Preparation of experiment for controlled laser damage of single-photon avalanche photodiode

Project report by
Audun Nystad Bugge

June 15, 2011

Supervisors
Postdoc Vadim Makarov
Professor Johannes Skaar

Norwegian University of Science and Technology
Department of Electronics and Telecommunications

Abstract

Quantum cryptography has been developed from being a theoretical proposal to having real world applications with companies developing and selling quantum key distribution (QKD) instruments commercially. Although laws of quantum physics guarantee perfect security in the key distribution, several vulnerabilities in the practical implementations have been described and demonstrated. In some of these attacks the eavesdropper has been able to acquire the entire secret key without being detected by the two parties sharing the key.

This study describes the preparations of an experiment aiming to use a high powered laser to damage a single photon avalanche diode to explore new weaknesses in these systems. Hopefully, and according to previous experience, the discovery of loopholes will ultimately lead to a cryptographic system that can be proven to be secure even when taking into account all possible ways to attack the implementation.

A basic QKD protocol is described and briefly explained, along with some examples of attacks that have been proposed and performed against QKD systems. The operation of avalanche photodiodes (APDs) is explained and it is shown how an APD can be incorporated into a circuit in order to work as a photon counting device.

An experimental setup was developed using a passive quenching circuit with a silicone APD, a weak signal laser and a powerful damaging laser at 807 nm together with multiple instruments intended for experimental control and characterization of the APD. All these instruments were connected to a computer and made programmable in order to fully automate the final experiment.

Finally, the breakdown voltage of the diode at $-25\,^{\circ}\mathrm{C}$ was estimated by two different methods. Gradually reducing the bias voltage while looking at the avalanche pulse to see where it disappears was tried first, then measuring the size of the avalanche pulse at several different bias voltages and extrapolating to zero. These two methods yielded similar, satisfactory results.

Contents

List of Figures

Chapter 1

Preface

Using a high power laser to damage optical components in a quantum key distribution (QKD) system may be the ultimate tool for an eavesdropper. Laser damage may make controllable, intentional changes to the components, changing their characteristics beyond the model used in the theoretical security proofs used to describe them. The effects of such damage has not been studied yet, but may be expected to increase the systems' vulnerability to different attacks previously described for healthy detectors.

This project is a part of a larger experiment at the quantum hacking group at the Department of Electronics and Telecommunications, NTNU. The goal of the project was to create an experimental setup which would provide a good platform for continued work on studying laser damage as a tool for eavesdropping communications in a QKD system.

A major part of the work was to set up a computer running Ubuntu Linux, connect all required instruments, make it all work with the Linux machine and develop libraries for programming the devices in C++. The project also had a significant electronics part. A home made single-photon avalanche diode (SPAD) circuit was modified, partially rebuilt and tested.

The ultimate goal of the work started by this project is to have a working setup which automatically applies gradually higher intensity laser light to the detector, runs a series of tests to characterize it, and at some predefined discrepancy from normal behavior stop to let the experimenter take manual control over further damage to the detector. This goal, however, is out of scope for student project such as this.

1.1 Acknowledgements

I would like to thank my supervisors, professor Johannes Skaar and postdoc Vadim Makarov, who have been very helpful and available to questions throughout the entire project. Especially during the finalization of this text they have been available day and night and provided very useful help and comments.

I would also like to thank PhD student Lars Lydersen for giving me some valuable hints and example code to get me started on serial port programming in C++.

Chapter 2

Introduction to quantum cryptography

This chapter gives a brief introduction to classical cryptography, a description of a basic protocol used for quantum cryptography and some considerations about weaknesses of real implementations of quantum cryptography systems.

The sections before section 2.2.2 is mostly based on [7].

2.1 Classical cryptography

The goal of cryptography is to make the contents of a message indecipherable for anyone except the intended receiver. Or, in other words, let Alice[1] and Bob share a message while ensuring that Eve is unable to understand it. Cryptography can be broken up into *encryption* and *decryption*, where encryption is the process of combining the message with a key to form a cryptogram. In order to read the cryptogram, one must decrypt it, which is done either by using the same key as was used for encryption or another, special key - this depends on the class of cryptosystem used. In order for the cryptosystem to be secure it should be impossible to decrypt the message without the correct key. In practice, however, this process is normally just extremely difficult - so difficult that it is impossible to crack before the secrecy of the message is no longer important.

Two main classes of cryptosystems exist - asymmetrical and symmetrical. They are classified in regards to whether the key used by the two parties is the different or equal, respectively.

2.1.1 Asymmetrical cryptosystems

Asymmetrical cryptosystems - commonly known as public-key cryptosystems, are based on computational difficulty. Two keys are generated by Bob - one private key which he keeps secret and one public key generated from the private key which he shares with anyone. The public key is used to encrypt messages that can be decrypted only by the private key. This system is based on one way functions - functions that are easy to do one way, while keeping it very hard to find the starting point from the result without

[1] Alice, Bob and Eve are standard names for the sender, receiver and eavesdropper, respectively.

some extra information. A standard example of this is the much-used RSA[2] algorithm based on multiplication of large primes and factorization of the result. In short, it is easy to find the product of two large prime numbers and also easy to find the remaining prime number if one is given the product and already has one of the factors. On the other hand, if just presented with the product of the two primes, it is difficult to find the original primes. "Difficult" in this context means that the computational time grows exponentially with the number of bits in the input, compared to polynomially for "easy".

The problem with this very convenient and much used system is that it has been impossible to prove that it actually is difficult to factorize the product. If someone invents an algorithm which makes it easy, public key cryptography based on factorization becomes useless. An algorithm developed by Peter Shor does exactly this, but would require a quantum computer to work [13].

2.1.2 Symmetrical cryptosystems

In a symmetrical cryptosystem, Alice and Bob is required to share a common, secret key beforehand. This key is used both for encryption and decryption, and is not based on one way functions like the asymmetrical cryptosystems.

The most relevant symmetrical cryptosystem in the context of quantum cryptography is the one-time pad, dating back to 1926 [17]. Assuming that the key is perfectly random, equally long to the message and is only used once, one-time pad is proven to be perfectly secure. If we consider a binary message (composed of 0s and 1s), the encryption is done by bitwise addition of the key to the message. For example, if the message is 10100110 and the key is 00101101, the encrypted message will be 10001011. Bob simply subtracts the key from the encrypted message and finds the original content.

One-time pad has the obvious disadvantage of requiring a secret key longer than or of equal length to the message to be encrypted. For this reason, it is only used for very specialized purposes today. Other symmetrical cryptosystems use much shorter keys and complicated algorithms which make them convenient for modern cryptography as we know it from all kinds of secure communications today, but like public-key cryptosystems they may be vulnerable to advances in cryptoanalysis.

2.2 Quantum cryptography

Quantum cryptography (QC) is a possible solution to the challenges facing the most common cryptosystems in use today. The basic idea of QC is to use a quantum channel to let Alice and Bob agree on a secret, random key which can be proven to be unavailable to Eve. See for example [10] for a full proof. This key may then be used in a one-time pad scheme to send the data over a classical insecure channel. The process of using a quantum channel to share a secret key is named quantum key distribution (QKD).

[2]RSA from Rivest, Shamir and Adleman, the surnames of the algorithm's creators

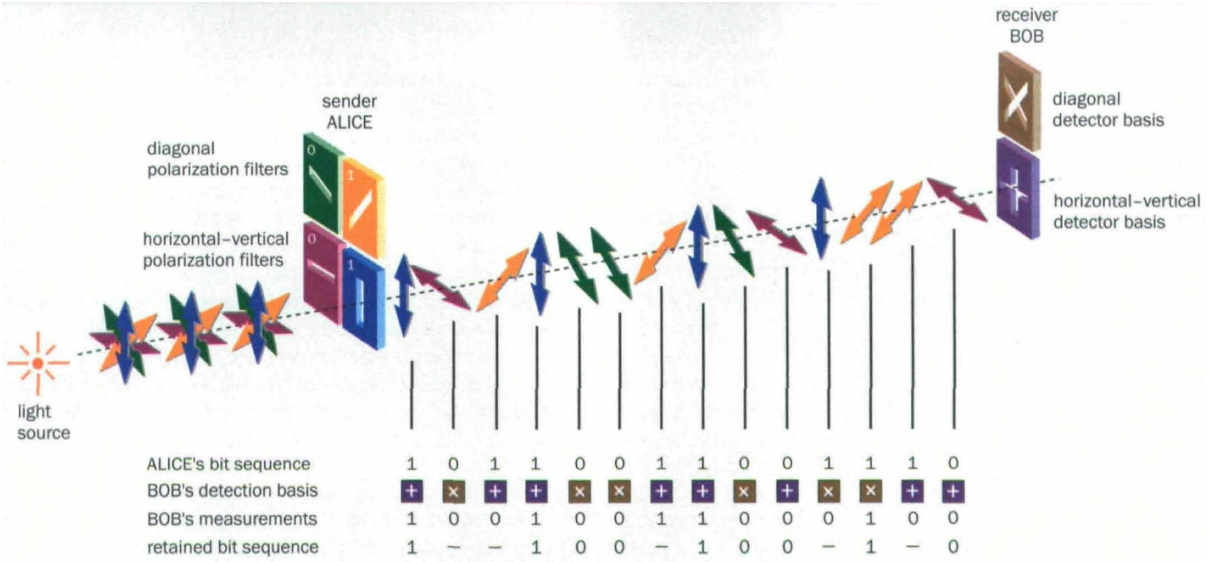


Figure 2.1: Explanation of QKD using the BB84 protocol. Figure from [15].

2.2.1 Quantum key distribution

We shall now investigate further the first protocol for QKD, described in 1984 by Bennett and Brassard[2] named BB84. The basic idea is to send each bit as a single photon quantum state where either the polarization or the phase of the photon describes its bit value. For example, if Alice transmits photons that are polarized either vertically, horizontally or $\pm 45°$, vertical and $45°$ polarizations would represent a 1 while horizontal and $-45°$ polarizations would represent a 0 (Fig. 2.1). Alice randomly chooses the rectilinear or the diagonal basis for each bit she sends and Bob, likewise, randomly chooses a basis when he measures the photon. This way, 50% of the photons will be measured in the wrong basis, which gives a random result for these bits.

After receiving the required amount of raw key data from Alice, communications will proceed on the public channel. We assume that Eve can read all data on the public channel, but not modify it. Such active eavesdropping would require extra countermeasures which are left out from this discussion. Now, Bob tells Alice which basis he used for measuring each received photon. Alice then responds which photons were measured in the correct basis (without disclosing the actual bit value, of course). The bits corresponding to photons which were measured in the wrong bias or not received at all are discarded by both Alice and Bob. The remaining key sequence is named the *sifted key*.

In the idealized case with true single photon states and without any loss, noise or eavesdropping, Alice's and Bob's sifted key is equal. Different eavesdropping schemes exist, but any eavesdropping will perturb the quantum states and thus be detectable by Alice and Bob. In the ideal case, Alice and Bob could compare some random parts of their keys, any discrepancies found indicate that eavesdropping has taken place. In a real system there will always be discrepancies, and more advanced algorithms must be used in order to determine if there has been any eavesdropping. A combination of error correction and privacy amplification will ensure that Alice and Bob have the same key

while minimizing Eve's information. These methods are thoroughly explained in [1], for example.

2.2.2 Real systems and weaknesses

As we have seen, QKD is in theory completely secure. The implementation in real systems, however, is not necessarily secure. Over the last decade several attacks on QKD systems have been proposed, and in recent years several proof of principle demonstrations have been conducted on commercial systems. A few examples will be presented here.

Faked state generation via detector blinding The general idea of a faked state attack is to intercept Alice's signal and prepare a new signal that is sent to Bob, tricking Bob into making the same measurement as Eve did.

It has been demonstrated in [5] that a QKD system can be eavesdropped by applying bright illumination to Bob's detector. In this regime, the avalanche photodiode (APD, see chapter 3) operates as a classical photodiode producing a photocurrent proportional to the applied optical power. An optical power higher than a threshold P_{th} will generate a photocurrent strong enough to register a count in the detector.

The setup in this example using polarization coding was such that continuous circularly polarized illumination was used for blinding all of Bob's 4 detectors. By applying a pulse of linearly polarized light in the polarization Eve used for detecting the signal with power $2P_{th}$, Eve forces Bob to make the same measurement as she did, thereby perfectly eavesdropping the QKD session.

An attack using the same principles was shown to work against a phase-encoded commercial QKD system in [9].

Reflectometry Reflectometry can be used by Eve to see which quantum states are sent by Alice, as described in [16]. This attack been named a "Trojan-horse attack"[6].

By applying short pulses of light in the quantum channel directed towards Alice at the same moment Alice is preparing her quantum states and measuring the reflections, Eve could be able to read out the entire key sent from Alice if she is not careful enough. To protect herself, Alice should employ filters that do not allow light other than the appropriate wavelengths, only activate her encoding optical components when the qubit is there and set an known upper bound to the mount of reflected light that could be exploited by Eve. A common way to protect against this kind of attack is to use a detector monitoring the energy of incoming light pulses.

2.2.3 Laser damage as a tool for eavesdropping

Preliminary testing of laser damage of APDs in our lab has given some interesting results. Two detectors were manually damaged by a high power laser, each showing different properties after the damage.

One of the damaged detectors developed a strong dark current effectively blinding it during normal operation. In this regime it was still controllable by a bright pulse as described in the faked state generation paragraph above. A detector damaged this way

could significantly enhance the possibilities of succeeding with such an attack, since no continuous, detectable blinding light would be required.

The other damaged detector lost all photosensitivity. One potential use of this could be to enable the use of the reflectometry based attack mentioned above even in the presence of a detector monitoring incoming light pulses. If Eve is able to controllably damage the monitoring detector in such a way, she could circumvent the security proof and apply the Trojan-horse attack while Alice thinks she is safely protected by the monitoring detector.

Chapter 3

Avalanche photodiodes

An avalanche photodiode (APD) is a photodiode which is strongly reverse-biased. At such high internal electric field in the diode, an exited carrier is accelerated very quickly, and may excite new carriers by impact ionization [11]. These new carriers may again trigger new carriers, creating a cascade effect. APDs used for quantum cryptography are operated at a bias higher than the breakdown voltage. At this high bias, an absorbed incoming photon carrying energy higher than the band gap of the material is able to trigger this effect, making it useful as a single photon detector. APDs operating is this range are also know as single-photon avalanche diodes (SPAD) and Geiger-mode avalanche diodes [3].

An APD operated in Geiger mode, or photon counting mode, is set up such that an avalanche is registered by some sensing circuitry while the avalanche is being quenched by either lowering the voltage below the breakdown voltage or by reducing the current to a level where it becomes probable that no carriers are crossing the barrier, this is named the *latching current* or *quenching current* of the diode.

Currently, silicon APDs by far provide the best performance, if one has the freedom to choose the wavelength independent of other factors [11]. That is, for light in the visible to near-infrared range ($\lambda_0 = 400\,\text{nm}$ to $1000\,\text{nm}$). For light in the range typically used for optical fiber communications ($1.3\,\mu\text{m}$, $1.55\,\mu\text{m}$), the photons have lower energy than the bandgap of Si, and is not absorbed. Thus, since the photons are not absorbed they will not trigger an avalanche in a Si APD. For this case, InGaAs/InP heterostructures are the best choice, but performance is significantly reduced compared to Si devices at shorter wavelengths [11].

3.1 APD circuits

This section is mainly based on [3].

3.1.1 General working principles

There are many different ways to set up a circuit utilizing a photodiode to detect single photons, but some principles are common for all setups. In order to make the diode sensitive to single photons, it has to be reverse biased above the breakdown voltage V_{br}. The diode then has an overvoltage $V_{over} = V_{bias} - V_{br}$ where V_{bias} is the applied bias

voltage. At this high reverse bias, a single charge carrier injected in the depletion zone can start a self-sustaining avalanche due to the high electric field. The avalanche pulse rises in a few nanoseconds, and the current will continue to flow until it is quenched by lowering the bias below the breakdown voltage. This is done either actively or passively, discussed in Secs. 3.1.3 and 3.1.4.

3.1.2 Dark counts

Since any free charge carrier in the depletion zone will start an avalanche, one needs to consider dark counts caused by thermal generation, trapped carriers and tunneling.

Thermal generation Dark counts by thermally generated carriers are dependent on the temperature of the diode, and the dark count rate due to this effect is relatively easily reduced by lowering the temperature. Increasing bias voltage also increases thermally generated dark counts due to two effects, namely field-assisted enhancement of the emission rate from generation centers and increased avalanche triggering possibility.

Trapped carriers During the avalanche pulse, some carriers get trapped in deep levels in the depletion layer. These are later released after a fluctuating delay and may then trigger a new avalanche. The number of trapped carriers depend upon the number of carriers crossing the junction, and is thus increased both by pulse length and intensity. The current intensity is proportional to the bias voltage, which is normally adjusted with respect to other factors. A way to reduce afterpulses is then to minimize the time before the pulse is quenched, and also to introduce a hold-off time after quenching where the bias voltage is kept low for some time after the pulse to allow trapped carriers to release without starting a new avalanche.

Tunneling At very high electric field intensity, tunnel-assisted direct band-to-band transitions may happen [4]. Even an extremely small tunneling current may cause a significant increase in the dark count rate. The bias voltage and thus the field intensity should be kept as low as possible in order to reduce tunnel-assisted generation.

3.1.3 Passive-quenching circuits

Schematics for a passive-quenching circuit (PQC) is shown in Fig. 3.1. C_s is the stray capacitance, the capacitance to ground from the diode terminal connected to R_L. C_S is typically a few pF. C_D is the junction capacitance in the diode, typically ~1 pF. R_L is a high value ballast resistor to quench the avalanche current, typical values are on the order of hundreds of kiloohms.

Before detecting a photon, the diode in Fig. 3.1 a) is reverse biased at V_{bias} through a large ballast resistor R_L. When an avalanche is triggered the current through the diode quickly rises as seen in Fig. 3.1 c). The current through the diode is then given as

$$I_d(t) = \frac{V_d(t) - V_{br}}{R_d} = \frac{V_{ex}(t)}{R_d} \tag{3.1}$$

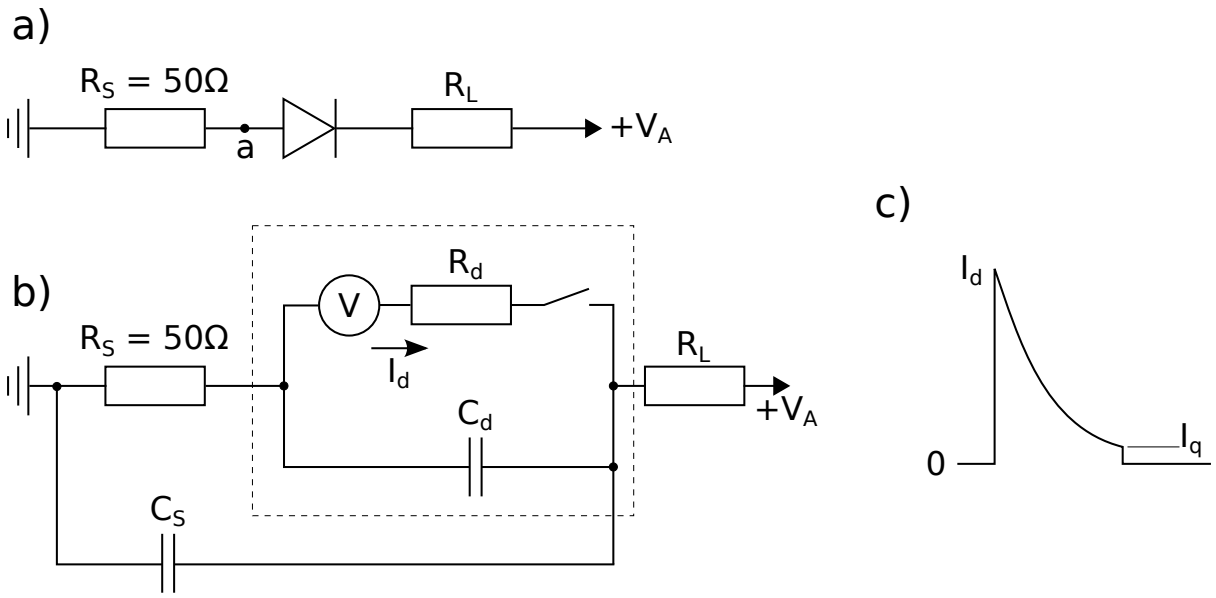


Figure 3.1: Passive-quenching circuit. a) Schematics of a simple passively quenched circuit in current-mode output. b) The equivalent circuit of subfigure a); the circuitry inside the dashed rectangle corresponds to the diode. c) The signal sensed at point A in subfigure a), which corresponds to the current through the diode. Figures redrawn from [3].

where $V_{ex}(t) = V_d(t) - V_{br}$ is the transient excess voltage over the diode. The diode current discharges the capacitances which results in the exponential decay of I_d and V_d towards their asymptotic values

$$I_a = \frac{V_{bias} - V_{br}}{R_L + R_d} \approx \frac{V_{over}}{R_L} \quad (R_L \gg R_d) \tag{3.2}$$

$$V_a = V_{br} + R_d I_a \tag{3.3}$$

When the current through the diode becomes very low the transport of carriers through the depletion zone becomes a statistical process. When I_d becomes lower than ~100 µA the probability for a carrier to cross the diode becomes small and drops quickly as the current is decreased further [8]. To ensure approximately equally long avalanche pulses, one must choose a value for R_L such that I_a is well below this quenching current $I_q < 100\,\mu A$. This results in a good slope in the avalanche current when it crosses I_q, leading to a well defined quenching time with fairly small jitter. If the value of R_L is too low such that $I_a \sim I_q$, the quenching time will have large jitter or even not be quenched at all for $I_a > I_q$.

After the avalanche is quenched the capacitances (C_s and C_d) are slowly charged through the ballast resistor R_L. This corresponds to a voltage recovery time constant

$$T_r = R_L(C_S + C_d). \tag{3.4}$$

This leads to a recovery time in the microsecond range for common values of the capacitances and ballast resistor. When the diode voltage rises above breakdown voltage, the

diode once again becomes susceptible for avalanches. The triggering probability is very low during the first part of recovery, increasing as the bias increases. An avalanche initiated before the voltage has fully recovered will be smaller than an avalanche occurring at V_{bias}, as seen from Eq. 3.1. This results in a smaller output signal - if this is smaller than the threshold of the comparator sensing the avalanche pulse, no signal is generated and the photon will not be counted.

3.1.4 Active-quenching circuits

The basic difference between an active-quenching circuit (AQC) and a PQC is that the AQC reacts back on the leading edge of the avalanche pulse and actively lowers the bias voltage in order to quench the avalanche, compared to the PQC where the avalanche is quenched by discharging the stray and diode capacitances. This requires a quenching pulse higher than the overvoltage and with opposite polarity to be created by a device in the circuit and applied across the diode to quench the avalanche.

3.1.5 Comparison of PQCs and AQCs

PQCs and AQCs have some general advantages and disadvantages to each other which will be presented here.

PQCs have a limited maximum count rate due to the long recovery time (Eq. 3.4). In an AQC, the count rate can be enhanced greatly since the diode essentially can be switched back on directly after the desired hold-off time. The hold-off time may be much lower than the recovery time of the PQC while still avoiding afterpulsing.

Small pulse triggering is not a factor in AQCs, since the reset transition time for the bias voltage is very short. This results in the bias voltage being practically always either at the desired overvoltage or at a quenching voltage below the breakdown voltage.

The main advantages of using a PQC instead of an AQC in are simplicity and robustness. Designing a PQC is simpler, since the avalanche is quenched automatically as long as the ballast resistor is large enough. In an AQC, on the other hand, logic must be used to create a potentially high voltage pulse to quench the avalanche. If something goes wrong in an AQC, the device may lock in an always-on situation or in some other way dissipate too much power which leads to damage of the device. In a PQC, the large ballast resistor automatically protects the device from that kind of failure, a feature which makes the PQC an attractive choice for an experiment where high count rates and accurate photon timing is not a priority.

Chapter 4

Experimental work

The experimental part of this project included establishing a measurement setup with instruments that all are controllable from a computer, modifications and optimizations to an existing PQC circuit and some manual characterization of these changes. Also, a lot of work was spent making everything work with the computer and making it programmable in C++. This is described in detail in chapter 5.

4.1 Circuit

The circuit used was based on design and work by Christian Kurtsiefer, Yong-Su Kim and Vadim Makarov. It consists of a high voltage supply, the APD assembly with a sensing comparator, output pulse generation, some voltage converters and a thermo-electric cooler (TEC) with temperature control and monitoring.

The circuit is of the passive quenching type, as described in section 3.1.3. A PQC is generally a cheaper and simpler construction than an AQC, and the increased performance of an AQC is not needed for the experiment at hand. Another significant advantage of a PQC is that it is much more robust and tolerant to errors and mistreatment by an inexperienced student. In an AQC there is a significant danger of destroying the APD from overheating if something goes wrong and the circuit is locked in oscillation, for example.

4.1.1 Modifications

Modifications of the circuit were done to achieve two specific goals:

- To reduce electrical noise in order to be able to reduce the threshold voltage of the sensing comparator and thereby improve the performance of the system.
- To add interfaces for all properties that need to be controlled or measured during the experiment such that the experiment could be automated from the computer.

Noise reduction

In order to reduce noise, noisy switching power supplies were made unnecessary by using an external linear power supply or moved to a separate printed circuit board (PCB). This

included resoldering the temperature control assembly on a separate PCB and reconnecting it to the main PCB and the APD and TEC unit.

A Instek CM1-BS014 power supply is used to supply clean, low noise 12 V and −5 V.

Addition of threshold voltage control

When the electronic noise was reduced, a potentiometer (R4 in Fig. 4.1) was added to control the voltage at pin 3 of the differential receiver, which is the potential the avalanche pulse is compared to.

Replacing the high voltage supply

The on-board high voltage supply was replaced with an external unit. We used a PerkinElmer SPCM-AQR which has been reverse engineered by our group [12]. The APD of this unit was broken, employing it in this setup was a way to make use of the otherwise unneeded device. This circuit was modified such that it could be controlled by an applied external voltage. By connecting the analog output from a National Instruments USB-6215 data acquisition unit (0 V to 10 V) through a voltage divider connected to the high voltage supply, the output voltage can be controlled linearly from 0 to 282 V (Fig. 4.2).

Adding test points

Multiple test points were added in order to debug the circuit during modifications and to characterize the diode (Sec.4.3). Most notable of these is the avalanche pulse probe seen in the box of Fig. 4.1. A coaxial cable is used to preserve the bandwith of the very short avalanche pulse. The 100 nF DC-blocking capacitor was added to ensure that the probe does not alter the potential in the circuit at the sensing terminal. This probe, having total impedance of 520 Ω, will reduce the size of the avalanche pulse sensed by the differentiator by about 20%, but otherwise not affect the circuit notably.

4.1.2 Results of modifications

After determining the minimum threshold voltage, we want to keep it set to this level permanently. The overvoltage (V_{over}) was adjusted over the whole range of interesting values up to about 60 V while monitoring the avalanche pulse, the output of the first comparator and the final signal output on a HP Agilent 54845A 1.5 GHz oscilloscope. The minimum threshold voltage was found as the lowest voltage where the final output signal was stable with no double counts or other noise to the output signal due to ringing or other effects in the circuit over the whole range of bias voltages.

After the noise reductions, the threshold voltage could be reduced to 50 mV compared to about 150 mV before. This change enables the circuit to detect significantly smaller avalanches which in practice means that a lower reverse bias can be used, while still generating detectable signal to the comparator.

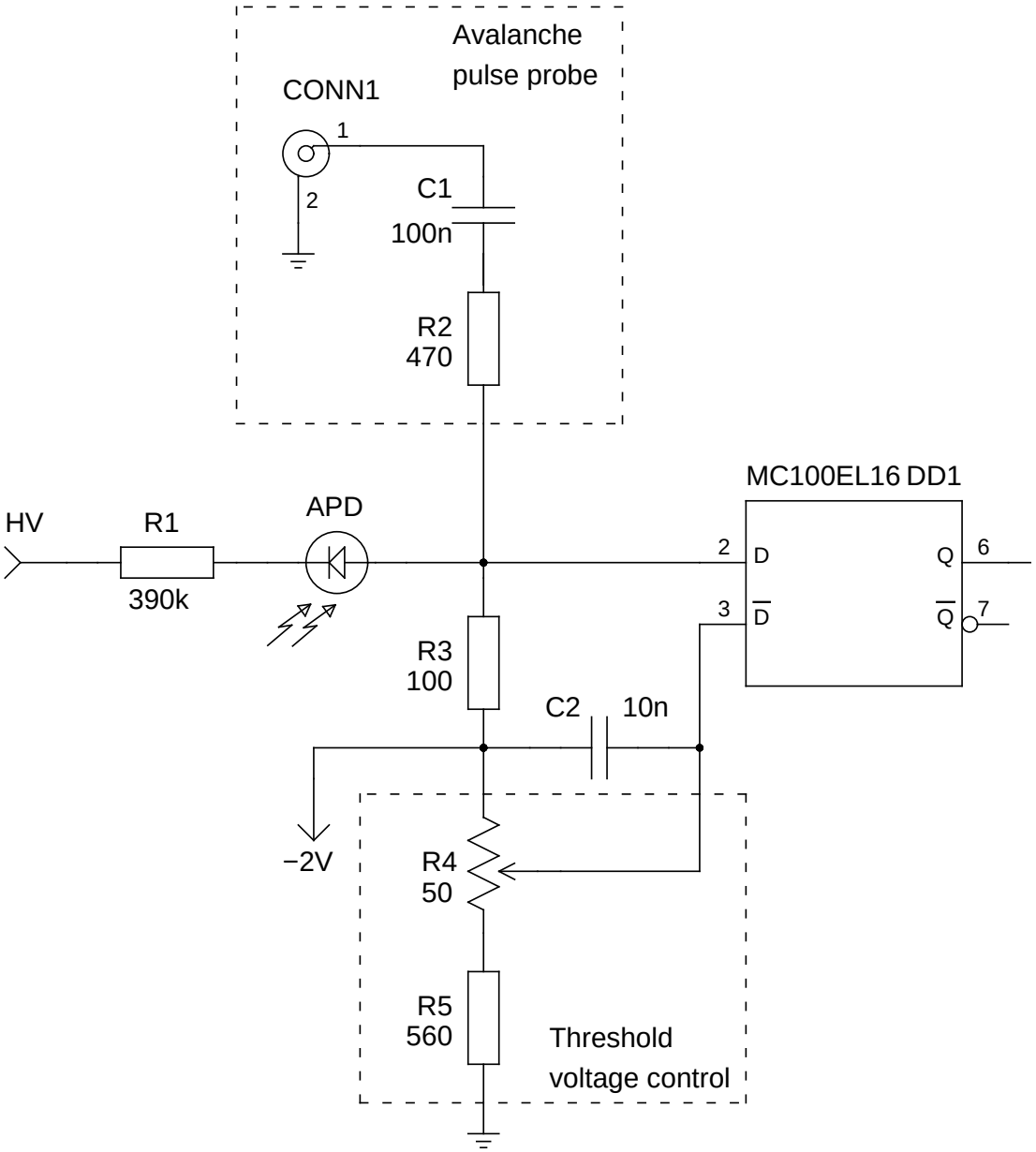


Figure 4.1: The APD circuitry used in this setup. Based on an original design by C. Kurtsiefer.

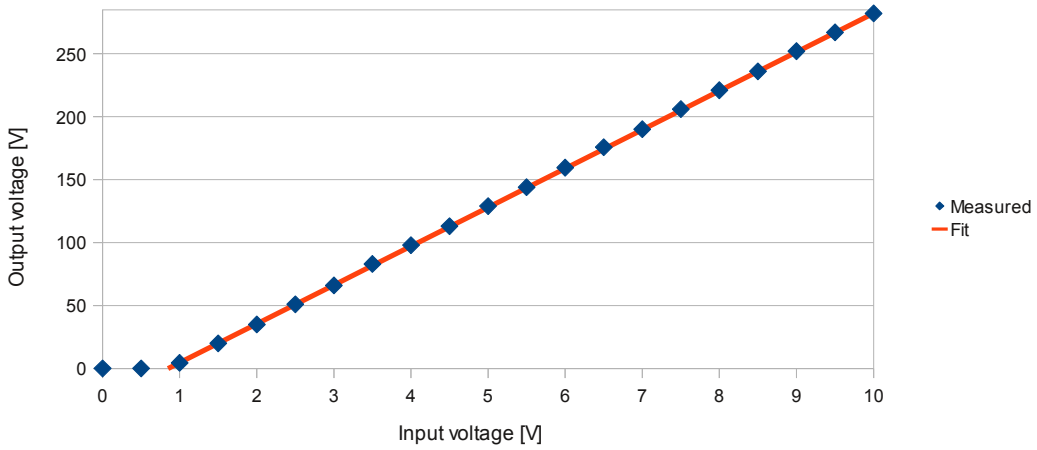


Figure 4.2: Input/output characteristic of the high voltage source. The threshold input voltage is around 0.8 V, which gives a slope of 31 V output per volt input above 0.8 V.

4.2 Experimental setup

The experimental setup was planned by Vadim Makarov, Sebastien Sauge and Lars Lydersen and is shown in Figs. 4.3 and 4.4. The intention of the setup is to be able to run the experiment automatically from a computer program. For this to work a number of instruments, both for characterization and control of experimental parameters, must be connected to the computer and work together in a C++ programming environment.

In order to run the complete experiment we need to be able to control both the signal laser and the damaging high power laser, enable and disable the shutter, adjust the bias voltage of the APD, measure the power of the damaging laser, measure the photon count per second registered by the APD, measure the bias voltage, measure the dark current through the APD and measure the temperature of the APD.

4.2.1 List of instruments used

Laser driver An Arroyo LaserPak model 485-08-05 custom current range 0-8500mA is used to power the 7 W laser. It has an RS-232 computer interface and current output to the laser diode.

Multimeters 3x Signametrics SMU-2055 to measure bias voltage, dark current and APD temperature. Connects to the computer via USB and comes with native C libraries. Supports all regular multimeter functionality. The USB connector is isolated from the measurement terminals.

Optical power meter Thorlabs PM100D with a Thorlabs S142C photodiode power sensor to measure the power of the damaging laser. Connects to the computer via USB and C libraries working through the National Instruments VISA framework.

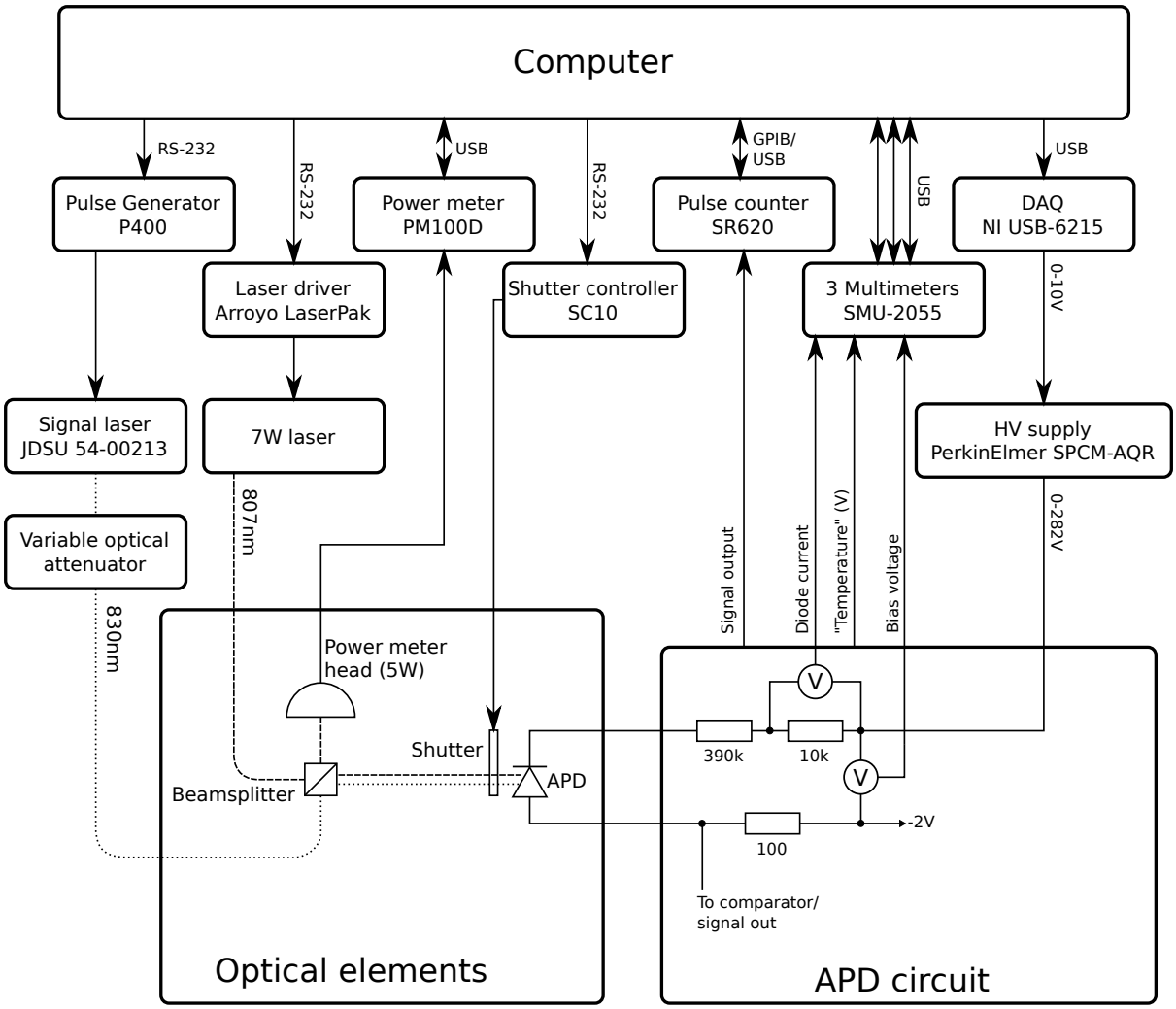


Figure 4.3: Schematic overview of the complete setup.

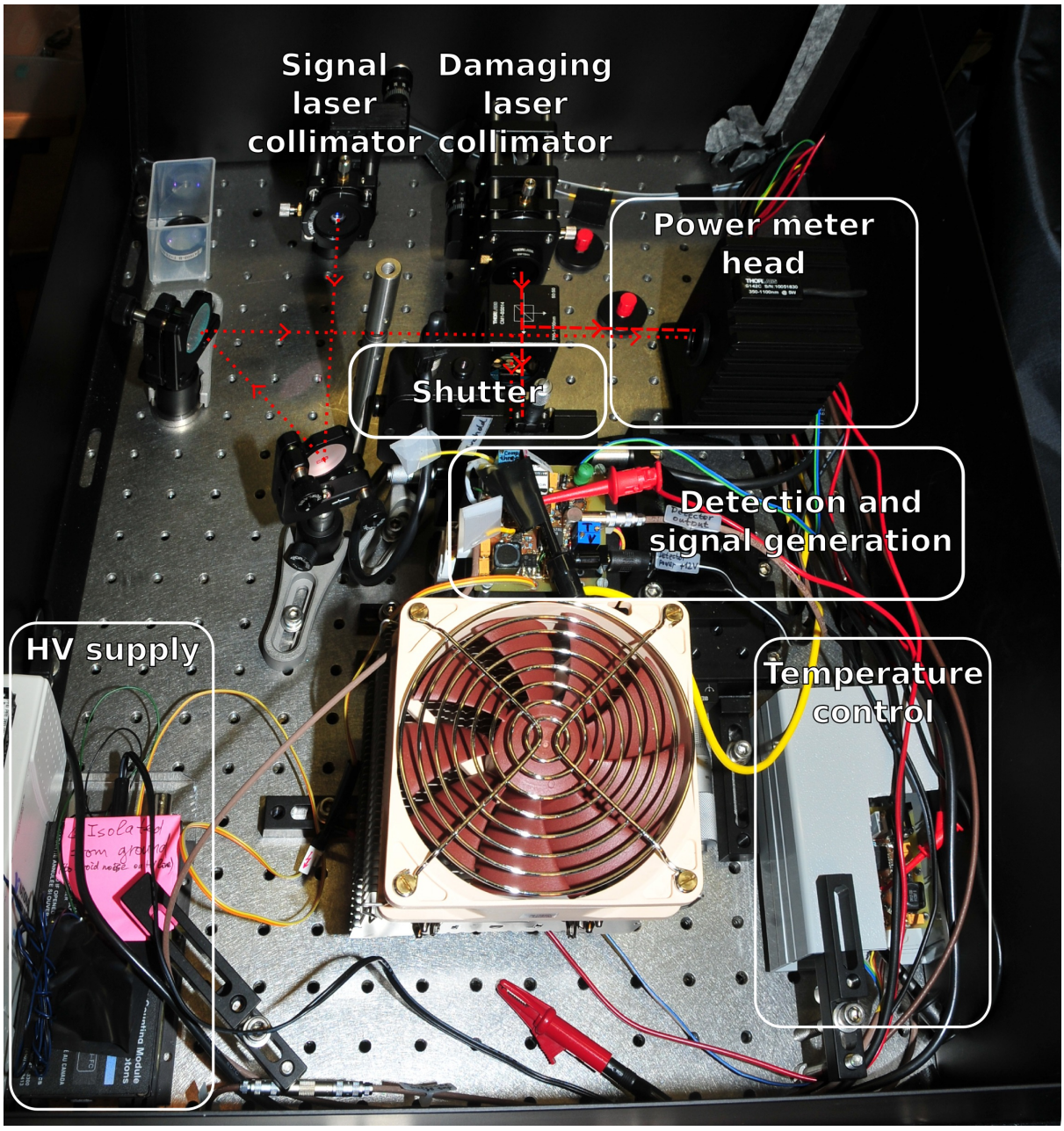


Figure 4.4: Optical part of the experimental setup.

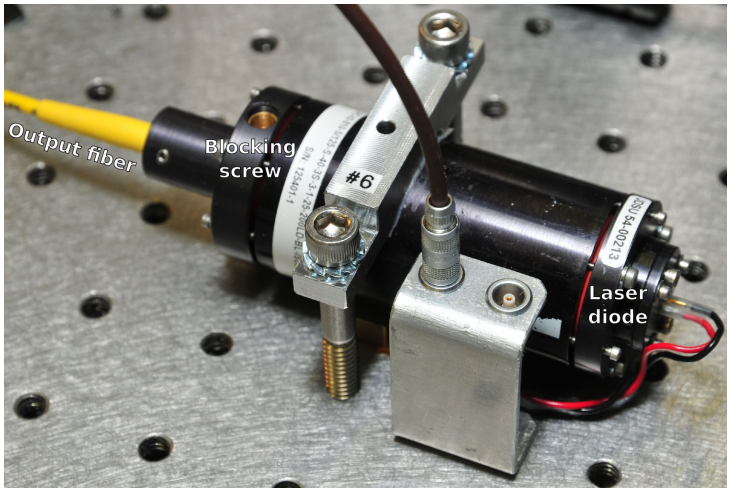


Figure 4.5: Signal laser assembly. OZ Optics laser-to-fiber coupler with an optical isolator installed with a JDSU 54-00213 laser diode, with a blocking screw to manually attenuate the laser output beam before it is coupled into the fiber.

Pulse counter Stanford research systems SR620 universal time interval counter for registering the photon counts from the detector. The device has both RS-232 and GPIB interfaces, we use the GPIB interface with a Prologix GPIB-USB controller since the RS-232 port is the standard but nevertheless rarely used DB-25 connector. Connected to the APD signal output through a coaxial cable.

Shutter controller Thorlabs SC10. Controls the shutter which blocks both lasers from illuminating the APD. Has an RS-232 computer interface.

Pulse generator Highland Technology P400. Used to generate short voltage pulses to power the signal laser. Has an RS-232 computer interface. Connected to the signal laser through a coaxial cable.

Controllable voltage output The analog output of a National Instruments USB-6215 data acquisition (DAQ) unit is used to control the voltage output controlling the high voltage supply. Is connected via USB and uses NI-DAQmx Base to be programmable in C in a linux environment.

4.2.2 Other devices used

Signal laser JDSU 54-00213 laser diode (Fig. 4.5). Connected to the variable attenuator by a 800 nm 5 µm core 125 µm cladding single-mode optical fiber. An 830 nm laser source is selected because the 800 nm range is suitable to be used with Si APDs, as discussed in the introduction of chapter 3.

The laser is controlled by using the P400 pulse generator as a voltage source. The maximum output voltage of the P400 (11.8 V) gives an operating current $I_{op} = \frac{11.8\,\mathrm{V} - 1.5\,\mathrm{V}}{50\,\Omega + 4\,\Omega} = 191\,\mathrm{mA}$ where the laser diode forward voltage drop is 1.5 V, the diode series resistance

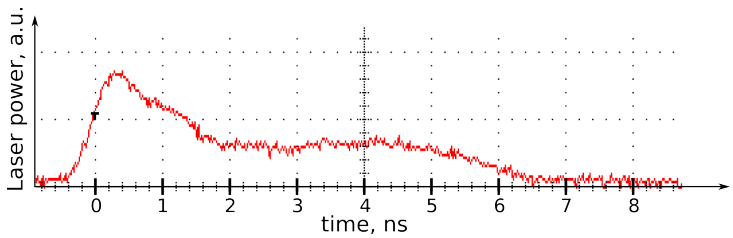


Figure 4.6: Unattenuated optical pulse power from the signal laser when powered by a 4 ns electrical pulse. Obtained from a Tektronix TDS 7104 with a Tektronix P6703B optical to electrical converter.

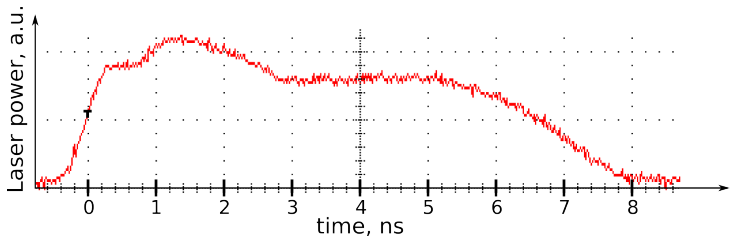


Figure 4.7: Unattenuated pulse from the signal laser when powered by a 5 ns electrical pulse. Conditions are otherwise equal to Fig. 4.6

is 4 Ω and the cable termination is 50 Ω (numbers from the datasheet, appendix B.1). This is lower than the stated typical operating current (270 mA), but enough to create a strong enough pulse for our purposes. Using an electric pulse of 4 ns (Fig. 4.6) was found to be the best choice. A longer electric pulse (Fig. 4.7) gives a much longer peak output (~5 ns) while a shorter pulse gives a very weak output with a broader profile. Neither of these pulses are ideal, but both are good enough for this experiment where we only need a regular supply of weak light pulses.

The output of the laser is attenuated by being partially blocked by an adjustable blocking screw before being coupled into the fiber, then attenuated again by a variable digital attenuator in order to achieve very dim light pulses.

Damaging laser Oclaro BMU7-808-02-R01 7 W laser diode operating at around 807 nm (datasheet and test data in appendix B.2). Connected to the setup by a 200 µm core multi-mode optical fiber to handle the high power.

Optical attenuator An OZ Optics DA-100 digitial variable attenuator was used to fine tune the attenuation of the light from the signal laser to get the desired pulse intensity.

Beam splitter Thorlabs CM1-BS014. Splits the damaging laser beam equally between the power meter and the APD. Also reflects the signal laser onto the APD.

Optical power sensor Thorlabs S142C photodiode power sensor rated for 5 W laser power.

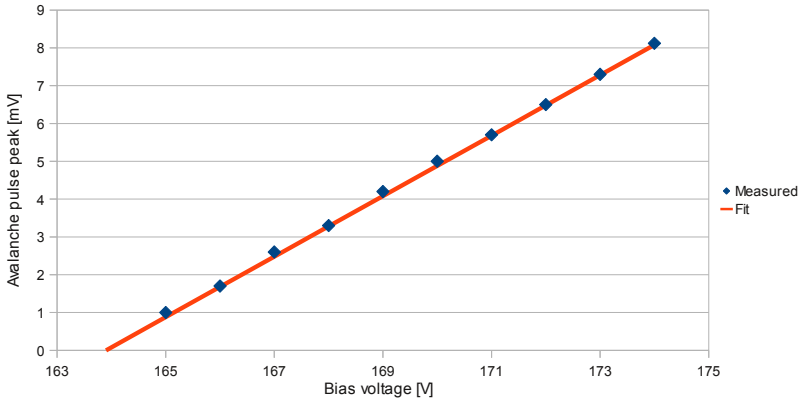


Figure 4.8: Measured peak avalanche pulse size at −25 °C as a function of bias voltage. The measurements are extrapolated to where the avalanche size would be 0 - at a bias voltage of 163.9 V

4.3 Diode breakdown voltage

After all modifications except replacing the HV supply were done, the breakdown voltage of the diode at −25 °C was investigated.

The breakdown voltage is an interesting figure because many of the APDs properties depend heavily on it. Since $V_{over} = V_{bias} - V_{br}$ (Sec. 3.1.1) and typical values for V_{br} are around 150 V and V_{over} around 10 V, it is clear that a relatively small change in the breakdown voltage will result in a relatively large change in overvoltage. This is interesting for our experiment, since it is one of the ways we may be able to alter the detector's characteristics. Since the applied bias voltage is constant, any small change in the breakdown voltage done by Eve will result in a significant change of behavior in Bob's detector.

In order to find the breakdown voltage we used the pulse generator connected to the signal laser as well as a synchronized trigger output to the oscilloscope. The laser signal was attenuated such that the circuit counted approximately one tenth of the pulses at around 10 V overvoltage, ensuring that we were working in the single photon regime. While triggering on the signal pulse we could see very small avalanches, thus giving a reasonably good approximation of the breakdown voltage. Using this method, the breakdown voltage was found to be 164.6 V at −25 °C. The true breakdown voltage is likely to be slightly lower than this since avalanches smaller than this is impossible to discern from noise.

It is also possible to extrapolate the breakdown voltage by making use of the fact that the peak avalanche current is proportional to the overvoltage (Eq. 3.1). By measuring the peak avalanche size at a range of different bias voltages and extrapolating to zero avalanche (Fig. 4.8) the breakdown voltage was found to be 163.9 V. This is a better approximation to the real breakdown voltage since this measurement is practically independent from noise, as seen from the good fit between the extrapolated line and the experimental data.

Chapter 5

Instrument library development

A major part of this project was to connect all the instruments listed in section 4.2.1 to the computer and get them to a state where they could be programmed without having to worry about low level interface commands. The instruments are connected via USB[1] RS-232[2] (serial port, some directly and some via RS-232 to USB adapters) and GPIB[3] (via a GPIB to USB adapter). In general, serial port and GPIB interfaces are simple in theory since they expect plain text commands and return data in plain text. USB devices on the other hand typically require more or less proprietary drivers with a given set of commands to control and read data from the device. This is often more convenient provided that the drivers are available for your computer setup and function as they are supposed to.

This experiment was set up on a computer running Ubuntu Linux. Ubuntu is a relatively easy to use Linux distribution and is used in other experiments in our lab.

5.1 Drivers and setup

5.1.1 Thorlabs PM100D

The Thorlabs power meter uses a USB interface and comes with drivers for Microsoft Windows working with NI VISA runtime by National Instruments. National Instruments only supply binaries for RPM[4]-based Linux distributions such as RedHat and SUSE which are not compatible with Ubuntu. However, these packages can be converted by the convenient tool "alien" in order to be installable on Ubuntu. After using alien to convert the packages NI VISA runtime was installed with only minor complications.

After installing NI VISA the instrument was still not detected by the system. After some debugging and troubleshooting it appeared to be due to a conflict between the NI VISA kernel module and the USB test and measurement class (usbtmc) module. Disabling usbtmc in `/etc/modprobe.d/blacklist.conf` fixed this issue.

Since the driver supplied was intended for Microsoft Windows, making it work under Linux was not necessarily an easy task. However, after unpacking the windows driver

[1] Universal Serial Bus
[2] Recommended Standard 232
[3] General Purpose Interface Bus
[4] Originally Red Hat Package Manger, now RPM Package Manager

and acquiring the source code, only minor modifications (Sec. A.1) were required in order to compile it as a dynamic library in Linux.

5.1.2 National Instruments USB-2615

The DAQ is connected via USB and uses National Instruments' NI-DAQmx base runtime as an interface to C/C++. NI-DAQmx base is, like NI VISA, only available as RPM packages. The installation procedure of NI-DAQmx base is more complicated due to a larger number of packages and some scripts intended to be run during installation which would fail without doing some manual work along the way. After installing NI-DAQmx base the DAQ worked fine, which was verified by compiling and running some of the sample programs supplied with the driver.

5.1.3 Signametrics SMU-2055

Signametrics has precompiled dynamic libraries which worked fine in our installation. The installation basically consisted of copying the libraries to a library directory and setting up some user access in `/etc/udev/rules.d/`, everything was described in the readme file. Also here some example code was supplied which was used to check that the instruments were working.

5.1.4 Stanford Research SR620

The SR620 has a GPIB connector which we use with a GPIB to USB converter. When the GPIB converter is properly set up, this interface works just like a regular serial port device.

5.1.5 RS-232 devices

The remaining instruments all have RS-232 serial port interfaces. The Arroyo laser driver and the Thorlabs shutter controller are connected directly to serial ports on the computer, the rest use RS-232 to USB converters which have their drivers included in the default Ubuntu kernel.

5.2 Supporting software

GtkTerm GtkTerm is a terminal for communicating with devices connected to serial ports. It is a simple, but very convenient application when working with serial port devices since it provides a way to test and debug commands quickly before incorporating them in the program. It is also very convenient for troubleshooting whenever something stops working without any apparent reason - which is an unfortunate reality once in a while when working with such a setup.

5.3 Serial port programming

In a Linux system, a serial port works just like a file. This means, in principle, that we can write to it and read from it just like we would for a regular text file. Like most things in the real world, it's not quite that simple. The port needs to be set up and configured in order to work. Some of the settings regard the computer side of the communications and can thus be set equally for all instruments, while others need to be adjusted specifically according to each instrument's setup.

Configuring and controlling serial ports in C/C++ in Linux is done using the POSIX[5] terminal interface [14]. In practice, this is accomplished by including the file `termios.h` and using its functions and the termios control structure. The base setup used here is shown in the `configurePort()`-method in appendix A.2.1.

5.4 Architecture

5.4.1 Intention

The the software libraries were intended to create a programming environment where all instruments behaved more or less the same, abstracting away the type of interface, instrument specific syntaxes and other quirks that take focus away from programming the characterization and laser damage routines. The program should automatically figure out where the instruments are connected and provide a basic `init()` function for each instrument that will initialize it and make it ready for use.

5.4.2 Solution

An object oriented design was chosen from the start, since this makes for a logical and convenient way to work with many different instances which share some of the same functionality. The basic idea is that each instrument has its own class which can be instantiated to an object containing all the functionality available in the instrument.

In order to avoid writing almost the same code for each instrument, all instrument classes using serial port interfaces inherit from a master `rs232_base` class (Sec.A.2) which contains some common functionality for connecting to, configuring and disconnecting from a serial port. The `rs232_base` class also contains a method for retrieving an identifier string from the connected device which is useful for automatically detecting which port each device is connected to, as well as some methods used to check for and handle errors that might occur.

5.5 Progress and results

This sections describes the status of development of the instrument libraries and a main control program combining everything together.

[5] Portable Operating System Interface for Unix

5.5.1 Instrument libraries

The instrument libraries were developed in such a way that once a common base with some utility functions were developed, adding more functionality later would be a matter of adding a new method containing just a few lines of code. All instruments except the PM100D are at a stage where adding more functionality is very easy. The SC10 has all its functionality implemented and working. The PM100D library only has a function for detecting if the PM100D is connected returning which identifier it uses.

5.5.2 Control program

The main program which is intended to run the entire experiment is at an early stage. It has a function that detects all instruments by querying all serial ports for an identification string and runs some device specific functions to check for presence of the USB devices. This function is vital for further development and debugging, since setting this up manually is tedious and error prone.

During the development of the instrument libraries some simple functionality tests for the different instruments were created in order to ensure that everything actually worked as expected. These tests are kept in the main file as commented out blocks of code for future reference and debugging.

Chapter 6

Conclusion

The goal of this project was to prepare an experimental setup for running automated laser damage and characterization of an APD. This task included setting up and modifying a single photon detector assembly, setting up and connecting all instruments to the computer and installing and configuring a computer environment in which all instruments are recognized and easily programmable in C++.

The detector circuit was modified to reduce electronic noise and lower the comparator threshold voltage. After disabling and replacing noisy circuit elements the threshold voltage could be reduced by about 2/3, significantly improving the circuit's small pulse sensitivity. The circuit operation was tested by manually using many of the instruments intended to be controlled by the computer for the final experiment.

All instruments have been set up, connected to their respective probes and other devices and connected to the computer. All required drivers and software have been installed on the computer, and the architecture of a collection of software libraries for communicating with the different devices have been constructed. Programming of base libraries for almost all instruments are well underway and easily extensible with more functionality. The computer program is able to detect all connected instruments and link them to unique identifiers which simplifies further development and possible modifications to the setup at a later time.

Controlled laser damage could prove to be a very powerful tool for an eavesdropper. Work in our lab before this project was started has indicated that damaged detectors could significantly improve the effectiveness of existing attacks, making continued work in this area very interesting.

6.1 Further work

As this project's focus was to prepare an experimental setup, there is naturally planned more work developing and using this setup.

First, there is some remaining programming on the base libraries and the core of the control program. The control program needs to have a good logging facility which writes data in a format which is usable for further analysis. Most of the instrument libraries will need some more functions implemented in order to do everything they are needed to do for running the experiment, as it is not possible to predict exactly which functions

will be required in advance. Some of the instruments libraries also need error checking implemented in order to handle any technical errors that might occur.

Then, a procedure for characterizing the APD must be planned and programmed. This routine must check a number of different properties of the diode to determine if there are any discrepancies in the characteristics. The characterization procedure will be tested and debugged on both a healthy APD and two the two damaged APDs mentioned in section 2.2.3.

Finally, a program that repeatedly increases the applied optical power from the damaging laser in small increments and runs the characterization routine must be programmed. This program must, when some predefined amount of discrepancy from normal behavior is reached, stop and let the experimenter proceed manually with further testing. The experimenter should then continue to apply the damaging laser in small increments while characterizing the diode fully between each damaging illumination. From the data acquired one can investigate which new properties are appearing and which consequences these might have for the setup's susceptibility to attacks by an eavesdropper.

Appendix A

Code examples

A.1 Patch to Thorlabs PM100D driver

```
--- PM100D_Drv-win.c
+++ PM100D_Drv.c
@@ -47,4 +47,4 @@
 #include <string.h>
-#include <utility.h>
 #include <ctype.h>
+#include <unistd.h>
 #include <visa.h>
@@ -87,3 +87,3 @@
 // dynamic error list
-typedef struct errDescrDyn_t  errDescrDyn_t;
+typedef struct errDescrDyn_t  errDescrDyn_t_;
 typedef struct errDescrDyn_t
@@ -92,3 +92,3 @@
    ViChar         descr[DRV_BUF_SIZE];
-   errDescrDyn_t  *next;
+   errDescrDyn_t_  *next;
 }  errDescrDyn_t;
@@ -1350,3 +1350,3 @@
          tmocnt--;
-         Delay(0.5);
+         usleep(500000);
          err2 = viClear(instr);
```

A.2 RS-232 base class

This is the main RS-232 class from which all serial port devices' libraries inherit.

A.2.1 Definition

```
#ifndef RS232_BASE_H
#define RS232_BASE_H

#include <iostream>
#include <cstring>
#include <termios.h>

using namespace std;
namespace RS232ns{

class RS232{
public:
        // Contructor. Argument is the port the device is found on, e.g. "/dev/ttyS0"
```

```
        RS232(string filedesc);
        // Destructor. Will close the port if it's still open
        ~RS232();
        // Initialize the serial port
        bool init();
        // Open the serial port
        void openPort();
        // Close the serial port
        void closePort();
        // Set up the serial port to work as we want it
        void configPort();
        // Get instrument information
        string idn();
        // Check if the instrument has failed in any way
        bool hasFailed();
        // Get last (or current, if available) error string from the instrument
        string getError();

protected:
        // Termios object containing options for the RS-232 connection
        struct termios options;
        // Newline character(s) used for command termination
        string nl;
        // True if the instrument has failed in any way
        bool failed;
        // The string last read from the instrument
        string lastReadOut;

        // Write the given command to the instrument.
        // Read data after issuing the command if readAfter is true
        string writeCmd(string command, bool readAfter);
        // Write the given command to the instrument.
        // Read data after issuing the command.
        string writeCmd(string command);
        // Read output from the instrument
        string readOut();
        // Instrument dependent method to get current error string
        string _getError();
        // Instrument dependent method to check if an error has occured
        bool checkError();
        // Strip newline character(s) from the given string
        string stripNewlines(string& str);

private:
        // File descriptor for the serial port
        int port;
        // Path identifying the port, eg /dev/ttyS0
        string port_identifier;
        // Initial termios settings, saved in order to be restored when we're done
        struct termios termios_save;
};

} // End namespace RS232ns
#endif


#include <iostream>
#include <cstdlib>
#include <cstring>
#include <cstdio>
#include <fcntl.h> //File control
#include <termios.h> //POSIX terminal control
#include <sys/ioctl.h>

#include "rs232_base.h"

using namespace std;
using namespace RS232ns;

namespace RS232ns{
```

```
#define RS232_READ_BUF_SIZE 8192

// Non-static functions
RS232::RS232(string filedesc)
{
        port_identifier = filedesc;
        port = -1;
        failed = false;
        lastReadOut = "";

        // Set default newline character(s)
        nl = "\r";
}

RS232::~RS232()
{
        closePort();
}

bool RS232::init()
{
        openPort();

        if (port<0)
        {
                cout << "RS232: Failed to open port " << port_identifier << endl;
                return false;
        }

        // Save current settings in order to restore at destructor
        tcgetattr(port, &options);
        memcpy(&termios_save, &options, sizeof(struct termios));

        configPort();

        return true;
}

void RS232::configPort()
{
        // Typical way of setting baud rate. Actual baud rate are contained
        // in the c_ispeed and c_ospeed members
        cfsetispeed(&options, B9600);
        cfsetospeed(&options, B9600);

        // Enable the receiver and set local mode
        options.c_cflag |= (CLOCAL|CREAD);

        // No parity, 8bits, 1 stop bit (8N1)
        options.c_cflag &= ~CSIZE;
        options.c_cflag &= ~CSTOPB;
        options.c_cflag &= ~PARENB;
        options.c_cflag |= CS8;

        // Turn off flow control
        options.c_cflag &= ~CRTSCTS;//CNEW_RTSCTS;

        // Make sure that Canonical input is off (raw input data)
        options.c_lflag &= ~(ICANON | ECHO | ECHOE | ISIG);

        // Raw output data
        options.c_oflag &= ~OPOST;

        // Turn off software control flow
        options.c_iflag &= ~(IXON | IXOFF | IXANY);

        // Ignore parity errors and break conditions
        options.c_iflag |= IGNPAR | IGNBRK;

        // Convert NL to CR, ignore CR
```

```
        // (ie strip newlines from data received on serial port)
        options.c_iflag |= INLCR | IGNCR;

        // Get at least one byte of data with no timeout when we read from the port
        options.c_cc[VMIN] = 1;
        options.c_cc[VTIME] = 0;

        // Write settings to the port
        tcsetattr (port, TCSANOW, &options);

        // Flush output and input buffers
        tcflush(port, TCOFLUSH);
        tcflush(port, TCIFLUSH);
}

void RS232::openPort()
{
        // Open in read/write mode, do not set as controlling tty,
        // do not wait for data from the serial port
        port = open(port_identifier.c_str(), O_RDWR | O_NOCTTY | O_NDELAY );

        // Required to get data when we use O_NDELAY
        fcntl(port, F_SETFL, 0);
}

void RS232::closePort()
{
        if (port > 0)
        {
                tcsetattr(port, TCSANOW, &termios_save);
                tcflush(port, TCOFLUSH);
                tcflush(port, TCIFLUSH);
                close(port);
                port = -1;
        }
}


string RS232::idn()
{
        return writeCmd("*idn?");
//         return readOut();
}


string RS232::writeCmd(string command, bool readAfter)
{
        // Append newline character(s)
        command = command + nl;

        // Flush any pending input/output data
         tcflush(port, TCIOFLUSH);

        int result;
        result = write(port, command.c_str() , static_cast<int>(command.length()));
        if (result == -1)
        {
                cout << "RS232: ERROR: write error on port " << port_identifier
                        << " with result " << result << endl;
                failed = true;
                return "";
        }

        if (!readAfter)
                return "";

        // Wait 100 msecs to allow the instrument to produce some data
        usleep(100000);

        return readOut();
```

```
}

string RS232::writeCmd(string command)
{
        return writeCmd(command, true);
}


string RS232::readOut()
{
        // Check if there's anything to read
        int bytes_avail;
        ioctl(port, FIONREAD, &bytes_avail);
        if (bytes_avail < 1)
                return "";

        string output;
        int bytes_read = RS232_READ_BUF_SIZE;
        char buf[RS232_READ_BUF_SIZE] = "";
        int j = 0;
        // Read while we get data from the instrument
        while (bytes_read == RS232_READ_BUF_SIZE)
        {
                bytes_read = read(port, buf, RS232_READ_BUF_SIZE);
                output += buf;
        }

        // Store the last read data for convenience
        lastReadOut = output;

        return output;
}

bool RS232::hasFailed()
{
        return failed;
}


string RS232::_getError()
{
        return "RS232: This instrument must be initialized by a instrument specific child class for getError() to work.";
}


string RS232::getError()
{
        return _getError();
}

bool RS232::checkError()
{
        cout << _getError() << endl;
        return true;
}

string RS232::stripNewlines(string& str)
{
        size_t pos = str.find(nl);
        if (pos != string::npos)
        {
                str.erase(pos, nl.length());
        }
}

}// End namespace RS232ns
```

A.3 SC10 implementation

The SC10 shutter controller has a limited set of functionality and makes a good example of how a complete library using the RS-232 base class. This example skips some C++ formalities and some of the functionality for clarity.

```
// Contructor. Argument is the port the device is found on, e.g. "/dev/ttyS0"
SC10::SC10(string filedesc): RS232(filedesc)
{
        // Ensure we use the correct newline character for this device
        nl = "\r";

        // Set the error identifier
        error_identifier = "Command error";

        current_error = "";
}

// Write the given command to the instrument.
// Read data after issuing the command since readAfter=true for the SC10
string SC10::writeCmd(string cmd, bool readAfter)
{
        // Run the real writeCmd()
        string output = RS232::writeCmd(cmd, readAfter);

        if (output.length() == 0)
                return "";

        // Hack away!

        // Strip the prefixed command returned from the instrument
        if (output.find(cmd) == 0)
        {
                output = output.substr(cmd.length());
        }

        // Strip annoying "> " which is returned when the instrument is ready
        if (output.find("> ") == output.length()-2)
        {
                output = output.substr(0, output.length()-2);
        }

        lastReadOut = output;

        // Reset current error
        current_error = "";

        if (checkError())
        {
                // An error occured

                // Erase newlines from the command
                stripNewlines(cmd);

                cout << "ERROR: SC10: And error occured while running command " << cmd <<
                ". " << endl << "Error message from device: " << _getError() << endl;
        }

        return output;

}

bool SC10::checkError()
{
        if (lastReadOut.length() == 0)
                return false;

        if (lastReadOut.find(error_identifier) != string::npos)
        {
```

```
                failed = true;
                current_error = lastReadOut;
                return true;
        }
        return false;
}

string SC10::_getError()
{
        return current_error;
}

// Set operating mode
// enum sc10_modes { MANUAL=1, AUTO, SINGLE, REPEAT, EXT_GATE };
void SC10::setMode(SC10::sc10_modes mode)
{
        stringstream cmd;
        cmd << "mode=" << mode;
        this->writeCmd(cmd.str());
}

SC10::sc10_modes SC10::getMode()
{
        string out = writeCmd("mode?");
        switch (atoi(out.c_str()))
        {
                case AUTO:
                        return AUTO;
                case MANUAL:
                        return MANUAL;
                case SINGLE:
                        return SINGLE;
                case REPEAT:
                        return REPEAT;
                case EXT_GATE:
                        return EXT_GATE;

                default:
                        cout << "SC10: ERROR: Unrecognized mode: " << out << endl;
                        return AUTO;
        }
}

// Check if the shutter is enabled
bool SC10::getEnabled()
{
        string enabled = writeCmd("ens?");

        return enabled == "1";
}

// Enable shutter
void SC10::enable()
{
        if (!getEnabled())
                writeCmd("ens");
}

void SC10::disable()
{
        if (getEnabled())
                writeCmd("ens");
}

// Set open time in ms
void SC10::setOpen(int msecs)
{
        stringstream cmd;
        cmd << "open=" << msecs;
        writeCmd(cmd.str());
```

```
}

int SC10::getOpen()
{
        string str = writeCmd("open?");
        return atoi(str.c_str());
}

// Set closed time in ms
void SC10::setClosed(int msecs)
{
        stringstream cmd;
        cmd << "shut=" << msecs;
        writeCmd(cmd.str());
}

int SC10::getClosed()
{
        string str = writeCmd("shut?");
        return atoi(str.c_str());
}
```

Appendix B

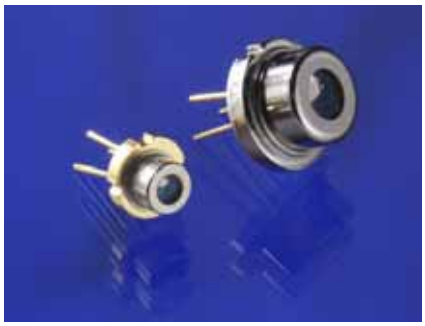
Datasheets

B.1 JDSU 54-00213

This is the datasheet for the JDSU 54-00213 laser diode.

Diode Lasers, Single-mode 50 to 200 mW, 810/830/852 nm

54xx Series



Key Features

- **200 mW kink-free power**
- **Narrow spectral width**
- **High efficiency**
- **Low astigmatism**
- **High reliability**

Applications

- Illumination
- Printing
- Sensing
- Medical applications
- Imaging

High-resolution applications including optical data storage, image recording, spectral analysis, printing, point-to-point free-space communications and frequency doubling all require diffraction-limited sources. Faster writing, wider dynamic range and better signal-to-noise ratio may be achieved with JDSU's high-reliability 5400 Series single-mode diode lasers.

Available in power levels up to 200 mW kink-free, this advanced diode laser combines a quantum well structure and a real-refractive index-guided single-mode waveguide to provide high power, low astigmatism, narrow spectral width and a single spatial mode Gaussian far field. Our 5400 Series diode lasers are among the most reliable high-power diode lasers available in the industry today.

The 5400 Series diode lasers operate in single longitudinal mode under some conditions. Like in all Fabry-Perot index-guided diode lasers, spectral broadening, mode hopping and longitudinal mode instability may occur due to small changes in drive current, diode-junction temperature or optical feedback.

The unique diode structure features high reliability with long operating life and very low early failure rate. The highest brightness (20 MW/cm^2 steradian) is provided by our 5430.

2

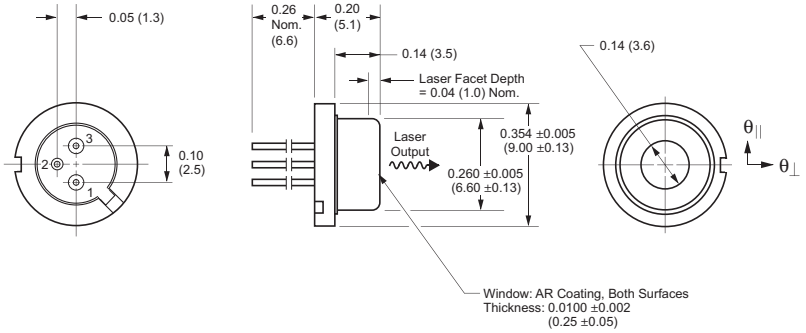
Dimensions Diagrams

(Specifications in inches [mm] unless otherwise noted.)
Standard Tolerances
inches: x.xx = ±0.02 mm: x.x = ±0.5
x.xxx = ±0.010 x.xx = ±0.25

Package Style: SOT-148 Window (G1)

Pinout

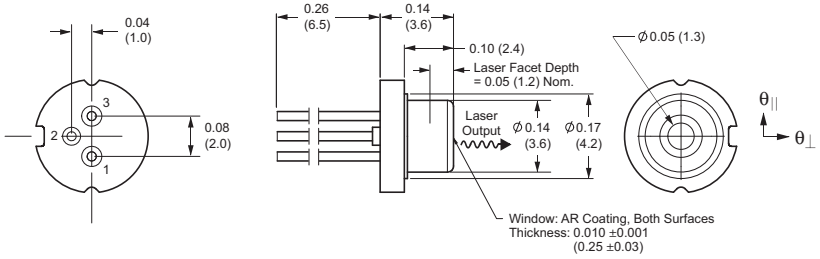
Pin	Description
1	Laser cathode (–)
2	Laser anode, MPD cathode and case ground
3	Monitor photodiode anode (+)



Package Style: TO-56 Window (J1)

Pinout

Pin	Description
1	Laser cathode (–)
2	Laser anode, MPD cathode and case ground
3	Monitor photodiode anode (+)



Available Configurations

5400 Series	5410 Series	5420 Series	5430 Series
5401-G1	5411-G1	5421-G1	5431-G1
5401-J1	5411-J1	5421-J1	5431-J1

Electro-optical Specifications

Parameter	Symbol	5400 Series			5410 Series			Unit
		Min.	Typ.	Max.	Min.	Typ.	Max.	
Laser Characteristics								
CW output power, kink-free[2]	P_O	–	–	50	–	–	100	mW
Center wavelength	λ_C	–	(note[5])	–	–	(note[5])	–	
Spectral width[1]	$\Delta\lambda$	–	3	5	–	3	5	nm
Slope efficiency	$\eta D = P_O/(I_{op}–I_{th})$	0.75	0.85	–	0.75	0.85	–	mW/mA
Conversion efficiency	$\eta = P_O/(I_{op}V_{op})$	–	30	–	–	30	–	%
Emitting dimensions	W x H	–	3 x 1	–	–	3 x 1	–	µm
FWHM beam divergence								
Parallel to junction	$\theta_{//}$	–	9	–	–	9	–	degrees
Perpendicular to junction	$\theta_{\perp}$	–	30	–	–	30	–	degrees
Threshold current	I_{th}	–	35	45	–	35	45	mA
Operating current	I_{op}	–	95	105	–	160	170	mA
Operating voltage	V_{op}	–	(note[4])	–	–	(note[4])	–	
Series resistance	R_S	–	4.0	6.0	–	4.0	6.0	Ω
Thermal resistance	R_{th}	–	60	–	–	60	–	°C/W
Recommended case temperature	T_C	-20	–	30	-20	–	30	°C
Absolute Maximum Ratings								
Reverse voltage	V_{rl}	–	–	3	–	–	3	V
Case operating temperature	T_{op}	-20	–	50	-20	–	50	°C
Storage temperature range	T_{stg}	-40	–	80	-40	–	80	°C
Lead soldering temperature	T_{is}	–	–	250	–	–	250	°C (5 sec.)
Monitor Photodiode								
Sensitivity								
G1 package	–	0.1	–	20	0.1	–	20	µA/mW
J1 package	–	3.0	–	24	3.0	–	24	µA/mW
Capacitance	–	–	6	–	–	6	–	pF
Breakdown voltage	V_{bd}	–	25	–	–	25	–	V
Operating voltage	V_{op}	–	10	–	–	10	–	V

1. Emission bandwidth for 90% integrated power.
2. Typical values at 25 °C and 0.6 NA collection optics.
3. Features common to all 5400 series diode lasers include:
   a. Duty factor of 100%.
   b. Temperature coefficient of wavelength is approximately 0.3 nm/°C.
   c. Temperature coefficient of threshold current can be modeled as:
   $I_{TH2} = I_{TH1} \exp[(T_2 – T_1)/T_0]$ where T_0 is a device constant of about 110 °K.
   d. Temperature coefficient of operating current is approximately 0.5 to 0.7% per °C.
4. Forward voltage is typically: $V_f = 1.5\ V + I_{op} \times R_s$.
5. Wavelength ranges for the 5400 and 5410 series:
   800-820 nm
   810-850 nm
   842-862 nm
   A variety of part numbers are available that each designate a particular subset within these wavelength ranges. Consult tables on page 5.
6. Astigmatism is less than 5 µm.

4

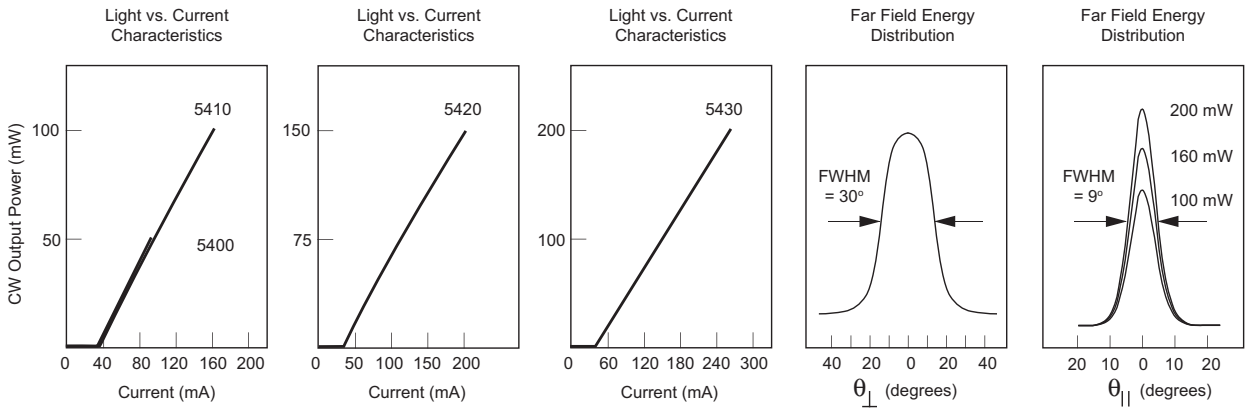
Electro-optical Specifications | Continued

Parameter	Symbol	5420 Series			5430 Series			Unit
		Min.	Typ.	Max.	Min.	Typ.	Max.	
Laser Characteristics								
CW output power, kink-free[2]	P_O	–	–	150	–	–	200	mW
Center wavelength	λ_C	–	(note[5])	–	–	(note[5])	–	
Spectral width[1]	$\Delta\lambda$	–	3	5	–	3	5	nm
Slope efficiency	$\eta D = P_O/(I_{op}–I_{th})$	0.75	0.85	–	0.75	0.85	–	mW/mA
Conversion efficiency	$\eta = P_O/(I_{op}V_{op})$	–	30	–	–	30	–	%
Emitting dimensions	W x H	–	3 x 1	–	–	3 x 1	–	µm
FWHM beam divergence								
Parallel to junction	$\theta_{//}$	–	9	–	–	9	–	degrees
Perpendicular to junction	$\theta_{\perp}$	–	30	–	–	30	–	degrees
Threshold current	I_{th}	–	35	45	–	40	50	mA
Operating current	I_{op}	–	210	230	–	270	300	mA
Operating voltage	V_{op}	–	(note[4])	–	–	(note[4])	–	
Series resistance	R_S	–	4.0	6.0	–	4.0	6.0	Ω
Thermal resistance	R_{th}	–	60	–	–	60	–	°C/W
Recommended case temperature	T_C	-20	–	30	-20	–	30	°C
Absolute Maximum Ratings								
Reverse voltage	V_{rl}	–	–	3	–	–	3	V
Case operating temperature	T_{op}	-20	–	50	-20	–	50	°C
Storage temperature range	T_{stg}	-40	–	80	-40	–	80	°C
Lead soldering temperature	T_{is}	–	–	250	–	–	250	°C (5 sec.)
Monitor Photodiode								
Sensitivity	–	0.1	–	20	0.1	–	20	µA/mW
Capacitance	–	–	6	–	–	6	–	pF
Breakdown voltage	V_{bd}	–	25	–	–	25	–	V
Operating voltage	V_{op}	–	10	–	–	10	–	V

1. Emission bandwidth for 90% integrated power.
2. Typical values at 25 °C and 0.6 NA collection optics.
3. Features common to all 5400 series diode lasers include:
   a. Duty factor of 100%.
   b. Temperature coefficient of wavelength is approximately 0.3 nm/°C.
   c. Temperature coefficient of threshold current can be modeled as:
   $I_{TH2} = I_{TH1} \exp[(T_2 – T_1)/T_0]$ where T_0 is a device constant of about 110 °K.
   d. Temperature coefficient of operating current is approximately 0.5 to 0.7% per °C.
4. Forward voltage is typically: $V_f = 1.5\ V + I_{op} \times R_s$.
5. Wavelength ranges for the 5420 series:
   800-820 nm
   810-850 nm
   842-862 nm
   Wavelength range for the 5430 series is limited to 820-840 nm.
   A variety of part numbers are available that each designate a particular subset within these wavelength ranges. Consult tables on page 5.
6. Astigmatism is less than 5 µm.

5

Typical Optical Characteristics



Ordering Information

For more information on this or other products and their availability, please contact your local JDSU account manager or JDSU directly at 1-800-498-JDSU (5378) in North America and +800-5378-JDSU worldwide or via e-mail at customer.service@jdsu.com.

Sample: 54-00202

Part Number	Power	Wavelength	Package
54-00202	50 mW	810 (±5)	5.6 mm TO-56
54-00203	50 mW	830 (-10/+20)	5.6 mm TO-56
Call for part number	50 mW	830 (-10/+20)	9 mm SOT-148
54-00204	50 mW	852 (±10)	5.6 mm TO-56
54-00205	100 mW	810 (±5)	5.6 mm TO-56
54-00206	100 mW	830 (-10/+20)	5.6 mm TO-56
54-00207	100 mW	852 (±10)	5.6 mm TO-56
Call for part number	150 mW	810 (±5)	9 mm SOT-148
54-00210	150 mW	810 (±5)	5.6 mm TO-56
Call for part number	150 mW	830 (±10)	9 mm SOT-148
54-00211	150 mW	830 (±10)	5.6 mm TO-56
Call for part number	150 mW	852 (±10)	9 mm SOT-148
54-00212	150 mW	852 (±10)	5.6 mm TO-56
Call for part number	200 mW	830 (±10)	9 mm SOT-148
54-00213	200 mW	830 (±10)	5.6 mm TO-56
54-00214	200 mW	852 (±10)	5.6 mm TO-56

User Safety

Safety and Operating Considerations

The laser light emitted from this diode laser is invisible and may be harmful to the human eye. Avoid looking directly into the diode laser or into the collimated beam along its optical axis when the device is in operation.

CAUTION: THE USE OF OPTICAL INSTRUMENTS WITH THIS PRODUCT WILL INCREASE EYE HAZARD.

Operating the diode laser outside of its maximum ratings may cause device failure or a safety hazard. Power supplies used with the component must be employed such that the maximum peak optical power cannot be exceeded. CW diode lasers may be damaged by excessive drive current or switching transients. When using power supplies, the diode laser should be connected with the main power on and the output voltage at zero. The current should be increased slowly while monitoring the diode laser output power and the drive current.

Device degradation accelerates with increased temperature, and therefore careful attention to minimize the case temperature is advised. For example, life expectancy will decrease by a factor of four if the case is operated at 50 °C rather than 30 °C.

A proper heatsink for the diode laser on a thermal radiator will greatly enhance laser life. Firmly mount the laser on a radiator with a thermal impedance of less than 2 °C/W for increased reliability.

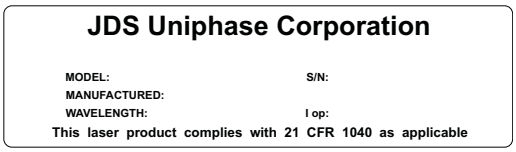
ESD PROTECTION – Electrostatic discharge is the primary cause of unexpected diode laser failure. Take extreme precaution to prevent ESD. Use wrist straps, grounded work surfaces and rigorous antistatic techniques when handling diode lasers.

Labeling

21 CFR 1040.10 Compliance

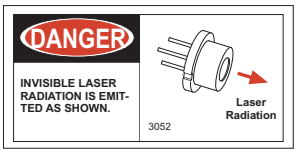
Because of the small size of these devices, each of the labels shown is attached to the individual shipping container. They are illustrated here to comply with 21 CFR 1040.10 as applicable under the Radiation Control for Health and Safety Act of 1968.

Serial Number Identification Label



Output Power Danger Label

Package Aperture Labels



G1, J1 Package Diodes

B.2 Oclaro BMU7-808-02

This is the datasheet for the Oclaro BMU7-808-02 7 W laser and a test sheet from a characterization of the unit used in this setup.

7W 808nm Uncooled Multimode Laser Diode Module

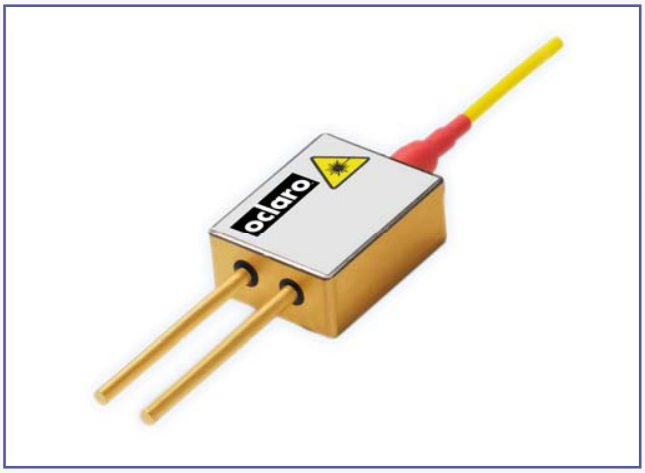
BMU7-808-02-R01/R02

Features:

- High output power of min. 7W
- 0.22NA 200µm core multimode optical fiber
- Hermetically sealed 2-pin package
- Floating anode/cathode
- High reliability
- Fiber protection sleeve and SMA connector
- RoHS compliant

Applications:

- Solid state laser pumping
- Medical
- Analytical
- Printing



The Oclaro BMU7-808-02-R01/R02 multimode laser diode module series has been designed to provide the highest power and reliability required for pumping solid-state lasers and for direct applications. The module includes a multimode laser diode chip with E2 front mirror passivation that prevents Catastrophic Optical Damage (COD) to the laser diode facet even at very high power levels. The coupling process allows for high output powers that are very stable with both time and temperature.

Operating Characteristics

Conditions unless otherwise stated:
Parameters at 25°C heat sink temperature and use of a thermal interface material rated for a thermal contact resistance of less than 1.3cm^2 K/W (0.2in^2 K/W). Optical fiber with 200µm core diameter and 0.22NA.

Parameter	Symbol	Typical	Unit
CW Output Power	P_{op}	min. 7	W
Center Wavelength BMU7-808-02-R01 BMU7-808-02-R02	 λ_{cR01} λ_{cR02}	 806 ± 3 803 ± 3	nm
Spectral Width (95% of Power)	$\Delta\lambda$	3	nm
Threshold Current	I_{th}	1.2	mA
Slope Efficiency	$\eta_D=P_{op}/(I_{op}-I_{th})$	0.97	W/A
Operating Current	I_{op}	8.5	A
Operating Voltage	V_{op}	2	V
Operating Temperature	T_{op}	25 ± 5	°C

Absolute Ratings

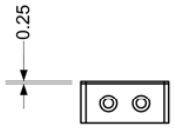
Parameter	Min	Max	Unit
ESD	-	500	V
Storage temperature	-40	85	°C
Lead soldering temperature	-	250	°C
Lead soldering time	-	10	Sec
Operating case temperature	15	60	°C
Relative humidity	5	85	%

Fiber Specification

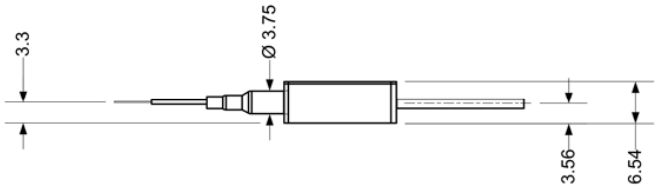
Parameter	Min	Typ	Max	Unit
Buffer diameter	305	315	335	µm
Cladding diameter	237	240	243	µm
Core diameter	196	200	204	µm
Numeric aperture	-	0.22	-	-
Fiber length	-	1.5	-	m
Fiber bend radius	25	-	-	mm

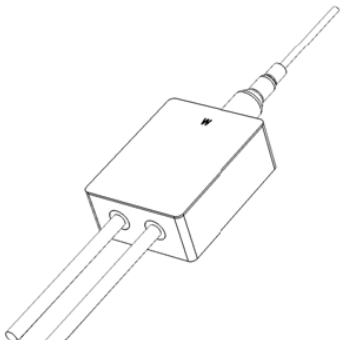
Package Dimensions (mm)

Back View (Pins Side):

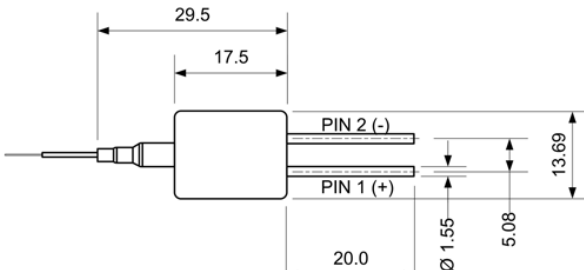


Side View:





Top View (Oclaro Label):



Remarks: - Mounting clip is available upon request
- Drawing does not show protection sleeve and SMA connector

RoHS Compliance

Oclaro is fully committed to environment protection and sustainable development and has set in place a comprehensive program for removing polluting and hazardous substances from all of its products. The relevant evidence of RoHS compliance is held as part of our controlled documentation for each of our compliant products. RoHS compliance parts are available to order, please refer to the ordering information section for further details.

Ordering Information

BMU7-808-02-R01	7W 806 ± 3 nm Multimode Laser Diode Module with 200µm 0.22NA fiber
BMU7-808-02-R02	7W 803 ± 3 nm Multimode Laser Diode Module with 200µm 0.22NA fiber

Contact Information

Oclaro Inc.
Worldwide Headquarters
2584 Junction Avenue
San Jose
CA 95134
USA

Tel: +1 408 383 1400
Fax: +1 408 919 1501

www.oclaro.com
APSEurope@oclaro.com

Important Notice

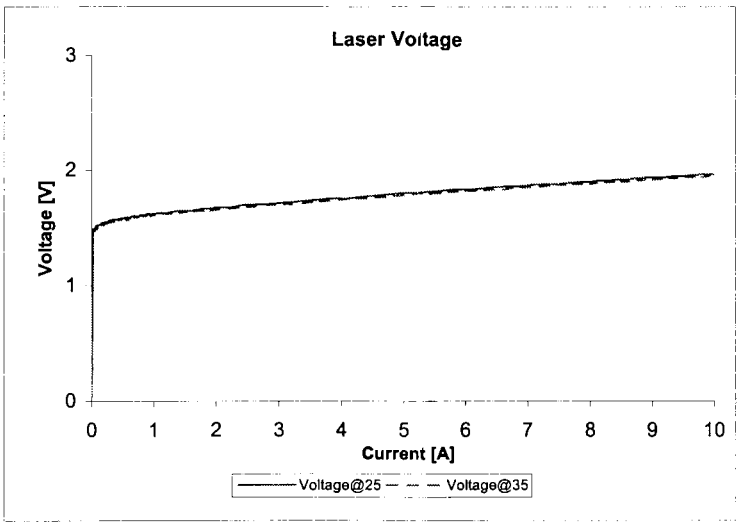
Performance figures, data and any illustrative material provided in this data sheet are typical and must be specifically confirmed in writing by Oclaro before they become applicable to any particular order or contract. In accordance with the Oclaro policy of continuous improvement specifications may change without notice. The publication of information in this data sheet does not imply freedom from patent or other protective rights of Oclaro or others. Further details are available from any Oclaro sales representative.

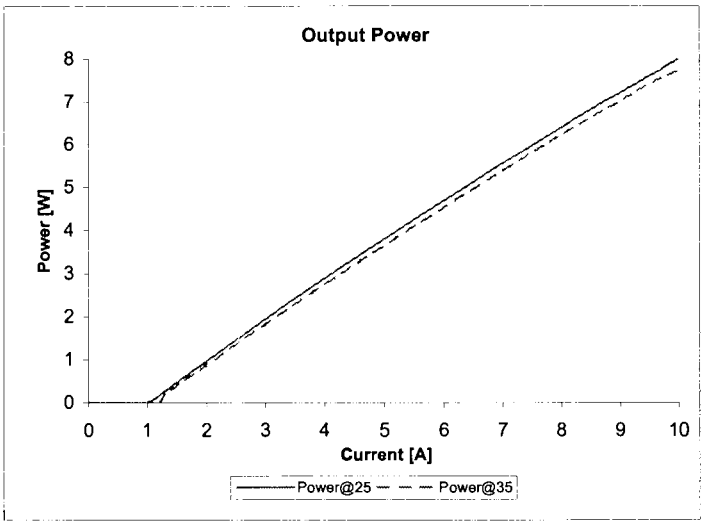
BMU6-808-200-02R Rev 1.1 August 2009

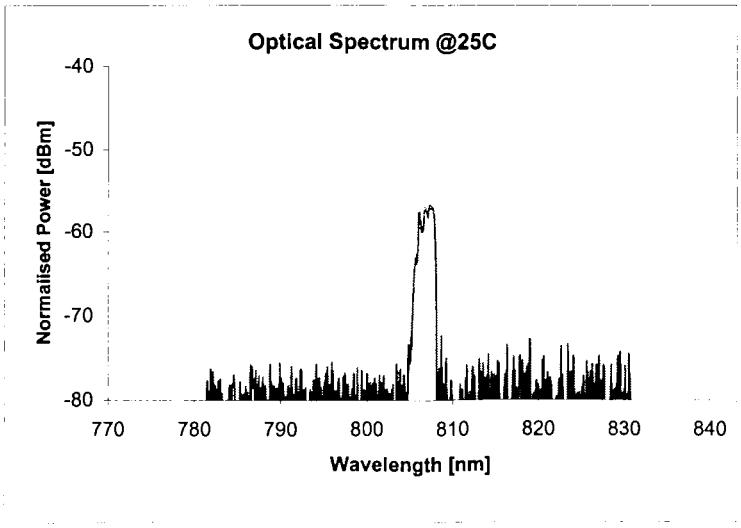
BMU7_808_02_R01

HL117477.001

Time and Date	20100613135316					
Serial Number	**HL117477.001**					
Chip ID						
Product Code	BMU7_808_02_R01					
Temperature	25	35				°C
Threshold Current	0.97	1.05				A
Rated Power	7	7				W
Operating Current @ 7.00W	8.7	9				A
Operating Voltage @ 7.00W	1.93	1.92				V
Center Wavelength @ 7.00W	806.93	809.91				nm
Spectral width (-13 dB) @ 7.00W	2.88	3.19				nm







Bibliography

[1] Charles H. Bennett, Franois Bessette, Gilles Brassard, Louis Salvail and John Smolin. Experimental quantum cryptography. *Journal of Cryptology*, 5:3–28, 1992. 10.1007/BF00191318.

[2] Charles H. Bennett and Gilles Brassard. Quantum cryptography : Public key distribution and coin tossing. *Proc. of IEEE Int. Conf. on Computers, Systems and Signal Processing, 1984*, 1984.

[3] S. Cova, M. Ghioni, A. Lacaita, C. Samori and F. Zappa. Avalanche photodiodes and quenching circuits for single-photon detection. *Appl. Opt.*, 35(12):1956–1976, Apr 1996.

[4] S. Cova, M. Ghioni, A. Lotito, I. Rech and F. Zappa. Evolution and prospects for single-photon avalanche diodes and quenching circuits. *Journal of Modern Optics*, 51:1267–1288, September 2004.

[5] Ilja Gerhardt, Qin Liu, Antia Lamas-Linares, Johannes Skaar, Christian Kurtsiefer and Vadim Makarov. Full-field implementation of a perfect eavesdropper on a quantum cryptography system. *Nat Commun*, 2:349–, June 2011.

[6] N. Gisin, S. Fasel, B. Kraus, H. Zbinden and G. Ribordy. Trojan-horse attacks on quantum-key-distribution systems. *Phys. Rev. A*, 73(2):022320, Feb 2006.

[7] Nicolas Gisin, Grégoire Ribordy, Wolfgang Tittel and Hugo Zbinden. Quantum cryptography. *Rev. Mod. Phys.*, 74(1):145–195, Mar 2002.

[8] Roland H. Haitz. Model for the electrical behavior of a microplasma. *Journal of Applied Physics*, 35(5):1370–1376, 1964.

[9] Lars Lydersen, Carlos Wiechers, Christoffer Wittmann, Dominique Elser, Johannes Skaar and Vadim Makarov. Hacking commercial quantum cryptography systems by tailored bright illumination. *Nat Photon*, 4(10):686–689, October 2010.

[10] D. Mayers. In N. Koblitz, editor, *Proceedings of Crypto 96, edited by N. Koblitz*, volume 1109, pages 343–357. Springer, New York, 1996.

[11] B. E. A. Saleh and M. C. Teich. *Fundamentals of photonics*, chapter 18.4, pages 767–775. John Wiley & Sons, Inc., 2nd edition, 2007.

[12] Sebastien Sauge, Lars Lydersen, Andrey Anisimov, Johannes Skaar and Vadim Makarov. Controlling an actively-quenched single photon detector with bright light, 2008, arXiv:0809.3408.

[13] Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Review*, 41(2):pp. 303–332, 1999.

[14] Michael R Sweet. *Serial Programming Guide for POSIX Operating Systems*. Michael R Sweet, 2010.

[15] Wolfgang Tittel, Grgoire Ribordy and Nicolas Gisin. Quantum cryptography. *Physics World*, pages 41–46, March 1998.

[16] Artem Vakhitov, Vadim Makarov and Dag R. Hjelme. Large pulse attack as a method of conventional optical eavesdropping in quantum cryptography. *Journal of Modern Optics*, 48(13):2023–2038, 2001.

[17] G S Vernam. Cipher printing telegraph systems for secret wire and radio telegraphic communications. *J Amer Inst Elect Eng*, 45(55):109–115, 1926.