

Reconocimiento de financiación

Arthur Blanco
Instrumentación experimental con galgas extensométricas

Proyecto realizado con la Beca Leonardo de Investigación Científica y Creación Cultural 2024 de la Fundación BBVA.

Este trabajo se enmarca en las actividades de investigación y difusión de resultados del proyecto *Cost-effective monitoring systems for enhanced and safe operation of renewable energies*.

La Fundación BBVA no se responsabiliza de las opiniones, afirmaciones y contenidos incluidos en el proyecto y/o de los resultados derivados del mismo, que son responsabilidad exclusiva de sus autores.

Low-Cost Structural Monitoring System

Contents

1 Introduction **2**
- 1.1 Internship Context . . . 2
- 1.2 Project Objectives . . . 3

2 Theoretical Study **3**
- 2.1 Stress and Strain in a Beam . . . 3
  - 2.1.1 Normal Bending Stress $\sigma(x, y)$ . . . 3
- 2.2 Strain Gauges . . . 4
- 2.3 Wheatstone Bridge . . . 5
- 2.4 Quarter-Bridge Configuration . . . 6
- 2.5 INA125P Operation . . . 6
- 2.6 Acquisition with the Arduino Uno . . . 7

3 Simulations with Falstad **7**

4 Equipment and Experimental Setup **8**
- 4.1 Equipment Used . . . 8

5 Experimental Work **8**
- 5.1 Strain Gauge Installation . . . 8
- 5.2 Measurements with DewesoftX . . . 9
- 5.3 INA125P and Arduino Uno Wiring . . . 10
- 5.4 Arduino Code and Filtering . . . 11
- 5.5 Initial Circuit Check . . . 11
- 5.6 INA125P Diagnostic . . . 12
- 5.7 Strain-Gauge Bridge Diagnostic . . . 12
- 5.8 Quarter-Bridge Construction and Test . . . 14

6 Results, Discussion and Limitations **15**

7 Conclusion **16**

Appendices **17**
- Appendix 1 – Diagnostic code and voltage readings . . . 17
- Appendix 2 – Full-bridge measurement code with filtering . . . 18
- Appendix 3 – Arduino resistance measurement code . . . 22
- Appendix 4 – Quarter-bridge measurement code . . . 23

1 Introduction

1.1 Internship Context

In the context of the energy transition, electricity production methods are changing and becoming more diverse. Wind, hydroelectric and nuclear energy are important examples of low-carbon electricity production. These technologies use large structures such as wind turbine blades, dams and nuclear facilities.

During operation, these structures are exposed to mechanical loads, vibrations and repeated loading cycles. These effects can create strain and, over time, may lead to damage. Monitoring their mechanical behaviour is therefore important to detect abnormal changes and reduce the risk of structural degradation.

Industrial monitoring systems can be expensive and complex to install. For this reason, it is interesting to study a simpler, low-cost and reproducible solution based on a *low-tech* approach.



Figure 1: Nuclear power plant



Figure 2: Hydroelectric dam



Figure 3: Wind turbine

1.2 Project Objectives

The aim of this project is to study a simple system for measuring strain in a metal structure. The system uses strain gauges connected in a Wheatstone bridge. The low voltage produced by the bridge is amplified by an INA125P instrumentation amplifier and then measured by an Arduino Uno.

The work first includes a theoretical study of the measurement chain and a simulation of the electronic circuit with Falstad. The circuit is then built and tested in the laboratory to check the amplifier, the strain-gauge bridges and the Arduino acquisition. The results are also compared with measurements obtained using the professional DewesoftX system.

The final objective is to evaluate whether a strain monitoring system can be made with common electronic components while remaining simple, low-cost and easy to reproduce.

2 Theoretical Study

2.1 Stress and Strain in a Beam

To improve measurement accuracy, the strain gauges should be placed in the area where the mechanical stress is highest. A theoretical study of the metal beam can therefore be used to identify this area.

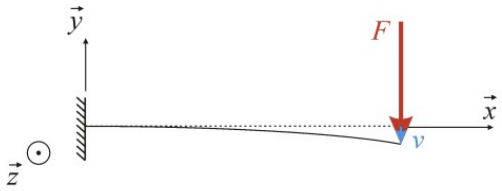


Figure 4: Cantilever beam

The shear force is given by:

$$\Sigma F_y = 0 \implies V(x) = F$$

The bending moment is given by:

$$\Sigma M = 0 \implies M(x) = F(L - x)$$

The second moment of area is:

$$I = \frac{le^3}{12}$$

2.1.1 Normal Bending Stress $\sigma(x, y)$

We have:

$$\sigma(x, y) = \frac{M(x)y}{I}$$

At $y = \frac{e}{2}$, the stress is maximum. Therefore:

$$\sigma_{max}(x) = \frac{M(x)\frac{e}{2}}{I} = \frac{6F(L - x)}{le^2}$$

The beam deflection can also be calculated:

$$El\frac{d^2y}{dx^2} = M(x) = F(L-x) \implies y(x) = \frac{F}{6El}(3Lx^2 - x^3)$$

The maximum stress is therefore located at $x = 0$, with:

$$\sigma_{max} = \frac{6FL}{le^2}$$

For this reason, the strain gauges are placed close to this area.

2.2 Strain Gauges

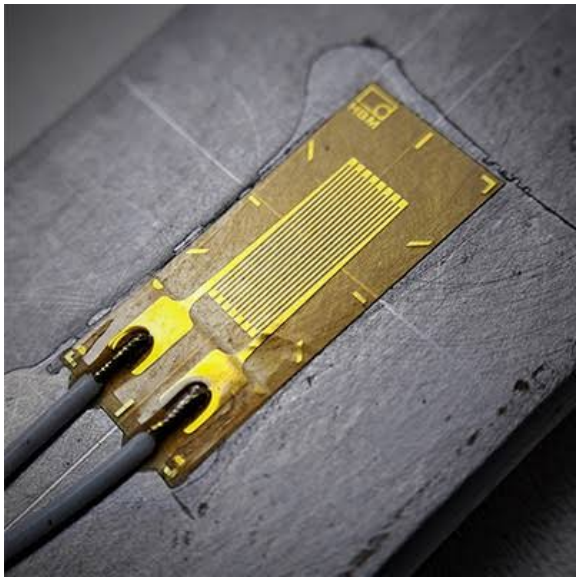


Figure 5: Strain gauge

A strain gauge is a resistive sensor used to measure the mechanical strain of a structure. It is made of a thin conductive track arranged in a grid on an insulating support. This shape increases the length of the conductor while keeping the sensor compact.

The gauge is bonded directly to the surface of the part, in the main strain direction. When the structure is stretched, the conductive track also becomes longer and its resistance increases. When the structure is compressed, its resistance decreases.

The relative resistance change is linked to strain by:

$$\frac{\Delta R}{R} = k\varepsilon, \tag{1}$$

where R is the initial gauge resistance, ΔR is the resistance change, k is the gauge factor and ε is the strain. For a metallic strain gauge, the gauge factor is usually close to 2.

Because these resistance changes are very small, the gauge is generally used in a Wheatstone bridge. This circuit converts the resistance variation into a voltage that can then be amplified and measured.

2.3 Wheatstone Bridge

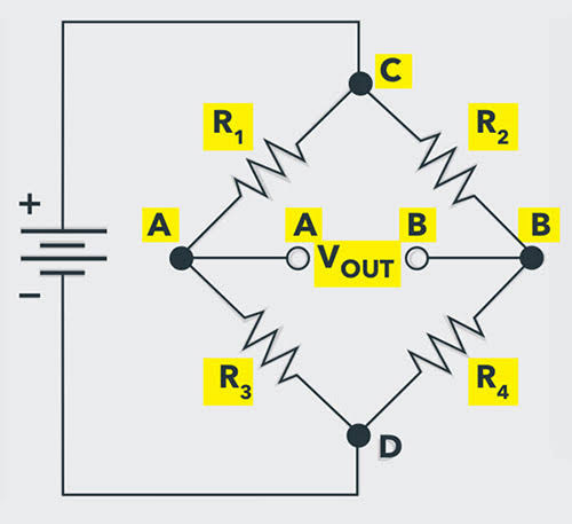


Figure 6: Wheatstone bridge

A Wheatstone bridge is a suitable low-cost solution for measuring strain in a beam. The strain gauges are connected as resistive elements in the bridge. When the gauges are deformed, their resistance changes and the output voltage of the bridge also changes. This output can then be used to monitor mechanical strain in the structure.

The bridge output is:

$$V_{out} = V_{in}\left(\frac{R_4}{R_2+R_4} - \frac{R_3}{R_1+R_3}\right)$$

The bridge is balanced when:

$$\frac{R_1}{R_3} = \frac{R_2}{R_4}$$

This is normally the expected condition when the bridge is at rest and no strain is applied.

In a full bridge, all four resistors are active strain gauges. With the convention:

$$V_{\text{out}} = V_B - V_A,$$

we consider the following strains:

$$\epsilon_1 = \epsilon_4 = +\epsilon \quad \text{and} \quad \epsilon_2 = \epsilon_3 = -\epsilon$$

Therefore, R_1 and R_4 are in tension, while R_2 and R_3 are in compression.

The relation between relative resistance change and strain is:

$$\frac{\Delta R_i}{R} = K\varepsilon_i,$$

where K is the gauge factor.

For small resistance changes, the bridge output is:

$$\frac{V_{\text{out}}}{V_{\text{in}}} \simeq \frac{K}{4}\left(\varepsilon_1 - \varepsilon_2 - \varepsilon_3 + \varepsilon_4\right).$$

This gives:

$$\frac{V_{\text{out}}}{V_{\text{in}}} \simeq \frac{K}{4}(4\varepsilon),$$

so:

$$V_{out} = K\epsilon V_{in}$$

In this ideal symmetrical configuration, this relation is exact. The strain can therefore be calculated from the output voltage:

$$\epsilon = \frac{V_{out}}{KV_{in}}$$

2.4 Quarter-Bridge Configuration

A quarter bridge uses one active strain gauge and three fixed resistors. When the gauge is strained, its resistance changes and the bridge becomes unbalanced, creating a differential output voltage.

For small strains, the relation between bridge output and strain can be approximated by:

$$\frac{V_{\text{bridge}}}{V_{\text{EX}}} \approx \frac{k\varepsilon}{4}, \tag{2}$$

where k is the gauge factor, ε is the strain and V_{EX} is the bridge excitation voltage.

This configuration is simple and low-cost, but it is more sensitive to temperature changes and resistor tolerances than a full bridge.

2.5 INA125P Operation

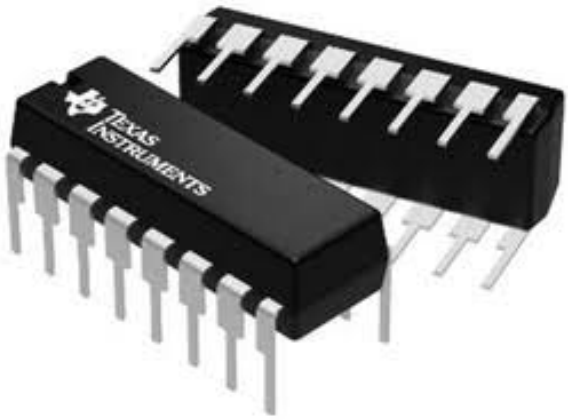


Figure 7: INA125P instrumentation amplifier

The INA125P is an instrumentation amplifier designed for low-level signals from bridge sensors. It is useful in this low-cost system because the differential voltage produced by the Wheatstone bridge is very small.

It also includes a precision voltage reference with several possible values, including 2.5 V. In this project, the 2.5 V reference is used to excite the bridge.

The amplifier gain is set with an external resistor R_G connected between pins 8 and 9:

$$G = 4 + \frac{60\,000}{R_G}. \tag{3}$$

The output voltage depends on the difference between V_{IN+} and V_{IN-}:

$$V_O = V_{\text{IAREF}} + G(V_{IN+} - V_{IN-}). \tag{4}$$

In this circuit, the INA125P is powered from 5 V and V_{IAREF} is set to about 2.5 V. When the bridge is balanced, the input difference is close to zero and V_O is therefore close to 2.5 V. When strain unbalances the bridge, the small differential voltage is amplified and can be measured by the Arduino Uno.

2.6 Acquisition with the Arduino Uno

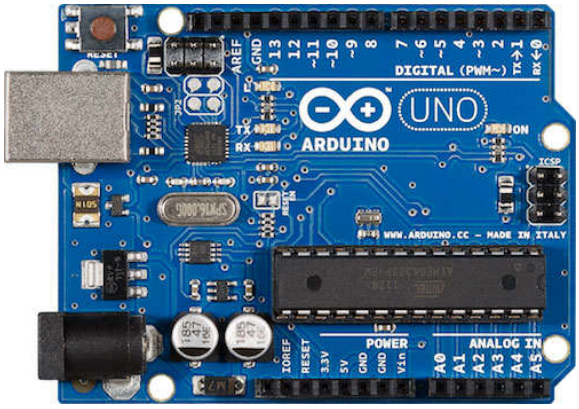


Figure 8: Arduino Uno board

The Arduino Uno operates at 5 V and has six analog inputs, from `A0` to `A5`. In this project, it is used to acquire the voltages from the INA125P and send the results to a computer.

The Arduino Uno analog-to-digital converter has a 10-bit resolution. A voltage between 0 V and 5 V is therefore converted into an integer value between 0 and 1023:

$$V = \frac{\text{ADC} \times 5}{1023}. \tag{5}$$

One ADC step is approximately:

$$\frac{5}{1023} \approx 4.89 \text{ mV}. \tag{6}$$

The amplified differential voltage can be estimated from the difference between the INA125P output and its reference:

$$\Delta V_O = V_O - V_{\text{IAREF}}. \tag{7}$$

The Arduino then processes the signal digitally and sends the values to the computer through the USB serial connection. The INA125P amplification is necessary because the voltage produced directly by the strain-gauge bridge is generally too small to be measured accurately with the Arduino Uno 10-bit ADC.

3 Simulations with Falstad

Before building the circuit, I simulated its basic operation with Falstad. This online simulator makes it possible to create a digital version of the circuit and observe the main voltages.

Falstad does not include the INA125P model, so I used a standard operational-amplifier differential stage to represent the amplification function. The Wheatstone bridge was modelled with four 350 Ω resistors.

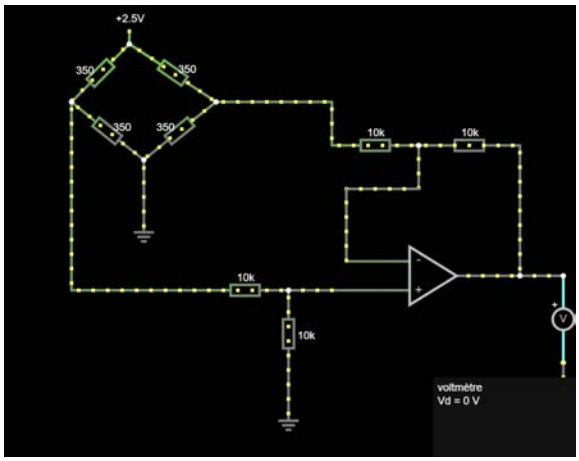


Figure 9: Simulation of the balanced bridge

When the structure is not strained, the four bridge resistances remain equal. The simulation shows that the differential output voltage is then close to zero.

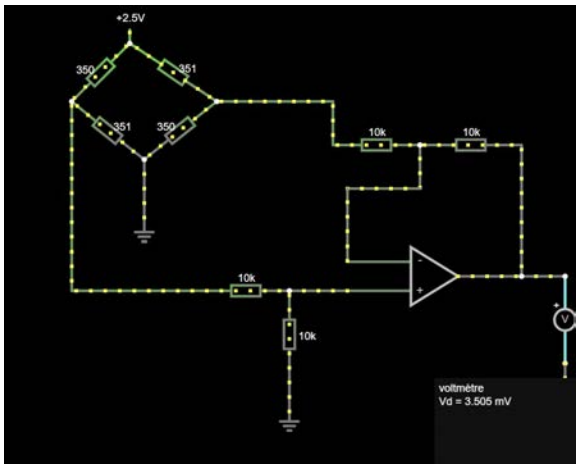


Figure 10: Simulation of the bridge under strain

When strain is introduced by changing the bridge resistances, the bridge becomes unbalanced and a non-zero output voltage appears. This confirms the expected operating principle of the measurement system.

The Falstad study therefore validated the basic behaviour before the laboratory tests. However, the simulation uses ideal components and a simplified amplifier instead of the real INA125P. Experimental tests were still necessary to observe practical effects such as noise, resistor tolerances and amplifier saturation.

4 Equipment and Experimental Setup

4.1 Equipment Used

The tests were carried out with the following equipment:

- metal bar;
- two pairs of strain gauges already wired as full Wheatstone bridges;
- one **INA125P** instrumentation amplifier, used to excite the bridge and amplify its small differential voltage;
- one **Arduino Uno**, used to acquire analog voltages and send data to the computer;
- a Dewesoft acquisition system with **DewesoftX** software, used as a reference for comparison with the Arduino measurements;
- one breadboard;
- jumper wires and connection cables;
- a set of resistors with different values, used to build and adjust the different Wheatstone bridge circuits and INA125P gain configurations;
- several individual strain gauges, used to build and test a quarter-bridge configuration;
- one computer with the Arduino IDE, used to program the board and display data in the Serial Monitor or Serial Plotter.

The Arduino Uno was used instead of the Arduino MKR WiFi 1010 that was planned at the beginning of the project. The MKR WiFi 1010 could not be connected correctly to the computer, so remote Wi-Fi transmission and display were not implemented during this stage of the project.

5 Experimental Work

5.1 Strain Gauge Installation

The first experimental step was to bond the strain gauges to the metal bar. Two pairs of gauges were available, already wired together as Wheatstone bridges.

Before bonding, the metal surface was carefully cleaned to remove dust, grease and other contamination that could reduce adhesion. The gauge was aligned in the required direction with adhesive tape. A thin layer of strong adhesive was then applied and the gauge was positioned on the prepared area. After bonding, adhesive tape was placed over the gauge to protect it from mechanical damage during handling.

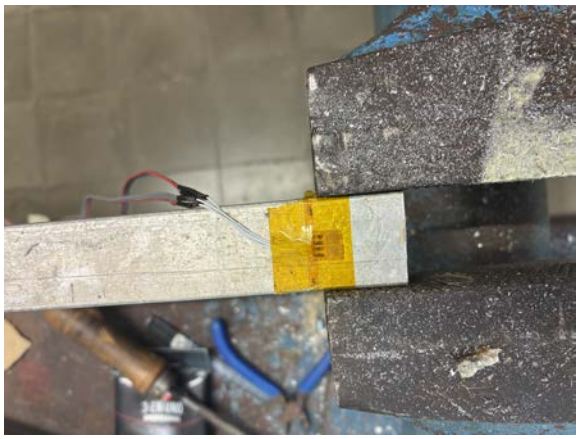


Figure 11: Bonding of the strain gauges

5.2 Measurements with DewesoftX

After the strain-gauge bridge was bonded to the metal bar, a first functional test was carried out with DewesoftX. The system can excite the bridge, measure its output in mV/V and convert this value into strain in μm/m. The `ZERO` function can also remove the initial offset from the displayed signal before a load is applied. These measurements were used as a reference for the later INA125P and Arduino tests.

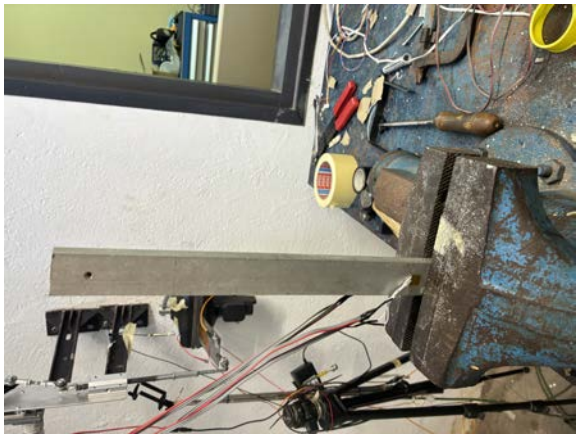


Figure 12: First test with DewesoftX

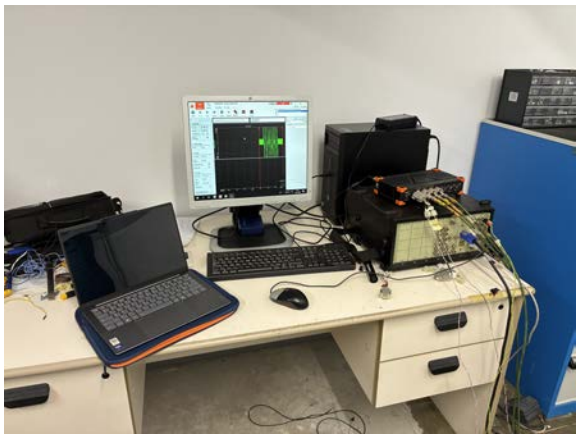


Figure 13: DewesoftX workstation

For this test, the bar was placed vertically and clamped to the workbench with a vice. The gauges were located

close to the clamped area, where the bending stress is high. A manual force was applied at the opposite end of the bar.

The following response was observed on the computer:

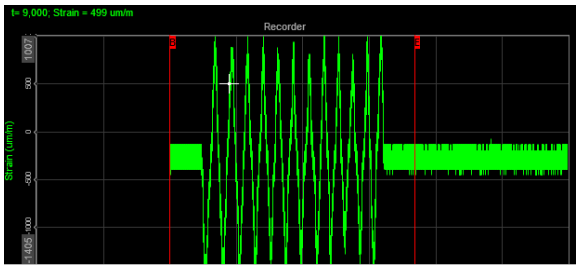


Figure 14: DewesoftX strain results

The oscillations caused by repeated bending in both directions were clearly visible. The measured values were approximately between $+1007$ μm/m and -1045 μm/m. This first test showed that the bridge reacted to bending and provided a useful reference before testing the low-cost measurement chain.

5.3 INA125P and Arduino Uno Wiring

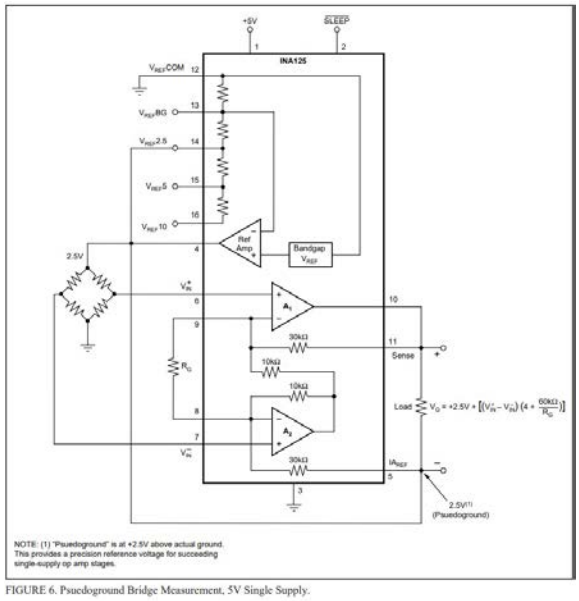


FIGURE 6. Psuedoground Bridge Measurement, 5V Single Supply.

Figure 15: INA125P wiring

The INA125P wiring followed the configuration recommended in the datasheet. The `SLEEP` pin was connected to the 5 V supply so that the amplifier remained active.

The INA125P was powered from the Arduino Uno 5 V supply and used a 2.5 V reference. The amplified output V_O was connected to Arduino input `A0`. The $IAREF$ voltage was measured on `A1`. The bridge input voltages were also measured directly: V_{IN+} was connected to `A2` and V_{IN-} to `A3`. This made it possible to monitor both the amplifier output and its input voltages, which was useful for checking the circuit and diagnosing bridge imbalance.

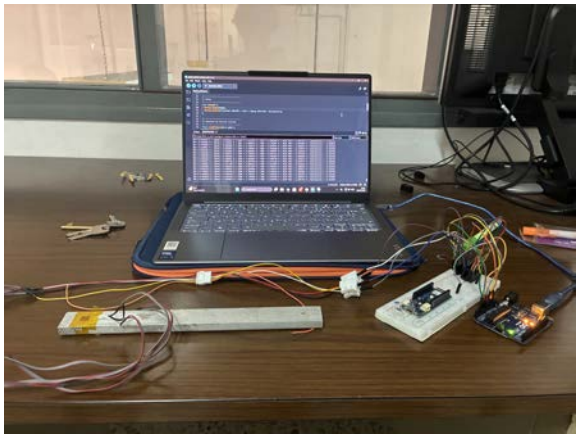


Figure 16: Complete INA125P and Arduino wiring

5.4 Arduino Code and Filtering

The signals from strain gauges are very small and can easily be disturbed by electrical noise. The INA125P amplifies the useful signal, but some disturbances can also be amplified. Digital processing is therefore useful to make the measurement more stable. In this project, averaging and low-pass filtering were tested to reduce noise and electrical interference.

First Test Code A first program was developed mainly to check the circuit and observe the different voltages. The Arduino measures the INA125P output on `A0`, the reference voltage on `A1`, and V_{IN+} and V_{IN-} on `A2` and `A3`.

To stabilise the measurement, each displayed value is the average of 50 successive ADC readings:

$$V_{\text{avg}} = \frac{1}{N}\sum_{i=1}^{N} V_i \tag{8}$$

with $N = 50$. The program also calculates $V_{IN+} - V_{IN-}$ to check bridge balance, and $V_O - V_{IAREF}$ to observe the amplified differential signal. In this first program, this last value is called "deformation", but it is still a voltage and not yet a strain value in μm/m.

See Appendix 1.

Full-Bridge Measurement Code A second, more complete program was developed to measure the full bridge and convert the signal into strain. With $R_G = 660\ \Omega$, the INA125P gain is about 95. The program also uses a gauge factor of 2.

The internal acquisition rate is 100 Hz. For each voltage, 16 ADC conversions are averaged. The first reading after changing the ADC channel is ignored so that the input can stabilise. An automatic tare is then calculated from 300 samples at start-up while the bar is unloaded. The measured signal is processed with a moving average over 10 values and then with an exponential low-pass filter using $\alpha = 0.12$. The program displays both a raw value and a filtered value in μm/m and also checks whether the INA125P output is close to saturation.

See Appendix 2.

5.5 Initial Circuit Check

The bar was clamped in the laboratory in the same way as during the DewesoftX test and was manually bent.

The circuit was first tested with the diagnostic code. The measured values were clearly abnormal: $IAREF$ was about 2.46 V, while the INA125P output V_O was only about 0.06 V. The bridge inputs were also strongly unbalanced, with approximately $V_{IN+} = 1.23$ V and $V_{IN-} = 2.50$ V. The differential input was therefore:

$$V_{IN+} - V_{IN-} = 1.23 - 2.50 = -1.27\ \text{V}. \tag{9}$$

With $R_G = 660\ \Omega$, the INA125P gain is:

$$G = 4 + \frac{60000}{660} \approx 94.9. \tag{10}$$

Using the amplifier output relation:

$$V_O = V_{IAREF} + G(V_{IN+} - V_{IN-}), \tag{11}$$

the theoretical requested output would be:

$$V_O \approx 2.46 + 94.9 \times (-1.27) \approx -118 \text{ V}. \tag{12}$$

This voltage is impossible because the INA125P is powered between 0 and 5 V. The amplifier therefore reaches its lower output limit and saturates. The measured value near 0.06 V no longer represents the bridge differential voltage correctly. Saturation was also observed with a lower gain using $R_G = 5.1$ kΩ.

The more complete strain program gave the same general result: very large and inconsistent strain values, with the INA125P output close to its lower limit. Because two different programs showed the same problem, the error was unlikely to come mainly from the digital processing. The bridge or its electrical connections therefore became the main suspect.

The tests also showed that the breadboard circuit was sensitive to noise. Touching or moving wires could create sudden changes in the measured values. During measurements, the circuit therefore had to remain as still as possible before the values were interpreted.

5.6 INA125P Diagnostic

To isolate the problem, the INA125P was tested with an artificial Wheatstone bridge made from four 220 Ω resistors. This replaced the strain gauges with known resistors and allowed the amplifier to be tested independently. The Arduino code from Appendix 1 was used.

The first test was performed without an external R_G resistor. In this configuration, the INA125P gain is 4 and the measured output was close to 2.5 V, as expected for a nearly balanced bridge.

A 660 Ω resistor was then connected as R_G, giving:

$$G = 4 + \frac{60000}{660} \approx 94.9. \tag{13}$$

The output increased to about 3.0 V. The two INA125P inputs were approximately 1.271 V and 1.266 V, giving a bridge imbalance of about 5 mV. With a gain close to 95, this small difference produces an output change of approximately:

$$95 \times 5 \text{ mV} \approx 0.48 \text{ V}. \tag{14}$$

This explains the change from an output close to 2.5 V to about 3.0 V. The small imbalance came from the tolerance of the 220 Ω resistors. This test confirmed that the INA125P correctly amplified the differential input and was not the main cause of the problem.

5.7 Strain-Gauge Bridge Diagnostic

After checking the INA125P, the strain-gauge bridge was disconnected from the amplifier and its resistances were measured directly. A simple Arduino ohmmeter was built with a voltage divider and a known resistance of about 990 Ω, made from three 330 Ω resistors in series.

The unknown resistance was calculated from:

$$R_{\text{measured}} = R_{\text{known}} \frac{V_{\text{measured}}}{V_{\text{supply}} - V_{\text{measured}}} \tag{15}$$

The code used for this test is given in Appendix 3.

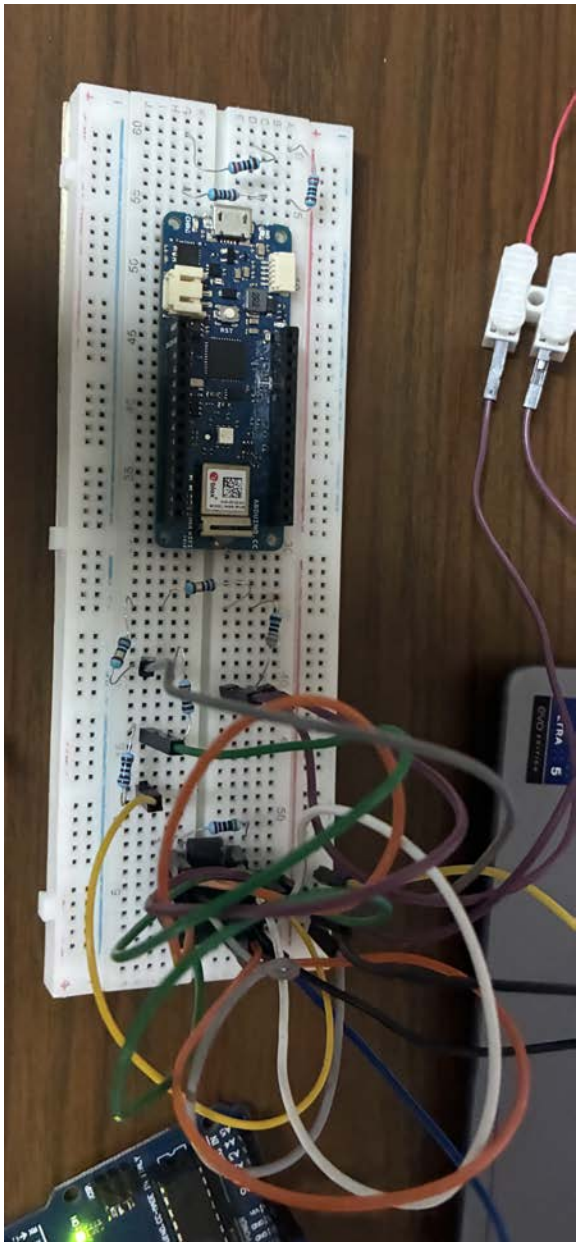


Figure 17: Resistance measurement test

All six combinations between $EX+$, $EX-$, $SIG+$ and $SIG-$ were measured. Here, SIG represents the bridge signal terminals and EX represents the excitation terminals. The results for the first bridge were:

Measured terminals	**Measured resistance**
$EX+$ – $EX-$	$\approx 700\ \Omega$
$EX+$ – $SIG+$	$\approx 350\ \Omega$
$EX+$ – $SIG-$	$\approx 350\ \Omega$
$EX-$ – $SIG+$	$\approx 350\ \Omega$
$EX-$ – $SIG-$	$\approx 1050\ \Omega$
$SIG+$ – $SIG-$	$\approx 700\ \Omega$

Because one gauge has a nominal resistance of about 350 Ω, the 1050 Ω value between $EX-$ and $SIG-$ is not normal. It corresponds approximately to three 350 Ω resistances in series:

$$350 + 350 + 350 = 1050\ \Omega. \tag{16}$$

This means that current cannot pass directly through the branch between $EX-$ and $SIG-$ and must travel through the other three branches. The bridge therefore contains an open branch. The fault may come from a damaged gauge, solder joint, track or connection.

The same resistance measurements were then carried out on a second bridge. This time, the values matched the expected behaviour of a full bridge made from four 350 Ω gauges. This comparison confirmed that the first bridge had a permanent electrical fault.

Because the bridge bonded to the metal bar could no longer be used, another method was needed to continue the strain measurements.

5.8 Quarter-Bridge Construction and Test

A quarter-bridge circuit was therefore built using a single strain gauge that was available. Its resistance was first checked with the Arduino ohmmeter from Appendix 3 and measured about 349.8 Ω, confirming that the gauge was electrically continuous.

The gauge was then bonded to a thin metal plate using the same procedure as before.

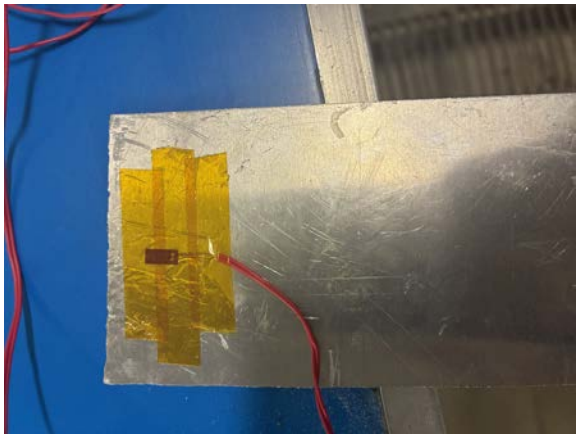


Figure 18: Bonding of the single strain gauge

The gauge was integrated into a quarter bridge and connected to the INA125P and Arduino Uno. Because only certain resistor values were available, an equivalent resistance of about 350 Ω was made with one 330 Ω resistor and two 10 Ω resistors in series. The other bridge branch used two 1 kΩ resistors so that both branches had a ratio close to 1/2. The bridge was excited with the INA125P 2.5 V reference.

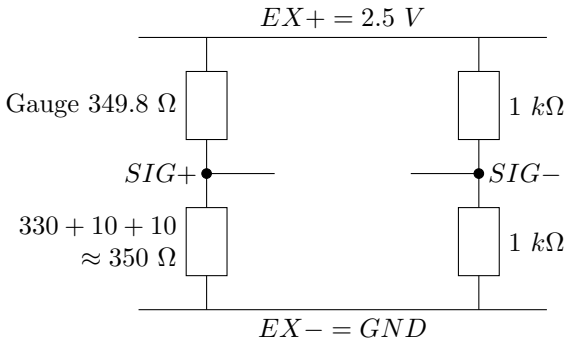


Figure 19: Quarter-bridge circuit using one strain gauge

A gain resistor $R_G = 5.1$ kΩ was used with the INA125P, giving a gain of about 15.8. A specific Arduino program for the quarter bridge was used (Appendix 4). It measures V_O, $IAREF$ and $EX+$, then calculates the bridge ratio and converts it into strain using the INA125P gain and a gauge factor of 2.

Before each test, the circuit was left at rest for a few seconds. A zero was made automatically at start-up with no load on the plate. A new zero could also be made manually by sending the letter `z` in the Serial Monitor. During this step, the plate, gauge and wires had to remain untouched.

The program uses a simple ADC stabilisation method: after changing analog channels, the first conversion is ignored and a second reading is taken after 50 μs. Unlike the previous full-bridge program, this quarter-bridge code does not use a moving average or a digital low-pass filter, so the measurements remain more sensitive to noise and wire movement.

For the bending test, one end of the thin metal plate was fixed to a table with a clamp. After zeroing the system, the free end of the plate was bent manually in both directions while the values were observed in the Serial Monitor.

The following representative values were obtained:

Condition	V_O (V)	Strain (μm/m)
Before zeroing	2.081	+250.7
After zeroing	2.087	0.0
Plate at rest	2.087	0.0
Bending in one direction	2.072	+752.2
Bending in the opposite direction	2.107	-1002.9

Table 1: Example values obtained during the quarter-bridge bending test

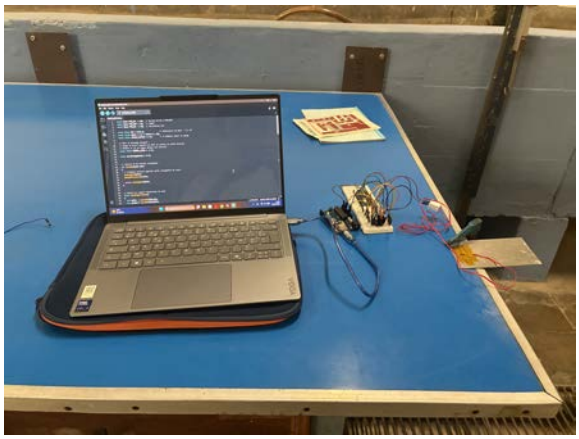


Figure 20: Quarter-bridge experimental setup

After zeroing, the measurement returned close to 0 μm/m when the plate was at rest. During bending, the measured strain became positive in one direction and negative in the other. This shows that the measurement chain can detect the direction of mechanical strain.

6 Results, Discussion and Limitations

The different tests made it possible to check the measurement chain step by step. The INA125P was first validated with an artificial bridge made from four 220 Ω resistors. The resistance test then identified an open branch in the first strain-gauge bridge. A second bridge gave normal resistance values and confirmed the diagnosis.

To continue the work, a single 349.8 Ω gauge was used in a quarter bridge. During the manual bending test, the output changed sign according to the bending direction, with representative values of about +752 μm/m and −1000 μm/m. This result validates the system qualitatively.

Short Comparison with DewesoftX

DewesoftX provided a cleaner and more direct reference measurement, with clear strain oscillations during bending and an integrated `ZERO` function. The low-cost INA125P–Arduino system also detected the direction of bending, but it was more sensitive to electrical imbalance, noise and ADC resolution. The DewesoftX zero can centre the displayed signal, but it does not repair an electrical fault in the bridge. For this reason, resistance measurements were still necessary to identify the damaged branch.

Step	Method	Main result
Reference test	DewesoftX	Bridge response visible during bending
Amplifier check	Artificial 220 Ω bridge	INA125P operation validated
Bridge diagnosis	Arduino resistance measurement	Open branch found between $EX-$ and $SIG-$
Final test	Single-gauge quarter bridge	Positive and negative response during bending

Table 2: Summary of the experimental approach

The main limitations are the electrical noise and the 10-bit resolution of the Arduino Uno. The final quarter-bridge program did not use averaging or digital filtering, and moving the wires could strongly change the readings. The measured strain also appeared in steps, for example around 250.7 μm/m, because one ADC count represented a relatively large change in calculated strain.

The quarter bridge also had an initial electrical imbalance. The software zero removed this offset from the displayed result but did not physically rebalance the bridge. Resistor tolerances probably contributed to this offset.

Finally, no calibration with a known load or known strain could be completed. The bending force was applied manually and its exact value was not known. Therefore, the strain values should not be considered accurate quantitative measurements. They mainly demonstrate that the complete measurement chain reacts correctly to bending.

7 Conclusion

This project made it possible to build and test a low-cost strain measurement chain using strain gauges, a Wheatstone bridge, an INA125P instrumentation amplifier and an Arduino Uno.

The different tests first confirmed the operation of the INA125P and then identified an electrical fault in one strain-gauge bridge. The Arduino ohmmeter was especially useful for locating an open branch. A quarter bridge with a single gauge was then built so that testing could continue. The bending tests showed a clear change in the signal and a change of sign with the bending direction, which qualitatively validated the measurement principle.

The project therefore shows that a simple and inexpensive system can detect strain changes. Several improvements would still be required for reliable quantitative measurements: precision resistors, better wiring, digital and analog filtering, a higher-resolution ADC, calibration with a known load and, in a future version, a Wi-Fi board for remote monitoring. In addition to the measurement system itself, the work also produced a useful diagnostic method for separating problems caused by the sensor, the amplifier or the acquisition system.

Appendices

Appendix 1 – Diagnostic code and voltage readings

This first program measures V_O, $IAREF$, V_{IN+} and V_{IN-} and averages 50 ADC readings.

```
const int pinVo   = A0;
const int pinRef  = A1;
const int pinVinP = A2;
const int pinVinM = A3;

const int N = 50;

void setup() {
  Serial.begin(9600);
  Serial.println("INA125P + Bridge + VIN+/VIN- debug system initialized");
}

float readFiltered(int pin) {
  long s = 0;

  for (int i = 0; i < N; i++) {
    s += analogRead(pin);
  }

  return (s / (float)N) * (5.0 / 1023.0);
}

void loop() {

  float vVo   = readFiltered(pinVo);
  float vRef  = readFiltered(pinRef);
  float vVinP = readFiltered(pinVinP);
  float vVinM = readFiltered(pinVinM);

  float deformation = vVo - vRef;
  float diffVin = vVinP - vVinM;

  Serial.print("VO = ");
  Serial.print(vVo, 6);

  Serial.print(" V | IAref = ");
  Serial.print(vRef, 6);

  Serial.print(" V | VIN+ = ");
  Serial.print(vVinP, 6);

  Serial.print(" V | VIN- = ");
  Serial.print(vVinM, 6);

  Serial.print(" V | Diff VIN = ");
  Serial.print(diffVin, 6);

  Serial.print(" V | Strain = ");
  Serial.println(deformation, 6);

  delay(200);
}
```

Appendix 2 – Full-bridge measurement code with filtering

This program converts the full-bridge signal into microstrain. It uses ADC averaging, automatic tare, a moving average and an exponential low-pass filter.

```
#include <Arduino.h>

const byte PIN_VO  = A0;
const byte PIN_REF = A1;
const byte PIN_EXC = A2;

const float RG_OHMS = 660.0f;
const float GAIN_INA = 4.0f + 60000.0f / RG_OHMS;

const float GAUGE_FACTOR = 2.0f;
const float BRIDGE_FACTOR = 1.0f;
const float SIGNAL_SIGN = 1.0f;
const float CALIBRATION_FACTOR = 1.0f;

const unsigned long SAMPLE_PERIOD_US = 10000UL;
const unsigned long DISPLAY_PERIOD_MS = 200UL;

const byte ADC_AVERAGES = 16;
const byte MOVING_AVERAGE_SIZE = 10;
const float FILTER_ALPHA = 0.12f;

const int TARE_SAMPLES = 300;

const float ADC_LOW_LIMIT  = 60.0f;
const float ADC_HIGH_LIMIT = 780.0f;

float tareMicrostrain = 0.0f;

float movingBuffer[MOVING_AVERAGE_SIZE] = {0};
float movingSum = 0.0f;
byte movingIndex = 0;
byte movingCount = 0;

float filteredMicrostrain = 0.0f;

unsigned long nextSampleTime = 0;
unsigned long lastDisplayTime = 0;

bool saturationDetected = false;


float readAveragedADC(byte pin)
{
  analogRead(pin);

  unsigned long sum = 0;

  for (byte i = 0; i < ADC_AVERAGES; i++)
  {
    sum += analogRead(pin);
  }

  return sum / (float)ADC_AVERAGES;
}


float readMicrostrain()
{
  float adcVo  = readAveragedADC(PIN_VO);
  float adcRef = readAveragedADC(PIN_REF);
  float adcExc = readAveragedADC(PIN_EXC);

  saturationDetected =
      adcVo < ADC_LOW_LIMIT ||
```

```
      adcVo > ADC_HIGH_LIMIT ||
      adcExc < 50.0f;

  if (adcExc < 1.0f)
  {
    return 0.0f;
  }

  float bridgeRatio =
      (adcVo - adcRef) / (GAIN_INA * adcExc);

  float microstrain =
      SIGNAL_SIGN *
      CALIBRATION_FACTOR *
      bridgeRatio *
      1000000.0f /
      (GAUGE_FACTOR * BRIDGE_FACTOR);

  return microstrain;
}


float updateMovingAverage(float newValue)
{
  if (movingCount < MOVING_AVERAGE_SIZE)
  {
    movingBuffer[movingIndex] = newValue;
    movingSum += newValue;
    movingCount++;
  }
  else
  {
    movingSum -= movingBuffer[movingIndex];
    movingBuffer[movingIndex] = newValue;
    movingSum += newValue;
  }

  movingIndex++;

  if (movingIndex >= MOVING_AVERAGE_SIZE)
  {
    movingIndex = 0;
  }

  return movingSum / movingCount;
}


void performTare()
{
  digitalWrite(LED_BUILTIN, HIGH);

  float sum = 0.0f;
  int validSamples = 0;

  for (int i = 0; i < TARE_SAMPLES; i++)
  {
    unsigned long startTime = micros();

    float value = readMicrostrain();

    if (!saturationDetected)
    {
      sum += value;
      validSamples++;
    }

    while ((unsigned long)(micros() - startTime)
           < SAMPLE_PERIOD_US)
```

```
    {
    }
  }

  if (validSamples > 0)
  {
    tareMicrostrain = sum / validSamples;
  }
  else
  {
    tareMicrostrain = 0.0f;
  }

  digitalWrite(LED_BUILTIN, LOW);
}


void setup()
{
  Serial.begin(9600);

  pinMode(LED_BUILTIN, OUTPUT);
  digitalWrite(LED_BUILTIN, LOW);

  delay(1500);

  performTare();

  filteredMicrostrain = 0.0f;

  nextSampleTime = micros();
  lastDisplayTime = millis();
}


void loop()
{
  unsigned long nowMicros = micros();

  if ((long)(nowMicros - nextSampleTime) >= 0)
  {
    nextSampleTime += SAMPLE_PERIOD_US;

    float rawMicrostrain =
        readMicrostrain() - tareMicrostrain;

    float averagedMicrostrain =
        updateMovingAverage(rawMicrostrain);

    filteredMicrostrain +=
        FILTER_ALPHA *
        (averagedMicrostrain - filteredMicrostrain);

    digitalWrite(
        LED_BUILTIN,
        saturationDetected ? HIGH : LOW
    );

    if (millis() - lastDisplayTime >= DISPLAY_PERIOD_MS)
    {
      lastDisplayTime += DISPLAY_PERIOD_MS;

      Serial.print("Raw_ue:");
      Serial.print(rawMicrostrain, 2);

      Serial.print("\tFiltered_ue:");
      Serial.print(filteredMicrostrain, 2);

      Serial.print("\tZero:");
```

```
203         Serial.println(0);
204       }
205     }
206 }
```

Appendix 3 – Arduino resistance measurement code

This program uses the Arduino as an ohmmeter with a voltage divider and a known 990 Ω resistor.

```
const byte pinMesure = A0;

const float resistanceConnue = 990.0;
const int nombreMesures = 200;

float lireMoyenne()
{
  long somme = 0;

  analogRead(pinMesure);

  for (int i = 0; i < nombreMesures; i++)
  {
    somme += analogRead(pinMesure);
  }

  return somme / (float)nombreMesures;
}

void setup()
{
  Serial.begin(9600);
}

void loop()
{
  float adc = lireMoyenne();

  Serial.print("ADC = ");
  Serial.print(adc, 2);

  if (adc < 2)
  {
    Serial.println(
      " | Very low resistance or short circuit"
    );
  }
  else if (adc > 1020)
  {
    Serial.println(
      " | Open circuit or broken wire"
    );
  }
  else
  {
    float resistanceInconnue =
        resistanceConnue * adc / (1023.0 - adc);

    Serial.print(" | Equivalent resistance = ");
    Serial.print(resistanceInconnue, 1);
    Serial.println(" ohms");
  }

  delay(500);
}
```

Appendix 4 – Quarter-bridge measurement code

This final program is used for bending tests with the single gauge in a quarter bridge. It performs zeroing and directly converts the signal into strain in μm/m.

```
const byte PIN_VO  = A0;
const byte PIN_REF = A1;
const byte PIN_EX  = A2;

const float RG = 5100.0;
const float GAIN = 4.0 + 60000.0 / RG;
const float GAUGE_FACTOR = 2.0;
const float SIGNAL_SIGN = -1.0;

float zeroBridgeRatio = 0.0;

int lireADC(byte pin)
{
  analogRead(pin);
  delayMicroseconds(50);

  return analogRead(pin);
}

float lireRapportPont()
{
  int adcVo  = lireADC(PIN_VO);
  int adcRef = lireADC(PIN_REF);
  int adcEx  = lireADC(PIN_EX);

  if (adcEx == 0)
  {
    return 0.0;
  }

  return (adcVo - adcRef) / (GAIN * adcEx);
}

void faireZero()
{
  delay(1000);
  zeroBridgeRatio = lireRapportPont();
}

void setup()
{
  Serial.begin(9600);

  delay(3000);

  faireZero();

  Serial.println("Measurement␣started");
  Serial.println("Send␣z␣to␣set␣a␣new␣zero");
}

void loop()
{
  int adcVo  = lireADC(PIN_VO);
  int adcRef = lireADC(PIN_REF);
  int adcEx  = lireADC(PIN_EX);

  float vo  = adcVo  * 5.0 / 1023.0;
  float ref = adcRef * 5.0 / 1023.0;
  float ex  = adcEx  * 5.0 / 1023.0;

  float bridgeRatio =
      (adcVo - adcRef) / (GAIN * (float)adcEx);

```

```
  float deltaBridgeRatio =
      bridgeRatio - zeroBridgeRatio;

  float deformation =
      SIGNAL_SIGN *
      4.0 *
      deltaBridgeRatio *
      1000000.0 /
      GAUGE_FACTOR;

  Serial.print("VO=");
  Serial.print(vo, 3);

  Serial.print("␣V␣|␣IAREF=");
  Serial.print(ref, 3);

  Serial.print("␣V␣|␣EX+=");
  Serial.print(ex, 3);

  Serial.print("␣V␣|␣Strain=");
  Serial.print(deformation, 1);

  Serial.print("␣um/m");

  if (vo < 0.3 || vo > 3.8)
  {
    Serial.print("␣|␣WARNING:␣SATURATION");
  }

  Serial.println();

  if (Serial.available() > 0)
  {
    char commande = Serial.read();

    if (commande == 'z' || commande == 'Z')
    {
      faireZero();
      Serial.println("New␣zero␣set");
    }
  }

  delay(500);
}
```