Politecnico di Torino

Testing and Certification

Sixth Laboratory Session

Implementation of a Digital Massmeter using an Arduino UNO Board and CZL635 Load Cell

Authors:
D'Addato Mauro
Fini Simone
Marmo Sebastiano
Triarico Antonio

Professor:
Carullo Alessio

18 December 2017

POLITECNICO
DI TORINO

1 Introduction

This laboratory session aimed to implement a mass meter which allowed to measure, with a good grade of uncertainty, a known mass first and an unknown one (represented by some simple bolts) at the end.
This laboratory report will illustrate both the hardware and the software development of the device and the consequent results provided by it.

1.1 Instrumentation

The instruments used to perform the described task are:

- Arduino UNO Board;
- CZL635 Load Cell;
- INA126 Instrumentation Amplifier;
- two $0.1\mu F$ capacitors;
- 18Ω resistor;
- $1k\Omega$ resistor;
- $6.8k\Omega$ resistor;
- $461.0 \pm 0.1\ g$ known mass;
- Philips PE1542 DC Power Supply;
- HP 34401A Multimeter (to measure the correct value of the components).

Accuracy Specifications ± (% of reading + % of range) [1]

Function	Range [3]	Test Current or Burden Voltage	24 Hour [2] 23°C ± 1°C	90 Day 23°C ± 5°C	1 Year 23°C ± 5°C	Temperature Coefficient /°C 0°C – 18°C 28°C – 55°C
DC Voltage	100.0000 mV		0.0030 + 0.0030	0.0040 + 0.0035	0.0050 + 0.0035	0.0005 + 0.0005
	1.000000 V		0.0020 + 0.0006	0.0030 + 0.0007	0.0040 + 0.0007	0.0005 + 0.0001
	10.00000 V		0.0015 + 0.0004	0.0020 + 0.0005	0.0035 + 0.0005	0.0005 + 0.0001
	100.0000 V		0.0020 + 0.0006	0.0035 + 0.0006	0.0045 + 0.0006	0.0005 + 0.0001
	1000.000 V		0.0020 + 0.0006	0.0035 + 0.0010	0.0045 + 0.0010	0.0005 + 0.0001
Resistance [4]	100.0000 Ω	1 mA	0.0030 + 0.0030	0.008 + 0.004	0.010 + 0.004	0.0006 + 0.0005
	1.000000 kΩ	1 mA	0.0020 + 0.0005	0.008 + 0.001	0.010 + 0.001	0.0006 + 0.0001
	10.00000 kΩ	100 µA	0.0020 + 0.0005	0.008 + 0.001	0.010 + 0.001	0.0006 + 0.0001
	100.0000 kΩ	10 µA	0.0020 + 0.0005	0.008 + 0.001	0.010 + 0.001	0.0006 + 0.0001
	1.000000 MΩ	5 µA	0.002 + 0.001	0.008 + 0.001	0.010 + 0.001	0.0010 + 0.0002
	10.00000 MΩ	500 nA	0.015 + 0.001	0.020 + 0.001	0.040 + 0.001	0.0030 + 0.0004
	100.0000 MΩ	500 nA // 10 MΩ	0.300 + 0.010	0.800 + 0.010	0.800 + 0.010	0.1500 + 0.0002

Fig. 1: Specifications, used to compute the errors on the various measurements, provided by the DMM datasheet

2 Theoretical Approach and Hardware Design

First of all, the CZL635 is a load cell based on a full bridge circuit composed of four strain gauges in push-pull configuration; the main problem with this sensor was about the voltage (proportional to the mass over it) it produced: indeed, since the Arduino resolution is 5mV (using its default reference of 5V), it was not able to properly detect changes in the sensor output, which was included between $10\mu V$ and 1mV for the range 10g $\div$ 1000g.

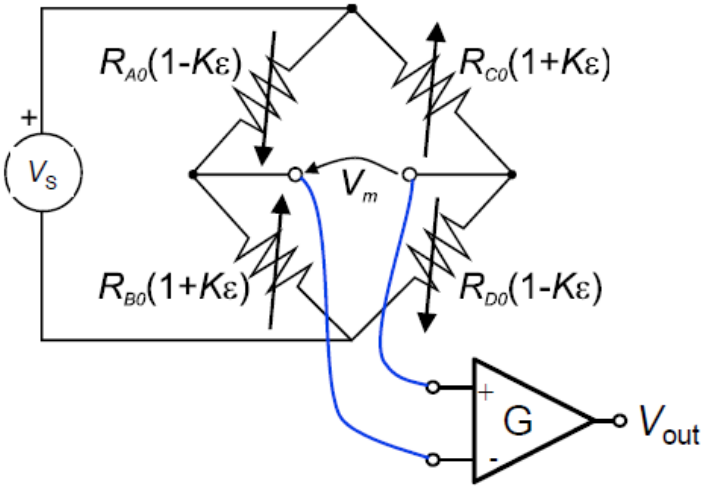

Fig. 2: Circuit design

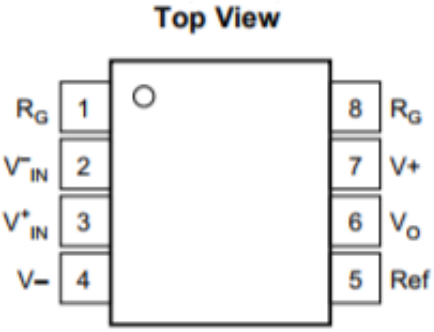

Fig. 3: External top view of the INA 126

For this reason, the signal had to be amplified in order to well fit the specifications of the UNO board. In the case of this experiment, an instrumentation amplifier (INA126) was used (the circuit design is showed by Fig.2) and the current relationship between the output of this device and the magnitude under measurement was:

$$V_{out} = G \cdot V_m = G \cdot S \cdot V_s \cdot m, \tag{1}$$

where

$$G = 5 + \frac{80k\Omega}{R_G}. \tag{2}$$

Having found the aformentioned formula (which was crucial during the software development, as it will be described in section 2), the following task to perform was to make the amplifier work correctly, starting from its power supply.
In fact, as its datasheet describes, the nominal values of positive and negative voltage to be applied on pins number 7 and 4, respectively, is $\pm 15V$.

6.3 Recommended Operating Conditions

over operating free-air temperature range (unless otherwise noted)

		MIN	NOM	MAX	UNIT
V+	V power supply	±135	±15	±18	V

Fig. 4: Recommended operating conditions taken from the original INA126 datasheet

So, a specific connection of the DC power supply was implemented to satisfy this requirement: two independent generators, the first one providing $14.8911 \pm 0.0013V$ and the second one $14.7338 \pm 0.0013V$, were connected in series, and between the positive output of the former and the negative of the latter the GND was connected (look at Fig.5 for a better comprehension).

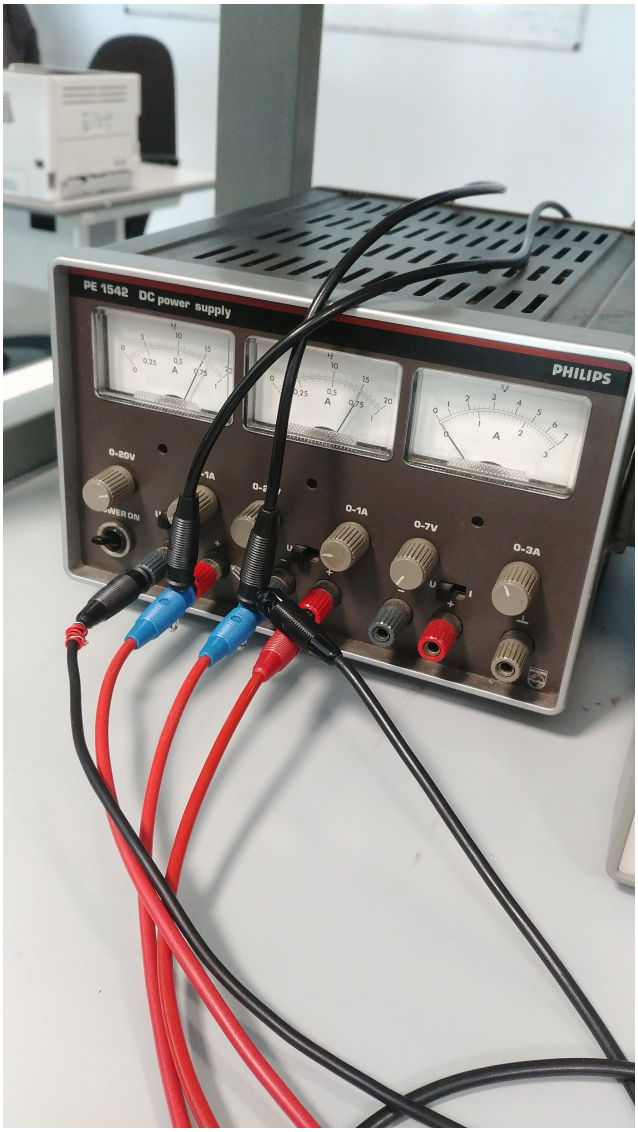

Fig. 5: Connection of the two independent power supplies

In this way, the cable on the right would have provided $14.8911 \pm 0.0013V$, while the one on the left $-14.7338 \pm 0.0013V$.
Once having arranged the power supply, it had to be connected to the amplifier with all the other components; specifically:

- the 18Ω was connected between pin number 1 and 8; in this way the complex gain would have been 4449 (actually, the real value of the resistor was $17.8080 \pm 0.0118\Omega$). Having a gain value like that, the relation

$$V_{out} \leq V_{FR} \tag{3}$$

  was largely respected, since the full scale range of the ADC was the 5V provided by the Arduino board;

- all the signal of interest, such as $V_{in+}, V_{in-}, V_{out}$;

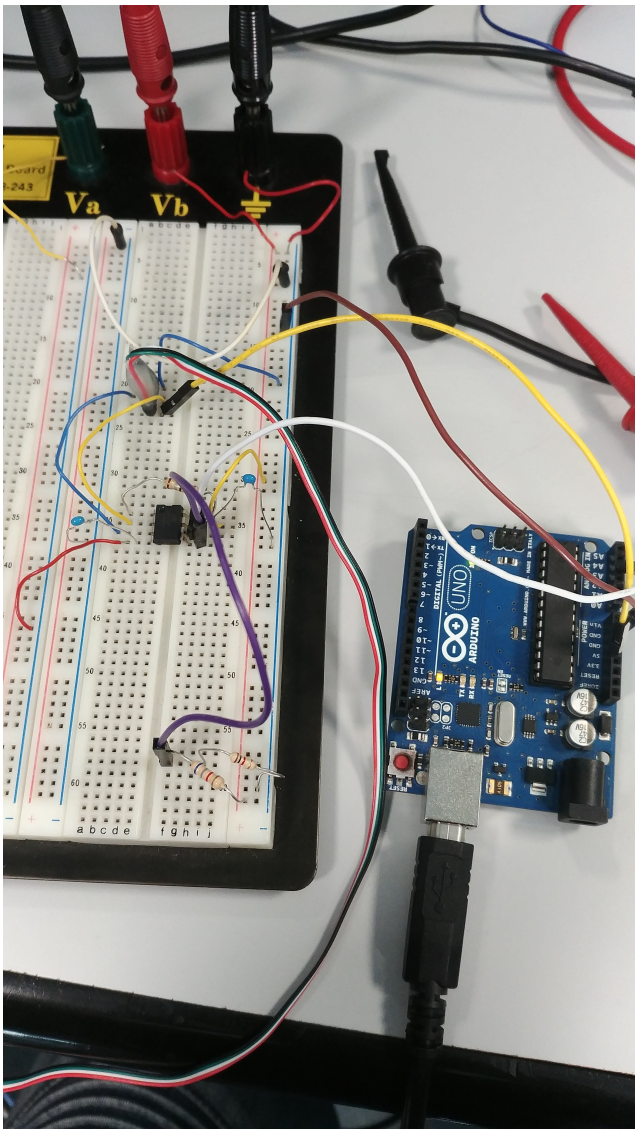

Fig. 6: Implementation of the physical circuit

- V_{ref}, which was used to compensate the offset error of the sensor. Indeed, without having anything on the sensor plate, the initial measured output of the amplifier was $-1.50361 \pm 0.00010V$, that would have literally burnt the electronic board. For this reason, it was chosen to compensate that value via hardware first (by means of a voltage divider in order to make the output voltage positive) and then via software.
  In particular, the divider was built using a resistance of $6.71332 \pm 0.10067k\Omega$ and another one of $0.98530 \pm 0.10009k\Omega$, and, applying voltage provided by the power supply, the result was $1.91954 \pm 0.00012V$.

Having arranged the entire circuit, composed of the sensor and the related conditioning circuit, the following thing to do was to write the firmware code.

3 Firmware Design

```
#define CALIBRATED
//#define FIND_CALIBRATION_OFFSET
//#define FIND_CALIBRATION_K

int N_bits = 10;
double S = 1e-6;
double output;
double G = 5 + (80000/17.808);
double mass;
double ref_mass = 461.0;

double D_offset = 69.0;
double K = 0.95;

void setup() {
  Serial.begin(9600);
  analogReference(DEFAULT);
  pinMode(A3,INPUT);          //input for the amplifier
  Serial.println("Setup completed");
}

void loop() {

  output= analogRead(A3);
  Serial.println(output);
  #ifndef CALIBRATED
    mass = output/(pow(2,N_bits)*S*G);
  #else
    mass = (output - D_offset)/K;
  #endif
  Serial.print("The measured mass is:");
  Serial.println(mass);
  delay(2000);

  //the following segment is about the determination of the real calibration function
  #ifdef FIND_CALIBRATION_OFFSET
  {
  D_offset = output;    //computed with m=0;
  Serial.print("The offset is:");
  Serial.println(D_offset);
  delay(2000);
  }
  #endif
  #ifdef FIND_CALIBRATION_K
  {
  K = (output - D_offset)/ref_mass;
  Serial.print("The measured K is:");
  Serial.println(K);
  delay(2000);
  }
  #endif
}
```

Fig. 7: Code written

The first thing which needs explanation is the *"#define"* section.
The first measure was taken with all the first three lines commented; in that way, the mass would have been affected by the offset error and computed in the "rawest" mode

$$m = \frac{D_{out}}{2^{N_{bits}} \cdot S \cdot G}, \tag{4}$$

where D_{out} is represented by the variable *output*. Having applied the V_{ref} computed previously, the following step was to find the value which would have compensated the offset that, at that point, would have been positive.
In order to do this, the second *#define FIND_CALIBRATION_OFFSET* was uncommented, and the result provided by the part of the code represented by Fig.8 was 69.0, value that was stored into the variable *D_offset*.

```
#ifdef FIND_CALIBRATION_OFFSET
{
D_offset = output;    //computed with m=0;
Serial.print("The offset is:");
Serial.println(D_offset);
delay(2000);
}
#endif
```

Fig. 8: Section of the code providing the offset value

Subsequently, the aformentioned *#define* was commented again and the last one (*#define FIND_CALIBRATION_K*) was uncommented, so as to find the real value of the term

$$K = 2^{N_{bits}} \cdot S \cdot G. \tag{5}$$

In order to complete that task, the known mass was posed on the sensor plate and, applying the formula

$$K = \frac{D_{out} - D_{offset}}{m_{known}}, \tag{6}$$

and after the execution of the portion of code showed by Fig.9, the computed K, whose value was equal to 0.95, was stored into the variable K.

```
#ifdef FIND_CALIBRATION_K
{
K = (output - D_offset)/ref_mass;
Serial.print("The measured K is:");
Serial.println(K);
delay(2000);
}
#endif
```

Fig. 9: Section of the code providing the K value

At the end, as long as the two valued were available, the last *#define* was commented again and the *#define CALIBRATED* was uncommented, in order not to computed the mass by means of the formula (4) but (5)

$$m = \frac{D_{out} - D_{offset}}{K} \tag{7}$$

4 Results

4.1 Measurement without V_ref and software offset

First of all, the unknown mass and the known one were measured without any kind of modification to the system (nor V_{ref} neither software offset were used).
Reminding that:

- the absolute accuracy of the Arduino ADC is $\pm 2LSB$;
- $S = 1\ \frac{mV}{V}$ and $\delta S = \pm 150\ \frac{\mu V}{V}$ at full range;
- $R_G = 17.8080 \pm 0.0118\ \Omega$;

and reminding that the formula used to compute the mass was (4) and the one used to calculate the overall uncertainty was:

$$\delta m = m \cdot \epsilon m = m \cdot \left(\frac{\delta D_{out}}{D_{out}} + \frac{\delta S}{S} + \frac{\delta G}{G} \right), \tag{8}$$

the results provided by the system are showed by Table 1 and Table 2.

ADC Output	Mass (g)	Uncertainty (g)
3.00	0.73	0.60

Table 1: Results regarding the unknown mass (a single bolt)

ADC Output	Mass (g)	Uncertainty (g)
114.00	24.77	4.15

Table 2: Results regarding the known mass

4.2 Measurement with V_ref and software offset

After V_{ref} had been applied and the software offset set, a second measurement was made using the formula (7).
In that case, the uncertainty calculation took into account different parameters, since K was estimated experimentally; in particular (knowing that $D_{out} = 507.00$):

$$\delta K = \left| \frac{\partial K}{\partial D_{out}} \right| \cdot \delta D_{out} + \left| \frac{\partial K}{\partial D_{offset}} \right| \cdot \delta D_{offset} + \left| \frac{\partial K}{\partial m} \right| \cdot \delta m \tag{9}$$

$$\delta m = \left|\frac{\partial m}{\partial D_{out}}\right| \cdot \delta D_{out} + \left|\frac{\partial m}{\partial D_{offset}}\right| \cdot \delta D_{offset} + \left|\frac{\partial m}{\partial K}\right| \cdot \delta K \tag{10}$$

Finally, the results provided are shown by Table 3 and Table 4.

ADC Output	Mass (g)	Uncertainty (g)
82.00	13.68	4.34

Table 3: Results regarding the unknown mass (a single bolt)

ADC Output	Mass (g)	Uncertainty (g)
508.00	462.11	8.53

Table 4: Results regarding the known mass