



DEPARTMENT OF PHYSICS

DEVELOPMENT OF AN ENERGY METER FOR NANOSECOND LIGHT PULSES

Hampus Elmteg

Bachelors thesis in Physics
2024, 180 Credits

Department of Physics, UNIVERSITY OF GOTHENBURG
Gothenburg, 11th June 2024

Abstract

The aim of this bachelor project was to develop an energy meter for nanosecond laser pulses by utilizing a microcontroller and thereby creating an Embedded Measurement System (EMS) as a cost-effective alternative to commercial energy meters. The sensor used was a biased photodetector operating in a linear regime of output power with respect to input power. The development involved using an oscilloscope to analyze of potential circuits, and then following up by developing a program on a Teensy 3.2 microcontroller to replace the oscilloscope. The results indicate that this is a viable method for measuring variations in energy of nanosecond light pulses, as the output signal in voltage is directly proportional to the pulses' energy content if the photodiode is in its linear regime.

Sammanfattning

Detta kandidatarbete syftade till att utveckla en energimätare för nanosekund-laserpulser genom att använda en microcontroller för att skapa ett inbyggt mätssystem som ett lågkostnads alternativ till kommersiella mätare. Detektorn som användes var en bakspänd fotodiod för dess linjära egenskaper mellan utgående ström och inkommande effekt. Utvecklingen innefattade analys med oscilloskop för att testa potentiella kretsar, vilket sedan följdes upp med utvecklingen av ett mätprogram för en Teensy 3.2 mikrocontroller. Resultaten indikerar att detta är en metod som fungerar för att mäta variationer i energiinnehållet för nanosekund-laserpulser, eftersom den utgående signalen är direkt proportionell mot pulsens energiinnehåll om fotodetektorn är i den linjära regimen.

Contents

1	Introduction	1
2	Theory	2
2.1	Lasers	2
2.2	Photodiodes	3
2.3	Microcontrollers	4
2.4	Electrical Components and RC-circuits	5
3	Method	8
3.1	Requirements	8
3.2	Analog Circuit	8
3.3	Microcontroller and Software	11
3.4	Measurement Setup	13
4	Results	18
4.1	Measurements	18
5	Analysis and Discussion	28
5.1	Analysis	28
5.2	Discussion	29
	Bibliography	30
	Appendix	I

Abbreviations

μ C Microcontroller

ADC Analog-to-Digital Converter

DAC Digital-to-Analog Converter

EMS Embedded Measurement System

ISR Interrupt Service Routine

LSB Least Significant Bit

MSB Most Significant Bit

SAR Successive Approximation Register

SNR Signal-to-Noise Ratio

UI User Interface

Chapter 1

Introduction

Light pulses in the nanosecond regime are common in the field of high resolution spectroscopy. The lasers are generally very stable but there are variations in the energy from pulse to pulse. Since the energy is used for calculations and calibration of e.g. Second Harmonic Generators these variations are of great interest. Tracing the variations in energy also enables corrections and reductions of the Signal-to-Noise Ratio (SNR) in spectra. The aim of this thesis is to explore the opportunity for developing such an energy meter using simple circuitry and a Microcontroller (μC) to enable the use of serial communication, and through that create a User Interface (UI) to process the data and use it for the previously mentioned topics.

Chapter 2

Theory

The following sections introduce the reader to relevant information for the scope of this thesis regarding lasers and then moves on to describe microcontrollers and some electrical components.

2.1 Lasers

Lasers provide a beam of light by utilizing the concept of stimulated emission where an atom or molecule becomes excited and then returns to its ground state, emitting a photon. By utilizing this property and the fact that the energy levels are quantized, one can generate a high intensity beam. This has applications in many fields of science, such as spectroscopy, where it allows us to study atoms and molecules with high precision. There are different types of lasers, such as solid-state lasers which have a very narrow wavelength spectra, and provide short pulses down to the femtosecond range.[1] Another type of laser is the dye laser. Its main advantage is that the user can tune the wavelength by switching the dye used, allowing for a wide spectral range compared to the solid-state lasers. They can either be designed to be pumped by another laser or by a flashtube. The stability of lasers and optical components can be reduced by thermal and mechanical instabilities.

The properties of lasers can also make them harmful, therefore great care has to be taken in order to avoid stray beams and safety equipment such as safety goggles that filter the laser light.

2.2 Photodiodes

A common choice for detecting and characterising optical sources are photodiodes. A photodiode is a semiconductor device in which the PN-junction is irradiated releases free electrons, causing a current to flow. In a photodiodes linear regime the current that is created due to the incoming light, I_{PD} , is linearly dependent on the response of the diode, $R(\lambda)$, which in turn is dependent of the wavelength λ and the power P of the pulse,

$$I_{PD} = R(\lambda)P.$$

Figure 2.1 presents the response curve, i.e. the diodes dependency on wavelength, for three biased, meaning they have a voltage applied over them, Si detectors from Thorlabs.

The wavelength also effects the response time of the diode, t_r . [2] This change in response time can be understood by the absorption coefficient which decreases as the wavelength increases. This is due to the fact that the probability of absorption is lower, thus the photon will on average travel further in the material. Ideally the charge carriers are created in the depleted region of the biased semiconductor, there the electric field quickly accelerates these charges, providing the photocurrent I_{PD} . Operating the detector under a bias changes the characteristics of the depleted region. When operating under a reverse bias the width of the depleted region is increased, lowering the response time, leading to a linear relationship between the optical power and the photocurrent. Biasing the detector does however have a few disadvantages, most interesting for this work is the so called dark current that flows in the circuit because of the applied voltage, which leads to the total current from the detector being the sum of dark and photocurrent. To be able to get an accurate reading from the photodiode this dark current has to be small relatively to the photocurrent.

The linear regime of a photodiode depends on several factors, such as the load resistance, reverse bias voltage and responsivity. [4] The output voltage generated by the photocurrent can be approximated as a power-voltage-sigmoid curve, where the lower end is limited by the dark current and noise, such as Johnson-Nyquist noise from thermal movement of charge carriers which increases with the load resistance and the noise inherent in the measurement system. The maximum voltage is limited by the reverse bias of the detector, and the steepness of the curve increases with resistance.

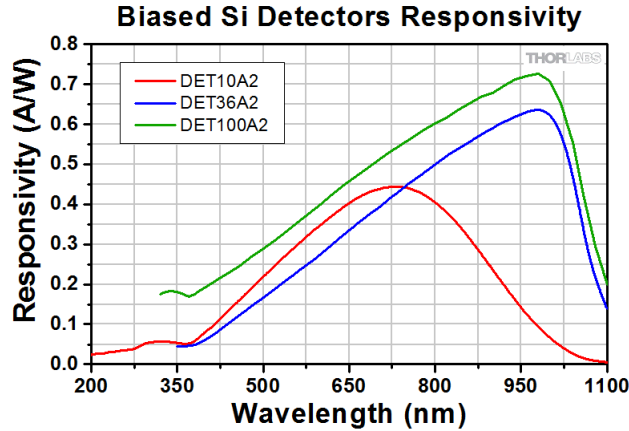


Figure 2.1: The response curves for the different types of biased Si detectors from Thorlabs.[3]. The responsivity is low for the UV-region and then starts rising at about 350 nm. The DET10A2 used in the setup has a lower maximum than the other two detectors, but is also able to detect wavelengths down to 200 nm.

2.3 Microcontrollers

A Microcontroller is a miniature computer in the format of an integrated circuit, designed to function as a stand-alone device.[5] It has a processing unit, memory and I/O hardware, dependent on the type of situation it is designed for. Today microcontrollers are all around us, the average home has around thirty of them, as they enable smart technologies. When interfacing sensors to a μC , the system becomes an Embedded Measurement System (EMS).[6] An advantage of digital systems is their ability to use interrupts. Interrupts are used to divert the normal flow of the program, and call an Interrupt Service Routine (ISR) before returning to its original state. Interrupts can be triggered by the hardware, enabling time sensitive code to run. On the other hand, time sensitive code, such as serial communication which is not triggered upon interrupts, do not respond well to being interrupted. So a well written program will disable and enable interrupts to protect such part of the code.

In many applications there is a need to read a voltage, a schoolbook example is when you want to know the current that flows through a resistor. The way that most microcontrollers determine the voltage level is by utilizing a Successive Approximation Register (SAR), which is an iterative algorithm that find the voltage level by changing the reference voltage of a comparator.[7] This reference voltage is supplied by a Digital-to-Analog Converter (DAC) which is controlled by the processor. It starts off by turning on the Most Significant Bit (MSB), if the the reference voltage

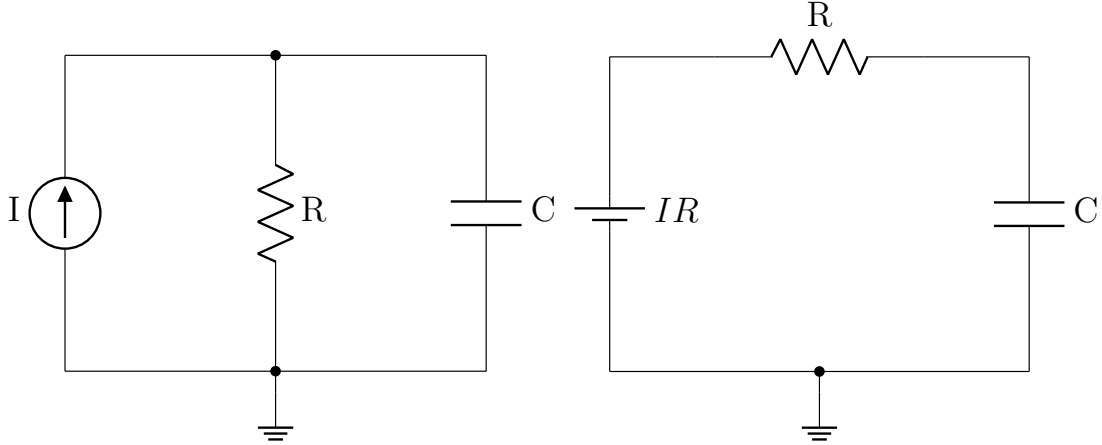


Figure 2.2: A parallel RC-circuit with a current source is equivalent with a series RC-circuit according to Thévenin's theorem.[8]

then exceeds the input the comparator will cause the MSB to flip again, and the next bit turns on. In this manner the SAR can approximate with an error of the Least Significant Bit (LSB).

2.4 Electrical Components and RC-circuits

While a μC is able to use digital techniques, available to the user in the program sketch, in order to manipulate signals, there are some things that are easier to do with analog electronics. An example of this is stretching a short pulse in the time domain. A simple way to extend a pulse in time is to use what is known as an RC-circuit. The left circuit in figure 2.2 shows an RC-circuit with a current generator, it is mathematically equivalent to the RC-circuit shown to the right, when the voltage source is set to IR . This allows us to calculate the voltage across the capacitor in the parallel circuit by using the common expression for the capacitor voltage in the series version.

The voltage over the capacitor u_C in the series RC-circuit after a time t with a voltage of $E = IR$ and time constant $\tau = RC$ and can be calculated as

$$u_C = E(1 - e^{-\frac{t}{\tau}})$$

$$u_C = IR(1 - e^{-\frac{t}{RC}}).$$

If $t \ll \tau$ this expression can be simplified using a Taylor expansion,

$$u_C = IR(1 - (1 - \frac{t}{RC} + \mathcal{O}(-\frac{t^2}{RC})))$$
$$u_C = IR(\frac{t}{RC} - \mathcal{O}(-\frac{t^2}{RC})).$$

Ignoring the higher order terms the results in the final expression,

$$u_C = \frac{It}{C}$$
$$u_C = \frac{Q}{C}.$$

This means that if the approximation $t \ll \tau$ is valid, then the voltage over the capacitor will be linear to the charge produced by the current source.

Another component of interest is the Operational Amplifier, or simply op-amp, which is a DC-component that uses a differential input to produce a signal. [9] It can be coupled with resistors and capacitors to vary the characteristics of the amplifier, such as trans impedance amplifiers, inverting or non-inverting amplifiers. A non-inverting amplifier consists of two resistors and is shown in figure 2.3. Following from the two golden rules of op-amps: firstly that there is no flow of current into the op-amp and secondly that with negative feedback the op-amp will equalize the two inputs. As current flows from a higher potential to a lower, for the non-inverting op-amp, this results in the output voltage is amplified by a gain factor $A = 1 + \frac{R1}{R2}$.

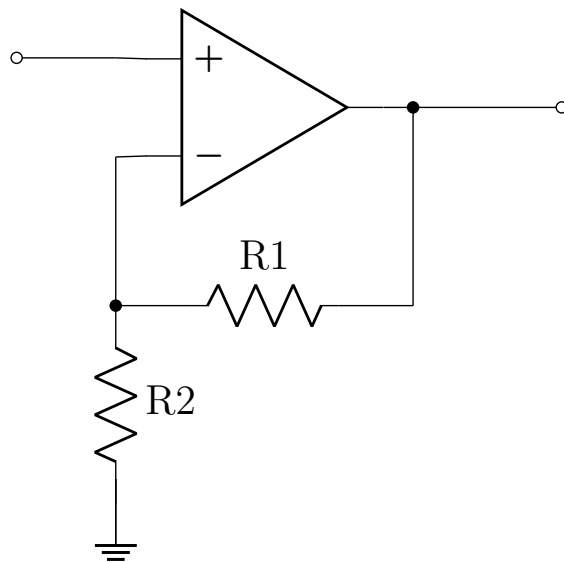


Figure 2.3: Schematic of a non-inverting amplifier. The amplifier will have a gain of $A = 1 + \frac{R1}{R2}$

Chapter 3

Method

The following chapter outlines the requirements that the EMS has to fulfill, the hardware and software used, as well as the setup it was tested in.

3.1 Requirements

The laser pulses that the EMS was tested and developed on were generated by a LAMBDA PHYSIK FL2002 dye-laser which was pumped with the second harmonic from a Nd:YAG laser. The setup delivers pulses with a width of 20 to 30 ns at a frequency of 10 Hz. Since dye lasers have a wide spectral range it follows that the detector has to cover a large range to avoid having to change it when switching dye. The implemented detector was a DET10A2 from Thorlabs, which covers a large wavelength range (200 to 1100 nm). For the EMS there was also a requirement for a User Interface (UI) so it could be integrated into different applications.

3.2 Analog Circuit

The signal from the photodiode when displayed on an oscilloscope is shown in figure 3.1. It had a width of approximately 20 ns, so it had to be stretched in order for us to be able to read it.

The circuit used in the final variant of the meter is shown in figure 3.2. Earlier stages consisted of only a parallel RC-circuit. The photodetector connects to the left side of the circuit, generating a small current once irradiated. The pulse is then stretched in time by the parallel RC-circuit consisting of R1 and C1, decaying with a time constant $\tau = R_1 C_1 = 1 \text{ M}\Omega \cdot 100 \text{ pF} = 100 \mu\text{s}$ and generating a voltage that

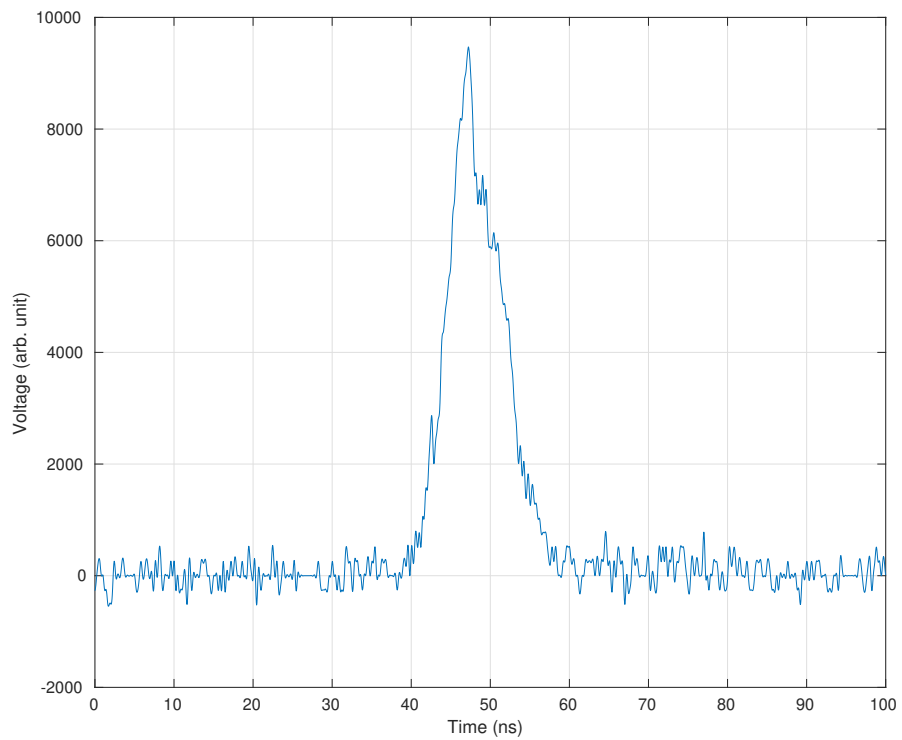


Figure 3.1: The pulse from the photodiode when displayed on an oscilloscope, the width is approximately 20 ns. The y-scale is proportional to the voltage.

is then amplified by the LF351 op-amp, in a non-inverted configuration which leads to a gain of $A = 1 + \frac{R3}{R2}$. The time constant was chosen as such to maintain the voltage long enough for the microcontroller to be able to read, while at the same time being able to completely discharge before the next pulse. The signal has then been processed in such a way that it can be measured with a high resolution, without removing the linear relationship between it and the pulse energy.

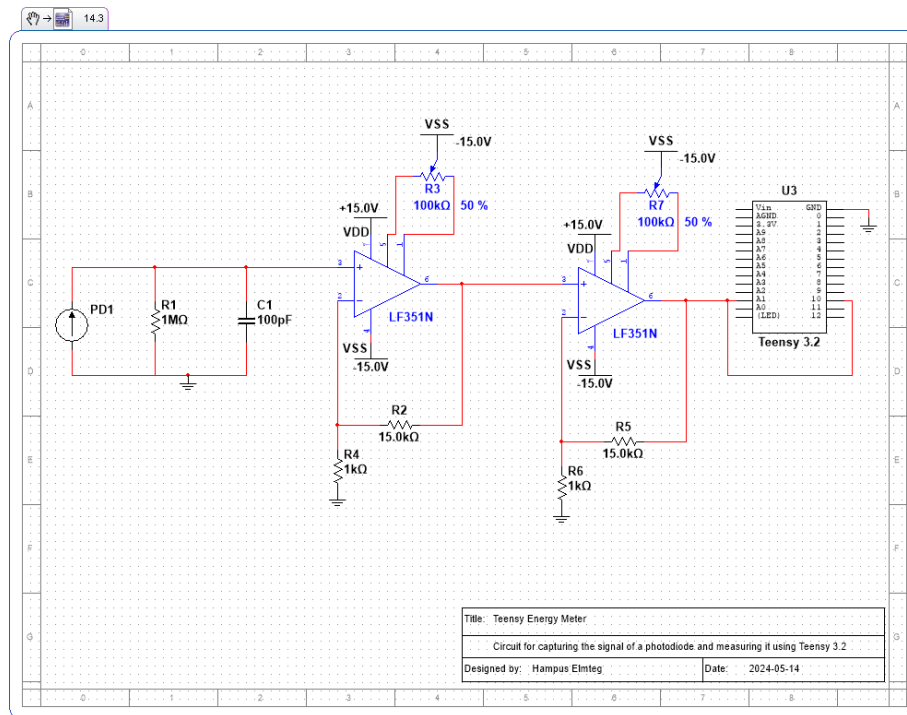


Figure 3.2: The circuit used for analog processing of the pulse. The short pulse enters on the left, and is then stretched by the R1 and C1. The two LF351 op-amps are configured as non-inverting amplifier using R2 and R4 resp. R5 and R6, resulting in a gain of $A = (1 + \frac{R2}{R4})(1 + \frac{R5}{R6})$. The result is a pulse that is stretched in time and amplified in order to be detected.

The stretched pulse can be seen in figure 3.3, the peak now last for several microseconds, and is suitable for reading.

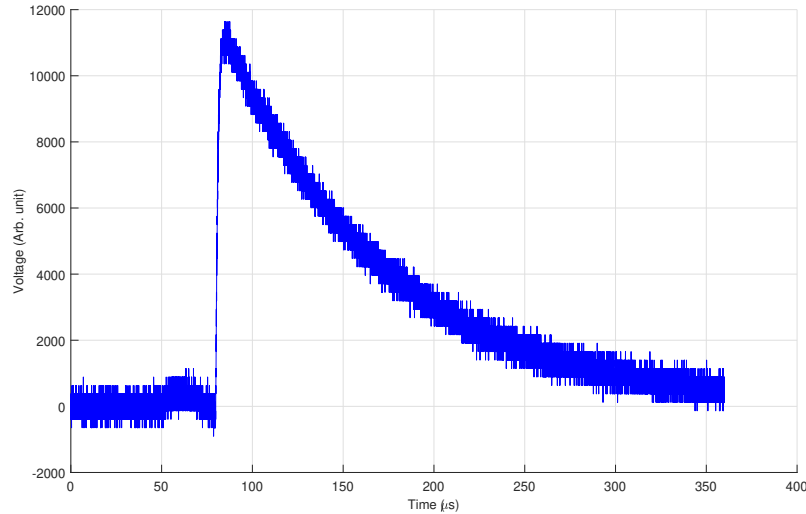


Figure 3.3: The pulse from the circuit shown in figure 3.2. The y-scale is proportional to the voltage.

3.3 Microcontroller and Software

The μC used was a Teensy 3.2 Development Board based on ARM Cortex-M4. It utilizes the same programming environment as the popular Arduino and has a large collection of libraries. [10] One of special interest for this project was the ADC library by Pedro Villanueva. [11] The library allows for easy customization of the reading of analog signals, i.e. the voltage present on a pin, allowing the user to set resolution, voltage reference and the number of samples to average over when reading. The ADC has a quantization error, when the resolution is ≤ 13 bits, of ± 0.5 LSB. Teensy 3.2 also features internal comparators than can utilize the internal 6-bit DAC, ranging from 0 to 3.3 V, as an internal reference, and when the external voltage rises higher than the internal reference it can trigger a hardware interrupt, followed by an ISR. Figure 3.4 shows how the comparator switches to a high state once the voltage has been read. The peak voltage however is delayed by several microseconds, prompting the need for a delay in the reading of the pulse once the interrupt was triggered. This was done using the IntervalTimer object which then could trigger the interrupt that reads the signal. In high-speed mode the comparator is specified to have a propagation delay of 200 ns, i.e. the time between the signal passing the threshold and the comparator switching its state. In contrast to the ADC there is no library for

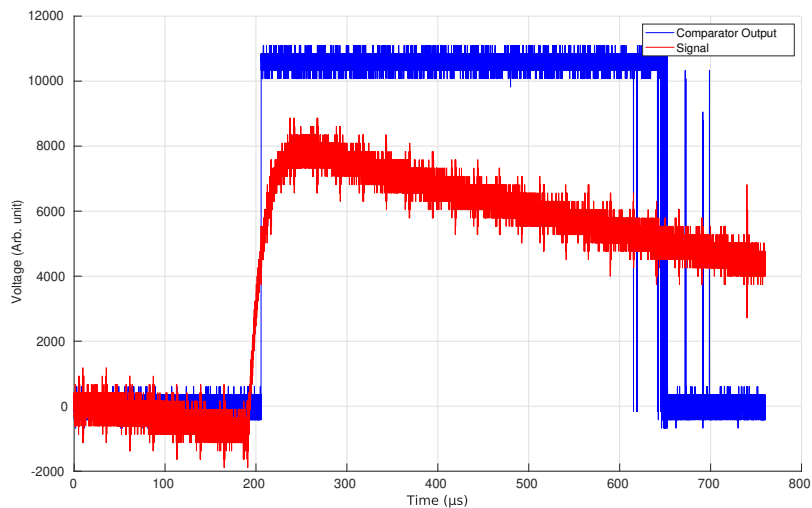


Figure 3.4: The stretched pulse, in red, and the comparator output, in blue, shown together as displayed on an oscilloscope. The y-scale is proportional to the voltage. The stuttering observed at the end of the curve was not present during the experiments. The y-scale is proportional to the voltage.

the comparators so they have to be set up by flipping bits in their control registers.[12] Since interrupts can change variables declared as volatile and thereby cause errors in time sensitive parts of the program, atomic blocks were used to protect the values of variables combined interrupt disables/enables for time sensitive code such as serial communication.

Since Teensy has Serial USB communication, it was utilized when creating the UI. As MATLAB has support for object oriented serial communication, this was encapsulated into the class, by including the serial port object as an attribute. Other than the constructor, the class has a public method for requesting one or more values from the μC . To create it, a private method for getting the current value was implemented. When several values are requested the MATLAB instance pauses for 100 ms before requesting a new value. To prevent a value from being sent twice the code in Teensy sends a zero value, since it has not read a new value before the previous request. Methods for changing the threshold and delay parameter were also implemented. The code used on Teensy and the UI in MATLAB can be found in the appendix.

3.4 Measurement Setup

The experimental setup can be seen in figure 3.5. The two lasers which were used to create the beam are here simplified into a single source, emitting a beam with a wavelength of around 600 nm. To have a variable attenuation Glan-Taylor prisms (GT) were used. Since the laser light was polarized and rotating the prisms shift the ratio of the p- and s-polarization, it attenuated the beam. The first one was on a rotational stage and the second one adjusted so the signal ratio during first peak, i.e. the 180 degree rotation, was as constant as possible. Starting out only one prism was used, figure 3.6 shows the signal when the prism was rotated, as it is evident from the figure the beam had to be cleaned further which prompted the use of a second one. Figure 3.7 shows the signal when two prisms were used. Following the polarizer there was a beamsplitter that enabled the use of another energy sensor to have a known reference when measuring the energy with our own energy meter. The sensor was a pyroelectric sensor from Gentec interfaced via Ulink, converting the signal to USB, that had a large energy range and high resolution. The photodiode was then connected either directly to the oscilloscope where the pulse was integrated using MATLAB, or to the analog circuit in figure 3.2, which in turn was connected to either an oscilloscope, for initial measurements, or a Teensy μC for testing.

The signals from the Gentec sensor and the EMS were interfaced via USB to MATLAB. The oscilloscope, either with the detector plugged straight in or through

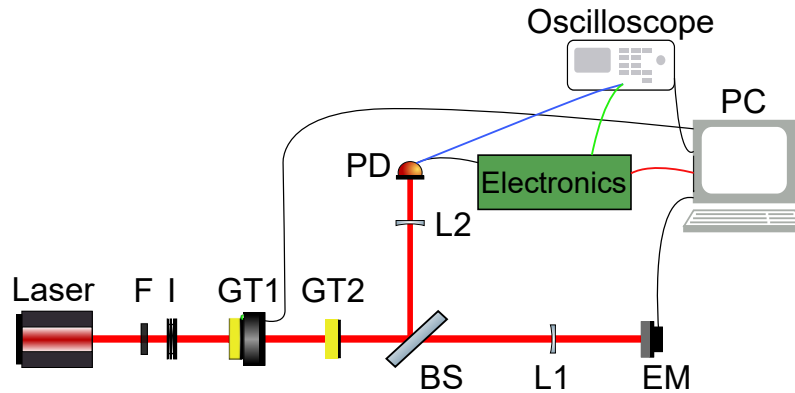


Figure 3.5: The setup to test the measurement system. The source of the pulses was a dye laser pumped by a solid-state laser. The beam was first attenuated using a filter F, followed by an iris I that could attenuate the beam further by decreasing its radius. The beam then passes through two Glan-Taylor (GT) Prisms that were rotated to attenuate the polarized beam. The first prism was placed in a rotational stage that was controlled by an electric motor. The second one was added to further clean the beam, due to the ratio between the beams varying greatly over the rotation of the first prism. The lenses L1 and L2 were plano-concave lenses to spread the beam over the entire detector material. The signal from the photodiode (PD) was either connected directly to the oscilloscope, or to electronics. The signal was acquired with either an oscilloscope or Teensy. They were then connected to a computer with MATLAB in order to plot the signals.

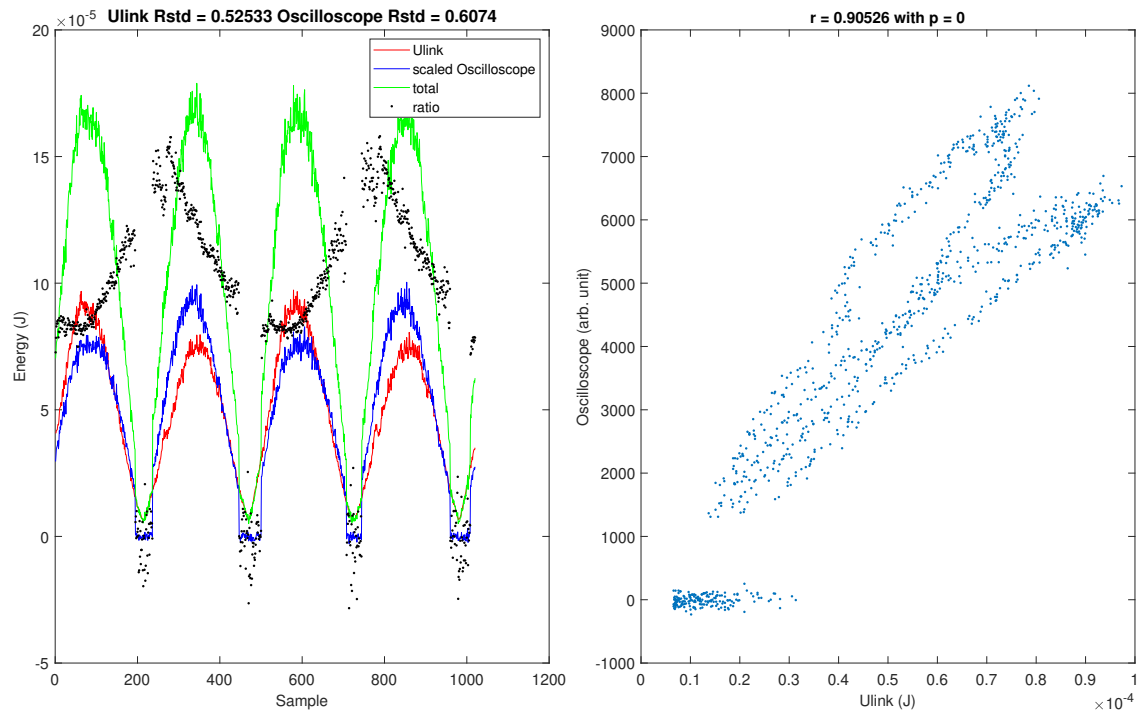


Figure 3.6: The signal from the photodiode integrated using the oscilloscope and MATLAB compared to the signal from Ulink. It is clear that the ratio between the two split beams was not constant when the motorized prism was rotated, resulting in the second prism being added. The ratio is scaled with a factor of 10^{-4} .

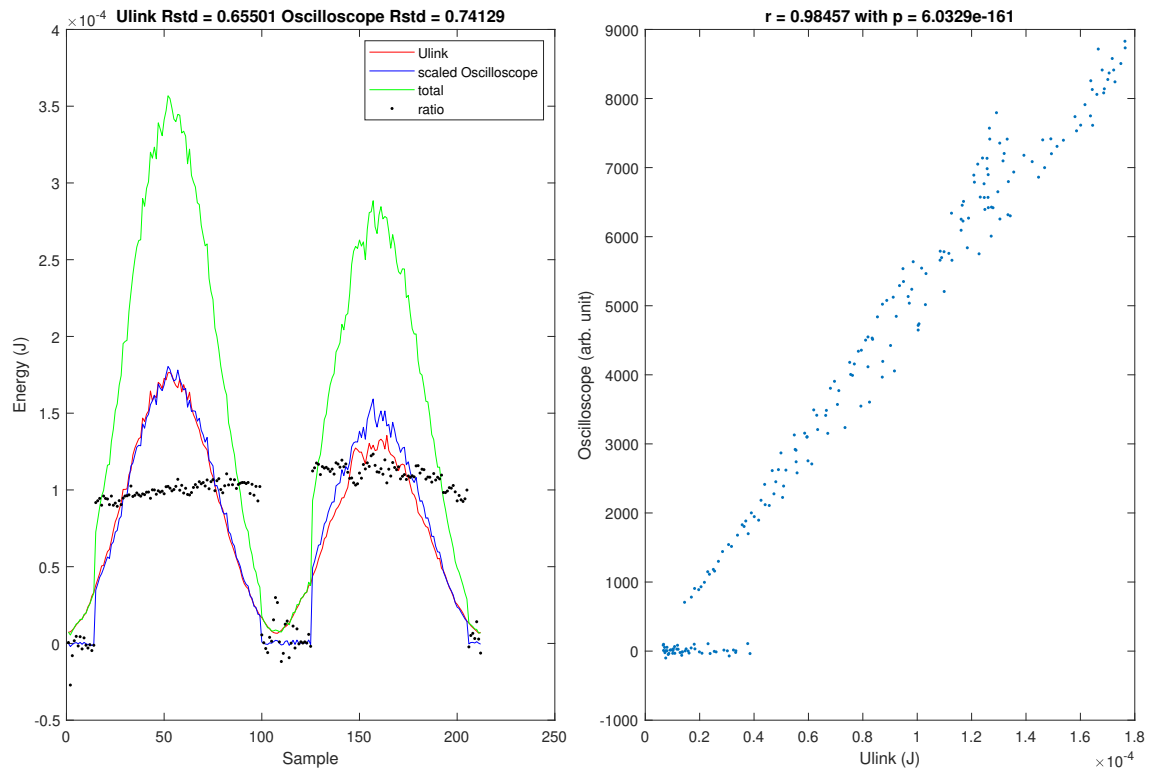


Figure 3.7: The signal from the photodiode integrated using the oscilloscope and MATLAB compared to the signal from the Ulink when the second prism was adjusted so the maximum signal from Ulink was achieved in the first peak. This peak was then used for further testing. The ratio is scaled with a factor of 10^{-4} .

a circuit, was interfaced using a network cable and VISA communication to MATLAB. The analysis of the measurements were done by plotting and calculating the Pearson correlation coefficient (r) and its p-value (p) for qualitative evaluation when comparing the Gentec sensor and our own. The Pearson correlation coefficient measures the strength of the linear relationship between two variables. [13] It takes values from -1 to 1 where larger values indicate a stronger relationship, a strong correlation is generally regarded as greater than ± 0.9 .

Chapter 4

Results

In what follows, the results of measurements taken both with an oscilloscope and Teensy are presented in comparison with the signal from the Ulink interface. In figures where curves are compared, one of them has been scaled using the ratio of the mean values over the entire run. For runs where the prism was rotated the ratio has been scaled so it is at 1 in the appropriate magnitude. When the prism was held still the measurements were made on the top of the first peak in figure 3.7.

4.1 Measurements

The initial measurements carried out are seen in figure 4.1 and 4.2, in which the photodiode was directly connected to the oscilloscope and the pulse integrated in MATLAB. The ratio between the Ulink and photodiode is not constant during the run, but it is very close. The Pearson correlation coefficient is very high, supporting that the energy is low enough for the photodiode to be in its linear regime. When the prism was at a constant angle the two signals show a good correlation as well.

The photodiode was then connected to the circuit, and the signal captured via an oscilloscope, figure 4.3 and 4.4. For the run where the prism was rotated we see a good correlation between the signals, however the relative standard deviations are a bit further apart and the r-value has decreased.

The results from the measurements with Teensy and Ulink are shown in figure 4.5 and 4.6. The runs are similar to those when the photodiode signal was integrated using MATLAB. There is however a large difference in figure 4.6 where it seems as if there is no correlation the values from Teensy and Ulink.

Looking at the left part of figure 4.6 there seems to be peaks that are delayed on the Ulink. Figure 4.7 shows the same data but the signal from Ulink has been

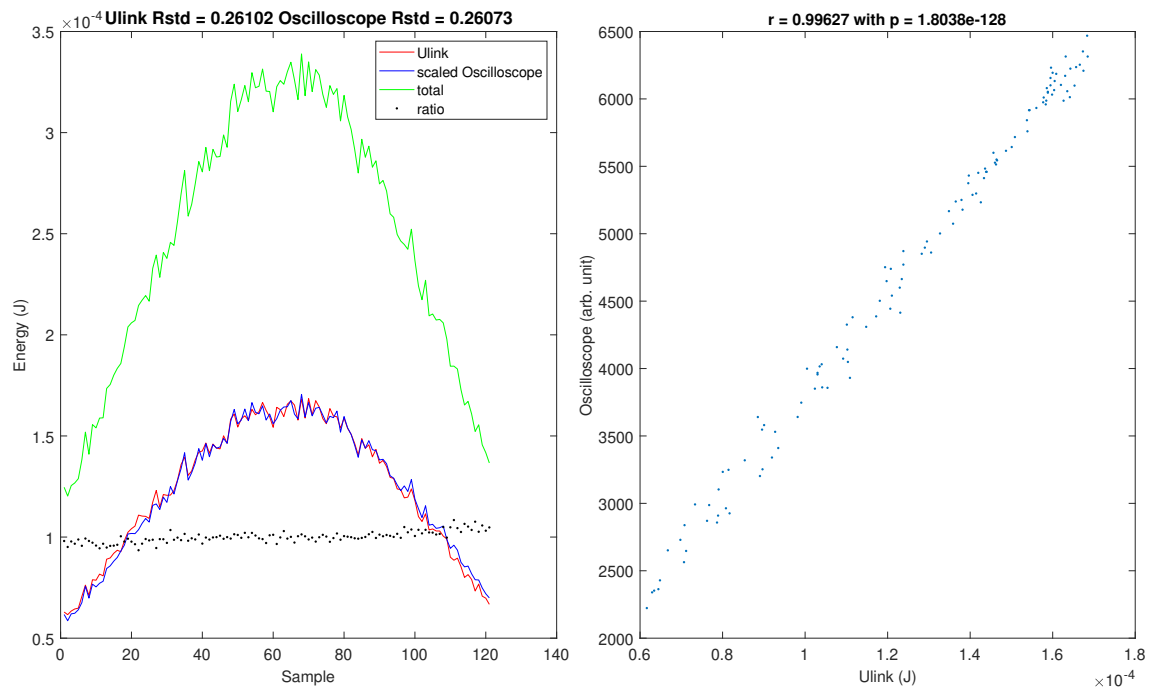


Figure 4.1: Comparison in between the signal from the Ulink and the photodiode, when connected directly to an oscilloscope and the curve integrated in MATLAB. The prism was rotated over the first peak. The ratio is scaled with a factor of 10^{-4} .

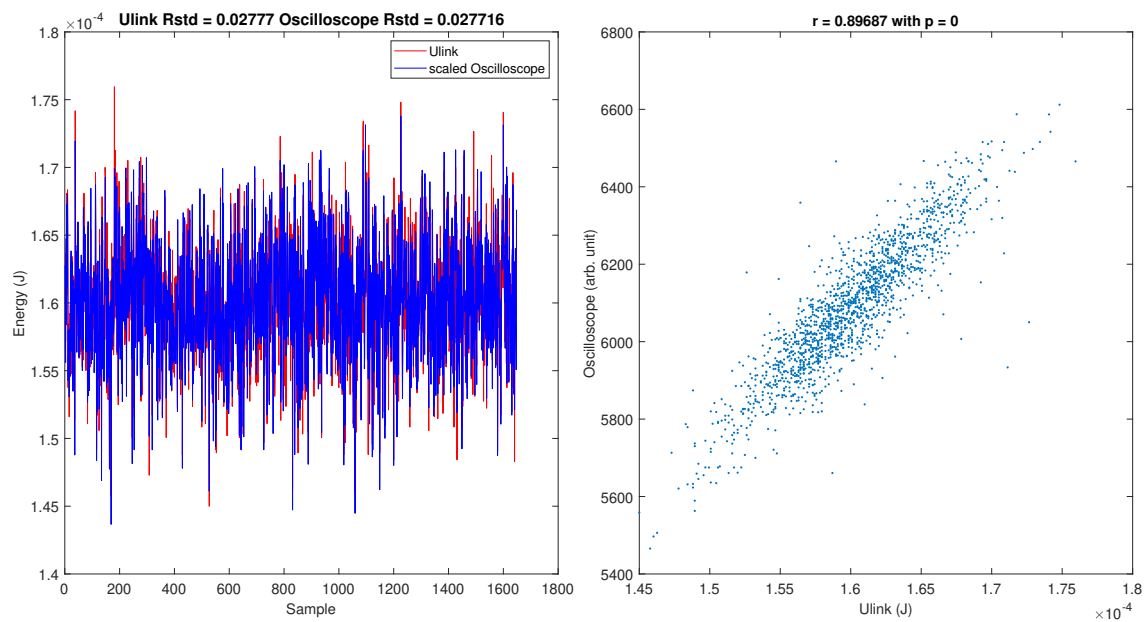


Figure 4.2: Comparison in between the signal from the Ulink and the photodiode, when connected directly to an oscilloscope and the curve integrated in MATLAB. The variations observed are from the laser as the optics where held still.

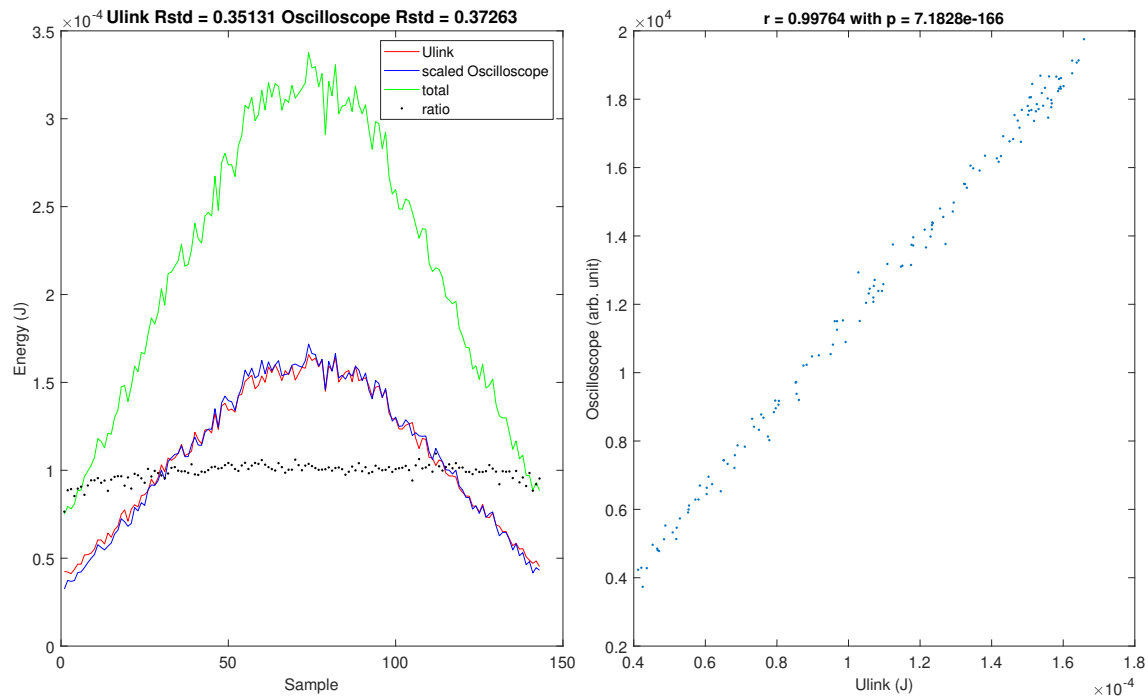


Figure 4.3: Comparison in between the signal from the Ulink and the photodiode, when connected to the circuit in figure 3.2 and then to an oscilloscope and one value of the curve read in MATLAB. The prism was rotated over the first peak. The ratio is scaled with a factor of 10^{-4} .

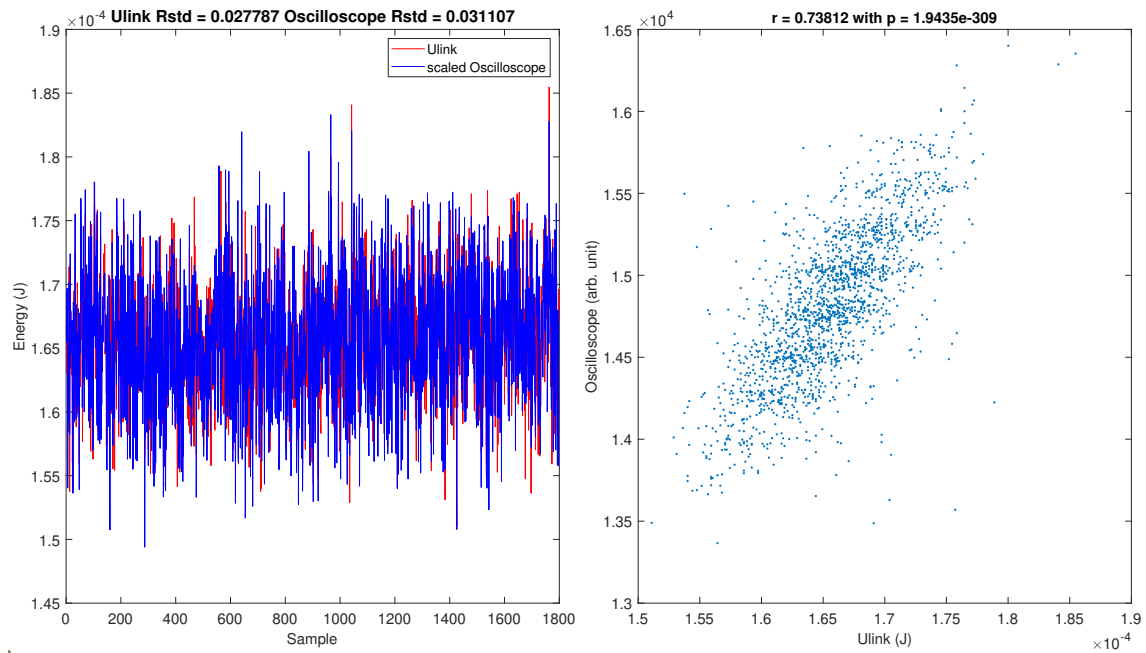


Figure 4.4: Comparison in between the signal from the Ulink and the photodiode, when connected to the circuit in figure 3.2 and then to an oscilloscope and one value of the curve read in MATLAB. The variations observed are from the laser as the optics where held still.

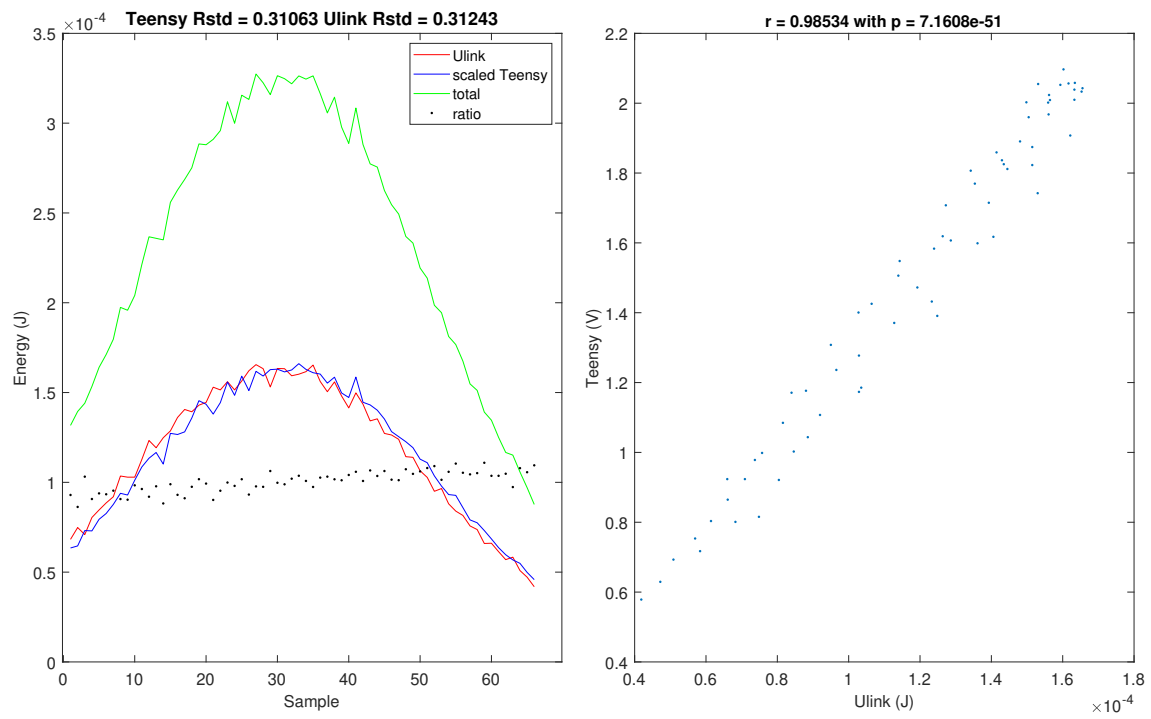


Figure 4.5: Comparison in between the signal from the Ulink and Teensy. The prism was rotated over the first peak. The ratio is scaled with a factor of 10^{-4} .

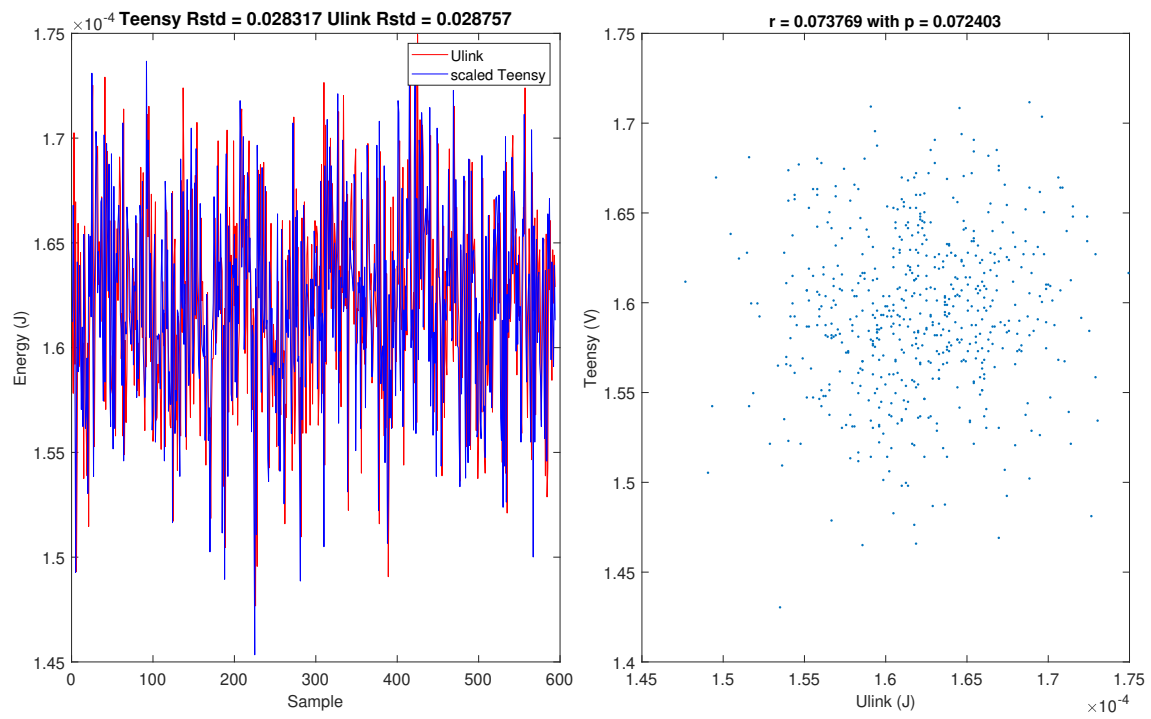


Figure 4.6: Comparison in between the signal from the Ulink and Teensy. The variations observed are from the laser as the optics where held still.

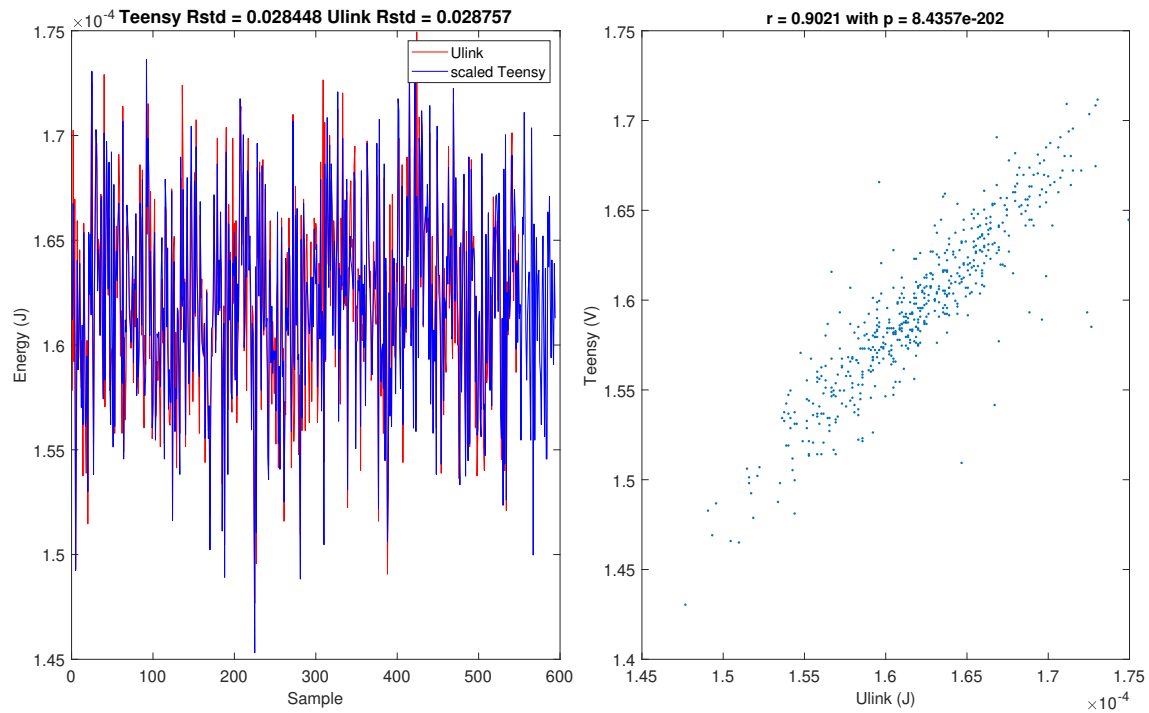


Figure 4.7: The same data as in figure 4.6 but the signal from the Ulink has been moved one step.

moved one step back. The r-value is now almost the same as when the photodiode signal was integrated using MATLAB.

The figures 4.8 and 4.9 show where the signals have been moved one step back, Ulink in 4.8 and Teensy in 4.9. Contrary to how in figure 4.7 Ulink was delayed by one step in order to match the signals, here the largest r-value, and visually correlated curves, appear when the Teensy is delayed one step.

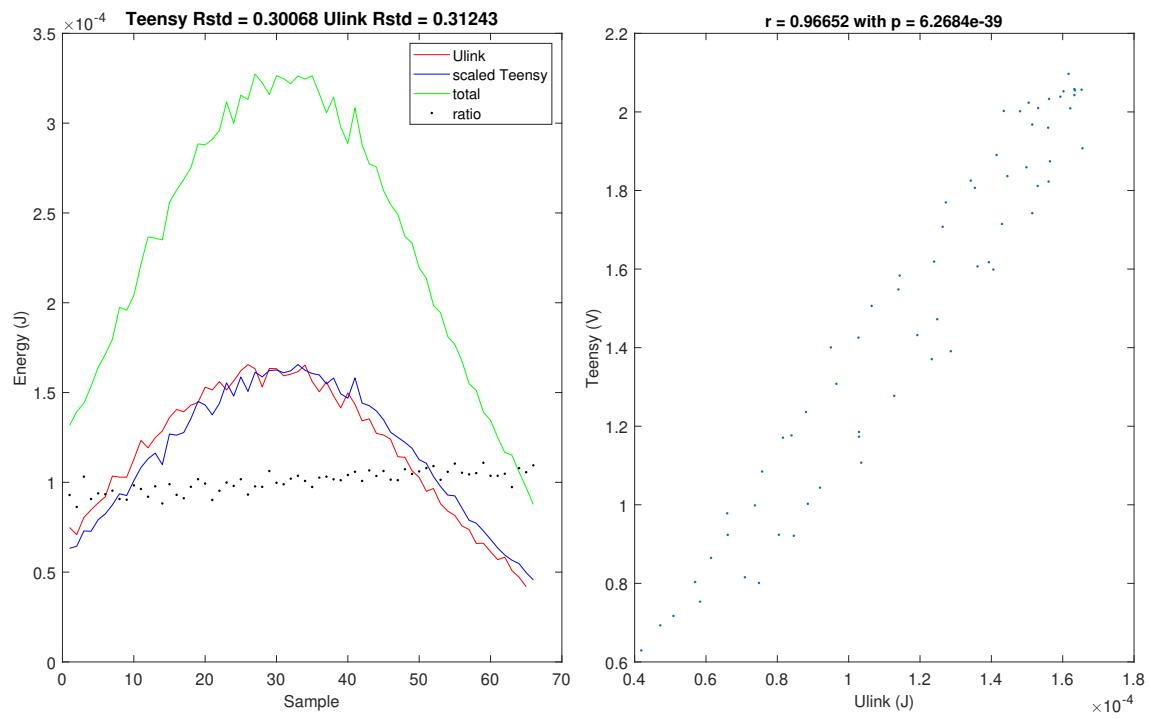


Figure 4.8: The same data as in figure 4.5 but the signal from the Ulink has been moved one step. The ratio is scaled with a factor of 10^{-4} .

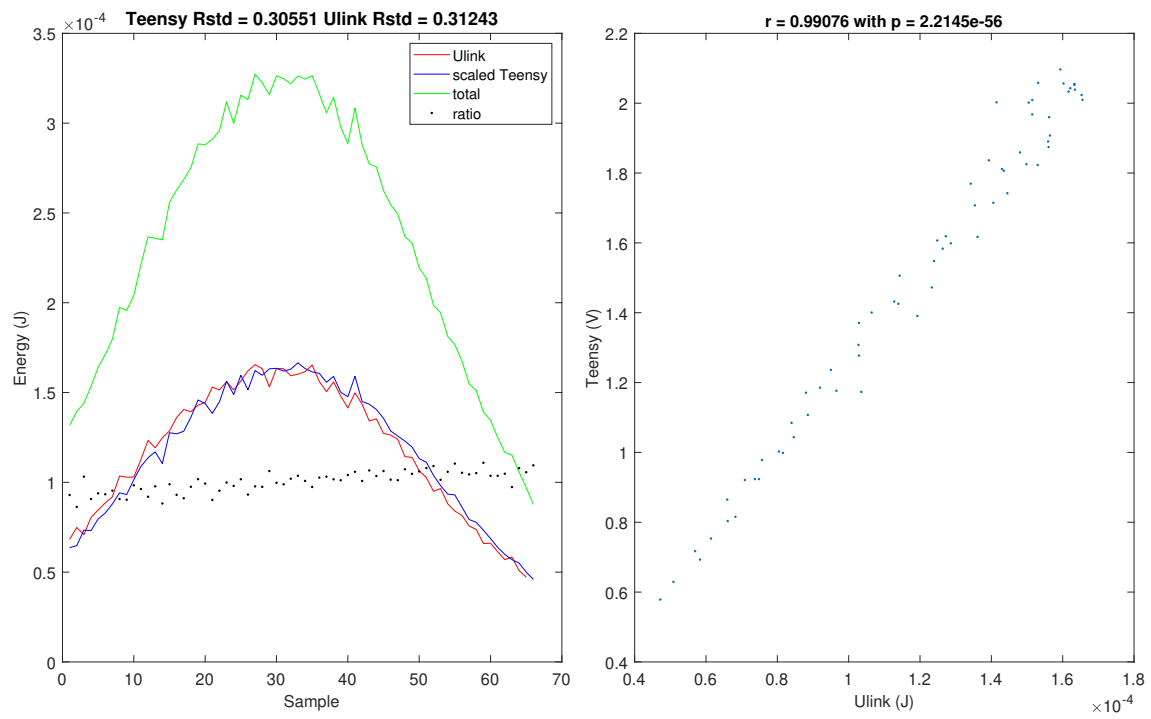


Figure 4.9: The same data as in figure 4.5 but the signal from the Teensy has been moved one step. The ratio is scaled with a factor of 10^{-4} .

Chapter 5

Analysis and Discussion

In this chapter the analysis of the results is presented together with a discussion about the work.

5.1 Analysis

The results from the measurements, where the photodiode signal was integrated using Matlab, indicate that the employed optical setup has sufficiently attenuated the beam intensity for sustaining the diode's linear regime. Figure 4.3 indicates that the added resistance, when using the RC-circuit, did not change this behavior. The deviations in the figures where the prisms were held still are likely due to the MATLAB script only reading a single value from the curve. This could result in a broadening due to the noise in the circuit and it would have been better to sample a number of points and take their mean value. When Teensy was used to measure the output signal from the circuit figure 4.6 indicates that there was only a very weak correlation between the two meters. This fault was likely due to delays when reading the signals with MATLAB, resulting in the values being delayed in the data. Otherwise, the conclusion would be that the circuit causes the photodiode to become unresponsive to small changes, and in accordance with figure 4.5 it would only be suitable to use this type of energy meter when the energy changed over a larger interval than the variations from the laser. Since the relative standard deviations are close we can assume that the accuracy of the meter is smaller than these variations. If the responsiveness of the diode had changed, or the circuit making the output too noisy, the relative standard deviations would not be so close, and therefore this further supports the assertion that the signal had been delayed in the data. The fact that the displacement of data for the figures where the Glan-Taylor prism was

rotated is the opposite to when it was not also indicates that the problem was in the reading of the signal.

5.2 Discussion

This work has been focused on the electrical circuit and the microcontroller. It shows that these two components are a viable alternative to more expensive systems. To summarize the accuracy of the meter, the results indicate that the circuit has a small effect on the signals correlation with energy and that the energy meter we designed is a viable alternative when the energy is in an appropriate range. The output of the meter is however in voltage, to translate it into energy there needs to be calibrated against a known reference. This was however not analyzed for the setup used as the main interest was observing variations in the pulses energy content. It also shows that the Teensy 3.2 microcontroller could be used for these applications.

Bibliography

- [1] W. Koechner, *Solid-state laser engineering* (Springer series in optical sciences 1), 6th rev. and updated ed. New York, NY: Springer, 2006, 747 pp., ISBN: 978-0-387-29094-2.
- [2] Thorlabs. "Photodiode rise time increases with wavelength of incident light". (), [Online]. Available: <https://www.thorlabs.com> (visited on 22/04/2024).
- [3] Thorlabs. "Free-space biased detectors". (), [Online]. Available: <https://www.thorlabs.com> (visited on 22/04/2024).
- [4] Thorlabs. "Unmounted photodiodes lab facts". (), [Online]. Available: <https://www.thorlabs.com> (visited on 07/05/2024).
- [5] *Microcontroller*, in *Wikipedia*, Page Version ID: 1218118301, 9th Apr. 2024. [Online]. Available: <https://en.wikipedia.org/w/index.php?title=Microcontroller&oldid=1218118301> (visited on 07/05/2024).
- [6] B. Lars. "Embedded measurement systems - göteborgs universitets publikationer". (), [Online]. Available: <https://gup.ub.gu.se/publication/183359> (visited on 23/04/2024).
- [7] *Successive-approximation ADC*, in *Wikipedia*, Page Version ID: 1219980122, 21st Apr. 2024. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Successive-approximation_ADC&oldid=1219980122 (visited on 07/05/2024).
- [8] *Thévenin's theorem*, in *Wikipedia*, Page Version ID: 1220669375, 25th Apr. 2024. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Th%C3%A9venin%27s_theorem&oldid=1220669375 (visited on 26/05/2024).
- [9] *Operational amplifier*, in *Wikipedia*, Page Version ID: 1221599672, 30th Apr. 2024. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Operational_amplifier&oldid=1221599672 (visited on 07/05/2024).

- [10] PJRC. "Teensy 3.2". (), [Online]. Available: <https://www.pjrc.com/store/teensy32.html> (visited on 07/05/2024).
- [11] V. Pedro. "ADC: ADC". (), [Online]. Available: http://pedvide.github.io/ADC/docs/Teensy_3_2_html/index.html (visited on 07/05/2024).
- [12] PJRC. "Https://www.pjrc.com/teensy/k20p64m50sf0.pdf". (), [Online]. Available: <https://www.pjrc.com/teensy/K20P64M50SF0.pdf> (visited on 29/04/2024).
- [13] J. S. Milton and J. C. Arnold, *Introduction to Probability and Statistics: Principles and Applications for Engineering and the Computing Sciences*. McGraw-Hill, 2003, 824 pp., page 418-425., ISBN: 978-0-07-119859-2.

Appendix

Here follows the program used on the Teensy 3.2 in order to read the voltage, and then transmit it via serial communication to the receiver when the receiver sends a "1" character, change the threshold with "2" and change the readDelay with "3".

```
#include <ADC.h>
#include <IntervalTimer.h>
#include "util/atomic.h"

const int readPin = A2;
volatile long pdVoltage = 0;
volatile bool pdHasBeenRead = false;
int userCommand;
int newThreshold;
uint32_t readDelay = 30; // microseconds

elapsedMicros sinceTest;

ADC *adc = new ADC();
IntervalTimer myTimer;

void setup() {
    pinMode(readPin, INPUT_DISABLE);

    adc->adc0->setAveraging(0); // set number of averages
    adc->adc0->setResolution(12); // set bits of resolution
    adc->adc0->setConversionSpeed(ADC_CONVERSION_SPEED::MED_SPEED); // change the conversion speed
    adc->adc0->setSamplingSpeed(ADC_SAMPLING_SPEED::HIGH_SPEED); // change the sampling speed
    adc->adc0->setReference(ADC_settings::ADC_REFERENCE::REF_3V3); // REMEMBER TO UPDATE IN PRINT
    adc->adc0->calibrate();
    adc->adc0->wait_for_cal();

    ComparatorSetup(12); // Set comparator to on

    NVIC_ENABLE_IRQ(IRQ_CMP1);
    NVIC_SET_PRIORITY(IRQ_CMP1,0);
    attachInterruptVector(IRQ_CMP1, cmp1_isr);

    myTimer.priority(1);

    sei()
}

void loop() {

    if (Serial.available()){
        cli();
        userCommand = Serial.read(); //fetch command
        if (userCommand == 49){
            ATOMIC_BLOCK(ATOMIC_RESTORESTATE)
                Serial.println(pdVoltage*3.3/adc->adc0->getMaxValue(),DEC);
            pdVoltage = 0;
        }
    }
}
```

```

    if (userCommand == 50) {
        Serial.flush();
        Serial.println("Threshold update");
        while (!(Serial.available() > 0)) {
            delay(1000);
        }
        ATOMIC_BLOCK(ATOMIC_RESTORESTATE)
        newThreshold = Serial.parseInt();
        Serial.println(newThreshold);
        ComparatorSetup(newThreshold);
        delay(50);
    }
    if (userCommand == 51) {
        Serial.flush();
        Serial.println("readDelay update");
        while (!(Serial.available() > 0)) {
            delay(1000);
        }
        readDelay = Serial.parseInt();
        Serial.println(readDelay);
        delay(50);
    }
}

sei();
}

}

void ComparatorSetup(int threshold) {
    SIM_SCGC4 |= SIM_SCGC4_CMP; //Clock to Comparator

    // Input pins select;
    CMP1_MUXCR = CMP_MUXCR_PSTM | CMP_MUXCR_PSEL(0) | CMP_MUXCR_MSEL(7);

    // Set filter and hysteresis
    CMP1_CRO = CMP_CRO_FILTER_CNT(7) | CMP_CRO_HYSTCTR(3);

    //Enable interrupt rising
    CMP1_SCR = CMP_SCR_IER | CMP_SCR_CFR;

    // Set control register
    CMP1_CR1 = CMP_CR1_PMODE | CMP_CR1_COS | CMP_CR1_OPE | CMP_CR1_EN;

    // enable DAC and set threshold
    CMP1_DACCR = CMP_DACCR_DACEN | CMP_DACCR_VRSEL | CMP_DACCR_VOSEL(threshold) ;
    if (threshold == 0) CMP1_DACCR = 0; //Disable DAC ==> reference = 0 V
}

FASTRUN void cmp1_isr() {
    cli()
    CMP1_SCR = B00010100; //Clear flag in CMP1
    myTimer.begin(readPulse,readDelay);
    sei()
}

FASTRUN void readPulse() {
    pdVoltage = adc->adc0->analogRead(readPin);
    myTimer.end();
}
}

```

Here follows the MATLAB class for communicating with the Teensy 3.2

```
classdef teensyEmeterV2
    %teensyEmeterV2 interface for teensy measurement
    % V2 uses writeread to get energy, inspired by Ulink class.

    properties
        teensySerial
    end

    methods (Access = private)
        function E = getEnergy(obj) % scalar
            str = writeread(obj.teensySerial,"1");
            str = string(str);
            E = str2double(str);
        end
    end

    methods (Access = public)
        function obj = teensyEmeterV2(useSerial)
            obj.teensySerial = serialport(useSerial,9600);
            configureTerminator(obj.teensySerial,"CR/LF");
            flush(obj.teensySerial);
        end

        function E = Measure(obj,n)
            E = zeros(1,n);
            for i = 1:n
                java.lang.Thread.sleep(100);
                e = getEnergy(obj);
                E(1,i) = e;
            end
        end

        function changeThreshold(obj, threshold)
            java.lang.Thread.sleep(1000);
            string1 = writeread(obj.teensySerial,"2"); %Set teensy to
                wait for the new threshold
            java.lang.Thread.sleep(500);
            string2 = writeread(obj.teensySerial,int2str(threshold));
                %Send new threshold
            fprintf(string1 + " " + string2 + "\n");
            java.lang.Thread.sleep(2000); %Wait 2 seconds to prevent
                errors
        end

        function changeDelay(obj, delay)
            java.lang.Thread.sleep(1000);
        end
    end
end
```

```
string1 = writeread(obj.teensySerial,"3"); %Set teensy to  
        wait for new delay  
java.lang.Thread.sleep(500);  
string2 = writeread(obj.teensySerial,int2str(delay)); %  
        Send new delay  
fprintf(string1 + " " + string2 + "\n");  
java.lang.Thread.sleep(2000); %Wait 2 seconds to prevent  
        errors  
end  
end  
end
```