

Study of New Plastic Scintillator Materials for Future Detectors

Ana Matias^{1,a}

¹Faculdade de Ciências da Universidade de Lisboa, Lisboa, Portugal

Project supervisor: R. Pedro, R. Machado

October 8, 2024

Abstract. Plastic scintillators are widely used for radiation detection in both scientific research and industrial applications. In Particle and Nuclear Physics, detecting ionizing particles is essential for high-energy experiments like those conducted at large accelerators, such as the ATLAS experiment at CERN. These scintillators are effective and low-cost, emitting light when exposed to radiation. Improvements have been made to increase their light output and durability. In this study, a Python-based software was developed to upgrade a scintillator's characterisation system at LIP's Laboratory of Optics and Scintillating Materials (LOMAC). The new software addresses key limitations of the previous system by automating tasks, offering greater flexibility, and making the measurements more accurate. As a result, the process is now faster and the data is more reliable, helping to enhance the equipment used in research for high-energy physics experiments.

KEYWORDS: Light emission; Experimental equipment optimization; Automated measurement process; Python-based software

1 Introduction

The objective of this project is to upgrade the software used at LIP's Laboratory of Optics and Scintillating Materials (LOMAC) to improve the efficiency and accuracy of data analysis for light response measurements. The existing system, based on LABView, presented several limitations in terms of flexibility, automation, and data processing. To address these issues, a new Python-based software was developed, allowing a more efficient data acquisition and analysis. This software aims to enhance the overall workflow of the laboratory, making the measurement process faster, more reliable, and easier to use.

1.1 Tilemeter: Purpose and functionality

The Tilemeter is an experimental apparatus designed to measure the light emission of scintillator samples. Its primary role is to quantify the light emitted when a scintillator interacts with radiation, such as beta particles, making it an essential tool for evaluating scintillator performance in both research and industrial settings.

The Tilemeter operates by positioning a radioactive source above a scintillator sample and recording the light response using a Wavelength Shifting (WLS) fiber. This fiber collects the scintillation light and transmits it to a Photomultiplier Tube (PMT), which converts the weak light signals into electrical signals that can be measured. The Tilemeter has its own measurement system called "steps," where 1 cm is equivalent to 400 steps allowing for more precise positioning of the radioactive.

The system is integrated with the LABView software that controls the source positioning, light response data and processes the results.

1.2 Light Response Measurement Process

The measurement of light response follows a set of steps using equipment like a radioactive source, Wavelength

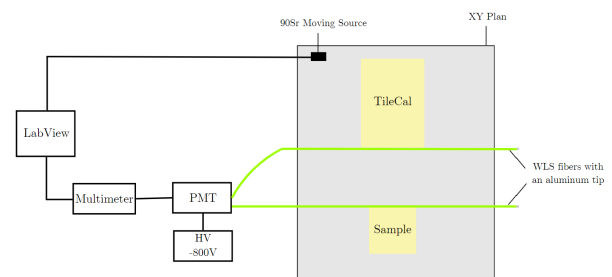


Figure 1: Experimental setup diagram of the Tilemeter showing the TileCal and Sample scintillators, both wrapped in Tyvek to improve fiber light collection.

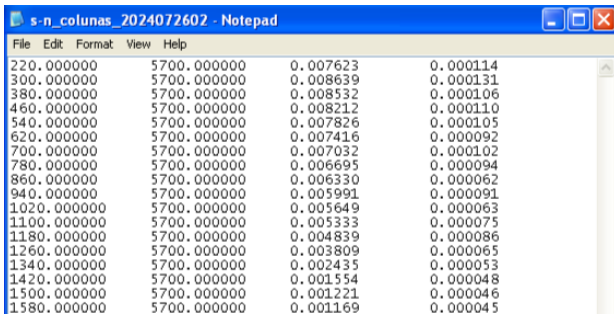
Shifting (WLS) fibers, and a Photomultiplier Tube (PMT), shown in the figure 1. The process is as follows:

- 1. Positioning the Radioactive Source:** The ^{90}Sr radioactive source is placed above the sample using a motor controlled by LABView.
- 2. Emission of Scintillation Light:** When beta particles from the source interact with the scintillator material, light is emitted and collected by the WLS fibers. Two distinct fibers are used: one for the sample and another for the reference scintillator (TileCal).
- 3. Light Collection and Detection:** Each WLS fiber transmits the collected light to a Photo Multiplier Tube (PMT), which detects and amplifies low light signals. The PMT is powered by a high-voltage -800 V supply and converts light into electrical signals. The two WLS fibers transport light independently to the PMT, ensuring that the light responses from both the sample and TileCal are measured separately.

^ae-mail: fc56664@alunos.fc.ul.pt

4. **Signal Measurement:** The electrical signal generated is measured by the multimeter, which integrates the signal over 400 ms. The multimeter is connected to both the computer and LABView, corresponding to each voltage reading with the position of the source.
5. **Noise Measurement:** Noise is measured at regular intervals and is subtracted from the light signal to improve accuracy.
6. **Data Acquisition and Control:** Throughout the measurement process, the LABView software continuously controls the step motor that moves the source, handles, and stores the data.

1.3 Data Acquired



x	y	Light Response [mV]	Standard Deviation
220.000000	5700.000000	0.007623	0.000114
300.000000	5700.000000	0.008639	0.000131
380.000000	5700.000000	0.008532	0.000106
460.000000	5700.000000	0.008212	0.000110
540.000000	5700.000000	0.007826	0.000105
620.000000	5700.000000	0.007416	0.000092
700.000000	5700.000000	0.007032	0.000102
780.000000	5700.000000	0.006695	0.000094
860.000000	5700.000000	0.006330	0.000062
940.000000	5700.000000	0.005991	0.000091
1020.000000	5700.000000	0.005649	0.000063
1100.000000	5700.000000	0.005333	0.000075
1180.000000	5700.000000	0.004839	0.000086
1260.000000	5700.000000	0.003809	0.000065
1340.000000	5700.000000	0.002435	0.000053
1420.000000	5700.000000	0.001554	0.000048
1500.000000	5700.000000	0.001221	0.000048
1580.000000	5700.000000	0.001169	0.000045

Figure 2: Example of an output file: The "s-n_colunas" file (short for signal-noise) shows results where the noise has already been subtracted. Lists the x and y coordinates, light response values, and standard deviations, in that order from left to right. The noise values are shown in another file with the same structure.

Using the data generated by the LABView software, we can create a plot of the run. In Figure 3 represents the light response (in blue) and the standard deviation of the noise (represented by error bars in red) as functions of the position along the x axis for the PEN #30 sample.

In this paper, most of the light response plots correspond to the sample PEN #30 Nov 23. This particular sample was chosen to analyze for a few reasons. Pure PEN samples generally exhibit the highest light response values. The "#30" in its name indicates that it was the 30th sample manufactured in November 2023. This particular sample had minimal defects, such as scratches or air bubbles, which ensures a more accurate and reliable analysis of its light response characteristics.

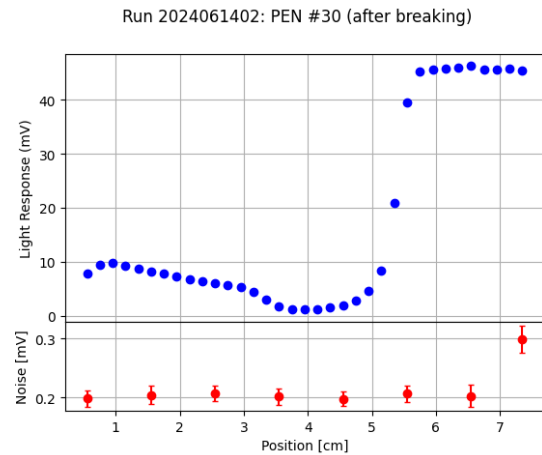


Figure 3: Light response and noise as a function of the distance traveled by the radioactive source, with data obtained with LABView software.

Light Response (Blue Data Points):

- Each point is an average of 30 measurements and represents the signal value after subtracting the most recent noise reading. The noise values are plotted in red below. Standard deviation bars are included, but they are not visible because the standard deviation values are very low compared to the light response values.
- **First Region (1 cm to 3.5 cm):** Lower light response with a peak of 10 mV, corresponding to the PEN #30 sample, whose physical size matches this interval (3 cm).
- **Second Region (Beyond 5.6 cm):** Higher light response, peaking at 47 mV, corresponding to the reference TileCal scintillator.
- The different slopes reflect the different morphologies of the scintillators (PEN sample: 2 mm, TileCal: 3 mm).
- The negative slope results from the increase in distance between the source and the WLS fiber, leading to light attenuation.

Noise Measurement (Red Data Points):

- Like in the light response plot, each point represents the average of 30 measurements; however, we can see the standard deviation bars since the scale is much smaller.
- The noise values remain consistently around 0.2 mV, with minimal variation between measurements. However, the last data point deviates, measuring 0.3 mV — a recurring anomaly seen in every run, suggesting a systematic issue rather than a random error. This will be discussed in more detail in Section 2.
- Noise measurements are updated every X steps to capture any fluctuations. In this plot, we can see that it was updated every 5 steps. Frequent updates help minimize the impact of noise on the final light response.
- Noise measurements are always taken at the same position, where the light response is minimal, to ensure a reliable baseline for accurate signal isolation.

2 Need for New Software / Limitations of LABView Software

The development of the new software arose from the limitations of the existing LABView-based system at LO-MAC. Although LABView is a widely used platform for data acquisition and experimental control, the system in place had several limitations that set back the efficiency and accuracy of light response measurements. Some examples of this limitation are as follows.

1. **Excessive Output Files:** LABView generated up to six files per run, even though only two were necessary, cluttering the workspace.
2. **No Parameter Tracking:** There was no automated system to save experiment parameters. Users had to log all "Run Parameters" manually in a notebook, leading to potential errors and inconsistencies.
3. **Manual Run Number Entry:** Entering run numbers for each experiment by hand increases the chance of mistakes and slows the workflow. For example, Run ID: 2024060301 refers to the first measurement (the last two digits, 01, show the order). This method can easily get messy, and it is easy to forget or skip a number. Keeping track of run numbers in a notebook and then entering them into the program for each new run can lead to more errors, especially in a fast-paced environment. This shows that a simpler process is needed.
4. **Inconvenient Data Visualization:** Data files had to be manually transferred to personal computers for analysis, which was time consuming and increased the risk of data loss.
5. **Rigid Measurement Control:** The system enforced a default of 30 measurements per position with no option to adjust this, leading to longer run times and less flexibility.
6. **Uncertainty in Run Duration:** The system did not provide an estimate of how long a run would take, making it difficult to plan and manage lab work effectively.
7. **Lack of Automation for Sequential Procedures:** LABView did not support the automation of sequential experimental procedures, requiring users to manually input parameters for each run, even if identical to previous ones.
8. **Inaccurate Final Noise Measurements:** A recurring issue was that the last noise measurement was consistently taken in an incorrect position, leading to abnormally low light response readings. As a result, the final data point was often unreliable.

9. **Single Position Readout:** To read the signal and standard deviation at a single position, a full run had to be completed, which was inefficient and time-consuming.

3 Development of a new Control and Acquisition software for the Tilemeter

This Python-based software, named `software_tilemeter_scan`, aims to overcome the limitations of LABView by offering more flexible, automated, and user-friendly experimental control.

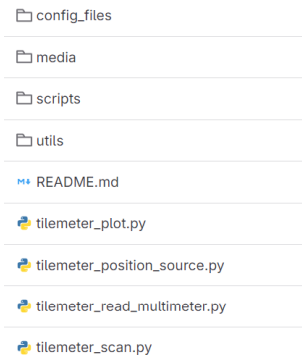


Figure 4: Structure of the package `software_tilemeter_scan`.

The package includes a `README.md` file and four main user macros (described in section 3.2): `tilemeter_plot.py`, `tilemeter_position_source.py`, `tilemeter_read_multimeter.py`, and `tilemeter_scan.py`. It also contains a utility `utils` folder that will be specified in the section 3.1, and additional directories for configuration files (with examples provided for different types of runs), scripts (including example files in `.bat` and `.ps1` formats), and a `media/` folder, which stores output files such as `.txt` files containing run data and generated figure plots.

3.1 `utils/` folder

This folder contains core code that are essential for the operation of the macros. Each core block has a different purpose within the workflow:

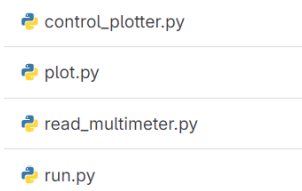


Figure 5: Files inside the `utils/` folder

• **control_plotter.py:**

Manages plotter movements, sets limits, and ensures the plotter stays within range with a 5-second delay. Stops the script if the coordinates are out of range.

• **plot.py:**

Creates 1D or 2D plots based on device movement. For the 1D plot, it identifies the direction of the movement and updates the y axis label of the plot accordingly. For 2D data, it generates heat maps of light response and noise, handles plot configurations, adjusts labels, and saves plots in a specified folder. Always produces two plots: light response and noise in plotter steps and other in centimeters.

• **read_multimeter.py:**

Communicates with a multimeter using the *pyvisa* library. Reads and processes signal measurements, subtracts noise to determine light response, calculates average and standard deviation, and optionally prints detailed measurement information.

• **run.py:**

Defines a Run class to manage experiments. Initializes instruments, calculates steps, tracks timing, generates a unique run ID, and stores experimental data and parameters.

3.2 Main User Macros

1. **tilemeter_scan.py:**

This macro controls the plotter's movements in both x and y directions, reads data from the multimeter and stores it in a new file. It produces a single output file that includes all scan data, parameters, run time, and optionally, the sample name if provided in the command line. Upon completion of the run, the macro generates and saves plots. The macro requires an "inputFile" argument that specifies the scan parameters.

There is a Verbose Mode implement where if flag is:

- **On:** Displays initial parameters and detailed point information, including data values and elapsed time intervals.
- **Off:** Shows a progress bar with point counts, elapsed time, and estimated time remaining.

Default values are used for missing or invalid parameters, with on-screen messages indicating any replacements.

The parameters for configuring the plotter's scanning process are specified in the input file (Figure 6) and include the following:

- "xSteps" and "ySteps": the number of plotter steps on the x axis and y axis independently;
- "xMax" and "yMax": the plotter movement interval on the x axis and y axis;
- "measurementsPerPoint": The number of measurements at every point for calculating the signal average at this position;
- "stepsUntilNoiseMeasure": The number of steps for updating the noise;

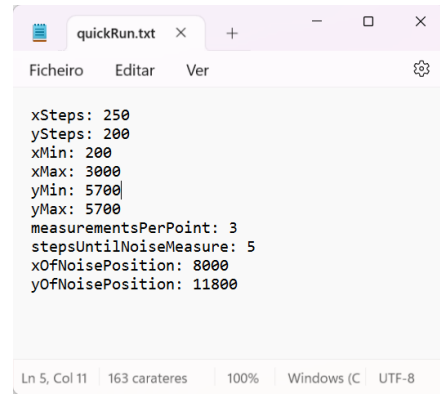


Figure 6: Example of a *inputFile* with the parameters for a quick run.

- "xOfNoisePosition" and "yOfNoisePosition": The noise reading position.

To run it, execute:

```
1 python tilemeter_scan.py "path-to-config-file"
2
3 # Example usage (Verbose OFF)
4 python .\tilemeter_scan.py .\initialParameters.txt PEN#30
5
6 # Example usage (Verbose ON)
7 python .\tilemeter_scan.py .\initialParameters.txt PEN#30 -h
```

This script provides a detailed solution that replaces all LABView functions with a more efficient and customizable approach. Designed for easy updates and flexibility, it manages the entire process, from configuration to data visualization.

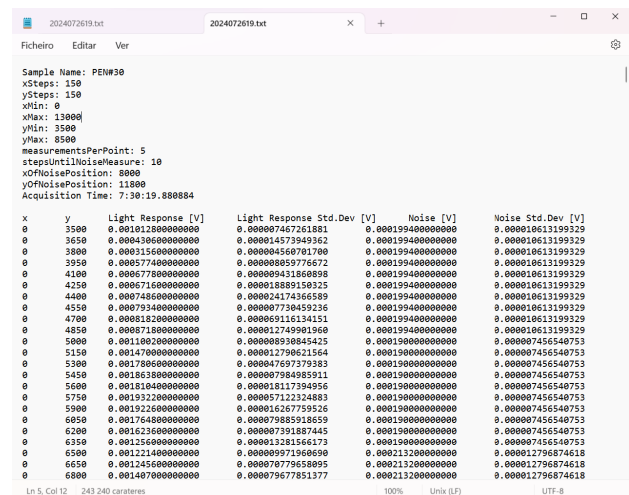


Figure 7: Example of a scan output file. The header includes a summary of all scan parameters, the sample name (if provided), and the run time. The data is organized into labeled columns.

2. tilemeter_plot.py:

This macro processes a data file (generated by *tilemeter_scan.py*) and produces plots for that run using the *plot.py* utility. This setup is beneficial because it allows you to visualize and modify data files while the plotter is executing another run, given that the plotter can only handle one run at a time.

To run it execute:

```
1 python .\tilemeter_plot.py "path-to-data-file"
2
3 # Example usage
4 python .\tilemeter_plot.py .\data\2024072306.txt
```

Here are some of the plots obtained with this package.

Plots from One Dimension Scans

In 1D it plots the light response according the position (either on x axis or y axis) and an error bar plot beneath showing the standard deviation of the noise at every point.

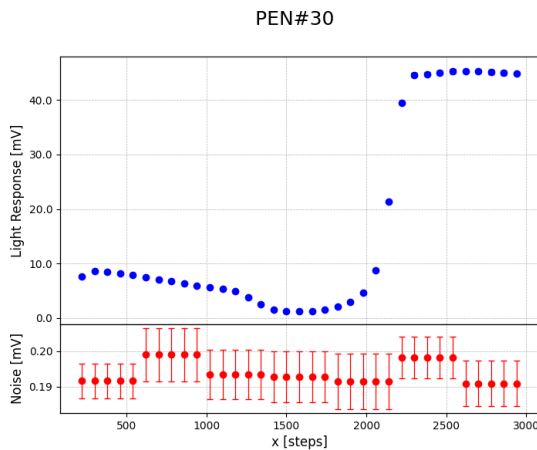


Figure 8: 1D plot with movement along the x axis.

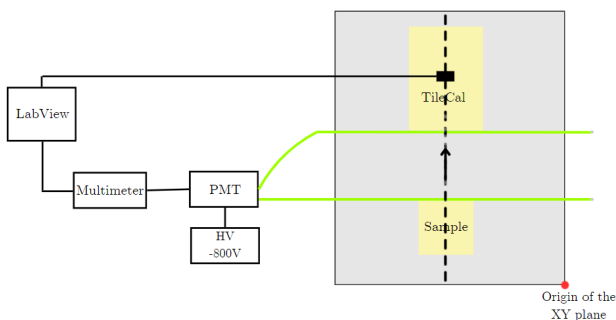


Figure 9: Schematic of the experimental setup with a dashed black arrow indicating the 1D movement along the x-axis, allowing the plotter to pass over both scintillators.

With the new software, the last noise measurement now falls within the range of previous measurements, ensuring that it is taken at the correct position.

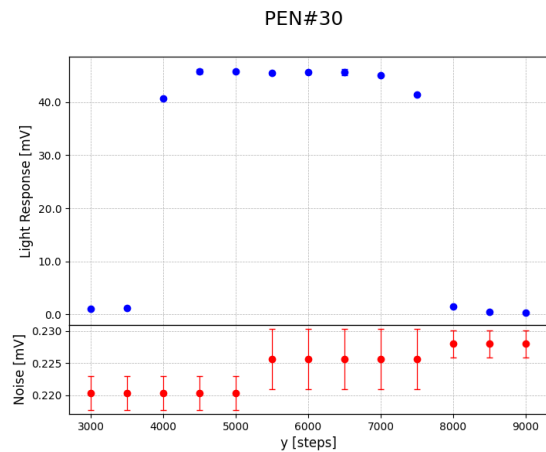


Figure 10: 1D plot with movement along the y axis.

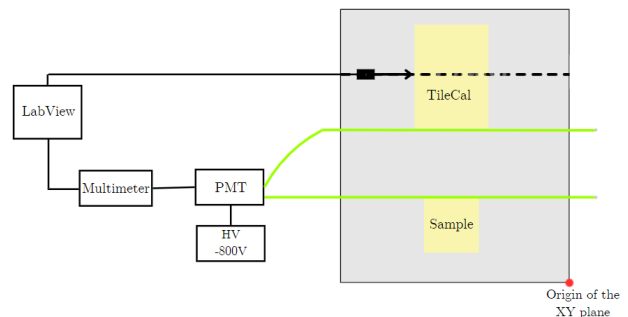


Figure 11: Schematic of the experimental setup, with a dashed black arrow showing the 1D movement along the y-axis, where the plotter is confined to the reference tile's width.

Plots from Two Dimensions Scans

In the 2D plot, the data is presented as a heat map, with the light response displayed on the left and the noise values on the right. Please note that both plots share the same color scheme, but have different scales.

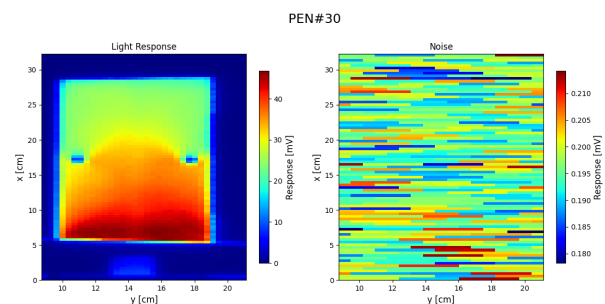


Figure 12: 2D plot where the left side displays the light response and the right the noise across the entire plotter plane.

The plot on the left shows the outline of the reference tile, located between $5 < x < 28$ and $10 < y < 18$, and the sample, which has a significantly lower light response and is located between $0 < x < 3$ and $13 < y < 16$. You can observe in the reference TileCal tile two characteristic holes. In both samples, a faint outline of the light response can be seen passing through the bottoms of the two plastic materials, which corresponds to the light emitted by the optical fibers. A notable feature of the reference tile is the increased light response near the optical fiber. This occurs because as the value of x increases, the position of the fiber moves further from the light source, resulting in reduced light absorption by the fiber. In contrast, when x is smaller, the fiber is closer to the source, leading to greater light gain.

The right plot illustrates the noise values, which update at various positions, represented by color variations that indicate changes in noise levels at specific intervals. While the color scheme matches the left plot, the scale is much finer, covering only a few decimal places of one millivolt. Despite its seemingly chaotic appearance, the noise values remain consistently low and similar throughout the plot.

3. `tilemeter_position_source.py`

This macro takes as two position arguments (x and y) from the command line and moves the plotter with the source to the specified coordinates.

To run it execute:

```
1 python .\tilemeter_position_source.py "x-
   position" "y-position"
2
3 # Example usage
4 python .\tilemeter_position_source.py 8000 11800
```

4. `tilemeter_read_multimeter.py`

This macro reads the signal and the associated standard deviation at the current plotter location without the need to perform a run.

To run it execute:

```
1 python .\tilemeter_read_multimeter.py
```

All macros include a help flag "-h" for user assistance.

4 Results and Conclusions

The development of the new Python-based software significantly improved the efficiency and accuracy of data analysis, effectively addressing the limitations of the previous system. By automating tasks and enhancing data visualization, this software improved the reliability of measurements and facilitated a more streamlined workflow in the laboratory. Furthermore, the switch to Python, a widely used programming language, makes the system more accessible to new users. This makes it easier for new users to adapt to the software, resulting in a shorter learning curve and smoother integration into lab workflows.

5 Acknowledgements

I am grateful to everyone who contributed to my work and my future. I am especially grateful to Rute Pedro and Rudnei Machado for their guidance and support throughout my two months at LIP, as well as for their invaluable mentorship and encouragement. I also extend my thanks to Luís Gurriana and Agostinho Gomes for their help with technical problems and for sharing their expertise. Finally, I am thankful for the collaborative environment at LIP, which greatly improved my learning experience.