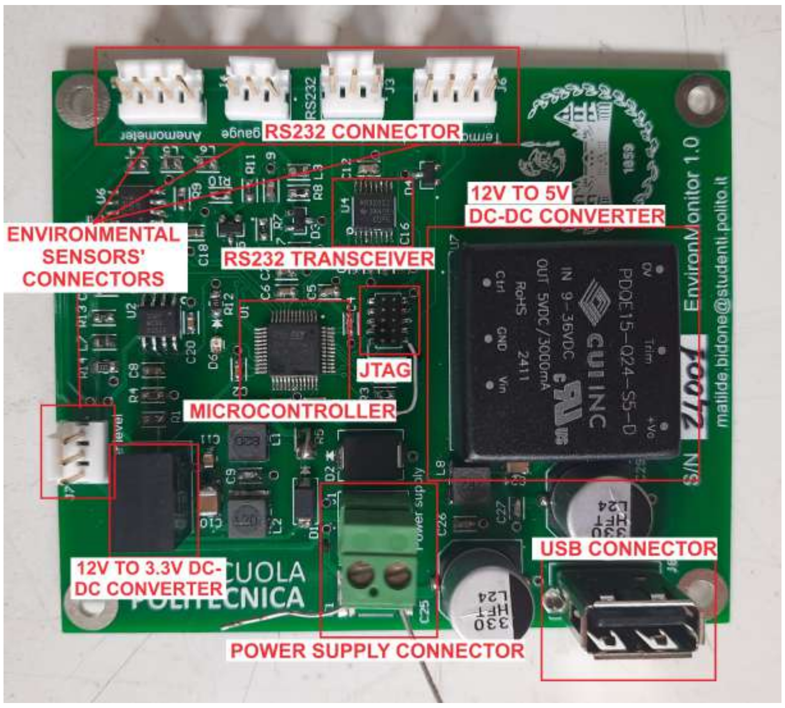


Figure 6.13: EnvironMonitor board - Top view

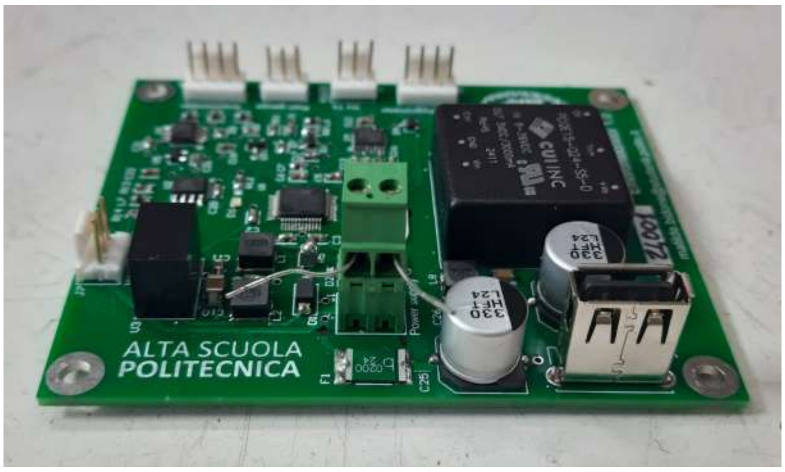


Figure 6.14: EnvironMonitor board - Side view

Chapter 7

Communication

The development of communication systems can be seen as the most important part of the project, as it allows the individual units to become a single system. We have two types of communication in the system:

- Communication within the single board, between the digital sensors and the microcontroller

- Long distance communication, between the microcontroller of each board and the centralized computer

Of course, these two types of communication use different protocols, each optimised for specific needs. Communication between a digital sensor and a microcontroller on the same PCB typically relies on short-range protocols like SPI or I2C. These protocols are designed for high-speed, low-power communication over very short distances, usually just a few centimeters, where signal integrity is easily maintained, and there is minimal risk of data corruption. However, when communicating between a microcontroller and an external computer over longer distances, different protocols, such as UART (Universal Asynchronous Receiver/Transmitter), RS232, or RS485 are necessary. These long-distance protocols are designed to handle potential issues like signal degradation, noise, and electromagnetic interference that can occur over longer cables. Additionally, they often include mechanisms for error checking and flow control, ensuring reliable data transfer despite the more challenging environment.

Fig.7.1 represents a summary of the communication protocols and interfaces employed in the system. The protocols used and the needed hardware connections are now analysed in detail.

7.1 Digital sensors - Microcontroller communication

This section explains the communication protocols used on the single board to allow the digital sensors to send data to the microcontroller.

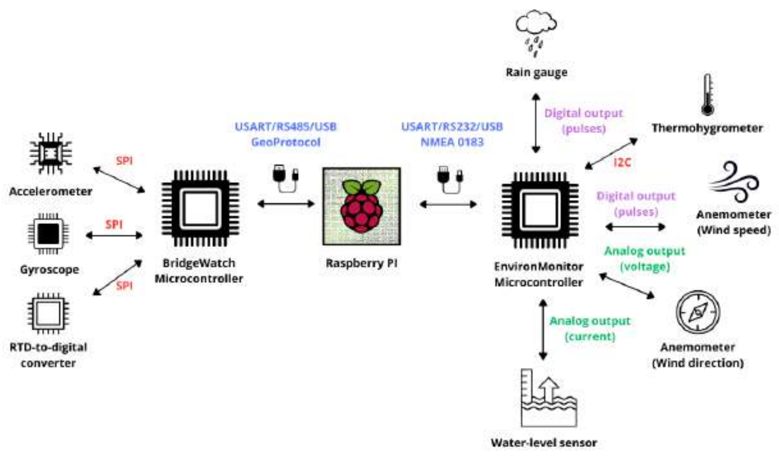


Figure 7.1: Sensors' outputs and communication protocols used in the proposed system

7.1.1 SPI

The Serial Peripheral Interface (SPI) protocol is a synchronous serial communication protocol introduced by Motorola, Inc. (Freescale Semiconductor). It is used to transfer data between a microcontroller and peripheral devices such as sensors, memory chips, and other microcontrollers. SPI operates in full-duplex mode, meaning data can be sent and received simultaneously. The protocol utilizes four main signals: MOSI (Master Out Slave In), MISO (Master In Slave Out), SCLK (Serial Clock), and SS (Slave Select):

- MOSI: Line for the master to send data to the slave.
- MISO: Line for the slave to send data to the master.
- SCLK: Dedicated line for the clock signal.
- SS/CS: Line for the master to select which slave to send data to.

Devices communicating via SPI are in a master-slave relationship. The master is the controlling device (usually a microcontroller), while the slave (usually a sensor) takes instructions from the master. In an SPI communication setup, the master device generates the clock signal (SCLK), which synchronizes data transmission between the master and slave devices. Data is transmitted from the master to the slave via the MOSI line and from the slave to the master via the MISO line. So, it is the job of the master to initiate communication and to supply the clock to the slave. The master can choose which slave it wants to talk to by setting the slave's CS/SS line to a low voltage level, and then pulling it up to disconnect the slave from the SPI bus. Each slave device typically has its own SS line, allowing the master to communicate with multiple slaves independently. For more information about this peripheral, see Appendix C.

This protocol is used on the BridgeWatch board, to allow communication between the microcontroller (which is the master), and accelerometer, gyroscope, and RTD-to-digital converter (which are the slaves). Fig. 7.2 shows the specific SPI architecture on the board.

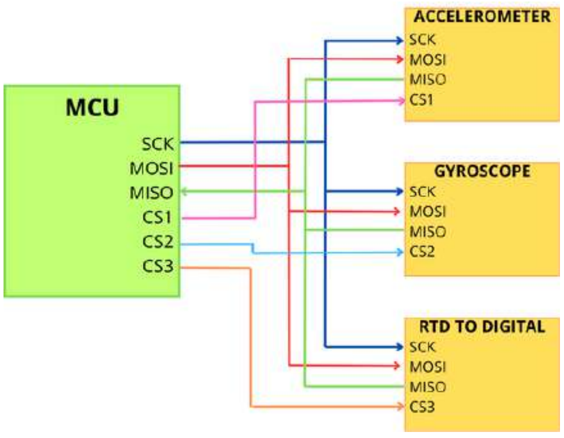


Figure 7.2: SPI architecture on BridgeWatch

7.1.2 I2C

The Inter-Integrated Circuit (I2C) protocol is a synchronous, multi-master, multi-slave, serial communication bus. It was designed by Philips in the 1980s to allow easy communication between components on the same PCB. Unlike other protocols, I2C uses only two bidirectional lines: a Serial Data Line (SDA) and a Serial Clock Line (SCL). These lines are pulled high with resistors and are shared among all devices on the bus, making I2C both space and pin-efficient. Fig. 7.3 shows the general I2C architecture.

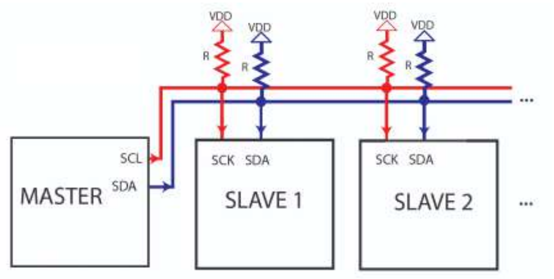


Figure 7.3: I2C architecture

I2C communication begins with the master device generating a start condition. This is indicated by the master pulling the SDA line low while SCL remains high. Following the start condition, the master sends out a 7- or 10-bit address that identifies the target slave device, along with a read/write bit to indicate the intended operation. Each bit is placed on the SDA line and synchronized with the clock pulses on the SCL line. The addressed slave device acknowledges by pulling the SDA line low during the next clock cycle, signaling that it is ready to proceed with the communication.

Data transmission in I2C occurs in 8-bit packets, known as bytes. After each byte is sent, the receiving device must acknowledge by pulling the SDA line low for one clock cycle. This acknowledgment process allows for error checking and ensures that the communication is proceeding correctly. If no acknowledgment (NACK) is received, it usually indicates a problem, such as the addressed slave not being present on the bus.

When data is written from a slave, the roles of the SDA line are reversed: the slave sends data and the master receives it. After the master has received the last byte, it sends a NACK followed by a stop condition generated by pulling SDA high while SCL is high. This stop condition indicates the end of the communication session.

In our case, the only sensor which communicates using this protocol is the thermohygrometer connected to the EnvironMonitor. In Fig. 7.4, from the schematic, we can note the typical I2C architecture. For more information about this peripheral, see Appendix D.

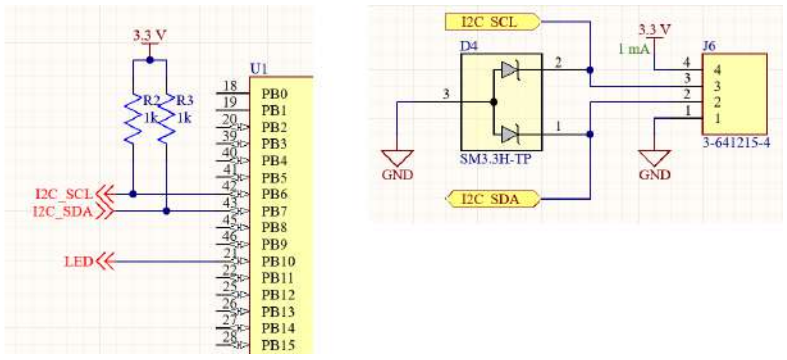


Figure 7.4: Thermohygrometer - I2C architecture

7.2 BridgeWatch - Raspberry communication

Please refer to Fig. 3.1 and Fig. 7.1 to get an idea of the general scheme of the connections and to better understand the following parts.

To enable communication between the BridgeWatch board and Raspberry Pi, three key protocols are used: UART, RS485, and a custom software protocol called GeoProtocol. Each of these serves a distinct purpose in the communication process:

- UART (Universal Asynchronous Receiver-Transmitter) is a hardware component responsible for creating and managing the data packets. It converts data between parallel format (used internally by devices) and serial format (for transmission), ensuring the data is sent and received correctly without requiring a clock signal. UART handles the framing of data, such as adding start and stop bits, to ensure that the receiver correctly interprets the incoming data. However, UART itself is limited to short-distance communication due to its voltage level and noise susceptibility.

- RS485 describes the electrical characteristics of the communication system. Specifically, it defines the voltage levels and the differential signaling required for reliable data transmission over longer distances and in electrically noisy environments.

- GeoProtocol is a custom software protocol that defines how the data packets are structured and interpreted during communication. It specifies the format of the messages, including the rules for how commands are sent and responded to.

In summary, UART handles the creation and transmission of data packets, RS485 manages the physical layer and ensures robust signal transmission, and GeoProtocol provides the structure and rules for the data exchanged between devices. Together, these components enable reliable and organized communication between the BridgeWatch board and the Raspberry Pi.

High-level vs low-level proocol

A software protocol and a UART serve distinct but complementary roles in communication systems.

The software protocol defines the overall format and structure of the data packets that are exchanged between devices. It outlines the specific sequence in which information should be arranged within a message. For example, a typical software protocol might specify that the first part of the packet contains the address of the device that should respond, followed by a command code indicating the action to be taken, and perhaps additional data or a checksum for error checking. This protocol is essential for ensuring that both the sender and receiver understand the meaning and context of the data being exchanged. It provides a blueprint for how to interpret the sequence of bytes sent over the communication line.

On the other hand, the UART (Universal Asynchronous Receiver-Transmitter) deals with the lower-level details of how each byte within the packet is transmitted and received. The UART defines the format of each individual byte, including elements like the start bit, the actual data bits, optional parity bits for error detection, and stop bits. This is crucial for ensuring that the physical transmission of each byte is done correctly, allowing the receiving device to reconstruct the data accurately.

In essence, while the software protocol organizes the higher-level structure of the data (how packets are formed and what each part of the packet means), the UART handles the lower-level details of how those packets are physically transmitted one byte at a time. Together, they ensure that data is not only sent correctly but also understood correctly by the receiving device.

7.2.1 UART

UART (Universal Asynchronous Receiver-Transmitter) is a crucial hardware component that enables devices to communicate with each other using serial data transmission. It is commonly integrated within microcontrollers and other digital devices to enable serial communication with other external devices. In practice, the UART converts parallel data from the device's bus into a serial stream of data bits, sends it through the communication medium, and then the receiving UART reassembles the bits into parallel form for the receiving device. Each UART contains a shift register, which is the basic method of converting between serial and parallel forms. UART communication is point-to-point which means that two UARTs communicate directly with each other. Despite being a hardware module, a UART also inherently incorporates a set of rules that govern how data is framed and transmitted. So, the hardware UART is the physical realization of the

UART protocol's rules, providing the necessary circuitry to send and receive serial data according to those rules.

Protocol

The UART protocol defines the rules and procedures for transmitting and receiving data serially, including aspects such as the format of data frames and timing requirements. UART protocol is one of the simplest communication protocols there is, as it requires only two wires for the devices to communicate with each other. The transmitter (TX) wire on device 1 is connected to the receiver (RX) wire of device 2. Likewise, the transmitter (TX) wire on device 2 is connected to the receiver (RX) wire of device 1. Ground (GND) wire is needed to keep both devices at the same reference voltage. UART communication can be simplex (data is only sent in one direction), half-duplex (each side talks but only one at a time) or full-duplex (both sides can transmit at the same time).

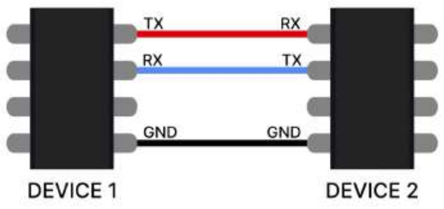


Figure 7.5: UART connection. Source: *What is the UART communication protocol* n.d.

One of the great advantages of the UART protocol is that it is asynchronous: the transmitter and receiver do not share a common clock signal. Instead, both devices agree on a common data transmission rate, known as the baud rate, before communication begins. The most common UART baud rates in use today are 4800, 9600, 19.2K, 57.6K and 115.2K. In addition to having the same baud rate, both sides of a UART connection must also use the same frame structure and parameters.

Frame format

A character is transmitted within a frame with a certain fixed format. As with most digital systems, a "high" voltage level is used to indicate a logical "1" and a "low" voltage level is used to indicate a logical "0", but the UART protocol doesn't define specific voltages or voltage ranges for these levels. In the idle state (where no data is being transmitted), the line is held high. This allows an easy detection of a damaged line or transmitter.

Because UART is asynchronous, the transmitter needs to signal that data bits are coming. This is accomplished by using the start bit. The start bit is a transition from the idle high state to a low state, and immediately followed by user data bits. After the data bits are finished, the stop bit indicates the end of user data. The stop bit is either a transition back to the high or idle state or remaining at the high state for an additional bit time.

The data bits are the user data or "useful" bits and come immediately after the start bit. There can be 5 to 9 user data bits, although 7 or 8 bits is most common. These data bits are usually transmitted with the least significant bit first.

A UART frame can also contain an optional parity bit that can be used for error detection. This bit is inserted between the end of the data bits and the stop bit. The value of the parity bit depends on the type of parity being used (even or odd):

- In even parity, this bit is set such that the total number of 1s in the frame will be even.

- In odd parity, this bit is set such that the total number of 1s in the frame will be odd.

To signal the end of the data packet, the sending UART finally drives the data transmission line from a low voltage to a high voltage for one to two bit duration.

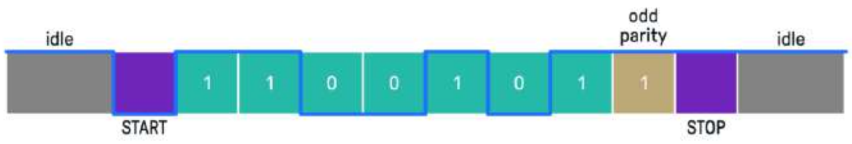


Figure 7.6: Protocol format. Source: *What is the UART communication protocol* n.d.

It is important to emphasise that the UART peripheral only deals with the serial data. It knows nothing about the actual voltages used to transmit the data. The electric signaling levels are handled by a driver circuit external to the UART. Common signal levels are RS-232, RS-485, and raw TTL for short debugging links. So, the UART usually does not directly generate or receive the external signals used between different items of equipment. Separate interface devices are used to convert the logic level signals of the UART to and from the external signaling levels, which may be standardized voltage levels, current levels, or other signals.

7.2.2 RS485

RS-485 (the RS standing for Recommended Standard) is also known as TIA-485 (Telecommunications Industry Standard) or EIA-485 (Electronics Industries Alliance). It is an electrical protocol standard (rather than a protocol), which was approved in 1983. It is a widely used communication standard that allows multiple devices to communicate over long distances using a single twisted pair of wires. Known for its robustness and noise immunity, RS 485 is suitable for industrial environments where electrical noise and interference are common. This standard supports multipoint communication, allowing several devices to connect to the same bus and communicate with each other.

RS-485 only specifies the physical layer (i.e. the electrical characteristics) of the drivers and receivers for the communication protocol. It does not specify or recommend any communications protocol, other standards define the protocols for communication over an RS-485 link.

How RS485 works

RS485 employs a differential signaling scheme: while one wire transmits the original signal, the other carries its inverse copy. This transmission method provides high resistance to common-mode interference and noise. The two signals are:

- A, which is low for logic 1 and high for logic 0.
- B, which is high for logic 1 and low for logic 0.

In addition to being differential, the two signal lines must also be balanced. Balanced signals are two lines that share a pair in a twisted pair cable with the same impedance on each line.

The fundamental operation of RS485 involves two wires, enabling half-duplex data transmission. In this mode, data can move in either direction between devices, but not simultaneously.

RS485 supports multi-drop communication, which means that multiple devices can be connected to the same bus, for a maximum of 32 nodes. RS485 uses a master-slave architecture, where one device acts as the master and controls the communication on the bus, while the other devices act as slaves. The master initiates the communication by sending a request, and the slaves respond accordingly (Fig. 7.7).

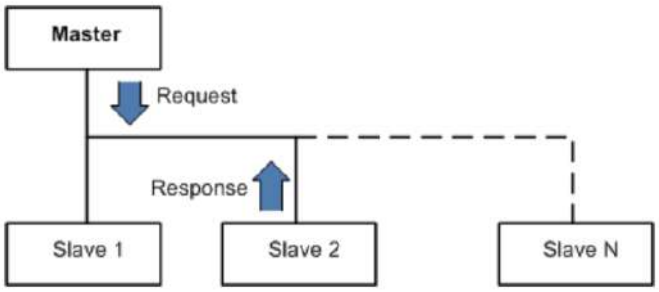


Figure 7.7: RS485 master-slave architecture. Source: *All about RS485 – How RS485 Works and How to Implement RS485 into Industrial Control Systems?* N.d.

When transmitting data, the master device sends a stream of bits over the A and B lines. On the receiving end, the slaves monitor the A and B lines and compare the voltages to determine the transmitted bits. RS485 uses differential signaling, which means it transmits data using two wires: A and B (sometimes labeled as D+ and D-). The difference in voltage between these two wires determines whether the signal represents a logical "0" or a logical "1".

- Logic 0 (Space): Occurs when the voltage on wire A is higher than on wire B.
- Logic 1 (Mark): Occurs when the voltage on wire A is lower than on wire B.

The differential voltage is the difference between the voltage on wire A and wire B, expressed as $V_A - V_B$. For RS485 to correctly detect a logic state:

- A logic 0 occurs when $V_A - V_B$ is at least +200 mV.
- A logic 1 occurs when $V_A - V_B$ is at least -200 mV.

In practical terms, the differential voltage is usually much higher than the 200 mV threshold to ensure reliability:

- Logic 0: $V_A - V_B$ is typically around +1.5V to +5V.
- Logic 1: $V_A - V_B$ is typically around -1.5V to -5V.

RS485 is designed to handle differences in ground potential between devices, which is common in long-distance communication. It allows for a common mode voltage range of -7V to +12V. This means that the voltage levels on the A and B lines, relative to the system ground, can vary within this range without affecting the differential signal interpretation.

Differential Signaling is effective for long distance communication since it assures noise immunity: since RS485 uses differential signaling, any noise that affects the wires typically affects both A and B equally. The differential receiver will subtract the voltage of wire B from wire A, effectively canceling out the noise.

RS-485 can be used with data rates up to 10 Mbit/s or, at lower speeds, distances up to 1,200 m (4,000 ft). The Fig. 7.8 illustrates the inverse relationship between data transmission speed and achievable distance.

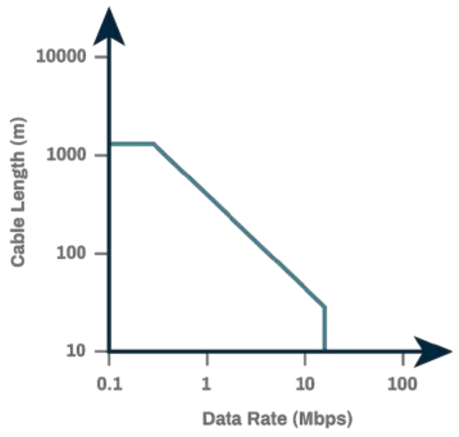


Figure 7.8: RS485 Data Rate vs Cable Lenght. Source: *Serial Communications Protocols - Part Four: RS-485 and Baud Rates* n.d.

To implement RS485 communication, three components are required:

- **RS485 transceiver:** This is a specialized integrated circuit that converts the UART signals from the microcontroller into RS485 differential signals. Fig. 7.9 shows a very standard RS485 transceiver, with one driver and one receiver in half-duplex configuration, while Fig. 7.10 shows the connection with the UART device. UART having dedicated transmit and receive lines allows it to operate as full-duplex, half-duplex, or even simplex, but since RS-485 is half-duplex, the UART connected to it will also operate at half-duplex.

- **Twisted pair cable:** RS485 requires a twisted pair cable to transmit the differential signals. The twisted pair helps to reduce electromagnetic interference (EMI) and crosstalk.
- **Termination resistors:** At both ends of the RS485 bus, termination resistors are required to minimize reflections and ensure proper signal integrity. Proper termination requires the matching of the terminating resistors to the characteristic impedance of the transmission cable. Because the RS-485 standard recommends cables with Z0 = 120 Ohm, the cable trunk is commonly terminated with 120 Ohm resistors, one at each cable end.

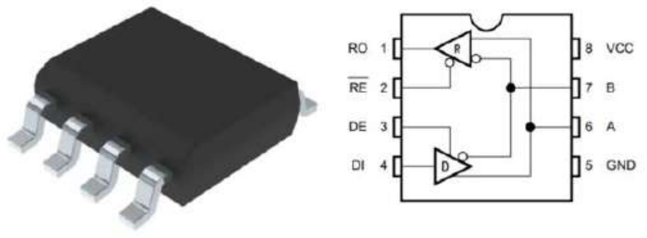


Figure 7.9: RS485 Transceiver. Source: *THVD1400DR - Digikey* n.d.

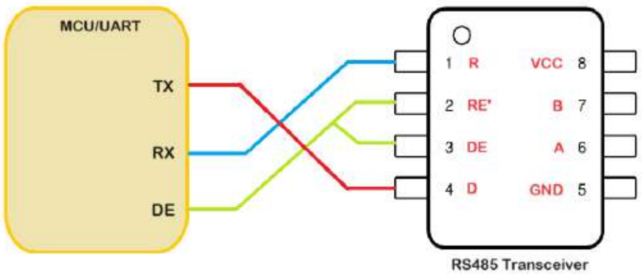


Figure 7.10: RS485 Transceiver connection

Fig. 7.11 shows a typical multi-drop RS-485 network where each device has a differential RS-485 transceiver and the link between devices is comprised of twisted pair cabling and termination resistors. In our system, we use RS485 to allow communication between BridgeWatch boards and Raspberry Pi: the former are the slaves, the latter the master (Fig. 7.12).

7.2.3 RS485 to USB converter

In a typical setup where a microcontroller communicates with a computer, the UART module in the microcontroller is used to format the data, while an RS485 transceiver converts the UART signals to the appropriate voltage levels and differential signals required for RS485 communication. This combination allows data to be transmitted over

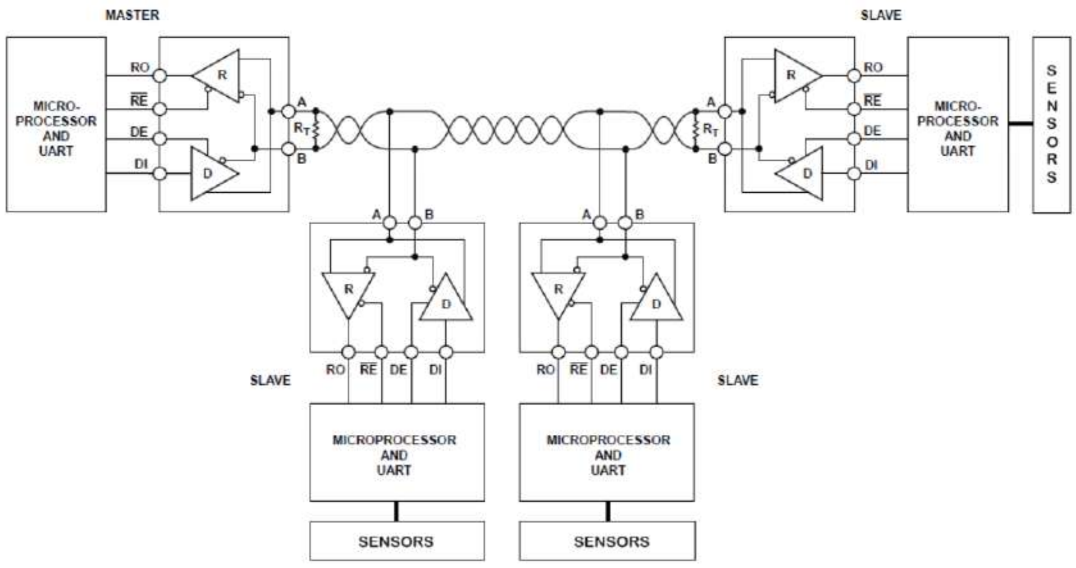


Figure 7.11: Common use of UART to RS-485. Source: Texas Instruments 2021

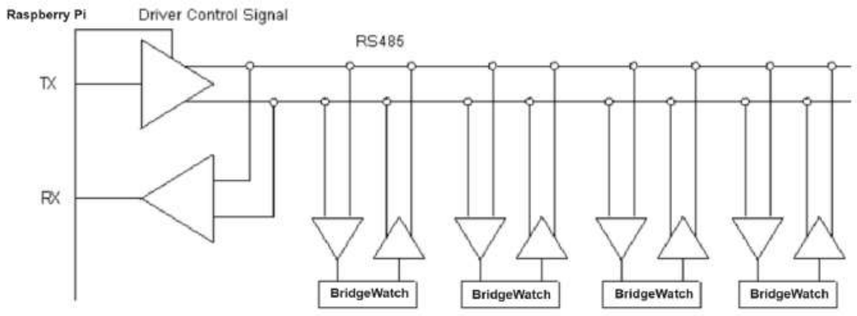


Figure 7.12: RS485 network implementation in our system

long distances from the microcontroller to the computer, ensuring robust and reliable communication even in noisy environments. The computer, typically equipped with an RS485 interface or connected to an RS485 to USB converter, can then receive and process the data.

In our system, a RS485 to USB converter is employed. The chosen device is the following:

RS Part Number: 687-7834
Manufacturer: FTDI Chip
Manufacturer Product Number: USB-RS485-WE-1800-BT

Figure 7.13: RS485 to USB converter. Source: *687-7834 - RS* n.d.

Fig. 7.14 shows the cable connections.

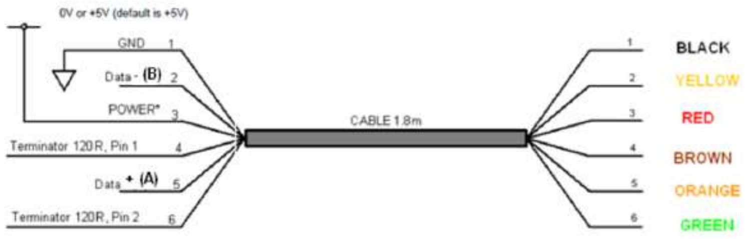


Figure 7.14: RS485 to USB converter cable connections

7.2.4 GeoProtocol

In a structural health monitoring system designed for deployment on a bridge, the use of a robust and meticulously defined software communication protocol is absolutely crucial. This protocol is not merely a technical requirement but a fundamental component that ensures the system's reliability and effectiveness over time. Given that such monitoring systems are typically installed in remote or challenging environments and are not subject to regular maintenance, it becomes imperative that the communication between sensors and monitoring equipment is flawless and resilient to errors.

A central aspect of a robust protocol is its ability to handle and correct for errors that may occur during data transmission. Structural health monitoring systems rely on continuous and accurate data collection to assess the condition of the bridge. If a bit is lost or corrupted during transmission, the consequences could be severe, leading to misinterpretations of the bridge's structural integrity. Bit loss in serial communication can occur due to various factors such as electromagnetic interference or synchronisation errors. A well-designed protocol will include error-detection mechanisms, such as checksums or cyclic redundancy checks (CRC), and will be capable of identifying when data packets have not been correctly received. It should then request a retransmission of these packets to ensure that the data received is accurate and complete. This process is essential because, in a system that is left unattended, there is no opportunity for manual intervention or repair when errors occur.

In addition, the protocol must be standardised and shared by all devices connected to the communication bus. This standardisation is essential to ensure that all components of the monitoring system interpret data in a consistent manner. The protocol defines how data packets are formatted, transmitted and interpreted, so it is essential to ensure that all devices can communicate effectively.

GeoProtocol is an ad-hoc software protocol written specifically for the purposes just listed.

Specifications

Each transmission consists of sending a so-called "data packet", which has the following structure:

- 2 fixed head-bytes (55,AA) to indicate the start of transmission
- 1 byte with the identifier of the current packet (a counter which is incremented after each packet is sent)
- The identifier of the sender (2 bytes)
- The identifier of the receiver (or broadcast if the message has to reach all boards) (2 bytes)
- An integer, used to specify the command sent (1 byte)
- 2 bytes used to indicate the length of the data to be sent
- A set of data structures (the actual data that is the subject of the transmission) for a maximum of 500 bytes, which are the parameters of the command
- 2 bytes of CRC, for data integrity

On the other hand, each time a data packet is sent and the receiver has been able to correctly interpret all the data sent, it must respond by sending an acknowledgement message so that the sender knows that the transmission was successful.

7.3 EnvironMonitor - Raspberry communication

To enable communication between the EnvironMonitor board and Raspberry Pi, three key protocols are used: UART, NMEA 0183 and RS232. The UART protocol has already been extensively described in the previous section. Let us take a quick look at the other two, and then analyse them in detail.

- As the RS485, the RS232 describes the electrical characteristics of the communication system. It employs voltage levels to represent binary data, with a positive voltage indicating a logical 0 and a negative voltage indicating a logical 1.
- NMEA 0183 is a standard software protocol originally used in maritime communication for transmitting navigational data between marine electronics devices. It defines a serial communication format that allows devices like GPS receivers, radar systems,

and sonar equipment to share information. I chose to use this protocol to communicate environmental data from the board to the Raspberry because, as it was also used by the GPS modules, it simplified the writing of the interpretation code.

7.3.1 NMEA 0183

NMEA 0183 is a widely used standard protocol designed for serial communication between marine electronic devices, such as GPS receivers, sonar systems, and radar units. Developed by the National Marine Electronics Association (NMEA), this protocol enables different devices on a marine network to exchange navigational and sensor data efficiently. Data is encoded in ASCII format, making it human-readable. The protocol uses a structured format where each message, known as a sentence, contains a series of characters that encode specific information.

- Each NMEA 0183 sentence begins with a dollar sign ('$') followed by a five-character sentence identifier.
- The first two characters of the identifier identify the type of device or the source of the data. For example: GP indicates a GPS receiver, GL refers to GLONASS data, GN is used for combined GPS and GLONASS systems (GNSS).
- The other three characters specify the type of data contained in the sentence. For example: GGA for Global Positioning System Fix Data, RMC for Recommended Minimum Specific GNSS Data, VTG for Course over ground and ground speed.
- After the identifier, the sentence includes various comma-separated fields, each representing different pieces of information. If a data field is empty, the commas remain, with no data between them, to maintain the structure.
- Following the data fields, an asterisk (*) marks the beginning of the checksum. The checksum is a two-character hexadecimal value that represents the XOR of all the characters between the $ and *. It is used to verify the integrity of the sentence during transmission.
- The sentence typically ends with a carriage return and line feed (<CR><LF>), which are standard characters to signify the end of a line or sentence.

As mentioned above, this protocol was chosen for data transmission by Enviromonitor because it is the same protocol used by GPS modules. In particular, a GPS module is connected to Raspberry so that it can continuously receive highly accurate timestamps, ensuring that the system's date and time are always synchronized with official global time standards (UTC). The two GPS sentences utilized in the system are:

```
$GPGGA,123519,4807.038,N,01131.000,E,1,08,0.9,545.4,M,46.9,M,,*47
$GPRMC,123519,A,4807.038,N,01131.000,E,022.4,084.4,230394,003.1,W*6A
```

The EnvironMonitor board sends two different sentences to Raspberry:

- One with the EVAMB identifier, with all the environmental data. For example:

  ```
  $EVAMB,24,3.2,270,11,5.28,10*51
  ```

  In order, the fields represent: temperature (°C), humidity (%), wind direction (°), wind speed (m/s), distance bridge-water (m).

- The other with the EVBAT identifier, with the battery charge in percentage. For example:

  ```
  $EVBAT,98*69
  ```

7.3.2 RS232

RS-232 is a long-established standard for serial communication, developed in the 1960s for connecting computers and peripheral devices like modems, printers, and other data terminal equipment (DTE) or data communication equipment (DCE). Despite its age, RS-232 remains relevant due to its simplicity and widespread use in legacy systems. The standard defines the electrical characteristics, timing, and the physical connectors used.

RS-232 operates using single-ended signaling, where the voltage on a single wire relative to a common ground determines the logical state. The protocol defines two primary signal levels:

- Logic 1 (Mark): This is represented by a voltage between -3V to -15V.
- Logic 0 (Space): Represented by a voltage between +3V to +15V.

Voltages between -3V and +3V are undefined and considered a "dead zone," which helps to avoid noise-induced errors.

The standard originally specified a 25-pin connector (DB25), but the more compact 9-pin (DB9) version represented in Fig. 7.15 became the most commonly used.

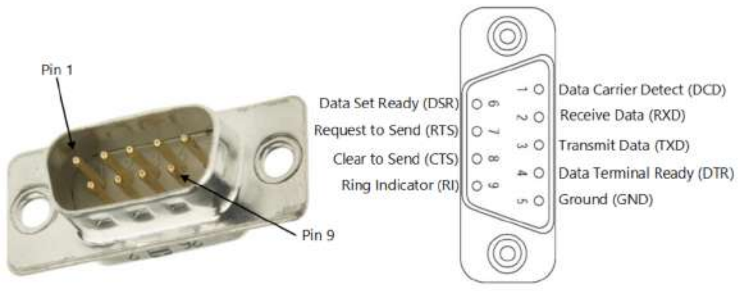


Figure 7.15: RS232 pinout. Source: *RS232 Information and Pinout* n.d.

It is important to note that not all pins are utilized in every implementation. The essential pins for communication include Transmit Data (TXD), Receive Data (RXD), and Signal Ground (GND). The remaining pins are typically employed for handshaking and flow control, but their usage depends on the specific setup. The majority of RS232 communications, and as it will be in our case, can be achieved by linking the three primary pins in Fig. 7.16. The dedicated transmit and receive lines allow the full-duplex communication, meaning data can be sent and received simultaneously.

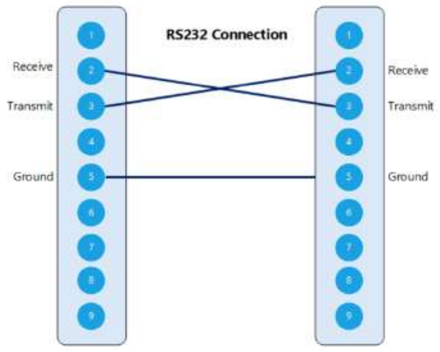


Figure 7.16: RS232 connections. Source: *RS232 Information and Pinout* n.d.

Just like RS485, UART (Universal Asynchronous Receiver-Transmitter) is used by devices for serial communication at the device level. However, because UART operates at TTL (Transistor-Transistor Logic) voltage levels, which are typically 0 to 5V or 0 to 3.3V, it is not directly compatible with the RS232 standard, which uses higher voltage levels for communication. To bridge this gap, a transceiver is necessary to convert the TTL signals from the UART into the appropriate voltage levels defined by RS232. This conversion is crucial for ensuring reliable communication between devices, especially over longer distances, where TTL signals would degrade and become unreliable.

7.3.3 RS232 to USB converter

Also in this case a RS232 to USB converter is employed to make the connection with the computer. The chosen device is the following:

RS Part Number: 687-7828
Manufacturer: FTDI Chip
Manufacturer Product Number: USB-RS232-WE-1800-BT_0.0

Figure 7.17: RS232 to USB converter. Source: *687-7828 - RS* n.d.

Chapter 8

Firmware

For this monitoring system, we need two different programmes:

- One running on the microcontroller of each board, to enable communication with the sensors on the board.
- One running on the Raspberry, to sinchronize and collect data from all the boards.

8.1 Raspberry firmware

The main program is obviously that running on the Raspberry. Its role is to manage communication between all the boards in the system, sending commands about when and what data to send, and acting as a centralised repository for all system data.

When it is launched, firstly it initializes the communication system and all the sensors on the BridgeWatch boards, and then it prints a command menu on the console asking the user to insert a choice. After executing the requested command, the menu is reprinted and the program continues in this way in a loop, as it can be seen in the pseudo-code in Fig.8.1.

The menu consists of six possible choices:

1. Set serial number of BridgeWatch
2. Print data from EnvironMonitor board
3. Start acquisition of BridgeWatch boards
4. Get info from BridgeWatch boards
5. Get info from GPS
6. Exit

A screenshot of the initial screen of the program is shown in Fig.8.2.

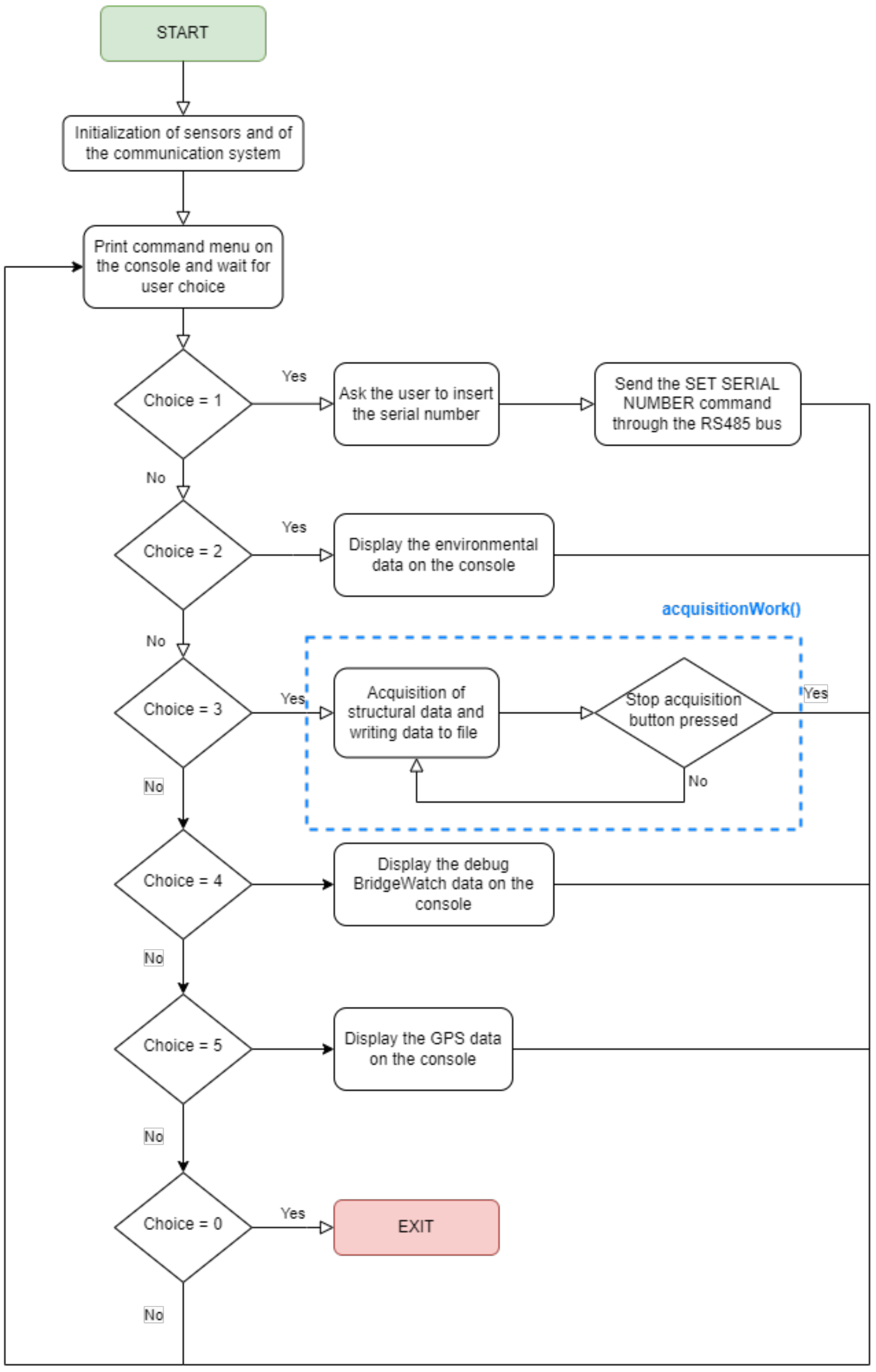
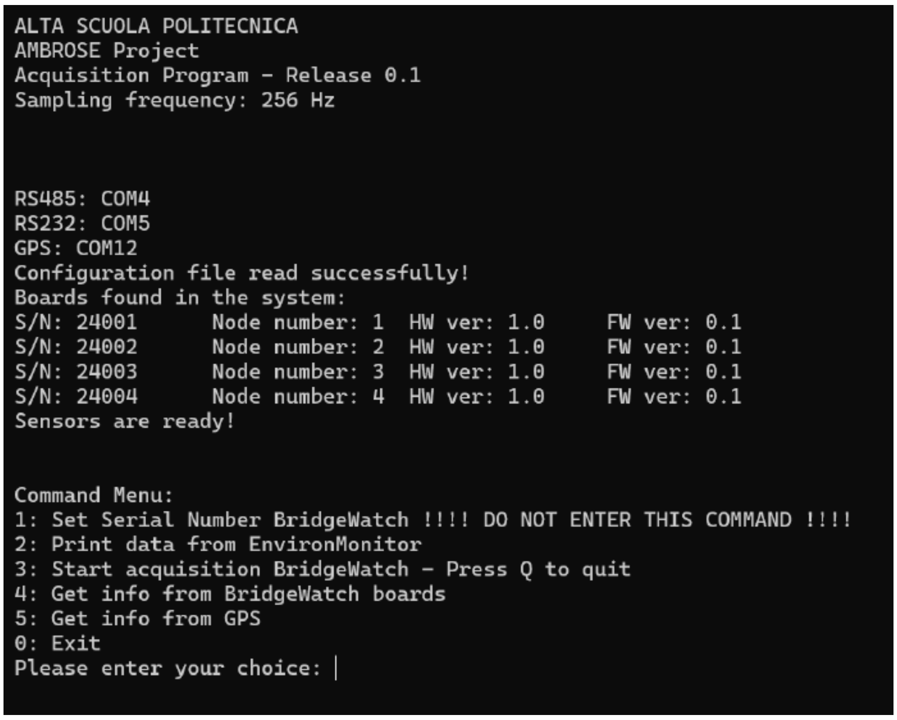

START
Initialization of sensors and of the communication system
Print command menu on the console and wait for user choice
Choice = 1
Yes
Ask the user to insert the serial number
Send the SET SERIAL NUMBER command through the RS485 bus
No
Choice = 2
Yes
Display the environmental data on the console
acquisitionWork()
No
Choice = 3
Yes
Acquisition of structural data and writing data to file
Stop acquisition button pressed
Yes
No
No
Choice = 4
Display the debug BridgeWatch data on the console
No
Choice = 5
Display the GPS data on the console
No
Choice = 0
Yes
EXIT
No

Figure 8.1: Pseudo-code



```
ALTA SCUOLA POLITECNICA
AMBROSE Project
Acquisition Program - Release 0.1
Sampling frequency: 256 Hz



RS485: COM4
RS232: COM5
GPS: COM12
Configuration file read successfully!
Boards found in the system:
S/N: 24001      Node number: 1  HW ver: 1.0     FW ver: 0.1
S/N: 24002      Node number: 2  HW ver: 1.0     FW ver: 0.1
S/N: 24003      Node number: 3  HW ver: 1.0     FW ver: 0.1
S/N: 24004      Node number: 4  HW ver: 1.0     FW ver: 0.1
Sensors are ready!


Command Menu:
1: Set Serial Number BridgeWatch !!!! DO NOT ENTER THIS COMMAND !!!!
2: Print data from EnvironMonitor
3: Start acquisition BridgeWatch - Press Q to quit
4: Get info from BridgeWatch boards
5: Get info from GPS
0: Exit
Please enter your choice: |
```

Figure 8.2: Screenshot of the initial screen of the program

8.1.1 Commands analysis

Each action is now explained in detail, from the point of view of what happens "on the Raspberry side". The "microcontroller side" is analyzed further on.

Initialization of the communication system

As said in Chap.7, each transmission using GeoProtocol consists of sending a so-called "data packet", with the identifier of the sender, the identifier of the receiver (or broadcast if the message has to reach all boards), an integer used to specify the command sent, and a set of data structures.

In order to make the programme more readable, all commands have been defined using an enumerative type, in the following way:

```
typedef enum {
    E_NOP,
    E_INIT,
    E_START,
    E_STOP,
    E_SET_SERIAL_NUMBER,
    E_SET_ADDRESS,
    E_PUT_CONFIG,
    E_SEND_DATA,
    E_REQUEST_DATA,
    E_GET_INFO,
    E_PUT_INFO,
} ED_COMMANDS;
```

A basic concept is that the bus used is common to all boards, and so any message is potentially always readable by all. Sending the message with the sender and receiver identifiers is crucial to ensure that each board understands when a message is addressed to it and when it should ignore it. The identifier used is an integer (called the node number) ranging from 1 to the total number of boards present.

The protocol used provides high-level functions that can be used to facilitate data transmission and reception. The two most used functions are:

```
    bool TxMessage(WORD wSender, WORD wDest, int eCode, const
    void *pTxData, int nDataLen);
```

used for sending data, where "eCode" is the integer of the command, "*pTxData" is a pointer to the data to send, and "nDataLen" is the lenght in bytes; and

```
    bool RxWaitMessage(WORD wSender, int eCode, int nTimeoutClk);
```

used for receiving data.

Each time the program starts, the Raspberry first reads a configuration file in which each serial number is associated with the corresponding node number on the bus, so that it knows how many boards are present. Then, since each board does not yet know its node number, the Raspberry broadcasts a message for each line read from the configuration file, in which it transmits the serial number and associated node number. To do so, the program fills a dedicated structure with the serial number and node number read on each line, and then sends it using the "E_SET_ADDRESS" command.

```
typedef struct {
 DWORD dwSerialNumber;
 WORD wNodeNumber;
} SD_Cmd_set_address;

SD_Cmd_set_address SSetAddress;
SSetAddress.dwSerialNumber = SNode.nSerialNumber;
SSetAddress.wNodeNumber = SNode.nNodeNumber;

g_CGP.TxMessage(AD_CONTROLLER, AD_BROADCAST,
E_SET_ADDRESS, &SSetAddress, sizeof(SSetAddress));
```

Each board checks if the serial number matches its own and if so, saves its node number in memory. As mentioned before, from now on the Raspberry will send messages either broadcast or using the node number as identifier to address a specific board. Then, if communication is successful, the board responds by sending a structure containing, among other things, hardware and firmware versions with the command "E_PUT_CONFIG", and these are printed on the console. If, however, the Raspberry fails to establish communication with one of the boards found in the configuration file, an error message is displayed.

Here is an example of the configuration file used:

```
# Configuration file. Usage <serial number> <node number>.
24001 1
24002 2
24003 3
24004 4
```

It was decided to use node numbers as addresses instead of serial numbers with a view to creating a more high-level program, reusable without modification even if one day the boards on the bus change, as all that is needed is to update the configuration file.

The last thing Raspberry does in this stage is to initialize all the sensors on BridgeWatch board, which is done by broadcasting a specific command on the bus.

```
g_CGP.TxMessage(AD_CONTROLLER, AD_BROADCAST, E_INIT,
&samplingFreq, sizeof(samplingFreq));
```

The only parameter passed with the command is the acquisition frequency chosen, which is used by the microcontroller to correctly set timers and sensors. The actions taken in response by the microcontroller will be analysed in the next section on firmware.

Set serial number of BridgeWatch

However, there is a problem with what was just said, which is that at the "very start of everything", each board does not even know its own serial number, so how can it compare with the one sent by the Raspberry during the "set address phase" to see if the node number refers to it? The only way to solve this problem is to connect only one board at a time to the bus, and have the Raspberry send a message in broadcast mode with the serial number of the connected board. This way, only the board in question will read it and be able to store it in flash memory, so that it is permanent even after a system shutdown.

This operation must be done once and only once at the beginning. Otherwise, once all the boards are connected to the bus, all of them would store the same serial number in the flash, and the system would be irreparably compromised.

So, when this command is selected, the program asks the user to enter the serial number and hardware version of the connected board, and then performs the broadcast transmission.

```
typedef struct {
 DWORD dwSerialNumber;
 DWORD dwHardwareVersion;
} SD_Cmd_set_sn;

SD_Cmd_set_sn SSetSn = { serialNumber, fwVersion };
```

```
g_CGP.TxMessage(AD_CONTROLLER, AD_BROADCAST,
E_SET_SERIAL_NUMBER, &SSetSn, sizeof(SSetSn))
```

Get info from EnvironMonitor board

For what concern the EnvironMonitor board, the communication system is completely different from that of BridgeWatch:

- Since we want to reserve the RS485 bus only for the structural data and this time it is a one-to-one communication, a RS232 bus is used
- The board does not wait for a request of data from Raspberry, but every second it send environmental data to Raspberry and then it is up to him to decide when to read what comes and when to ignore it. So, it is a unilateral communication since Raspberry only receives data.
- A different software protocol is also used. In this case it has been decided to use the NMEA 0183 protocol. This because it is the same protocol used by the GPS sensors and this reduces the amount of software needed.

The whole part of the program involved in communicating with EnvironMonitor board is carried out in a parallel thread to that of the main programme. By separating the handling of serial communication into a dedicated thread, the main program thread remains responsive and able to perform other operations without being stuck waiting for data from the board. Furthermore, distributing the workload over several threads improves the overall performance of the program and allows system resources to be used more efficiently.

So, the program is written such that EnvironMonitor continuously sends data, and the parallel thread is constantly waiting for a new character to arrive on the UART. When it recognizes that an entire sentence is arrived, it decodes it by parsing the sentence to extract all the fields of interest, and finally fills a data structure. The data structure is therefore a variable that each second is updated with the data just received. The only thing the Raspberry has to do when the command is selected, is to read the data structure and print data on the console.

Start acquisition of BridgeWatch

This command is the core of the whole program. When it is chosen, the program starts to acquire data from sensors and stops only when the user presses a quit button. The acquisition period is structured in the following way:

- For each minute of acquisition, the program produces a text file with all the data acquired by the sensors.
- For each minute of acquisition, 59 seconds are actually used to acquire, while the last second is used to resynchronise all the boards and write the data to the file.

This last point is crucial because in order to have data from the different nodes that can be correlated, they must all have been acquired at the same time. Since the boards are designed to sample on the rising edge of a PWM generated by each microcontroller, the resynchronisation part consists of stopping and restarting simultaneously the timer on each board. This is done when Raspberry sends the "E_START" command.

```
g_CGP.TxMessage(AD_CONTROLLER, AD_BROADCAST, E_START,
&samplesPerFile, sizeof(samplesPerFile));
```

This command requires as parameter the number of samples to acquire during the acquisition period, so that when the board finishes collecting all the required samples, it can stop itself.

To implement the acquisition part, the program needs to have a set of information stored for each node. For this reason, a data structure ad hoc was created, that contains:

- Serial number
- Node number
- The structure already used in the initialisation phase with firmware version, hardware version, etc.
- The number of samples sent by the node
- An array containing the frames sent by the node, where each frame is composed by the external temperature, the accelerometer temperature, the three accelerations and the three rotation velocities

A vector of these structures (one for each node) is instantiated as follows, and is filled at the beginning as the prorgamme reads the configuration file.

```
typedef struct {

    int nSerialNumber;
    int nNodeNumber;
    SD_Config SNodeConfig;
    int nSamples;
    array<array<float, E_CHANNELS_NUM>, FRAMES_PER_FILE> vfFrames;

} SD_System_nodes;

vector<SD_System_nodes> g_vSSystemConf;
```

The program then continues to iterate through the nodes in the system using a for loop until the acquisition period ends (59 seconds) or the user presses the quit button. During each iteration, the following actions are executed:

- The program sends the "E_REQUEST_DATA" message and waits for the data package in response. The firmware on the microcontroller works such that the board collects data from the sensors at the startup sampling rate, and then holds it in a FIFO until the Raspberry asks for it. Then, at each request, the board will send the data collected in the interval between that request and the previous request all together in one packet.

- It iterates on the number of frames received, perform the conversion of each value of the frame in the correct Measuring Unit, and copies it in the data array of the node (vfFrames)
- It updates the number of samples received (nSamples)

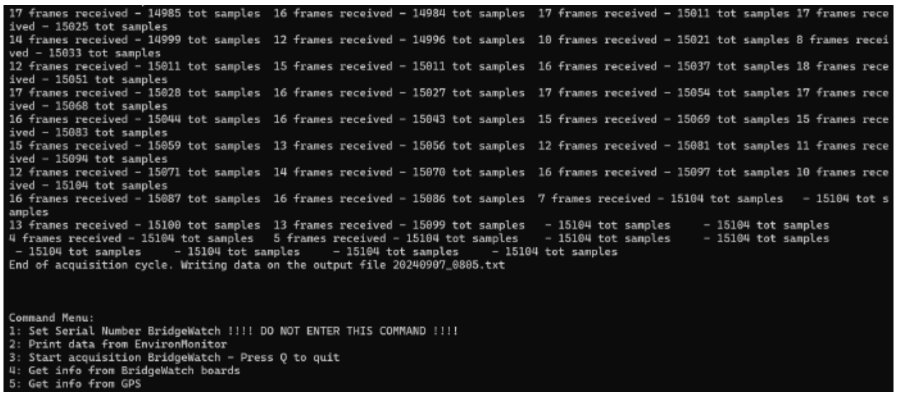
The total number of samples received is used to know when the program has to leave the for loop. In fact, the total number of frames to be received is:

$$\text{num frames} = f_s \times \text{sampling period}$$

Using a sampling frequency of 256 Hz and an acquisition period of 59 seconds:

$$\text{num frames} = 256 \times 59 = 15104 \text{ frames}$$

Then, when all the boards finished to send the correct number of frames (or the user wants to quit), all the collected data are written to a text file. The name of the text file is <year><month><day>_<hour><minute>, for example a file could be *20240829_1703.txt*. The day and the time are extract from the information sent by the GPS module connected to the Raspberry Pi. At the same time another file with the information from the EnvironMonitor board is created, with the same name but with a _EV suffix. Fig.8.3 shows a screenshot of the programme at the end of the acquisition cycle. The programme has been written so that, during the acquisition cycle, it displays the frames arriving from each board live on the screen (one column for each board).



```
17 frames received - 14985 tot samples  16 frames received - 14984 tot samples  17 frames received - 15011 tot samples 17 frames rece
ived - 15025 tot samples
14 frames received - 14999 tot samples  12 frames received - 14996 tot samples  10 frames received - 15021 tot samples 8 frames recei
ved - 15033 tot samples
12 frames received - 15011 tot samples  15 frames received - 15011 tot samples  16 frames received - 15037 tot samples 18 frames rece
ived - 15051 tot samples
17 frames received - 15028 tot samples  16 frames received - 15027 tot samples  17 frames received - 15054 tot samples 17 frames rece
ived - 15068 tot samples
16 frames received - 15044 tot samples  16 frames received - 15043 tot samples  15 frames received - 15069 tot samples 15 frames rece
ived - 15083 tot samples
15 frames received - 15059 tot samples  13 frames received - 15056 tot samples  12 frames received - 15081 tot samples 11 frames rece
ived - 15094 tot samples
12 frames received - 15071 tot samples  14 frames received - 15070 tot samples  16 frames received - 15097 tot samples 10 frames rece
ived - 15104 tot samples
16 frames received - 15087 tot samples  16 frames received - 15086 tot samples  7 frames received - 15104 tot samples   - 15104 tot s
amples
13 frames received - 15100 tot samples  13 frames received - 15099 tot samples   - 15104 tot samples     - 15104 tot samples
4 frames received - 15104 tot samples   5 frames received - 15104 tot samples    - 15104 tot samples     - 15104 tot samples
 - 15104 tot samples     - 15104 tot samples     - 15104 tot samples     - 15104 tot samples
End of acquisition cycle. Writing data on the output file 20240907_0805.txt


Command Menu:
1: Set Serial Number BridgeWatch !!!! DO NOT ENTER THIS COMMAND !!!!
2: Print data from EnvironMonitor
3: Start acquisition BridgeWatch - Press Q to quit
4: Get info from BridgeWatch boards
5: Get info from GPS
```

Figure 8.3: Screenshot of the programme at the end of the acquisition cycle

Below are the first few lines of the files produced to see the formatting:

```
Name of the file: 20240829_1703.txt
Time;Brd1_Temp;Brd1_AccX;Brd1_AccY;Brd1_AccZ;Brd1_AccTemp;Brd1_GyrX;
Brd1_GyrY;Brd1_GyrZ;Brd2_Temp;Brd2_AccX;Brd2_AccY;Brd2_AccZ;
Brd2_AccTemp;Brd2_GyrX;Brd2_GyrY;Brd2_GyrZ;Brd3_Temp;Brd3_AccX;
Brd3_AccY;Brd3_AccZ;Brd3_AccTemp;Brd3_GyrX;Brd3_GyrY;Brd3_GyrZ;
Brd4_Temp;Brd4_AccX;Brd4_AccY;Brd4_AccZ;Brd4_AccTemp;Brd4_GyrX;
```

```
Brd4_GyrY;Brd4_GyrZ
s;degC;g;g;g;degC;rad/s;rad/s;rad/s; degC;g;g;g;degC;rad/s;rad/s;
rad/s;degC;g;g;g;degC;rad/s;rad/s;rad/s;degC;g;g;g;degC;rad/s;
rad/s;rad/s
0;24.8948;0.0040741;-0.0377312;0.978764;32.1823;-0.000130495;...
0.00390625;24.8948;0.00442886;-0.0377808;0.978474;32.2928;...
0.0078125;24.8948;0.00456238;-0.0382004;0.977684;32.2928;...

Name of the file: 20240829_1703_EV.txt
TimePast;WindSpeed;WindDir;Rain;WaterLevel;Temp;Hum
s;m/s;deg;mm/h;m;degC;%
0;0.78;6;7.21;14.99;24.21;66.2;
1;0.39;6;7.19;14.99;24.21;66.2;
2;1.56;6;7.21;15;24.18;66.2;
```

The first line describes the quantities, the second line the measurement units, and from the third line on there are the data. The pseudo-code of the function is in Fig.8.4.

Now a clarification. The firmware on the microcontroller samples all quantities (i.e. external temperature, sensor temperature, accelerations and angular velocities) at 256 Hz. However, as temperatures vary much more slowly than the other quantities, it would not make sense to keep all the temperatures collected in the microcontroller's small memory and then transmit them all on the serial line. For this reason, it was decided to keep only the last ambient temperature and the last sensor temperature recorded, and to include only these in each packet sent by the board. In any case, the number of data requests to be sent is in the order of 50 per second. This means that there is no loss of temperature's variation data.

So, the parameter part of the data package obtained in response from each board is structured as follows:

- 2 bytes with the external temperature
- 2 bytes with the accelerometer temperature
- A number of structure of 15 bytes each containing the three acceleretions and three rotation velocities (3 bytes each acceleration, and 2 bytes each rotation velocity)

Get info from BridgeWatch board

This is a command commonly use during the debug phase, since it returns the minimum free space reached by the queue on the microcontroller where the frames are saved waiting to be sent. It is an index of whether a data loss is taking place or whether the allocated space is sufficient.

```
g_CGP.TxMessage(AD_CONTROLLER, SNode.nNodeNumber, E_GET_INFO,
NULL, 0)
```

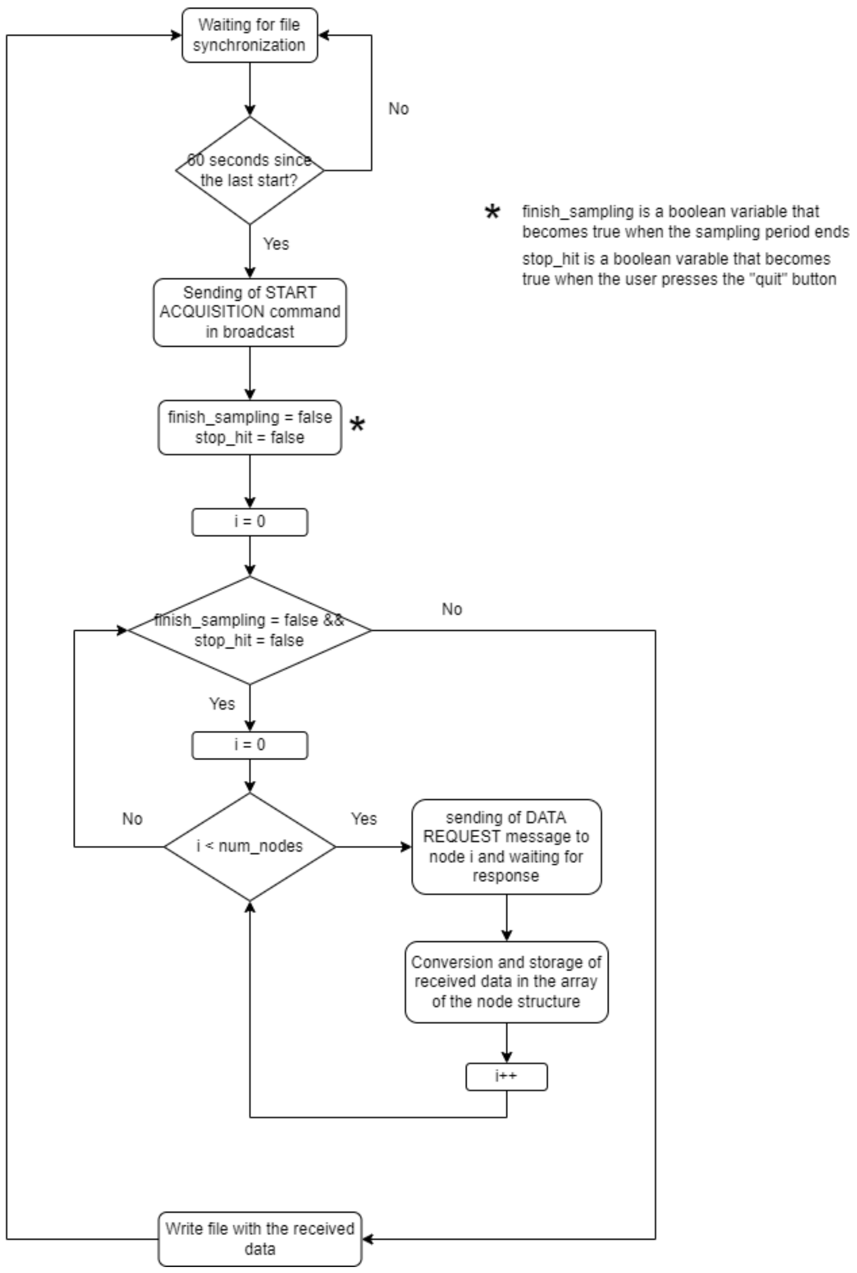

Waiting for file synchronization
60 seconds since the last start?
No
Yes
Sending of START ACQUISITION command in broadcast
finish_sampling = false
stop_hit = false
*
* finish_sampling is a boolean variable that becomes true when the sampling period ends
stop_hit is a boolean varable that becomes true when the user presses the "quit" button
i = 0
finish_sampling = false && stop_hit = false
No
Yes
i = 0
i < num_nodes
No
Yes
sending of DATA REQUEST message to node i and waiting for response
Conversion and storage of received data in the array of the node structure
i++
Write file with the received data

Figure 8.4: Pseudo-code of the acquisition function

Get info from GPS

The GPS works by sending information to a GPS receiver every second using the NMEA 0183 communication standard. It sends different sentences containing different types of data, but the programme is designed to decode only two sentences, which are the ones with the information we are interested in:

- GPGGA (Global Positioning System Fix Data): to retrieve infromation about coordinates and altitude, and the number of satellites in use.
- GPRMC (Recommended Minimum Specific GPS/Transit Data): to retrieve information about the current date and time.

The part of the program that deals with GPS communication is done in another parallel thread. This thread works in the same way as the EnvironMonitor thread: the GPS is constantly sending data, and the program is constantly waiting for a new character to arrive on the UART. When it recognises that a complete sentence has arrived, it decodes it by parsing the sentence to extract all the fields of interest and filling a data structure. The GPS data structure is therefore a variable that is updated every second with the data just received. When the command is selected, the Raspberry reads the data structure and prints the data to the console. Here there is the infinite loop of the GPS thread:

```
while (true) {
    while (pHw->RS232_Read(hUart, &cRead, 1,   100000) >= 1) {
          if (cRead == '$') {
            inSentence = true;
            sNMEAsentence.clear();
            sNMEAsentence.push_back(cRead);
          }
          else if (inSentence && cRead == '\r') {
            inSentence = false;
            parse_nmea_sentence(sNMEAsentence, SMessage);
            interpret_nmea_fields(SMessage);
            SMessage.fields.clear();
            SMessage.identifier.clear();
          }
          else if (inSentence) {
            sNMEAsentence.push_back(cRead);
          }
    }
}
```

8.2 Microcontroller firmware

8.2.1 STM32CubeIDE

The firmware for the microcontroller was written using STM32CubeIDE (version 1.13.2). It is an advanced integrated development environment (IDE) developed by STMicroelectronics, specifically designed for STM32 microcontrollers. It serves as a comprehensive platform that combines multiple tools to facilitate the development, debugging, and deployment of applications for STM32 devices. STM32CubeIDE integrates the STM32CubeMX graphical configuration tool, which simplifies the initialization and configuration of microcontroller peripherals through an intuitive interface. This environment supports a wide range of features, including project management, code editing, and debugging functionalities.

One of the main characteristics of STM32CubeIDE is its ability to generate initialization code based on the configurations set in STM32CubeMX, ensuring that developers can quickly set up their projects with minimal manual intervention. The IDE supports a variety of debugging tools and techniques, including live variable watch, real-time expression evaluation, and peripheral register view, which enhances the debugging process.

A key component of STM32CubeIDE is the possibility of using Hardware Abstraction Layer (HAL). HAL is a set of standardized APIs provided by STMicroelectronics to abstract the hardware specifics of STM32 microcontrollers, allowing developers to write code that is portable across different STM32 devices. By using HAL, developers can avoid dealing with low-level hardware details and instead focus on application logic. This layer significantly reduces the complexity and effort involved in configuring and managing the microcontroller peripherals. In this project, I have chosen to use the Hardware Abstraction Layer (HAL) instead of lower-level abstractions like the Low-Layer (LL) library, since it simplifies the configuration and control of hardware peripherals.

8.2.2 BridgeWatch .ioc file

The .ioc file in STM32CubeIDE is a crucial project configuration file. It stores settings for microcontroller peripherals, pin configurations, clock configurations, and middleware options. This file allows developers to visually configure their STM32 microcontroller and then generate the corresponding initialization code automatically. Writing the firmware starts from the base automatically generated by compiling the .ioc file, so the first step is to correctly configure it.

The .ioc file of BridgeWatch board is in Fig.8.5.

As it can be seen, it graphically represents all the microcontrollers' pins. By activating the desired peripherals, the programme automatically lights up the pins used by them in green. Let us now analyse the peripheral configurations used by BridgeWatch.

SPI

One of the main goals of the BridgeWatch board is to collect the data from the structural sensors (accelerometer, gyroscope, PT100 (with the RTD to digital converter)). All the sensors listed above used the SPI interface to communicate, so it is necessary having this pheripheral active on the microcontroller. To implement such an interface, 3 pins of the microcontroller serve as SCK, MOSI, and MISO, and other 3 pins have been configured as GPIO output and act as the chip select (CS). Then the SCK, MISO, and MOSI have been connected to those of the sensors, and each CS to that of the corresponding sensor, as reported in Fig.7.2.

The parameters of the SPI are in Fig.8.6. Reading the datasheets, the RTD to digital converter has the most stringent maximum SPI clock equivalent of 5MHz, so a prescaler of 32 is a good value to lower the initial frequency of 80 MHz. Since each sensor needs a different configuration of CPOL and CPHA, they were initialized and then they are changed run time before each communication.

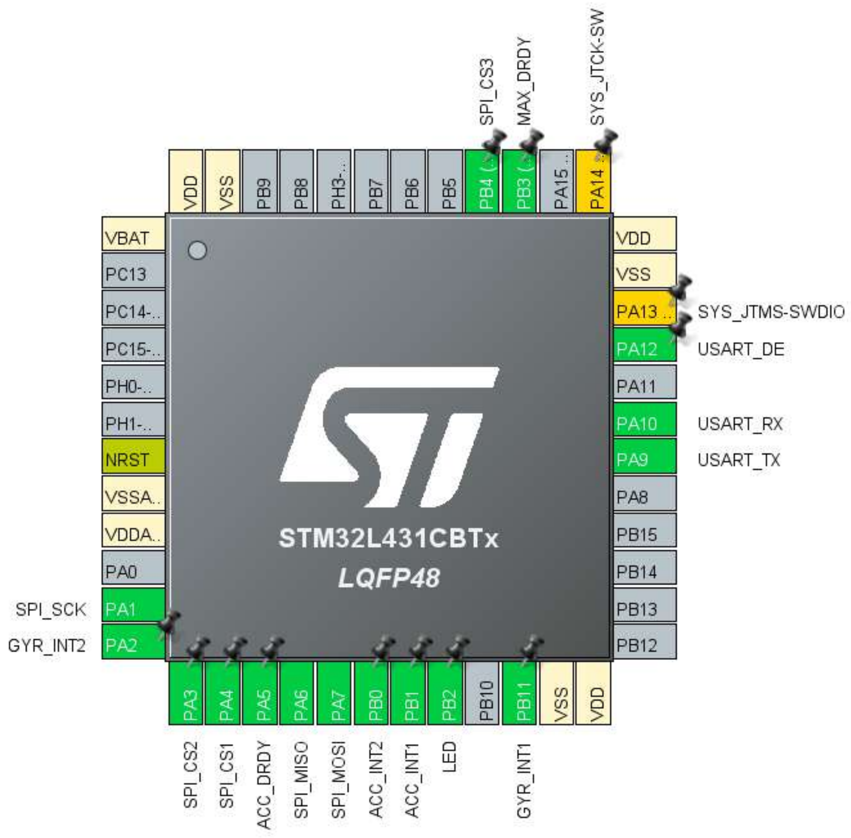


Figure 8.5: BridgeWatch .ioc file

UART

The second main goal of the BridgeWatch board is to send the collected data to Raspberry. To do so, the UART pheripheral is needed. The name on the microcontroller is USART since it can also be synchronous, but we use it in asincronous mode. To make the program as general as possible, the baud rate is set to 115200 which is a standard value, but it will be then changed runtime depending on the information sent by Raspberry.

An important parameter is the DMA which is enabled both in transmission and reception. Performance and efficiency gains are significant whit the enabling of DMA since it allows data to be transferred directly between memory and the UART peripheral, without involving the CPU for each byte of data. This relieves the CPU of the burden of managing individual data transmissions and receptions, reducing latency, improving data throughput and freeing up processor resources for other tasks. DMA is particularly beneficial in high-speed communication scenarios or when dealing with large amounts of data, as it minimizes the risk of data loss or buffer overflow by ensuring that data is handled efficiently and consistently.

Another important question is the need for a GPIO pin in Output mode to control the

DE (Data Enable) of the RS485 transceiver. The DE pin on the transceiver is used to control the direction of data flow—whether the transceiver is in transmit mode or receive mode. This is necessary because RS485 is half-duplex, meaning thatit cannot send and receive data at the same time on the same bus.

- Transmit Mode: When the DE pin is set high (logic level 1), it enables the transmitter. This allows data from the UART to be sent out over the RS485 bus. The transceiver drives the bus lines with the data from the UART.
- Receive Mode: When the DE pin is set low (logic level 0), it disables the transmitter and enables the receiver. In this state, the transceiver listens to the RS485 bus and sends any incoming data to the UART for processing.

GPIO

GPIO are the General Purpose Input/Output pins. In addition to those necessary for the SPI and UART (all configured as outputs), other pins must be enabled to allow the correct setting of the accelerometer, the gyroscope, the RTD-to-digital converter, and the LED. In particular:

- MAX_DRDY is connected to the RTD-to-digital converter, and is used from it to signal that a conversion is ready. It is an input pin.
- ACC_INT1 and ACC_INT2 are two general purpose interrupts used by the accelerometer to signal different events. They are input pins.
- GYR_INT1 and GYR_INT2 are two general purpose interrupts used by the gyroscope to signal different events. They are input pins.
- LED pin is configured as output and is used to control the pull-up configuration of the LED.

Timer

The timer is used in PWM mode to generate the signal required to sample at 256 Hz. In fact, the accelerometer can be set to operate in a mode in which one of its dedicated pins is connected to an external signal and the accelerometer samples on each rising edge of that signal. Channel 1 of timer 2 was then activated in 'PWM generation CH1' mode, which means that the generated PWM is output on a pin of the microcontroller. The main parameters were then set, as shown in Fig.8.7:

- The prescaler is set to 0 and the clock source is set to "internal clock source", so the frequency is of 80 MHz.
- The counter period determines the frequency of the PWM signal. It defines the maximum value the timer's counter will count up to before it resets back to zero. In other words, it specifies the total duration of one PWM cycle (the period of the

waveform). By setting the counter period, you effectively set the PWM frequency. The PWM frequency is calculated using the formula:

$$\text{PWM Frequency} = \frac{\text{Timer Clock Frequency}}{(\text{Counter Period} + 1)}$$

In our case, we want a PWM frequency of 256 Hz, so we have:

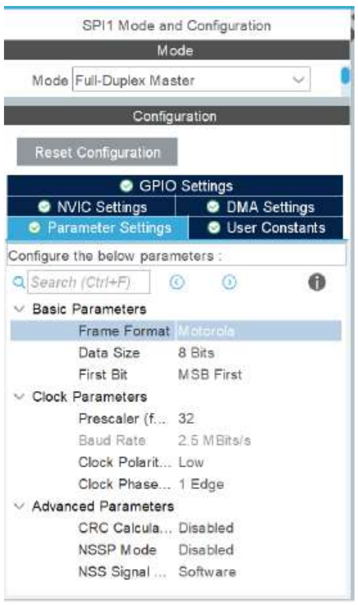
$$256\ \text{Hz} = \frac{80\ \text{MHz}}{(\text{Counter Period} + 1)}$$

and this leads to a counter period of [(80M/256) - 1] = 312499.

- The pulse value controls the duty cycle of the PWM signal, which is the proportion of time the signal remains high (active) during a single period. The duty cycle is calculated as:

$$\text{Duty Cycle} = \frac{\text{Pulse}}{(\text{Counter Period} + 1)} \times 100\%$$

  To have a duty-cycle of 50%, the pulse is set to 156250, which is the half of the counter period.



SPI1 Mode and Configuration

Mode

Mode: Full-Duplex Master

Configuration

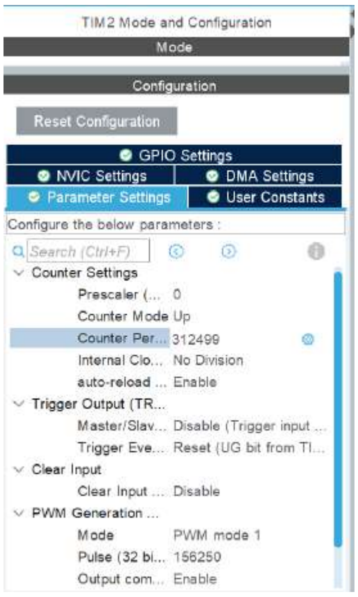
Configure the below parameters :

Basic Parameters
- Frame Format: Motorola
- Data Size: 8 Bits
- First Bit: MSB First

Clock Parameters
- Prescaler (f...: 32
- Baud Rate: 2.5 MBits/s
- Clock Polarit...: Low
- Clock Phase...: 1 Edge

Advanced Parameters
- CRC Calcula...: Disabled
- NSSP Mode: Disabled
- NSS Signal ...: Software

Figure 8.6: SPI parameters



TIM2 Mode and Configuration

Mode

Configuration

Configure the below parameters :

Counter Settings
- Prescaler (...: 0
- Counter Mode: Up
- Counter Per...: 312499
- Internal Clo...: No Division
- auto-reload ...: Enable

Trigger Output (TR...
- Master/Slav...: Disable (Trigger input ...
- Trigger Eve...: Reset (UG bit from TI...

Clear Input
- Clear Input ...: Disable

PWM Generation ...
- Mode: PWM mode 1
- Pulse (32 bi...: 156250
- Output com...: Enable

Figure 8.7: Timer parameters

8.2.3 EnvironMonitor .ioc file

The .ioc file of EnvironMonitor board is in Fig.8.5.

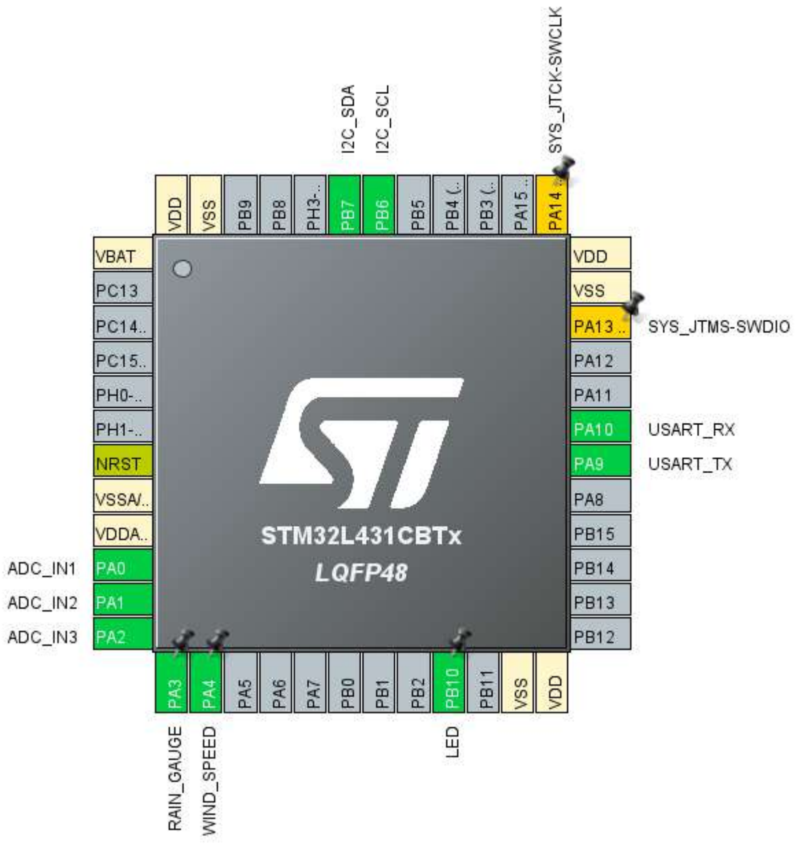


Figure 8.8: EnvironMonitor .ioc file

I2C

The I2C peripheral is used to allow the communication between the microcontroller and the thermohygrometer. When the peripheral is enabled, two pins become green: I2C_SDA and I2C_SCL. As said in Cap.7, these are the only two needed signal.

UART

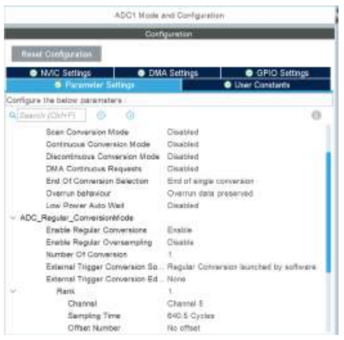
The configuration of the UART is the same as for the BridgeWatch board. The only difference is that in this case there is no need for a pin to command Data Enable of transceiver as we are using the RS232 bus instead of RS485. For a UART to RS232 transceiver, the data enable pin is not needed because RS232 communication is full-duplex, meaning it has separate lines for transmitting (TX) and receiving (RX) data.

ADC

The Analog-to-Digital Converter present on this microcontroller has a 12-bit resolution. It is used to sample three signals:

- The battery voltage, in order to be able to know its state at any time.
- The output of the water level sensor. It is a current in the range from 4 to 20 mA, and it is converted into voltage by means of a precision resistor.
- The output of the anemometer for wind direction, which is already a voltage.

In order to do so, three channel of the ADC have been enabled as single-ended input. The main setting parameters are shown in Fig.8.9. The ADC is configured such that it does a single conversion, after which the EOC flag is set. The channel to be converted is initially set to Channel 5 and then changed run-time in a for loop that spans all the enabled channels. The sampling time set to 640.5 Cycles means that the input will be sampled for a duration corresponding to 640.5 ADC clock cycles, which is needed to allow sufficient time for stable signal acquisition.



```
Scan Conversion Mode               Disabled
Continuous Conversion Mode         Disabled
Discontinuous Conversion Mode      Disabled
DMA Continuous Requests            Disabled
End Of Conversion Selection        End of single conversion
Overrun behaviour                  Overrun data preserved
Low Power Auto Wait                Disabled
ADC_Regular_ConversionMode
Enable Regular Conversions         Enable
Enable Regular Oversampling        Disable
Number Of Conversion               1
External Trigger Conversion So...  Regular Conversion launched by software
External Trigger Conversion Ed...  None
  Rank                             1
    Channel                        Channel 5
    Sampling Time                  640.5 Cycles
    Offset Number                  No offset
```

Figure 8.9: ADC parameters

GPIO

Three GPIO pins are needed:

- RAIN_GAUGE is a pin configured as EXTI (External Interrupt). EXTI pins are used to generate interrupts based on external events. The EXTI mechanism allows certain GPIO pins to be configured to detect changes (such as rising edges, falling edges, or both) and generate interrupts accordingly. In this case it has been configured to trigger a rising edge of the external signal, which is the output of the rain gauge.
- WIND_SPEED is configured as EXTI. Also in this case it triggers a rising edge of the external signal, which is the output of the anemometer for wind speed.
- LED, configured as Output. It is used to control the pull-up configuration of the LED.

8.2.4 Commands analysis

All the actions previously analysed from the "Raspberry's side" will now be analysed from the "microcontroller's side".

Initializaton of the communication system

The "SET_ADDRESS" message is sent in broadcast to all boards, and has as parameters the serial number and the corresponding node number. So, when the message arrives, each boards firstly checks if the serial number is the same as its own, and if so it saves its node number in a variable and answers sending to Raspberry its configuration structure with the command "E_PUT_CONFIG". This allows also the Raspberry to store all the important information about each node in a special structure.

To complete the initialisation of the system, the Raspberry then sends the "INIT" command with the sampling frequency as a parameter (the sampling frequency is set in the main program and is not fixed, so the program is as general as possible). When the microcontroller receives this command, it executes the following steps, as shown in the following part of the code:

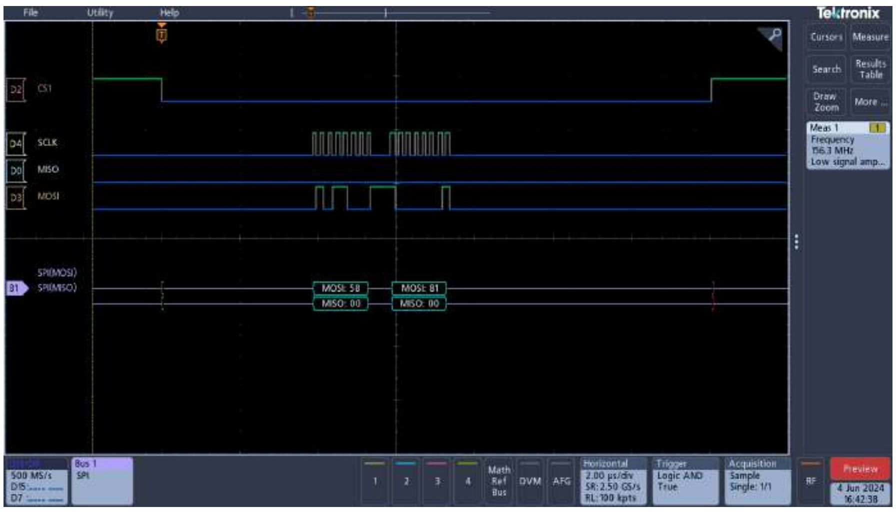
- This updates the timer parameters to generate the correct PWM. This will be used as the input signal to sample the data.
- It initialises all the sensors (accelerometer, gyroscope and RTD-to-digital converter) using custom functions.

```
case E_INIT:
    nSamplingFreq = ((DWORD*)SGP.m_vbRxMsgData)[0];

    GP_AckRxMessage(&SGP); //send the acknowledge

    nPeriodTim = FREQ_CLK / nSamplingFreq;
    nPulsePWM = nPeriodTim / 2; // duty-cycle 50%

    // UPDATE TIMER
    // Update timer period
    __HAL_TIM_SET_AUTORELOAD(&htim2, nPeriodTim);
    // Update pulse value
    __HAL_TIM_SET_COMPARE(&htim2, TIM_CHANNEL_1, nPulsePWM);

    // SENSORS INITIALIZATION
    adxl355_init(E_RANGE_2G, E_ODR_250, E_EXT_SYNC_INT_CLK);
    max31865_init();
    i3g4250d_init();

break;
```

This is an extract of the function used to initialize the accelerometer, where I set the range equal to 2g:

```
// SET RANGE
val = adxl355_readReg(RANGE);
val &= 0xFC; //set to 0 bit 0,1
switch(eRange) {
   case E_RANGE_2G: //01
       val |= 0x01;
       break;
   case E_RANGE_4G: //10
       val |= 0x02;
       break;
   case E_RANGE_8G: //11
       val |= 0x03;
       break;
   default:
       break;
}
adxl355_writeReg(RANGE,val);
```

According to the data sheet, the RANGE register has the address 0x2C. To perform a write operation, the byte containing the address must be shifted one bit to the left and the LSB must be equal to 0. Then we must send the byte containing the data to be written. Fig.8.10 shows a screenshot of the oscilloscope during transmission: the first byte (0x58) is the address shifted one bit to the left, while the second byte ends with 0x1 to set the selected range value.



Meas 1
Frequency
156.3 MHz
Low signal amp...

MOSI: 58
MISO: 00
MOSI: 81
MISO: 00

500 MS/s
D15
D7

Bus 1
SPI

Horizontal
2.00 µs/div
SR: 2.50 GS/s
RL: 100 kpts

Trigger
Logic AND
True

Acquisition
Sample
Single: 1/1

4 Jun 2024
16:42:38

Figure 8.10: Setting the range value in the accelerometer

Set serial number of BridgeWatch

Each board uses a data structure to save its main settings. The structure is written to the flash on the first boot and then on each subsequent boot it is read from the flash and saved to the RAM. This structure comprises four fields:

- Serial number
- Hardware version
- Firmware version
- Version of the structure itself

As said before, the "SET_SERIAL_NUMBER" command is sent in broadcast but with only one board connected to the bus at a time. When the board receives this command, firstly it checks if its actual serial number is zero, and then continues to execute actions only if the condition is true. In fact, initially the flash memory contains all zeroes so when the board read the configuration structure from the flash the first time, it reads zero as serial number. This is an extra check we added to prevent a command sent at the wrong time from compromising the whole system. If the old serial number is zero, then the board extracts serial number and hardware version from the message sent by the Raspberry, copies them into its configuration structure (firmware version and structure version are already written as definitions in the firmware), and finally writes the structure on the flash,

Collection of data from sensors

All the sensors listed above used the SPI interface to communicate. To implement such an interface, 3 pins of the microcontroller serve as SCK, MOSI, and MISO, and other 3 pins have been configured as GPIO output and act as the chip select (CS). Then the SCK, MISO, and MOSI have been connected to those of the sensors, and each CS to that of the corresponding sensor, as reported in Fig.7.2.

Then, the requirements say us that the sampling frequency must be of 256 Hz. The accelerometer has a configurable setting that allows a pin to be connected to an external signal, and to sample on the rising edge of that signal. So, the microcontroller has been used to generate a PWM of the desired frequency on a GPIO configured as output, and then that pin was connected to the trigger pin of the accelerometer. Since the accelerometer set a flag when it finished the measure, I decided to use that flag as the trigger signal to read the accelerations, the rotations from the gyroscope, and the temperature from the RTD to digital converter. The gyroscope, on the other hand, has been configured to run at the maximum sampling frequency, so that when the accelerations are ready, the rotations in its registers are as up-to-date as possible. This because it hasn't a programmable sampling frequency of 256 Hz and to coordinate the two sensor as much as possible to do the acquisition at the same time. For the RTD to digital converter, the configurable one-shot conversion setting was used. The 3 rotations (one for each axis), 3 angular velocities, and the temperature are saved in a single data structure, which is then added to a queue waiting to be sent to the Raspberry.

Print data from EnvironMonitor board

As seen before, this command has an effect only on the actions performed by the Raspberry since the microcontroller sends one packet of data each second without the need of any specific request. So, we will now analyze the part of the code relative to the collection and sending of the environmental data.

The microcontroller has to communicate with different sensors, each with its own output type. The main characteristics are summarized in the table 8.1 (and shown in Fig.7.1).

Sensor	Analog/digital	Output
Thermohygrometer	Digital	I2C interface
Anemometer for wind speed	Digital	One pulse per revolution of the Anemometer flag
Anemometer for wind directon	Analog	20 kOhm potentiometer
Rain gauge	Digital	One pulse per 0.3mm of rain falling
Water level sensor	Analog	4-20 mA

Table 8.1: Output of the environmental sensors

To do so, 2 pins were configured as GPIO EXTI (Input Pin with External Interrupt) and connected to the rain gauge and anemometer wind speed, 2 pins were configured as ADC input and connected to the water level sensor and anemometer for wind direction, and 2 pins have been used for the I2C communication (SDA and SCL). The scheme of the pseudo-code is in Fig.8.11.

Each second the microcontroller:

- Interrogates the thermo-hygrometer and converts temperature and humidity in the correct measurement units. Then saves them in two variables.
- Forces the ADC to make a conversion on the channel connected to the outputs of the anemometer for wind direction, water level sensor, and battery voltage. Then, converts the digital values sampled by the ADC to the correct measurement units, and saves them in three variables.
- Creates a NMEA data packet also with the data from rain gauge and from anemometer for wind speed (which are in two variables constantly updated by the EXTI callback) and send them with the UART.

As just mentioned, at the same time the microcontroller is always ready to handle the events of a new tick from the rain gauge or anemometer for wind speed as soon as they arrive:

- When a new tick from the rain gauge arrives, a function calculates the time elapsed since the last interrupt and, from it, the rainfall rate in mm/h. When a new tick arrives the microcontroller also starts a timer to count 10 minutes, after which the

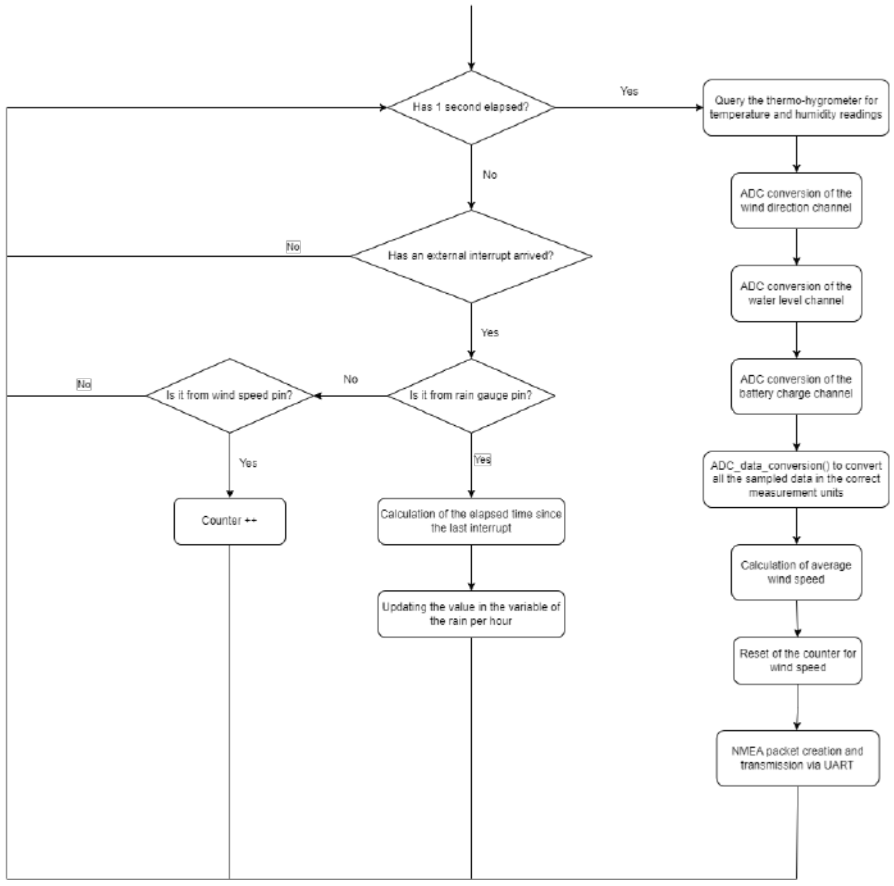

Has 1 second elapsed?
Yes
Query the thermo-hygrometer for temperature and humidity readings
No
ADC conversion of the wind direction channel
No
Has an external interrupt arrived?
ADC conversion of the water level channel
Yes
No
No
Is it from wind speed pin?
Is it from rain gauge pin?
ADC conversion of the battery charge channel
Yes
Yes
ADC_data_conversion() to convert all the sampled data in the correct measurement units
Counter ++
Calculation of the elapsed time since the last interrupt
Calculation of average wind speed
Updating the value in the variable of the rain per hour
Reset of the counter for wind speed
NMEA packet creation and transmission via UART

Figure 8.11: EnvironMonitor pseudo-code

value of the rain quantity saved is set to zero. This is necessary because otherwise, when the rain stops, the amount of rain in mm/h remains the same.

- When a new tick from the anemometer for wind speed arrives, a counter is incremented. Then, to calculate the average wind speed, I used the formula given on the anemometer datasheet which is: V = P(2.25/T), where:
  - V = speed in mph
  - P = no. of pulses per sample period
  - T = sample period in seconds

  I used 1 second as the sample period (T), so every second, in addition to doing everything in the list above, it also resets the pulse counter to zero. To get the m/s unit of measurement, I divided the obtained result by 1.6 to get km/h and by 3.6 to get m/s.

Chapter 9

Cost analysis

As one of the basic requirements of this system is that it should be low cost, we will now carry out a cost analysis to assess the final cost, including everything, for each board. The prices of all the components and the printing of the PCB are included in the analysis, but the price of the machine soldering is not included as we decided to solder everything by hand in the lab to save money. The component prices listed in 9.1, 9.2, 9.3 are those on the Digikey (*Digikey* n.d. or RS (*RS* n.d.) or Amazon (*Amazon* n.d.) websites as of today, 28/08/2024.

- The total price of the prototype system that includes 4 BridgeWatch boards and 1 EnvironMonitor board and doesn't include the rain gauge and water level sensor is:

$$4 * 178 + 310 + 785 = 1807 \approx 1800\text{€ (VAT excluded)} \tag{9.1}$$

- The total price of the complete system that includes 10 BridgeWatch and all sensors would be:

$$10 * 178 + 1500 + 785 = 4065 \approx 4100\text{€ (VAT excluded)} \tag{9.2}$$

Two considerations:

- These prices do not include mechanical supports such as plates, brackets, etc. (also custom made), the cost of the pole for the bridge, and other costs such as the support for mounting the solar panel and the water level sensor.
- These prices do not include VAT. In Italy VAT is 22% of the total price.

In terms of the running costs of the entire bridge-mounted system, the following must be considered:

- The cost of the electrical energy consumed
- The cost of the SIM card used to send the collected data to the server

BridgeWatch	
Component	**Unitary Cost (VAT excluded) [€]**
Microcontroller	2.04
Accelerometer	73.97
Gyroscope	9.96
Transceiver	0.94
DC-DC converter	2.30
RTD-to-digital converter	8.84
JTAG connector	1.28
PCB mount header 4 position (3)	3.03
Other components (resistors, capacitors, inductors, diodes, etc.)	≈ 5
PCB printing	7.84
PT100 (external)	18.66
Protection box	43.66
Total	**177.52 $\approx$ 178**

Table 9.1: BridgeWatch cost analysis

EnvironMonitor	
Component	**Cost (VAT excluded) [€]**
Microcontroller	2.04
DC-DC converter 12V to 3.3V	2.30
DC-DC converter 12V to 5V	19.71
Transceiver	1.38
Opamp (2)	4.70
PCB mount header 3 position (3)	2.55
PCB mount header 4 position (2)	2.02
2.5V voltage reference	0.91
JTAG connector	1.28
USB-A connector	0.51
Other components (resistors, capacitors, inductors, diodes, etc.)	≈ 5
PCB printing	8.31
Rain gauge (external)	261.39
Anemometer (external)	235.95
Thermohygrometer (external)	24.21
Water level sensor (external)	927
Total without water level sensor and rain gauge	**310.87 $\approx$ 310**
Total	**1499.26 $\approx$ 1500**

Table 9.2: EnvironMonitor cost analysis

Mechanical Supports and other components	
Component	**Cost (VAT excluded) [€]**
Box for EnvironMonitor, Raspberry and battery	235.33
Cable	300
Raspberry Pi 4	61.46
Battery	63.85
Solar panel	122.94
Total	**783.58 $\approx$ 785**

Table 9.3: Mechanical supports and other components cost analysis

Chapter 10

Laboratory validation

The test has been done without using the battery for power and using the laptop instead of the Raspberry as the central computer, which makes the activity much less time-consuming. Fig.10.1 shows a diagram of the connections made.

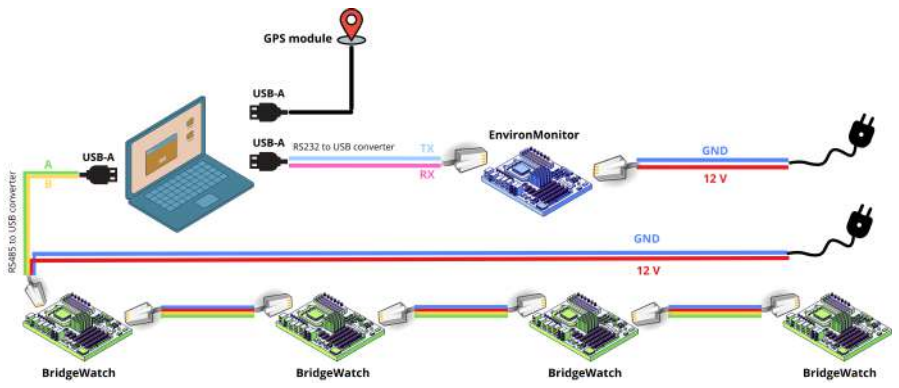


Figure 10.1: Diagram of the connections for the test

10.1 Test of the BridgeWatch boards

Three different tests were carried out to validate the BridgeWatch board system:

- A first preliminary test to check that the acquisition and communication between the sensors and the microcontrollers, and between the microcontrollers and the computer, worked correctly (Section 10.1.2).
- A test carried out in a DET (Department of Electronics and Telecommunications) laboratory using a shaker set to vibrate at different frequencies. The aim was to check that at each frequency the FFT of the data collected had a single peak at that value (Section 10.1.3).

- A test carried out in a laboratory at DISEG (Department of Structural, Civil and Geotechnical Engineering) using a model building. First an ambient vibration test and then a forced vibration test were carried out. The objective was twofold: to verify that the data obtained were consistent with the data obtained from reference accelerometers simultaneously present on the structure, and to verify that the natural frequencies extracted from the data obtained were consistent with the known ones (Section 10.1.4).

10.1.1 Cables

To give supply to the board and to allow the communication, we need some custom-made cables. Fig.10.2 represents the cable used for testing the BridgeWatch boards, while Fig.10.3 is an enlargement of the combination of power supply cable and RS485-to-USB cable into a single 4-wire cable (the yellow one), with a diagram of the connection alongside. In brief:

- The red wire of the power supply cable (12V) has been connected to the red wire of the yellow cable, which is used for power supply.

- The black wire of the power supply cable (GND) has been connected to the black wires of the other two cables, which are the Ground.

- The orange wire of the RS485-to-USB converter cable (A, see Fig.7.14) has been connected to the green wire of the yellow cable, which is used for that purpose. The yellow wire (B) has been connected to the white wire of the RS485-to-USB converter cable.

Fig.10.4 shows the connection of the yellow cable to the BridgeWatch board, where it can be seen the role of each wire.

10.1.2 Preliminary Test

A very simple preliminary test was carried out to verify the correct operation of the BridgeWatch system. To test the developed system, the four BridgeWatch boards were placed on a table connected in series, and the acquisition program was started. Then a hit was given to the table and after a short time the recording was stopped. The aim is then to plot the acceleration on the z-axis on a graph and to see a change at the moment of impact. Fig.10.5 shows the test setup. It can be seen that it fits perfectly with the scheme in Fig.10.1.

Fig.10.6 shows a plot of the z-axis acceleration recorded by each board; it can clearly be seen that the BridgeWatch boards recorded the change at the moment of impact, as there is a peak. This preliminary test was used to check that all initialisation and communication between the sensors and the microcontroller, and between the microcontroller and the computer, was working correctly.

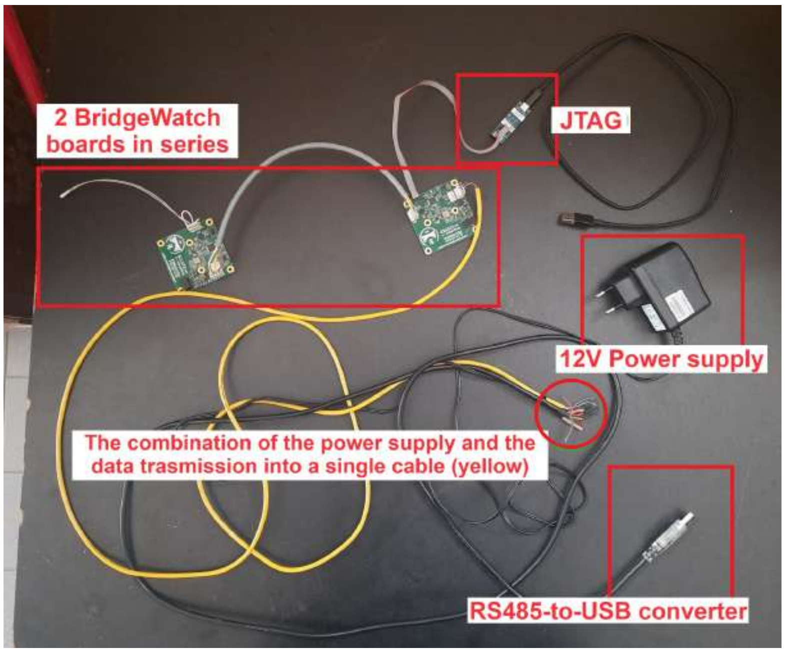

2 BridgeWatch boards in series
JTAG
12V Power supply
The combination of the power supply and the data trasmission into a single cable (yellow)
RS485-to-USB converter

Figure 10.2: BW cable (1)

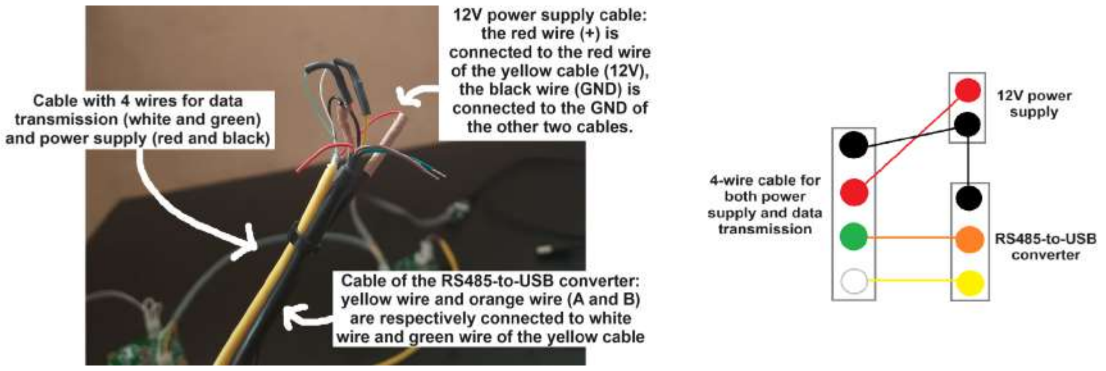

12V power supply cable: the red wire (+) is connected to the red wire of the yellow cable (12V), the black wire (GND) is connected to the GND of the other two cables.
Cable with 4 wires for data transmission (white and green) and power supply (red and black)
Cable of the RS485-to-USB converter: yellow wire and orange wire (A and B) are respectively connected to white wire and green wire of the yellow cable
12V power supply
4-wire cable for both power supply and data transmission
RS485-to-USB converter

Figure 10.3: BW cable (2)

10.1.3 Shaker Test

To test the actual operation of the BridgeWatch board, a laboratory test was carried out using a shaker.

A shaker is a device used to generate vibrations for testing and analysis purposes, commonly used in structural dynamics, vibration testing and even calibration of sensors such as accelerometers. The movement of the shaker can be very precisely controlled to produce vibrations with specific amplitudes and frequencies: it can simulate different types of vibration, from sinusoidal (pure tones) to more complex random or shock waveforms.

During the test, the shaker was programmed to produce a series of sinusoidal vibrations,

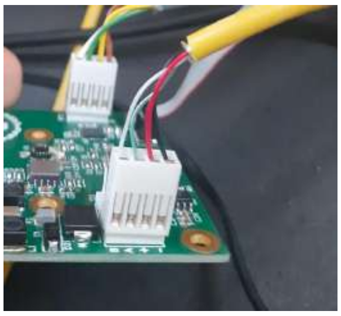


Figure 10.4: BW cable (3)

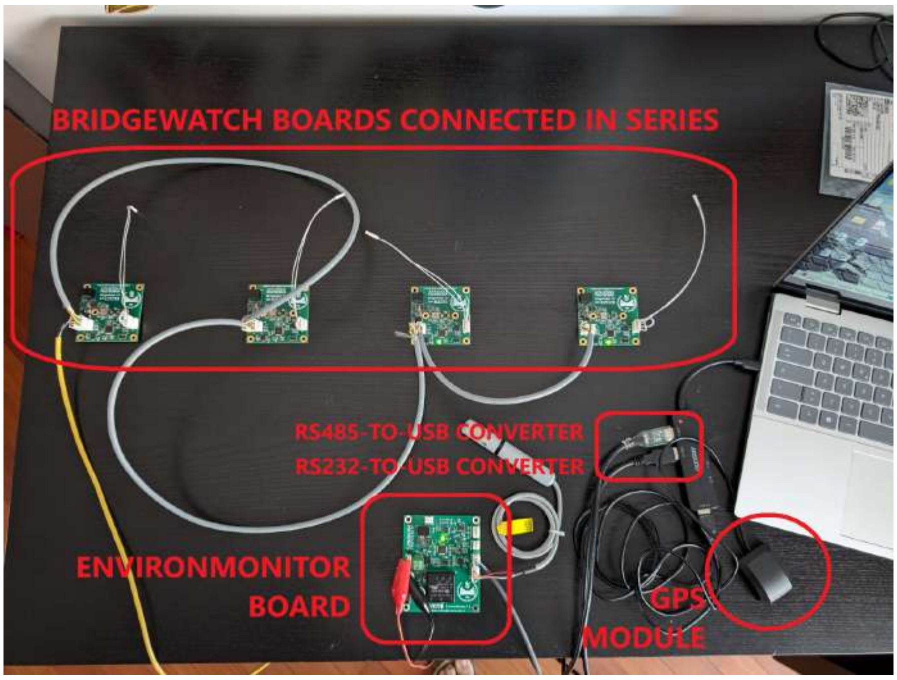


Figure 10.5: Testing of BridgeWatch board setup

each at a different frequency but with the same amplitude of acceleration. A BridgeWatch board was attached to the shaker and used to measure these vibrations. Fig.10.7 shows the shaker used and the test setup. The recorded signals were then analysed using Fast Fourier Transform to determine their frequency content. The primary objective of this analysis was to verify that the peak frequency observed in the FFT results corresponded to the frequency set on the shaker. This process ensured that the accelerometer and data acquisition system were accurately capturing and reflecting the input frequencies of the shaker.

5 different tests were carried out:

1. Frequency = 10 Hz, amplitude = 1g

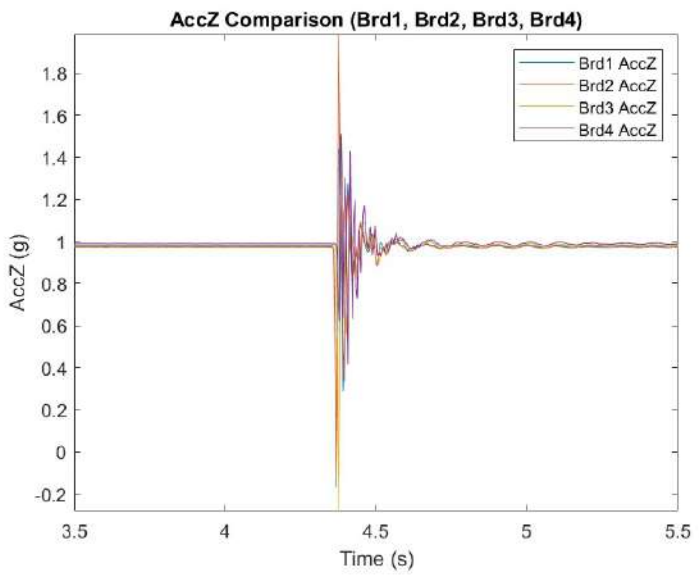


Figure 10.6: BridgeWatch testing output graph

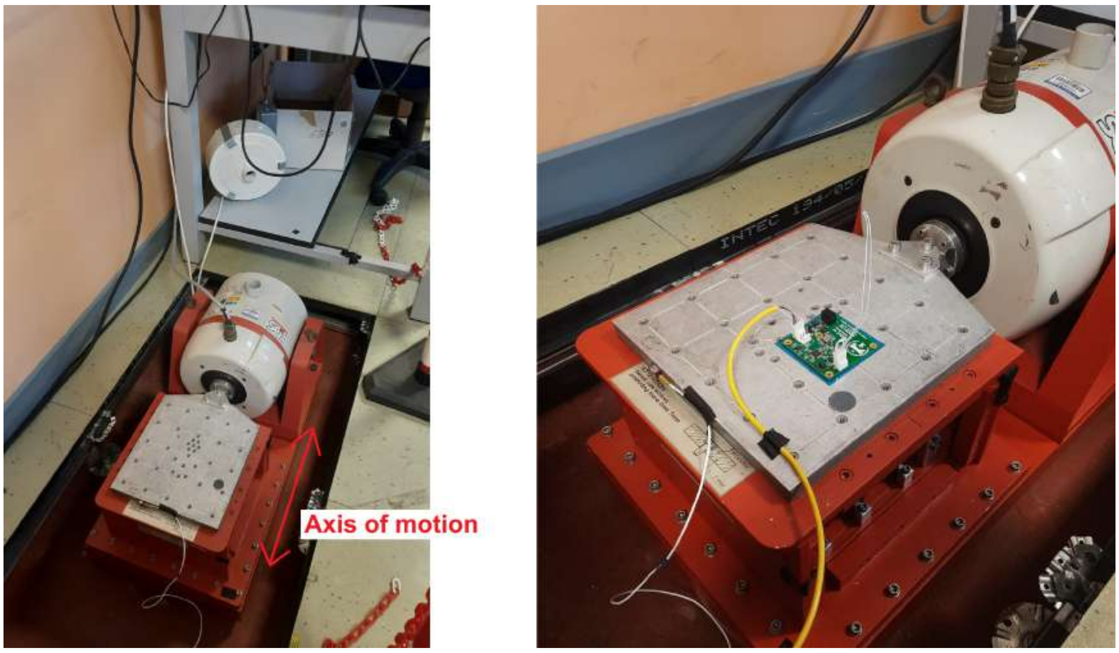


Figure 10.7: Shaker Test setup

2. Frequency = 20 Hz, amplitude = 1g

3. Frequency = 50 Hz, amplitude = 1g

4. Frequency = 100 Hz, amplitude = 1g
5. Frequency = 5 Hz, amplitude = 0.5g (It was halved for physical reasons explained later)

The Matlab code used to perform the FFT is reported in Appendix E. It takes as input a data file containing the acquired accelerations (in our case, the accelerations along the y-axis, as we used a horizontal shaker), applied an Hanning window to reduce the spectral leakage, and then performs the FFT on the windowed signal. Finally, the code plots both the time-domain signal and the frequency-domain spectrum, marking the peak frequency on the FFT plot.

Hanning window. When performing an FFT on a signal, the assumption is that the signal is periodic within the sampled time window. If the signal does not begin and end at the same value, discontinuities at the window edges can create artificial high-frequency components. This effect is called spectral leakage.

The Hanning window is a tapering function that smoothly reduces the amplitude of the signal toward zero at the start and end of the window, thereby minimizing the sharp discontinuities at the boundaries. This smooth tapering helps reduce the influence of spectral leakage, improving the clarity of the FFT results by suppressing noise and false peaks in the frequency spectrum. As a result, the true frequencies in the signal can be more easily identified, leading to more accurate analysis.

Results

Fig.10.8 and Fig.10.9 show the results for the 10 Hz input signal. The identified peak is at 10.01695 Hz, which corresponds to an error of 0.17%. The plot of the input signal is very similar to a sinusoid.

Fig.10.10 and Fig.10.11 show the results for the 20 Hz input signal. The identified peak is at 20.01695 Hz, which corresponds to an error of 0.08%. The plot of the input signal is very similar to a sinusoid.

Fig.10.12 and Fig.10.13 show the results for the 50 Hz input signal. The identified peak is at 50.06780 Hz, which corresponds to an error of 0.13%. The signal plot is starting to look a little less like a sine wave even though it still resembles it a lot. This is because we are going up in frequency.

Fig.10.14 and Fig.10.15 show the results for the 100 Hz input signal. The identified peak is at 100.11864 Hz, which corresponds to an error of 0.12%. In this case, the signal plot significantly differs from the original signal.

This is because a signal of 100 Hz is relatively close to the Nyquist frequency (128 Hz). When signals are sampled near the Nyquist frequency, fewer samples per cycle are available to represent the waveform, resulting in poor resolution of the signal and distortion in the frequency domain. At a sampling rate of 256 Hz, a 100 Hz signal results in only 2.56 samples per cycle. Ideally, you want a higher number of samples per cycle (more than 10 samples per cycle) to accurately capture the shape of the waveform. With only 2-3 samples per cycle, it's difficult to faithfully capture the details of the waveform, resulting in distortion.

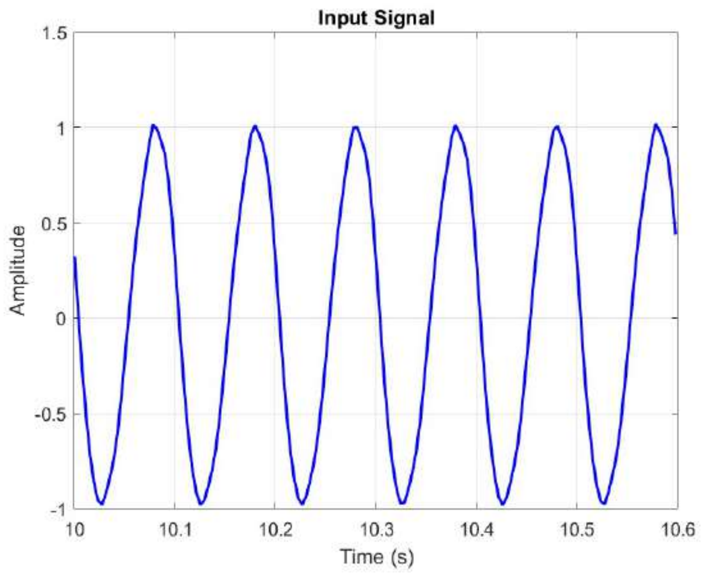


Figure 10.8: 10 Hz input signal

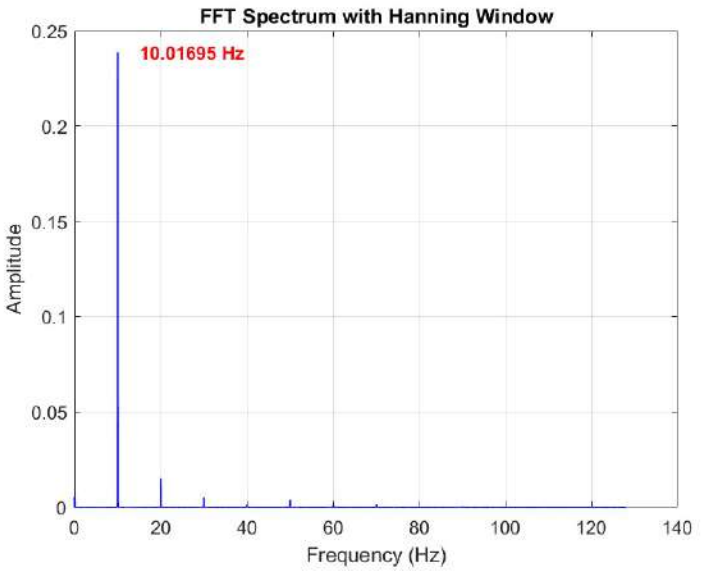


Figure 10.9: FFT of the 10 Hz sinusoidal signal

The small peak at 104 Hz (and so not on a multiple of 100 Hz) is likely due to spectral leakage. Windowing helps smooth the edges and reduce leakage, but it can't fully eliminate it.

Fig.10.16 and Fig.10.17 show the results for the 5 Hz input signal. The identified peak is at 5.00000 Hz, which is a perfect result. This is because the lower the frequency of the

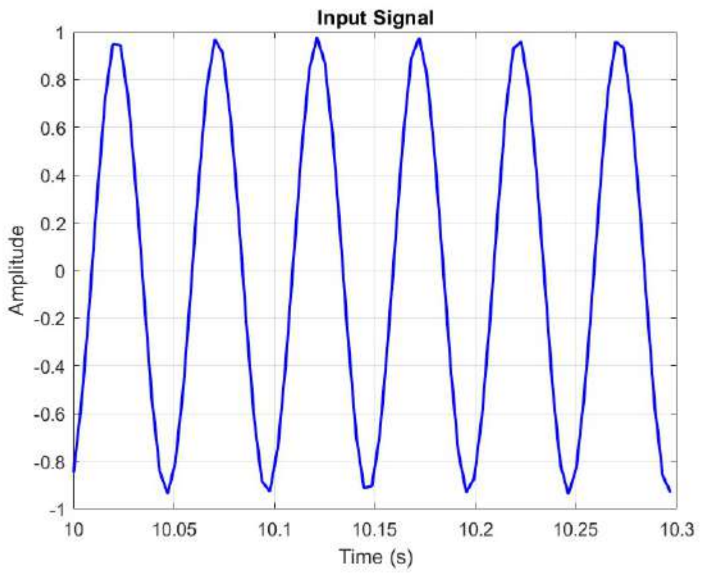


Figure 10.10: 20 Hz input signal

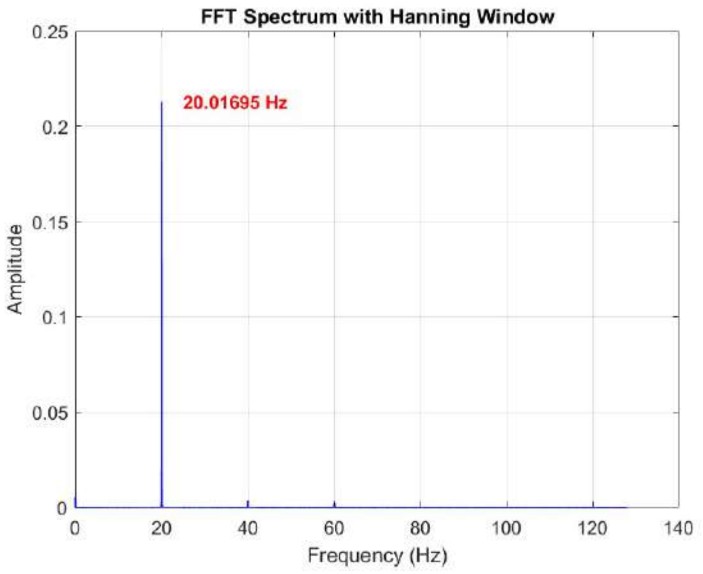


Figure 10.11: FFT of the 20 Hz sinusoidal signal

signals, the better the signal can be reconstructed, since we have more sampling points. However, as you can see, the input signal looks very different from a sine wave. This is not an error in the reconstruction, but the shaker itself, which is unable to produce a perfect sine wave at low frequencies. This is for 3 main reasons:

- Large displacements required: At lower frequencies, the shaker must move much

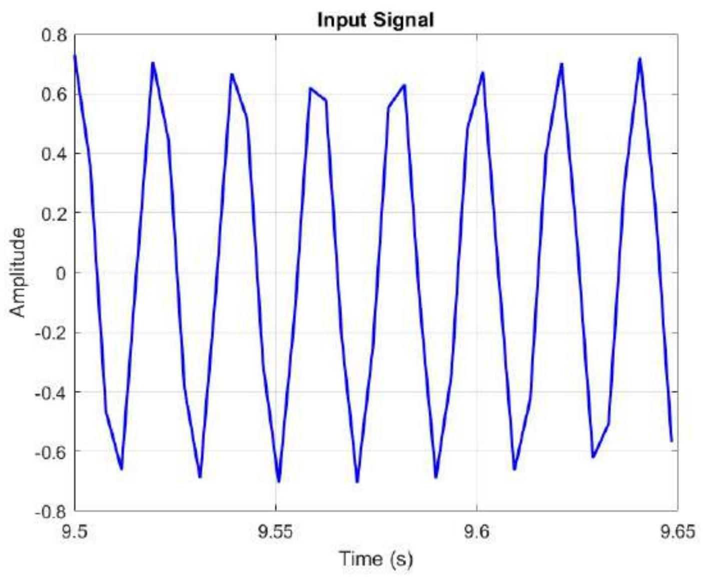


Figure 10.12: 50 Hz input signal

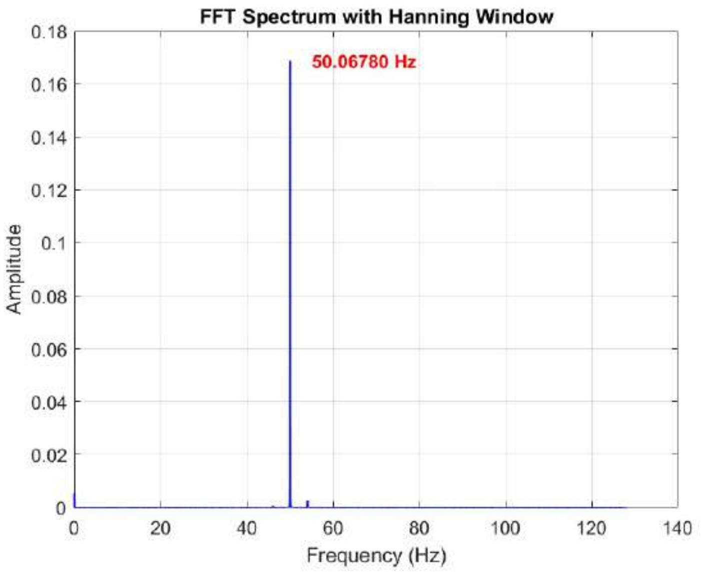


Figure 10.13: FFT of the 50 Hz sinusoidal signal

further (large displacement) to maintain the same constant acceleration, since displacement is inversely proportional to the square of frequency. This can be physically demanding for the shaker.

- Performance limitations: Because of these large displacements, the power requirements of the shaker increase at low frequencies as the shaker has to push and pull over

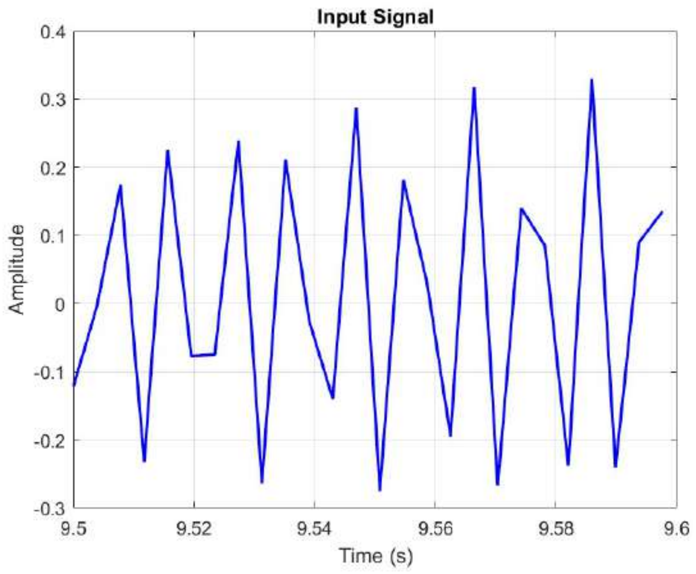


Figure 10.14: 100 Hz input signal

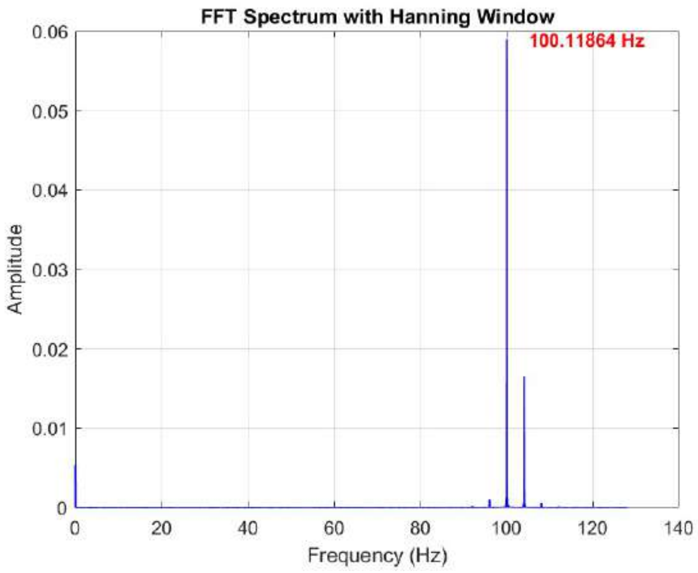


Figure 10.15: FFT of the 100 Hz sinusoidal signal

much greater distances. This makes it more difficult for the system to perform well at lower frequencies.

- Mass and inertia: At lower frequencies, the mass of the moving parts of the shaker becomes more significant. The shaker has to overcome the inertia of its moving mass over longer strokes, making it less efficient.

Because the input signal is not a perfect sine wave, we can correctly see not only the peak in the FFT diagram, but also some harmonics (at multiples of the peak frequency).

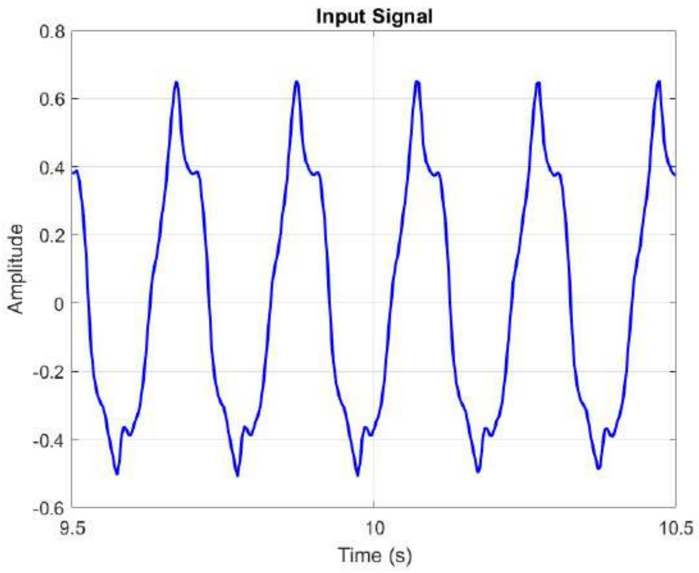


Figure 10.16: 5 Hz input signal

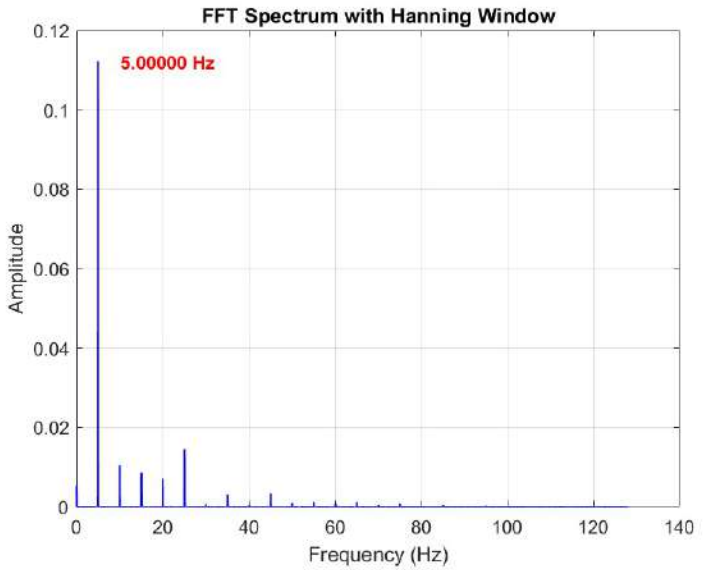


Figure 10.17: FFT of the 5 Hz sinusoidal signal

10.1.4 Model Building Test

A further test was carried out on a model structure in the DISEG laboratory. The purpose of the test was to extract the natural frequencies of the structure using the accelerations measured by the BridgeWatch system and to verify that they matched those of the structure. Pierpaolo Dragonetti in his thesis (Dragonetti et al. 2023) carried out an extensive study of the building model, extracting natural frequencies, modal shapes and damping ratios of different experimental setups by means of cameras and accelerometers, so we have some "golden values" to make the comparison. His results are reported in Tab.10.1.

Vibrational mode	**Natural frequency [Hz]**	**Damping ratio [%]**	**Interpretation**
1	2.94 ± 0.03	1.26 ± 0.49	First flexural along Y direction
2	5.96 ± 0.10	2.39 ± 0.26	First flexural along X direction
3	7.72 ± 0.05	1.97 ± 0.16	First torsional
4	8.89 ± 0.03	1.14 ± 0.31	Second flexural along Y direction
5	13.65 ± 0.03	0.59 ± 0.10	Third flexural along Y direction
6	26.43 ± 0.26	1.44 ± 0.52	Second flexural along X direction
7	30.52 ± 0.12	1.31 ± 0.13	Second torsional
8	64.04 ± 0.11	0.60 ± 0.29	Third flexural along X direction
9	73.04 ± 0.17	1.18 ± 0.29	Third torsional

Table 10.1: Identified modal shapes, natural frequencies and damping ratios of the model building

The structure in question is a three-storey structure made of extruded aluminium. It consists of four columns of rectangular section bolted to three square plates of 400 mm side to side and 300 mm distance between each plate. The connections are made by steel angles with equal flanges 20 mm wide. The frame is fixed to a slab foundation consisting of a 500x500 mm steel plate resting on four supports. Fig.10.18 gives an overview of the frame.

The experimental setup is shown in Fig.10.19. A BridgeWatch board (red circle) was placed in the centre of each plate, and in addition two other reference accelerometers (blue circle) were attached to the structure during the test phase to verify that the data acquired by the two different systems were consistent. In order to securely attach each board to the aluminium plate without damaging either one or the other, a piece of duct tape was attached to the back of the board and a piece to the plate at the point where the board was attached, and then glue was put on to stick the two pieces of tape together.

Fig.10.20 shows a closer view of the top plate of the structure, where the two reference sensors are located. There is also a representation of the reference system, which is the same for both the acquisition systems. The structure is less rigid along the X-direction, and more rigid along the Y-direction. It is important to note that the reference system used by Dragonetti is reversed compared to this one, so that the flexural modes identified by him along Y correspond to ours along X and vice versa.

The test consists of two phases:

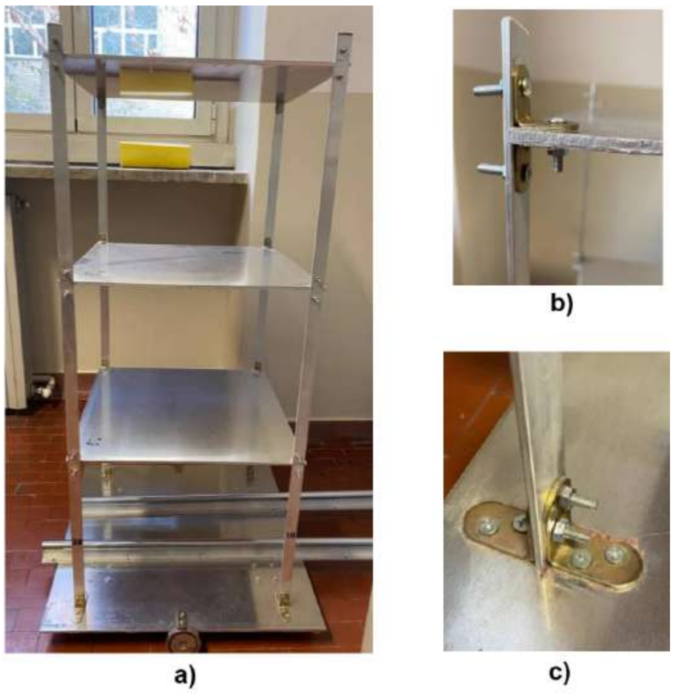


Figure 10.18: **a)** Model building. **b)** Detail of column-plate join. **c)** Representation of column-base slab joint. Source: Dragonetti et al. 2023

- **Ambient vibration test (output only):** During this phase, data acquisition was conducted for approximately 20 minutes without any external or forced input so the only vibrations captured during this period were ambient vibrations (white noise). These vibrations are very weak and random, but cover a wide range of frequencies and therefore naturally excite the resonant frequencies of the structure. After processing the data with a Fast Fourier Transform (FFT), we therefore expect the resulting graph to show distinct peaks corresponding to the natural frequencies of the structure.
- **Forced vibration test (input/output):** In the second phase, external excitation was introduced by a series of hammer blows applied at specific points on the structure. These impacts generate a delta function in the time domain, which theoretically excites all frequencies simultaneously (since the Fourier transform of a delta function is a constant over all frequencies). The accelerometers then measure the response of the system to this excitation. Similar to the environmental test, the data acquired, when analysed in the frequency domain, should show peaks at the natural frequencies of the structure, consistent with the results of the ambient vibration test.

  In particular, for this test we used a hammer and gave a series of blows to the top plate in the direction not constrained by the supports. A pause of about one minute was allowed between each blow to allow the structure time to extinguish the free decay. Then one blow at a time was extracted in the time domain and the FFT was performed on that signal.

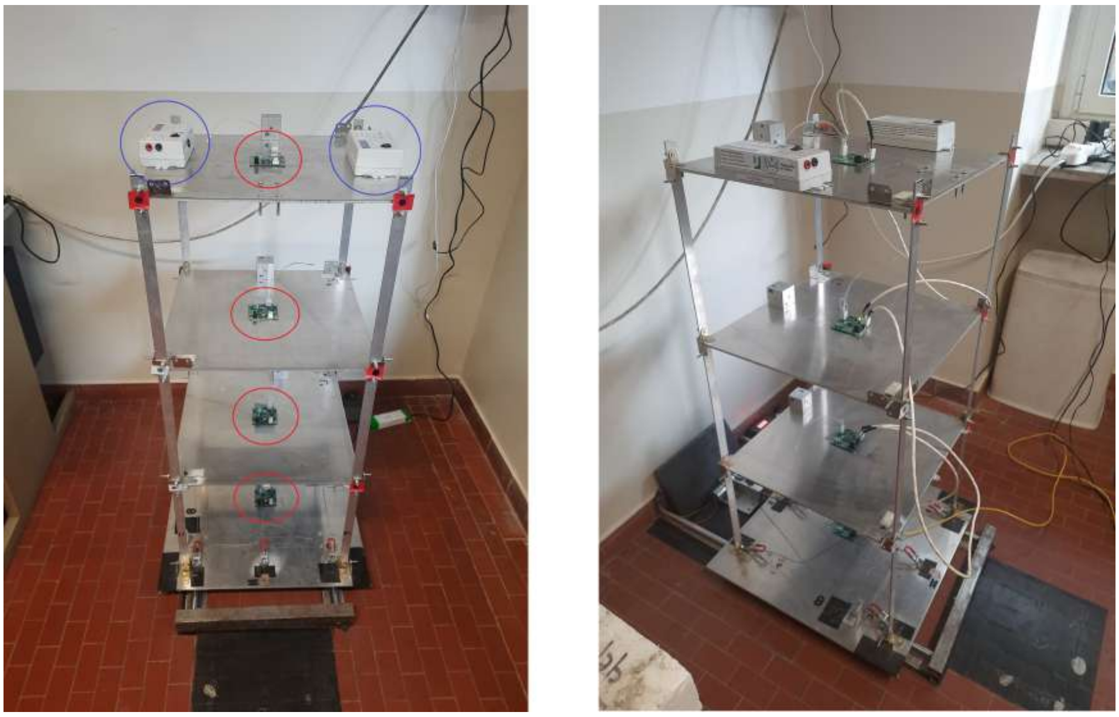


Figure 10.19: Model building test - Experimental setup

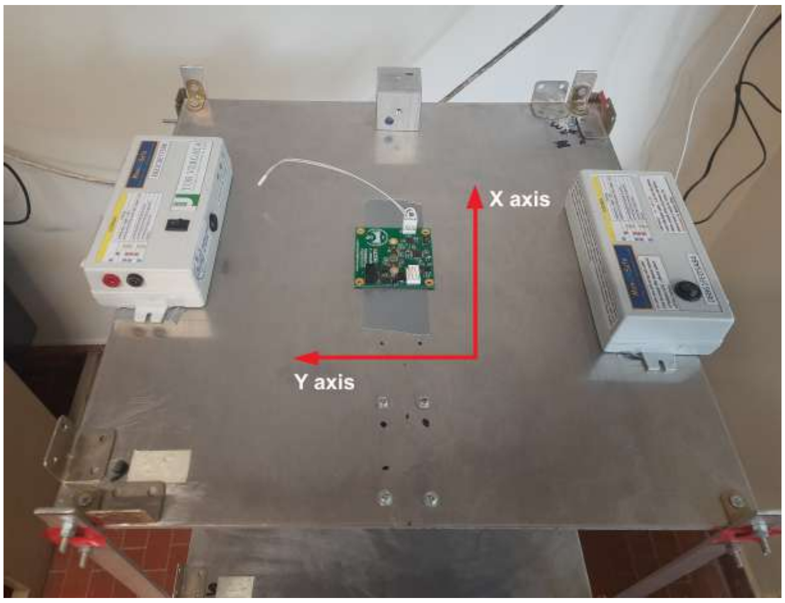


Figure 10.20: Model building - Top plate

Data analysis - Ambient vibration test. In addition to the FFT, the Stochastic Subspace Identification (SSI) algorithm was used to analyse the ambient vibration data.

The SSI algorithm is a widely used method for system identification, particularly in the field of structural health monitoring, where it is applied to output-only vibration data. Unlike traditional modal identification methods, which require both input and output data, SSI can work only with response measurements (output data), making it well suited to environmental vibration testing where the excitation is unknown or random. The SSI algorithm works by constructing a mathematical model of the dynamics of the structure. A key output of the SSI process is the stabilisation diagram: this plot shows the identified system poles (frequencies and damping ratios) for different model orders (levels of complexity). Each identified mode is plotted as a point on the diagram. Stable poles, which represent the true natural frequencies of the structure, remain consistent across different model orders, while spurious or noise modes disappear.

By superimposing the output of the FFT on the stabilisation diagram, we should therefore expect the frequencies of the stable poles to coincide with those of the peaks, and they should coincide with the natural frequencies of the structure.

I used Matlab to do the above analysis, and in Fig.10.21 there is the output graph. I have marked the frequencies of the stable poles with a green vertical line, and we can see that they correspond more or less obviously to the FFT peaks, as we would expect (the FFT has been carried out on the accelerations aquired by Board3, which is the one placed at the higher level). Their values are given in Tab.10.2. As can be seen, they are slightly shifted towards zero with respect to those extracted by Dragonetti, but they are still close. There are two main reasons for this:

- A possible reason for this could be the increase in the mass of the structure during the test compared to the setup used during the Dragonetti test. In fact, the two reference accelerometers have a non-negligible mass that could have influenced that of the structure. When the mass of a structure increases, its natural frequencies tend to decrease. This relationship can be explained using the classic formula of a mass-spring system:

$$f_n = \frac{1}{2\pi}\sqrt{\frac{k}{m}}, \tag{10.1}$$

  where f_n is the natural frequency, k is the stiffness of the system, and m is the mass. As can be seen from the formula, the natural frequency is inversely proportional to the square root of the mass. Thus, when the mass m increases, the frequency f_n decreases. This is because the larger mass resists acceleration, causing the system to oscillate more slowly.

- Another potential reason for the observed shift in natural frequencies could be the use of glue to attach the boards to the model structure. Glue can introduce additional damping to the system. Damping reduces the amplitude of vibrations and can alter the energy distribution across frequencies. This can make the natural frequencies appear lower than they actually are, as the damping dissipates vibrational energy and alters how the system responds to excitation. Furthermore, it can introduce a layer of flexibility between the accelerometer and the structure. This additional flexibility can slightly lower the stiffness (represented by k in the mass-spring system formula 10.1), resulting in a reduction in the natural frequencies.

Thus, both the increase in mass from the accelerometers and the potential reduction in stiffness from the adhesive could account for the lower natural frequencies observed in this experiment compared to those obtained in the Dragonetti's test.

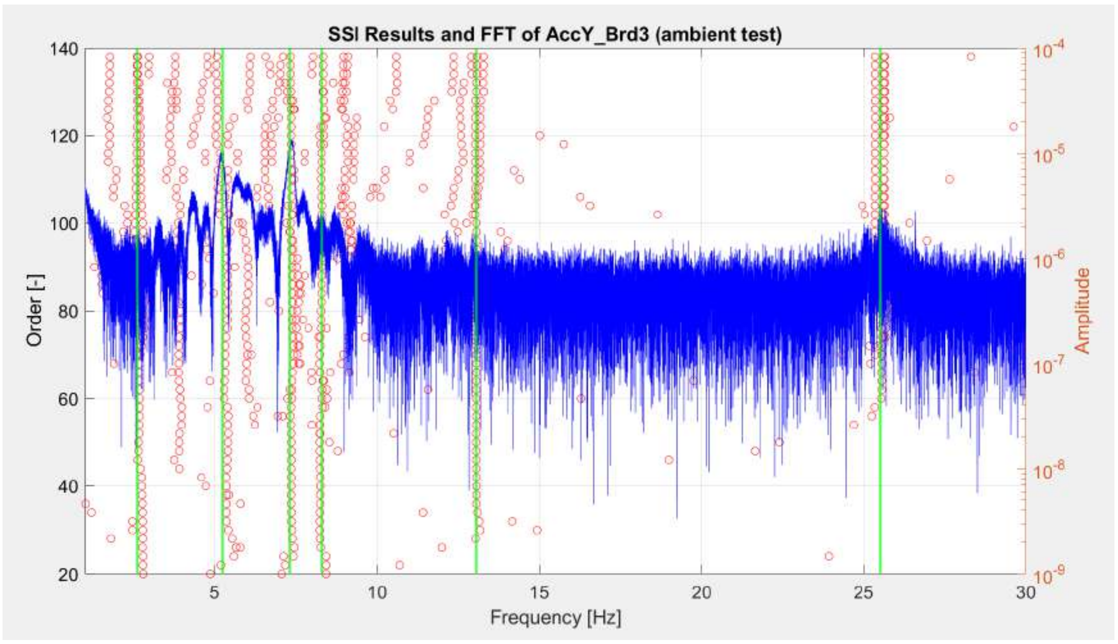


Figure 10.21: Stabilization diagram vs FFT - ambient test

Vibrational mode	Natural frequency (known value) [Hz]	Natural frequency found by the graph [Hz]
1	2.94 ± 0.03	≈ 2.60
2	5.96 ± 0.10	≈ 5.24
3	7.72 ± 0.05	≈ 7.30
4	8.89 ± 0.03	≈ 8.30
5	13.65 ± 0.03	≈ 13.06
6	26.43 ± 0.26	≈ 25.50

Table 10.2: Natural frequencies found by the graph vs values found by Dragonetti

Data analysis - Forced vibration test. Fig.10.22 shows the series of four hammers in the time domain. As mentioned above, a single hammer at a time was then extracted, and the FFT analysis was performed on this limited signal. The extraction of the single hammers in the time domain is shown in Fig.10.23, while the results for each hammering are shown in the figures 10.24, 10.25, 10.26, 10.27 respectively. In the FFT graph, you can see some green vertical lines which are placed exactly at the frequencies identified by the ambient vibration test. You clearly see that in all cases they correspond to the peaks. This is very important as it proves that the identified natural frequencies remain consistent in both tests.

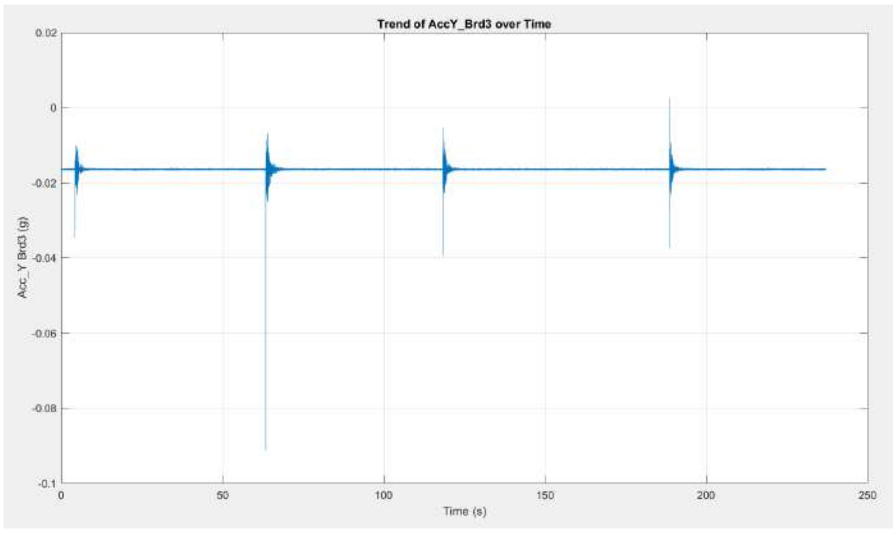


Figure 10.22: Forced vibration test - Series of hammers in the time domain

Comparison with the reference accelerometers

The reference accelerometers are wireless sensors developed by the Moni2BSafe research team led by Prof. Abruzzese of the University of Tor Vergata in Rome. The chip inside is the ADXL355 MEMS accelerometer, exactly the same as on the BridgeWatch boards.

The comparison between the data collected by the two systems was carried out for the data relating to the series of hammers given during the forced vibration test. As in the previous section, a single hammer was extracted and analysed using the FFT. The resulting plots for the two systems were then superimposed to check that the trend of the frequency response was more or less the same. This was done first for the accelerations recorded along the X-axis and then for those recorded along the Y-axis.

Fig.10.28 shows the comparison along the X-direction. It can be clearly seen that the two spectra can almost be superimposed. In particular, the position of the peaks is almost identical. The 3 main peaks are very obvious and correspond to the first 3 flexural modes along X direction. The vertical green lines represent the natural frequencies of the first 3 flexural modes along Y direction identified by Dragonetti (which, as mentioned earlier, correspond to our modes along x). Our peaks are slightly shifted towards 0 with respect to his, for the reasons explained in the previous paragraphs.

Fig.10.29 shows the comparison along the Y-direction. It can be clearly seen that the two spectra can almost be superimposed. In particular, the position of the peaks is almost identical.

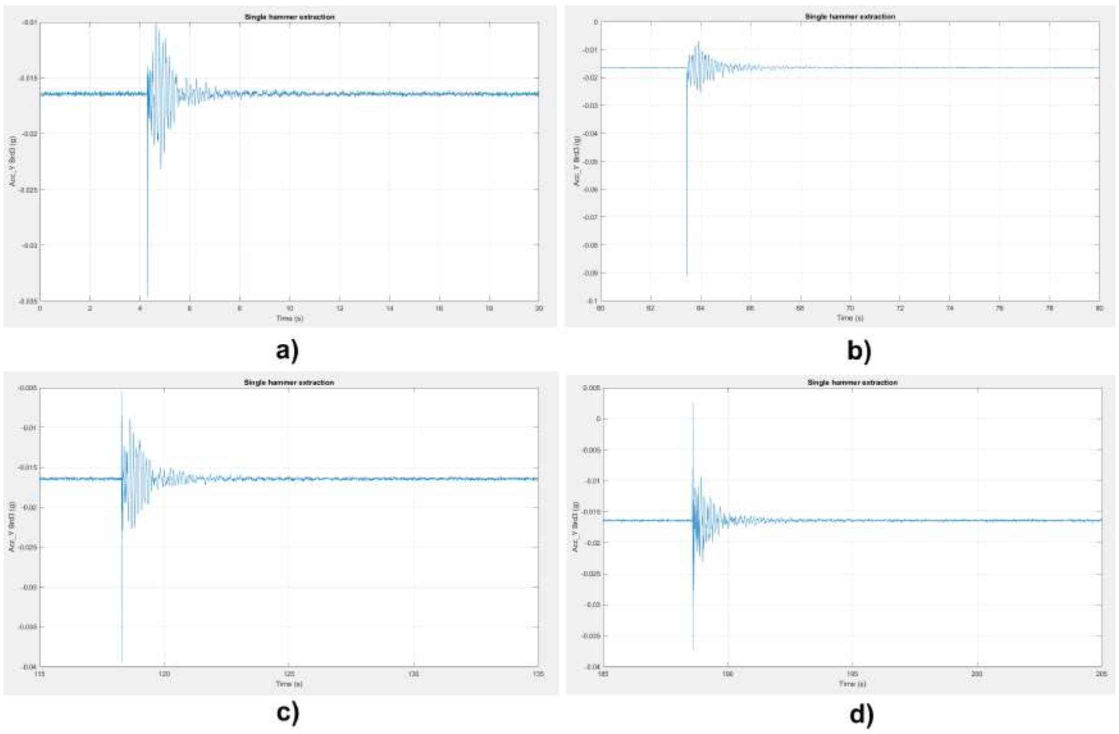


Figure 10.23: Forced vibration test - Single hammers extraction. **a)** First hit, **b)** Second hit, **c)** Third hit, **d)** Fourth hit

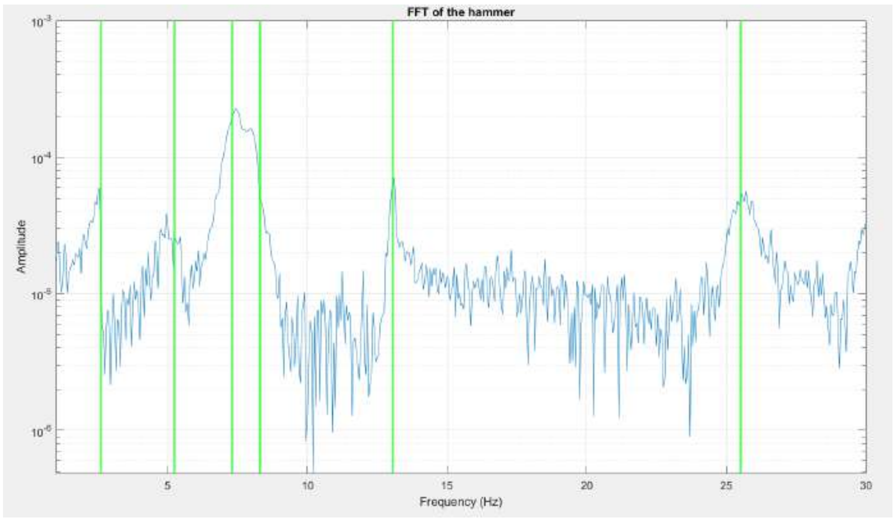


Figure 10.24: Forced vibration test - First hit analysis

It is important to note that several peaks can be observed this time, corresponding to both the flexural modes along Y-direction, the flexural modes along X-direction and the

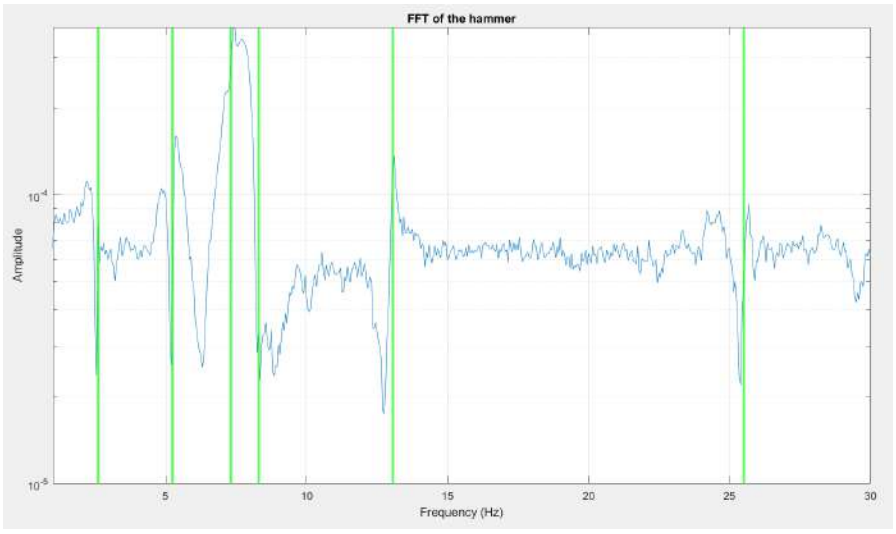


Figure 10.25: Forced vibration test - Second hit analysis

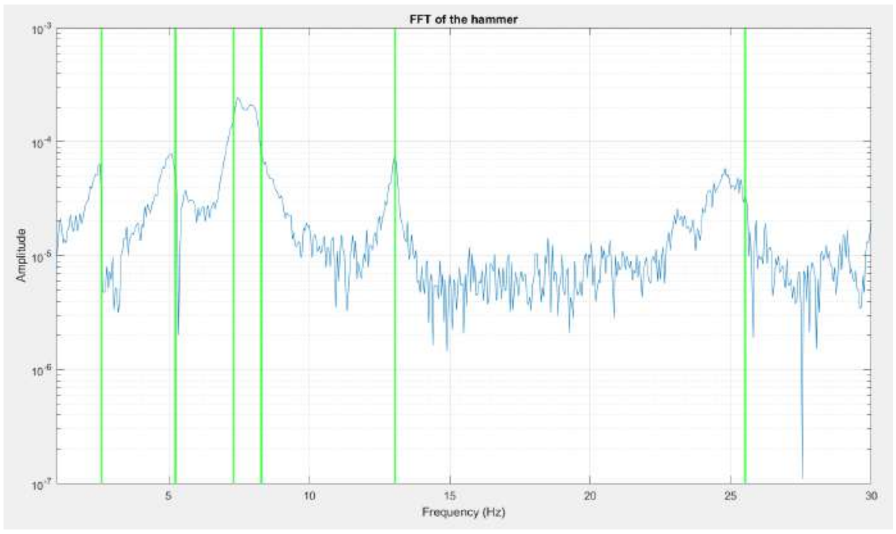


Figure 10.26: Forced vibration test - Third hit analysis

torsional ones. This is because in the direction where the structure is much stiffer, such as the y-axis, the high rigidity limits the displacement specifically along the y-axis. As a result, any slight motion caused by the flexural modes along x and torsional modes is easily detectable in this direction. In contrast, along the more flexible x-axis, the response is dominated by the flexural modes along x, and these large displacements overshadow the smaller contributions from other modes, making them less visible in the x-axis FFT. The increased sensitivity of the stiffer y-axis to even small cross-axis vibrations allows all modes to be captured more clearly.

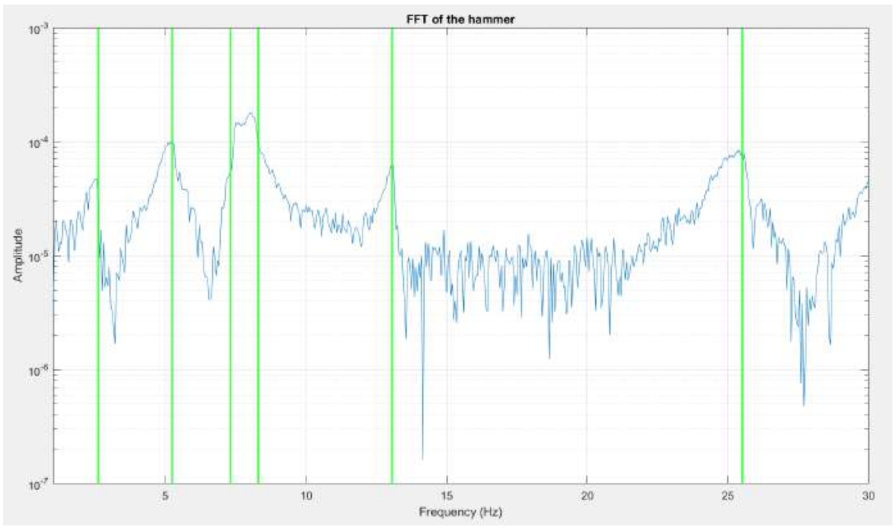


Figure 10.27: Forced vibration test - Fourth hit analysis

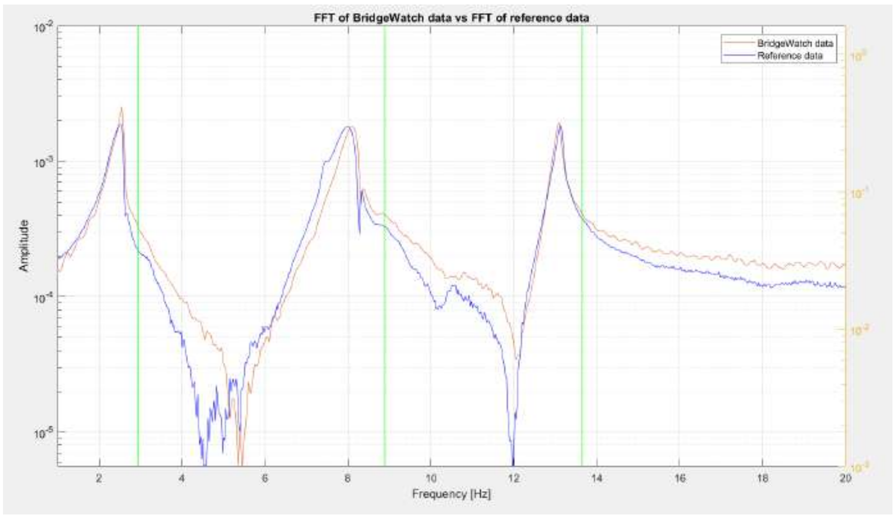


Figure 10.28: Comparison reference accelerometers vs BridgeWatch - X direction

The vertical green lines represent the natural frequencies of the first 6 vibration modes identified by Dragonetti. Our peaks are slightly shifted towards 0 with respect to his.

As a final analysis, an approximate calculation of the signal-to-noise ratio (SNR) of the two acquisition systems was made. SNR is measured as the ratio of the peak height over the root-mean-square deviation (RMSD or standard deviation) value of the noise floor. Noise was extracted in the 14-20 Hz band, while the dominant signal frequency (peak height) corresponds to the third vibration mode. The result were:

- SNR = 26.63 dB, for the Abruzzese's system

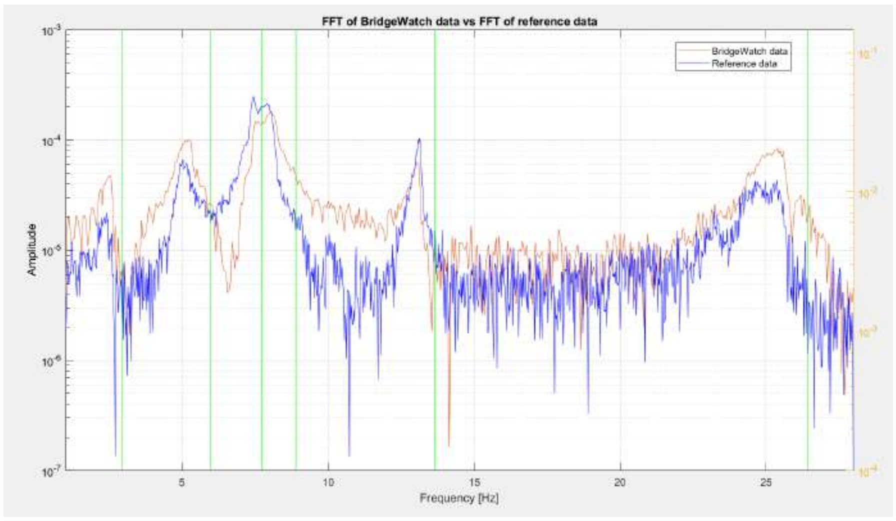


Figure 10.29: Comparison reference accelerometers vs BridgeWatch - Y direction

- SNR = 26.21 dB, for the BridgeWatch system

An SNR of 26.21 dB indicates a relatively good signal quality, meaning that the signal strength is about 20 times stronger than the noise level. In structural monitoring this value is generally acceptable for accurately capturing key vibrational characteristics, such as the primary structural modes. However, while the signal is significantly clearer than the noise, there may still be minor interference or noise that could affect the detection of weaker or higher-order modes.

10.1.5 Climate Cell Test

This test was carried out to see how the accelerometers responded to changes in temperature. The setup consisted of placing two BridgeWatch boards in a climate cell while the other two were left at room temperature. The data acquisition was then started and the temperature inside the cell was raised to different levels. Finally, the values obtained were analysed with a Matlab code and displayed graphically. The Matlab code is reported in Appendix F.

Fig.10.30 shows the climate cell used and the instrument screen showing the graph of temperature variation over time. As can be seen from the figure, the cell was taken from room temperature to 5 different temperatures: 20°C, 5°C, -10°C, 40°C, 60°C.

Fig.10.31 shows the arrangement of the boards during the test: as mentioned above, two were placed inside the cell (Brd2 and Brd4) and the other two were left outside (Brd1 and Brd3).

Results

Fig.10.32, 10.33, 10.34, 10.35 show the test results. The graphs illustrate the relationship between the acceleration values recorded by the accelerometer (on z-axis) and the varying

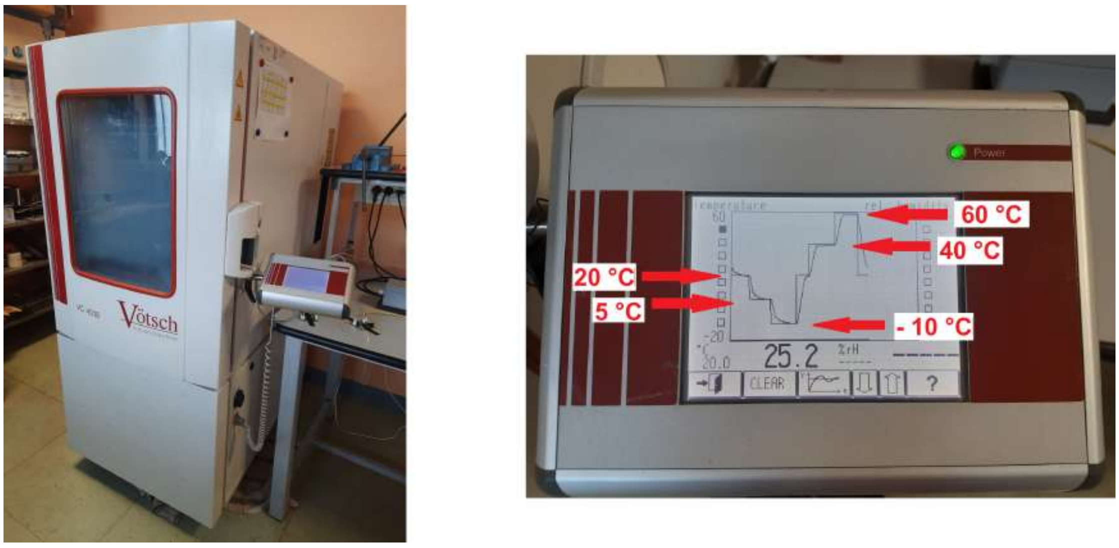


Figure 10.30: Climate cell used for the test

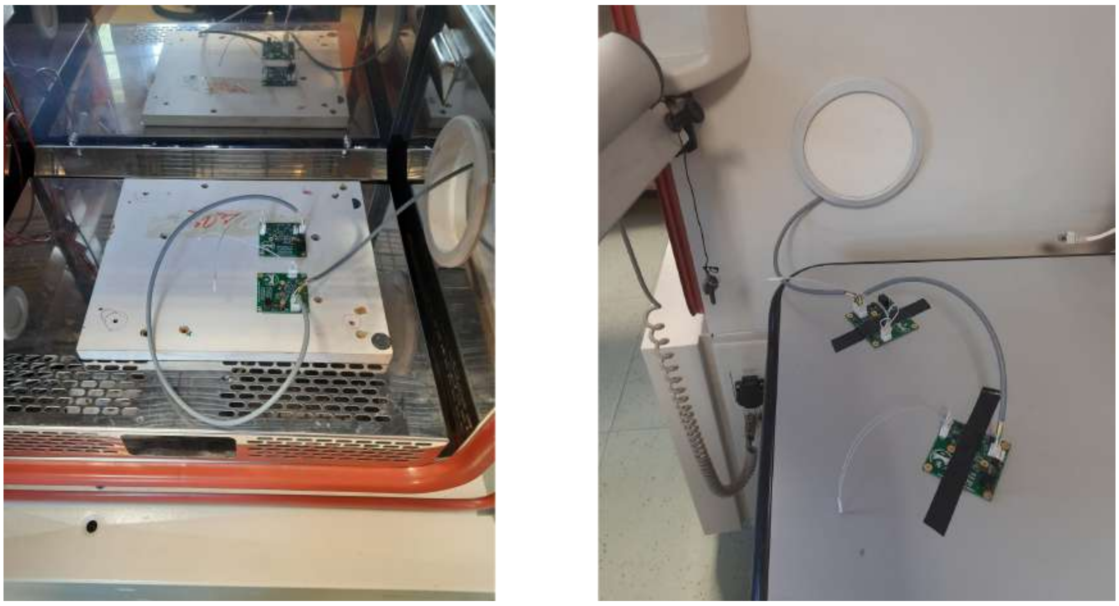


Figure 10.31: Climate cell test setup

temperature. The y-axis represents the measured acceleration in units of g (standard gravity), while the x-axis represents the temperature in degrees Celsius (°C).

- Green dots represent the acceleration values measured at different temperatures after the test began.
- Blue circles mark the accelerations value at room temperature, before the thermal

variations affected the readings.

- The red dashed line indicates the average acceleration value measured at room temperature. This line helps compare how the acceleration values change relative to the baseline measurement at room temperature.

The data shows a slight variation in the accelerometer's response as the temperature increases from below 0°C to 60°C. At lower temperatures, the acceleration values tend to be slightly below the average value at room temperature, while as the temperature rises, the values exhibit an increasing trend.

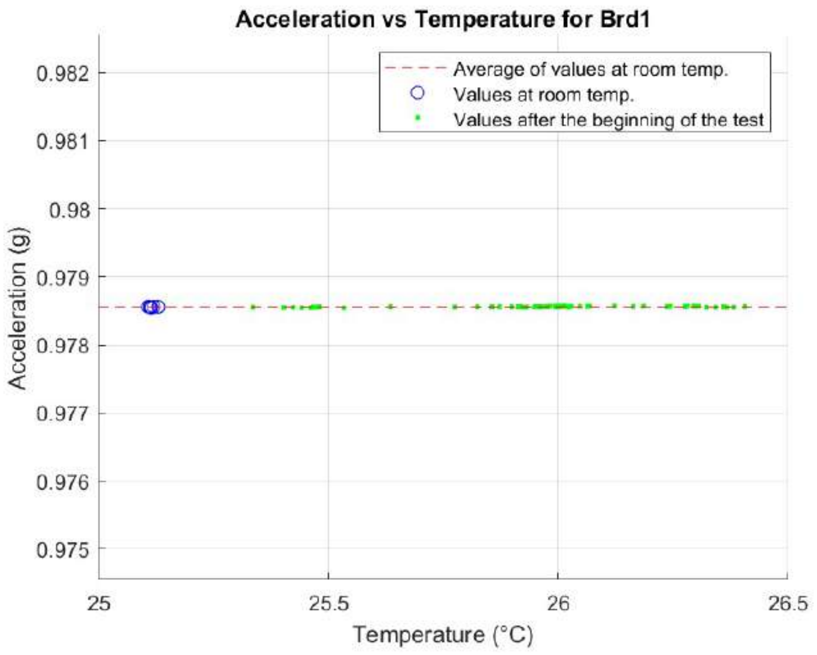


Figure 10.32: Climate cell test - Brd1 results

10.2 Test of the EnvironMonitor board

10.2.1 Cables

The EnvironMonitor board needs a cable for the power supply and a cable for the data transmission. Fig.10.36 shows the custom cable used for RS232 transmission. It is the union of the RS232 to USB converter cable and a 3-wire cable with the connector for the board on one end and the RS232 connector on the other. To build the correct RS232 configuration, the white wire has been used to connect the RX pin of the RS232 connector and the TX of the board, and the blue wire vice versa. This implements the full-duplex communication.

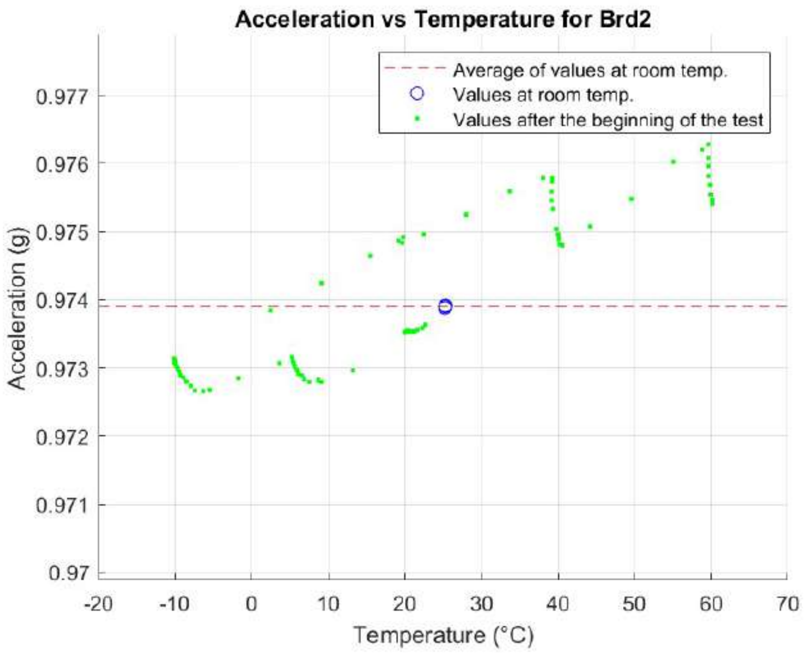


Figure 10.33: Climate cell test - Brd2 results

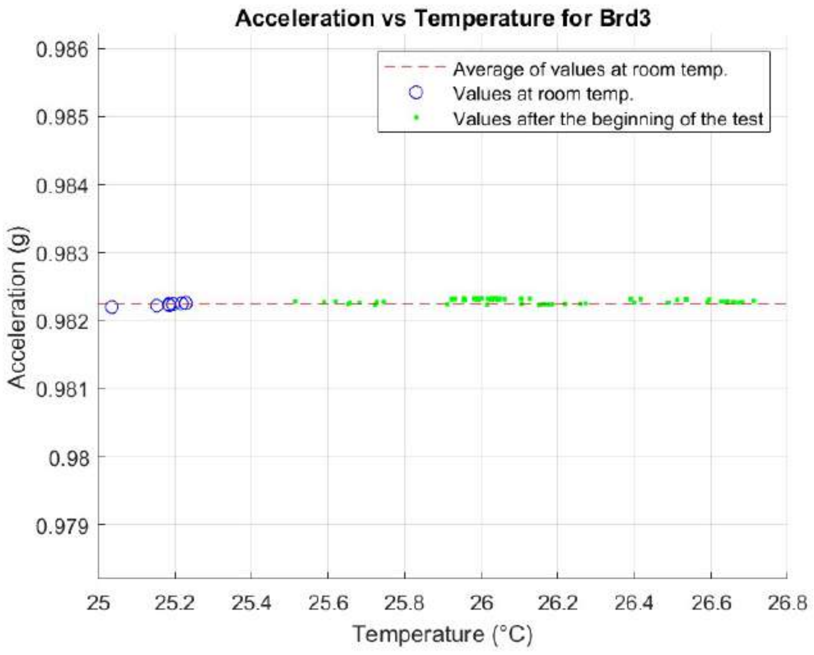


Figure 10.34: Climate cell test - Brd3 results

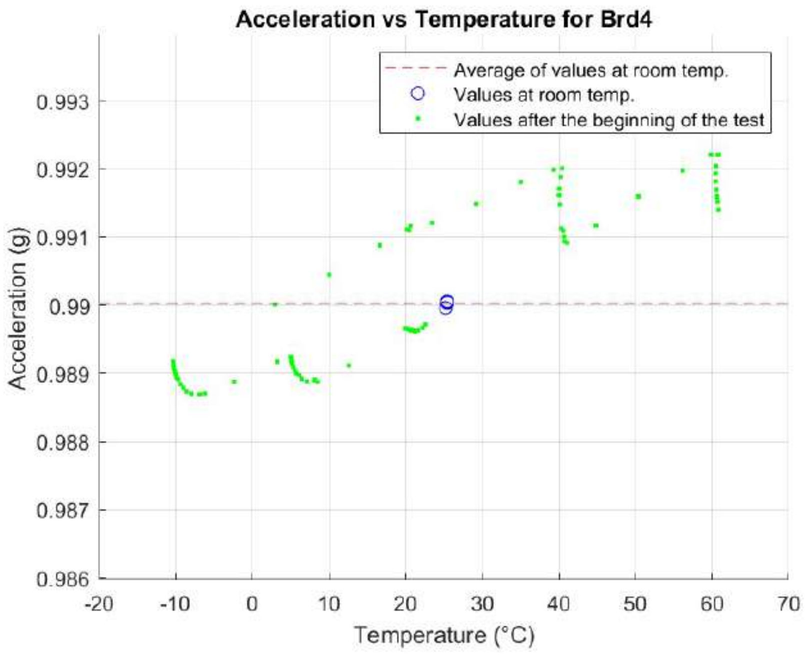


Figure 10.35: Climate cell test - Brd4 results

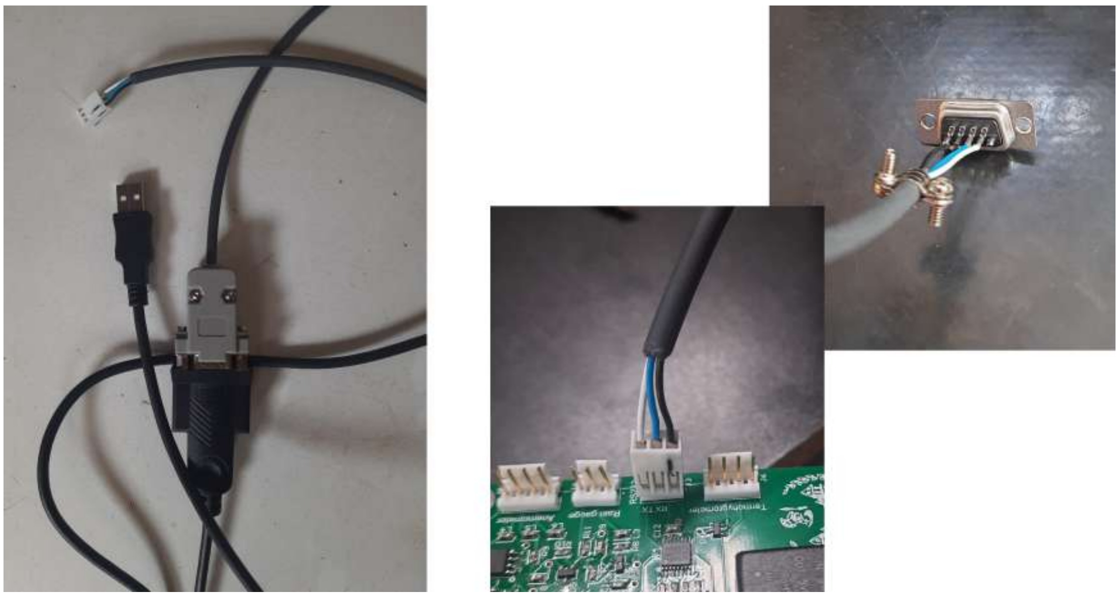


Figure 10.36: EV communication cable

We also need to create the correct connectors for the sensor inputs. The following explains how the one for the thermo-hygrometer was created, but the same procedure was followed for all the other sensors. With reference to Fig.10.37:

1. The sensor as it was when purchased
2. The cable is crimped to crimp contacts using a crimping tool.
3. The crimp contacts are inserted into the appropriate receptacles. The sensor is now ready for connection to the board.
4. The sensor is connected to the PCB.

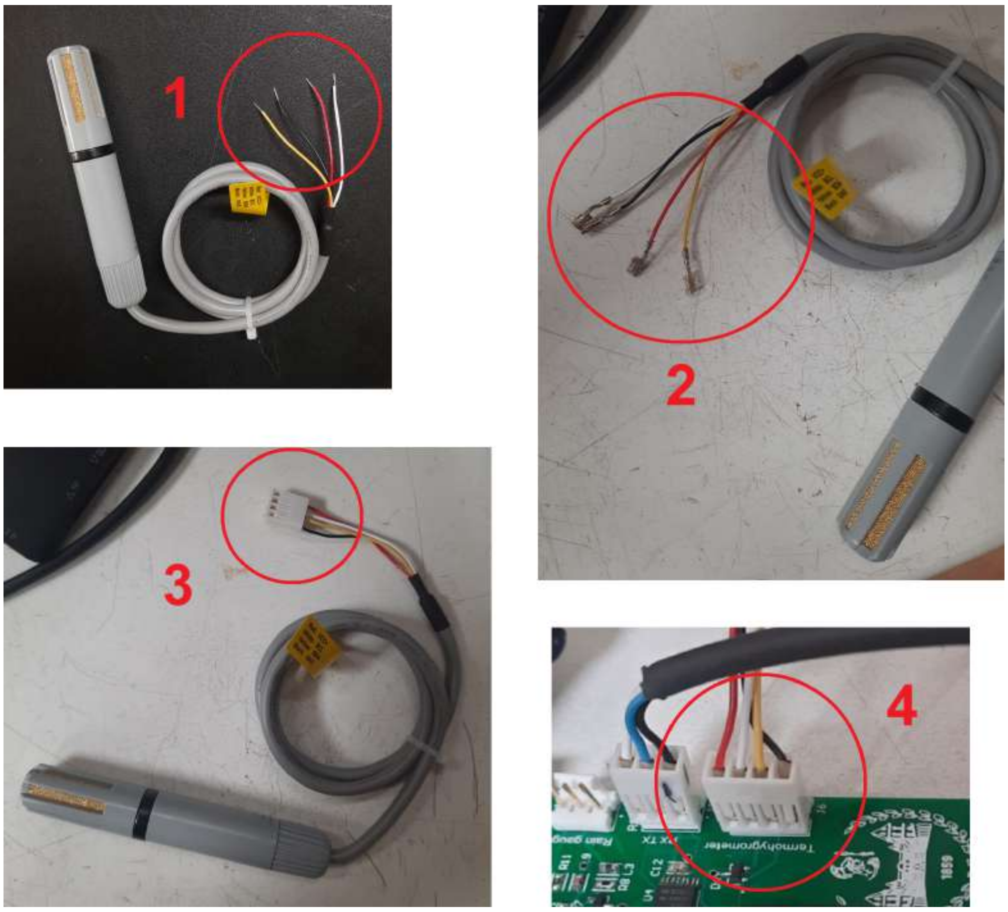


Figure 10.37: Thermo-hygrometer connector

10.2.2 Test

The test involved connecting all the planned sensors to the inputs, and checking that the board was correctly communicating environmental information to the Raspberry. As the rain gauge and water level sensor were not purchased, we simulated their outputs in this way:

- The output of the water level sensor is a current between 4 and 20 mA. We therefore connected a current generator to the input connector that would produce a current within this range.
- The output of the rain gauge is a switch that closes to ground every time the spoon tips (every 0.2 mm of rain). We then connected to the input connector a signal generator programmed to generate a square wave between 0V and 3.3V, which would simulate the switch closing every time it went to ground.

Fig.10.38 shows the EnvironMonitor board with all the environmental sensor connected. Fig.10.39 shows the wave generator to simulate the rain gauge and the current generator to simulate the water-level sensor.

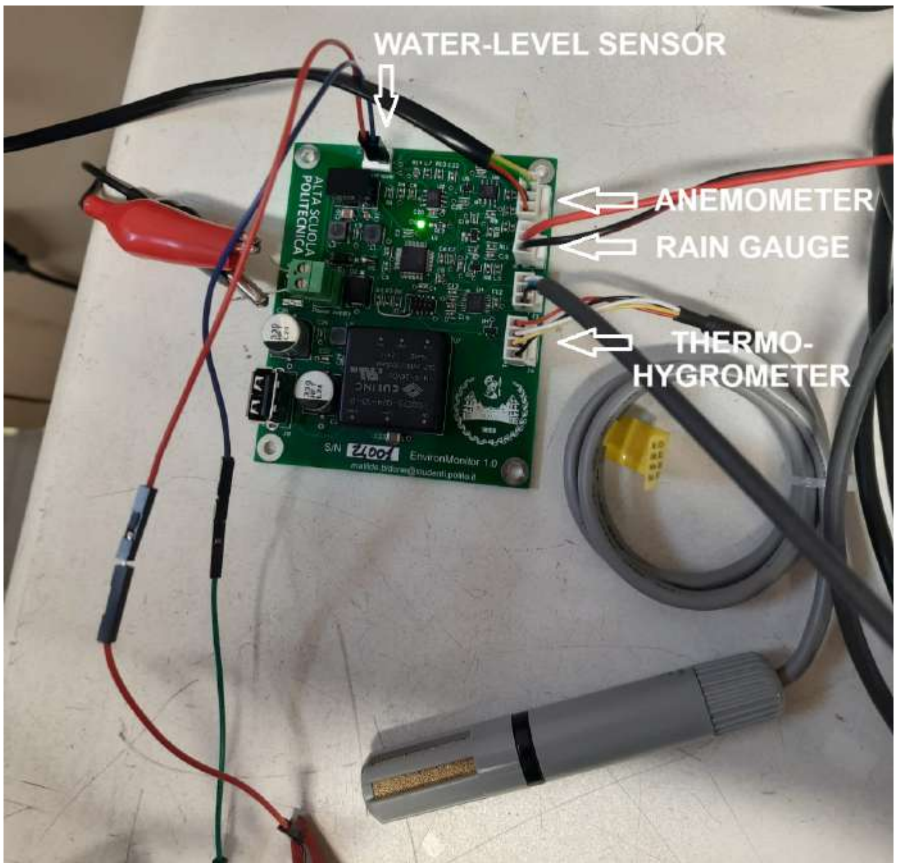
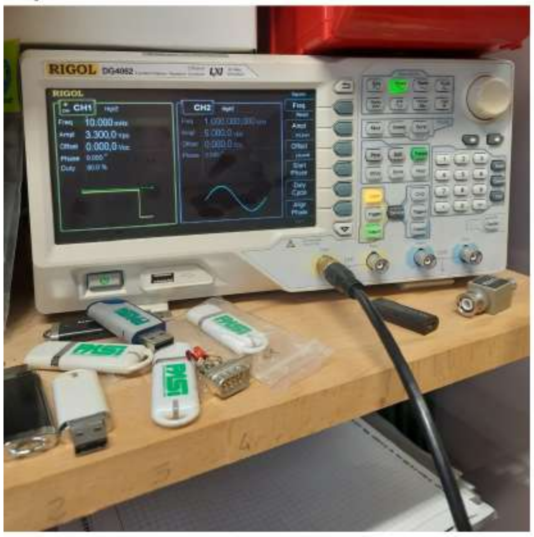


Figure 10.38: All sensors connected to EnvironMonitor board

10.2.3 Results

Two different situations were created, and for each one the programme was asked to print out the data coming from the board so that it could check that it matched the state of the sensors.

a)


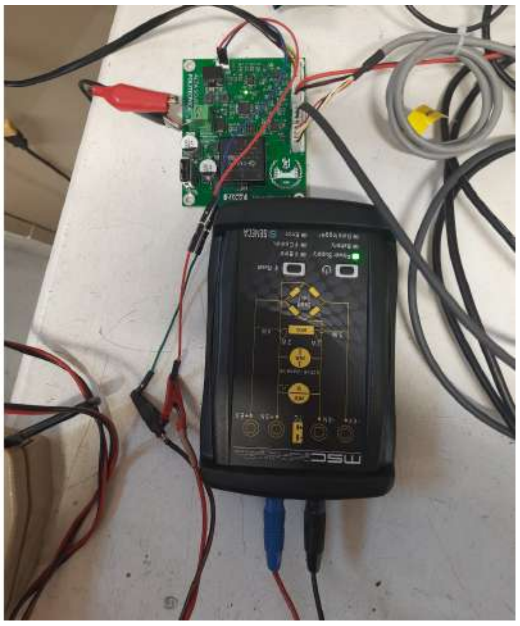
b)


Figure 10.39: **a)** The wave generator. **b)** The current generator.

First case

The setup for the first test was the following:

- The wave generator for the rain gauge was set to a frequency of 0.01 Hz. This means that it simulates a tipping spoon that tips one time each 100 seconds, and so a total of: 0.2 mm * 36 = 7.2 mm/h.
- The current generator for the water-level sensor was set to produce a current of 12mA. Since the range of the sensor is from 4mA to 20mA (corresponding to a distance of 0m to 30m), a current of 12mA corresponds to a distance of 15m.
- The wind speed anemometer has not been set in motion, while the vane indicating the direction has been placed approximately in the same direction as the mast (which corresponds to 0 degrees), so it should indicate a few degrees.
- The thermo-hygrometer was left as it was on the table, so it should record the real temperature and humidity in the room.

Fig.10.40 shows the setup, while Fig.10.41 shows the screenshot of the program with the environmental data. As you can see, all the data is consistent with the expected results.

Second case

The setup for the second test was the following:

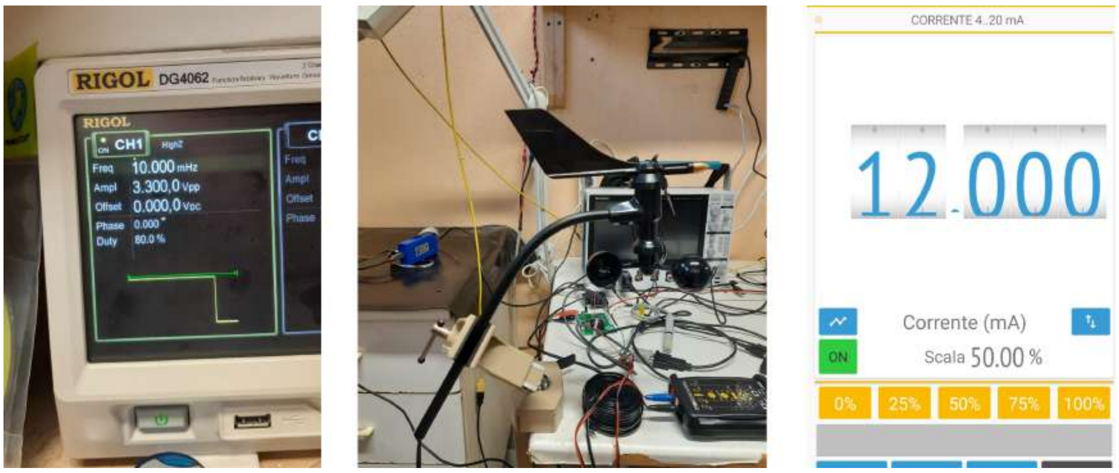
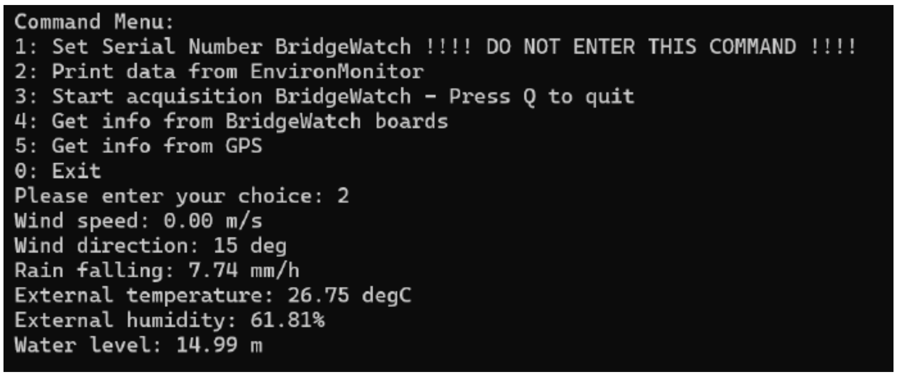


Figure 10.40: First test setup



```
Command Menu:
1: Set Serial Number BridgeWatch !!!! DO NOT ENTER THIS COMMAND !!!!
2: Print data from EnvironMonitor
3: Start acquisition BridgeWatch - Press Q to quit
4: Get info from BridgeWatch boards
5: Get info from GPS
0: Exit
Please enter your choice: 2
Wind speed: 0.00 m/s
Wind direction: 15 deg
Rain falling: 7.74 mm/h
External temperature: 26.75 degC
External humidity: 61.81%
Water level: 14.99 m
```

Figure 10.41: First test result

- The wave generator for the rain gauge was set to a frequency of 1 Hz. This means that it simulates a tipping spoon that tips one time each second, and so a total of: 0.2 mm * 3600 = 720 mm/h.
- The current generator for the water-level sensor was set to produce a current of 20mA, which corresponds to a distance of 30m.
- The wind speed anemometer has not been set in motion, while the vane indicating the direction has been placed approximately in the opposite direction to the mast (which corresponds to 0 degrees), so it should indicate close to 180 degrees.
- At the time I asked for the data to be sent, I was breathing on the thermo-hygrometer, so you can expect the recorded humidity to be higher than in the previous case.

Fig.10.42 shows the setup, while Fig.10.43 shows the screenshot of the program with the environmental data. As you can see, all the data is consistent with the expected results.

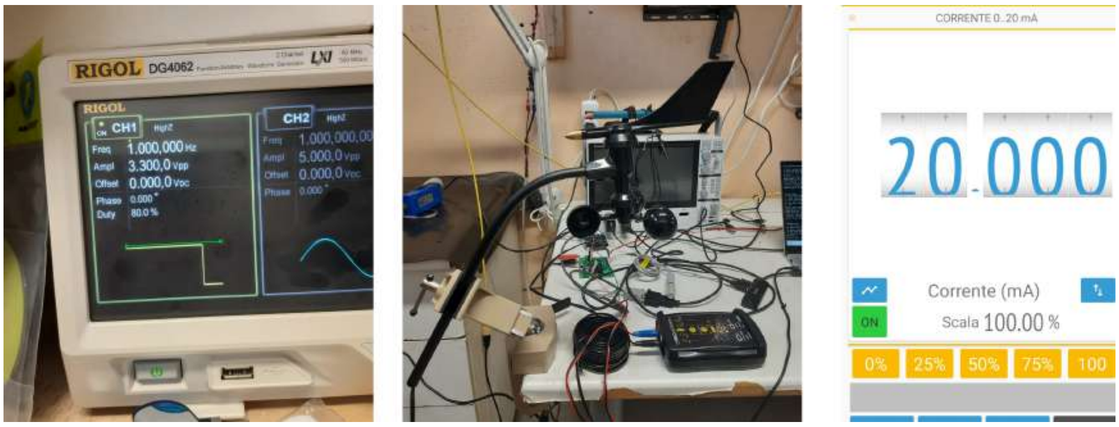
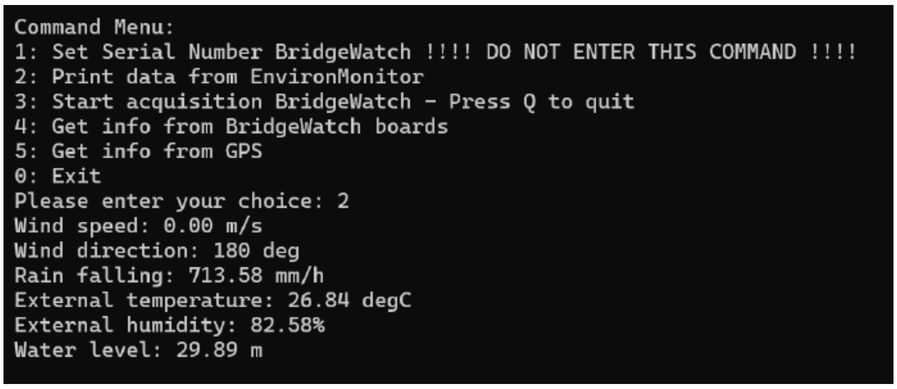


Figure 10.42: Second test setup



```
Command Menu:
1: Set Serial Number BridgeWatch !!!! DO NOT ENTER THIS COMMAND !!!!
2: Print data from EnvironMonitor
3: Start acquisition BridgeWatch - Press Q to quit
4: Get info from BridgeWatch boards
5: Get info from GPS
0: Exit
Please enter your choice: 2
Wind speed: 0.00 m/s
Wind direction: 180 deg
Rain falling: 713.58 mm/h
External temperature: 26.84 degC
External humidity: 82.58%
Water level: 29.89 m
```

Figure 10.43: Second test result

Chapter 11

Conclusions

Given the requirements set out in the Chap.3, a prototype was built to meet them all. In particular:

1. **Sensor Types and Specifications** All the sensors listed were used in the system, and for each one a model was selected that met all the characteristics described in its requirements table.

2. **Data Acquisition and Sampling**

   - **Appropriate Sampling Rate:** As explained in the Sec.3.2.1, the choice of a sampling frequency of 256 Hz is suitable for capturing the typical frequencies of concrete bridges without loss of information.
   - **Real-Time Monitoring:** The system is designed to allow real-time monitoring. In fact, data is sent in real time (about 50 times per second) to the Raspberry, which sends it directly to a server where it can be downloaded remotely for analysis.
   - **Data Synchronization:** The synchronisation of the boards is ensured by the fact that every minute the Raspberry sends the resynchronisation command. As explained in the Chap.8, this synchronises the generation of the square wave used for sampling in each Bridgewatch board. You can also correlate BridgeWatch and EnvironMonitor data, as each line of the output files starts with the time in seconds to which the data in that line refers.
   - **Data Integrity:** As explained in the Chap.7, data integrity is ensured through the use of software protocols during data transmission that implement error checking and retransmission in case of error.

3. **Environmental Requirements**

   - **Weather Resistance:** The system is designed for outdoor use. All sensors have been selected with a temperature range of -40 to +80 degrees, the cable is guaranteed for outdoor use (shielded and UV resistant) and, as shown in the Chap.5, each board has its own IP66 protective case.

4. **Power**

    - **Low Power Consumption:** All the main chips have also been chosen with power consumption in mind. The total power consumption of the system was analysed in Sec.5.3. With the chosen battery, the system can run for 3 days without recharging from the solar panels.
    - **Solar Power Capability:** The system is designed to be powered by a battery recharged by solar panels.

5. **Installation and Accessibility**

    - **Non-Intrusive Installation:** The system can be applied to the bridge in a non-intrusive way. In fact, as explained in Chap.5, the Bridgewatch cards are designed to be attached to a metal plate that can be glued to the bridge without any major work.
    - **Remote Access:** The system is designed to allow remote access to the data. In fact, as shown in Fig.3.1, the Raspberry sends the data via a SIM to a server, where it can then be downloaded from your computer.
    - **Modularity:** The system has been designed in a modular way. In fact, it is easily adaptable to different bridges, as all you need to do is change the number of BridgeWatch boards to use. The system is already set up to have different numbers of cards connected to the bus (as explained in the Chap.8, all you need to do is edit the configuration file).

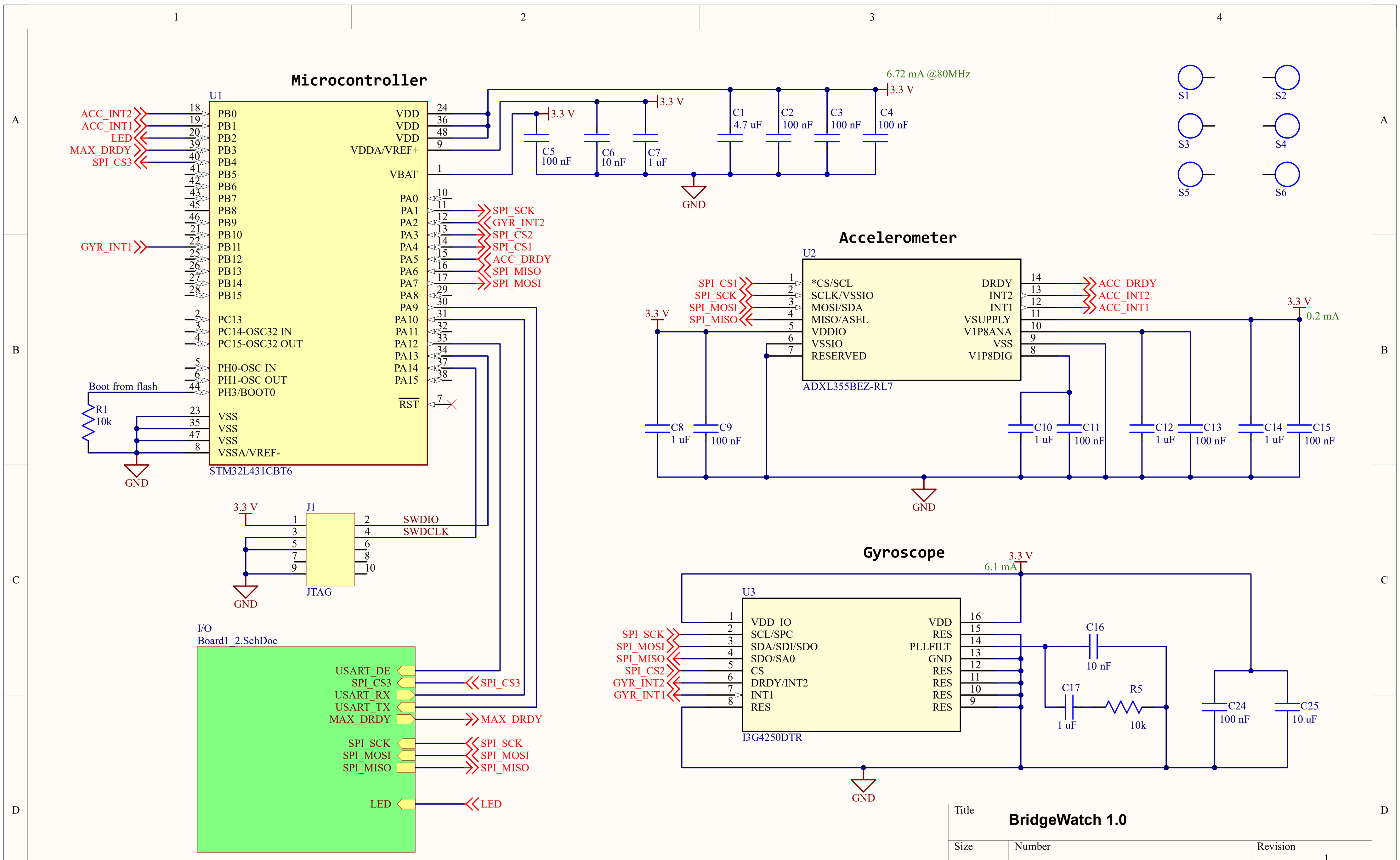
6. **Cost Considerations**

    - **Cost-Effective Solutions:** An in-depth analysis of construction costs is provided in Chap.9. Compared to similar products on the market, this system has significantly lower costs.

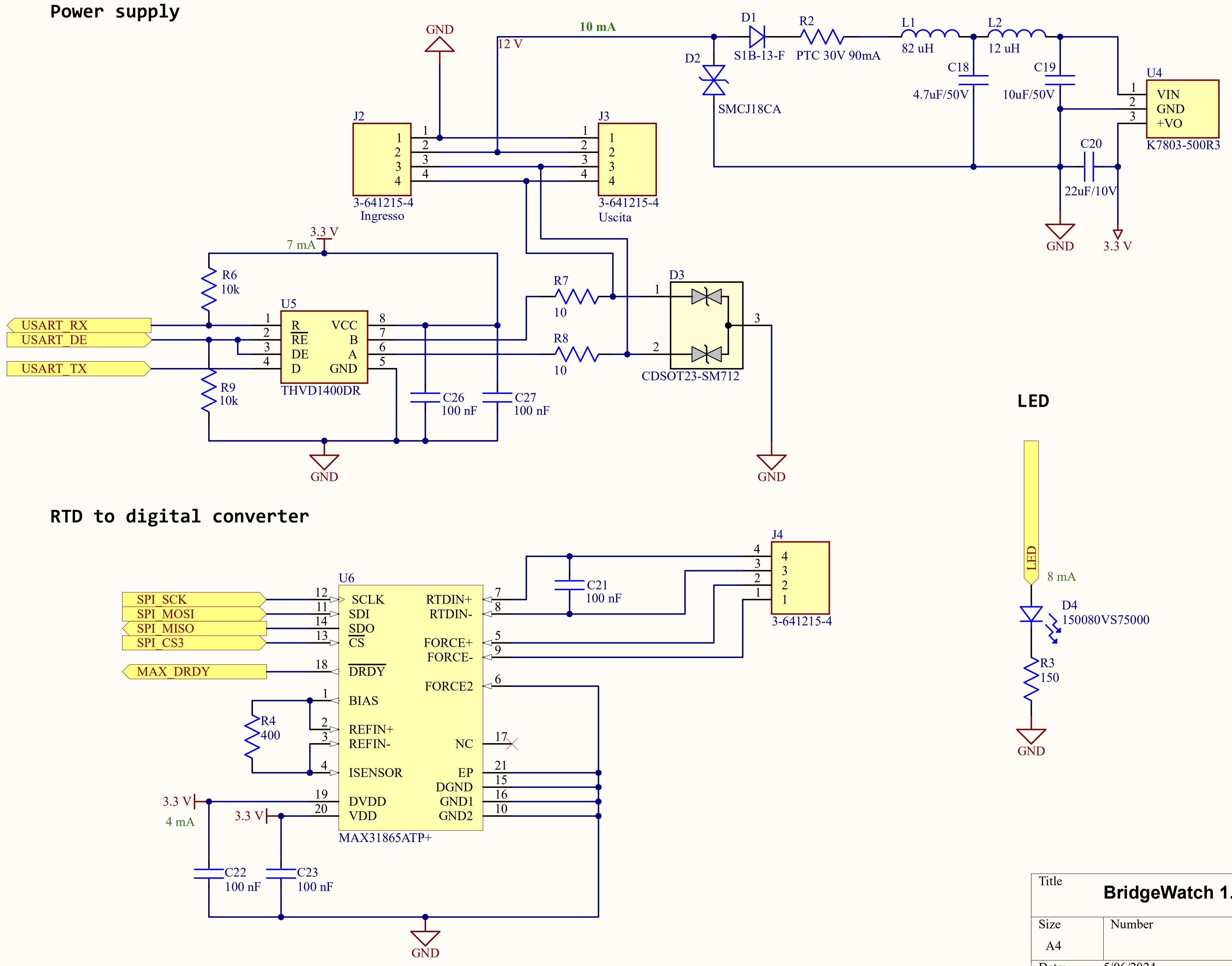
In Chap.10, all of the numerous tests that were carried out to verify the actual functioning of the system were presented. It can be concluded that the system allows the correct extraction of natural frequencies and modal shapes and therefore fully meets the purposes for which it was designed.

Appendix A

BridgeWatch schematics



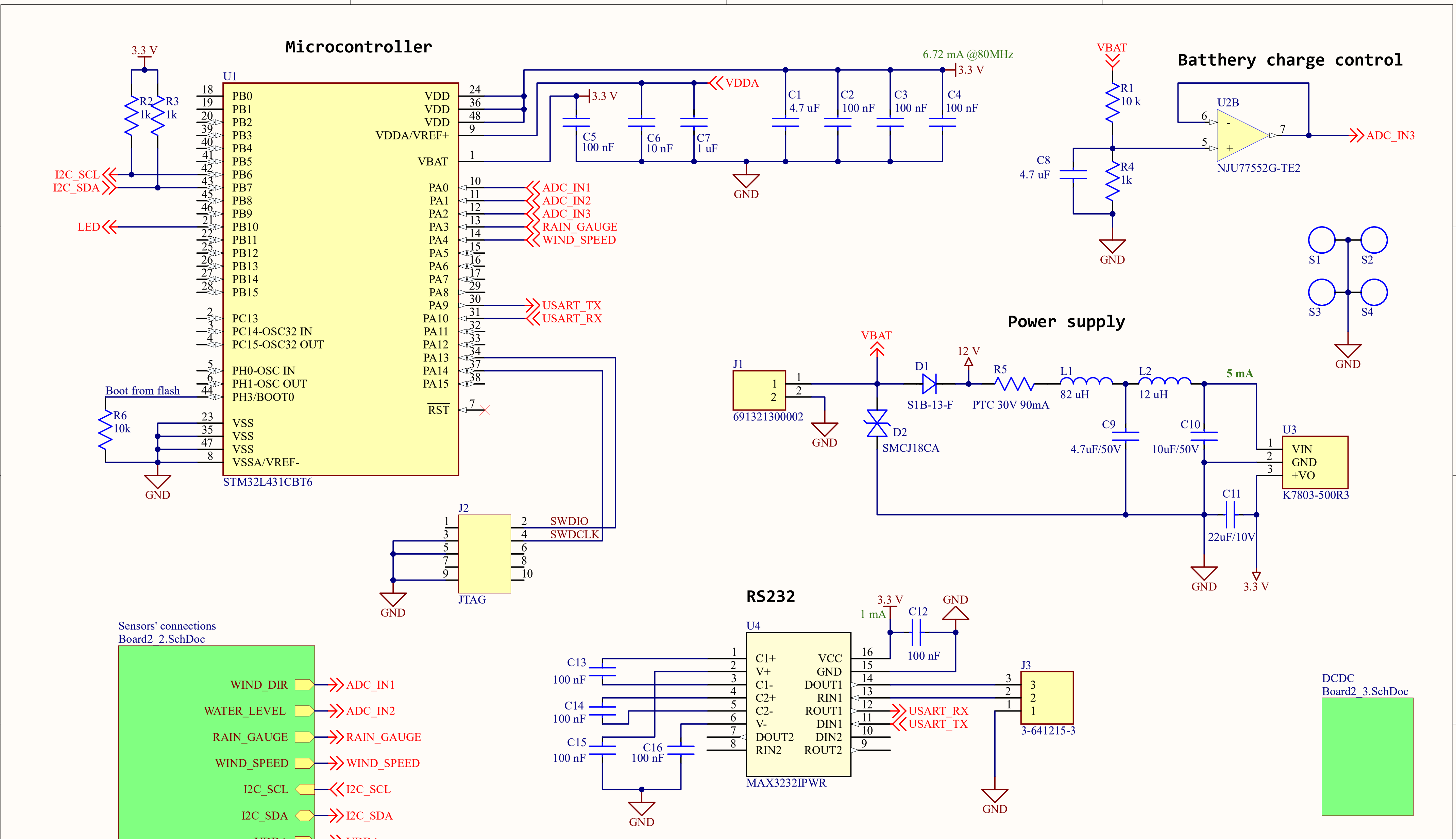
Title	BridgeWatch 1.0	
Size	Number	Revision
A4		1
Date:	5/06/2024	Sheet of 1 of 2
File:	C:\Users\..\Board1.SchDoc	Drawn By: Matilde Bidone



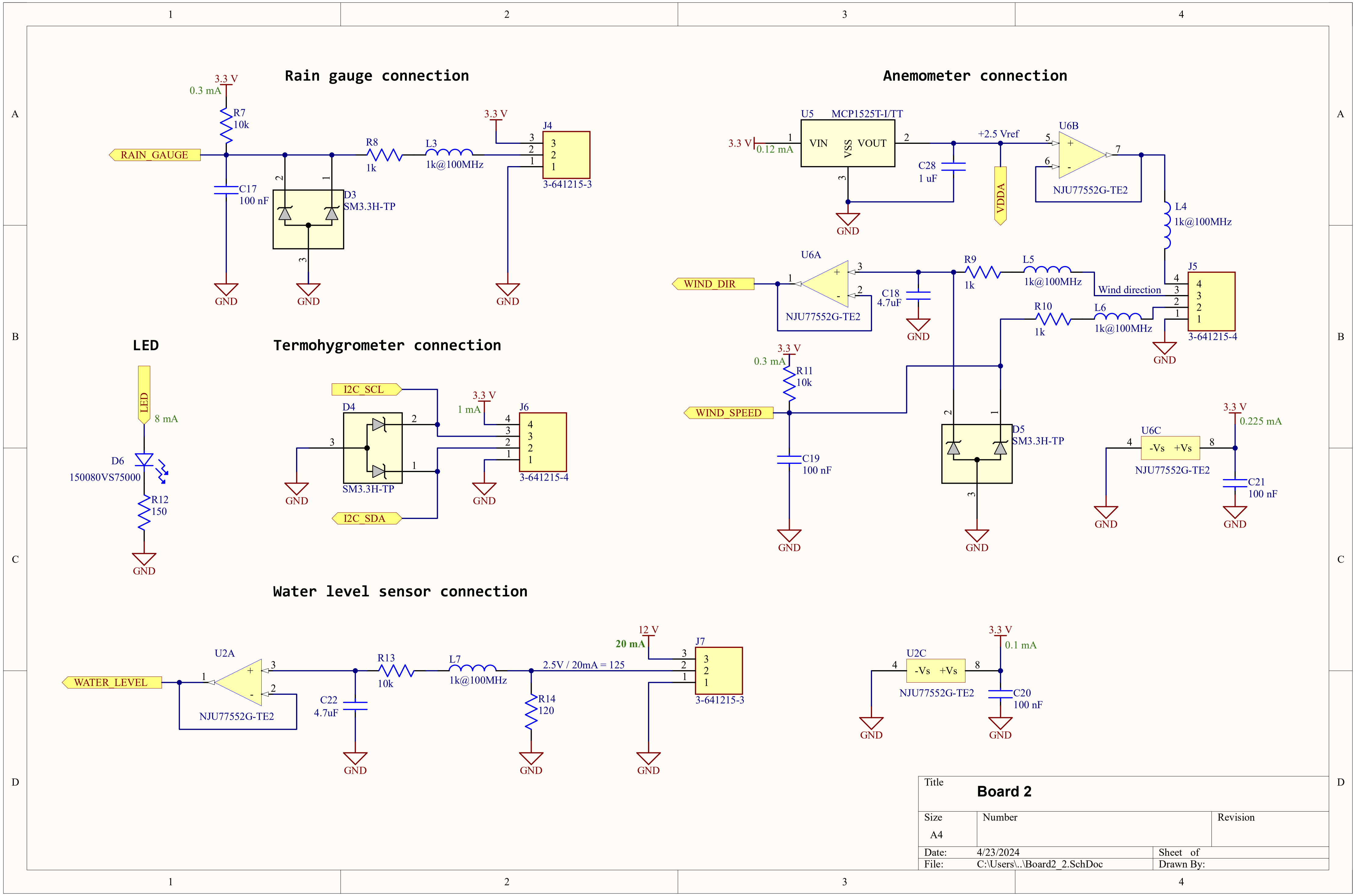
Title	BridgeWatch 1.0	
Size	Number	Revision
A4		
Date:	5/06/2024	Sheet of 2 of 2
File:	C:\Users\..\Board1_2.SchDoc	Drawn By: Matilde Bidone

Appendix B

EnvironMonitor schematics



Title	**Board 2**	
Size A4	Number	Revision
Date:	4/23/2024	Sheet of
File:	C:\Users\..\Board2.SchDoc	Drawn By:



Title Board 2

Size A4

Number

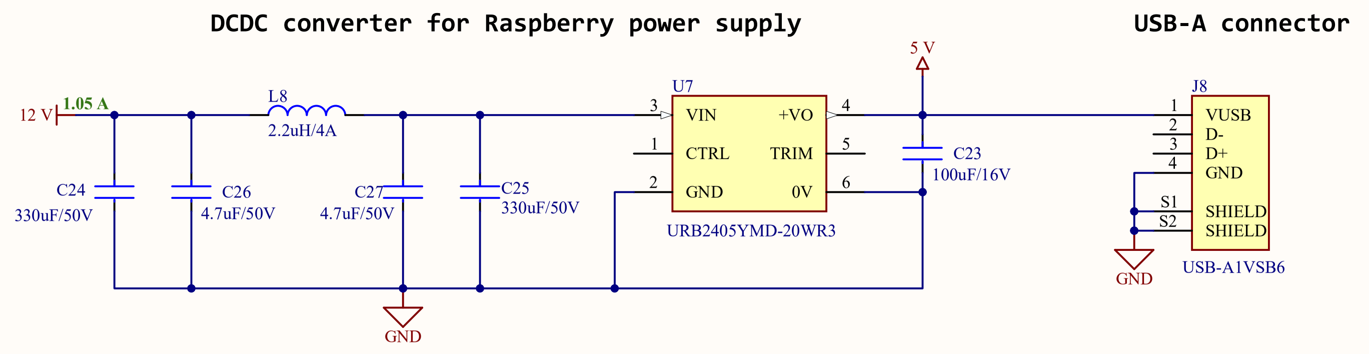
Revision

Date: 4/23/2024

Sheet of

File: C:\Users\..\Board2_2.SchDoc

Drawn By:



Title		
Size A4	Number	Revision
Date: 4/23/2024		Sheet of
File: C:\Users\..\Board2_3.SchDoc		Drawn By:

Appendix C

SPI protocol

SPI communication is configured using two parameters: clock polarity (CPOL) and clock phase (CPHA). These parameters determine the timing relationship between the clock signal and the data lines, resulting in the four possible modes of operation (Mode 0 to Mode 3) reported in Fig. C.1 CPOL determines the idle state of the clock signal, while CPHA specifies whether data is sampled on the leading or trailing edge of the clock pulse.

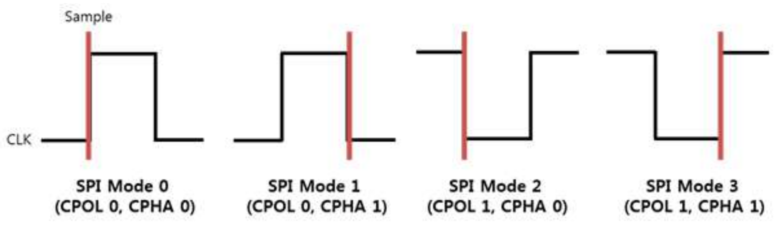


Figure C.1: SPI modes. Source: *SPI* n.d.

In SPI communication, both the master and slave devices use shift registers to send and receive data simultaneously. When a data transfer occurs, the master sends data to the slave through the MOSI (Master Out Slave In) line, while the slave simultaneously sends data back to the master through the MISO (Master In Slave Out) line. Both the master and slave shift registers are connected in such a way that, with each clock pulse provided by the master, the data bits are shifted out of the master's register and into the slave's register, while simultaneously, the slave's data is shifted into the master's register. The data bits are shifted in MSB first. Fig. C.2 shows a simplified SPI architecture with only one master and one slave. This full-duplex nature of SPI allows for a continuous exchange of data between the master and slave in every clock cycle, making it an efficient and fast method for bidirectional communication.

A typical operation performed using SPI is reading from or writing to registers within a peripheral device (slave). This process is generally carried out through a series of well-defined steps, orchestrated by the master device. Here's how this communication typically works:

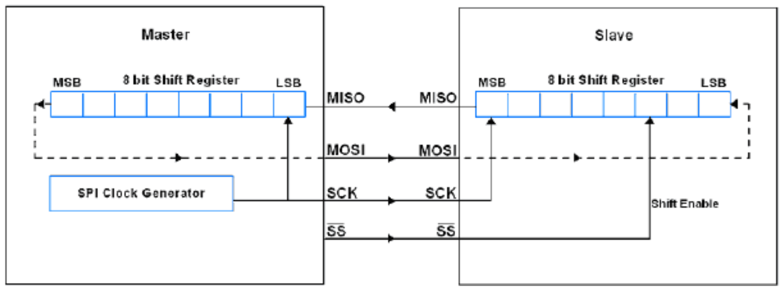


Figure C.2: SPI architecture. Source: *SPI in AVR ATmega16/ATmega32* n.d.

1. Slave Selection: The master device begins by pulling the Chip Select (CS or SS) line low, which selects the specific slave device it wants to communicate with.

2. Sending the Command: The master then sends a command byte over the MOSI (Master Out Slave In) line. This command typically includes information about whether the master intends to read from or write to a specific register within the slave. For example, the most significant bit (MSB) of the command might indicate a read (0) or write (1) operation, while the remaining bits specify the register address.

3. Writing Data (for a write operation): If the operation is a write, the master immediately follows the command byte with the data byte (or bytes) that it wants to write to the specified register. The data is clocked into the slave device on each rising or falling edge of the clock signal (SCLK), depending on the SPI mode being used.

4. Reading Data (for a read operation): If the operation is a read, after sending the command byte, the master continues to clock the SCLK. The slave, in response, sends the contents of the specified register back to the master over the MISO (Master In Slave Out) line. The master receives this data, typically on the opposite clock edge from when it was sent.

5. Terminating the Communication: Once the read or write operation is complete, the master pulls the CS line high again. This de-selects the slave device and ends the current SPI transaction. The SPI bus is then free for further communications, either with the same or a different slave device.

One of the key advantages of SPI is its high data transfer rate, which can be significantly faster than other communication protocols like I2C. This is due to its simple, straightforward design and lack of addressing overhead. However, SPI also has some limitations, including the need for more pins (one for each SS line) and the lack of standardized error-checking mechanisms.

Appendix D

I2C protocol

Fig. D.1 shows a detail of the start and stop conditions, while Fig. D.1 shows an example of a transmission.

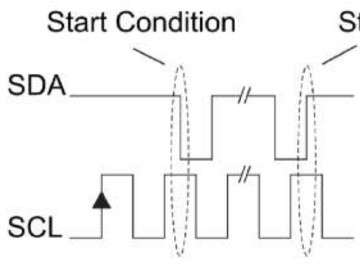


Figure D.1: I2C start and stop conditions. source: *I2C Primer: What is I2C? (Part 1)* n.d.

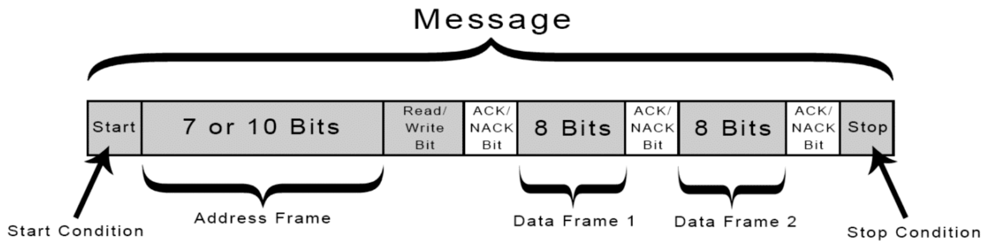


Figure D.2: I2C transmission. Source: *Basics of the I2C Communication Protocol* n.d.

One of the unique features of I2C is its ability to support multiple masters on the same bus, although only one master can control the bus at any given time. Arbitration is used to manage multiple masters. If two masters start communication simultaneously, arbitration occurs on a bit-by-bit basis, with the master that first detects a mismatch between the intended and actual SDA line value withdrawing from the communication. This ensures that only one master controls the bus at a time, preventing data corruption.

Clock stretching is another important feature where the slave can hold the SCL line low to pause communication if it needs more time to process data. The master waits until the slave releases the SCL line before continuing the data transfer, allowing for reliable communication even when the slave operates slower than the master.

The speed of I2C can vary depending on the mode being used. Standard mode supports up to 100 kbps, Fast mode up to 400 kbps. I2C is highly versatile due to its simplicity and efficiency in managing communication with multiple devices. However, it does have some limitations, such as the relatively lower data transfer rate compared to protocols like SPI and the fact that it can be sensitive to noise, especially on longer lines. Despite these limitations, I2C remains a popular choice for inter-device communication in embedded systems, particularly where simplicity and scalability are essential.

Appendix E

Matlab Code for the Shaker Test

```
data = load('accY10.txt');  % Load data from the file

% Signal parameters
fs = 256;                       % Sampling frequency (Hz)
N = length(data);                 % Number of samples
t = (0:N-1)/fs;                   % Time vector for the input signal

% Define the time range (from 10 to 10.2 seconds)
time_start = 10;
time_end = 10.2;

% Find the indices corresponding to the time range
idx_range = find(t >= time_start & t <= time_end);

% Plot the input signal in the specified time range
figure;                           % Create a new figure for the input
    signal
plot(t(idx_range), data(idx_range), 'b', 'LineWidth', 1.5);
title('Input␣Signal');
xlabel('Time␣(s)');
ylabel('Amplitude');
grid on;

% Create the Hanning window and apply it to the signal
w = hann(N);                      % Create the Hanning window
windowed_data = data .* w;         % Apply the window to the signal

% Compute the FFT of the windowed signal
X = fft(windowed_data);            % FFT of the windowed signal
X_mag = abs(X)/N;                  % Normalize the magnitude
f_axis = (0:N-1)*(fs/N);           % Frequency axis
```

```

% Take only the first half of the spectrum (up to fs/2)
X_mag_half = X_mag(1:N/2);      % First half of the spectrum
f_axis_half = f_axis(1:N/2);    % Corresponding frequencies

% Find the maximum amplitude and its position
[max_amplitude, index_max] = max(X_mag_half);

% Find the corresponding frequency
peak_frequency = f_axis_half(index_max);

% Display the peak frequency
fprintf('The␣peak␣frequency␣is␣%.5f␣Hz\n', peak_frequency);

% Plot the FFT spectrum
figure;                               % Create a new figure for the FFT spectrum
plot(f_axis_half, X_mag_half, 'b', 'LineWidth', 1); % Plot the magnitude spectrum
title('FFT␣Spectrum␣with␣Hanning␣Window');
xlabel('Frequency␣(Hz)');
ylabel('Amplitude');
grid on;

% Annotate the peak value next to the peak
text(peak_frequency + 5, max_amplitude, ...     % Shift the text 5 units to the right
    sprintf('%.5f␣Hz', peak_frequency), ...     % Display the peak frequency value
    'VerticalAlignment', 'middle', ...          % Align the text at the middle of the peak
    'HorizontalAlignment', 'left', ...          % Align the text to the left of the specified position
    'FontSize', 10, 'FontWeight', 'bold', 'Color', 'r');  % Customize text appearance
```

Appendix F

Matlab Code for the Climatic Cell Test

```matlab
% Folder path containing the files
folderPath = 'C:\Users\matil\Politecnico␣Di␣Torino␣Studenti␣
    Dropbox\Matilde␣Bidone\TESI\SOFTWARE\cella_climatica';

% Get the list of all text files in the folder
fileList = dir(fullfile(folderPath, '*.txt'));

% Preallocate vectors to store the means
numFiles = length(fileList);
mean_Brd1_Temp = zeros(numFiles, 1);
mean_Brd1_AccZ = zeros(numFiles, 1);
mean_Brd2_Temp = zeros(numFiles, 1);
mean_Brd2_AccZ = zeros(numFiles, 1);
mean_Brd3_Temp = zeros(numFiles, 1);
mean_Brd3_AccZ = zeros(numFiles, 1);
mean_Brd4_Temp = zeros(numFiles, 1);
mean_Brd4_AccZ = zeros(numFiles, 1);

% Process each file
for k = 1:numFiles
    % Current file name
    fileName = fullfile(folderPath, fileList(k).name);

    % Set import options
    opts = detectImportOptions(fileName, 'Delimiter', ';');
    opts.DataLine = 3;  % Start reading data from the third line
    opts.VariableNamesLine = 1;  % The second line contains
        variable names

    % Read data into a table
    data = readtable(fileName, opts);

    % Extract relevant columns
    Brd1_Temp = data.Brd1_Temp;       % Extract Brd1_Temp column
    Brd1_AccZ = data.Brd1_AccZ;       % Extract Brd1_AccZ column
    Brd2_Temp = data.Brd2_Temp;       % Extract Brd2_Temp column
    Brd2_AccZ = data.Brd2_AccZ;       % Extract Brd2_AccZ column
    Brd3_Temp = data.Brd3_Temp;       % Extract Brd3_Temp column
    Brd3_AccZ = data.Brd3_AccZ;       % Extract Brd3_AccZ column
    Brd4_Temp = data.Brd4_Temp;       % Extract Brd4_Temp column
    Brd4_AccZ = data.Brd4_AccZ;       % Extract Brd4_AccZ column

    % Calculate mean of each vector and store in the corresponding vector
    mean_Brd1_Temp(k) = mean(Brd1_Temp);
    mean_Brd1_AccZ(k) = mean(Brd1_AccZ);
    mean_Brd2_Temp(k) = mean(Brd2_Temp);
    mean_Brd2_AccZ(k) = mean(Brd2_AccZ);
    mean_Brd3_Temp(k) = mean(Brd3_Temp);
    mean_Brd3_AccZ(k) = mean(Brd3_AccZ);
    mean_Brd4_Temp(k) = mean(Brd4_Temp);
    mean_Brd4_AccZ(k) = mean(Brd4_AccZ);
end


% Calculate the mean of the first 8 values for each dataset
mean_first_8_AccZ_Brd1 = mean(mean_Brd1_AccZ(1:8));
mean_first_8_AccZ_Brd2 = mean(mean_Brd2_AccZ(1:8));
mean_first_8_AccZ_Brd3 = mean(mean_Brd3_AccZ(1:8));
mean_first_8_AccZ_Brd4 = mean(mean_Brd4_AccZ(1:8));

% Fixed y-axis range
yRange = 0.008;

% Data to plot for Brd4
x1_Brd4 = mean_Brd4_Temp(1:8);  % Temperatures of the first 8 values
y1_Brd4 = mean_Brd4_AccZ(1:8);  % Accelerations of the first 8 values

x2_Brd4 = mean_Brd4_Temp(9:end);  % Temperatures from the remaining values
y2_Brd4 = mean_Brd4_AccZ(9:end);  % Accelerations from the remaining values

% Create plot for Brd4
figure;
hold on;
yline(mean_first_8_AccZ_Brd4, 'r--', 'DisplayName', 'Average␣of␣values␣at␣room␣temp.');
```

```matlab
plot(x1_Brd4, y1_Brd4, 'bo', 'DisplayName', 'Values␣at␣room␣temp.');
plot(x2_Brd4, y2_Brd4, 'g.', 'DisplayName', 'Values␣after␣the␣beginning␣of␣the␣test');
xlabel('Temperature␣( C )');
ylabel('Acceleration␣(g)');
title('Acceleration␣vs␣Temperature␣for␣Brd4');
legend('show');
ylim([mean_Brd4_AccZ(1) - yRange/2, mean_Brd4_AccZ(1) + yRange/2]);  % Set y-axis limits
grid on;
hold off;

% Data to plot for Brd2
x1_Brd2 = mean_Brd2_Temp(1:8);  % Temperatures of the first 8 values
y1_Brd2 = mean_Brd2_AccZ(1:8);  % Accelerations of the first 8 values

x2_Brd2 = mean_Brd2_Temp(9:end);  % Temperatures from the remaining values
y2_Brd2 = mean_Brd2_AccZ(9:end);  % Accelerations from the remaining values

% Create plot for Brd2
figure;
hold on;
yline(mean_first_8_AccZ_Brd2, 'r--', 'DisplayName', 'Average␣of␣values␣at␣room␣temp.');
plot(x1_Brd2, y1_Brd2, 'bo', 'DisplayName', 'Values␣at␣room␣temp.');
plot(x2_Brd2, y2_Brd2, 'g.', 'DisplayName', 'Values␣after␣the␣beginning␣of␣the␣test');
xlabel('Temperature␣( C )');
ylabel('Acceleration␣(g)');
title('Acceleration␣vs␣Temperature␣for␣Brd2');
legend('show');
ylim([mean_Brd2_AccZ(1) - yRange/2, mean_Brd2_AccZ(1) + yRange/2]);  % Set y-axis limits
grid on;
hold off;

% Data to plot for Brd1
x1_Brd1 = mean_Brd1_Temp(1:8);  % Temperatures of the first 8 values
y1_Brd1 = mean_Brd1_AccZ(1:8);  % Accelerations of the first 8 values

```

```
x2_Brd1 = mean_Brd1_Temp(9:end);  % Temperatures from the remaining values
y2_Brd1 = mean_Brd1_AccZ(9:end);  % Accelerations from the remaining values

% Create plot for Brd1
figure;
hold on;
yline(mean_first_8_AccZ_Brd1, 'r--', 'DisplayName', 'Average␣of␣values␣at␣room␣temp.');
plot(x1_Brd1, y1_Brd1, 'bo', 'DisplayName', 'Values␣at␣room␣temp.');
plot(x2_Brd1, y2_Brd1, 'g.', 'DisplayName', 'Values␣after␣the␣beginning␣of␣the␣test');
xlabel('Temperature␣( C )');
ylabel('Acceleration␣(g)');
title('Acceleration␣vs␣Temperature␣for␣Brd1');
legend('show');
ylim([mean_Brd1_AccZ(1) - yRange/2, mean_Brd1_AccZ(1) + yRange/2]);  % Set y-axis limits
grid on;
hold off;

% Data to plot for Brd3
x1_Brd3 = mean_Brd3_Temp(1:8);  % Temperatures of the first 8 values
y1_Brd3 = mean_Brd3_AccZ(1:8);  % Accelerations of the first 8 values

x2_Brd3 = mean_Brd3_Temp(9:end);  % Temperatures from the remaining values
y2_Brd3 = mean_Brd3_AccZ(9:end);  % Accelerations from the remaining values

% Create plot for Brd3
figure;
hold on;
yline(mean_first_8_AccZ_Brd3, 'r--', 'DisplayName', 'Average␣of␣values␣at␣room␣temp.');
plot(x1_Brd3, y1_Brd3, 'bo', 'DisplayName', 'Values␣at␣room␣temp.');
plot(x2_Brd3, y2_Brd3, 'g.', 'DisplayName', 'Values␣after␣the␣beginning␣of␣the␣test');
xlabel('Temperature␣( C )');
ylabel('Acceleration␣(g)');
title('Acceleration␣vs␣Temperature␣for␣Brd3');
legend('show');
ylim([mean_Brd3_AccZ(1) - yRange/2, mean_Brd3_AccZ(1) + yRange/2]);  % Set y-axis limits
```

```
grid on;
hold off;
```

Bibliography

150080VS75000 - Digikey (n.d.). URL: https://www.digikey.it/it/products/detail/w%C3%BCrth-elektronik/150080VS75000/4489924?s=N4IgTCBcDaIOwGYwFoAsBOAHANmQRmQDkAREAXQF8

687-7828 - RS (n.d.). URL: https://it.rs-online.com/web/p/adattatori-di-interfaccia-e-convertitori/6877828?searchId=f2637abe-524b-40ae-98da-2c82aabcf46c&gb=s.

687-7834 - RS (n.d.). URL: https://it.rs-online.com/web/p/adattatori-di-interfaccia-e-convertitori/6877834?searchId=5a86a2e9-84fb-4acf-be04-1ecde2bf9204&gb=s.

Acellent Technologies (n.d.). https://www.cbinsights.com/company/acellent-technologies-1.

Adhikari, R. S., A. Bagchi, and O. Moselhi (2014). «Automated condition assessment of concrete bridges with digital imaging». In: *Smart Structures and Systems* 13.6, pp. 901–925. DOI: 10.12989/sss.2014.13.6.901.

ADXL354/ADXL355 Low Noise, Low Drift, Low Power, 3-Axis MEMS Accelerometers Data Sheet (n.d.). URL: https://www.analog.com/adxl355?doc=adxl354_355.pdf.

AeroCone Rain Collector with Flat Base for Vantage Pro2 and EnviroMonitor (tipping spoon) - SKU 6464, 6464M (n.d.). URL: https://www.davisinstruments.com/products/aerocone-rain-collector-with-flat-base-for-vantage-pro2?srsltid=AfmBOooqwWZB7gijAPDfDT1T14fPBOH9qACPXpA5rZrjkhbR9yTtdIsv.

All about RS485 – How RS485 Works and How to Implement RS485 into Industrial Control Systems? (N.d.). https://www.seeedstudio.com/blog/2021/03/18/how-rs485-works-and-how-to-implement-rs485-into-industrial-control-systems/?srsltid=AfmBOoqyX7MEtR627LWb5BHOPjWCjdpIl8SvrpY79G6k6cwx8bMYGldj.

Altium Designer 24 (n.d.). https://www.altium.com/it/altium-designer?srsltid=AfmBOordKoQ_p7UJlzAN5Lee5Q9h1sVrQ8S7mhxi8aV30yymtuFWzqwi.

Amazon (n.d.). https://www.amazon.it/ref=nav_logo.

Anemometer for Weather Monitor or Wizard - SKU 7911 (n.d.). URL: https://www.davisinstruments.com/products/anemometer-for-weather-monitor-or-wizard?srsltid=AfmBOorVXBST38PFtyT-y8J2DhhKjs9w2w8QIMrIlarGuJO63P3UxT_A.

Ásgrímsson, Davíð, Ignacio Gonzalez, Giampiero Salvi, and Raid Karoumi (2021). «Bayesian Deep Learning for Vibration-Based Bridge Damage Detection». In: *Structural Health Monitoring Based on Data Science Techniques*, pp. 27–43. DOI: 10.1007/978-3-030-81716-9_2.

Basics of the I2C Communication Protocol (n.d.). https://www.circuitbasics.com/basics-of-the-i2c-communication-protocol/.

Bassoli, Elisa, Loris Vincenzi, Marco Bovo, and Claudio Mazzotti (2015). «Dynamic identification of an ancient masonry bell tower using a MEMS-based acquisition system». In: *2015 IEEE Workshop on Environmental, Energy, and Structural Monitoring Systems (EESMS) Proceedings*, pp. 226–231. DOI: 10.1109/EESMS.2015.7175882.

Battery - Amazon website (n.d.). URL: https://www.amazon.it/BATTERIA-SPEED-L3-80Ah-POSITIVO-DESTRA/dp/B0876Y6PLM/ref=sr_1_6?__mk_it_IT=%C3%85M%C3%85%C5%BD%C3%95%C3%91&crid=2E2W7WU3EIKXB&dib=eyJ2IjoiMSJ9.lyOE98gxYbA96Sr1KwXRD4xanHX00kALgX8c87BDSJ2Rp-QekRfnZUXPYi-fdKRK7jrJ36UIJ_qI7YOA7JPv__gX6ZueH6Qe9xmsDQE4ajqiQeooIpUFuJ227ebTT_WogyRu8ZoNFKoIsvOQzOvFdYneUGOVp7s8b
RRb1tRvmo5v4cNpbV3sjMT81niLicqBz8DQrlZxlBOBRSznQeNDvvwG-AY.4ipslroGbPbkgabQc-_qoOSfvbKRLyyo4yI_GXKvOxE&dib_tag=se&keywords=batteria+al+piombo+70+ah&qid=1709200032&sprefix=batteria+al+piombo+70+ah%2Caps%2C129&sr=8-6.

Bedon, C., E. Bergamo, M. Izzi, and S. Noè (2018). «Prototyping and validation of MEMS accelerometers for structural health monitoring—The case study of the Pietratagliata cable-stayed bridge». In: *Journal of Sensor and Actuator Networks* 7.3, p. 30. DOI: 10.3390/jsan7030030.

Bedon, Chiara, Enrico Bergamo, Matteo Izzi, and Salvatore Noè (2018). «Prototyping and Validation of MEMS Accelerometers for Structural Health Monitoring—The Case Study of the Pietratagliata Cable-Stayed Bridge». In: *Journal of Sensor and Actuator Networks* 7.3, p. 30. DOI: 10.3390/jsan7030030.

Berica Cavi (n.d.). URL: https://bericacavi.com/cavi/li2ycyv-tp-per-esterno/.

Bremer, K., M. Wollweber, F. Weigand, M. Rahlves, M. Kuhne, R. Helbig, and B. Roth (2016). «Fibre Optic Sensors for the Structural Health Monitoring of Building Structures». In: *Procedia Technology* 26, pp. 524–529. DOI: https://doi.org/10.1016/j.protcy.2016.08.065.

Cha, Y. J., W. Choi, and O. Büyüköztürk (2017). «Deep Learning-Based Crack Damage Detection Using Convolutional Neural Networks». In: *Computer-Aided Civil and Infrastructure Engineering* 32.5, pp. 361–378. DOI: 10.1111/mice.12263.

Chen, Hua-Peng and Yi-Qing Ni (Feb. 2018). «Introduction to Structural Health Monitoring». In: pp. 1–14. ISBN: 9781119166436. DOI: 10.1002/9781119166641.ch1.

Cigada, Alfredo, Massimiliano Lurati, Matteo Redaelli, and Marcello Vanali (Jan. 2007). «Mechanical Performance and Metrological Characterization of MEMS Accelerometers and Application in Modal Analysis». In.

D'Alessando, Antonino and Salvatore Scudero (Jan. 2021). «MEMS-Based System for Structural Health Monitoring and Earthquake Observation in Sicily». In: pp. 89–95. ISBN: 978-3-030-64593-9. DOI: 10.1007/978-3-030-64594-6_10.

Data Sheet AM2315C Humidity and Temperature Module (n.d.). URL: https://www.digikey.it/it/products/detail/adafruit-industries-llc/5182/15204091?utm_adgroup=&utm_source=google&utm_medium=cpc&utm_campaign=PMAX%20Shopping_Product_Low%20Performers&utm_term=&productid=15204091&utm_content=&utm_id=go_cmp-20112759545_adg-_ad-__dev-c_ext-_prd-15204091_sig-Cj0KCQjwjY64BhCaARIsAIfc7YYEAJ2ZkvThe2cf6p8bxP1MVRvQ0FrHFi4488GEOQr7ThtLQV0yoTsaAuY

wcB&gad_source=1&gclid=Cj0KCQjwjY64BhCaARIsAIfc7YYEAJ2ZkvThe2cf6p8bxP1MVRvQ0FrHFi4488GE
wcB.

Datasheet MAX31865ATP+ (n.d.). URL: https://www.analog.com/media/en/technical-documentation/data-sheets/max31865.pdf.

Digikey (n.d.). https://www.digikey.it/.

Doebling, Scott, Charles Farrar, and Michael Prime (Mar. 1998). «A Summary Review of Vibration-Based Damage Identification Methods». In: *The Shock and Vibration Digest* 30, pp. 91–105. DOI: 10.1177/058310249803000201.

Dragonetti, Pierpaolo, Marco Civera, and Gaetano Miraglia (2023). «Experimental validation of system identification techniques for structural health monitoring by means of cameras and accelerometers». In.

Faulkner, Karen, James Brownjohn, Ying Wang, and Farhad Huseynov (Sept. 2020). «Tracking bridge tilt behaviour using sensor fusion techniques». In: *Journal of Civil Structural Health Monitoring* 10. DOI: 10.1007/s13349-020-00400-9.

Friis, Tobias, Silas Christensen, Silja Nielsen, Martin Lollesgaard, Torben Bangsgaard, and Martin Havelykke (June 2024). «Digital Health Monitoring of the Great Belt Suspension Bridge». In.

Fukuda, Y., M. Q. Feng, and M. Shinozuka (2010). «Cost-effective vision-based system for monitoring dynamic response of civil engineering structures». In: *Structural Control and Health Monitoring* 17.8, pp. 918–936. DOI: 10.1002/stc.360.

Gasparin, E., G. Santi, and A. Nussbaumer (2010). «Eddy Current Crack Monitoring System for Structural Health Monitoring (SHM) Applications». In: *Proceedings of the International Conference on Structural Health Monitoring of Intelligent Infrastructure (SHMII)*. École polytechnique fédérale de Lausanne (EPFL). URL: https://infoscience.epfl.ch/record/140956.

Gentile, Claudio, Alessandro Ruccolo, and Federico Canali (2019). «Continuous monitoring of the Milan Cathedral: dynamic characteristics and vibration-based SHM». In: *Journal of Civil Structural Health Monitoring* 9, pp. 671–688. DOI: 10.1007/s13349-019-00361-8. URL: https://doi.org/10.1007/s13349-019-00361-8.

Girolami, Alberto, Davide Brunelli, and Luca Benini (2017). «Low-cost and distributed health monitoring system for critical buildings». In: *2017 IEEE Workshop on Environmental, Energy, and Structural Monitoring Systems (EESMS)*, pp. 1–6. DOI: 10.1109/EESMS.2017.8052686.

Gragnaniello, Chiara, Giulio Mariniello, Tommaso Pastore, and Domenico Asprone (2024). «BIM-based design and setup of structural health monitoring systems». In: *Automation in Construction* 158. DOI: 10.1016/j.autcon.2023.105245.

Ho, H.-N., K.-D. Kim, Y.-S. Park, and J.-J. Lee (2013). «An efficient image-based damage detection for cable surface in cable-stayed bridges». In: *NDT and E International* 58, pp. 18–23. DOI: 10.1016/j.ndteint.2013.04.006.

Huffington Post (2018). *CNR: "In Italia migliaia di ponti troppo vecchi, oltre la vita la durata di vita per cui erano stati progettati" — huffingtonpost.it*. Available at: https://www.huffingtonpost.it/2018/08/14/cnr-in-italia-migliaia-di-ponti-troppo-vecchi-oltre-la-vita-la-durata-di-vita-per-cui-erano-stati-progettati_a_23502132/.

I2C Primer: What is I2C? (Part 1) (n.d.). https://www.analog.com/en/resources/technical-articles/i2c-primer-what-is-i2c-part-1.html.

I3G4250D MEMS Motion Sensor: 3-axis Digital Output Gyroscope (n.d.). URL: https://www.st.com/resource/en/datasheet/i3g4250d.pdf.

Indhu, R., G. Ram Sundar, and H. Summia Parveen (2022). «A Review of Machine Learning Algorithms for vibration-based SHM and vision-based SHM». In: pp. 418–422. DOI: 10.1109/ICAIS53314.2022.9742818.

Integrated Development Environment for STM32 (n.d.). https://www.st.com/en/development-tools/stm32cubeide.html.

Jang, Shinae et al. (July 2010). «Structural Health Monitoring of a Cable-Stayed Bridge Using Smart Sensor Technology: Deployment and Evaluation». In: *Smart Structures and Systems* 6. DOI: 10.12989/sss.2010.6.5_6.439.

Jrade, Ahmad, Farnaz Jalaei, Jieying Jane Zhang, Saeed Jalilzadeh Eirdmousa, and Farzad Jalaei (2023). «Potential Integration of Bridge Information Modeling and Life Cycle Assessment/Life Cycle Costing Tools for Infrastructure Projects within Construction 4.0: A Review». In: *Sustainability* 15.20. DOI: 10.3390/su152015049.

Lynch, Jerome, Yang Wang, Kenneth Loh, Jin-Hak Yi, and Chung-Bang Yun (Dec. 2006). «Performance monitoring of the Geumdang Bridge using a dense network of high-resolution wireless sensors». In: *Smart Materials and Structures* 15. DOI: 10.1088/0964-1726/15/6/008.

Marina Marzulli (2023). *A 5 anni dal crollo del Ponte Morandi, quanto sono sicure le infrastrutture in Italia?* https://www.fleetmagazine.com/sicurezza-ponti-italia/.

MAX31865ATP+ - Digikey (n.d.). URL: https://www.digikey.it/it/products/detail/analog-devices-inc-maxim-integrated/MAX31865ATP/3681476?s=N4IgTCBcDaILIEEAaBmAjADgGwFYEBUAFAagFoA5AERAF0BfIA.

MAX3232IPWR - Digikey (n.d.). URL: https://www.digikey.it/it/products/detail/texas-instruments/MAX3232IPWR/484752?s=N4IgTCBcDa4JwDYC0BGAzCgDJ1SByAIiALoC%2BQA.

McGuire, Brian, Rebecca Atadero, Caroline Clevenger, and Mehmet E. Ozbek (2016). «Bridge Information Modeling for Inspection and Evaluation». In: *Journal of Bridge Engineering* 21.4. DOI: 10.1061/(ASCE)BE.1943-5592.0000818.

MCP1525T-I/TT - Digikey (n.d.). URL: https://www.digikey.it/it/products/detail/microchip-technology/MCP1525T-I-TT/443710?s=N4IgTCBcDaILIGEAKBGArGNAVAtASQHossFc

Melchiorre, Jonathan, Amedeo Manuello Bertetto, Marco Martino Rosso, and Giuseppe Carlo Marano (2023). «Acoustic Emission and Artificial Intelligence Procedure for Crack Source Localization». In: *Sensors* 23.2, p. 693. DOI: 10.3390/s23020693.

MISTRAS Group - Bridge Monitoring (n.d.). https://www.mistrasgroup.com/how-we-help/monitoring/bridges/.

Modares, M. and N. Waksmanski (2013). «Overview of Structural Health Monitoring for Steel Bridges». In: *Practice Periodical on Structural Design and Construction* 18, pp. 187–191. DOI: 10.1061/(ASCE)SC.1943-5576.0000173.

Modares, Mehdi and Natalie Waksmanski (2013). «Overview of Structural Health Monitoring for Steel Bridges». In: *Practice Periodical on Structural Design and Construction* 18.3, pp. 187–191. DOI: 10.1061/(ASCE)SC.1943-5576.0000154.

Moreno-Gomez, Alejandro, Carlos A. Perez-Ramirez, Aurelio Dominguez-Gonzalez, Martin Valtierra-Rodriguez, Omar Chavez-Alegria, and Juan P. Amezquita-Sanchez (2018).

«Sensors Used in Structural Health Monitoring». In: *Archives of Computational Methods in Engineering* 25, pp. 901–918. DOI: 10.1007/s11831-017-9217-4.

Mutlib, Nadom Khalifa, Shahrizan Bin Baharom, Ahmed El-Shafie, and Mohd Zaki Nuawi (2015). «Ultrasonic health monitoring in structural engineering: buildings and bridges». In: *Structural Control and Health Monitoring* 22.11, pp. 1387–1408. DOI: 10.1002/stc.1800.

Nair, Arun and C. S. Cai (2010). «Acoustic Emission Monitoring of Bridges: Review and Case Studies». In: *Engineering Structures* 32.6, pp. 1704–1714. DOI: 10.1016/j.engstruct.2010.02.020.

NJU77552G-TE2 - Digikey (n.d.). URL: https://www.digikey.it/it/products/detail/nisshinbo-micro-devices-inc/NJU77552G-TE2/10671850?s=N4IgTCBcDa4IxgJwFoByApAqgd

OSMOS Group (n.d.). https://www.cbinsights.com/company/osmos-group.

OSMOS Group - Weigh in motion & Deformation (n.d.). https://www.osmos-group.com/en/solutions/weigh-motion-deformation.

Papazian, John M. et al. (2007). «Sensors for monitoring early stage fatigue cracking». In: *International Journal of Fatigue* 29.4, pp. 848–856. DOI: 10.1016/j.ijfatigue.2007.01.007.

PDQE15-Q24-S5-D - Digikey (n.d.). URL: https://www.digikey.it/it/products/detail/cui-inc/PDQE15-Q24-S5-D/10230147?s=N4IgTCBcDaIIwAYwFoBsYCsBmZA5AIiALoC%2BQA.

Picozzi, M. C. et al. (2009). «Wireless technologies for the monitoring of strategic infrastructures: An ambient vibration test on the Fatih Sultan Mehmet suspension bridge in Istanbul, Turkey». In: *Bulletin of Earthquake Engineering* 8, pp. 671–691.

Platinum sensor with round ceramic housing for low temperatures (n.d.). URL: https://www.digikey.it/it/products/detail/innovative-sensor-technology-usa-division/P0K1-281-2K-B-150-R-S/13686761.

Rainieri, C., Danilo Gargaro, and Giovanni Fabbrocino (Apr. 2019). «Hardware and Software Solutions for Seismic SHM of Hospitals». In: pp. 279–300. DOI: 10.1007/978-3-030-13976-6_12.

Redazione ANSA (2018). *Cnr: in Italia migliaia i ponti troppo vecchi - Notizie - Ansa.it — ansa.it*. Available at: https://www.huffingtonpost.it/2018/08/14/cnr-in-italia-migliaia-di-ponti-troppo-vecchi-oltre-la-vita-la-durata-di-vita-per-cui-erano-stati-progettati_a_23502132/

Redazione BitMAT (2021). *In Italia ci sono 1,5 milioni di ponti. Quanti sono sicuri? — itismagazine.it*. Available at: https://www.itismagazine.it/featured/in-italia-ci-sono-15-milioni-di-ponti-quanti-sono-sicuri/.

RS (n.d.). https://it.rs-online.com/web/?srsltid=AfmBOooPgEm3FhmRyKaUnch_8XGTtU_ew2qEO3Fe4dlYGOlhq-jrL1sX.

RS232 Information and Pinout (n.d.). https://satoms.com/rs232-information-and-pinout/.

Saisi, A., C. Gentile, and M. Guidobaldi (2015). «Post-earthquake continuous dynamic monitoring of the Gabbia Tower in Mantua, Italy». In: *Construction and Building Materials* 81, pp. 101–112. DOI: 10.1016/j.conbuildmat.2015.02.010.

Salawu, Olukunle S. (1997). «Detection of Structural Damage through Changes in Frequency: A Review». In: *Engineering Structures* 19.9, pp. 718–723.

Scuro, C., F. Lamonaca, S. Porzio, G. Milani, and R. S. Olivito (2021). «Internet of Things (IoT) for masonry structural health monitoring (SHM): Overview and examples of innovative systems». In: *Construction and Building Materials* 272, p. 123091. DOI: 10.1016/j.conbuildmat.2021.123091.

Serial Communications Protocols - Part Four: RS-485 and Baud Rates (n.d.). https://resources.altium.com/p/serial-communications-protocols-rs-485.

Shokravi, Hoofar, Hooman Shokravi, Norhisham Bakhary, Mahshid Heidarrezaei, Seyed Saeid Rahimian Koloor, and Michal Petrů (June 2020). «Vehicle-Assisted Techniques for Health Monitoring of Bridges». In: *Sensors* 20.12. DOI: 10.3390/s20123460.

Siringoringo, Dionysius and Yozo Fujino (Aug. 2018). «Seismic response of a suspension bridge: Insights from long-term full-scale seismic monitoring system». In: *Structural Control and Health Monitoring* 25. DOI: 10.1002/stc.2252.

SPI (n.d.). https://docs.tizen.org/iot/guides/peripheral-io-api-spi/.

SPI in AVR ATmega16/ATmega32 (n.d.). https://www.electronicwings.com/avr-atmega/atmega1632-spi.

Stannard, Liam (2021). *What is BIM? Building Information Modeling Explained.* https://www.bigrentz.com/blog/what-is-bim?srsltid=AfmBOoonipLB4RmOpbcojnGM46Biz-bB-njlj7uT4eeVhL6QYlVNLQsO.

STM32L431CBT6 - Digikey (n.d.). URL: https://www.digikey.it/it/products/detail/stmicroelectronics/STM32L431CBT6/6621811?s=N4IgTCBcDaICwE4DsBaAjEhBWVA5AIiALoC%
2BQA.

Tan, Chengjun, Nasim Uddin, Eugene J. OBrien, Patrick J. McGetrick, and Chul-Woo Kim (2019). «Extraction of Bridge Modal Parameters Using Passing Vehicle Response». In: *Journal of Bridge Engineering* 24.9. DOI: 10.1061/(ASCE)BE.1943-5592.0001477.

Texas Instruments (2021). *THVD1400, THVD1420 3.3-V to 5-V RS-485 Transceivers in Small Package with ±12-kV IEC ESD Protection datasheet.*

THVD1400DR - Digikey (n.d.). URL: https://www.digikey.it/it/products/detail/texas-instruments/THVD1400DR/13636656?s=N4IgTCBcDa4JwDYC0AVAEgNQCIEYAsADAVgEoDCKSAcliAL
2BQA.

Ubertini, F., G. Comanducci, N. Cavalagli, AL. Pisello, AL. Materazzi, and F. Cotana (2017). «Environmental effects on natural frequencies of the San Pietro bell tower in Perugia, Italy, and their removal for structural performance assessment». In: *Mechanical Systems and Signal Processing* 82, pp. 307–322. DOI: 10.1016/j.ymssp.2016.05.025.

VEGAPULS C 23 Sensore radar con cavo integrato per la misura di livello continua su liquidi (n.d.). URL: https://www.vega.com/it-it/prodotti/catalogo-prodotti/misura-di-livello/radar/vegapuls-c-23.

VX7803-500 - Digikey (n.d.). URL: https://www.digikey.it/it/products/detail/cui-inc/VX7803-500/7350282?s=N4IgTCBcDaIIwAYwFoAsZUGZkDkAiIAugL5A.

Weng, Jian-Huang, Chin-Hsiung Loh, Jerome P. Lynch, Kung-Chun Lu, Pei-Yang Lin, and Yang Wang (2008). «Output-only modal identification of a cable-stayed bridge using wireless monitoring systems». In: *Engineering Structures* 30.6, pp. 1820–1830. DOI: 10.1016/j.engstruct.2007.10.002.

What is the UART communication protocol (n.d.). https://soldered.com/learn/what-is-the-uart-communication-protocol/.

Wu, Tiange, Guowei Liu, Shenggui Fu, and Fei Xing (2020). «Recent Progress of Fiber-Optic Sensors for the Structural Health Monitoring of Civil Infrastructure». In: *Sensors* 20.16, p. 4517. DOI: `10.3390/s20164517`.

Yang, Y. B., C. W. Lin, and J. D. Yau (2004). «Extracting bridge frequencies from the dynamic response of a passing vehicle». In: *Journal of Sound and Vibration* 272.3-5, pp. 471–493. DOI: `10.1016/S0022-460X(03)00378-X`.

Yu, Tzuyang, Qixiang Tang, and Sanjana Vinayaka (2024). «Identifying structural properties of a steel railway bridge for structural health monitoring using laser Doppler vibrometry». In: *Automation in Construction* 160, p. 105320. DOI: `https://doi.org/10.1016/j.autcon.2024.105320`.

Zinno, Raffaele, Sina Shaffiee Haghshenas, Giuseppe Guido, Kaveh Rashvand, Alessandro Vitale, and Ali Sarhadi (2023). «The State of the Art of Artificial Intelligence Approaches and New Technologies in Structural Health Monitoring of Bridges». In: *Applied Sciences* 13.1, p. 97. DOI: `10.3390/app13010097`.