Treball de Fi de Grau

Grau en Enginyeria en Tecnologies Industrials (GETI)

Development of a test bench for the electronics of ionizing radiation detectors

Author: Salvador Poveda Hospital

Directors: Ariel Tarifeño Saldivia

Alfredo de Blas del Hoyo

Convocatory: 06/2019

Escola Tècnica Superior
d'Enginyeria Industrial de Barcelona

Contents

Abstract

In the present final degree project a low cost test bench is presented. A test bench is an environment used to verify the correctness of devices. In this case, the test bench is used to test preamplifiers and digitizers of the nuclear instrumentation laboratory. These instruments are used for ionizing particle detection.

The initial problematic of the project was to investigate if it was possible to do a testing working bench with a cheap digital circuit as microcontrollers. After a study of the actual market, the Arduino Due was chosen. Arduino is an open-source electronics platform based on easy-to-use hardware and software. In the initial stage of the project, the attention was focused on the documentation about the Arduino boards. It was quickly observed that the sampling time delivered by the Aduino IDE was not acceptable for a nuclear test bech. When inquiring into the technical datasheet of the Atmel SAM3X microchip, used by the Arduino Due, it was achieved making signals with sampling time acceptable for nuclear instrumentation.

Once discovered the potential of Arduino Due, a GUI was made to fully customize the shape of the pulses generated by the Arduino board. Therefore, the development of a cheap testing workbench was achieved.

Chapter 1

Introduction

1.1 Objectives

The final degree project (TFG) consists on the development of a test bench for the neutron detector electronics and digital acquisition systems of the nuclear instrumentation laboratory. This system must emulate the electronic response of the detector, be configurable and low cost.

From the main objective of developing a test bench, built-in secondary objectives have been established:

- Generate a digital pulse with exponential decay, having a similar frequency spectrum as an ideal one.
- Make an user-friendly GUI, to control the test bench pulse generator.
- Test the trapezoidal filter of the digitizer with a double exponential pulse.
- Develop a test bench able to be connected to the Detector Input, dispensing with the charge terminator and with precise control of the charge transmitted.
- Calculate the transfer function of an ideal charge preamplifier to add more functionalities to the test bench.

The average price of the pulse generators used as test bench in the market are around 3.000 €. In this project, we will try to do a pulse generator containing the necessary features to test preampifiers and digitizers with a digital circuit, easily configurable and low cost. Currently in the market, there is a wide choice of low cost microcontrollers. We decided to start investigating with the Arduino boards.

Arduino is an open-source hardware and software company that designs and manufactures single-board microcontroller kits for building digital devices. Arduino simplifies the task of using the microcontroller because their boards are already bootloaded and all the passive components and the clock crystal are already installed. Moereover they have their own integrated development environment (IDE) that allows us to program the Arduino very easily. Arduino manufactures many different boards. We decided to choose the Arduino Due since it has a DAC

output that will be very useful for the generation of pulses. The price for one Arduino Due turns around 40 €. A Raspberry Pi could also have been used because it also presents digital output pins and DAC outputs (normally used for audio). But at the end, the Arduino Due was chosen since the Raspberry Pi is a small computer, it is more complex to use and, above all, more expensive.

1.2 Scope

The scope is to develop a low cost test bench that will be used to verify the correctness of the preamplifiers and digitizers. For instance, the test bench can be used for:

- Adjusting the pole-zero cancellation of the preamplifier. This prevents undershoot effect and therefore erroneous measuring. The preamplifier has a potentiometer inside where it can be adjusted for an optimal response. The response is seen in an oscilloscope while the potentiometer is moved until the optimal response is got. The test bench can generate different decay times, therefore this test can be repeated multiple times with different decays to make sure the pole-zero cancellation is well done.

- Investigating the pile-up effect by making multiple impulses in a small time difference. The pile-up effect is another potential error of peak area measurement cause by the dead time of the treatment system.

1.3 How this TFG is structured

This TFG consist of 7 chapters and covers the following topics:

Chapter 2: Fundamentals of detector electronics. Quick introduction to LIN and the test bench found in the market.

Chapter 3: Digital pulse generator. The Arduino Due has been used to Generate a digital pulse by only programming the Digital to Analog Converter (DAC). An user-friendly GUI is presented and the results of testing the digitizer with our pulse generator.

Chapter 4: Analog pulse generator. Generate an analog pulse by connecting the outputs of the Arduino to an external circuit made of passive components. The external components then gives a exponential shape to the pulse.

Chapter 5: Preamplifier transfer function. A prediction of the response of an ideal preamplifier has been made to add more features to the test bench. For this, a transfer function of an ideal charge preamplifier has been made and then tested using Matlab.

Chapter 6: Project time line, project cost and environmental impact. Exposing the Gantt chart of the TFG. Explanation of the engineering and material costs. Environmental impact caused by the test bench.

Chapter 7: Conclusions. Conclusions and proposal of further investigation and work.

Chapter 2

Instrumentation for ionizing radiation detectors

2.1 Background

I had the pleasure of realizing my TFG at the Advanced Nuclear Technologies (ANT) more precisely at the "Laboratorio de Instrumentación Nuclear" (LIN). At the LIN, the experts work on the ionizing particle detection, they have multiple research lines: development of detector, development of preamplifier and detector's data analysis. In the Figure 2.1 it can be seen all the signal chain for the detection of ionizing radiation, more precisely for charge preamplifiers.

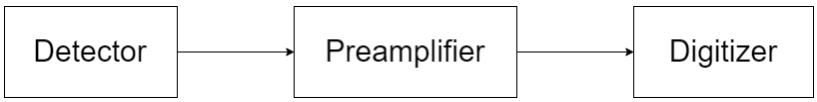

Figure 2.1: Signal chain for pulse counting

The research group at ANT has already developed some detectors, as the neutron BELEN detector, and some preamplifiers. From all these developments it has emerged the need to develop a test bench for the preamplifiers and digitizers. In the last decades, all the ionizing radiation detection electronics have been modernized. It has gone from analog systems to digital systems, due to that the digital electronics has been evolving rapidly and its robustness are improving more and more.

The preamplifier used during all the test is the Canberra Model 2006 Proportional Counter. The preamplifier presents 3 inputs and 1 output. The 3 inputs are a High Voltage Input, the Detector Input and the Test Input. To test the preamplifier only the Test Input an the Detector Input are going to be used. The Test Input is made to ideally receive an exponential pulse. That is why our test bench has to make exponential pulses. The preamplifier output is a step voltage pulse whose amplitude is proportional to the total charge collected in each event. The pulse decays with a time constant of 50 μs.

2.2 State of art

In the market, it can be found many types of digital and analog pulse generators. All the pulse generator have one thing in common, which is that they are excessively expensive, all exceed one thousand euros. Complex pulse generators, completely configurable, can be found. For example the Keysight Technologies Pulse Generator, where an USB can be connected with a file containing an arbitrary pulse and the generator reproduces it with a tiny sampling time (the generator can process data in the order of 56 Gb/s).

At the LIN, the pulses to test the preamplifiers and digitizer are done with pulses generator as the Orthec Model 419 Precision Pulse Generator or the TG1000 Function Generator. The Ortec Model 419 is a Precision Pulse Generator that simulates the detection of a nuclear particle reaction in a semiconductor or scintillation radiation detector, as well as serving as a specialized pulse generator for use with pulse processing instrumentation. It can be calibrated to read directly in terms of equivalent energy deposition in semiconductors, and the rise time of the pulse may be varied to simulate the collection time constant in the radiation detector [Manual,419]. Its cost is not published online but it is estimated to be between 3000 € to 4000 €. The TG1000 is a DDS function generator. DDS (direct digital synthesis) is a technique for generating waveforms digitally using a phase accumulator, a look-up table and a DAC [TG]. Its costs around 500 €(but we have to point out that it is a generator of functions and not a pulse generator that is what is ideally sought to test the devices).

The features of both devices are briefly presented in the table below. Additionally, the Ortec Model 419 just can do pulses with an exponential decay fixed with a decay constant of 200 μs and the TG1000 cannot do pulses with exponential decays but can do triangular, square and sinus, even if for preamplifiers just the square pulse is useful. Moreover, both of the devices are not portable. The Ortect 419 operating power is fed from a power supply trough a NIM standard module. And the TG1000 needs to be supplied directly from the electrical AC network.

	Variable Amplitude	Variable Rise Time	Variable Frequency	Decay constant
Ortec Model 419 Pulse Generator	0 to $\pm$1 V	5 ns, 20, 50, 100, and 250 ns.	✗	200 μs
TG1000 Function Generator	5 mV to 20 V pk-pk	✗	0.001 Hz to 10 MHz	✗

Table 2.1: Comparative table of LIN generators

On the internet, it can be found wave generators made with the Arduino Due. For example, from the Arduino Project Hub [WaveGen], a blogger propose a user-friendly program that let the user make arbitrary waves forms. But all the proposed wave generators made with Arduino have a large sampling time, unabling them for being a test bench for nuclear instrumentation.

Chapter 3

Digital pulse generator

3.1 Objectives

The option of creating a digital impulse has been studied. For this an Arduino Due has been provided by the Laboratory of Nuclear Instrumentation. The objective is to test if a signal made from digital signals can generate an impulse of the same order of the Ortec Model 419 Pulse Generator, with a similar frequency spectrum.
Later on, an user-friendly GUI has been made to automatically write arduino files containing the pulses to be directly transferred to the Arduino Due.
Finally, the GUI has been tested with a digitizer to confirm the performance of the Arduino and the GUI's option of the second pulse has been used to see how the impulse interacted with the trapezoidal filter of the digitizer.

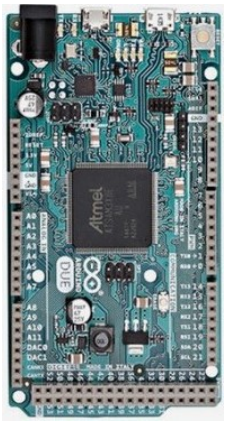

Figure 3.1: Arduino Due

3.2 Frequency spectrum

First of all, the frequency spectrum has to be checked. This analysis is very important, if the frequency spectrum of the digitized pulse is not similar to the ideal one, there is no sense on keeping going with this solution because the output signal will go though devices that may have analog and digital filters. Therefore, even if the signal looks similar but has different frequency spectrum, the response made by the devices will be different.

Matlab has been used to make the fast Fourier transform and thus be able to perceive if our digital signal suffered great disturbances of frequencies compared to the ideal exponential signal. To put us in the worst case it has been purposely made that the digitization of the signal is extremely poor and we established the tipical decay constant of 50 μs. The result is shown in Figure 3.2. As we can see in the figure the error es very small, it shouldn't cause problems with the preamplifier. Therefore, we can proceed to investigate the ways to generate a digital

exponential pulse.

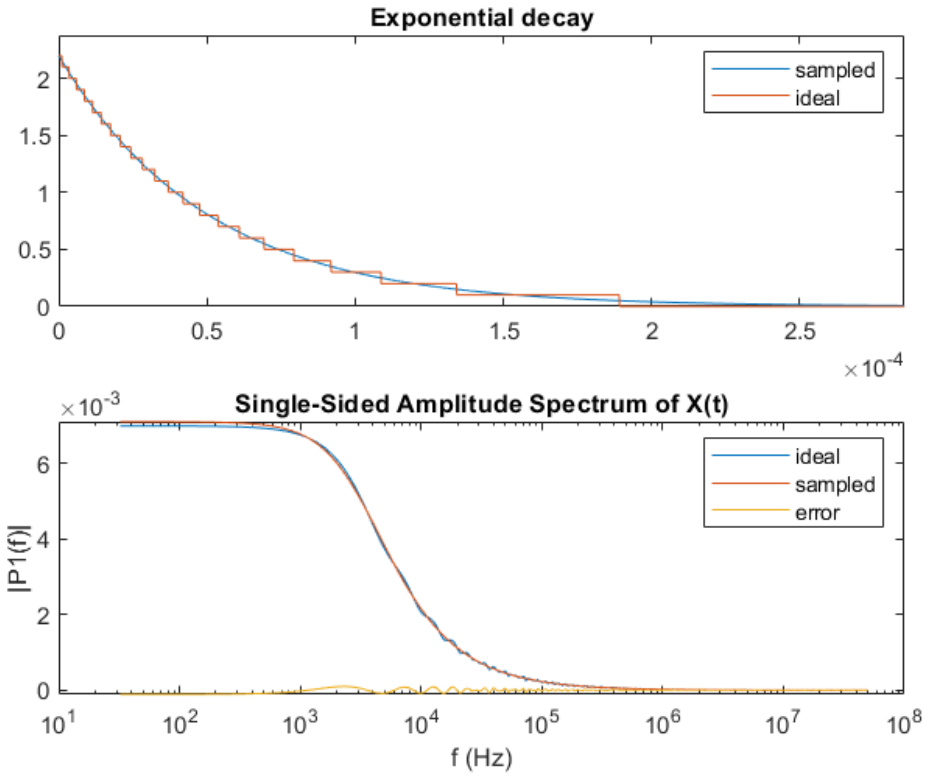

Figure 3.2: Fast Fourier transform

3.3 Arduino IDE

The initial purpose of Arduino was to facilitate the use of microcontrollers. Therefore, they propose to use their own integrated development environment (IDE). The user then can easily program the Arduino using the Processing programming language. But what the Proccesing language does is to call function and libraries of the IDE written in C/C++. For our purpose of making fast signals, this is very inefficient. Therefore, even if it is more complex, to reduce the time between signals, we could write the code in C. We could also write the code in Atmel Studio since the microcontroller is an Atmel.

To test the inefficiency of writing the code in the Arduino IDE, the following program has been written:

```
void loop () {
   analogWrite(DAC0, 4095)
   analogWrite(DAC0, 0)
   analogWrite(DAC0, 4095)
   analogWrite(DAC0, 0)
 }
```

The result analyzed with the oscilloscope showed that the time the microcontrollers takes to run one line (sampling time) of the code is 3.48 μs, with a rise time of 500 ns. This is too much

time for a instrumentation test bench. Moreover, the time taken to come back to the beginning of the loop is even greater that $3.48\ \mu$s.

3.4 Digital to analog controller (DACC)

The Arduino Due has two very important characteristics. It has an Atmel SAM3X with an internal clock of 84 MHz and a 12 bits digital to analog converter (DAC) output.

To create the exponential decay we could simply write all the value in an infinite while loop to continuously generate the exponential impulse. Nevertheless, a simple command to the DAC output takes $3.48\ \mu$s and with a rise time of 500 ns, moreover the CPU will be constanly used to generate the signal and in case we would want to add more utilities it would be nice to have it free. It would be seriously inconvenient for the pulse generator to have this features, so others solutions have been explored. Microprocessor developer have designed a fast way to send digital impulses to the outputs. SAM3X include a peripheral direct memory access (DMA) functionality to the digital DACC. With the help of this module, we van move list of value from RAM to the DACC's conversion-data register with minimal CPU intervention. According to the SAM3X datasheet [Atmel], the list of values have to be defined in `DACC_TNPR`.

When the peripheral has finished transferring all the data from the buffer to the DAC, an automatic interrupt provokes a CPU intervention. The processor needs to know how long is the transferred data, according to the SAM3X datasheet [Atmel], the parameter `DACC_TNCR` defines this length.

The following code shows how the peripheral direct memory access of the DACC has to be defined. The *if* statement is used to check that the previous pulse is fully sent.

```
void DACC_Handler(void) {
   if((dacc_get_interrupt_status(DACC) & DACC_ISR_ENDTX) == DACC_ISR_ENDTX){
      DACC->DACC_TNPR = pulse;
      DACC->DACC_TNCR = length;
    }
 }
```

Moreover, in the code, we have to indicate that the *DACC Handler* is going to be used. For this, we have to enable the DAC clocking, the DACC's transmission buffer interrupt and indicate which DAC output is going to be used. The following code shows how to set up the DACC:

```
void dac_setup () {
   pmc_enable_periph_clk (DACC_INTERFACE_ID) ;
   dacc_reset(DACC);
   dacc_set_transfer_mode(DACC, 0);
   dacc_set_channel_selection(DACC, 0);
   dacc_enable_channel(DACC, 0);
   NVIC_DisableIRQ(DACC_IRQn); //optional
   NVIC_ClearPendingIRQ(DACC_IRQn); //optional
   NVIC_EnableIRQ(DACC_IRQn);
   dacc_enable_interrupt(DACC, DACC_IER_ENDTX);
   DACC->DACC_PTCR = 0x00000100;
```

```
}
```

The time that takes the Arduino to send a data through the periferical of the DMA is probably the most critical value for the accuracy of our program. First it was calculated with the oscilloscope but once started the tests with the digitizer we could see that the frequency was not sufficiently accurate and we had to modify the value of the step time. In the end we concluded that the step time to send a DAC signal is 619.05 nanoseconds this is equivalent of 52 clock times. This is a very competitive result because it allows us to be in the range of values where impulse generators for nuclear instrumentation are. The rise time for a impulse of amplitude 1V is 240 ns, this value is between the rise time values that allow to select Ortec Model 419, consequently this rise time is valid to simulate a particle impulse. One of the negative aspects is that with Arduino we can not select the rise time that we want unlike Ortec Model 419 but as we saw when we tested it (where rise times were very wrong with the rise time they said they do) it is not a major drawback.

3.5 Graphical user interface (GUI)

A GUI have been programmed with PyQt. PyQt is an open source toolkit software for creating GUI's that runs on various operating systems such as Windows, GNU/Linux, Mac OS X.

In this GUI, the user can generate exponential pulses. For this the user has to enter the frequency in hertz, the decay time in microseconds, and the amplitude in volts. In case the user wants to add a second impulse in the middle of the first one, he has to introduce the decay time, amplitude and a time delay (TD) in microseconds for the second impulse. Then the user can simulate the pulse to check it visually. This plot has been made using the library matplotlib.

To add more flexibility to the program, we have added the option to read a text file with the amplitude of the points and a time base and generate the file of the Arduino automatically. Currently the time base is fixed but for future work it would be necessary to let the user set his own time base in the .txt file and then the program itself will transfer and interpolate the points in its own time base (which is the step time to send a data that will be explained later).

At the moment, when a file is generated, it is automatically saved in the folder where the program is located. For future work, the option on selecting the folder where the folder has to be saved should be added.

The GUI program is a *.exe*, this implies it can only run on Windows operating system. The source file has to be compiled in the others operating systems to make sure the

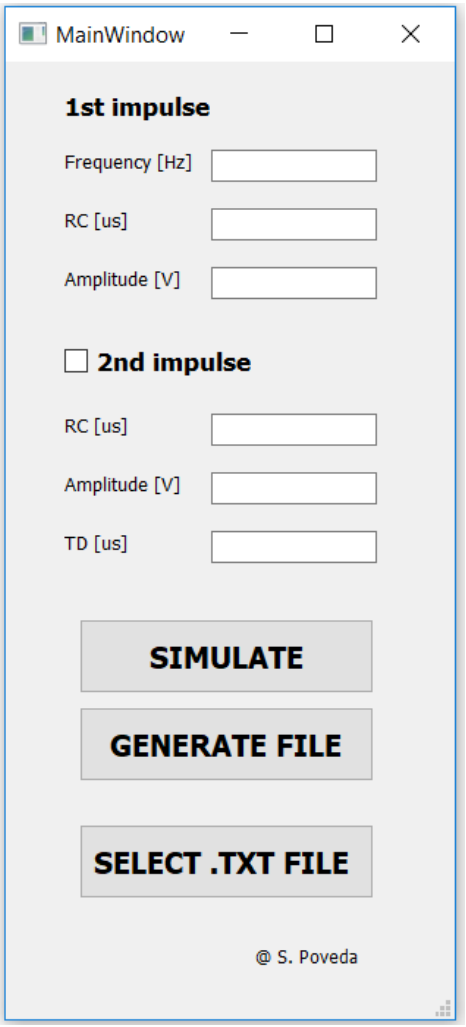

Figure 3.3: GUI

program can be used all the platforms. Before compilation, the programmers has tho make sure all the libraries used are downloaded.

For future works, it would be interesting that when the users clicks the button 'Simulate', the curve is plotted in the same window, not in a separate one. Moreover, the program's computation time could be optimized and the script could be simplified by making functions separate from the `main()`. It would simplify the understanding of the script for future programmers that would want to implement more features to the software.

3.6 Specifications

3.6.1 Frequency

The fact that we have a step time of 619.05 ns, it makes the time delay (TD) to be a multiple of 619.05 ns. Likewise we have the same problem with the frequency. For example, if we want an exact frequency of 1000 Hz we could never get it, we will get a frequency of $1/(619.05[\text{ns}] \times 1615) = 1000.23$ Hz or $1/(619.05[\text{ns}] \times 1616) = 999.62$ Hz. The more the frequency decreases the more this error will decrease. For the moment, the GUI only can make frequencies from 807.69 Hz to 1615.38 Hz.

A frequency test has been made programming the Arduino, with the GUI, with different values of frequency and digitize them with the digitizer. This test allowed us to precisely adjust the step time of the Arduino. Once the step time was optimized, the frequency test was repeated and the results are shown in Figure 3.4. As we can see in the figure, the maximum error (for the frequencies we tested) is 0.4%, a very reasonable value. In future works, it would be recommended to enlarge the range of frequencies to be able to make lower frequencies.

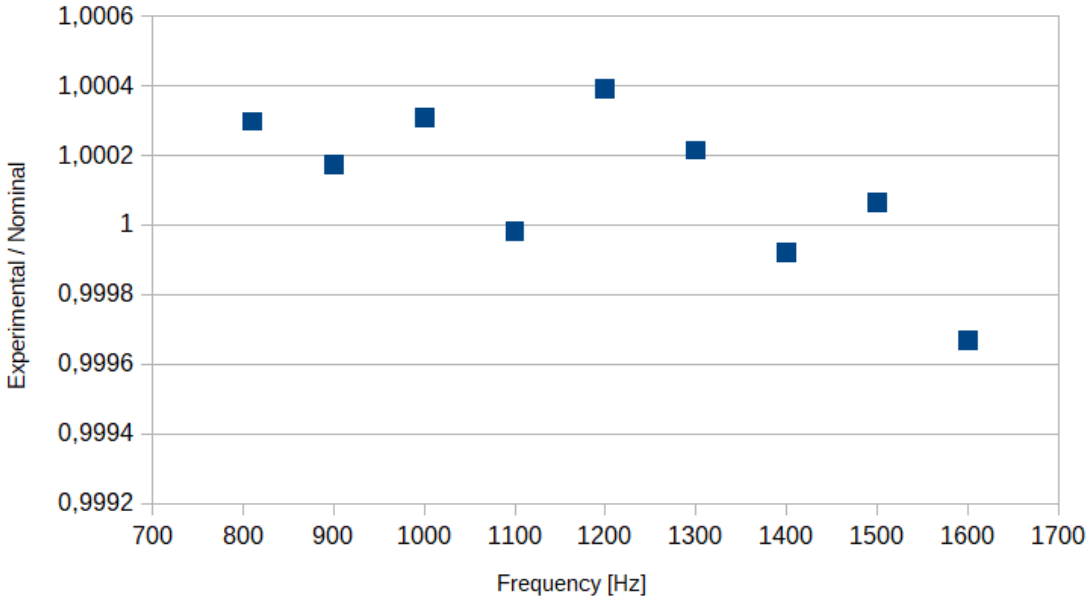

Figure 3.4: Frequency test

3.6.2 Amplitude

An amplitude test has also been made to verify the linearity of the amplitude that the user demands. The results of the amplitude test are shown in Figure 3.5. As shown in the figure, $R^2 = 0.99967$ this means the variance of the amplitude is very low and therefore we can ensure good linearity.

The DAC output has an output range from 0.55 V to 2.75 V with a resolution of $(2.75-0.55)/(2^{12}) = 0.537$ mV. The DAC output can be displaced from 0 V to 2.2 V by adding some diodes at the output or other more complex circuit. This resolution is very precise to be such a cheap device but the problem is that when we want pulses smaller than 10 mV we can not ensure them since the noise at the output of the Arduino is 3 mV.

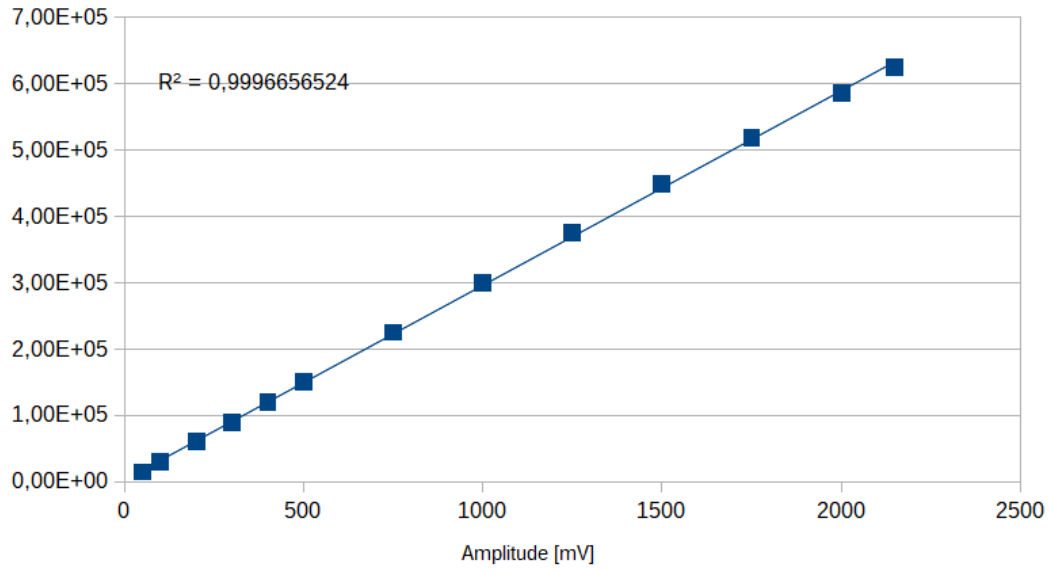

Figure 3.5: Amplitude test

The digital pulse generator has been named LinGen. The table summarizing the features of the LIN devices of Chapter 2 is been retaken and we have added another row with the features of LinGen:

	Variable Amplitude	Variable Rise Time	Variable Frequency	Decay constant
Ortec Model 419 Pulse Generator	0 to ± 1 V	5 ns, 20, 50, 100, and 250 ns.	✗	200 μs
TG1000 Function Generator	5 mV to 20 V pk-pk	✗	0.001 Hz to 10 MHz	✗
LinGen	0 to 2.2 V	fixed at 240 ns	808 Hz to 1615 Hz	5 μs to 206 μs

Table 3.1: Comparative table of generators

3.7 Application, trapezoidal filter

In the laboratory we have made a test to know how the digitizer reacts to a double impulse. To do this the DAC pin and the GND pin of the Arduino have been connected directly to the digitizer. To realize this connection we had to improvise a cable, we took a coaxial cable and peel it from both ends. One end to connect it to the Arduino pins and the other end inserted into a Lemo connection compatible with the digitizer.

The digitizer makes a trapezoidal filter made of 3 tales (L,G,L) where the flat tale in the middle illustrates a proportional quantity to the amplitude of the impulse. The trapezoidal filter is calibrated for an exponential input with a decay constant of 50 μs, therefore all the signals used for this test will have a decay constant of 50 μs. At the same time, if we send a single exponential signal and observe that the second tale is completely flat this means that the signal really has a decay constant of, in this case, 50 μs. If for example we put a higher decay constant we would see that the tale in the medium slightly increase or if we put a lower time constant, the middle tale would slightly decrease.

The methodology used is the following, we have taken the results of the frequency, the amplitude and the trapezoidal filter in difference cases of delay time for the second impulse. When the second impulse has a delay time that puts it in:

- Outside the trapezoidal filer, with a TD = 20.0 μs, Figure 3.6.
- Inside the third tale, with a TD = 8.0 μs, Figure 3.7.
- Inside the second tale, with a TD = 3.7 μs, Figure 3.8.
- Inside the first tale, with a TD = 0.619 μs, Figure 3.9.

We have repeated this test for different amplitudes (A_1 represents the amplitude of the first impulse, A_2 represents the amplitude of the second impulse):

- $A_1 = A_2 = 1$ V
- $A_1 < A_2$, with $A_1 = 0.5$ V and $A_2 = 1$ V
- $A_1 > A_2$, with $A_1 = 1$ V and $A_2 = 0.5$ V

In total 12 experiments.

In the Figure 3.10, it can be seen the amplitudes detected by the trapezoidal filter, that is to say, the maximum value seen by the trapezoidal filter. It is necessary to emphasize, that the value of the axes are arbitrary.
Looking at this figure, we can start understanding the effect of the delay time.

- When TD = 20 μs, the filter captures both of the amplitudes. As we can observe the values are 616 and 308, respecting well the proportion of the values of $A_1 = 0.5$ V and $A_2 = 1$ V.
- When TD = 8 μs, the second impulse is located in the third tale of the trapezoidal filter. It has time to detect the two impulses and returns the greater amplitude. In this case 616, passed in volts, 1 V.

- When TD = 4 μs, the second impulse is located in the middle of the second tale. The filter detect the first impulse, start detecting the second one but don't have time to fully detect it. The final amplitude result is the first of the first impulse plus a part of the second one. In this particular case we obtain 712, passed in volts, $1.16 < 1.5$ V.

- When TD = 1 μs, the second impulse is located at the start of the first tale. The filter has time to fully detect both amplitudes. The result is similar to when it only detects one impulse but the amplitude is the sum of both amplitudes. It can be said that the filter detects it has it was only one but with a big rise time.

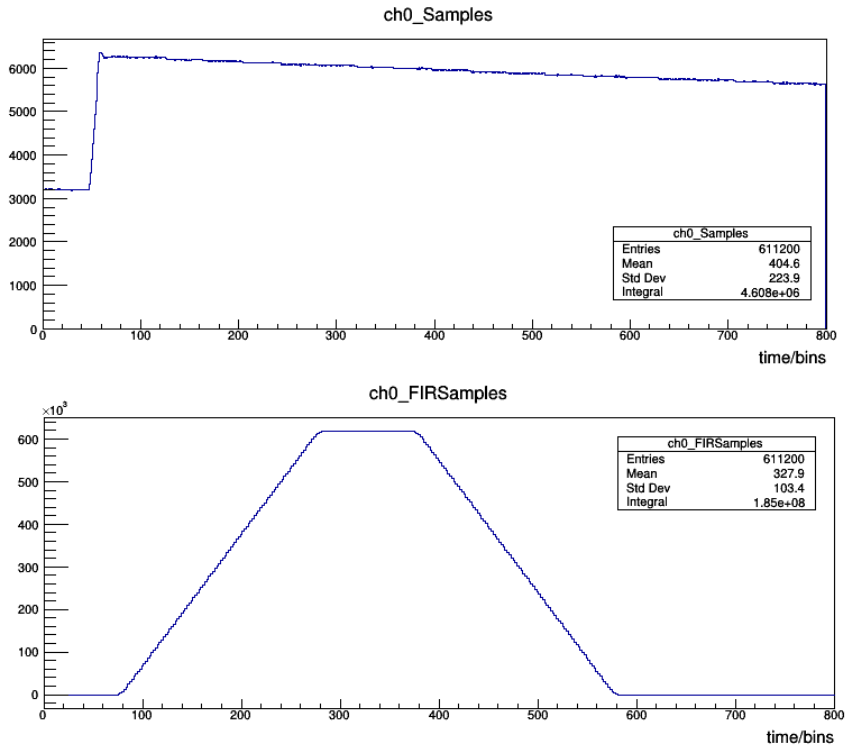

Figure 3.6: Samples and trapezoidal filter with A_1 equal to A_2 and TD = 20 μs

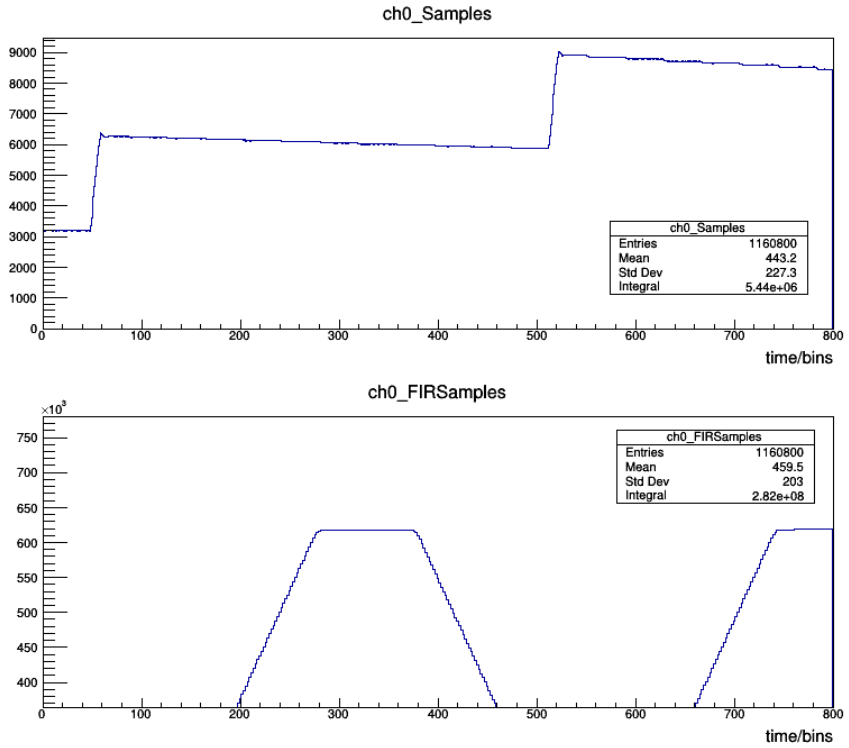

Figure 3.7: Samples and trapezoidal filter with A_1 equal to A_2 and TD = 8 μs

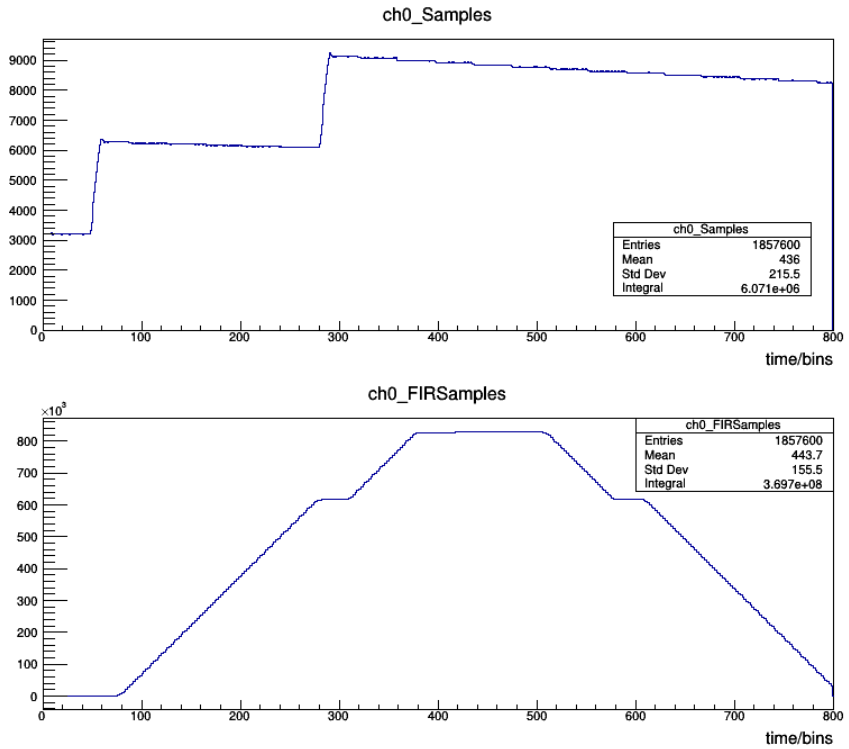

Figure 3.8: Samples and trapezoidal filter with A_1 equal to A_2 and TD = 3.7 μs

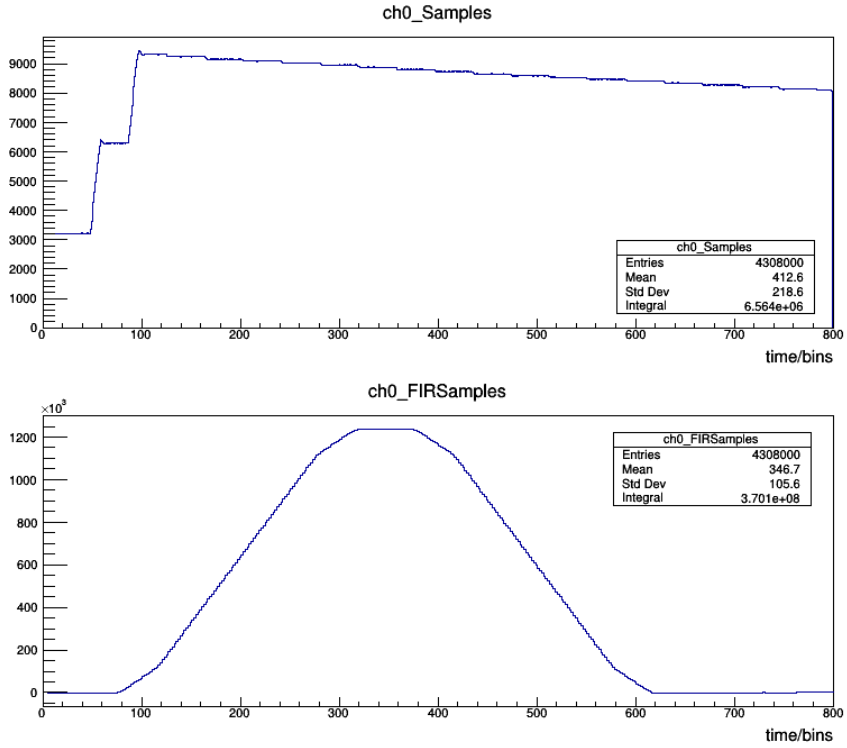

Figure 3.9: Samples and trapezoidal filter with A_1 equal to A_2 and TD = 0.619 μs

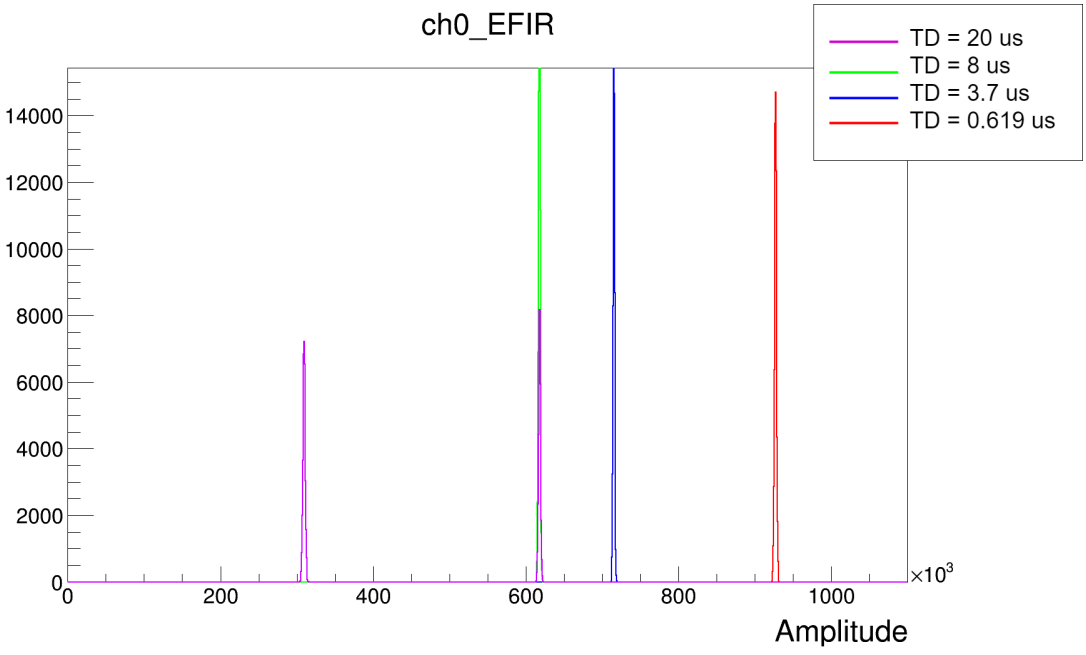

Figure 3.10: Amplitudes detected with the trapezoidal filter

Chapter 4

Analog pulse generator

4.1 Objectives

The electronic circuit developed has the purpose of simulating the detector's charge transmission. Consequently, it would be more desirable that the designed circuit would have to be connected at the Detector Input of the preamplifier and not to the Test Input (again with cheap components such as Arduino). Additionally, we want to make sure we know exactly the charge transmitted to the preamplifier because this is what the ionizing particle detector generates. For this reason, we have decided making the charge impulse with the discharge of a capacitor. We have to make sure the capacitor is discharged fully in the preamplifier with no other charge sources. A relay has been chosen to make this galvanic isolation.

This chapter takes a different approach than the one given by the Ortec Model 419 Precision Pulse Generator (when it is planed to connect it directly to the preamplifier input). In the Ortec Model 419 manual, they propose to use a charge terminator, that is found attached at the back of the device, to connect it directly to the Detector Input. A charge terminator is quite small device with an equivalent circuit shown in Figure 4.1 that allows converting an output voltage pulse to a charge pulse. We are also going to start investigating whether the charge pulse can be made without the charge terminator.

This chapter is arranged in the way that it can be seen the progression through time until the final design:

- Simple capacitor discharge through a relay.
- Capacitor discharge through a relay with attenuator.
- Capacitor discharge through a relay with attenuator and voltage divider.

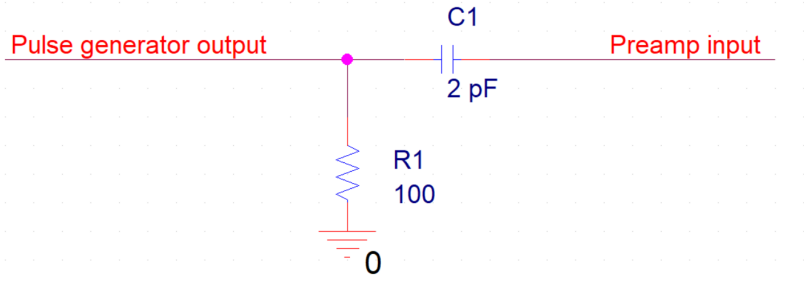

Figure 4.1: Charge terminator

4.2 First approach

We started by making a circuit with just the basic elements, as shown in Figure 4.2. As it can be seen, we only use two digital Arduino outputs. The relay was recycled from old nuclear instrumentation, the relay model is PRMA1. The signal sequence that the Arduino sends is the following:

1. Firstly the microcontroller supply a current to the capacitor C_1. The capacitor then starts accumulating charge.

2. The microcontroller stops supplying the capacitor. A zener diode is connected in inverse biasing so that the capacitor doesn't discharge trough the Arduino.

3. Secondly, the Arduino activates a relay with a 5 V digital pulse from one of the digital pins of the Arduino. The capacitor is then discharged trough the relay and goes to the output.

4. The microcontroller stops supplying the relay.

5. The cycle is repeated.

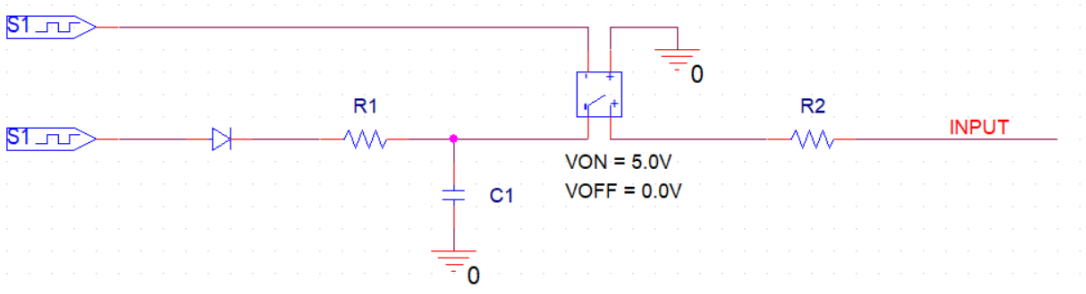

Figure 4.2: Simple circuit

The result obtained with the discharge of the capacitor is shown in Figure 4.3. The response has been captured connecting an oscilloscope of internal resistance of 1 MΩ in parallel with R_2. For this test, the capacitor is fully charged with a tension of 4.8 V (= 5.5-07). The component values are $R_1 = 220\ \Omega$, $C_1 = 100$ nF, $R_2 = 2200\ \Omega$. The result present many mechanical rebounds made by the relay. Mechanical rebounds can be seen zoomed in Figure 4.4. The circuit could also have induced noise from the relay inductance, it will be especially noticeable when we go down to very low charge levels.

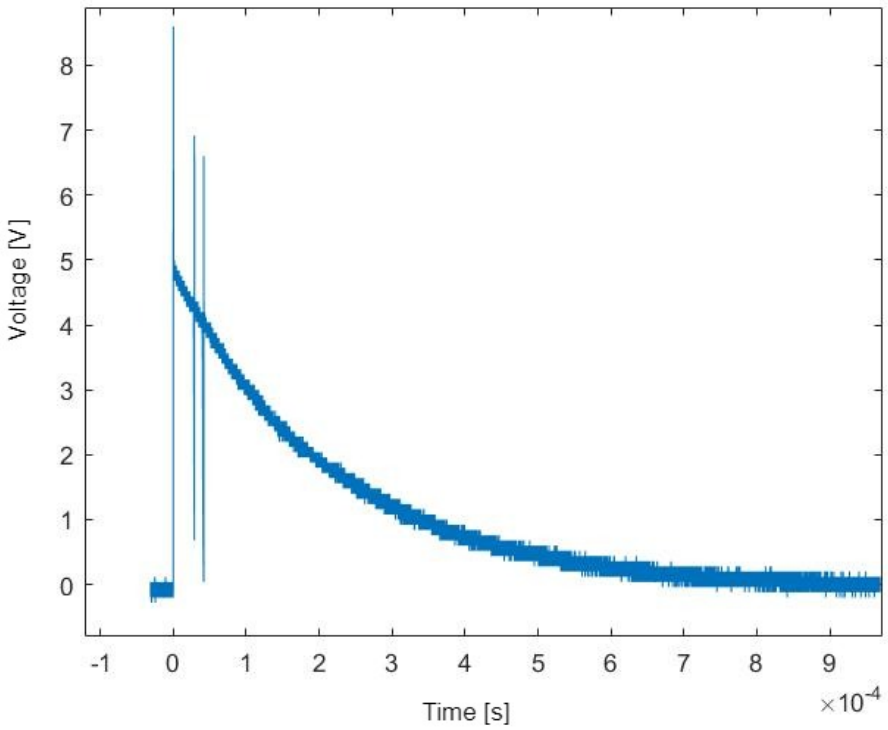

Figure 4.3: Pulse generated by the simple circuit

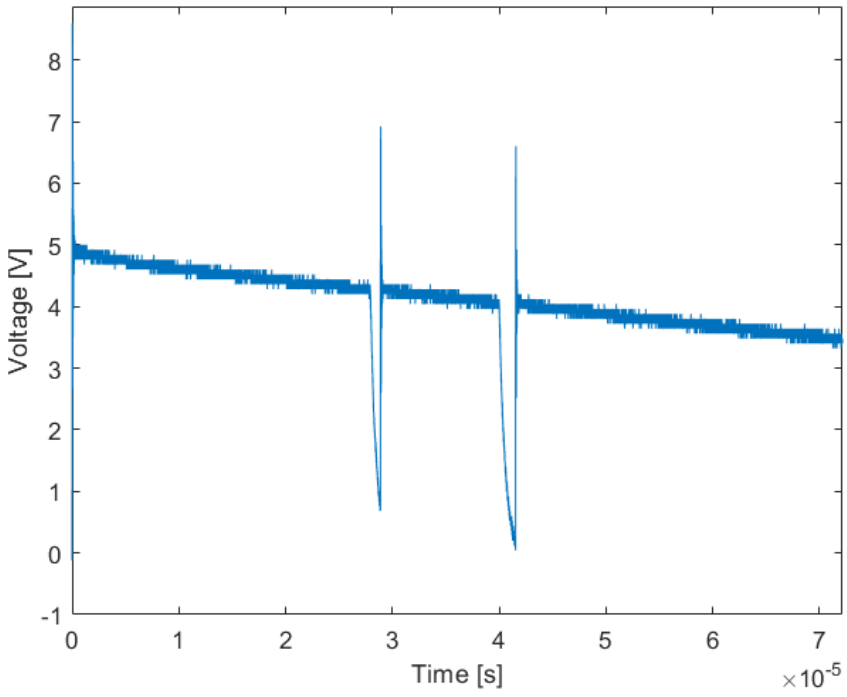

Figure 4.4: Zoomed pulse generated by the simple circuit

4.3 Capacitor discharge through a relay with attenuator

To fix the disturbances in the signal a second capacitor is added, as shown in Figure 4.5. Moreover, changing the value of this capacitor changes the rise time of the signal. This capacitor takes the functionality of an attenuator. The result is obtained in Figure 4.6, as it can be seen the mechanical disturbances made by the relay disappeared. The components value are the same as in the simple circuit plus a second capacitor $C_2 = 56$ nF. The negative aspect of this circuit is that a big value of C_2 is needed to erase the initial peak, this yields to a quite long rise time (around 1 μs).

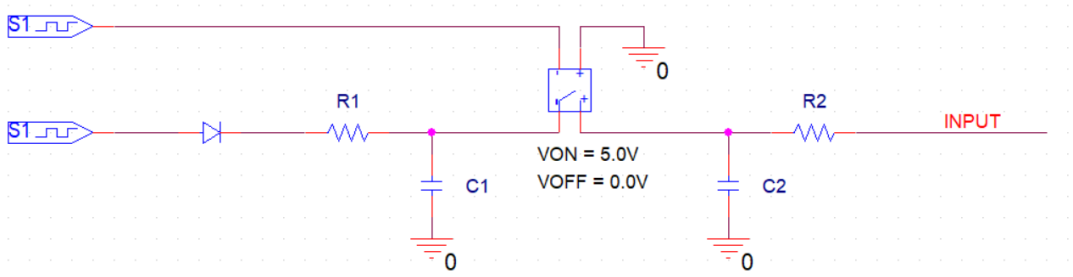

Figure 4.5: Circuit

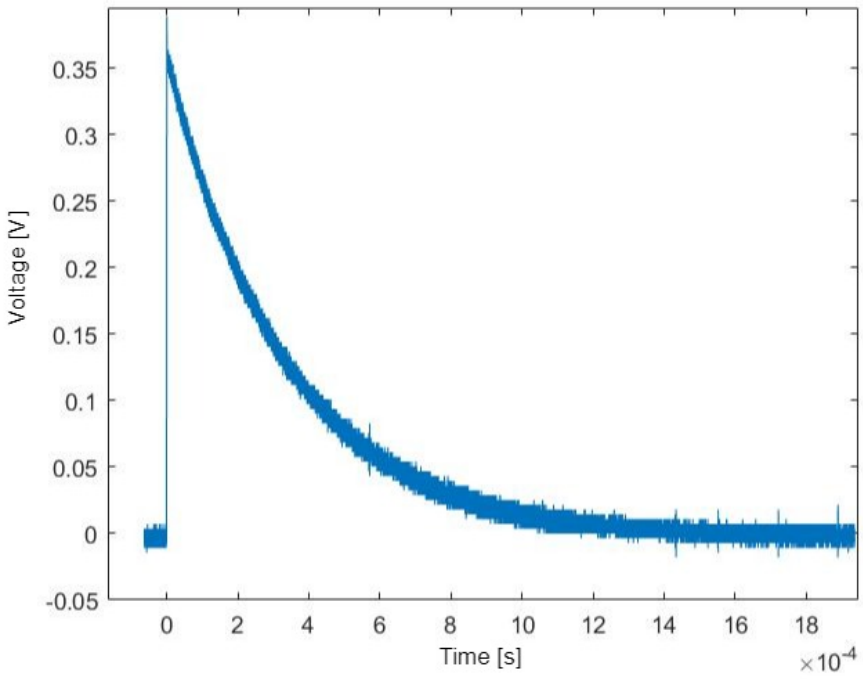

Figure 4.6: Pulse with attenuator

If we want to implement this circuit to the Detector Input entrance, we need to find a way of regulating the charges inside the capacitor because when the preamplifier is tested with a function generator, it expects to detect values of the order of mili-volts. Charges absorbed by C_1 can be regulated by fixing the time of charge of the capacitor with the following equation:

$$Q = Q_0(1 - e^{-t/\tau}) \tag{4.1}$$

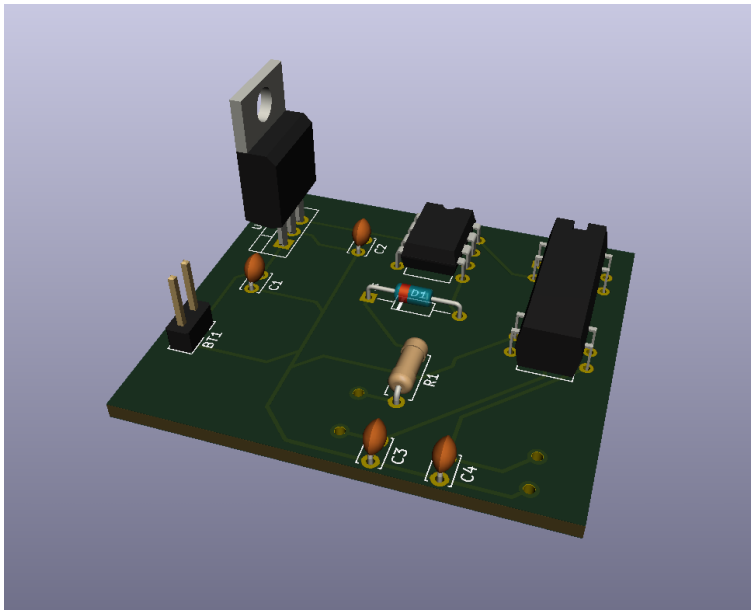

Figure 4.7: 3D design of the PCB

with,

$$Q_0 = VC_1$$

The voltage detected by the oscilloscope will be differe nt of the voltage stored in the capacitor C_1:

$$V_2 = V_1 \frac{C_1}{C_1 + C_2} .$$

A preliminary design was made with KiCad in which we substituted R_1 and R_2 for potentiometers. As it can be seen from the 3D view of the PCB, shown in Figure 4.7, there is a 7805 voltage regulator because the PCB was to be sourced with a battery. In this case, the time that Arduino delivers current to the capacitor 1 is constant, what makes the charge inside vary is the potentiometer 1 because we can regulate the charge time. With the potentiometer 2, we regulate the discharching time.

The main advantage of making a PCB is that firstly the noise created from the breadboard would disappear and then we could put it inside a Faraday cage. A Faraday cage, also known as Faraday shield, is an enclosure that is used to shield things from electromagnetic fields (both static and non-static). The certain types of electromagnetic radiation of the environment will disappear (distributed around the cage). Others tricks to attenuate the noise are, for example, shielding of cables, terminals connection to ground, a well-made ground connection, high quality PCBs, foresee the current loops and consider the possibility of line filters.

4.4 Capacitor discharge through a relay with attenuator and voltage divider

Using the charge time of a capacitor, to regulate the voltage, is not practical since it has been observe a notable difference between the theoretical voltage and the experimental voltage. This can be caused by several factors such as that the decay constant is not accurate due to the variability of the values of the passive components or also the fact that the step input of the Arduino is not perfect, it has a rise time that is not considered in the equation (Eq. 4.1).

At the end, controlling the charge time has been discarded to transmit small charges to the preamplifier and we have decided to implement a voltage divider. A voltage divider produces a V_{out} that is a portion of V_{in}. This is simply made by connecting two resistors in series. The final circuit with the voltage divider is shown in Figure 4.8. The second resistor of the voltage divider is a potentiometer (the one that goes to ground). This potentiometer regulates the potential difference of the capacitor. If the potential difference trough the potentiometer is too low, the current won't flow through the capacitor because of the diode. For silicon diodes, the built-in potential is approximately 0.7 V (0.3 V for germanium and 0.2 V for Schottky), approximately because of the diode's nonlinear current-voltage characteristics. To fix this problem we could add a resistor in series with the potentiometer so that the potential difference do not go below the built-in potential.

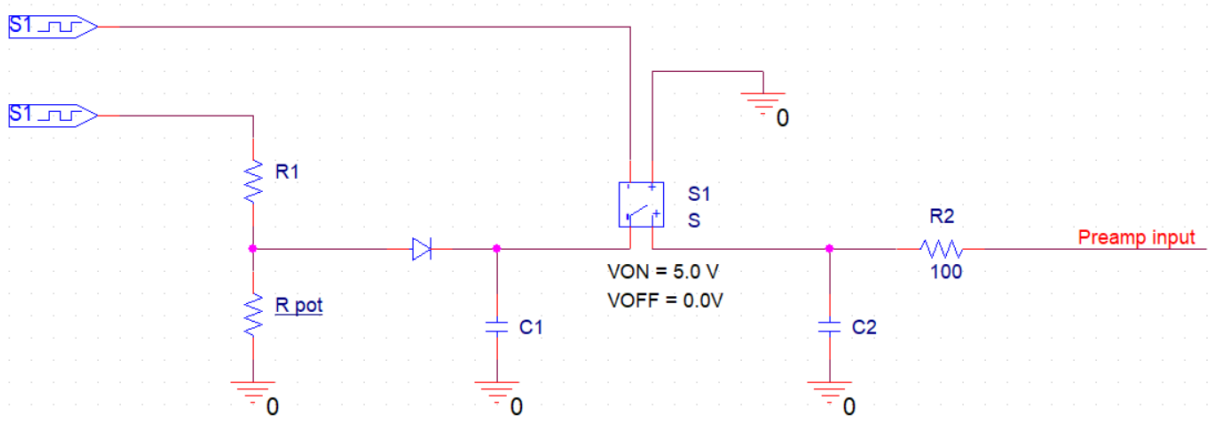

Figure 4.8: Circuit

As it can also be observed in Figure 4.8, the second resistor is 100 Ω. This is due to the signal reflection effect. The signal reflection occurs when there is an impedance mismatches in the cable. This impedance mismatches can produce errors in the transmission of signal such as echo, attenuation, standing waves or ringing. To avoid this unwanted effect, the nuclear instrumentation fixes its devices and coaxial cables with an characteristic impedance of 93 Ω. Therefore, in our circuit it is done the same but with a shunt resistor of 100 Ω because it is cheaper and the difference is not so big to have signal reflection. Although it seems surprising for its high cost, the charge terminator of the Ortec Pulse Generator also has a shunt resistance of 100 Ω.

4.5 Test

The circuit, shown in Figure 4.8, has been tested by connecting it to the preamplifier Detector Input. The values of the components of the circuit were: $R_1 = 100\,\Omega$, $R_{\text{pot}} = 440\,\Omega$, $C_1 = 22\,\text{pF}$,

$C_2 = 2.2$ pF and $R_2 = 100\,\Omega$. Very small values of capacitors were taken in order to store small amounts of charge

$$Q = CV = \left(22 \times 10^{-12}\right)\left(5\frac{100}{540} - 0.7\right) = 4.97 \times 10^{-12}\ \text{C}.$$

The results obtained were not as expected. There was a lot of noise coming out of the circuit that were later amplified by the preamplifier and the charge signal was too big so consequently the preamplifier saturated and gave us values of 12 V. In the end, it was decided to discard this circuit design and it has been concluded that it was much more practical to use a charge terminator.

Chapter 5

Preamplifier transfer function

5.1 Objectives

The objective of this study is to obtain the transfer function of an ideal charge preamplifier to be able to extract the response function. This is no more than a theoretical exercise. It can be useful for future designs of preamplifier. The researches that will develop the preamplifier will be able to choose the most suitable components so that the effect of undershoot is not appreciable.

In particle detection, we assume that the discharges have a step shape, but this is not necessarily true, it can have a slow decay. This decay can cause a slight zero crossover or undershoot of the output pulse, which then recovers back to zero through time. In case that another discharge is detected when the output remains negative, the amplitude of the pulse will be slightly lower [Knoll,2010]. Causing an erroneous reading of the amplitude. This undershoot problem can be minimized by putting a pole-zero cancellation circuit just after the integrator and therefore get correct measurements.

Canberra Model 2006 manual's functional circuit is shown in Figure 5.1. In this study, we are going to extract the transfer function from the test input to the pole-zero (P/Z) included of an ideal charge preamplifier taking as reference the function schematic of the Canberra Model 2006, even if the schematic is far from the complex system of the Canberra Model 2006 circuit. To facilitate the understanding of the later formulas, the components of the integrator and pole-zero circuit have been named as shown in Figure 5.2.

The methodology used to obtain the transfer function is as follows:

1. Transfer function of the amplifier ($V_{\text{int}}/V_{\text{in}}$).
2. Transfer function of the Pole-Zero cancellation circuit ($V_{\text{pz}}/V_{\text{int}}$).
3. Gain of the Canberra Model 2006 calculated experimentally.

The last part of the functional circuit (shown in Figure 5.1) is a buffer. The buffer is an operational amplifier that provides a gain to the signal and where the signal is not affected by whatever load is placed the circuit. But in this study we are not going to study it deeply. In our ideal transfer function it will be modeled as a constant gain K that is going to represent the gain

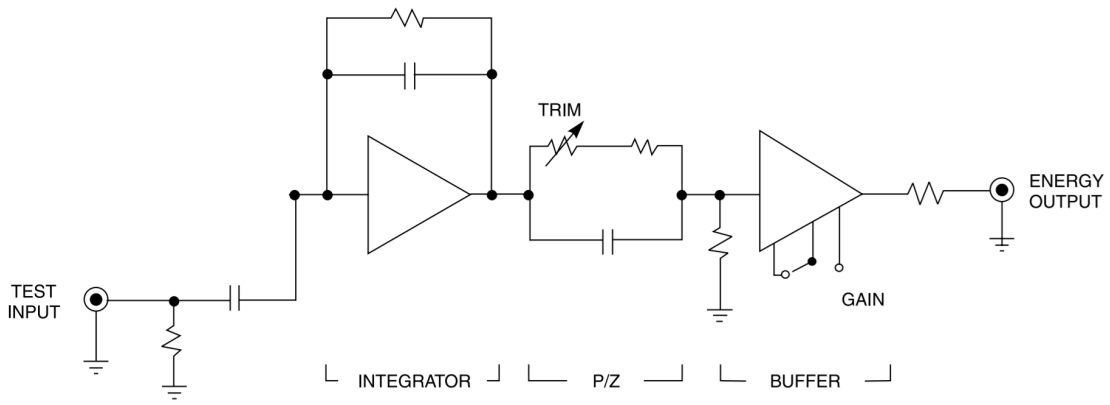

Figure 5.1: Functional schematic

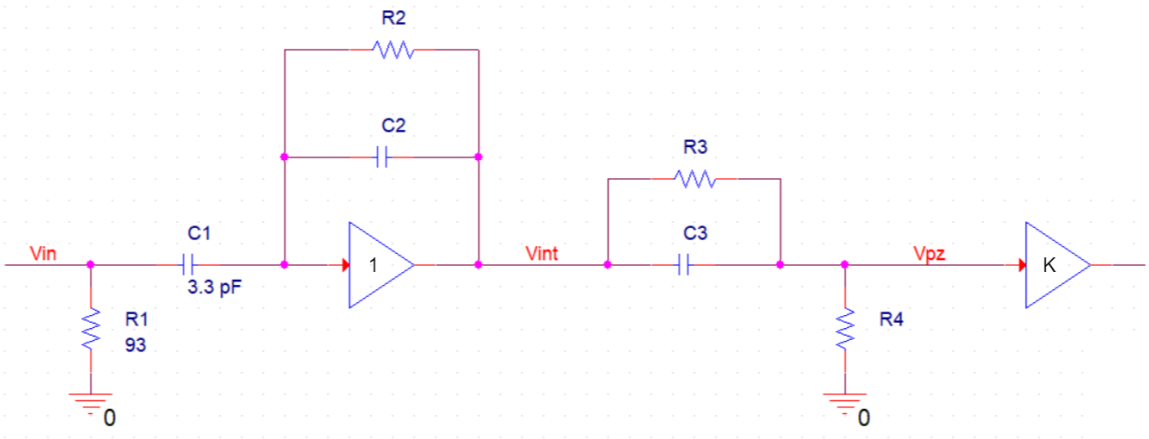

Figure 5.2: Test Input, Integrator and Pole-Zero functional schematic

of the whole preamplifier.

5.2 Voltage at Test Input

In the Test Input we want to generate impulses with exponential decay to force in the output of the preamplifier an undershoot. For this reason an external $C_{\text{ext}}R_{\text{ext}}$ circuit has been made, so that an input signal can be generated with a known decay time. The external circuit connected to the input circuit is shown in Figure 5.3. V_{ext} is a positive squared signal, simply made with an Arduino.

The entry of the circuit has been simplified, the value of C_{ext} has to be at least 1000 times greater than C_1, so that the effect of C_1 can be omitted from the external circuit connected to the Test Input. This way the input function drastically simplifies.

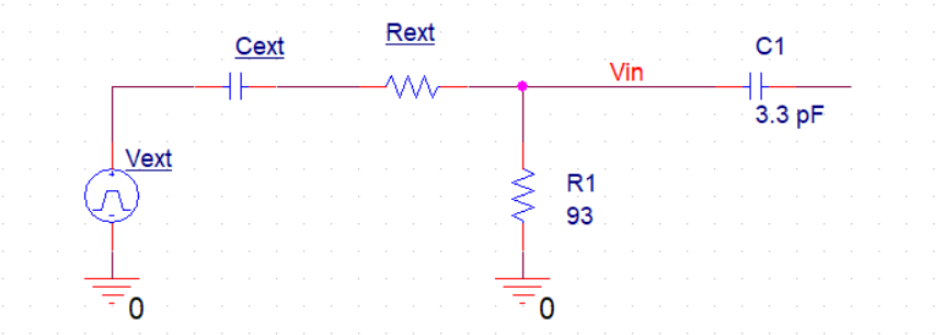

Figure 5.3: External circuit connected to the Test Input circuit

Below the output V_{in} in Laplace frequency domain:

$$\begin{aligned} V_{\text{ext}}(s) &= V/s \\ V_{\text{in}}(s) &= I(s)R_1 \\ I(s) &= \frac{V_{\text{ext}}(s)}{\frac{1}{sC_{\text{ext}}}R_{\text{ext}} + R_1} \end{aligned}$$

$$\Rightarrow V_{\text{in}}(s) = \frac{1}{s + \frac{1}{C_{\text{ext}}(R_{\text{ext}}+R_1)}} \frac{V_{\text{ext}}R_1}{R_{\text{ext}} + R_1}$$

The input function transformed in temporal domain is

$$\mathscr{L}^{-1}\{V_{\text{in}}(s)\} = \frac{R_1}{R_{\text{ext}} + R_1} e^{-\frac{t}{C_{\text{ext}}(R_{\text{ext}}+R_1)}} .$$

As it can be seem in the equation, an impulse with exponential decay is obtained.

5.3 Amplifier transfer function

To extract the transfer function of the preamplifier, we used the fact that the sum of the currents at the entrance of the operational amplifier (V_n) is equal to zero:

$$\frac{V_n - V_{\text{in}}}{\frac{1}{sC_1}} + \frac{V_n - V_{\text{int}}}{R_2} + \frac{V_n - V_{\text{int}}}{\frac{1}{sC_2}} = 0$$

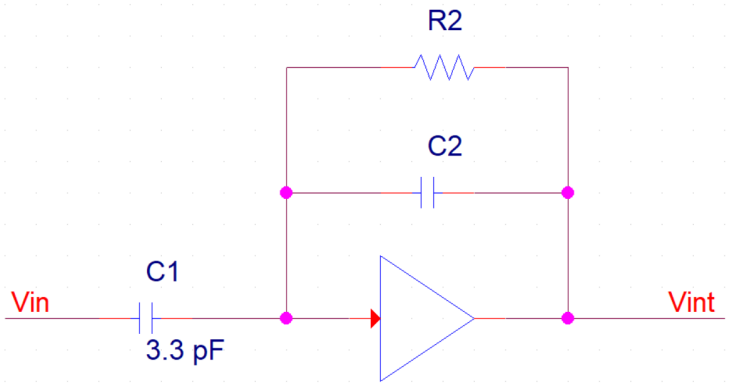

Figure 5.4: Integrator circuit

The operational amplifier has been hypothesized ideal: $V_n = 0$.

Therefore,

$$\frac{-V_{\text{in}}}{\frac{1}{sC_1}} + \frac{-V_{\text{int}}}{R_2} + \frac{-V_{\text{int}}}{\frac{1}{sC_2}} = 0$$

$$\Leftrightarrow V_{\text{in}} sC_1 + V_{\text{int}} \left(\frac{1 + sR_2C_2}{R_2} \right) = 0$$

Transfer function:

$$G_1(s) = \frac{V_{\text{int}}}{V_{\text{in}}}$$

$$\boxed{G_1(s) = \frac{-s\frac{C_1}{C_2}}{s + \frac{1}{R_2C_2}}}$$

5.4 Poles and zeros transfer function

With the aim of minimizing the effect of the undershoot, the book Knoll Radiation Detection Measurement propose a pole-zero cancellation circuit, shown in Figure 5.5. It is the one used in the ideal charge preamplifier.

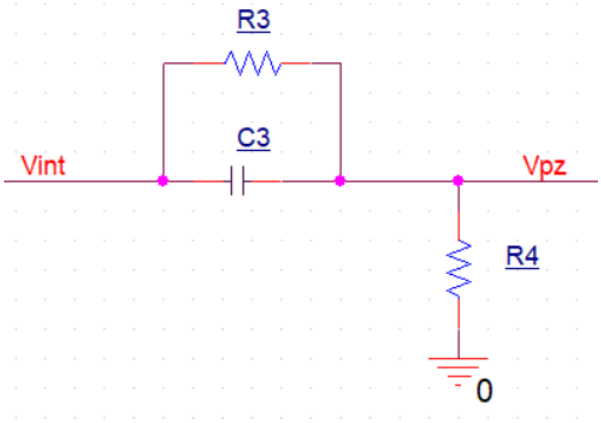

Figure 5.5: Pole-Zero circuit

Using Ohm's law we get:

$$V_{\text{int}} = I(R_3 \parallel C_3) + IR_4$$

$$\Leftrightarrow I = \frac{V_{\text{int}}}{R_4} \frac{\frac{1}{R_3C_3} + s}{\frac{1}{R_4C_3} + \frac{1}{R_3C_3} + s}$$

$$V_{\text{pz}} = IR_4$$

$$\Leftrightarrow V_{\text{pz}} = V_{\text{int}} \frac{\frac{1}{R_3C_3} + s}{\frac{1}{R_4C_3} + \frac{1}{R_3C_3} + s}$$

$$\boxed{G_2(s) = \frac{\frac{1}{R_3C_3} + s}{\frac{1}{R_4C_3} + \frac{1}{R_3C_3} + s}}$$

Now we have the transfer function of the pole and zero circuit, $G_2(s)$.

Multiplying both transfer function:

$$\boxed{G_1(s)\, G_2(s) = \frac{V_{\text{int}}}{V_{\text{in}}} \frac{V_{\text{pz}}}{V_{\text{int}}} = \frac{V_{\text{pz}}}{V_{\text{in}}} = \frac{-s\frac{C_1}{C_2}}{s + \frac{1}{R_2C_2}} \frac{\frac{1}{R_3C_3} + s}{\frac{1}{R_4C_3} + \frac{1}{R_3C_3} + s}}$$

5.5 System response

5.5.1 Exponential entrance

When there is an exponential entrance, $\frac{V}{s+\frac{1}{\tau_{\text{ext}}}}$ in Laplace, we get:

$$V_{\text{pz}}(s) = \frac{V}{s+\frac{1}{\tau_{\text{ext}}}} \frac{-s\frac{C_1}{C_2}}{s+\frac{1}{R_2C_2}} \frac{\frac{1}{R_3C_3}+s}{\frac{1}{R_4C_3}+\frac{1}{R_3C_3}+s} .$$

Using the anti-transformation we get the generic response of and ideal preamplifier for and exponential entrance:

$$\Rightarrow \mathscr{L}^{-1}\{V_{\text{pz}}(s)\} = -V\frac{C_1R_4\tau_{\text{ext}}(C_2R_2-C_3R_3)}{C_2(\tau_{\text{ext}}-C_2R_2)(C_2R_2R_3+C_2R_2R_4-C_3R_3R_4)}\, e^{-\frac{t}{C_2R_2}}$$

$$+V\frac{C_1R_2R_4(\tau_{\text{ext}}-C_3R_3)}{(\tau_{\text{ext}}-C_2R_2)(R_3\tau_{\text{ext}}+R_4\tau_{\text{ext}}-C_3R_3R_4)}\, e^{-\frac{t}{\tau_{\text{ext}}}}$$

$$-V\frac{C_1R_2R_3\tau_{\text{ext}}(R_3+R_4)}{(C_2R_2R_3+C_2R_2R_4-C_3R_3R_4)(R_3\tau_{\text{ext}}+R_4\tau_{\text{ext}}-C_3R_3R_4)}\, e^{-t\frac{(R_3+R_4)}{C_3R_3R_4}} .$$

5.5.2 Compensate response step entrance

Compensate means that we erase the effect of C_3R_3 exponential decay by forcing $C_3R_3 = C_4R_4$.

Exponential entrance

Step entrance: $\frac{V}{s+\frac{1}{\tau_{\text{ext}}}}$, (and $C_2 = C_3$)

$$V_{\text{pz}}(t) = \frac{V}{(R_3\tau_{\text{ext}}+R_4\tau_{\text{ext}}-C_3R_3R_4)}\left((C_1R_2R_4)\, e^{-\frac{t}{\tau_{\text{ext}}}} - \tau_{\text{ext}}(R_3+R_4)\, e^{-t\frac{(R_3+R_4)}{C_3R_3R_4}}\right)$$

Step entrance

Step entrance: V/s.

The response is:

$$V_{\text{pz}}(t) = -V\frac{C_1}{C_2}\, e^{-t\frac{R_3+R_4}{C_3R_3R_4}} .$$

If we suppose that $C_1 = C_2$, therefore:

$$V_{\text{pz}}(t) = -V\, e^{-t\frac{R_3+R_4}{C_3R_3R_4}} .$$

We could have taken the response for the exponential entrance and equalize $\tau_{\text{ext}} = \infty$ and we would have obtained the same result.

5.5.3 Conclusions of the system response

The conclusions of the calculations above are that if the pole-zero circuit is compensated in accordance with the constant R_2C_2 of the integrator, we obtain a signal that stops depending on the time constant of the integrator and it depends totally on the pole-zero circuit and especially on the R_4 resistance that goes to ground.

5.6 Sub-impulse analysis

5.6.1 Experimental data and simulation

To be able to later compare our theoretical model with values of the Canberra Model 2006 preamplifier output, we have arbitrarily chosen the values of the theoretical model with the help of the first UPC preamplifier model designed by Alfredo de Blas. In the following table, resistors are in ohms and capacitors in farads.

```
R1 = 65;
C1 = 3.3*10^(-12);
R2 = 100*10^(6);
C2 = 3.3*10^(-12);
Vin = 0.50;
R3 = 15000;
C3 = 22*10^(-9);
R4 = 2400;
```

I have simulated, with Matlab using the function `iplace()`, different system response with different time constant for the input, to be able to see the effect of the time constant on the sub-impulse. For this, I have plotted the sub-impulse value divided by the minimum value (maximum absolute value) of the response with the time constant of the input. And I have compare it with the experimental data. The results are shown in Figure 5.6. As it can be seen in the figure, the experimental results are not nailed to the results obtained with the theoretical model but at least we can say that the model resembles the reality. Therefore, if the decay constant is known, we could predict the sub-impulse the preamplifier generates. Hence we can say that the ideal preamplifier model find above is similar to

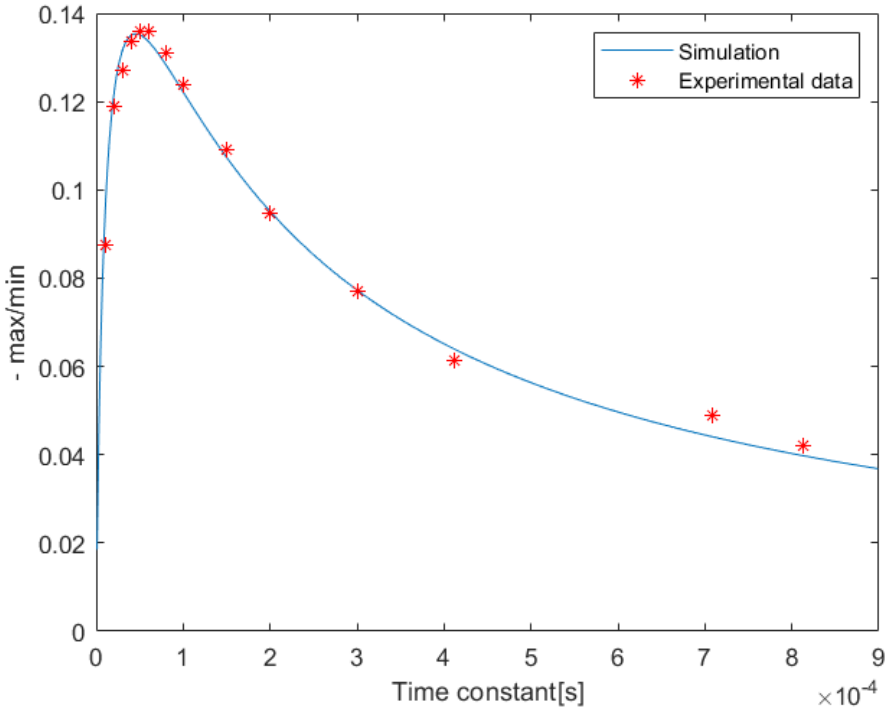

Figure 5.6: Comparison of the experimental sub-impulse with the model's sub-impulse

5.7 Preamplifier gain

The gain of the Canberra Model 2006 has been calculated experimentally to be able to compare it to the ideal transfer function and see if the ideal theoretical model goes in the right direction.

In this case, the gain only represents a constant that multiplies the transfer function. The experimental results can be seen in Figure 5.7. To make this experience the TG2000 function generator has been used to generate a step entrance. From the linear regression shown in Figure 5.7, it can be seen that the gain is 4.45. Therefore, in our transfer function, the gain factor will be $K = 4.45$.

In the figure below, Figure 5.8, there is a comparison of response obtained with the Canberra preamp with the response of the transfer function, again with Matlab and using the function `iplace()`. It can be seen the similarity is almost perfect. For this comparison the external circuit to make the exponential decay had this values: $C_{\text{ext}} = 363$ nF, $R_{\text{ext}} = 0$ and $V = 1 \times \frac{93}{93+50} = 0.65$ V.
In Figure 5.9 there are several more comparison. in most cases the similarity is almost perfect too.

In conclusion, this study presents quite satisfactory results although a simplified and ideal model has been used.

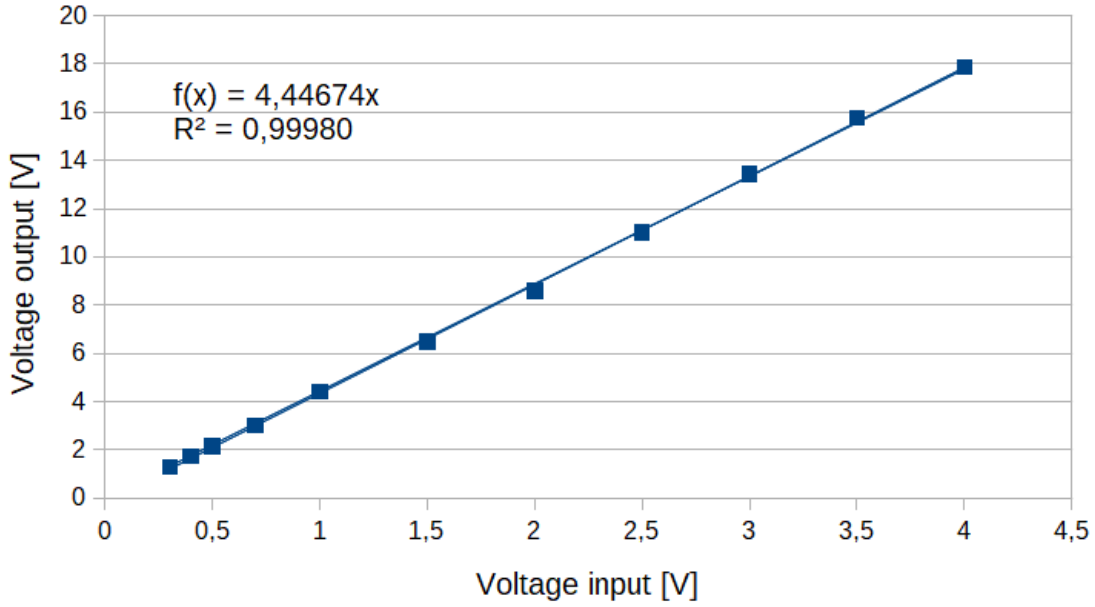

Figure 5.7: Canberra Model 2006 gain

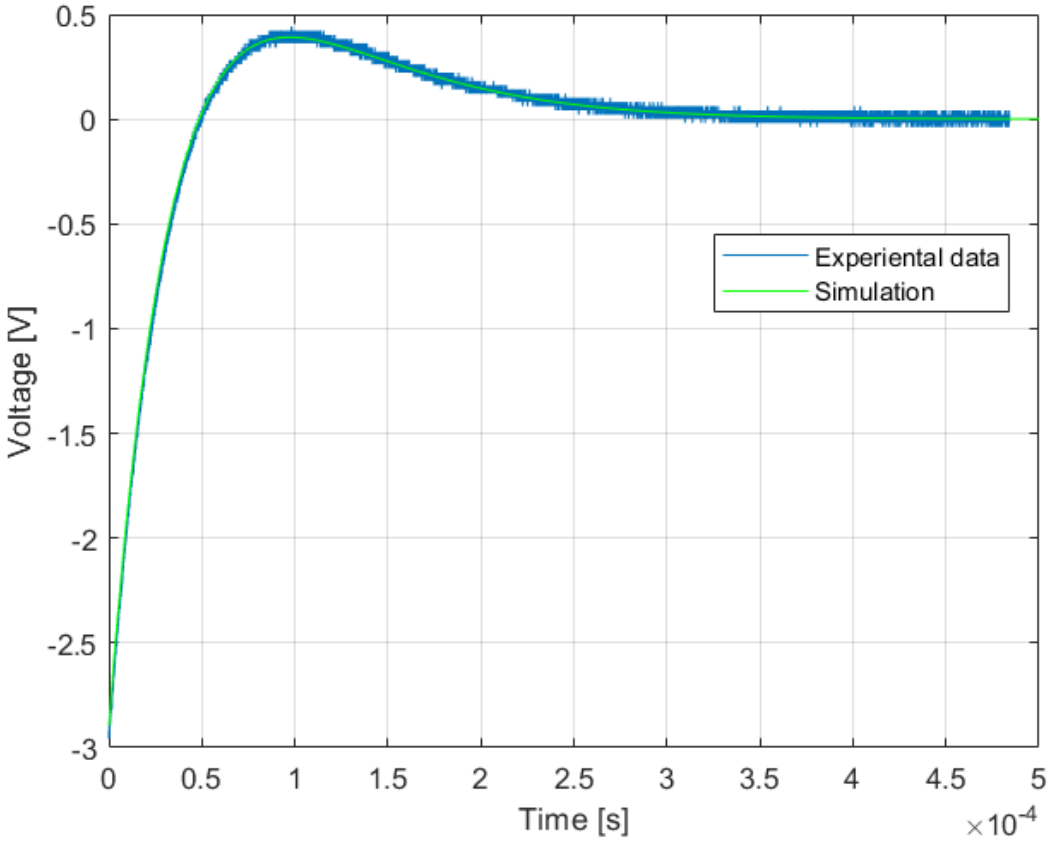

Figure 5.8: Preamp response comparison

5.8 Further investigation

As we said at the beginning, this chapter is a theoretical exercise that tries to understand the response, of and ideal charge preamplifier, to voltages at the 'test input'. So far, the transfer function obtained $V_{\text{out}}/V_{\text{in}}$ seems to work well. In future works it would be interesting to:

- Obtain the transfer function $V_{\text{out}}/Q_{\text{in}}$ since we detect charges and not voltages.
- Obtain the transfer function so that Q_{in} is at the input of the sensor and not the 'test input'.
- Derive the function to see if there is any more efficient combination of values to compensate the undershoot

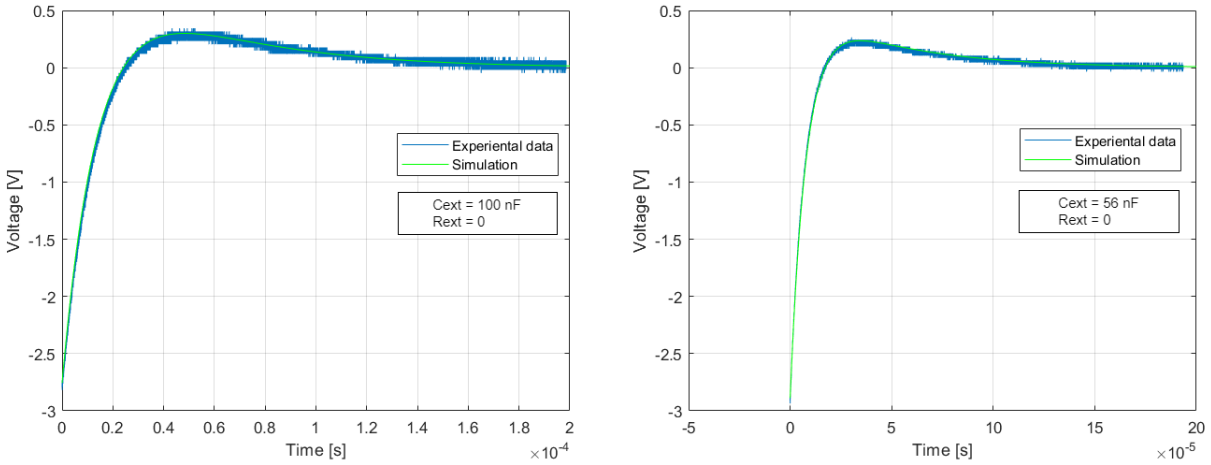

Figure 5.9: Preamp response comparison

Chapter 6

Project time line, project cost and environmental impact

6.1 Project time line

The time assigned to the project has been around five months. The Gantt chart is shown in the figure below to illustrate how the task were distributed during this period.

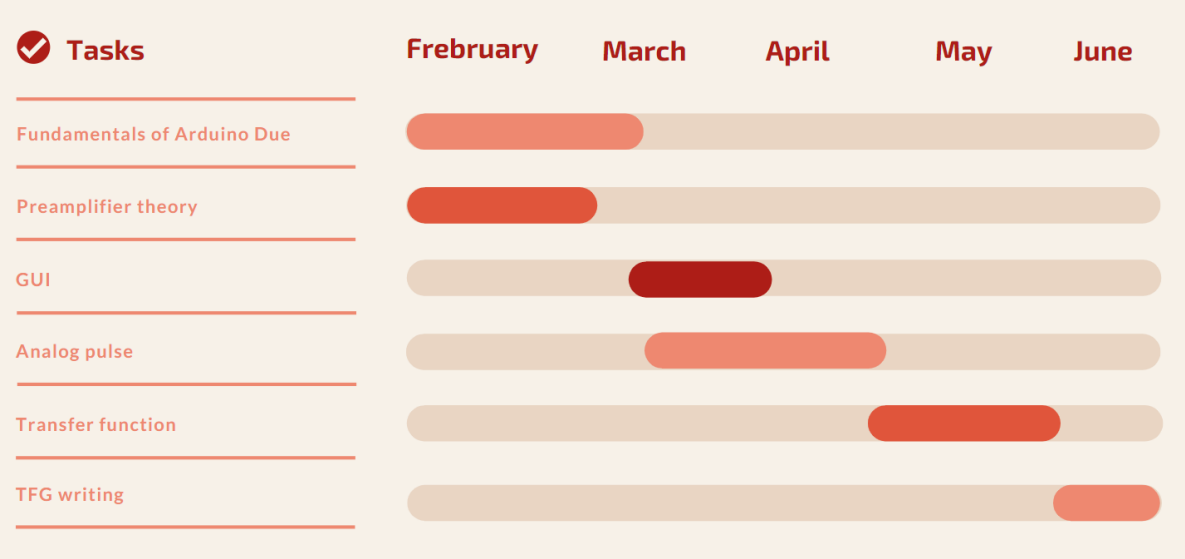

Figure 6.1: Gantt chart

As it can be seen in the chart, I first focused on learning how to use the Arduino microcontroller to be able to do the digital pulse, then we investigated the possibility of doing an analog pulse by connecting an external circuit to the Arduino, but seeing that the results were not good, it was decided to devote more time to the prediction of the preamplifier response making the transfer function of an ideal one.

6.2 Project cost

The cost of the development of a test bench can be itemized in two: the engineering cost and the material cost.

The engineering cost has been calculated based on the amount of hours spent on the project. The university approximates this value assigning the TFG a total of 12 credits, equivalent to 360 hours. It has been considered a salary of 25€/hour for a junior engineer. Therefore, the engineering cost is 9000€.

The material cost consist on all the components used for the physical development of the test bench. A summary of all components costs can be seen in the following table.

Concept	Cost [€]
Arduino Due	23
Resistors	6,6
Capacitors	4,2
Relay	3,3
VCN connector	2,1
PCB	3
Faraday cage	12
Total components	54,2
VAT (21%)	11,4
TOTAL	65,6

6.3 Environmental impact

The environmental impact of the project is almost non-existent. There was only electrical consumption, the one used for computers and for the test bench that can be fed through a small lipo battery. Also, paper was not used.

In addition, this bench test is designed to check the preamplifiers and digitizers that are used for the detection of ionizing particles. These equipments are fundamental to detect high doses of ionizing radiation harmful for the environment and therefore be able to act in case of anomaly.

Chapter 7

Conclusions

The digital pulses with exponential decay generated by the Arduino Due are surprising. It has been verified that using the peripherals of the microcontroller, small signal times delays are obtained through the DAC. These changes allow the Arduino Due to use it as a generator of pulses with exponential decay. In addition, a GUI has been made to simplify the configuration of the pulse. With the GUI, the user can modify the amplitude, the frequency and the decay time. Furthermore, a second impulse can be added which allows to test more in depth both the preamplifier and digital acquisition systems. Finally, the GUI can also read a text file that allows the user to enter the amplitude points he wants. Therefore, the objectives of making a test bench, low cost, configurable has been achieved. For future work it would be interesting making the GUI more user-friendly.

We wanted to go one step further and tried to make an analogue pulse. The Arduino was connected to an external circuit, made of passive components to give the exponential shape. The whole circuit was thought to be directly connected to the Detector Input. Unfortunately, the results were not achieved. The charge transmitted saturated the preamplifier and the noise was to high. For future work, we recommend using a charge terminator to control the charge of the Detector Input entrance and finding ways to attenuate the noise, for example shielding of cables, terminals connection to ground, a well-made ground connection, high quality PCBs, foresee the current loops, consider the possibility of line filters and a metallic box to block the the electromagnetic fields of the environment.

In order to add more functionalities to the bench test, we tried to predict the output of the preamplifier. For this, the transfer function of an ideal charge preamplifier has been made. The results of the transfer function were compared with the reponse of the Canberra Preamplifier, and we could observe that the results were very similar. Since the first results are positive, the possibility opens up to new investigations on the prediction of the preamplifier output signal. In further investigation, it is recommended to progressively go from an ideal model to a real one and change the input variable of the transfer function to charges (in coulombs) instead of voltage.

Bibliography

[Knoll,2010] Knoll, G. F. (2010). *Radiation detection and measurement.* John Wiley & Sons.

[Preamp] Canberra Industries, Inc. (2007). *Model 2006 Proportional Counter Preamplifier*.

[Atmel] SAM3X Atmel (2015). *SAM3A Series SMART ARM-based MCU datasheet.*

[Manual,419] Advanced Measurement Technology, Inc. (2002) *Model 419 Precision Pulse Generator Operating and Service Manual.*

[TG] Aim & Thurlby Thandar Instruments. *TG1000 & TG2000 Instruction Manual*

[dacc] Robert Kein (May 04, 2016). *Understanding and Using the SAM4S Digital-to-Analog Converter* Retrieved from https://www.allaboutcircuits.com

[WaveGen] Bruce Evans (December 24, 2017). *Arduino Due Arbitrary Waveform Generator* Retrieved from https://create.arduino.cc/projecthub