Quantum Random Number Generator

In search of true randomness

EE3L11: Hardware, software and PCB design

R. Boshuisen
A. Kasem
R. Sobhi

Delft University of Technology

Quantum Random Number Generator

In search of true randomness

by

Student Name	Student Number
Name	Number
R. Boshuisen	4721381
A. Kasem	4158423
R. Sobhi	4094220

Instructor:	R. Ishihara
Teaching Assistant:	I. Varveris, S. YU, S. Nur
Institution:	Delft University of Technology
Place:	Faculty of Electrical Engineering, Delft
Project Duration:	04, 2021 - 07, 2021

Cover Image: QRNG + https://www.livescience.com/63770-quantum-randomness-beacon-created.html

Abstract

The purpose of this project is to develop a small mobile prototype of a Quantum Random Number Generator (QRNG). Using the intrinsically random quantum effect of photon, random numbers with true randomness are generated, which will then be used to generate a One-Time Password (OTP). The system is designed using only off the-shelf components, such that the QRNG is affordable for a new user group to whom existing QRNGs are too expensive and complex to operate. The system's schematic can be seen in Figure 1. The laser is driven by the laser driver to generate a light beam. This light beam is focused on one point through the aspheric lens and the polymer polarizer creates left handed circularly polarized light. Then at the beam splitter it will be split into a horizontal and a vertical component, which are then measured by two different detectors. Depending on which detector measures a higher voltage, a 1 or a 0 is returned. With this, a random bit stream is created and used as the seed for the OTP generator software, run on the Microcontroller (MCU). This OTP is then returned to the computer, ready for use and valid for 30 seconds.

Firstly, a programme of design requirements is set up. These requirements are based on desired end product features and limitations set by the project proposal, such as size, type of RNG and components that can be used. Next, the following project subsystems are completed and tested:

- **The power supply, laser driver circuit and main circuit have been designed, built and tested**. Nucleo F303k8 development board is used, from which the OTP generator software is run on the MCU. It has 64 kB of storage, to comply with the minimum requirement of 20 kB for the OTP software, and a clock speed of 72 kHz, where a minimum 50 kHz is needed. The power supply must deliver at least 320 mW of power to power the entire system and can deliver 1.5W of power through USB connection. The power supply can also deliver power from a battery instead, in case power delivery through USB is not possible. The laser, aspheric lens, polymer polarizer and beamsplitter must all work on the same, or at least very close, wavelength which was chosen to be 635 nm. The circuits are built and tested on a breadboard using Through-Hole (TH) components. The power supply through USB and by battery were both tested having a voltage deviation of less than 1%. The operating voltage of the laser is expected to be 2.7 V with an operating current of 55-85 mA. During testing, these were found to be 2.7 V and 65 mA, respectively. These system characteristics fall within expected and acceptable parameters.
- **The Printed Circuit Board (PCB) has been designed and tested**. A PCB was deemed necessary to keep the QRNG small and also to avoid noise found in cables and components. To achieve this, 3 separate PCBs were designed in EasyEDA and printed. Surface mount device (SMD) were used instead of TH-components, since SMDs have lower classical error tolerance. The PCBs have been tested for ground loops by measuring the voltage differences from the main print to the other prints. A minor imbalance of 0.05 V was found. Steady IC voltage inputs were ensured due to placement of pre-calculated 1 μF and 1 μF decoupling capacitors. The trace and via widths were set to 1 mm to ensure that the passing currents through the traces can be comprehended.
- **The OTP software has been written and implemented**. The application for the QRNG system is an OTP generator created in C++ using the HMAC-SHA-1 algorithm. For the secret key, the quantum random number from the flip-flop is stored to use as a serial to parallel converter implemented in the software. The seeds concatenated with a rolling counter that changes state every 30 seconds are put through the hashing function, which is then dynamically truncated down to a shorter hash. Then, at last, the software generates the decimal representation of the desired length of 6 digits. After compilation of the software, the memory usage was around 13% in ram and 33% in flash, well within the memory budget set for the thesis, having enough room for improvements and additional features of the micro-controller. The clock output for the flip-flop was a stable 100 kHz and the password generation takes place within 1 second, in order to keep the waiting period between consecutive code requests minimal.

The laser driver circuit gave a stable power output from the laser, the PCB design has been checked for stability with the use of the testpoints and software worked as expected, generating a new OTP every 30 seconds while also generating a new seed within the desired time. To verify the randomness of the QRNG, a statistical test (NIST test) was performed on the output of the detector circuit. This output consists of a string of zeros and ones generated by the QRNG. The statistical test resulted in the measured output not being random. This issue can be attributed to the addition of the sensor sub-circuit, where the sensor amplifier seemed to be clipping due to a too high amplification factor of 170dB. This problem could be solved by either lowering the sensor system's amplification, so as to stay within the operation range, or decreasing the laser's output even further from the set 50% maximum power.

Preface

We would like to thank Dr. R. Ishihara, S. Nur, I. Varveris and S. YU and for their continuous guidance during the thesis and project. We as a group have learned a great deal regarding QRNGs and the importance of their applications.

Group C1
Delft, August 2021

Contents

List of Figures

List of Tables

1 Introduction

1.1. Random Numbers

A random number is a number, where the outcome is completely unpredictable. In the case of a binary scenario, this would mean the probability to obtain a 1 or a 0 should be equal to $\frac{1}{2}$ with no correlation to past or future measurements. Randomness is used in many different applications like cryptography, simulation, gaming and lotteries. Conventional Random Number Generator (RNG) can be grouped in two categories: True Random Number Generator (TRNG) and Pseudo Random Number Generator (PRNG).

TRNGs use some physical process that is unpredictable or, at least, hard to predict such as sound card data, disk access times, thermal noise in electronic circuits and free running oscillators.

PRNGs usually generate a sequence of random numbers through a deterministic algorithm. Usually a small string of bits, called the seed, is used as the input for the deterministic algorithm to output a sequence of bits which seem uniformly distributed. However these conventional RNGs generally do not offer provable randomness. In some applications this provability is essential. “In applications where provability is essential, randomness sources (if involved) must also be provably random otherwise the whole chain of proofs will collapse.”[26] Quantum random number generators use a single intrinsically random quantum effect to generate random numbers. In this case the quantum effect of light (photon) is used as the carrier of qubit, but there are many other quantum effects that can be used to generate random numbers. The pros and cons for using these different methods for random number generation can be seen in Table 1.1.

Method	Pros	Cons
PRNG	Fast number generation rate	Predictable if seed can be determined or guessed
	Reproducible outcome	Correlation to previous values
TRNG	Unpredictable or difficult to predict	Limited generation rate
		Vulnerable to external attack
QRNG	Fast number generation rate	Sensitive to light and vibrations
	True randomness	

Table 1.1: Comparison of PRNG, TRNG and QRNG

1.2. Problem Definition

QRNGs already exist, however these are usually very expensive, large systems which are difficult to operate. There is also a lot of research on how to make QRNGs in different ways to generate quantum random numbers and create very high quality QRNGs, but these setups are usually bulky and expensive. These kinds of setups are not suited for private use or small businesses.

The goal of this project is to develop a prototype of a small mobile quantum random number generator (QRNG) based on a single qubit using photon and off-the-shelf products to find a practicable application for this QRNG. It did however quite early on in the project become clear that it would not be feasible to measure single photons, so instead the difference between the two detectors is measured. The application that is decided on is a one-time password (OTP) generator which pulls its secret (seed) from the QRNG to generate a password that is only valid for one login session. This OTP generator software resides in the MCU on a development board that is used. The entire system is shown in Figure 1.1. The laser driver makes sure the laser emits a smooth, stable and reliable output. The polarizer then only allows light of a certain polarization orientation and amplitude to pass, all other orientations of light are filtered out. This is important, because the randomness should be based on the quantum effect of the spin of photons and not the chaoticness of the beam of light emitted by the laser. At the beamsplitter, for each photon there is a 50% chance to be transmitted horizontally and a 50% chance to be reflected vertically, because the beamsplitter will split the spin-up photons from the spin-down photons.[20] These photons are measured at the detectors (photodiodes) and the signals of these detectors are compared. However since these are so small, they have to be amplified first and then depending on which of the signals was stronger, a 0 or a 1 is outputted at the comparator. The board is connected and powered by USB and also delivers power to the entire system. The OTP is generated by the Microcontroller (MCU) on this board and delivers this generated password to the PC.

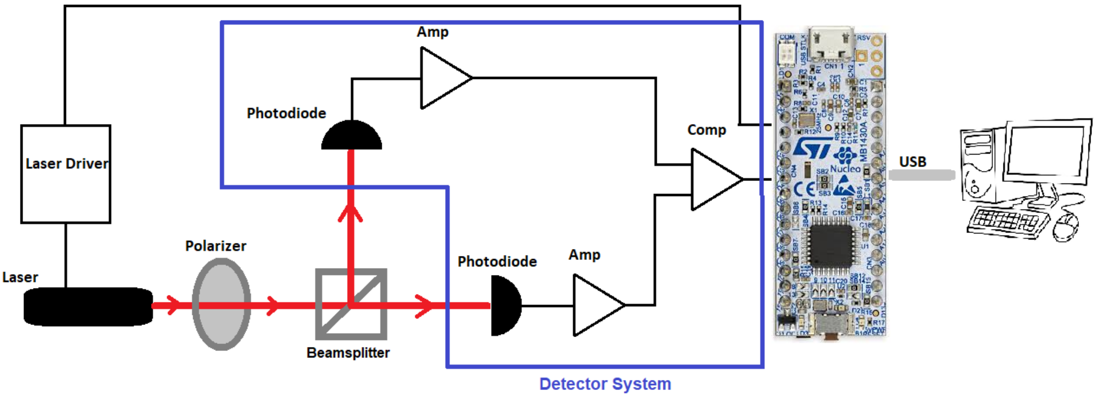


Figure 1.1: QRNG based OTP generator

1.3. System Subdivision

The group tasked to build this system is divided into two groups. This subgroup is tasked with the following:

- The hardware, which includes the laser driver circuit, the MCU and power supply.
- The PCB design, which includes the pcb design for the laser driver circuit and also the detector. Note that the circuit design of the detector part of the system is done by the other subgroup.
- The OTP generator software, where the generated random number by the QRNG is used to generate a one-time password.

The detector system as seen in Figure 1.1 will not be discussed as this task belongs to our sister group.

1.4. Thesis Outline

The main focus of this thesis is on the hardware, PCB design and OTP software as described in the system subdivision. In chapter 3, the requirements for the subsystems this group is tasked with are discussed. Chapter 4 will cover the Hardware, chapter 5 will cover the PCB design and chapter 6 will cover the OTP software. Chapter 4-6 will include the design, implementation, testing and results of their respective subsystems. In chapter 7 the statistical tests performed on the system will be discussed to see how well it performs as a RNG. And lastly, chapter 8 contains the conclusion and future work and recommendations are discussed.

2

Programme of Requirements

This chapter defines the backbone of the project, namely the requirements. These requirements can be sub-categorised in functional and non-functional requirements. The requirements are set for the physical dimensions of the system, the power supply, MCU, PCB and software.

2.1. Overall system requirements

These requirements apply to the overall system such as the size of the system, the type of components used and the way the random numbers are generated.

2.1.1. functional requirements

- Entire system should be mobile.
- Only off the shelf components are used for the system.
- Random numbers generated need to be based on Qubit using the quantum effect of photon spin.

2.1.2. non-functional requirements

- The physical size of the entire system needs to be no larger than 20 cm x 15 cm x 5 cm.

2.2. Power supply

The entire system should be mobile. This leads to the QRNG being able to run independent of external power sources unless the QRNG unless the device delivering the power is the same device the QRNG is used to generate a OTP for.

2.2.1. functional requirements

- The power supply circuit needs to power the entire system.
- The power supply needs to provide stable voltage rail outputs.
- The power supply circuit needs to be powered through a mobile unit.
- The power supply circuit needs to fit in its enclosure.

2.2.2. non-functional requirements

- The output power needed to power the entire circuit is 320 mW, so the power supply should be able to at least deliver this amount of power.
- 5 V & 2.5 V stable output voltage rails are needed to power all the components.
- The entire system needs to be powered through USB or a 9 V battery.
- The size of the PCB power supply circuit is 55 mm x 45 mm.

2.3. Micro controller

A MCU will be used to run the OTP software and connect to computers via USB. It should be fast enough to generate passwords at a decent rate, have enough storage and be able to drive the flip-flop as shown in the requirements below.

2.3.1. functional requirements

- The bitrate should be high enough for the delay to be no more than 1 second before a password is generated.
- The MCU should have enough storage space for the password generation code.
- The development board should have a USB interface so the system can connect to computers and other devices which can make use of the one-time password. The system can also be powered by USB.
- The development board should fit in the enclosure of the system together with the other parts.

2.3.2. non-functional requirements

- The user of the QRNG should not have to wait more than 1 second for a random key to be generated. The longest key a user is expected to generate is 256 characters. Each character consists of 8 bit which means a total of 2048 bits for the longest key. This would mean the minimum bit rate is 2 kB/s. This translates to a clockspeed of 2 kHz for the MCU.
- Since 5 V & 2.5 V voltage rails are used to power all components, the development board should run on this voltage.
- The MCU should have a storage of at least 20 kB as this is the expected size for all the code that has to be stored on the MCU.
- The MCU should be programmable by PC.

2.4. Laser driver and optical parts

The laser is driven by a laser driver. Since lasers are quite sensitive, the laser driver should include some circuit protections. The laser output should be kept stable in order to reduce the variations at the detectors due to laser instability. The optical parts should have a negligible impact on the laser beam power whilst still fulfilling their purpose.

2.4.1. functional requirements

- The laser should have a stable output. By stable it is meant that the the lasing is continuous and the lasing wavelength has a maximum deviation of $\pm 1\%$.
- The laser driver should be able to support a wide operation range for driving voltages and currents, for compatibility and re-usability.
- Circuit protections for the laser system.
- The aspheric lens' influence on the laser beam power should be negligible.
- The beamsplitter's influence on the laser beam power should be negligible.

2.4.2. non-functional requirements

- Over-current protection, electrostatic discharge protection and a soft-start circuit are the minimum requirements regarding circuit protections for the laser system. This will extend the lifetime of the laser significantly.
- The aspheric lens should maintain a high transmission rate (> 99%), and low reflectance (< 1%) for high efficiency of the system
- The beamsplitter should have a high transmission rate (> 95%).

2.5. PCB

The requirements for PCB are set to ensure that the PCBs can be correctly manufactured by the printing house and to ensure that the components physically fit onto the PCBs.

2.5.1. functional requirements

- Only necessary amount of layers to be used.
- Trace width needs to handle current going through trace.
- Creepage clearance needs to be defined that can handle voltages found in circuit.
- Physical size PCBs should fit in enclosure

2.5.2. non-functional requirements

- 4 layers: 1 layer for power signal, 1 layer for ground signal and 2 layers for signals.
- Max current in circuit is 55 mA. So the trace width of 1 mm will be implemented.
- Max voltages in circuit is 5 V. So a creepage distance of 1 mm will be implemented.
- The size of the laser and photo diode mount is 40 mm x 40 mm. So the size of the PCBs for the laser and photo diode circuits is 40 mm x 40 mm.

2.6. Software

The requirements set on the software have been chosen in such a way that the system is ensured to work properly, while also being extendable for future use.

2.6.1. functional requirements

- Should be within the memory limits mentioned in the hardware specs.
- Scaleable code design.
- Fast code without blocking functions.
- Secure transmission of the OTP.

2.6.2. non-functional requirements

- Code design is made in such a way the system can be adapted to the bit-length and MCU ports when required, while keeping the manual tracking of IO ports low.
- The code should be able to keep pulling the new binary value while also computing the new hash/OTP.
- Create a secure hash with the use of a non vulnerable hashing algorithm.

These requirements are taken into consideration throughout the thesis for the final system. The respective chapters will discuss these requirements in more detail.

3 Hardware

In this chapter the choice of MCU, the laser driver circuit and the power supply for the entire system will be discussed. Some light will be shed on the motivation behind the design choices, how the testing was done and the results will be displayed.
After properly designing and testing this circuit with through-hole components on a breadboard, the next step will be to to move to a PCB using SMD components. However that is to be discussed more in-depth in the next chapter.

3.1. Design choices

Firstly, the design choices will be stated and motivated. This will be done for the MCU, laser driver circuit and power supply.

3.1.1. Microcontroller unit

The following requirements were set for a MCU on a development board:

- **USB interface**, to connect with other devices that need the OTP such as computers and handheld devices.
- **Sufficient storage**, for the software design and future updates. The OTP software is estimated to be around 10 kB, when comparing to existing OTP implementations. Including the code from our sister group C2, the total storage needed is estimated to be about 20 kB.
- **Small in size**, small enough so the enclosure of the entire system is not bigger than 20 x 15 x 5 cm. The development board should be smaller than 60 mm x 40 mm when considering how much space is needed for the optical parts and the cables.
- **Sufficient clockspeed**, an initial minimum of 2 kHz was set while having a maximum of 1 second wait time for password generation. However, group C2 tasked with designing the detector system has increased the rate of bit generation to 50 kbit/s. This means that the MCU will receive a bit 50,000 times per second so a minimum clockspeed of 50 kHz is needed. This will increase the password generation by 25-fold. Since modern MCUs on development boards have minimum clockspeeds in the range of multiple MHz, this will not be an issue.
- **Deliver enough power**, the system will be powered by USB. USB can carry a maximum voltage of 5 V. Development boards work at different current values, so the maximum power differs per board. The MCU on these boards also consume a certain amount of power, so the maximum power delivered and the power consumed by the boards have to be compared. The power consumption for the other parts can be seen in Table 3.1. The power consumption of the laser is based an estimate of the average consumption of lasers which are viable for this system, this will be discussed more in depth in the next subsection. The power consumption of the resistors were initially estimated to be in the order of a few milli Watts considering the current through these resistors are in the low milli Amperes and the resistances are in the low kilo Ohms.

	Power consumption
Laser	10 mW
Resistors	10 mW
Detector circuit	250 mW
Total power consumption	270 mW

Table 3.1: Estimated power consumption by Circuit (without development board)

Then when choosing a board, it must be considered that the size is smaller than 60 mm x 40 mm and fulfills the power requirements:

$$Total\ power\ delivery > Total\ power\ consumed$$

$$5V * I > 270mW + MCU\ power\ consumption$$

The following boards were some of the top contenders:

	Power consumption	Dimensions	Memory
Arduino Pro Mini	16 mW	18 mm x 33 mm	32 kB
Arduino Nano	95 mW	18 mm x 45 mm	32 kB
STM32 Nucleo F303k8	50 mW	18.5 mm x 50 mm	64 kB

Table 3.2: Comparison of development boards

Comparing these specifications, the STM32 Nucleo F303k8 was chosen because of the middle ground in power consumption whilst still yielding more memory and fulfilling the size requirements. Also worth noting is that it is about half the price of the other development boards. Now that the choice for a development board is made, the system total power consumption can be revisited:

	Power consumption
Laser	10 mW
Resistors	10 mW
Detector circuit	250 mW
STM32 Nucleo F303k8	50 mW
Total power consumption	320 mW

Table 3.3: Estimated power consumption by Circuit (with development board)

Through USB the board can deliver a maximum power of 5 V $*$ 300 mA $= 1.5$ W. This covers the total power consumption of 320 mW by a large margin.

3.1.2. Laser driver circuit

The optical parts of the circuit are the most specialized parts. As such the rest of the parts were chosen keeping this in mind. When talking about the optical parts, the beamsplitter, polarizer, laser and aspheric lens are meant. First the architecture of the system and the function of these parts will be discussed. A schematic of these optical parts can be seen in Figure 3.1.

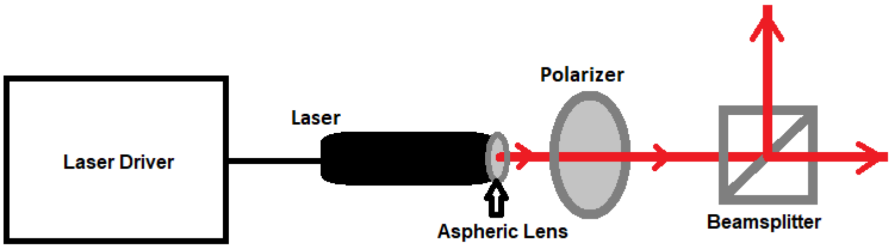


Figure 3.1: Schematic of optical parts used in the laser driver circuit

- **The laser**, this is the photon source. Some important specifications to take note here are the wavelength, optical output power and physical size of the laser. The optical parts only operate at a certain wavelength, so the wavelengths for all the optical parts must coincide. The laser must also be linearly controllable over a certain temperature range, because the laser becomes warmer as it stays turned on. The impact this has on lasing wavelength and output power must be minimized.
- **Aspheric lens**, which is mounted on the laser to focus the laser on one point. In order to catch the entire light beam from the laser with the detectors, an aspheric lens is mounted on the laser. A visual representation of the effect of an aspheric lens vs a regular lens can be seen in Figure 3.2.

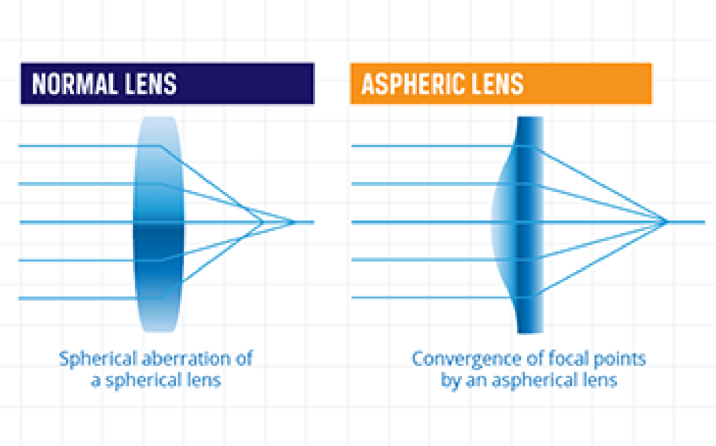

NORMAL LENS
ASPHERIC LENS
Spherical aberration of a spherical lens
Convergence of focal points by an aspherical lens

Figure 3.2: Normal lens vs aspheric lens[9]

- **Polymer polarizer**, this part is a type of optical filter which filters out all light waves that aren't of a certain type of polarization. Unpolarized light is chaotic and unpredictable, it has planewaves of different amplitudes and unpredictable phase differences. This will impact the values measured at the detectors in a unpredictable way. This is also a type of randomness but not the desirable randomness. As laid out in the requirements, the randomness should be based on the quantum effect of photon spin. See Figure 3.3 for reference what circular polarized light looks like. Note the 90° phase difference and equal amplitude of the planewaves, these are the key characteristics of circularly polarized light.

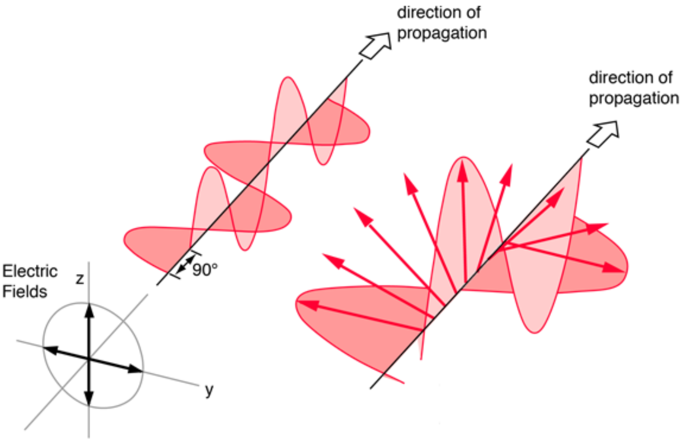


Figure 3.3: An example of right-handed circularly polarized light[4]

- **Beamsplitter**, this is where the photons emitted by the laser each have a 50% chance to be transmitted horizontally and 50% chance to be reflected vertically. Or in other words, it's a 50/50 chance for each photon to either move horizontally or vertically past the beamsplitter as seen in Figure 3.1. It's important to keep in mind that the beamsplitter also works on the same wavelength as the other optical parts.

These parts have been chosen such that the wavelength coincide or at least are very close for all parts as seen in Table 3.4.

Optical part	Wavelength
Laser	635 nm
Aspheric Lens	350-700 nm
Polymer Polarizer	633 nm
Beamsplitter	420-680 nm

Table 3.4: Optical parts wavelengths

The slight difference between the operating wavelength of the laser and polymer polarizer will have an influence on the transmitted light, however this will be very small. This deviation in wavelength scales linearly with the transmitted power[24]:

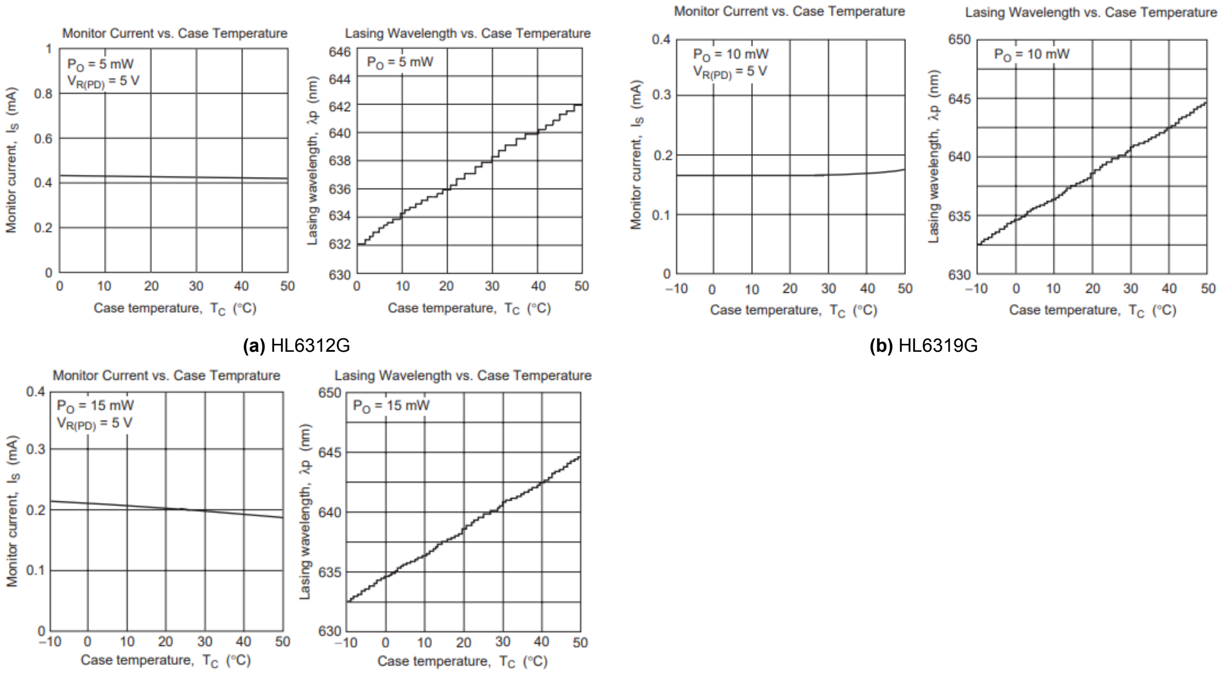
$$Adjusted\,Power = Power * \frac{Wavelengthdeviation}{PolarizerWavelength}$$

This is a power loss of 0.3%, which is quite negligible.
These optical parts also have to be mounted on a cage plate and be held together in a specific setup, since it is sensitive to deviations in the positioning. This limits the choice of parts since the cage plate and the parts come from the same manufacturer (Thorlabs):

- **Laser**, as seen in Table 3.5 the lasers all have a lasing wavelength of 635 nm. The optical output power and drive current do differ. Typically lower values for the optical output power and drive current would mean less power consumption and overall better for the system. Furthermore, when inspecting characteristic curves for the monitor current and lasing wavelength, the most efficient laser is the HL6312G as seen in Figure 3.4. The monitor current, which controls the optical output power of the laser diode, is the most stable for the HL6312G. The lasing wavelength also changes less with rising case temperatures, even if it is only slightly less than the other laser diodes. These factors led to the HL6312G being used as the laser diode in this system.

	Wavelength	Optical output power	Typical drive current
HL6312G	635 nm	5 mW	55 mA
HL6319G	635 nm	10 mW	70 mA
HL6321G	635 nm	15 mW	85 mA

Table 3.5: Comparison of laser wavelength, optical output power and drive current



(a) HL6312G

(b) HL6319G

(c) HL6321G

Figure 3.4: Comparison of laser Monitor current and Lasing wavelength vs. Case temperature[10][11][12]

- **Aspheric lens**, the C060TMD-A is the only mounted aspheric lens which fits the laser with a diameter of 9 mm at Thorlabs. With a light transmission of almost 100% and a reflectance of less than 0.25% as seen in Figure 3.5 this lens meets the requirements.

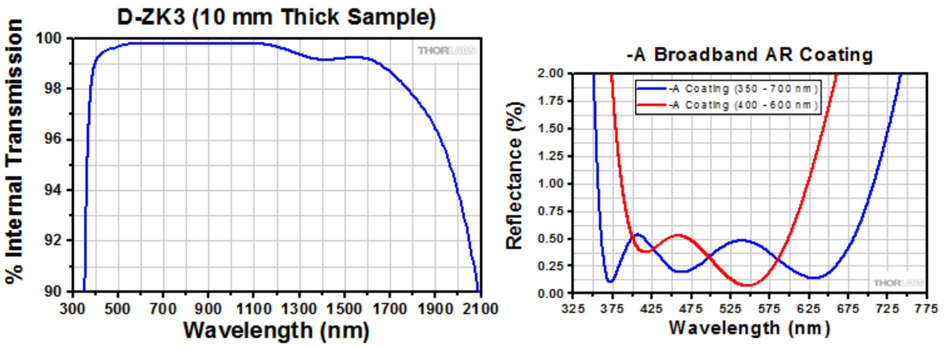
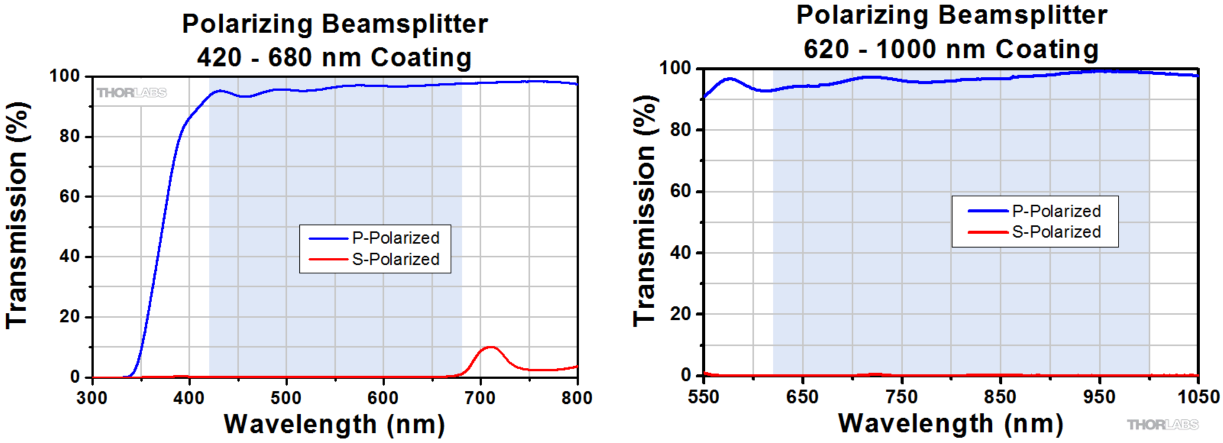


Figure 3.5: Light transmission and reflectance of aspheric lens C060TMD-A[22]

- **Polymer polarizer**, there are two circular polarizers that work at a wavelength of 633 nm. A left-handed and a right-handed circular polarizer, the difference is what direction the electric field seems to be rotating. As seen in the example of right-handed circular polarized light Figure 3.3, the electric field seems to be rotating clockwise when viewed from the light source. In this example the planewave in the z-direction had a $+90°$ phase difference from the planewave in the y-direction. If the planewave in the z-direction had a $-90°$ phase difference from planewave in the y-direction, then it would be left-handed circularly polarized. The direction of the circularly polarized light does not matter in this application and the other specifications of the polarizers are identical for the same wavelength. So either one is sufficient, but CP1L633 is chosen.
- **Beamsplitter**, there are 2 choices for the beamsplitter. They have the same specifications except for the transmission over their active wavelengths and the wavelength the beamsplitters work on. As seen in Figure 3.6 the transmission at 635 nm for the CCM1-PBS251/M is higher, so this

beamsplitter was chosen.



(a) CCM1-PBS251/M

(b) CCM1-PBS252/M

Figure 3.6: Transmission of polarized light by beamsplitters[2]

The laser must be able to deliver a stable and reliable output. For this purpose, a laser driver is used. The laser driver is chosen such that it fulfills these requirements while also maintaining the versatility to interchange the laser diode with minimal effort when needed:

- **Wide operation range**, this refers to the supply voltage and laser drive current. So the laser can easily be replaced by another model if necessary even if it works on a different drive current.
- **Circuit protections**, laser diodes can have lifetimes exceeding $100,000$ hours under ideal conditions, however they are very easily damaged. Usually this damage is caused by electro-static discharge, excessive current levels and spikes, or transients.[16] So the laser driver must at least include the following types of protection:
    - **electro-static discharge protection**, as this is one of the primary causes for laser damage.
    - **Overcurrent protection** for protection against overly high current levels and spikes.
    - **Slow-start circuit**, to protect the laser against turn-on transients by having an overdamped turn-on response.

Considering these characteristics and the HL6312G (laser diode) typical operating current of 55 mA and operating voltage of 2.7 V, there were 2 options for the laser driver. The comparison of these laser drivers iC-WKN and iC-WKL can be seen in Table 3.6.

	Supply Voltage	Laser Drive Current	Turn-on delay	Control error
iC-WKN	2.4-15 V	10-300 mA	70 us	0.3%
iC-WKL	2.4-6 V	5-70 mA	70 us	0.3%

Table 3.6: Comparison of laser drivers iC-WKN and iC-WKL[15][14]

The two laser drivers are very similar and they both meet the requirements. The decision was made to go with the iC-WKN because of its wider operation range, this allows for future adjustments if necessary. The optical output power of the laser is adjustable by adjusting the monitor current of the laser diode. The monitor current can be adjusted by changing the value of resistor R_{M} in Figure 3.7. It was decided to use the output power at half of the maximum as an initial starting point which is 2.5 mW. This way there is room for increasing the output power if this is necessary for the detector circuit. From the laser datasheet [10], it was found that a monitor current (I_{s}) of around 0.23 mA would result in an optical power output of 2.5 mW.

$$R_{\mathrm{M}} = \frac{V_{\mathrm{MDA}}}{I_{\mathrm{MD}}} = \frac{V_{\mathrm{cc}} - V_{\mathrm{R(LD)}}}{I_{\mathrm{s}}} \tag{3.1}$$

With $V_{\mathrm{cc}} = 5\ V$ and a voltage drop $V_{\mathrm{R(LD)}} = 2\ V$ this gives $R_{\mathrm{M}} \approx 13\ \mathrm{k}\Omega$.

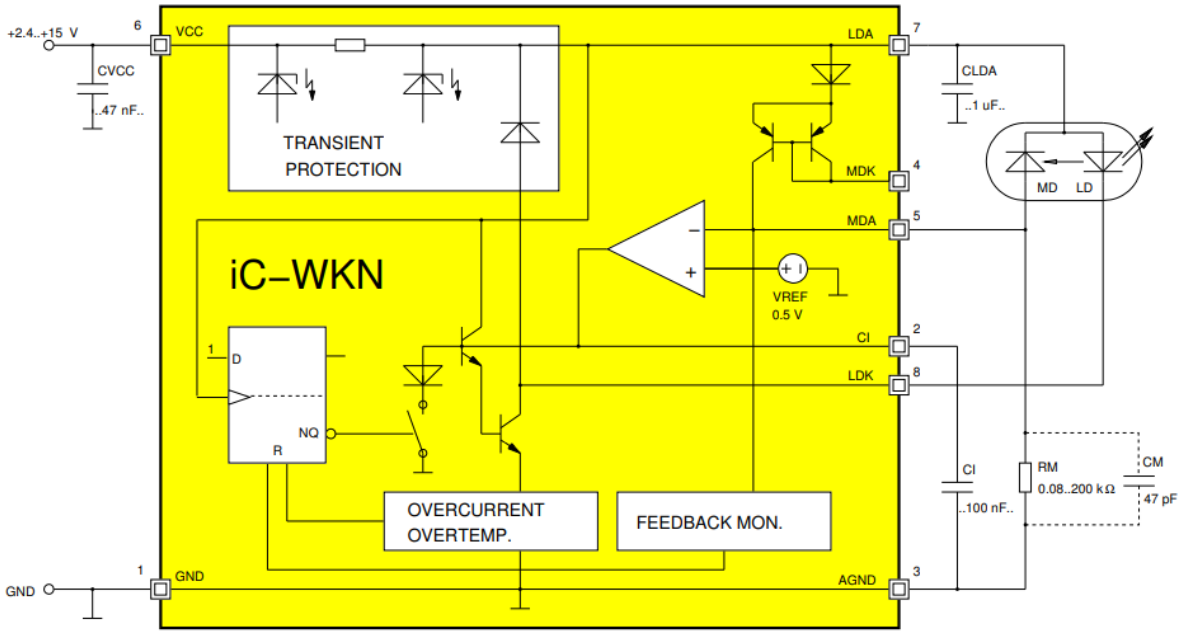


Figure 3.7: Laser driver schematic[15]

3.1.3. Power supply

When powering the system by USB through the development board, the maximum allowed current is $I_{max} = 300$ mA and the voltage over USB is $V_{USB} = 5$ V. Considering the power consumption of our current system as shown in Table 3.1, this will suffice. However, in case changes are made to the system or expansions are added or even in the case that it is not possible to deliver a stable 5 V by USB, there should be a system in place to accommodate this easily by using a power supply in the form of a battery. For this purpose a 9 V battery is used, however this system works on 5 V and 2.5 V across the board. So a voltage reference IC is used to deliver 5 V and 2.5 V.
Another way of supplying a stable voltage is by pulling a parallel line from the USB to the 5 V supply pin of the STM32 Nucleo.
Then a simple switch in the form of a pinheader is used to enable or disable 1 or both of the power supplies. So the choice can be made to have the power be supplied by USB or by a 9 V battery.

3.2. Circuit design

Now that the requirements have been set and the parts have been chosen, a circuit schematic is made with the chosen parts implemented. As seen in Figure 3.8 the main schematic consists of the MCU, the 5 V and 2.5 V voltage regulators, and a header. The header is necessary to deliver power because the final design consists of boards for the main circuit, laser driver circuit and the detector system which must all be supplied with power, however this will be discussed more in depth in the next chapter. The comparator and flip-flop are part of the detector system designed by our sistergroup C2.

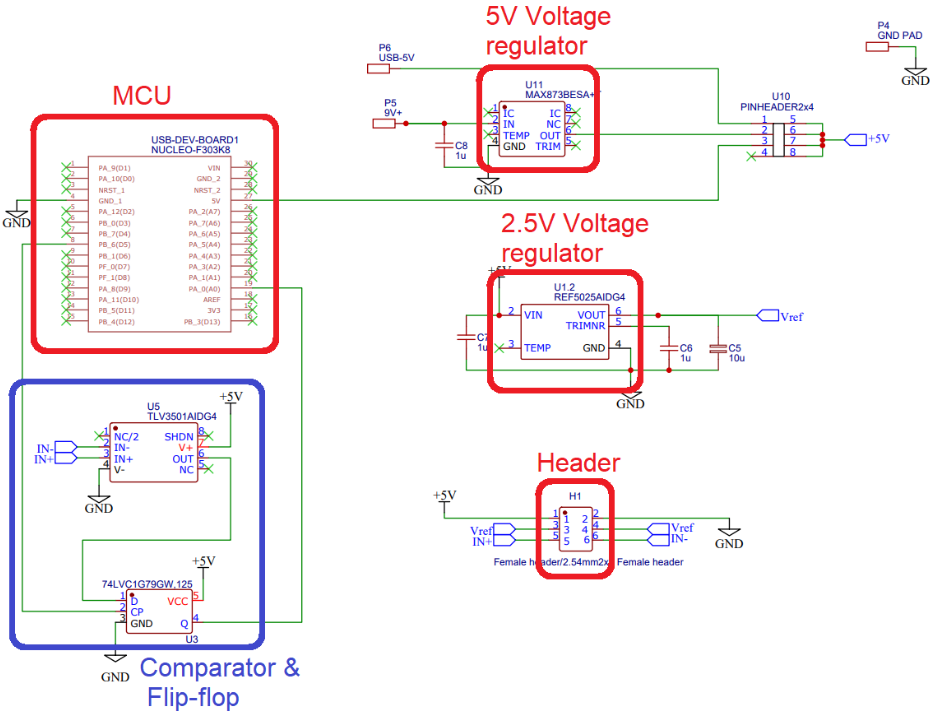


Figure 3.8: Main circuit with with external power capabilities using a switch

As seen in Figure 3.9 the laser driver circuit consists of the laser driver, the laser itself and the output power control. As stated earlier the output power of the laser can be controlled with the resistor R_M. There is also a header for this schematic because this is on a separate board that must be supplied with power.

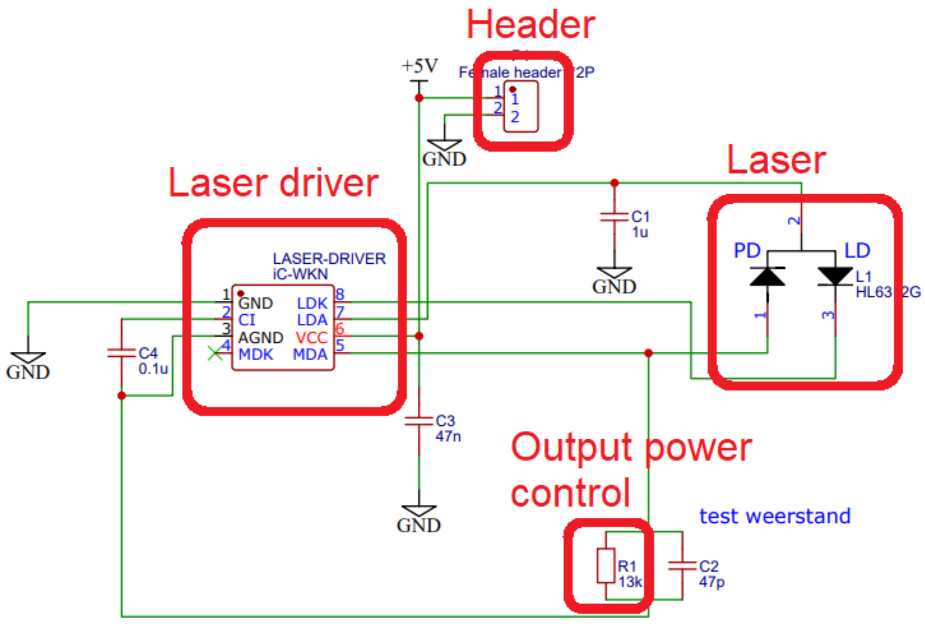


Figure 3.9: Laser driver circuit with power header

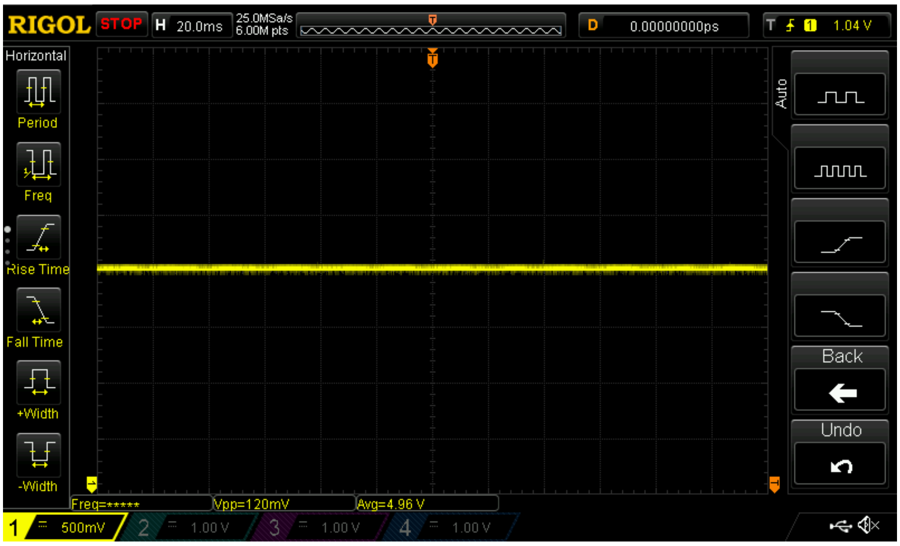
3.3. Testing

For the testing of this system, the main circuit and laser circuit were installed on breadboards using through-hole components. For the main circuit tests have to be done whether a stable 5 V can be delivered when using USB for power supply or when the 9 V battery is used.

For the laser driver circuit, it was measured what the operating voltage and current of the laser was to see if the laser was being driven properly. The operating voltage of the laser should be 2.7 V. The typical operating current of the laser is 55 mA with a maximum operating current of 85 mA. So the results are expected to be within these parameters.

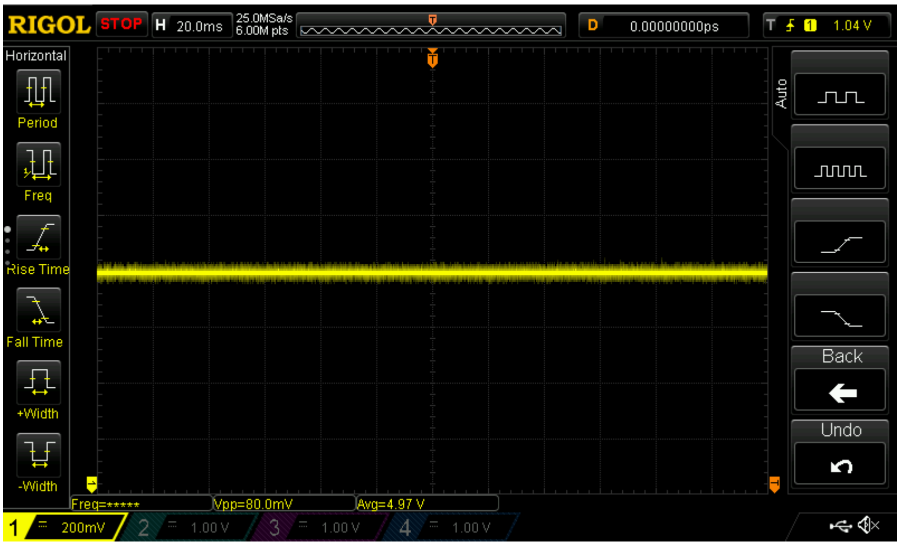
3.4. Results

In Figure 3.10 the voltage delivered by USB is seen. It is not a perfect 5 V however there is a deviation of less than 1%, which falls within acceptable terms. In Figure 3.11, the power is delivered by a 9 V battery. The voltage is measured after the voltage has been converted to 5 V by the voltage regulator. This is also not a perfect 5 V, however slightly closer to 5 V than when power is delivered by USB.



H 20.0ms 25.0MSa/s 6.00M pts
D 0.00000000ps
T 1.04 V
Freq=***** Vpp=120mV Avg=4.96 V
1 500mV 2 1.00 V 3 1.00 V 4 1.00 V

Figure 3.10: 5 V delivered by USB



H 20.0ms 25.0MSa/s 6.00M pts
D 0.00000000ps
T 1.04 V
Freq=***** Vpp=80.0mV Avg=4.97 V
1 200mV 2 1.00 V 3 1.00 V 4 1.00 V

Figure 3.11: 5 V delivered by 9 V battery after 5 V voltage regulator

The operating voltage measured at the laser is 2.7 V and the operating current measured at the laser is 65 mA as seen in Figure 3.12. The operating voltage is exactly 2.7 V as expected and the operating current falls withing the bounds of 55-85 mA. In Figure 3.13 the laser driver circuit can be seen on a breadboard and turned on while testing.

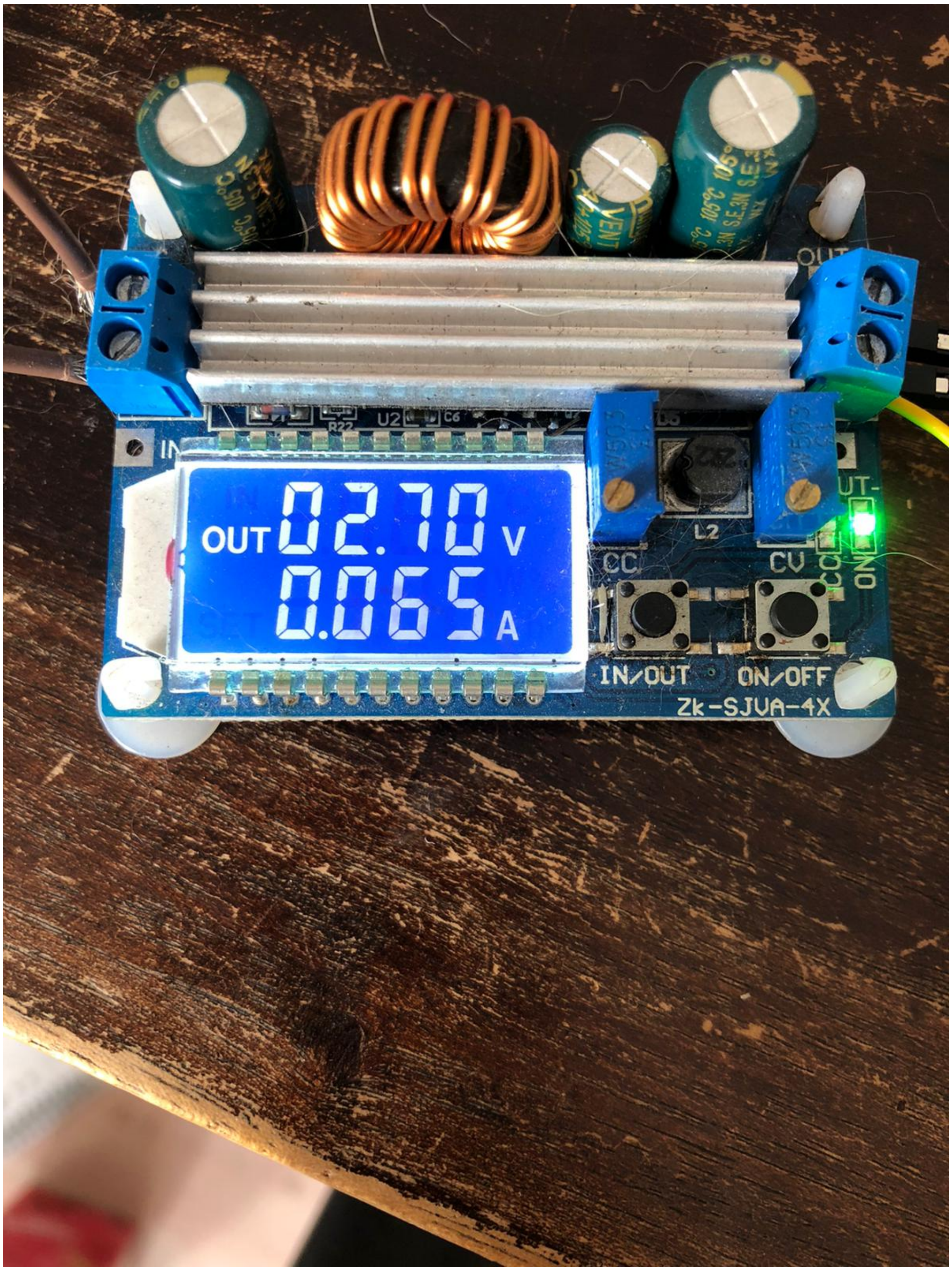


Figure 3.12: Operating voltage and current measured at the laser

3.5. Conclusion

The requirements for the MCU, laser driver circuit and power supply have been set. The parts for these systems have been chosen and they meet the requirements. Using these parts a main circuit consisting of MCU and the power supply has been designed and implemented on a breadboard for testing. The voltage deviation of power supply was less than 1 % from the desired voltage, this falls within acceptable parameters. A laser driver circuit has also been designed consisting of the laser driver, laser and resistor to control the optical power output of the laser. This circuit has also been implemented on a breadboard for testing. The laser driving voltage and current were measured and these were also within acceptable parameters. Optical parts such as the aspheric lens, polymer polarizer and beamsplitter have also been chosen. These parts have been chosen while keeping the efficiency of the system as high as possible by applying strict precision constraints such as high transmission rates (> 99%) and low reflectance rates(<1 %). This concludes the hardware design of the system. The next step is PCB design which will be discussed in the next chapter.

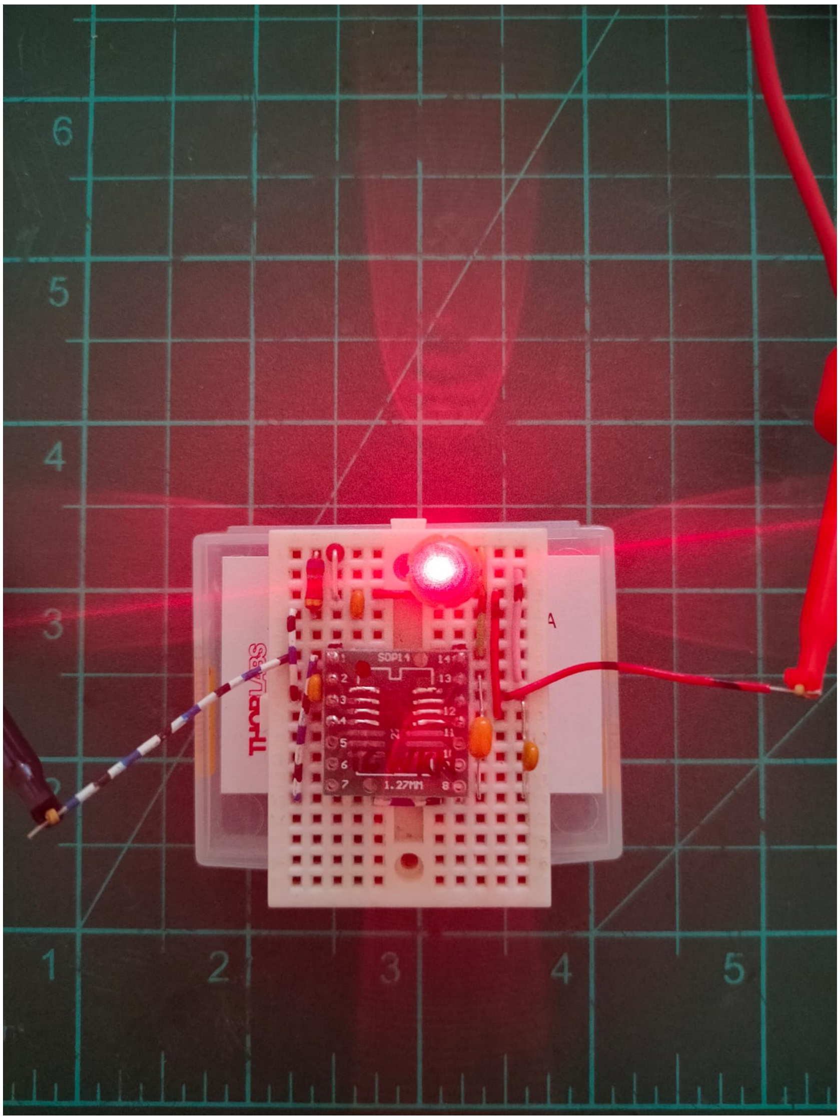


Figure 3.13: Laser circuit on a breadboard

4
PCB design

As one of the objectives was to create a mobile QRNG, it was deemed necessary to create a Printed circuit board (PCB). This to keep the prototype small but also to avoid noise found in cables and components. SMDs have low classical error tolerance compared to through hole components. So it was decided to fit all SMDs on the print.
To achieve the sub goal of designing a PCB, there had to be some literature study conducted on PCBs and component choices.

4.1. PCB design guide

To gather knowledge about PCB's. The book "Complete PCB Design Using OrCAD Capture and PCB Editor"[21] was consulted. All information found on this subject was gathered from this book. The design of the QRNG PCB does not include high speed signals so the focus was set on the following points:

- Board stack up and layers
- Routing and Via's
- Footprints and pads
- PCB Printing factory guidelines
- Ground loop

4.1.1. Board stack up and layers

A PCB is created by stacking layers on top of each other, some of these layers are conducting and some isolated. A standard PCB has the following layers in the given order:

- Silkscreen: This layer is needed to add text on the print to point out component values and names.
- Solder mask: This layer acts as a thin lacquer-like layer of polymer for protection against oxidation and to prevent solder bridges from forming between closely spaced solder pads
- Footprints and pads: Each component has a physical size to be found in the data sheet. With a footprint the physical dimensions of the components can be created and have it is space reserved on the PCB.
- Copper layers: These are the layers that have routes in them to conduct and connect the circuit.
- Isolated layers: These are layers that isolate between the conducting layers.
- Solder mask bottom: A solder mask but on the other side of the PCB
- Silkscreen bottom: A silkscreen but on the other side of the PCB.

Between each copper layer there is an isolating layer, to prevent conducting layers from causing a short circuit. A cross section of the layers consecutive to the first isolated layer is shown in Figure 4.1. It is of great importance to calculate how thick the conducting and isolating layers are. Since the designed QRNG is not of high speed we can use standardized thicknesses while keeping in mind what the printing limitations are of the PCB printing factory.

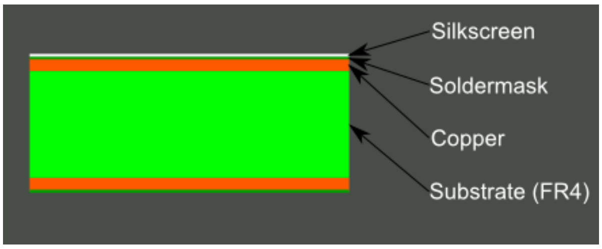


Figure 4.1: PCB stack up

4.1.2. Routing and Via's

To connect pins with one another a trace is created in the copper layer. it is advised to calculate how thick these traces should be. A trace resembles a wire and should be capable of handling the current crossing though it. For calculating the width of traces a standardized tool has been used [3]. It is often necessary to route via different conducting layers. For this a via is used. Via thickness should be taken into account, the via should be able to handle the current passing through it. For this the same tool that is used for trace width calculation is used.

4.1.3. Footprints and Pads

Footprints are a mapping of the physical component on the print with all its dimensions and pins. To do this correctly one must carefully read the mechanical part of data sheets. It is also helpful to use the same size of components as much as possible on a print. This reduces errors in footprint designs. footprints can be subdivided into two groups. Surface mount device (SMD) and Through hole (TH). SMD components have much lower error tolerances than TH components which automatically lowers the classical noise of the circuits.

Pads surround pins and are used for soldering the pins onto the print. Careful pad size choices are important. Once must keep enough space on the pad for solder to hold on to.

4.1.4. Ground loop

A ground loop is defined as the presence of a conducting path that is formed by multiple ground connections. This happens when ground does not have the same voltage potential on every part of the PCB.

Ground loops affect PCBs as a cause of noise interference. If one component on the loop is noisy then this noisy component is now a noise source for all other components on the loop.

The best way to minimize ground loops is though using a ground plane in stead of ground lines. It also helps to avoid splitting ground planes.[8]

Using all the knowledge from the theory gathered above it is good practice to design a PCB with the following summarized guideline rules:

- Ground plane boards are a good design choice because they allow signal routing through a via. It also significantly reduces the ground impedance and noise.
- Multiple ground planes are very advantageous, since they lower the board's ground impedance and reduce radiation in a common way;
- The power and ground planes must be rigorously coupled together;

4.2. Design targets and requirements

Targets and a deliverable were set and needed to be met for the PCBs. The deliverable is a physical PCB and the targets were derived from the PCB design guide.

The following targets had to be met:

- Choosing between placing all components of the system on a single or on multiple PCBs.
- Defining clearances on the board that agree with the PCB printing company.
- Minimize board and grounding noise through defining different ground planes for analog and digital components and avoiding ground loops.
- Placing capacitors to stabilize input voltages for ICs.

- deduce suitable trace width and via width.
- The footprint sizes should be large enough to be placed by hands duo to lack of machinery.
- deduce minimal amount of layers needed to fit all the components.

4.3. Designing

The PCB is designed using the design targets and requirements as an entry point.

4.3.1. CAD tool

The CAD design tool of choice is EasyEDA. This free CAD program can be accessed through multiple people at the same time via internet, keeping in mind that the group worked separately from home this seemed convenient. It is also is directly connected with JLCPCB: the PCB printing house.

4.3.2. Footprints

The Choice has been made to use a small number of different size footprints, this makes the possibility for footprint errors low. The chosen footprints are SOIC-8 for ICs and 0805 SMDs for passive components. These sizes are the smallest possible physical size components that can be soldered without machinery but by hand.

4.3.3. PCB layers and constraints

When looking at the circuit of the entire system the following data can be found: There is 2 power signals(5V and ground) and 26 different component signals. The minimal amount of layers needed to fit these signals is 4. So a 4 layer PCB was used. To meet the target of coupling the power planes and the advised guideline of using planes for power signals there will be 2 layers reserved. One for the 5V signal and the other for the ground signal. the remaining 26 component signals fit on 2 layers so 2 layers are reserved for that. 1 of these layers containing all the traces in horizontal directions while the other in vertical directions. The trace width can be calculated using the following formulas:

$$A = \frac{I}{k * T_r}$$

$$w = \frac{A}{T * 1.378}$$

Where A is the cross-section area of the trace in mm2, T_r the maximum accepted temperature rise and T is the trace thickness. One can also use the rule of thumb where minimum trace route width to apply is 1.0 mm/A. In this project we deemed the rule of thumb enough duo to the maximum current in the circuit being 55 mA. So the width of all traces was 1 mm. The via's diameter should match the trace width for the reason that both the trace as the connected via need to handle the same amount of current passing through them. So the via diameter is also 1.0 mm.

A rule of thumb is that 1 mm distance is kept per 1000 v difference between 2 voltage lines. This is the creepage distance. The PCB printing house can accommodate this demand. All other PCB house printing constraints can be found here [5] . All these values are used within the CAD software as DRC rules and therefor all constraints are meet, ensuring that the PCB can be printed.

4.3.4. Decoupling capacitors

Capacitors were used to stabilize the input voltage on ICs. The following formula was used to calculate the capacitor values for the ICs [6]:

$$C = \frac{I}{2\pi f V_i c}$$

Using this formula with regard to all implemented ICs, a $0.1uF$ decoupling capacitor was used for the laser driver and 1uF decoupling capacitors was used for the voltage regulators.

4.3.5. single vs multiple PCBs

The choice was initially made to design 1 single PCB. This configurations can be found in Appendix A The laser and photo diodes would be mounted on a stand and connected via cables to the board. This decision was purely based on PCB manufacturing price. 1 PCB is cheaper to print then multiple PCBs. While designing and testing the circuit of group C2 it was noticed that photo diodes are hyper sensitive to noises found in the connecting cables. To avoid this issue a set of 3 PCBs were designed where the laser and photo diodes are directly soldered on the PCBs directly and a main print including the power supply and the MCU. Furthermore using multiple PCBs makes testing and debugging in later stages easier as the system is now modular. The final design was the configuration with multiple PCBS. This configurations can be seen in the figures Figure 4.2, Figure 4.3 and Figure 4.4

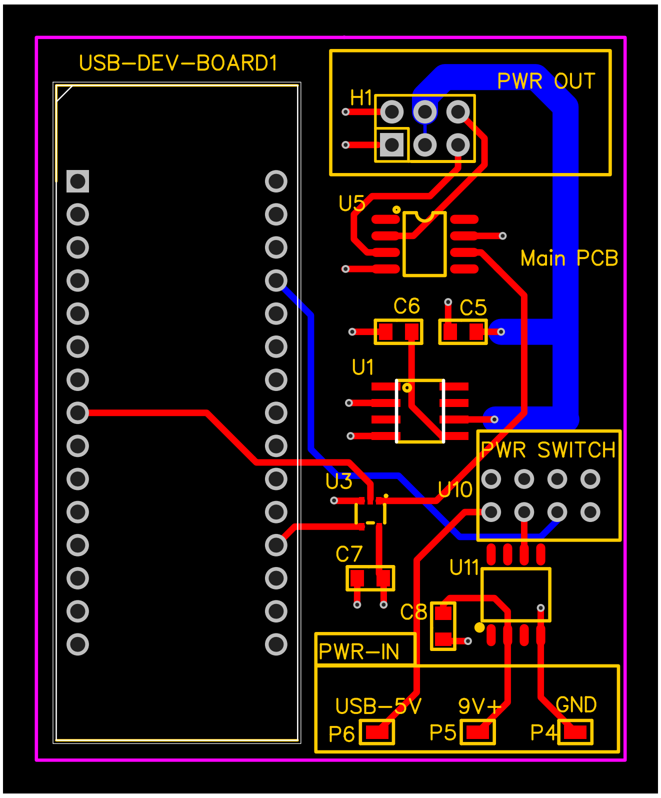


Figure 4.2: 4-layer main PCB design with 3 way external power capabilities

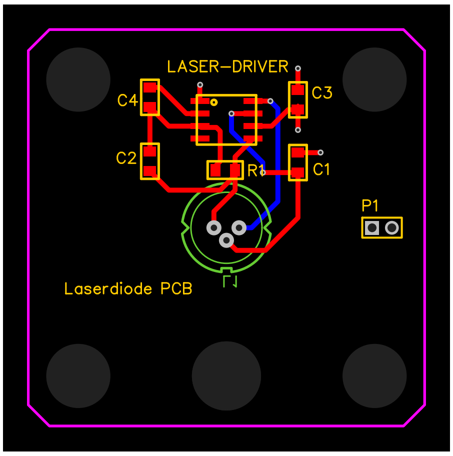


Figure 4.3: 4-layer laser PCB design in cage plate dimensions

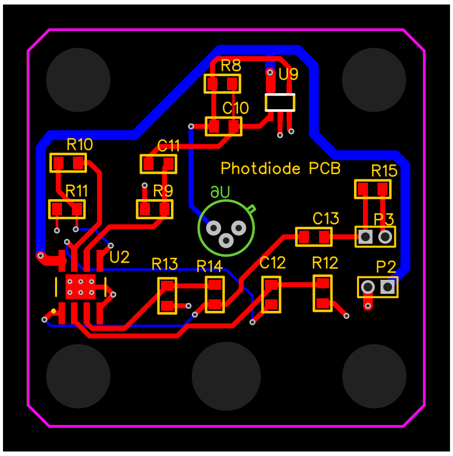


Figure 4.4: 4-layer sensor PCB design in cage plate dimensions

4.4. Testing

Testing the PCB meant finding out if the design requirements were meet. Within the scope of PCB it was necessary to test if all the desired voltages were to be found on the traces. Whether the circuit worked or not is not within the scope of PCB testing. The following tests were performed to validate if all PCB requirements were meet:

- Are there any ground loop issues on the boards: This is tested through measuring the current through the grounds.
- Do the ICs have a stable voltage input duo to the decoupling capacitors? Via an oscilloscope the input voltages of the ICs were measured to see what the ripple is.
- Are the desired voltages to be found on the traces? Via a multi meter the voltages of the main voltage rails on the PCB were measured.

PCB	vcc (V)	Vref (V)	main to daughterboard resistance (mΩ)
Main	4,936	2,499	-
Sensor 1	4,930	2,495	≤ 100
Sensor 2	4,925	2,499	≤ 100
Laser	4,932	-	≤ 100

Table 4.1: Voltage comparison between main and daughter-boards

In Table 4.1 the measurement analysis is shown from the PCB design. From this it can be seen that the use of multiple boards does create a minor imbalance in the voltage sources, hence a small resistance change in the boards. In the case of the *vcc* the small deviation is less of an issue as it only reduces the possible voltage swing, however for the reference voltage with the 170 dB amplifications, this can result in unwanted behaviour for the output at the flip-flop.

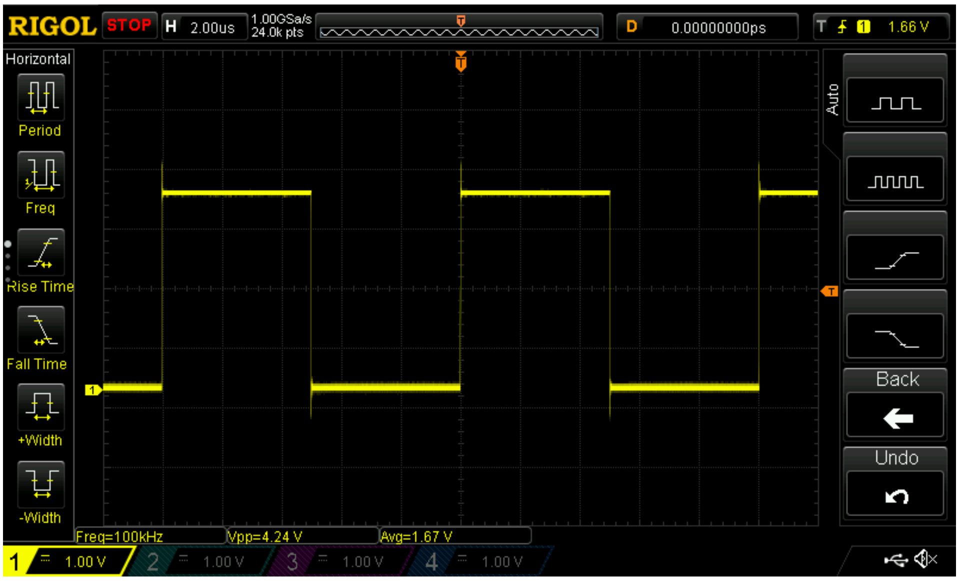


Figure 4.5: 100 KHz clock signal from the MCU, to the flip-flop

For the flip-flop a clock-rate was required, which was set on 100 KHz to achieve the desired bit-rate from the programme of requirements in Chapter 2, this has been verified with the aid of an oscilloscope while the system is operational as shown in Figure 4.5.

4.5. Results

The targets and requirements were met. The PCBs have clearances on the board that agree with the PCB printing company, trace and via widths have been calculated to fit the current going through them, board and grounding noise have been kept to a minimum through defining different ground planes for analog and digital components, ground loops are not present, decoupling capacitors with calculated values have been placed in front of the input pin of the ICs and these now have stable input voltages.

4.6. Conclusion

All the requirements and target were meet as seen in the subsection results and the PCBs were tested positive. The section "PCB design" can be seen as a success. Below in Figure 4.6 the 3 separate PCBs can be found.

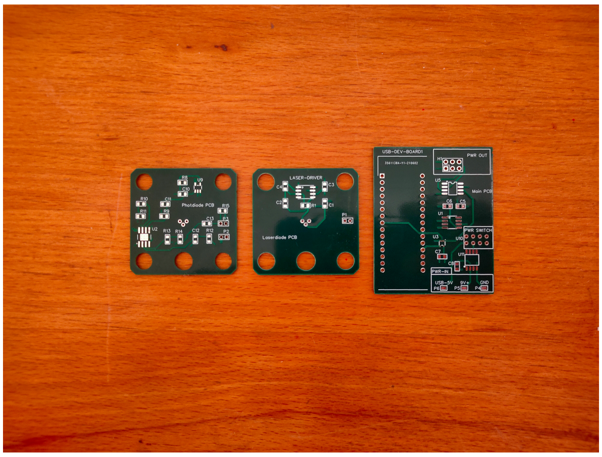


Figure 4.6: PCBs

5
OTP software

Now the hardware design is finished, the final step is the design the software to compliment the system. In the end the One-Time Password (OTP) algorithm has been chosen to show one potential use-case and a way to visualise whether or not the system and software behaves as expected. For the initial iteration the HMAC-Based One-Time Password (HOTP) was chosen due to it is simplicity and due to there not being a real time clock on the stm32, however this has been changed to the Time-Based One-Time Password (TOTP) for it dynamic changing output. For this the prototype has the client and server on the same chip to maintain the synced clock between them.

5.1. One-Time Password (OTP)

The One-Time Password, has two most commonly used algorithms, the previously mentioned HOTP and TOTP.

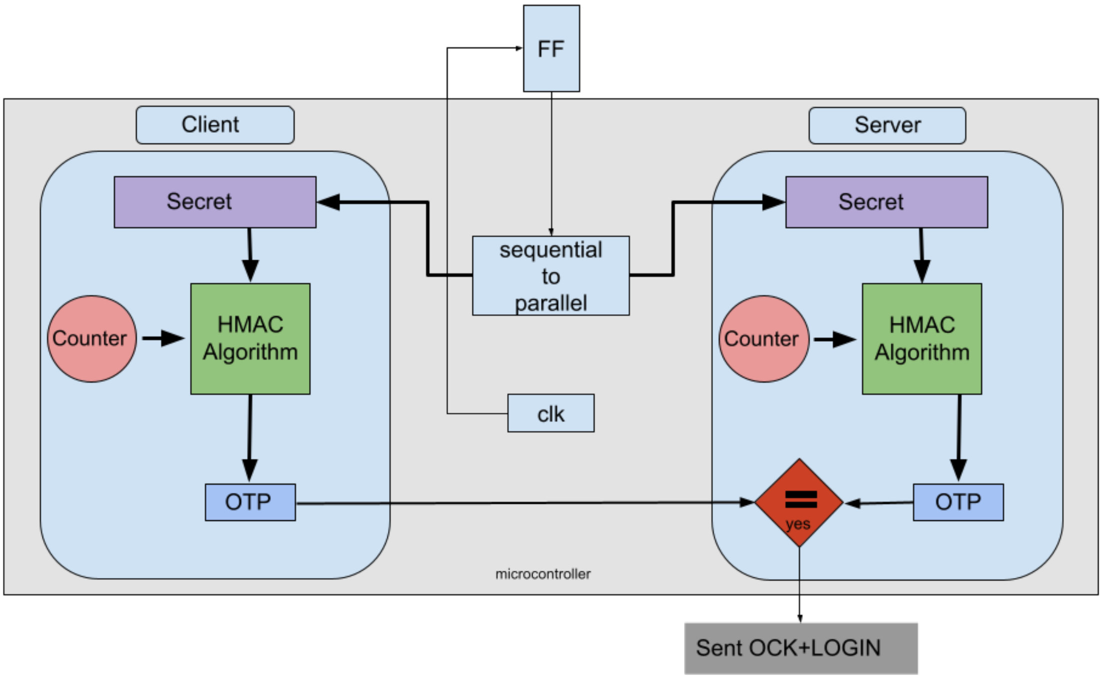


Figure 5.1: OTP process visualised

5.1.1. Design

As mentioned before the client and server are implemented on the same micro-controller as shown in Figure 5.1 for testing the prototype. The only external factor on the micro-controller is the flip-flop, which returns the quantum generated random number as input and receives a clock signal for changing the binary state. Internally the micro-controller will change the sequential binary value from the flip-flop to a parallel bit-stream for the secret key, also known as the seed.

In the current implementation the the client and server are done on the same hardware and do not require any external linkage, however after testing the server will be an external factor, requiring the USB serial interface to communicate with the external system. In the current setup as seen in the before mentioned image, the serial monitor is used to visualise the one time password, and seedphrase.

5.1.2. HMAC-Based One-Time Password (HOTP)

The original One-Time password algorithm, also known as the Event-based OTP is the HMAC-Based One-Time Password (HOTP). The moving factor, the rolling counter, in this case is event based, which means once the OTP is validated the counter is incremented once on the server. The secret and the counter are concatenated together to create a one way hash that can be validated on the server with the aid of the secret and server counter. How the hashing works will be explained further in Section 5.2.1.

5.1.3. Time-Based One-Time Password (TOTP)

The newer OTP algorithm is the Time-based OTP (TOTP), in this algorithm the rolling counter is changing state every 30 or 60 seconds. A consequence of this is that the OTP is only valid for a short time-frame, including a small breezing period where the old password is still valid when the counter already updated, this is then validated on the server with the current counter and accepted or denied. From this it can already be determined it is more secure against hacks as due to not being static as the previous algorithm.

5.1.4. HOTP vs TOTP

Both OTP methods offer a One-Time Password but the main difference is in the period the code is valid. The HOTP algorithm is valid until it is used, or when a a subsequent counter is used, this gives a large window of possible values to be accepted, and thus more prone to hacks. The reason of this is that the HOTP algorithm increments the counter each request, however it is not known if the code is actually used, thus the server accepts a large range above the current counter. The TOTP based algorithm is less prone for this as the counter is dynamically changed over time, reducing the validation window for the accepted codes. However due to the UNIX time being used, an out of sync clock could be the difference in a code being accepted or not, resulting in the re-sync being required for either the client or server.

The trade-off is whether more secure or less prone to syncing issues is the more important factor for the use case. In the case of the the product being designed for the project, security is most important, resulting in the TOTP algorithm being the chosen one.

5.2. OTP algorithm

From M'Raihi et al. [19], the standard of the OTP protocol is explained in details. With the use of this article it was found out the protocol consists of 3 main sections, namely the hash generation using the HMAC-SHA-1 algorithm[7], generate a 4 byte dynamic truncation string and at last use the truncated string to generate the One-Time Password with the desired numbers, usually between 6 and 9 numbers long.

5.2.1. Hashing

The first step is to create a hashed version of the secret key generated from the quantum generated bit-stream, reducing it is size to the desired outcome and embed the current counter in it. The algorithm derives two keys from the secret, namely the inner(I_key_pad) and outer(O_key_pad) keys. The first round, the `hash(i_key_pad || message)` produces a partial hash from the inner key and the message. The second round produces the final hash derived from the previously generated hash together with the outer key. The main reason for the multiple rounds is to counter length extension attacks, which is a way if padding the known working key to a multiple of the hash length, resulting in the hash check thinking it is the correct code while it is not. With the addition of the second round the message is not anymore concatenated to the hash and thus not vulnerable to the extension attack.

The implemented HMAC hash algorithm is shown below in it is C-style pseudocode:

```
function hmac is
    input:
        secret:     Bytes    // Array of bytes
        counter:    Bytes    // Array of bytes to be hashed
        hash:       Function // The hash function to use (e.g. SHA-1)
        blockSize:  Integer  // The block size of the hash function (e.g. 64 bytes)
        outputSize: Integer  // The output size of the hash function (e.g. 20 bytes)

    // Keys longer than blockSize are shortened by hashing them
    if (length(secret) > blockSize) then
        key ← hash(secret) // secret is outputSize bytes long

    // secrets shorter than blockSize are padded to blockSize by padding with zeros
    if (length(secret) < blockSize) then
        secret ← Pad(secret, blockSize) // Pad key with zeros blockSize bytes long

    o_key_pad ← secret xor [0x5c  blockSize]   // Outer padded key
    i_key_pad ← secret xor [0x36  blockSize]   // Inner padded key

    return  hash(o_key_pad || hash(i_key_pad || counter))
```

In the design, the HMAC-SHA-1 algorithm is used, which reduces the above code from `HMAC(secret,counter,hash,blockSize,outputSize)` to the more readable version used later on in the chapter as `HMAC(secret,counter)`, where the already populated hash, blocksize and outputSize have been assigned the desired inputs from the SHA-1 algorithm as: `HMAC(secret,counter,SHA_1(),64,20)`.

5.2.2. truncating

To reduce the byte size of the previously derived hash(`hmac_result`) from Section 5.2.1 even further, which is 20 bytes in the SHA-1 algorithm, the use of the following truncation code is used:

```
int offset   =  hmac_result[19] & 0xf ;
int bin_code = (hmac_result[offset]  & 0x7f) << 24
    | (hmac_result[offset+1] & 0xff) << 16
    | (hmac_result[offset+2] & 0xff) << 8
    | (hmac_result[offset+3] & 0xff) ;
```

The offset takes the lower 4 bits of the least significant bit (LSB) of the last byte array from the hash. Due to the binary &-operator this value will be between 0 and 15 depending on the hash, the offset is used to take the next 4 bytes in the array for the dynamic truncation of the hash. In short it takes the array `hash[10]...hash[13]`. This is true in case `hash[10]` most significant bit (MSB) is 0, otherwise the bitwise operator changes this back to zero. This has been done to counter the ambiguity whether the data is signed or unsigned.

5.2.3. key generation

The last step is changing the dynamic truncated hash to a One-Time Passwords, which could have any range, but mostly common between 6 and 9 as mentioned previously. With the use of the modulo operator on the hash, the remainder will be extracted from the truncated binary code, this remainder is then converted to an integer value and results in the final OTP code being found.

```
    HOTP = bin_code % 10^(OTP_length)
```

5.2.4. example

As an example, a certain hash has been created using the HMAC-SHA-1 algorithm as mentioned in Section 5.2.1, where the secret key and counter have been used, for this example the outcome of the hash function is:

0	8	16	24	32	40	48	56	64	72	80	88	96	104	112	120	128	136	144	152
1f	*86*	*98*	*69*	*0e*	*02*	*ca*	*16*	*61*	*85*	*50*	*ef*	*7f*	*19*	*da*	*8e*	*94*	*5b*	*55*	*5a*

Figure 5.2: 20 byte SHA-1 hash of random input

As can be seen the byte length of this string is 20 bytes, which is the standard length for the used algorithm as mentioned earlier.

Dynamic Truncation

From here the dynamic truncation algorithm has been applied, for this first the offset is obtained using the 19th byte, which is `0x5a`, applying the bitmask on the 2 bytes result in the output `0xa`. With the defined offset in decimal being 10, the substring can be found with the above mentioned bitmask in Section 5.2.2, in this case `hash[10] = 0x50`, which has the most significant bit already being zero, resulting in the final string `hash[10]...hash[13] = 0x50ef7f19`.

HOTP

Now that the truncated string is found the last step is generating the One-Time Password. In this example 6 digits has been used, as it is the most common amount and for sake of simplicity. After the use of the modulo operator as defined in Section 5.2.3, the final value being obtained is:
`HOTP = 0x50ef7f19 % 10^6 = 872921`.

5.3. Code

For the software design C++ has been used, not only for it being commonly used, but also due to its speed and extended libraries to be used in the design. For the prototype the use of an OTP library is used, the SimpleHOTP[18] library is a library designed for micro-controllers, with the HMAC-SHA-1 algorithm included. The main purpose of the design was to give a proof of concept of what could be an use-case and in addition a way to verify the behavior of the hardware/software design.

5.3.1. Main

The code consists of two main parts, namely the setup and loop function. One part to mention is the following line of code `DigitalOutputPin<D5> clk;`, the code can become prone to errors once a lot of digital reads and writes are required, to counter this template meta-programming has been used to make it a single object and no need to keep track of pin names. The full code of this can be found in Section C.2. Furthermore the full main can be found in Section C.1.

Global variables

```
DigitalPin<D5> clk;         //pin D5 = clk
DigitalPin<A0> sensor_in;   //pin A0 = in

Timer clock = Timer();

int counter = clock.get();
char secret[] = "secret code";
```

The global assigned variables after the library imports are the two digital pins used, one being the clk for the flip-flop and the other for the readout of the flip-flop state. As seen in Figure 5.1 the other subgroups design specification required a clk input to function like designed. Furthermore the clock is being initialized globally and the counter gets his initial state, which will be zero. The last part is the secret, the secret will be assigned later on in using the sensor inputs bit-stream, this will be the seed for the OTP algorithm. At last the OTP is being initialized with the first counter and the secret.

Setup

```
        clk.initialize(OUTPUT);
        sensor_in.initialize(INPUT);
        Serial.begin(9600);
```

In the setup the first step was initializing the pinmode of the clock to be an output and the sensor being the input repectively. Next the serial port has been initialized with a baudrate of 9600 to be able to read the OTP output. Like all other code the timer can be found at Section C.3.

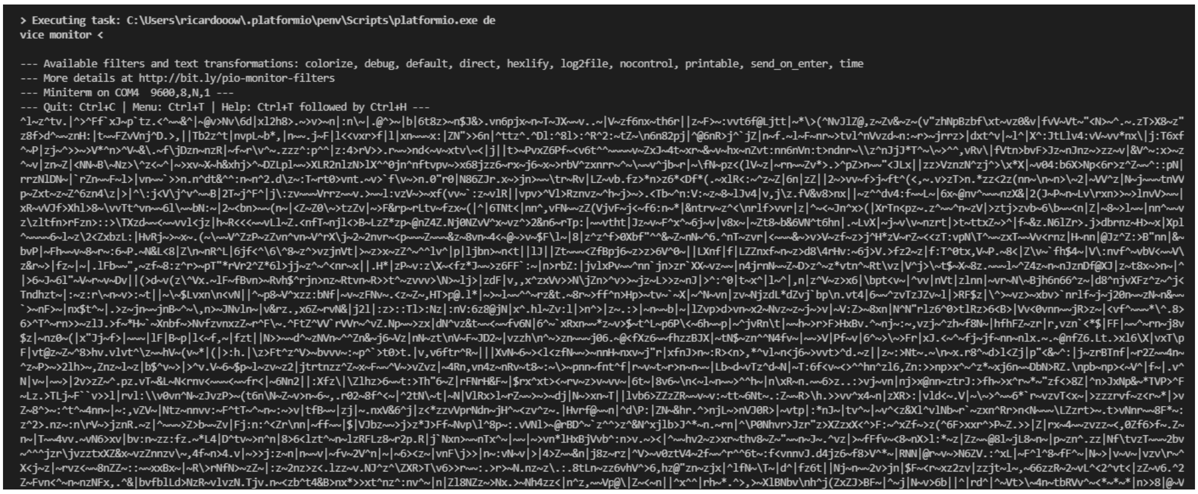
Loop

```
    clock.update()
        counter = clock.get();
        auto otp = hotp.generateHOTP(counter);
        Serial.println(otp);
```

In the main loop for each iteration the new counter is obtained from the timer class, this class generates a counter in the background with the use of `static_cast<int> ((millis() - start )/30000);`, this results in having a floored integer that only changes every 30 seconds. From there the new OTP is generated and send over the serial connection to the computer. This gives the possibility to check whether or not the system works as expected.

5.4. Results

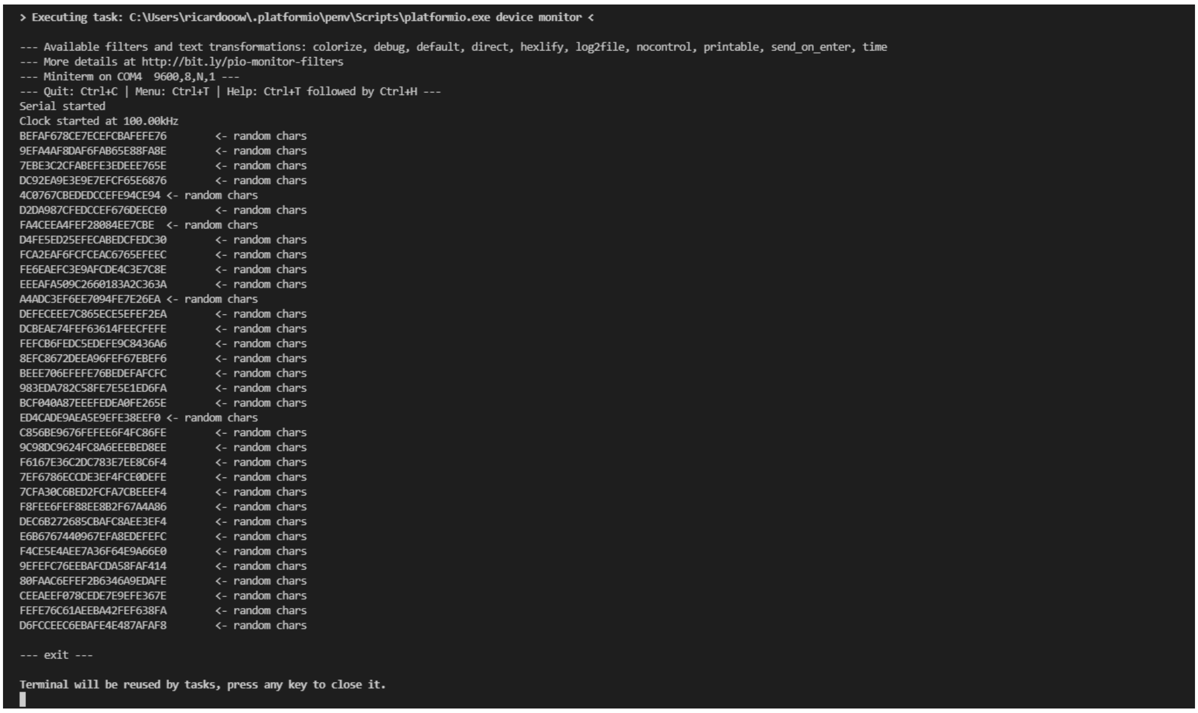
With the use of the code found in Appendix C the software and hardware have been tested in the digital domain.The first test conducted is using the serial line to monitor the bitstream from the random number generator. For this the binary values have been stored in an array and then displayed in ascii characters as shown in Figure 5.3, concluding that both the serial monitor, software and hardware are working as expected.



```
> Executing task: C:\Users\ricardooow\.platformio\penv\Scripts\platformio.exe de
vice monitor <

--- Available filters and text transformations: colorize, debug, default, direct, hexlify, log2file, nocontrol, printable, send_on_enter, time
--- More details at http://bit.ly/pio-monitor-filters
--- Miniterm on COM4  9600,8,N,1 ---
--- Quit: Ctrl+C | Menu: Ctrl+T | Help: Ctrl+T followed by Ctrl+H ---
^l~z^tv.|^>^Ff`xJ~p`tz.<^~~&^|~@v>Nv\6d|xl2h8>.~>v>~n|:n\~|.@^>~|b|6t8z>~n$J&>.vn6pjx~n~T~JX~~v..~|V~zf6nx~th6r||z~F>~:vvt6f@Ljtt|~*\>(^NvJlZ@,z~Zv&~z~(v"zhNpBzbf\xt~vz0&v|fvV~Vt~"<N>~^.~.zT>X8~z"
z8f>d^~~znH:|t~~FZvVnj^D.>,||Tb2z^t|nvpL~b*,|n~~.j~F|l<<vxr>f|l|xn~~~x:|ZN">>6n|^ttz^.^Dl:^8l>:^R^2:~tZ~\n6n82pj|^@6nR>j^`jZ|n~f.~l~F~nr~>tvl^nVvzd~n:~r>~jrrz>|dxt^v|~l^|X^:JtLlv4:vV~vv*nx\|j:T6xf
^~P|zj~^>>~>V*^n>^V~&\.~f\jDzn~nzR|~f~r\v^~.zzz^:p^^|z:4>rV>>.r~~>nd<~v~xtv\~<|j||t>~PvxZ6Pf~<v6t^^~~~~v~ZxJ~4t~xr~&~v~hx~nZvt:nn6nVn:t>ndnr~\\z^nJjJ*T^~\~>^^,vRv\|fVtn>bvF>Jz~nJnz~>zz~v|&V^~:x>~z
^~v|zn~Z|<NN~B\~Nz>\^z<~^|~>xv~X~h&xhj>^~DZLpl~~>XLR2nlzN>lX^^0jn^nftvpv~>x68jzz6~rx~j6~x~>rbV^zxnrr~^~\~~v^jb~r|~\fN~pz<(lV~z|~rn~~Zv*>.>^pZ>n~~"<JLx||zz>VznzN^zj^>\x*X|~v04:b6X>Np<6r>z^Z~~^::pN|
rrzNlDN~|`rZn~~F~l>|vn~~`>>n.n^dt&^^:n~n^2.d\z~:T~rt0>vnt.~v>`f\v~>n.0"r0|N86ZJr.x~>jn>~~\tr~Rv|LZ~vb.fz>*n>z6*<Df*(.~xlR<:~^z~Z|6n|zZ||2~>vv~f>j~ft^(<,~.v>zT>n.*zz<2z(nn~\n~n>\~2|~VV^z|N~j~~~tnVV
p~Zxt~z~Z^6zn4\z|>|^\:j<V\j^v^~~B|2T~j^F^|j\:zv~~~Vrrz~~v.>~~l:vzV~>~xf(vv~`:z~vlR||vpv>^Vl>Rznvz~^h~j>~>.<Tb~^n:V:~z~8~lJv4|v,j\z.fV&v8>nx||~z^^dv4:f~~L~|6x~@nv^~~~nzX&|2(J~P~n~Lv\rxn>>~>lnvV>~~|
xR~vVJf>Xhl>8~\vvTt^vn~~6l\~~bN:~|2~<bn>~~(n~|<Z~Z0\~>tzZv|~>F&rp~rLtv~fzx~(|^|6TNt<|nn^,vFN~~zZ(VjvF~j<~f6:n~*|&ntrv~z^<\nrlf>vvr|z|^~<~Jn^x>(|XrTn<pz~.z^~~^n~zV|>ztj>zvb~6\b~~<n|Z|~8~>l~~|nn^~~v
z\zltfn>rFzn>::>\TXzd~~<~~vvl<jz|h~R<<<~~vLl~Z.<nfT~njl<>B~LzZ*zp~@nZ4Z.Nj0NZvV^x~vz^>2&n6~rTp:|~~vtht|Jz~v~F^x^~6j~v|v8x~|~Zt8~b&6VN^t6hn|.~LvX|~j~v\v~nzrt|>t~ttxZ~>^|f~&z.N6lZr>.j>dbrnz~H>~x|Xpl
^~~~~6~l~z\2<ZxbzL:|HvRj~>~x~.(~\~~V^ZzP~zZvn^vn~V^rX\j~2~2nvr~<p~~~Z~~~&z~8vn~4<~@~>v~$F\l~|8|z^z^f>0Xbf"^^&~Z~nN~^6.^nT~zvr|<~~~&~>v>V~zf~z>j^H*zV~rZ~<<zT:vpN\T^~~zxT~~Vv<rnz|H~nn|@Jz^Z:>B"nn|&~
bvP|~Fh~~v~8~r~:6~P.~N&L<8|Z\n~nR^L|6jf<^\6\^8~z^>vzjnVt|>~z>x~zZ^~^^lv^|p|ljbn>~n<t||lJ||Zt~~~<ZfBpj6~z>z>6V^0~||LXnf|f|LZZnxf~n~z>d8\4rHv:~6j>V.>fz2~z|f:T^0tx,V~P.~8<|Z\v~`fh$4~|V\:nvf^~vbV<~~V\
z&r~>|fz~|~|.lFb~~",~zf~8:z^r>~pT"*rVr2^Z*6l>jj~z^~^<nr~x||.H*|zP~v:z\X~<fz*J~~>z6FF`:~|n>rbZ:|jvlxPv~~^nn`jn>zr`XX~vz~~|n4jrnN~~Z~D>z^~z*vtn^~Rt\vz|V^j>\~t$~X~8z.~~~l~^Z4z~n~nJznDf@XJ|z~t8x~>n~|^
|>6~J~6l"~V~r~v~Dv||(>d~v(z\^Vx.~lF~fBvn>~Rvh$^rjn>nz~Rtvn~R>>t^~zvvv>\N>~lj>|zdF|v,,x^zxVv>>N\jZn>^v>>~jz~L>>z~nJ|>^:^0|t~x^|l~^|,n|z^V~z>x6|\bpt<v~|^vv|nVt|zlnn|~vr~N\~Bjh6n66^z~|d8^njvXFz^z~^j<
Tndhzt~|:~z:r\~n~v>:~t||~\~$Lvxn\n<vN||^~p8~V^xzz:bNf|~v~zFNv~.<z~Z~,HT>p@.l*|~>~l~~^^~rz&t.~8r~>ff^n>Hp>~tv~`~X|~^N~vn|zv~NjzdL*dZvj`bp\n.vt4|6~~^zvTzJZv~l|>RF$z|\^>~vz>~xbv>`nrlf~j~j20n~~zN~n&~~
`>~nF>~|nx$t^~|.>z~jn~~jnB~^~\,n>~JNvln~|v&rz.,x6Z~rvN&|j2l|:z>::Tl>:Nz|:nV:6z8@jN|x^.hl~Zv:l|>n^>|z~.:>|~n~~b|~|lZvp>d>vn~x2~Nvz~z~j~>v|~V:Z>~8xn|N^N"rlz6^0>tlRz>6<B>|Vv<0vnn~~jR>z~|<vf^~~~*\^.8>
6>^T^~rn>>~zlJ.>f~*H~`~Xnbf~>NvfzvnxzZ~r^F\~.^FtZ^VV`rVVr~^vZ.Np~~>zx|dN^vz&t~~<~~fv6N|6^~`xRxn~~*z~v>$~t^L~p6P\<~6h~~p|~^jvRn\t|~~h~>r>F>HxBv.^~nj~:~,vzj~^zh~f8N~|hfhFZ~zr|r,vzn`<*$|FF|~~^rn~j8v
$z|~nz0~(|x"Jj~f>|~~~|lF|B~p|l<~f,~|fzt||N>>~~d^~zNVn~^^Zn&~j6~Vz|nN~zt\nV~F~JD2~|vzzh\n^~>zn~~~j06.~@<fXz6~~fhzzBJX|~tN$~zn^^N4fv~|~~>V|Pf~v|6^~>\~>Fr|xJ.<~^~fj~jf~nn~nlx.~.~@nfZ6.Lt.>xl6\X|vxT\p
F|vt@z~Z~^8>hv.vlvt^\z~~hV~(v~*|(|>:h.|\z>Ft^z^V>~bvvv~:~p^`>t0>t.|v,v6ftr^R~|||XvN~6~><l<zfN~~>~nnH~nxv~j"r|xfnJ>n~:R><n>,*^vl~n<j6~>vvt>^d.~z||z~:>Nt~.~\n~x.r8^~d>l<Zj|p"<&~^:|j~zrBTnf|~r2Z~~4n~
^z~P>~>2lh>~,Znz~l~z|b$^v~>|>^v.V~6~$p~l~zv~z2|jtrtnzz^Z~x~F~~^V~>vZvz|~4Rn,vn4z~nRv~t8~:~\>~pnn~fnt^f|r~v~t~r>n~n~~|Lb~d~vTz^d~N|~T:6f<v~<>^^hn^zl6,Zn:>>np>x^~^z*~xj6n~~DbN>RZ.\npb~np><~V^|f~|.v^
N|v~|~~>|2v>zZ~^.pz.vT~&L~N<rnv<~~~<~~fr<|~6Nn2||:Xfz\|\Zlhz>6~~t:>Th"6~Z|rFNrH&F~|$rx^xt><~rv~z>v~vv~|6t~|8v6~\n<~l~n~~>^^h~|n\xR~n.~~6>z..:>vj~vn|nj>x@nn~ztrJ:>fh~>x^r~*~"zf<>8Z|^n>JxNp&~*TVP>^F
~Lz.>TLj~F``v>>l|rvl:\\v0vn^N~zJvzP>~(t6n\N~Z~v>n~6~,.r02~8f^<~|^2tN\~t|~N|VlRx>l~rZ~~>~>dj|N~>xn~T||lvb6>ZZzZR~~v~v:~tt~6Nt~.:Z~~R>\h.>>vv^x4~n|zXR>:|vld<~.V|~\~>^~~6*`r~vzvT<x~|>zzzrvf~z<r~*|>v
Z~8^>~:^t^~4nn~|~:,vZV~|Ntz~nnvv:~F^tT~^~n~:~>v|tfB~~|zj|~.nxV&6^j|z<*zzvVprNdn~jH^~<zv^z~.|Hvrf@~~n|^d\P:|ZN~&hr.^>njL~>nVJ0R>|~vtp|:*nJ~|tv^~|~v^<z&Xl^vlNb~r`~zxn^Rr>n<N~~~\LZzrt>~.t>vNnr~~8F*~:
z^2>.nz~:n\rV~>jznR.~z|^~~~>Z>b~~Zv|Fj:n:^<Zr\nn|~ff~~|$|VJbz~~>j>z*J>Ff~Nvp\l^8p~:.vVNl>~@rBD^~`z^^>z^&N^xjlb>J^*~n.~rn|^\P0Nhvr>Jzr"z>XZzxX<^>F:~^xZf~>z(^6F>xxr^>P~Z.>>|Z|rx~4~~zvzz~<,0Zf6>f~.Z~
n~|T~~4vv.~vN6>xv|bv:n~zz:fz.~*L4|D^tv~>n^n|8>6<lzt^~n~lzRFLz8~r2p.R|j`Nxn>~~nTx^~|~~|~>vn*lHxBjVvb^:n>v.~><|^~~hv2~z>xr~thv8~Z~"~~n~J~.^vz|>~fFfv~<8~nX>l:*~z|Zz~~@8l~jL8~n~|p~zn^.zz|Nf\tvzT~~~2bv
~^^^jzr\jvzztxXZ&x~vzZnnzv\~,4f~n>4.v|~>>j:z~n|n~~v|~fv~2V^n|~|~6><z~|vnF\j>>|n~:vN~v|>|4>Z~~&n|j8z~rz|^V>~v0ztV4~2f~~^r^^6t~:f<vnnvJ.d4jz6~f8>V^*~|RNN|@r~v~>N6ZV.:^xL|~F^l^8~fF^~|N~>|v~v~|vzv\r~^
X<j~z|~rvz<~~8nZZ~::~~xxBx~|~R\>rNfN>~zZ~|:z~2nz>z<.lzz~v.NJ^z^\ZXR>T\v6>>r~~:.>r>~N.nz~z\.:.8tLn~zz6vhV^>6,hz@"zn~zjx|^lfN~\T~|d^|fz6t||Nj~n~~2v>jn|$F~<r~xz2zv|zzjt~l~,~66zzR~2~vL^<2^vt<|zZ~v6.^2
Z~Fvn<^~n~nzNFx,.^&|bvfblLd>NzR~vlvzN.Tjv.n~<zb^t4&B>nx*>>xt^nz^:nv^~|n|Zl8NZz~>Nx.>~Nh4zz<|n^z,~~Vp@\|Z~<~n||^x^^|rh~*.^>,>~XlBNbv\nh^j(ZxZJ>BF~|^~j|N~v>6b||^|rd^|^~Vt>\~4n~tbRVv^~<*~*~*|n>>8|@~V
```

Figure 5.3: Random number stream for the secret generation

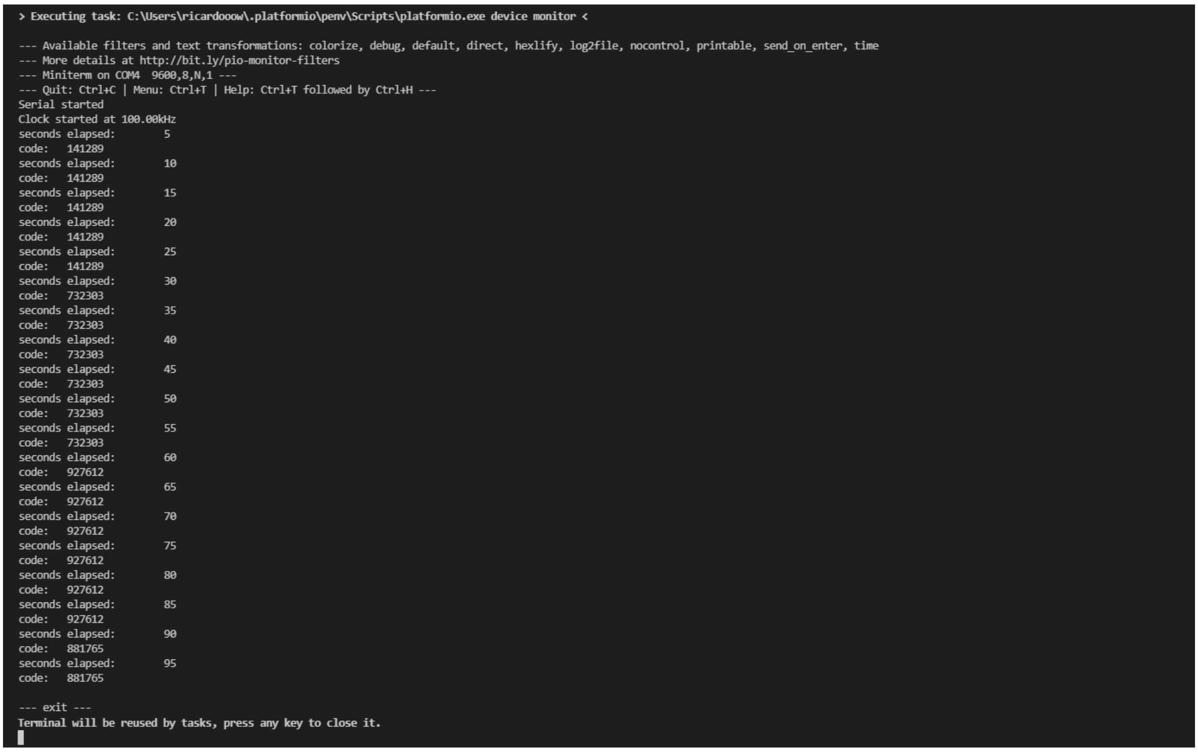
The The next test conducted is generating the secret seed used for the OTP with the additional counter as explained before, for testing purposes the datastream of new seeds have been limited to every second. As can be seen in Figure 5.4, the seed is unique in every iteration as would be expected from the random number generator. For testing purposes the seed generation has been limited to 1 seed every second.



```
> Executing task: C:\Users\ricardooow\.platformio\penv\Scripts\platformio.exe device monitor <

--- Available filters and text transformations: colorize, debug, default, direct, hexlify, log2file, nocontrol, printable, send_on_enter, time
--- More details at http://bit.ly/pio-monitor-filters
--- Miniterm on COM4  9600,8,N,1 ---
--- Quit: Ctrl+C | Menu: Ctrl+T | Help: Ctrl+T followed by Ctrl+H ---
Serial started
Clock started at 100.00kHz
BEFAF678CE7ECEFCBAFEFE76        <- random chars
9EFA4AF8DAF6FAB65E88FA8E        <- random chars
7EBE3C2CFABEFE3EDEEE765E        <- random chars
DC92EA9E3E9E7EFCF65E6876        <- random chars
4C0767CBEDEDCCEFE94CE94 <- random chars
D2DA987CFEDCCEF676DEECE0        <- random chars
FA4CEEA4FEF28084EE7CBE  <- random chars
D4FE5ED25EFECABEDCFEDC30        <- random chars
FCA2EAF6FCFCEAC6765EFEEC        <- random chars
FE6EAEFC3E9AFCDE4C3E7C8E        <- random chars
EEEAFA509C2660183A2C363A        <- random chars
A4ADC3EF6EE7094FE7E26EA <- random chars
DEFECEEE7C865ECE5EFEF2EA        <- random chars
DCBEAE74FEF63614FEECFEFE        <- random chars
FEFCB6FEDC5EDEFE9C8436A6        <- random chars
8EFC8672DEEA96FEF67EBEF6        <- random chars
BEEE706EFEFE76BEDEFAFCFC        <- random chars
983EDA782C58FE7E5E1ED6FA        <- random chars
BCF040A87EEEFEDEA0FE265E        <- random chars
ED4CADE9AEA5E9EFE38EEF0 <- random chars
C856BE9676FEFEE6F4FC86FE        <- random chars
9C98DC9624FC8A6EEEBED8EE        <- random chars
F6167E36C2DC783E7EE8C6F4        <- random chars
7EF6786ECCDE3EF4FCE0DEFE        <- random chars
7CFA30C6BED2FCFA7CBEEEF4        <- random chars
F8FEE6FEF88EE8B2F67A4A86        <- random chars
DEC6B272685CBAFC8AEE3EF4        <- random chars
E6B6767440967EFA8EDEFEFC        <- random chars
F4CE5E4AEE7A36F64E9A66E0        <- random chars
9EFEFC76EEBAFCDA58FAF414        <- random chars
80FAAC6EFEF2B6346A9EDAFE        <- random chars
CEEAEEF078CEDE7E9EFE367E        <- random chars
FEFE76C61AEEBA42FEF638FA        <- random chars
D6FCCEEC6EBAFE4E487AFAF8        <- random chars

--- exit ---

Terminal will be reused by tasks, press any key to close it.
```

Figure 5.4: Generating a new secret every second

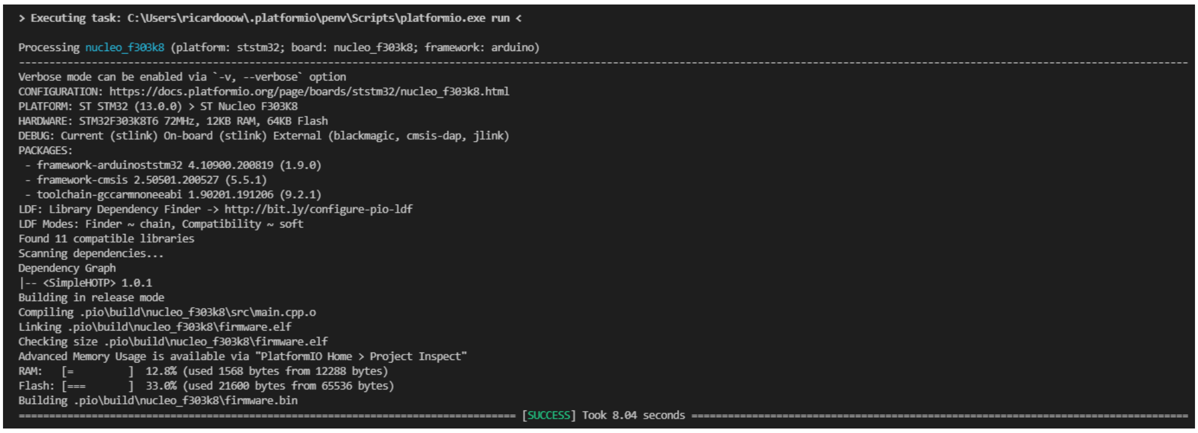
The last test conducted is the OTP generation algorithm, with the use of a single generated seed, the counter is set to increment every 30 seconds. From here the OTP is generated and as shown in Figure 5.5 the password changes every 30 seconds while being static in between those timeframes as expected from the algorithm.



```
> Executing task: C:\Users\ricardooow\.platformio\penv\Scripts\platformio.exe device monitor <

--- Available filters and text transformations: colorize, debug, default, direct, hexlify, log2file, nocontrol, printable, send_on_enter, time
--- More details at http://bit.ly/pio-monitor-filters
--- Miniterm on COM4  9600,8,N,1 ---
--- Quit: Ctrl+C | Menu: Ctrl+T | Help: Ctrl+T followed by Ctrl+H ---
Serial started
Clock started at 100.00kHz
seconds elapsed:        5
code:   141289
seconds elapsed:        10
code:   141289
seconds elapsed:        15
code:   141289
seconds elapsed:        20
code:   141289
seconds elapsed:        25
code:   141289
seconds elapsed:        30
code:   732303
seconds elapsed:        35
code:   732303
seconds elapsed:        40
code:   732303
seconds elapsed:        45
code:   732303
seconds elapsed:        50
code:   732303
seconds elapsed:        55
code:   732303
seconds elapsed:        60
code:   927612
seconds elapsed:        65
code:   927612
seconds elapsed:        70
code:   927612
seconds elapsed:        75
code:   927612
seconds elapsed:        80
code:   927612
seconds elapsed:        85
code:   927612
seconds elapsed:        90
code:   881765
seconds elapsed:        95
code:   881765

--- exit ---
Terminal will be reused by tasks, press any key to close it.
```

Figure 5.5: One-Time Password generation test with 30 second window

The total memory usage of the implementation is 1568 out of the possible 12288 bytes in ram and 21600 out of the 65536 bytes in flash memory, which is well withing the micro-controllers limits. This concludes that the code and the hardware software interface is working as desired from the subgroups software design.



```
> Executing task: C:\Users\ricardooow\.platformio\penv\Scripts\platformio.exe run <

Processing nucleo_f303k8 (platform: ststm32; board: nucleo_f303k8; framework: arduino)
--------------------------------------------------------------------------------------------------------------------------------------------------------------------
Verbose mode can be enabled via `-v, --verbose` option
CONFIGURATION: https://docs.platformio.org/page/boards/ststm32/nucleo_f303k8.html
PLATFORM: ST STM32 (13.0.0) > ST Nucleo F303K8
HARDWARE: STM32F303K8T6 72MHz, 12KB RAM, 64KB Flash
DEBUG: Current (stlink) On-board (stlink) External (blackmagic, cmsis-dap, jlink)
PACKAGES:
 - framework-arduinoststm32 4.10900.200819 (1.9.0)
 - framework-cmsis 2.50501.200527 (5.5.1)
 - toolchain-gccarmnoneeabi 1.90201.191206 (9.2.1)
LDF: Library Dependency Finder -> http://bit.ly/configure-pio-ldf
LDF Modes: Finder ~ chain, Compatibility ~ soft
Found 11 compatible libraries
Scanning dependencies...
Dependency Graph
|-- <SimpleHOTP> 1.0.1
Building in release mode
Compiling .pio\build\nucleo_f303k8\src\main.cpp.o
Linking .pio\build\nucleo_f303k8\firmware.elf
Checking size .pio\build\nucleo_f303k8\firmware.elf
Advanced Memory Usage is available via "PlatformIO Home > Project Inspect"
RAM:   [=         ]  12.8% (used 1568 bytes from 12288 bytes)
Flash: [===       ]  33.0% (used 21600 bytes from 65536 bytes)
Building .pio\build\nucleo_f303k8\firmware.bin
================================================================================= [SUCCESS] Took 8.04 seconds =================================================================================
```

Figure 5.6: Compiler statistics of the OTP software implementation

6 Statistical tests

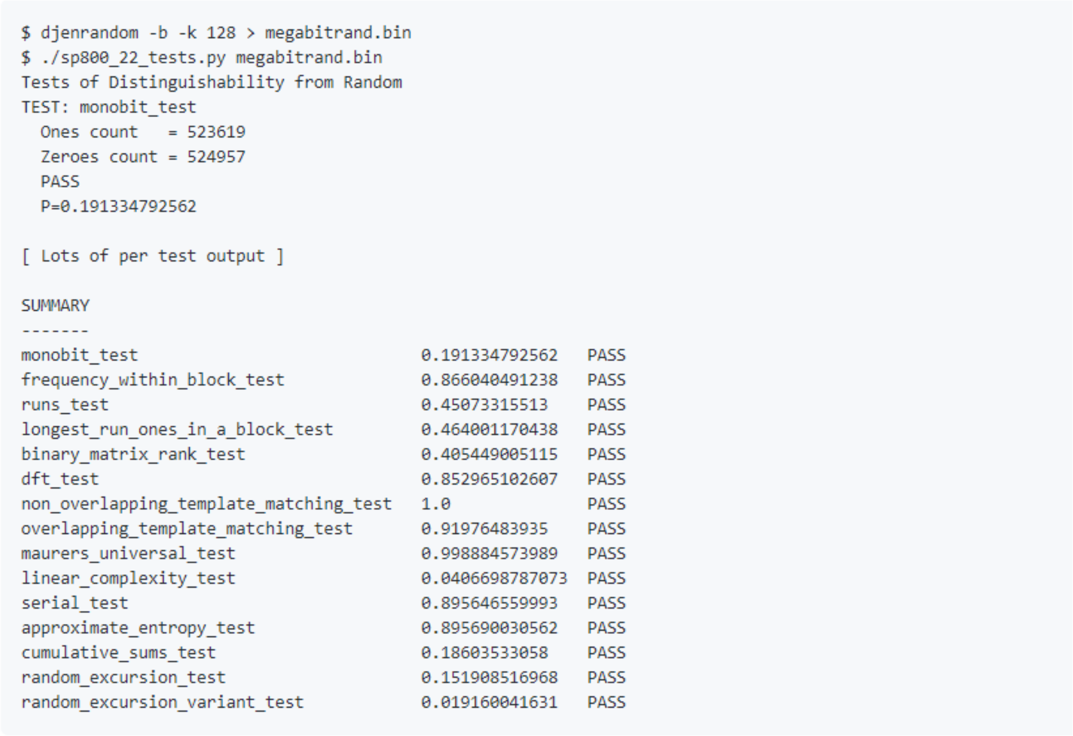
In order to verify the randomness of our QRNG, we must perform some tests on the output of the detector circuit. This output of the detector circuit will come in the form of a string of zeros and ones. Common practice is to use some kind of statistical test. Even though these tests do not prove randomness, they do however offer a first step in verifying that the RNG does not deviate from randomness.

6.1. NIST SP 800-22

The statistical test that is used is the NIST SP 800-22 A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications[1]. This is a statistical test suite designed by the National Institute of Standards and Technology of the U.S. Department of Commerce specifically with the purpose of testing RNG for cryptographic applications. This is a fitting test since the application of this system is a one time password generator.

6.2. Testing

The test suite is offered on the NIST government website. However, this version tends to crash a lot and even give wrong results[17]. So instead a python implementation of the SP800-22 Rev 1a PRNG test suite[17] is used for testing. The python implementation provides a separate python file for each of the 15 different tests. It also provides a separate python file that reads a binary data file and sends it to each of the tests, then the results of these tests are output at the end. The NIST SP 800-22 manual states that a file must be prepared consisting of a 1,000,000 bits long stream. So a .bin file consisting of 1,000,000 bits long must be collected from our QRNG and use this as the input for the NIST SP 800-22 python implementation. And this must not only be done once, but multiples of these files must be prepared so the robustness of the QRNG can be checked. The output will look as shown in Figure 6.1. A set of P-values is produced for each statistical test. A certain percentage of P-values are expected to give a fail as a result according to the significance level α The significance level is the probability of rejecting the null hypothesis when it is true. A sequence only passes the statistical test if the $P - value \geq \alpha$.

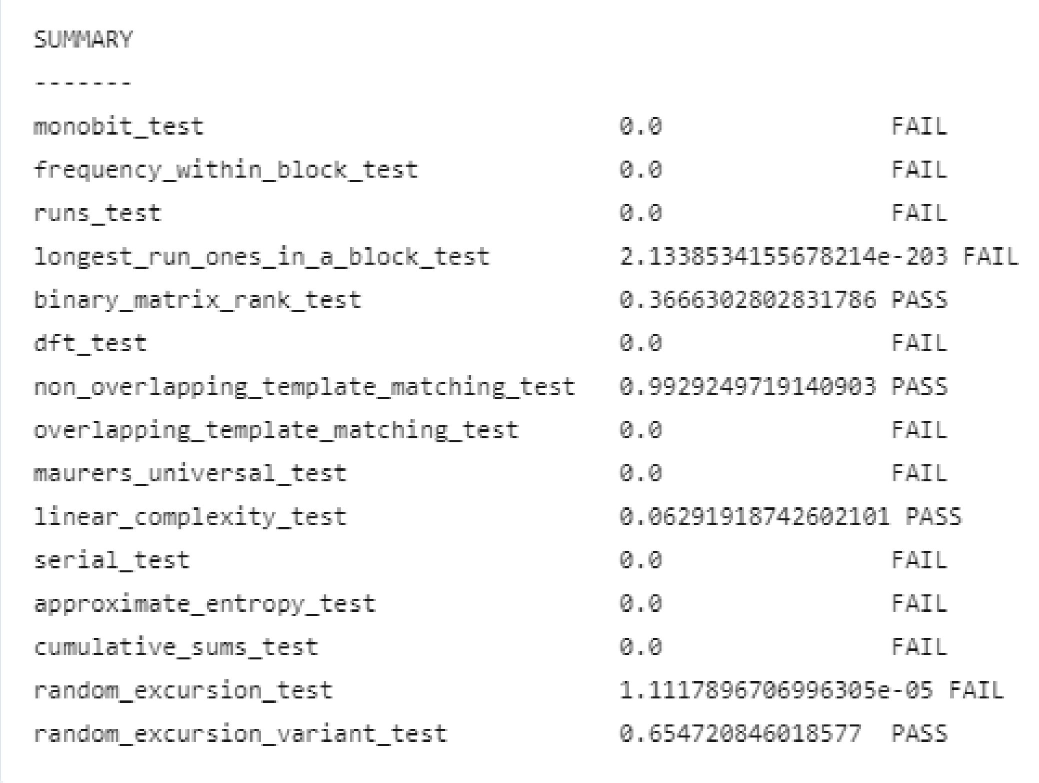


```
$ djenrandom -b -k 128 > megabitrand.bin
$ ./sp800_22_tests.py megabitrand.bin
Tests of Distinguishability from Random
TEST: monobit_test
  Ones count   = 523619
  Zeroes count = 524957
  PASS
  P=0.191334792562

[ Lots of per test output ]

SUMMARY
-------
monobit_test                             0.191334792562   PASS
frequency_within_block_test              0.866040491238   PASS
runs_test                                0.45073315513    PASS
longest_run_ones_in_a_block_test         0.464001170438   PASS
binary_matrix_rank_test                  0.405449005115   PASS
dft_test                                 0.852965102607   PASS
non_overlapping_template_matching_test   1.0              PASS
overlapping_template_matching_test       0.91976483935    PASS
maurers_universal_test                   0.998884573989   PASS
linear_complexity_test                   0.0406698787073  PASS
serial_test                              0.895646559993   PASS
approximate_entropy_test                 0.895690030562   PASS
cumulative_sums_test                     0.18603533058    PASS
random_excursion_test                    0.151908516968   PASS
random_excursion_variant_test            0.019160041631   PASS
```

Figure 6.1: NIST SP 800-22 Python implementation output example

6.3. Results

In Figure 6.2 the results from the statistical test are shown. Multiple runs were done with similar results.



```
SUMMARY
-------
monobit_test                            0.0                FAIL
frequency_within_block_test             0.0                FAIL
runs_test                               0.0                FAIL
longest_run_ones_in_a_block_test        2.1338534155678214e-203 FAIL
binary_matrix_rank_test                 0.3666302802831786 PASS
dft_test                                0.0                FAIL
non_overlapping_template_matching_test  0.9929249719140903 PASS
overlapping_template_matching_test      0.0                FAIL
maurers_universal_test                  0.0                FAIL
linear_complexity_test                  0.06291918742602101 PASS
serial_test                             0.0                FAIL
approximate_entropy_test                0.0                FAIL
cumulative_sums_test                    0.0                FAIL
random_excursion_test                   1.1117896706996305e-05 FAIL
random_excursion_variant_test           0.654720846018577  PASS
```

Figure 6.2: NIST SP 800-22 Python implementation output

6.4. Conclusion

The important things to look at here are whether a fail or pass was received for the tests and the P-value. Typically, there are 3 events that may occur as a result of the testing:

1. The P-value analysis does not indicate a randomness deviation.
2. The P-value analysis clearly shows there is a deviation from randomness.
3. The P-value analysis does not conclude anything.

From the results in Figure 6.2 it is clear that the test was failed since zeroes were received for the P-value for multiple tests.

6.5. Discussion

This failed test can be attributed to the fact that the laser is broken after trying to overcome the sensor circuit deficiency. So the random numbers that were used for this test were not generated by the quantum effect of photon spin, but rather by noise. If the system was working, we would expect different results. At least some P-value higher than 0 for all tests whether it be a fail or pass. Revisiting this test with a working laser would be a recommendation for future work.

7

Conclusion and future works

7.1. Conclusion

The requirements for the MCU, laser driver circuit and power supply circuit were meet: The power supply circuit delivers 1.5 W. Enough to power the entire 320 mW system while maintaining a stable 5 V output with an output voltage deviation of less than 1 %. The laser driver circuit operates the laser at a continuous 2.7 V and the laser maintains a stable power output of 2.5 mW, which is half it's maximum output power with a wavelength of 635 nm. The MCU chosen has sufficient storage to store the required software of 20 kB and has a sufficient clock speed to handle the required 50 kbit/s. The entire hardware circuit has been tested and works accordingly as proved in chapter 4. The PCBs designed for this system have clearances that agree with the PCB printing company. The trace and via width calculated of 1 mm are sufficient to handle the current passing through the PCBs. The beforehand calculated decoupling capacitors were sufficient to maintain stable input voltages on the ICs. Ground loops were tested and in chapter 5 it was proven not to have an impact on the circuits. The software outputs the desired OTP every 30 seconds, the seedphrase is randomly generated from the quantum random number generator and is able to create a new unique seed within a second. The serial connection between the prototype and computer are able to communicate and give details about he clock frequency, current seed and one time password for the client server side implementation. To conclude the OTP implementation is still modular with the use of the template class, while staying well withing the memory limits of the micro-controller.

7.2. future works

Now that the initial prototype has been completed, one can look to make the Quantum Random Number Generator (QRNG) prototype more production ready. One of the first steps is to post-process the Quantum Random Number Generator (QRNG) into the digital domain, this reduces the possibility of active attacks (changing the physical part of the random number generator, for example tempering with the laser, resulting in always a single state)[27]. The cryptographic algorithm used in the OTP case is the SHA-1, known as an one-way hash, which requires the counter and secret on both devices to be known. The next step will be to create a deterministic wallet, also known as the crypto wallet. This process uses a symmetric key, possible to be encrypted and decrypted.

Furthermore for the project the use of Signh(2013)[25] and Nadeem(2005)[23] will be more of use to choose the right algorithm for the increase in speed of the system, while also maintaining the Strong security.

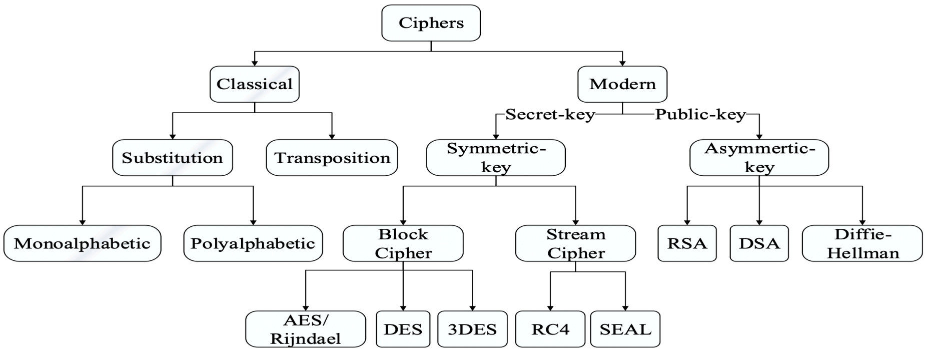


Figure 7.1: Classification of Encryption Methods[25]

In addition the approach shown in Thomas(2020) [29] will be of use, this will split the implementation between the secure element (FPGA design to increase throughput and safety) and the software/hardware to interface with another system and/or Internet of Things (IoT), for which a micro-controller with serial interface could be used. In addition to this design the power consumption will go down due to the Field Programmable Gate Array (FPGA) being more efficient than the use of a Microcontroller (MCU) as researched in Thomas(2009) [28] together with Tsoi(2003)[30] which researched the use of different implementations of the random number generator.

In the future one should focus on decreasing the prototype physical size to a more standard size like a USB stick, for this the optical system should be replaced with the use of fiber-optics. After doing a little research the optical parts used in the prototype are already available for fiber-optics, reducing the size. In addition a custom made micro-controller will be used, this reduces the size required even more.

8
Discussion

Throughout the project many new skills have been obtained. Starting with finding knowledge for the design with the aid of a literature study, which in the end resulted in the concept design depicted today. Furthermore the PCB design was a first for the team, which lead in having to dive into the books for the details on how to design them, including the trace dimensions, layer placements, ground loops. In addition during the PCB design data-sheets have been used to study the dimensions of the parts, this has been used to create custom footprints, symbols, and even 3D components for the final paper. For the circuit design a lot of collaboration was required to have both the laser circuit specifications on the same line as the sensor circuit, this includes the power-budget, for this the data sheets also have been used a lot to keep everything as coherent as possible. At last a lot of knowledge has been gained from the software design, not only about the OTP system and the inner-workings of the SHA-1 algorithm, but also to read in the data-sheet of the micro-controller, pin-configuration and the final source-code to work with the chosen design.

References

[1] E. et al. Barker. “A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications”. In: (2010). URL: https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-22r1a.pdf.

[2] *Broadband Polarizing Beamsplitter Cubes*. Accessed on June 15th, 2021. URL: https://www.thorlabs.com/newgrouppage9.cfm?objectgroup_id=4137.

[3] *Calculating trace width*. Advaced circuits. URL: https://www.4pcb.com/trace-width-calculator.html.

[4] *Classification of Polarization*. Accessed on June 15th, 2021. URL: http://hyperphysics.phy-astr.gsu.edu/hbase/phyopt/polclas.html#c3.

[5] *Constraints values EasyEDA*. EasyEDA. URL: https://jlcpcb.com/capabilities/Capabilities.

[6] *Decoupling capacitor calculations*. Sierra circuits. URL: https://www.protoexpress.com/blog/decoupling-capacitor-use/.

[7] D Eastlake 3rd and Paul Jones. *Rfc3174: Us secure hash algorithm 1 (sha1)*. 2001.

[8] *Ground loop theory*. Cadence. URL: https://resources.pcb.cadence.com/blog/2019-what-is-a-ground-loop-and-how-to-minimize-its-harmful-consequences.

[9] Bill Hill. *Aspheric Lens*. Accessed on June 15th, 2021. 2017. URL: https://escooptics.com/blogs/news/concepts-in-light-and-optics-lenses-part-4-aspheres.

[10] *HL6312G/13G - AlGaInP Laser Diodes*. Opnext Japan, Inc. 190 Kashiwagi, Komoro-shi, Nagano 384-8511, Japan, 2005. URL: https://www.thorlabs.com/drawings/59685548d361dcc7-D8884119-93E6-7C93-10FF2DFF1134DCE7/HL6312G-MFGSpec.pdf.

[11] *HL6319G/20G - AlGaInP Laser Diodes*. Opnext Japan, Inc. 190 Kashiwagi, Komoro-shi, Nagano 384-8511, Japan, 2005. URL: https://www.thorlabs.com/drawings/59685548d361dcc7-D8884119-93E6-7C93-10FF2DFF1134DCE7/HL6320G-MFGSpec.pdf.

[12] *HL6321G/22G - AlGaInP Laser Diodes*. Opnext Japan, Inc. 190 Kashiwagi, Komoro-shi, Nagano 384-8511, Japan, 2005. URL: https://www.thorlabs.com/drawings/59685548d361dcc7-D8884119-93E6-7C93-10FF2DFF1134DCE7/HL6322G-MFGSpec.PDF.

[13] *iC-WK, iCWKL Application Notes - 2.4V CW LASER DIODE DRIVER*. ICHaus. Am Kuemmerling 18 - D-55294 Bodenheim, Germany. URL: https://www.ichaus.de/upload/pdf/WKN_datasheet_C1en.pdf.

[14] *iC-WKL - 2.4V CW LASER DIODE DRIVER*. ICHaus. Am Kuemmerling 18 - D-55294 Bodenheim, Germany. URL: https://www.ichaus.de/upload/pdf/WK__WKL_datasheet_D2en.pdf.

[15] *iC-WKN - 15V CW LASER DIODE DRIVER*. ICHaus. Am Kuemmerling 18 - D-55294 Bodenheim, Germany. URL: https://www.ichaus.de/upload/pdf/WKN_datasheet_C1en.pdf.

[16] *ILX Application notes #3 Protecting your laser diode*. ILX Lightwave corporation. 31950 Frontage Road, Bozeman, MT 59771-6310, USA. URL: https://www.newport.com/medias/sys_master/images/images/hc0/h01/8797049454622/AN03-Protecting-Your-Laser-Diode.pdf.

[17] David Johnston. *dj-on-github/sp800_22_tests*. Accessed on May 14th, 2021. 2018. URL: https://github.com/dj-on-github/sp800_22_tests.

[18] Jonathan Lusky. *jlusPrivat/SimpleHOTP*. Accessed on June 1st, 2021. 2019. URL: https://github.com/jlusPrivat/SimpleHOTP.

[19] David M’Raihi et al. “HOTP: An HMAC-Based One-Time Password Algorithm”. In: *Internet Request for Comments* (2005). URL: https://datatracker.ietf.org/doc/html/rfc4226.

[20] Xiongfeng Ma et al. “Quantum random number generation”. In: *npj Quantum Information* 2.16021 (2016). DOI: `https://doi.org/10.1038/npjqi.2016.21`.

[21] Kraig Mitzner. “Complete PCB Design Using OrCAD Capture and PCB Editor”. In: (2009), p. 488.

[22] *Molded GlassAspheric Lens*. Accessed on June 15th, 2021. URL: `https://www.thorlabs.com/newgrouppage9.cfm?objectgroup_id=3810&pn=C060TMD-A`.

[23] Aamer Nadeem and M Younus Javed. “A performance comparison of data encryption algorithms”. In: *2005 international Conference on information and communication technologies*. IEEE. 2005, pp. 84–89.

[24] *Polarizer*. Accessed on June 15th, 2021. URL: `https://www.thorlabs.com/newgrouppage9.cfm?objectgroup_id=10498`.

[25] Gurpreet Singh. “A study of encryption algorithms (RSA, DES, 3DES and AES) for information security”. In: *International Journal of Computer Applications* 67.19 (2013).

[26] M Stipčević. “Quantum random number generators and their use in cryptography”. In: *2011 Proceedings of the 34th International Convention MIPRO*. IEEE. 2011, pp. 1474–1479.

[27] Berk Sunar, William J Martin, and Douglas R Stinson. “A provably secure true random number generator with built-in tolerance to active attacks”. In: *IEEE Transactions on computers* 56.1 (2006), pp. 109–119.

[28] David Barrie Thomas, Lee Howes, and Wayne Luk. “A comparison of CPUs, GPUs, FPGAs, and massively parallel processor arrays for random number generation”. In: *Proceedings of the ACM/SIGDA international symposium on Field programmable gate arrays*. 2009, pp. 63–72.

[29] T. Thomas et al. “Memory FORESHADOW: Memory FOREnSics of HArDware CryptOcurrency wallets–A Tool and Visualization Framework”. In: *Forensic Science International: Digital Investigation* 33 (2020), p. 301002.

[30] Kuen Hung Tsoi, KH Leung, and Philip Heng Wai Leong. “Compact FPGA-based true and pseudo random number generators”. In: *11th Annual IEEE Symposium on Field-Programmable Custom Computing Machines, 2003. FCCM 2003.* IEEE. 2003, pp. 51–61.

Acronyms

BOM Bill of materials. 37

FPGA Field Programmable Gate Array. 34, 37

GERBER open ASCII vector format for printed circuit board (PCB) designs. 37

HOTP HMAC-Based One-Time Password. 23, 24, 37

IoT Internet of Things. 34, 37

MCU Microcontroller. i, 2, 34, 37

OTP One-Time Password. i, 23, 37

PCB Printed circuit board. 17, 37
PCB Printed Circuit Board. i, 37
PRNG Pseudo Random Number Generator. 1, 37

QRNG Quantum Random Number Generator. i, 34, 37

RNG Random Number Generator. 1, 37

SMD Surface mount device. i, 18, 37

TH Through hole. 18, 37
TH Through-Hole. i, 37
TOTP Time-Based One-Time Password. 23, 37
TRNG True Random Number Generator. 1, 37

A Circuit

A.1. Single PCB design

A.1.1. PCB_V1

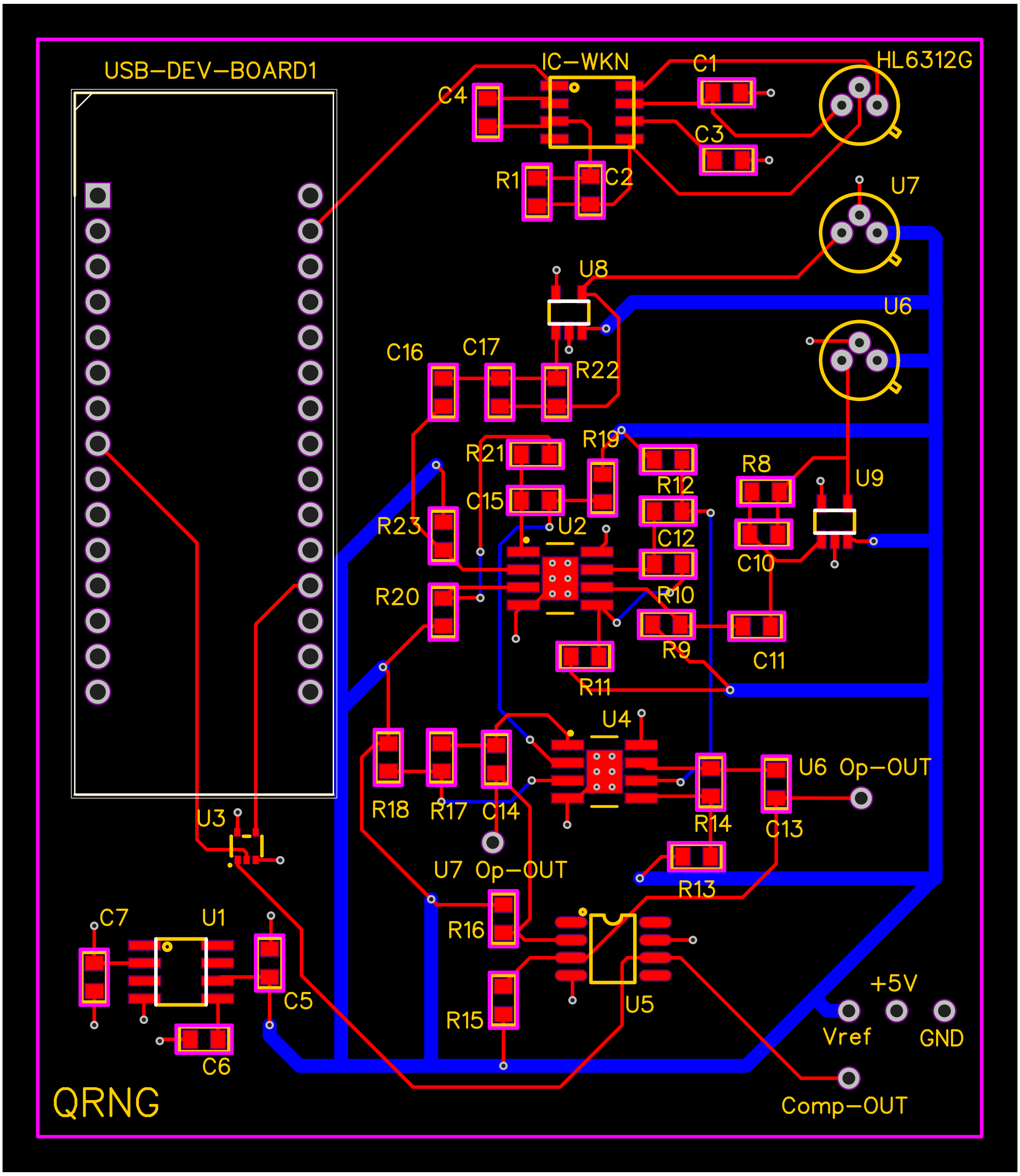


Figure A.1: 4-layer PCB design

A.1.2. PCB_V2

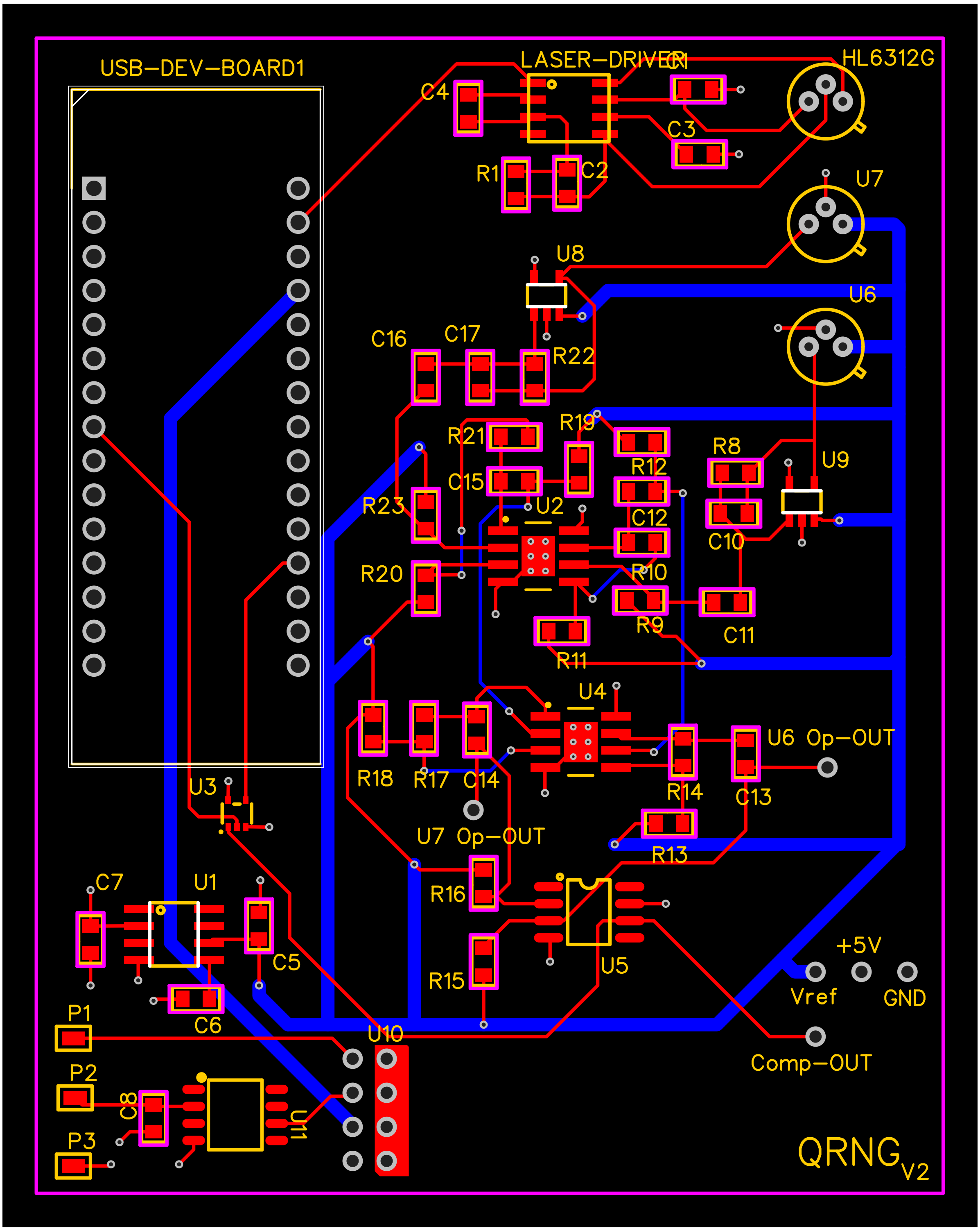


Figure A.2: 4-layer PCB design with 3 way external power capabilities

A.2. Multi PCB design

A.2.1. PCB_V1

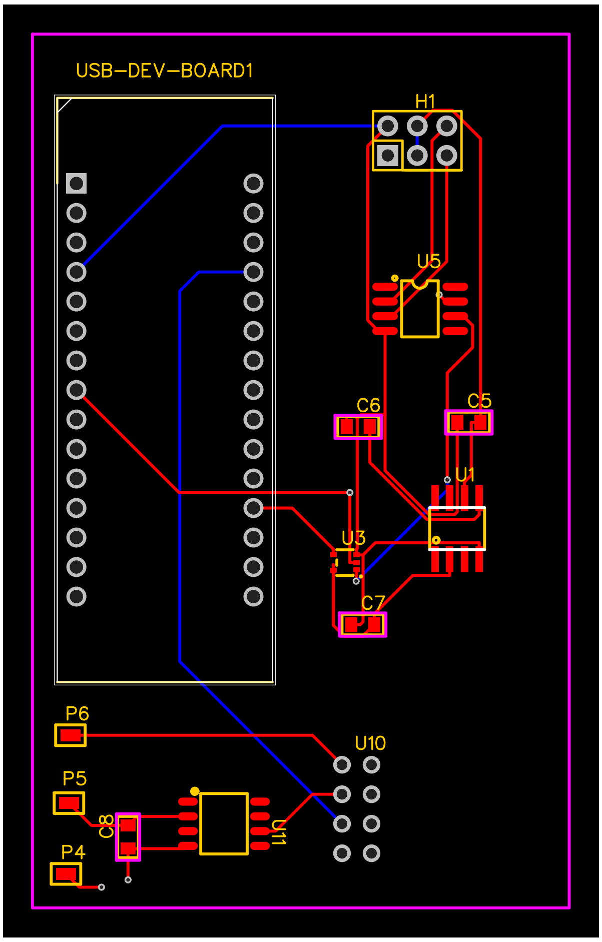


Figure A.3: 4-layer main PCB design with 3 way external power capabilities

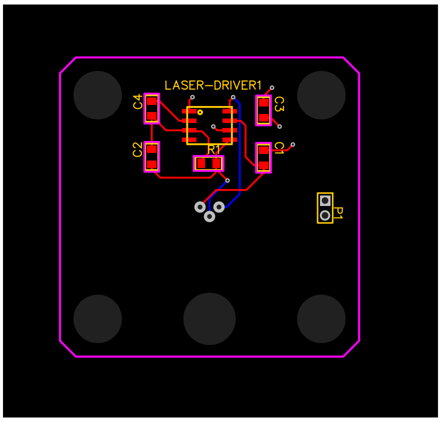


Figure A.4: 4-layer laser PCB design in cage plate dimensions

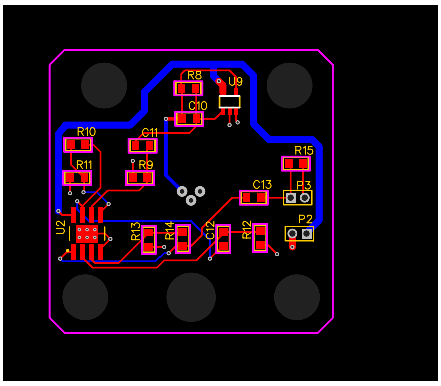


Figure A.5: 4-layer sensor PCB design in cage plate dimensions

A.2.2. 3D view

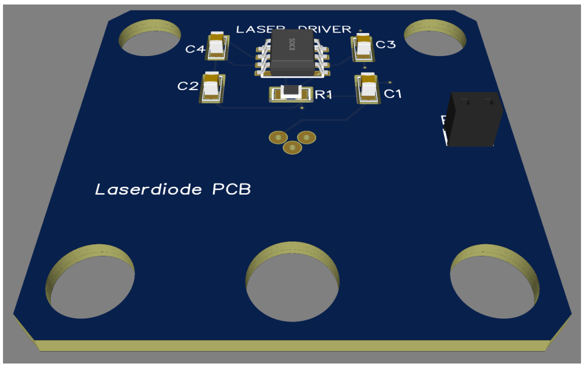


Figure A.6: Laser PCB

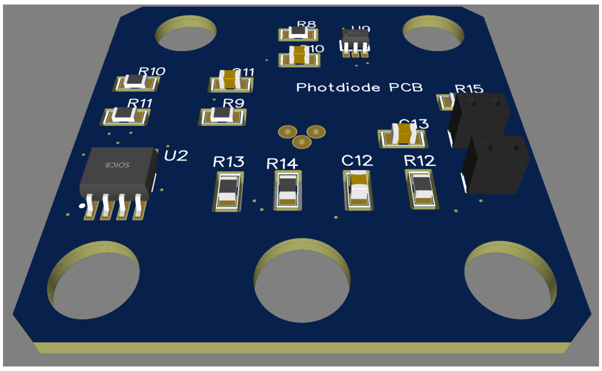


Figure A.7: Sensor PCB

A.2.3. PCB_V2

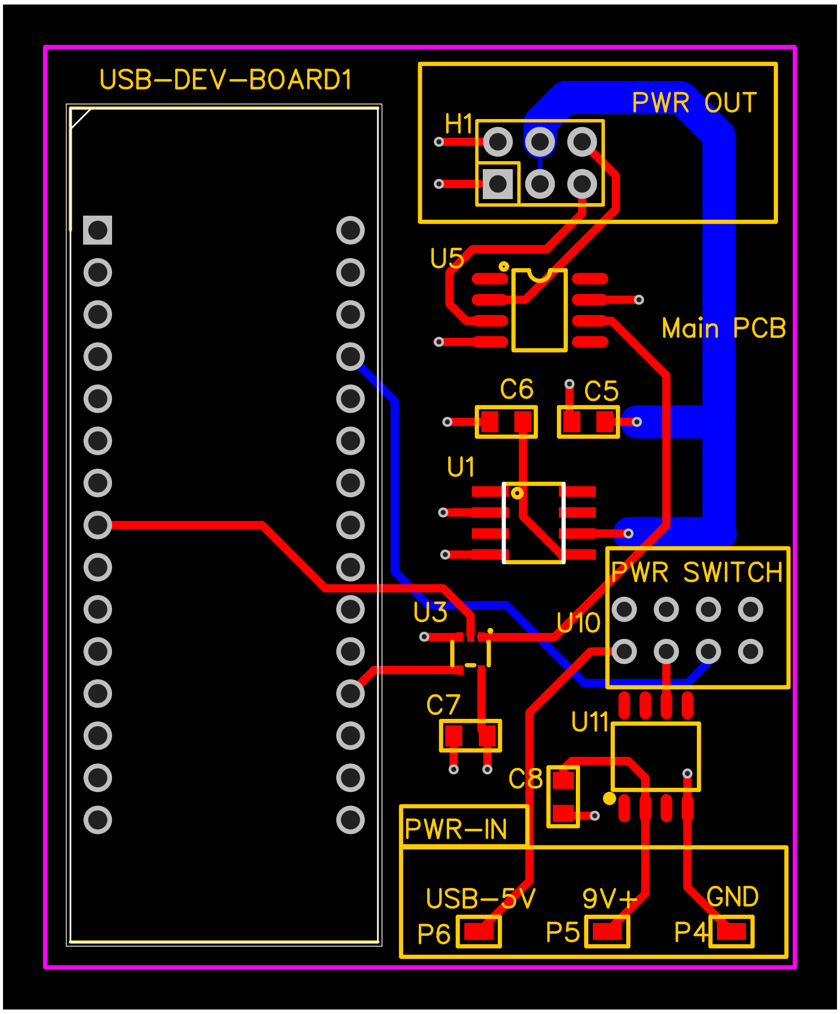


Figure A.8: 4-layer main PCB design with 3 way external power capabilities

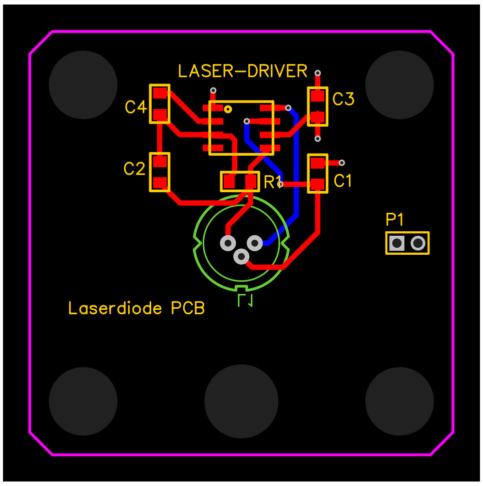


Figure A.9: 4-layer laser PCB design in cage plate dimensions

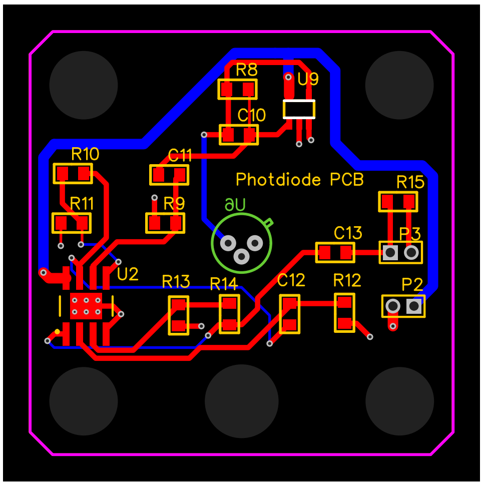


Figure A.10: 4-layer sensor PCB design in cage plate dimensions

B

Schematic

B.1. Circuit design

B.1.1. Main Circuit

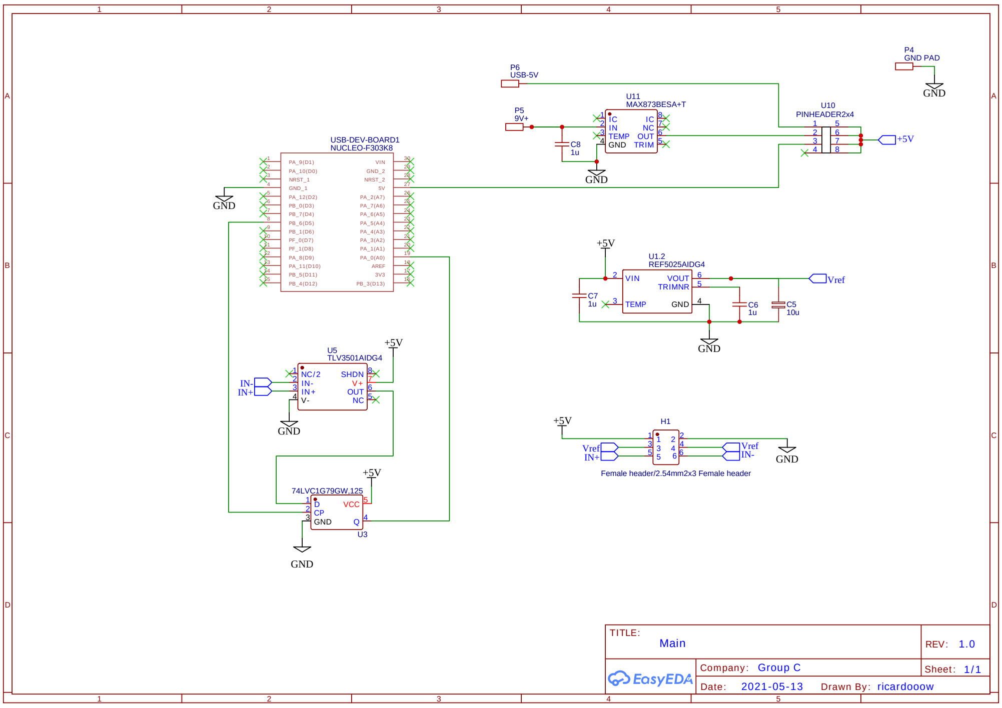


Figure B.1: Main circuit with with external power capabilities using a switch

B.1.2. Laser Circuit

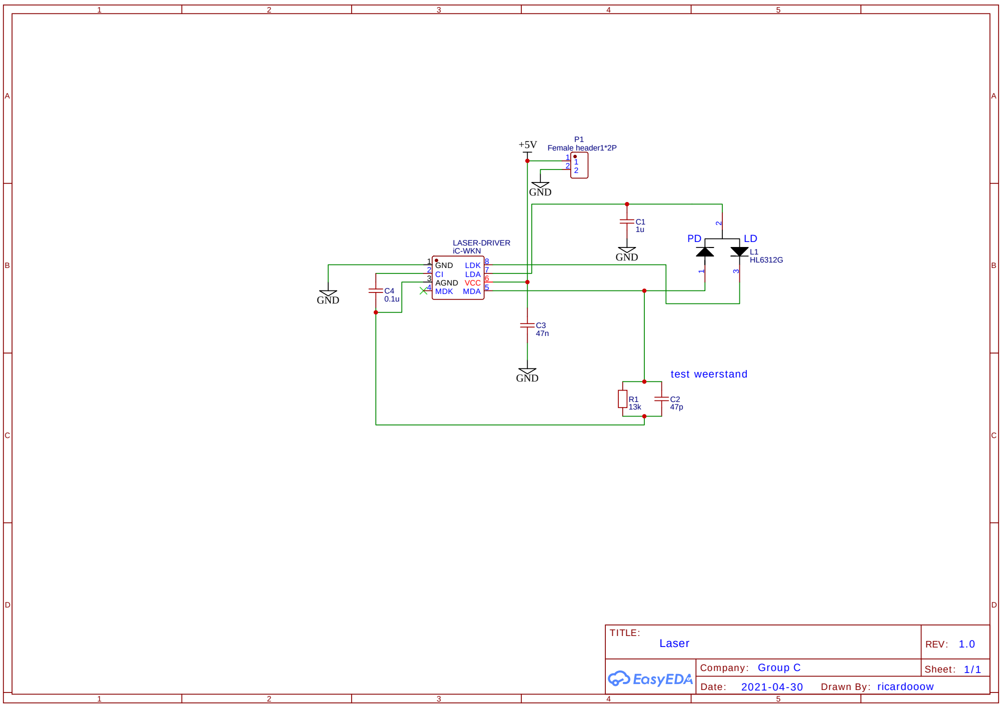


Figure B.2: Laser driver circuit with power header

C

Code

C.1. Main.cpp

```
#include <Arduino.h>
#include <pin.hpp>
#include <SimpleHOTP.h>
#include <Timer.hpp>

DigitalPin<D5> clk;         //pin D5 = clk
DigitalPin<A0> sensor_in;   //pin A0 = in

Timer clock = Timer();

int counter = clock.get();
char secret[] = "secret code";

Key key(secret,sizeof(secret)-1);
SimpleHOTP hotp(key, counter);

void setup(){
        clk.initialize(OUTPUT);
        sensor_in.initialize(INPUT);
        Serial.begin(9600);

}

void loop(){
    clock.update()
        counter = clock.get();
        auto otp = hotp.generateHOTP(counter);
        Serial.println(otp);
}
```

C.2. Pin.hpp

```
#include <Arduino.h>

template <uint8_t PIN>
class DigitalPin
{
public:
    static void initialize(int IO)
    {
        pinMode(PIN,IO);
    }

    static void set(bool state)
    {
        digitalWrite(PIN,state);
    }

    static int get()
    {
        return digitalRead(PIN);
    }

    static void toggle()
    {
        digitalWrite(PIN, !digitalRead(PIN));
    }
};
```

C.3. Timer.hpp

```
class Timer
{
private:
    uint32_t start;
    int counter;
public:
    Timer(){
        start = millis();
    }
    void Update(){
        counter = static_cast<int> ((millis() - start )/30000)

    }
    int get(){
        return counter;
    }
};
```

C.4. Main_full.cpp

```cpp
#include <SimpleHOTP.h>

// Include external files:
#include <config.h>
#include <RandomBitExtractor.h>
#include <Timer.hpp>

// Create objects:
RandomBitExtractor RBE;
Timer clk = Timer();

// Define system variables:
int16_t LifeLedCounter;     // Create an integer to use for the life led
int16_t counter = clk.get();
uint8_t secret_length = 12;
uint8_t secret[] = "secret code!";


Key key(secret, sizeof(secret)-1);
SimpleHOTP hotp(key, counter);

void setup() {
  #ifdef USE_SERIAL
    // Start the serial communication:
    Serial.begin(9600);
    Serial.println("Serial started");
  #endif

  RBE.set_timer(DETECTOR_CLOCK_FREQUENCY);

  // Define the outputs:
  pinMode(LED_BUILTIN, OUTPUT);
}

void loop() {
    // Blink the life led to indicate the system is operational
  digitalWrite(LED_BUILTIN, LifeLedCounter++ > 0);
  if(RBE.CharAvailable() > secret_length){
        int elapsed = (millis() - clk.epoch());
    if(!((elapsed) % 1000)){
                int minutes = (elapsed / 60000);
                int seconds = (elapsed) / 1000;
                Serial.print("seconds elapsed: \t");
                  Serial.println(seconds);
                for(int i = 0; i < secret_length; i++){
                      char val = RBE.ReadChar();
                      secret[i] = val;
                    Serial.print(val,HEX);
            }
                Serial.println("\t<- random chars");
            Serial.print("code:\t");
                clk.Update();
            counter = clk.get();
            auto otp = hotp.generateHOTP(counter);
            Serial.println(otp);
        }
  }
}

void DetectorClockCallbackWrapper(){
  RBE.ClockPeriodCallback();
}
```

C.5. RandomBitExtractor.cpp

```
#include <RandomBitExtractor.h>

extern void DetectorClockCallbackWrapper();
void ClockPeriodCallback();

RandomBitExtractor::RandomBitExtractor(){
  ClockTimerChannel =
   ↪  STM_PIN_CHANNEL(pinmap_function(digitalPinToPinName(CLOCK_OUTPUT),
   ↪  PinMap_PWM));
}

void RandomBitExtractor::set_timer(uint32_t frequency){
  HardwareTimer *ClockTimer = new HardwareTimer(TIM16);
  ClockTimer->setPWM(ClockTimerChannel, CLOCK_OUTPUT, frequency, 50,
   ↪  DetectorClockCallbackWrapper);
  #ifdef USE_SERIAL
    Serial.print("Clock started at ");
    Serial.print(frequency/1e3);
    Serial.println("kHz");
  #endif
  pinMode(DETECTOR_INPUT, INPUT);
}

void RandomBitExtractor::ClockPeriodCallback(){
  if(BufferPointer < DETECTOR_BUFFER_LENGTH)
    RandomBitsBuffer[BufferPointer++] = digitalRead(DETECTOR_INPUT);
}

uint16_t RandomBitExtractor::DataAvailable(){
  return BufferPointer;
}

uint16_t RandomBitExtractor::CharAvailable(){
  return BufferPointer / 7;
}

char RandomBitExtractor::ReadChar(){
  uint8_t val = 0;
  for(int i = 0; i < 7; i++){
    val += ReadBit();
    val = val << 1;
  }
  return val;
}

bool RandomBitExtractor::ReadBit(){
  while(!DataAvailable()){delay(1);}
  return RandomBitsBuffer[--BufferPointer];
}
```

C.6. RandomBitExtractor.hpp

```
#ifndef _RANDOMBITEXTRACTOR_
#define _RANDOMBITEXTRACTOR_

#include "config.h"
#include <Arduino.h>

class RandomBitExtractor{
  public:
    RandomBitExtractor();
    void set_timer(uint32_t);
    void ClockPeriodCallback();
    uint16_t DataAvailable();
    uint16_t CharAvailable();
    bool ReadBit();
    char ReadChar();
  private:
    uint32_t ClockTimerChannel;
    bool RandomBitsBuffer[DETECTOR_BUFFER_LENGTH];
    uint16_t BufferPointer;
};


#endif
```

C.7. config.hpp

```
#define USE_SERIAL   // Use the serial monitor to print information

#define DETECTOR_CLOCK_FREQUENCY 100e3


// Define pin in/outputs:
#define CLOCK_OUTPUT D5
#define DETECTOR_INPUT A0
#define DETECTOR_BUFFER_LENGTH 500
```

D

Calculations

D.1. Laser driver biasing point

From the laser datasheet [10], it was chosen to make the output power half of the maximum. This resulted in a monitor current (I_s) of around 0.23 mA. From Equation D.1 and the laser driver datasheet [15][13] it then was found that $R_\mathrm{M} \approx 13\mathrm{k}\Omega$ gave the desired output current. This will give enough leeway to adapt the output power if desired.

$$R_\mathrm{M} = \frac{V_\mathrm{MDA}}{I_\mathrm{MD}} = \frac{V_\mathrm{cc} - V_\mathrm{R(LD)}}{I_\mathrm{s}} \tag{D.1}$$

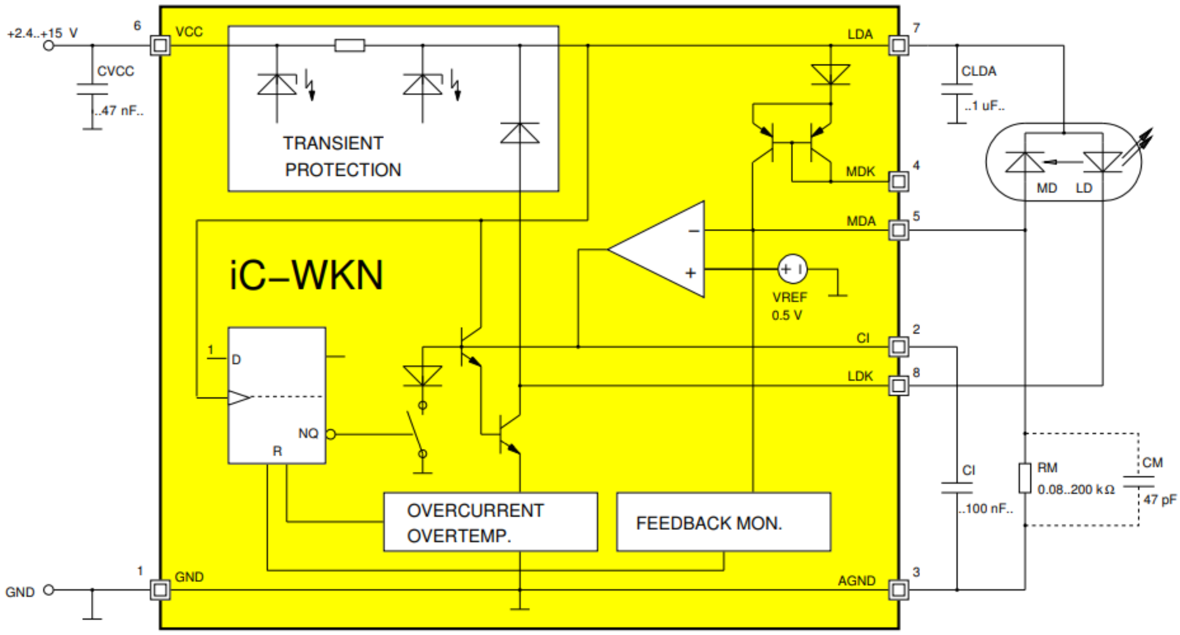


Figure D.1: Laser driver schematic[15]

E
Planning

Below a detailed planning is to be found for group C1.

Week	Tasks
1	Literature study and defining the groups
2	Designing the system
3	Designing the system
4	Ordering parts and designing PCB
5	Design different power sources
6	Build and test circuit without optical parts
7	Debugging system and designing second PCB
8	Work on software
9	Complete unfinished tasks + thesis
10	Testing and comparing system with QRNG based on quantum computer
11	Grand Finale

Table E.1: Time schedule

F

Figures

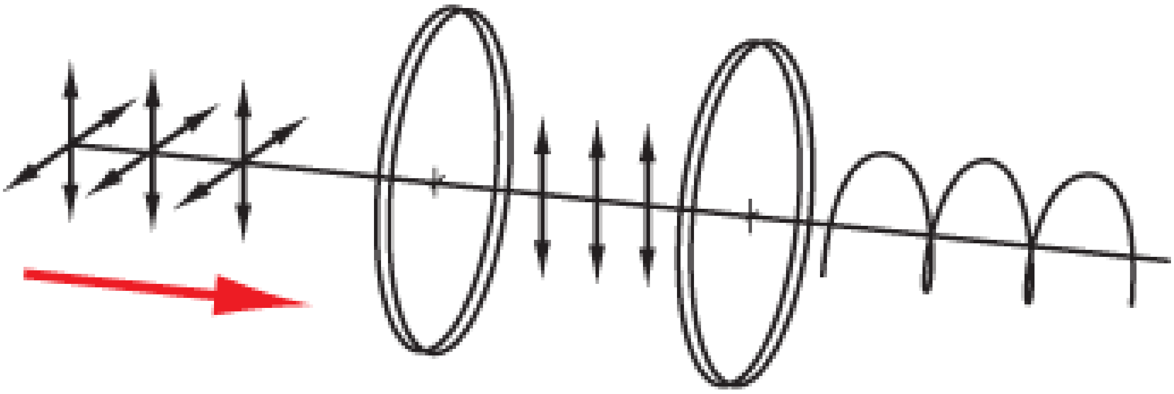


Figure F.1: Creation of left-handed circularly polarized light from unpolarized light[24]