UNIVERSITATEA BABEŞ-BOLYAI
BABEŞ-BOLYAI TUDOMÁNYEGYETEM
BABEŞ-BOLYAI UNIVERSITÄT
BABEŞ-BOLYAI UNIVERSITY

FACULTATEA DE FIZICĂ
Str. Mihail Kogălniceanu nr.1
Cluj-Napoca, RO-400084
Tel: +4(0)264-405300 | FAX: +4(0)264-591906
secretariat.phys@ubbcluj.ro
www.phys.ubbcluj.ro

LUCRARE DE DIPLOMĂ

Coordonator ştiinţific **Absolvent**

dr. Borbély Sándor Csiszér Csanád

2025

UNIVERSITATEA BABEŞ-BOLYAI
BABEŞ-BOLYAI TUDOMÁNYEGYETEM
BABEŞ-BOLYAI UNIVERSITÄT
BABEŞ-BOLYAI UNIVERSITY

FACULTATEA DE FIZICĂ
Str. Mihail Kogălniceanu nr.1
Cluj-Napoca, RO-400084
Tel: +4(0)264-405300 | FAX: +4(0)264-591906
secretariat.phys@ubbcluj.ro
www.phys.ubbcluj.ro

LUCRARE DE DIPLOMĂ

CONSTRUIREA ȘI OPERAREA UNUI DISPOZITIV DE NUMĂRARE A FOTONILOR CORELAT ÎN TIMP (TCSPC)

Coordonator ştiinţific dr. Borbély Sándor

Absolvent Csiszér Csanád

2025

UNIVERSITATEA BABEŞ-BOLYAI
BABEŞ-BOLYAI TUDOMÁNYEGYETEM
BABEŞ-BOLYAI UNIVERSITÄT
BABEŞ-BOLYAI UNIVERSITY

FACULTATEA DE FIZICĂ
Str. Mihail Kogălniceanu nr.1
Cluj-Napoca, RO-400084
Tel: +4(0)264-405300 | FAX: +4(0)264-591906
secretariat.phys@ubbcluj.ro
www.phys.ubbcluj.ro

ÁLLAMVIZSGA DOLGOZAT

IDŐ-KORRELÁLT EGYFOTONSZÁMLÁLÓ BERENDEZÉS (TCSPC) ÉPÍTÉSE ÉS MŰKÖDTETÉSE

Témavezető tanár dr. Borbély Sándor

Hallgató Csiszér Csanád

2025

Kivonat

A dolgozat bemutatja egy idő-korrelált egyfotonszámláláson (TCSPC - Time-Correlated Single Photon Counting) alapuló mérőrendszer tervezését és megépítését, amely lehetővé teszi fluoreszcens anyagok élettartamának meghatározását. A rendszer célja kifejezetten alacsony intenzitású fluoreszcens jelek detektálása, így különös figyelmet fordítottunk a zajcsökkentésre és árnyékolásra. A berendezés fő egységei egy gerjesztő UV lézer, két egyfoton detektor (SPAD - Single Photon Avalanche Diode) és egy jelszámláló kártya, amely vezérlését saját fejlesztésű szoftver biztosítja.

A mért jeleket konvolúciós és dekonvolúciós módszerekkel is elemeztük, az utóbbi bizonyult hatékonyabbnak. A rendszer működését klorofilloldat vizsgálatával demonstráltuk, a kapott élettartam értékek összhangban vannak a szakirodalmi adatokkal.

Az eredmények igazolják, hogy a saját építésű mérőrendszer alkalmas fluoreszcens oldatok élettartamának meghatározására. A dolgozatban bemutatott módszer rugalmas és könnyen bővíthető, így akár további fejlesztéssel ellátva, eszközt biztosíthat más egyetemi karokon tevékenykedő kutatók számára is.

Abstract

The thesis presents the design and construction of a time-correlated single photon counting (TCSPC) device that allows fluorescent lifetime measurements. The system is specifically designed for detecting low-intensity fluorescent signals, so special attention was paid to noise reduction and shielding. The main components of the device are an UV laser for excitation, two single photon avalanche detectors (SPAD), and a time-to-digital converter card, which is controlled by a custom-developed software.

The measured signals were analyzed using both convolution and deconvolution methods, with the latter proving to be more effective. The system's operation was demonstrated by examining a chlorophyll solution, with the obtained lifetime values being in agreement with literature data.

The results confirm that the custom-built measurement system is suitable for determining the lifetimes of fluorescent solutions. The method presented in the thesis is flexible and easily expandable, making it a potential tool for researchers in other university departments with minimal further development.

Tartalomjegyzék

1. Rövidítések

A dolgozatban használt rövidítések listája:

- TCSPC: Time-Correlated Single Photon Counting – idő-korrelált egyfotonszámlálás
- FP: Fluorescent Photon – fluoreszcens foton
- QY: Quantum Yield – kvantumhozam
- PMT: Photomultiplier Tube – fotoelektron sokszorozó cső
- APD: Avalanche Photodiode – lavina fotodióda
- SPAD: Single-Photon Avalanche Diode – egyfotonos lavina fotodióda
- IRF: Instrument Response Function – műszer válaszfüggvénye
- TTL: Transistor-Transistor Logic – tranzisztor-tranzisztor logika

2. Bevezető

2.1. Fluoreszcencia

A fotolumineszcencia olyan fizikai jelenség, amely során egy anyag elnyel egy fotont, gerjesztett állapotba kerül, majd ezt követően visszatér egy alacsonyabb energiaszintre, miközben újra fotont bocsát ki. A fluoreszcencia esetében a fotonelnyelés és kibocsájtás közti időintervallum jellemzően néhány nanoszekundum, ami jól mérhető modern optikai technikákkal.

Fluoreszcencia során az anyagot alkotó molekulák gerjesztett állapotba kerülnek, innen nem-radiatívan relaxálnak egy alacsonyabb energiaszintre, majd a gerjesztő fotonnál kisebb energiájú fotont kibocsátva lekerülnek az eredeti (kiindulási) szintre. Mivel a kibocsátás nem azonnali, az így keletkező fény időbeli viselkedéséből fontos információk nyerhetők ki az anyag szerkezetére, kölcsönhatásaira és környezetére vonatkozóan. A fluoreszcencia jelenség kvantitatív jellemzéséhez a kvantumhozamot és az élettartamot használjuk: előbbi a kibocsátott és elnyelt fotonok arányát, utóbbi a fényintenzitás lecsengésének karakterisztikus idejét írja le.

A fluoreszcencia kvantumhozam (QY - quantum yield ϕ) a kibocsátott (N_{FP}) és elnyelt fotonok (N_{abs}) arányát adja meg:

$$\phi = \frac{N_{FP}}{N_{abs}}. \tag{1}$$

A kvantumhozam értéke 0 és 1 között mozog, ahol az 1 azt jelenti, hogy minden elnyelt foton után egy FP keletkezik, míg a 0 azt jelenti, hogy egy elnyelt foton sem eredményez fluoreszcens kibocsátást. A kvantumhozamot jellemezhetjük az egységnyi idő alatt kibocsátott FP-ok (k_{FP}) és a fluoreszcens, nem-radiatív (k_{NR}) és ütközés általi tranziciók (k_{ET}) rátáinak arányával is:

$$\phi = \frac{k_{FP}}{k_{FP} + k_{NR} + k_{ET}}, \tag{2}$$

A fluoreszcencia élettartam (τ) az emisszióval záruló relaxáció időállandóját jelenti. Segítségével a következőkeppen jellemezhetjük a fluoreszcencia időbeli viselkedését:

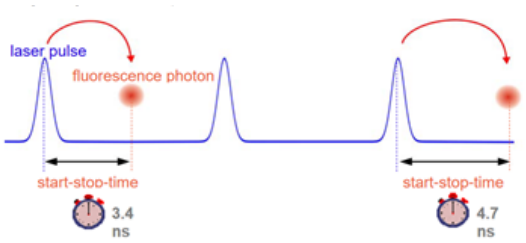
$$p(t) \propto e^{-\frac{t}{\tau}}, \tag{3}$$

ahol $p(t)$ a fluoreszcens molekula radiatív relaxációjának valószínűsége az idő függvényében.

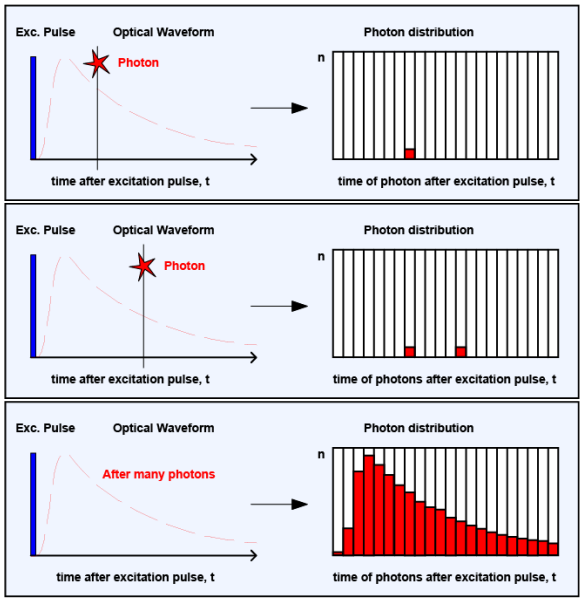
2.2. TCSPC

A fluoreszcencia élettartamának pontos mérése fontos szerepet játszik a biofizikában, anyagtudományban és orvosi diagnosztikában is. A legelterjedtebb technika ezen mennyiség mérésére az *idő-korrelált egyfotonszámlálás (Time-Correlated Single Photon Counting – TCSPC)*, amely lehetővé teszi a fluoreszcens fotonok kibocsátásának időbeli statisztikai elemzését.

A TCSPC során a mintát egy időzített fényimpulzus gerjeszti, majd a kibocsátott fluoreszcens fotonok érkezési idejét (2.1. ábra) egy rendkívül pontos időzítő rögzíti. Fontos hogy ciklusonként csak egyetlen fotont regisztráljunk, másképp a detektor holtideje miatt a mérés pontatlan lehet. A mérési ciklusokat ismételve, a detektor által észlelt fotonok érkezési idejét egy hisztogramon ábrázoljuk (2.2. ábra), mind a gerjesztő impulzus, mind a fluoreszcens fotonok esetében. A hisztogramokból a fluoreszcencia élettartamára vonatkozó információk nyerhetők ki, amelyeket általában exponenciális illesztéssel elemeznek. A TCSPC technika



2.1. ábra. TCSPC egy mérési ciklusának vázlata [14]



2.2. ábra. Fotonérkezési statisztika felépítése több mérési ciklus során [1]

nagy előnye, hogy rendkívül érzékeny, képes detektálni a nagyon gyenge fluoreszcens jeleket is, csupán idő kérdése egy jó minőségű mérés elvégzése.

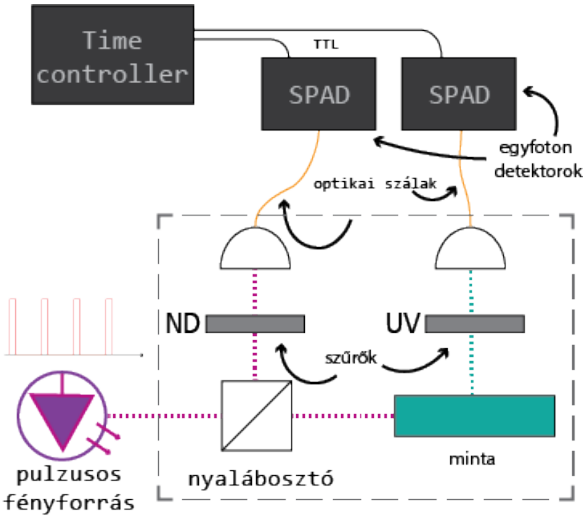
A kereskedelemben kapható eszközök időfelbontása pár picoszekundumtól kezdődik, a gerjesztési impulzusok $10kHz$-től akár $100MHz$-ig is terjedhetnek, de ez a választás a kutatási igénytől függ. A nagy felbontású TCSPC moduljairól ismert *Becker & Hickl*, olyan rendszereket kínál, amelyek $1.1ps$ időfelbontást képesek elérni. Termékeik többcsatornás

TCSPC technikákat támogatnak, ezzel nem csak idő, hanem hullámhosszfüggő elemzést is végre lehet egyszerűen hajtani [3]. A *PicoQuant* rendszerei rugalmasságra lettek tervezve, támogatják a különböző időzítési módokat és interfészeket, és egycsatornás és többcsatornás alkalmazásokhoz egyaránt alkalmasak [6]. A *HORIBA* DeltaFlex TCSPC rendszere egy moduláris és testreszabható megoldás a fluoreszcencia élettartamának mérésére. Pikoszekundumtól másodpercekig terjedő széles élettartam-tartományt támogat, és kompatibilis különböző fényforrásokkal, beleértve a pulzáló diódalézereket és LED-eket, így sokféle kísérleti felépítéshez alkalmazható [7].

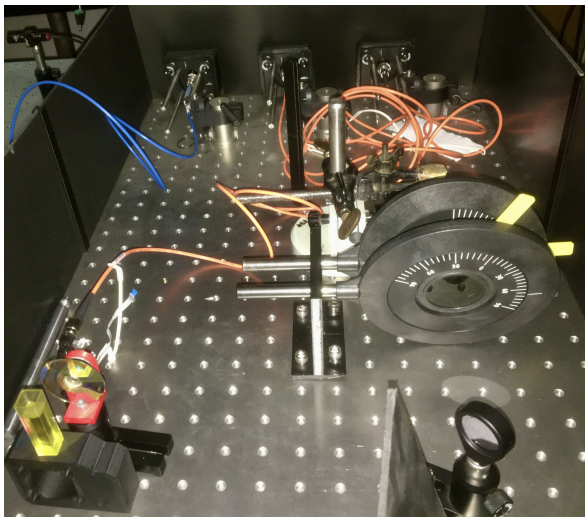
3. Kísérleti eszköz

A dolgozat célja egy TCSPC berendezés megépítése és működésének demonstrálása. Ellentétben a legtöbb kereskedelemben kapható rendszerrel, amelyek általában a gyors adatbegyűjtéstre helyezik a hangsúlyt, mi úgy terveztük a berendezésünket, hogy nagyon kis intenzitású fluoreszcens jeleket (alacsony koncentrációjú fluoreszcens oldatokat) is tudjon elemezni.

Egy standard TCSPC berendezés három fő részből áll: egy gerjesztő fényforrásból, egyfoton detektrokból és egy számlálóból (3.1. ábra).



3.1. ábra. A TCSPC elrendezés általános felépítése

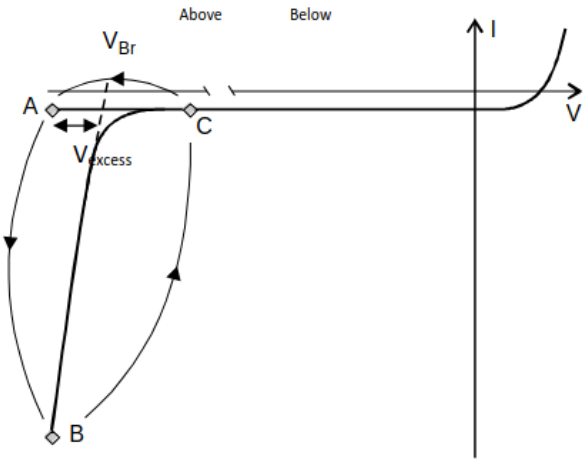


3.2. ábra. A 3.1 ábrának megfelelő optikai asztalon megépített berendezés

3.1. Detektor

A TCSPC berendezések detektorai, egyfoton detektorok. Ezek a detektorok nagy érzékenységükkel egyetlen foton érkezését is észlelni tudják. Jellemzően két típusú egyfoton detektorral találkozhatunk: fotoelektron-sokszorozó csöveket (photomultiplier tubes - PMT) vagy félvezető alapú detektorokat, lavina diódákat (avalanche photodiodes - APD) használnak.

A PMT-k működése a fotoelektromos hatáson alapul, ahol a fotonok egy fém felületébe csapódnak, és elektronokat ütnek ki. Ezeket az elektronokat a csőben levő nagy elektrosztatikus tér gyorsítja, majd több egymás után elhelyezett dinóda segítségével többszörösen felerősíti. Minden dinódánál az elektronok ütközése további elektronokat szabadít fel, így az eredeti foton által kiváltott elektronáram jelentősen megnő. Ez a többszörös erősítés teszi lehetővé, hogy



3.3. ábra. Az APD működési ciklusa[12]

a PMT-k egyetlen foton detektálására is alkalmasak legyenek. Az így keletkező elektromos impulzusokat a további elektronika dolgozza fel [2].

Az APD-k működése inkább hasonlít a Geiger-Müller csövekhez, innen az alternatív Geiger-módú lavinadiódák elnevezés. Az lavinadiódát előfeszített állapotban működtetik (bias voltage), a letörési feszültségnél valamivel nagyobb feszültséggel. A dióda egy metastabil állapotban (3.3. ábra A) van addig, amíg egy elsődleges töltéshordozó nem keletkezik. Ekkor elegendő egyetlen foton is ahhoz, hogy egy lavina-áramot indítson el, effektív végtelen erősítést biztosítva (3.3. ábra B). Ezt az makroszkópikus áramot a megfelelő elektronika képes detektálni. Ahhoz, hogy az eszközben ne szálljon el az áram - meghibásítva a diódát - a lavina-áramot le kell fojtani és a diódát ki kell üríteni a töltéshordozóktól. A lavina-áram lefojtására a diódát vissza kell kapcsolni normál működési állapotba, azaz a bias feszültséget le kell csökkenteni a letörési feszültség alá (3.3. ábra C). Ez a folyamat diódától függően néhány nanoszekundum, a kiürítés alatt nem képes fotonokat érzékelni, ez az idő a holtidő, ami a TCSPC mérésekben figyelmet igénylő tényező. Egy másik fontos tényezőt, a sötét detektálási rátát (dark noise) is figyelembe kell venni, ami nem fotonok, hanem véletlenszerűen keletkező töltéshordozók által okozott áram, és hamis pozitív detektálásokat eredményezhet. Az egyik legegyszerűbb módja a véletlenszerűen (termikus mozgásból) keletkező töltéshordozók számának csökkentésének, ha a detektort hűtjük [12].

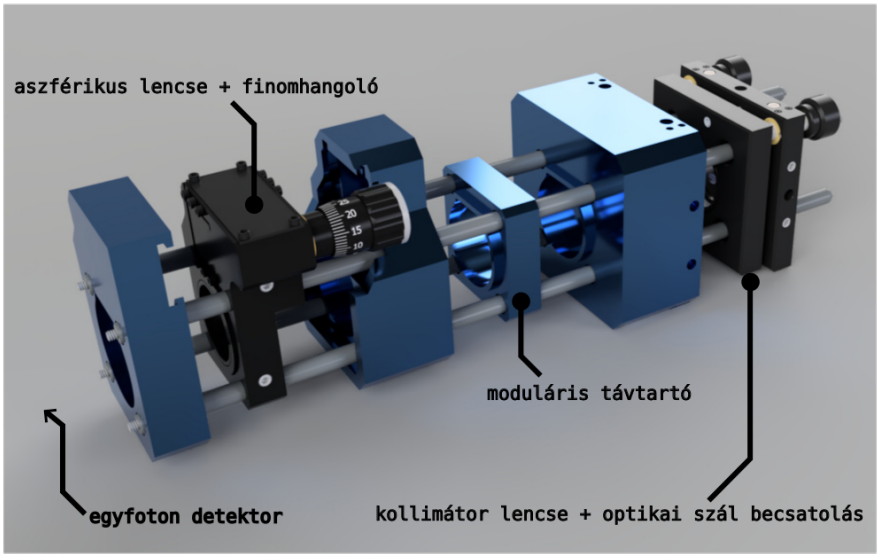
A PMT-k (fotoelektron-sokszorozó csövek) hagyományosan elterjedtek voltak a TCSPC mérésekben, mivel nagy érzékenységet és alacsony dark noise szintet kínálnak. Azonban idővel egyre inkább előtérbe kerültek a félvezető alapú lavina diódák (APD-k), mivel ezeknek több előnyük is van: kompaktabbak, alacsonyabb működési feszültséget igényelnek, nagyobb

kvantumhatásfokkal rendelkeznek a látható tartományban, valamint kevésbé érzékenyek a mágneses térre és mechanikai behatásokra. Emellett az APD-k időfelbontása is kedvezőbb lehet, ami különösen fontos a TCSPC alkalmazásokban, ahol a detektor időbeli válasza meghatározza a mérés pontosságát [9].

Az általunk használt detektor két ID Quantique *ID120-STD* típusú APD. Ez a modell főleg látható és közeli infravörös tartományban működik hatékonyan, $800nm$-nél éri el a maximás érzékelési hatásfokát (80%). Holtideje $1\mu s$, dark noise szintje pedig $1000Hz$ alatti. A detektorokat a beépített Peltier elemmel $-40°C$-ra hűtjük, hogy a dark noise szintet minimálisra csökkentsük [11].

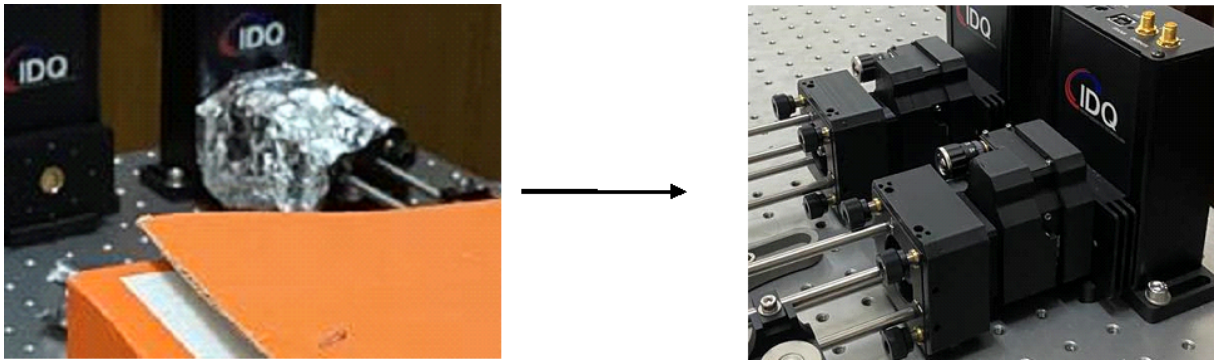
A fotonok az optikai szálból detektorba való irányítását egy optikai ketrec (optical cage) segítségével oldottuk meg. A ketrec négy rúdból álló keret, amin a különböző opto-mechanikai foglalatokban mindenféle optikai elemeket (lencsék, szűrők, detektorok) pozicionálhatjuk és rögzíthetjük. Az első elem egy kicsatoló rendszer(optical coupler), amely az optikai szálból kilépő széttartó sugarakat párhuzamos nyalábbá alakítja. A kollimátort tartalmazó keret orientációját három csavarral tudjuk megfelelően hangolni, annak érdekében, hogy a kollimátorból kilépő párhuzamos nyaláb, a cage közepén haladó egyenes mentén terjedjen. Ezeket követte egy aszférikus lencse, melynek feladata a párhuzamos nyalábot a detektor $500\mu m$ átmérőjű aktív felületére irányítani. Ennek csupán egy szabadsági foka volt: egy csavar a cage iránya mentén tudta előre-hátra tolni. A rendszerünk finombeállítását úgy folytattuk le, hogy az optikai szál szabad végét egy kis teljesítményű égő elé fogattuk, míg a másik felét becsatoltuk a cage-be. A detektor számítógépes interfészén figyelve a fotonszámot és a különböző csavarokat finoman mozgatva, maximumokat kerestünk. Közben oda kellett figyeljünk, hogy a fényforrás nem legyen túl erős, hiszen ha a detektorba túl sok foton jut be (szaturáció), akkor az általa mért fotonszám csökkeni kezd. Ezt figyelmen kívül hagyva könnyen megtörténhet az, hogy habár jó irányba mozdítottunk el egy csavart a detektor kimenetéből mégis azt látjuk, hogy a fotonszám csökken.

A korai hangolási kísérleteknél feltűnt, hogy mennyire befolyásolják a méréseket a kívülről beszűrődő fény, ezért ezen zajforrás minimalizálása érdekében több lépést is tettünk. Eleinte a rendszert csupán alufóliával takartuk be, utána árnyékoló burkot terveztünk hozzá *Fusion360* programban, majd PLA-ból kinyomtattuk. Továbbá azt is tapasztaltuk, hogy annyira érzékeny a detektor, hogy optikai szál mentén a palástján átszűrődő fényt is kijelzi, ezért az egész berendezés (beleértve detektorok, minta, optika, fényforrás) köré árnyékoló dobozt építettünk.



3.4. ábra. Optikai cage. A fény jobbról balra halad, a detektor irányába. A kék színű elemek az általunk tervezett és nyomtatott árnyékoló burok elemei.

Úgy a burkoló, mint a doboz tervezésekor és építésekor szem előtt tartottuk, hogy ezek modulárisak, a jövőben könnyen bővíthetők, más kísérletre átalakíthatóak legyenek.



3.5. ábra. Alufóliával majd árnyékoló burokkal árnyékolt detektorok

Az opto-mechanikai beállítások után a detektorok előfeszítését is be kellett állítanunk. Ezt a detektorokkal érkező szoftveren keresztül tudtuk elvégezni, USB-n keresztül csatlakoztatva a számítógéphez. A bias feszültséggel egyensúlyozni tudunk az érzékenység és alacsony dark noise szint között. Ez a feszültség a detektorokban levő félvezető geometriájától is függ, ugyanarra a bias feszültségre a különböző típusú detektorok más-más érzékenységet és dark noise szintet produkálnak. Úgy etalonáltuk a detektorokat, hogy egy fotonpár generátor egy-egy kimenetét a két detektorhoz csatlakoztattuk, majd szoftverrel úgy állítottuk be a feszültségeket, hogy a detektált fotonszámok megegyezzenek.

3.2. Fényforrás

A TCSPC berendezések általában egy kék vagy ibolyán túli lézeres vagy LED-es fényforrást használnak, amely képes gerjeszteni a fluoreszcens mintát. Olyan fényforrásra van szükségünk, aminek kikapcsolási karakterisztikus ideje nincs hasonló skálán a FP-ok élettartamával, valamint az általa generált fényipulzus alakja minél inkább közelít egy Dirac-deltához. A fluoreszcens fotonok élettartama általában néhány nanoszekundum, ezért az adatfeldolgozás szempontjából fontos, hoogy a gerjesztő impulzus ne legyen sokkal nagyobb időskálán.

Először LED-ekkel próbálkoztunk, ezek vezérlésére egy MOSFET tranzisztort használtunk, amit egy Raspberry Pi Pico mikrokontroller vezérelt. Ez a vezérlő megoldás nem bizonyult elég gyorsnak, így a mikrokontrollert egy jelgenerátorra (*Tektronix AFG 3252*) cseréltük a kapcsoló elektronikát meghagyva. Több UV LED-el kísérleteztünk, de a gyenge intenzitás és hosszú lecsengés jelentősen megnehezítette a FP élettartam mérését. A LED jelének hosszú lecsengése nem csak az elektron-lyuk rekombinációjának tudható be, hanem egyes LED-eknél a tokozat gyantája is fluoreszkált.

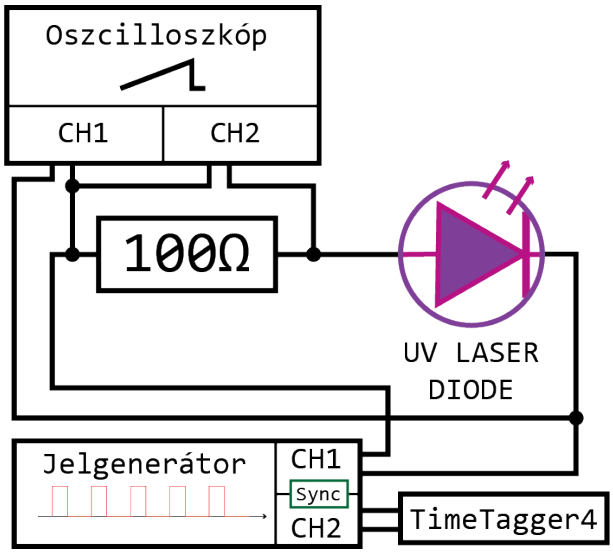
A kapcsoló elektronika nemkívánatos tranziens zajokat generált (csengés, visszaverődés), ezért a lézer ki-be kapcsolását a tűzőlapos, tranzisztoros vezérlés helyett egyszerűen a jelgenerátorra bíztuk. Ugyan a LED-es gerjesztés nem volt elegendően sikeres, de a gerjesztő impulzus alakja jelentősen javult és a jelgenerátor is képes volt elégséges elektromos teljesítményt szolgáltatni úgy a LED-hez, mint a lézerhez.

A berendezés végső fényforrása egy 405 nm-es hullámhosszú 20 mW teljesítményű lézerdióda, aminek a fényjele sokkal élesebb kikapcsolással rendelkezett, mint azt korábban a LED-eknék láttuk.

3.3. Számláló

TCSPC során nem csak számolnunk kell a fotonokat, hanem ezek érkezési idejéből eloszlást kell gyártsunk. A detektorok minden foton érzékelésekor a kimenetükön küldenek egy TTL jelet. A számláló feladata, hogy ezekhez a TTL jelekhez rendeljen egy időbélyeget.

Jelen esetben a számláló három csatornán fogad jeleket: a fényforrást meghajtó jelgenerátortól (A), a fluoreszcens ág detektorától (B) és a referencia ág detektorától (C). A számláló az *Trigger* csatornán érkező jel emelkedő élénél elindít egy belső számlálót, majd az *A* és *B* csatornákon érkező jelek emelkedő és ereszkedő éleinél is elmenti az időbélyeget (i.e. a belső számláló állását). Azért mentjük el mind az ereszkedő, mind az emelkedő éleket, hogy



3.6. ábra. Ahhoz, hogy a az elektromos gerjesztést és a diódán átfolyó áramot is mérhessük, a jelgenerátor jelét (párhuzamosan a lézerrel) valamint egy, a lézerrel sorba kapcsolt 100Ω-os ellenállás feszültségesését is mérjük. A jelgenerátor jelét egy 50Ω impedanciájú csatlakozón keresztül továbbítjuk. A lézer és az ellenállás közötti feszültségesést egy oszcilloszkóppal rögzítjük, amely lehetővé teszi a gerjesztő impulzus alakjának és időtartamának, valamint a dióda elektromos válaszának elemzését.

ellenőrizni tudjuk hogy teljes csomagokat kaptunk-e, azaz az *A* és *B* csatornákon érkező jelek közül az egyik sem veszett el.

A berendezésünkben a számláló egy *Cronologic TimeTagger4*[4] típusú kártya, amit egy laboratóriumi számítógép PCIe foglalatába illesztettünk. A kártya rendelkezik négy all-purpose és egy külön trigger csatornával, mi csak hármat használtunk (trigger, A, B). Az időbélyegek kártyáról való kimentéséért felelős programot C/C++ programozási nyelven írtuk a gyártó által biztosított példaprogramból kiindulva. A program létrehoz mindkét detektor csatornára egy-egy előre definiált hosszúságú tömböt, amely mindegyik eleméhez rendeltünk egy $500ps$ hosszúságú időablakot. Minden detektálási eseménykor a program azonosítja az időbélyegnek megfelelő indexű elemet, majd az értékét növeli eggyel. Az adatok kimentésekor csatornánként még vizsgáltuk az első beérkező fotonok időbélyegét. Ezt összehasonlítva az összes adott csatornán bejövő jellel ellenőrizni tudjuk, hogy a detektorok túlcsordulása miatt nem vesztettünk-e el adatokat. Ideális esetben, amikor gerjesztésenként legtöbb egy fotont fogunk be az optikai szállal, a két eloszlás azonos kell legyen.

3.4. Optikai elrendezés

Az optikai elrendezés a fény útját hivatott megszabni, a gerjesztő fényforrástól a detektorig. A lézer fényét egy nyalábosztó két irányba osztja, ahogy ez a 3.1 ábrán szaggatott vonallal bekeretezett részen látható: egyik a fluoreszcens ág, a másik a referencia ág.

A fluoreszcens ágban a nyaláb a kémcsőben tárolt fluoreszcens oldatra irányul. Fontos, hogy a mintát tároló kémcső anyaga átlátszó legyen a gerjesztő fény számára, és ne fluoreszkáljon. A kémcsőhöz oldalirányból, tartjuk oda az optika szálat, ami a FP-okat a detektorhoz továbbítja. A kémcső és az optikai szál közé egy piros tartományban áteresztő szűrőt ($\lambda < 600nm$) rakunk, hogy a gerjesztő nyalábból kiszóródó fotonokat is kiszűrjük. Ezt a szűrőt ha egy piezoval mozgatható diffrakciós rácsra cseréljük, akkor hullámhossz szertinti felboltást is mérhetünk, de ez a jelen berendezésben nem volt cél.

A referencia ágban a nyaláb előbb egy polarizátor-analizátor szűrőn halad át, majd egy gyűjtőlencsés becsatolón és egy optikai szálon keresztül a második detektorhoz jut. A polarizátor-analizátor szűrőre azért van szükség, hogy a referencia jelet mérő detektort ne szaturáljuk, mert így hamis referencia fényjelet kapnánk.

4. Adatfeldolgozás

4.1. Konvolúció

Legyen $f_m(t)$ a mért fényjel, ami a fluoreszcens fotonok érkezési valószínűségi eloszlását írja le idő függvényében, feladatunk viszont a vizsgált fluoreszcens molekula fotonkibocstásának időbeli eloszlását megadó $f(t)$ függvény meghatározása. A két jel között azért van különbség, mert a gerjesztő fényjel ($IRF(t)$ - Instrument Response Function) nem egy Dirac-delta szerű, hanem egy időben kiterjedt, bizonyos formával rendelkező impulzus.

Egy $t = 0$ időpillanatban alkalmazott Dirac-delta szerű gerjesztés esetén a fluoreszcens molekula időbeli viselkedése az alábbi exponenciális lecsengésű függvénnyel írható le:

$$f(t) = I_0 \cdot e^{-\frac{t}{\tau}}, \tag{4}$$

Abban az esetben, a gerjesztés a t' időpontban történik, akkor a fluoreszcens fotonok időbeli eloszlása módosul:

$$f(t) = \begin{cases} I_0 e^{-\frac{t-t'}{\tau}} & t > t' \\ 0 & t < t' \end{cases}. \tag{5}$$

Nem ideális gerjesztés esetén az $IRF(t)$-el jellemmzett gerjesztő impulzus felbontható dt‘ hosszúságú diszkrét szakaszokra. Kellően kis dt‘ esetén minden egyes szakasz egy különálló Dirac-delta gerjesztésnek tekinthető. A $(t‘, t‘ + dt‘)$ indőintervallumhoz rendelhető gerjesztés hozzájárulása a t időpontban detektált fluoreszcencia fotonszámhoz:

$$dF_m(t) = C \cdot IRF(t') \cdot I_0 \cdot e^{-\frac{t-t'}{\tau}} dt', \tag{6}$$

ahol a C együttható tartalmazza a gerjesztőimpulzus amplitúdóját és a fluoreszcencia folyamat Φ kvantumhatékonyságát. Összegezve az összes szakas hozzájárulását a következő összefüggéshez jutunk:

$$F_m(t) = C \int_0^\infty I_0 \cdot IRF(t') \cdot e^{-\frac{t-t'}{\tau}} dt' = C \int_{-\infty}^\infty IRF(t') \cdot f(t-t') dt', \tag{7}$$

amit normálva

$$f_m(t) = C \int_{-\infty}^\infty IRF(t') \cdot f(t-t') dt' \tag{8}$$

megkaptuk a kísérletileg mért fotoneloszlást. Tehát a mért fényjel az exponenciális lecsengés és a gerjesztő fényjel konvolúciójaként írható fel:

$$f_m(t) = f(t) * IRF(t). \tag{9}$$

4.2. Dekonvolúció

Ahhoz hogy a fluoreszcens molekula időbeli viselkedését megkapjuk, a mért fényjelet dekonvolúcióval kell feldolgozni. Tudjuk azt, hogy két függvény konvolúciójának Fourier-transzformáltja a két függvény Fourier-transzformáltjának szorzataként írható fel:

$$\mathcal{F}\{f(t) * g(t)\} = \mathcal{F}\{f(t)\} \cdot \mathcal{F}\{g(t)\}, \tag{10}$$

ahol $\mathcal{F}\{f(t)\}$ a $f(t)$ függvény Fourier-transzformáltja. Valamelyik függvényt tehát inverz Fourier transzformációval ($\mathcal{F}^{-1}$) úgy tudjuk visszafejteni, hogy:

$$f(t) = \mathcal{F}^{-1}\left\{\frac{\mathcal{F}\{f(t) * g(t)\}}{\mathcal{F}\{g(t)\}}\right\}. \tag{11}$$

Tehát a fluoreszcens molekula időbeli viselkedését a következőképpen tudjuk meghatározni:

$$f(t) = \mathcal{F}^{-1}\left\{\frac{\mathcal{F}\{f_m(t)\}}{\mathcal{F}\{IRF(t)\}}\right\}. \tag{12}$$

A dekonvolúciónál gyakran előfordul, hogy a Fourier transzformálandó fügvények zaja miatt, a dekonvolúció során még zajosabb eredményt kapunk. A zaj csökkentésére Wiener szűrőt használunk, amely a következőképpen alklamazunk:

$$f_W = \mathcal{F}^{-1}\left\{\frac{||\mathcal{F}\{IRF(t)\}||}{||\mathcal{F}\{IRF(t)\}|| + \alpha}\frac{\mathcal{F}\{f_m(t)\}}{\mathcal{F}\{IRF(t)\}}\right\}, \tag{13}$$

ahol α egy manuálisan beállított konstans.

4.3. Rekonvolúció

A rekonvolúció egy alternatív módszer a fluoreszcens élettartam meghatározására. A dekonvolúcióval ellentétben itt indirekt módon határozzuk meg a fluoreszcens molekula időbeli viselkedését [13].

Első lépésben feltételezünk egy paraméteres modellt, például egy exponenciális függvények lineáris kombinációját:

$$f^{modell}(t;\bar{\tau}_i,\bar{c}_i) = \sum_{i=1}^{n} c_i e^{-\frac{t}{\tau_i}}, \tag{14}$$

ahol c_i az i-edik exponenciális függvény amplitúdója, τ_i pedig az i-edik exponenciális lecsengés karakterisztikus ideje.

Ezt követően kiszámítjuk a modell és a mért gerjesztő fényjel konvolúcióját, megkapva a modell szerint jósolt fluoreszcens fényjelet:

$$f_m^{modell}(t;\bar{\tau}_i,\bar{c}_i) = f^{modell}(t;\bar{\tau}_i,\bar{c}_i) * IRF(t). \tag{15}$$

Utolsó lépésben a modell és a mért fényjel közötti eltérést minimalizáljuk, például a legkisebb négyzetek módszerével.

4.4. A fényjel zajának szűrése

A mért fényjelben zaj is jelen van, ez a beszűrődő nem kívánatos fotonokból és a detektor termikus zajából adódik. Ezt úgy tudjuk lekezelni, hogy a gerjesztést kikapcsolva futtatva a mérést meghatározzuk az alapzajt (háttérzajt) mindkét ágra külön, amit a fényjelekből kivonunk.

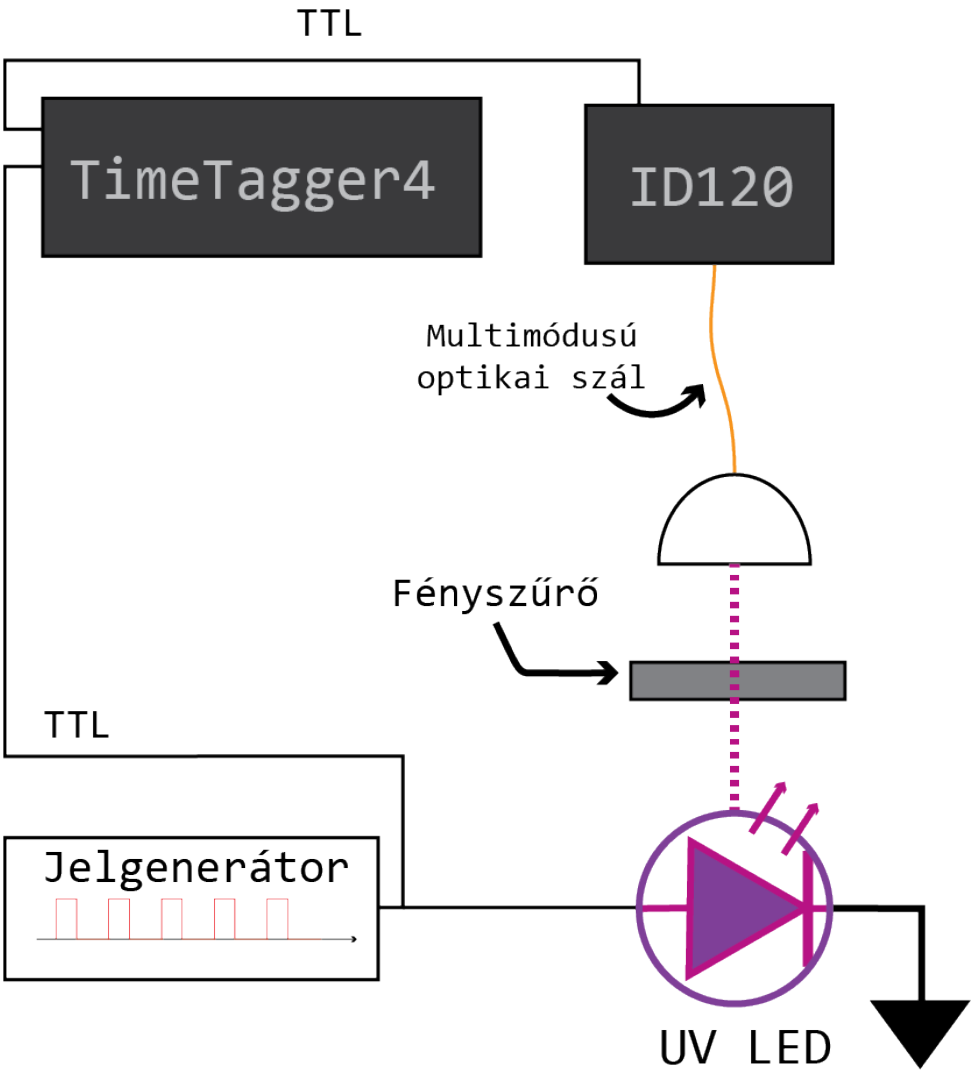
A statisztikai zaj mérsékelésére mozgóátlagot használunk. Ez adott időablakban (w) lévő értékek átlagolását jelenti, így csökkentve a véletlenszerű zaj hatását. A mozgóátlagot a következőképpen számíthatjuk ki:

$$\bar{x}_i = \frac{1}{2w}\Sigma_{i-w}^{i+w} x_i \tag{16}$$

Az elemzés során az exponenciális kitevőket keressük, ezért tudnunk kell, hogy a mozgóátlag nem módosítja a kitevőket:

$$\begin{aligned}
\bar{I} &= \frac{1}{\Delta t}\int_{t-\frac{1}{2}\Delta t}^{t+\frac{1}{2}\Delta t} I_0 e^{-\frac{t}{\tau}} dt = -\frac{\tau}{\Delta t} I_0 \, e^{-\frac{t}{\tau}}\Big|_{t-\frac{1}{2}\Delta t}^{t+\frac{1}{2}\Delta t} \\
&= -\frac{\tau}{\Delta t} I_0 \left[e^{-\frac{t}{\tau}-\frac{\Delta t}{2\tau}} - e^{-\frac{t}{\tau}+\frac{\Delta t}{2\tau}} \right] \\
&= -\frac{\tau}{\Delta t} I_0 \left[e^{-\frac{\Delta t}{2\tau}} - e^{+\frac{\Delta t}{2\tau}} \right] \cdot e^{-\frac{t}{\tau}} \\
&= C \cdot e^{-\frac{t}{\tau}},
\end{aligned} \tag{17}$$

tehát csak egy arányossági tényezőben változik.

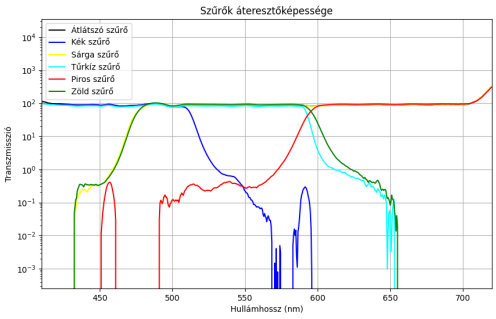


5.1. ábra. Berendezés felépítése a gerjesztő impulzus alakjának meghatározásához.

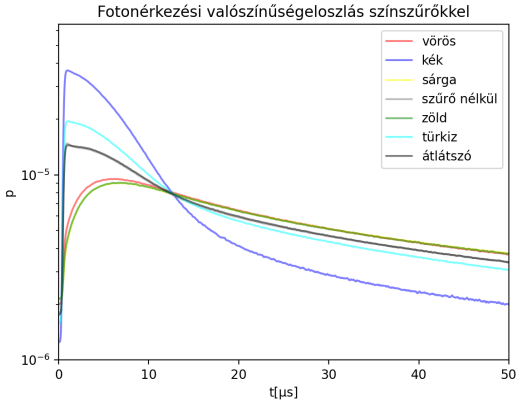
5. Gerjesztő impulzus alakjának meghatározása

A fluoreszcens viselkedés meghatározásához elengedhetetlen a gerjesztő impulzus alakjának ismerete. Ennek már akkor nekiláttunk amikor még LED-ekkel dolgoztunk. A berendezésünkben csak egyik ágat nézzük, a másikat nem használjuk, ahogy az a 5.1. ábrán is látható.

Először egy egyszerű UV LED-del próbálkoztunk. A vezérlő feszültség kikapcsolása után furcsán hosszú lecsengést tapasztaltunk, jóval hosszabbat, mint tipikus kisebbségi töltéshordozó élettartam. A jelalak első ránézésre egy gyors lecsengés után következő plató-szerű lecsengésből állt, ezek forrásaként két külöböző jelenségre gyanakodtunk. Hogy megnézzük, mi lehet a különböző lecsengések okozója, a LED és az optikai szál közé színszűrőket tettünk. Ezen mérések kiértékelésekor úgy tekintettük, hogy a gerjesztő jelalak



5.2. ábra. A színszűrők áteresztőképesség.



5.3. ábra. Az epoxygyanta tokozatú UV LED fényjele különböző színszűrőkkel.

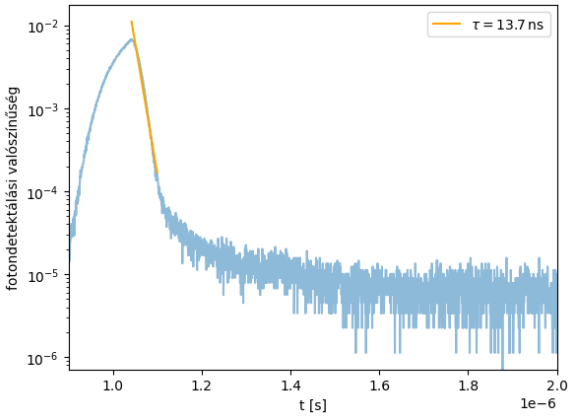
Dirac-delta szerű, így nem kellett dekonvolúcióval foglalkoznunk. Továbbá a fluoreszcens jel maximum utáni részét exponenciális függvények összegével közelítettük:

$$P_{modell}(t) = C_z + \sum_{0}^{N_{exp}} C_i e^{-\frac{t}{\tau_i}}, \tag{18}$$

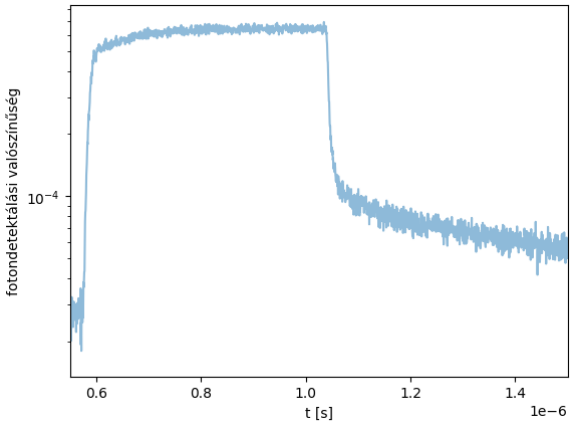
ahol C_z a háttérzaj, C_i az egyes exponenciális tagok amplitúdója, τ_i pedig az egyes exponenciális tagok élettartama. A lassú lecsengés a hosszabb hullámhosszú tartományokra jellemző és sok komponensű ($10 \sim 120\mu s$), míg a rövidebb hullámhosszú tartományokban a domináns élettartam konszisztensen $2\mu s$ körüli (5.3. ábra). A LED által kibocsátott fotonok $400nm$ körüliek, a gyors lecsengést a LED-ben kikapcsolás utáni rekombinációs folyamatnak tulajdonítottuk, míg a lassú lecsengést összetettsége és nagy hullámhossz tartománya miatt a LED epoxy tokozatának fluoreszcens viselkedésének.

Hogy ennek a magyarázatnak bizonyosságát erősítsük, a hétköznapi UV LED-et egy kvarc tokozatú UV LED-re cseréltük. Tudva azt, hogy a kvarc tokozat nem fluoreszkál, a méréstől azt várjuk, hogy a gyors lecsengés legyen domináns, míg a lassú lecsengés közel eltűnjön. A kvarc tokozatú LED esetében a gyors lecsengés valóban dominált, ugyan az epoxy tokozatú LED-nél jóval kisebb $\sim 14ns$ élettartammal (5.4. ábra). Az élettartamok közötti különbség a LED-ek különböző parazitakapacitásából adódhat, de ezt nem vizsgáltuk mélyremenően.

A berendezésünk szempontjából fontos, hogy a gerjesztő jel ne tartalmazzon hasonló vagy nagyobb élettartamú komponenseket, mint a vizsgálandó minta, mert megnehezíti, ha nem pont ellehetetleníti a vizsgálni kívánt fluoreszcens jelet (elkülöníthetetlenné teszi a nemkívánatos és vizsgálandó lecsengéseket). Ez a probléma kiküszöbölhető lett volna a gerjesztő fényjel



5.4. ábra. A kvarc tokozatú UV LED fényjele 500ns hosszúságú elektromos gerjesztéssel.

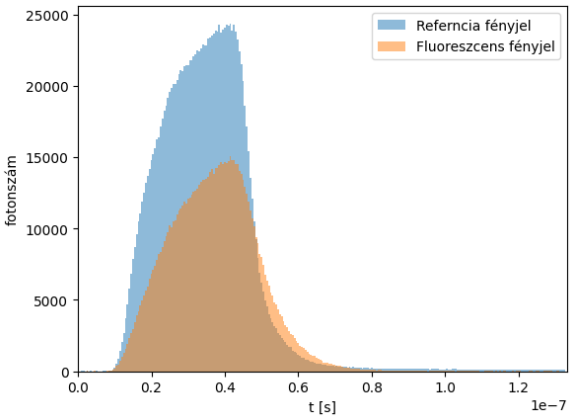


5.5. ábra. 500ns hosszúságú elektromos pulzus által vezérelt lézer fényjele.

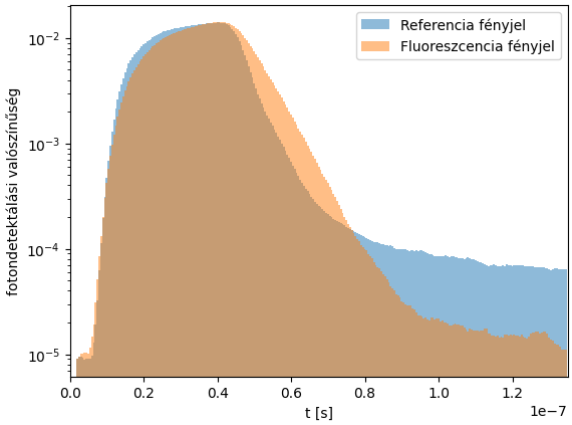
szűrésével, az epoxy tokozat esetében, de ahogy azt a kvarcled esetében láthattuk, a gyorsabbik lecsengés élettartama sem ideális a ns nagyságrendű fluoreszcens élettartamok vizsgálatához.

A gyorsabb ki és bekapcsolás érdekében a lézeres gerjesztést választottunk. Ugyanolyan $5V$ amplitúdójú $500ns$ hosszúságú impulzussal gerjesztve sokkal jobb válaszideje van (5.5. ábra).

A jobb válaszideje miatt, a gerjesztés hosszúságát egy egész nagyságrenddel le tudtuk csökkenteni. A lézeres gerjesztésnél is megjelenik egy lassú lecsengés ami nem fluoreszcenciából adódik, hanem feltehetően az optikai ketrecben használt kollimátor és aszférikus lecséktől. Ezek a lencsék infravörös és látható fényre vannak optimalizálva (felületük $810nm$ hullámszosszúságú fényre tervezett tükrözésgátló réteggel van bevonva), így ultraibolya tartományban nagy visszaverőképeséggel rendelkeznek, így egy rezonátorként működik. Ezt kikuszöbölhető lenne, ha a mostani lencséket az UV tarományban nagy áteresztőképességű modellekre cserélnénk, de jelenleg ezek nem állnak rendelkezésünkre.



6.1. ábra. A fluoreszcens és referencia jel nyers adatai. A láthatóság kedvéért a gerjesztési periódus $1/50$-ed részét mutatjuk



6.2. ábra. A fluoreszcens és referencia jel normált mozgóátlagolt adatai, logaritmikus skálán.

6. Klorofilloldat fluoreszcenciaidejének vizsgálata

A berendezésünk működésének és a kiértékelő eszközeink alkalmazhatóságának demonstrálására, klorofilloldat fluoreszcens élettartamának meghatározását választottuk.

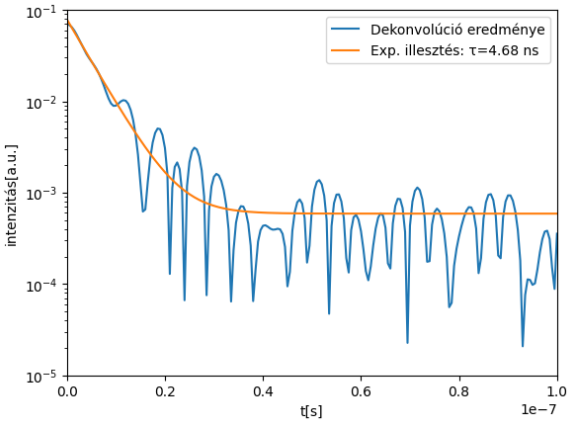
A méréshez a 3.1 ábrán feltüntetett elrendezést használtuk, lézer gerjesztéssel. A klorofillt cikória levélből nyertük ki, tömény metilalkoholban hagyva ázni. Mielőtt a méréshez használtuk volna, a klorofillt hígítottuk, hogy a fluoreszcens fotonok kevésbé reabszorálódjanak, ezzel növelve az optikai szálba jutó fotonok számát.

A jelgenerátort $150kHz$ gerjesztési frekvenciájú, $5V$ amplitúdójú, $50ns$ hosszú négyszögjelre állítottuk.

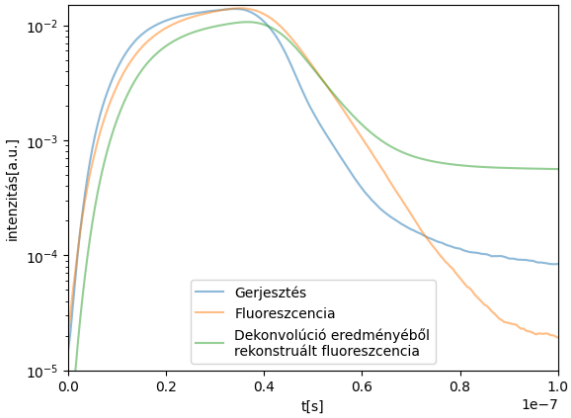
A fényjel jó meghatározásához a referenciaágban elegendő volt 10^8 gerjesztési ciklust mérni, a fluoreszcens ágon viszont az alacsonyabb intenzitás miatt 10^9 ciklust kellett mérni.

Az lenne az elvárásunk, a fluoreszcens jel konvolúciós természete miatt, hogy az a gerjesztő jelhez képest egy időbeli késéssel rendelkezzen, ami a 6.1 és 6.2 ábrák alapján teljesül is, viszont a két jel összehasonlításakor azt is észrevettük, hogy a maximum után a fluoreszcens ág válasza gyorsabban lecsengett, a gerjesztő jelhez képest, ami nem magyarázható konvolúció segítségével. Ahogy ezt a 5. fejezetben említettük ez nagy valószínűséggel a cage-ben levő lencsék bevonata által létrehozott rezonátorba csapdázott fotonoknak tulajdonítható. Ezt a jelenséget a jelenlegi berendezésünkkel képtelenek vagyunk kiküszöbölni, de a jövőbeli fejlesztések során érdemes lesz figyelembe venni. Az adatfeldolgozás során ezt úgy kezeltük, hogy eldobtuk a gerjesztő jelünk azon szakaszát, ahol ez a rezonáns zaj dominált.

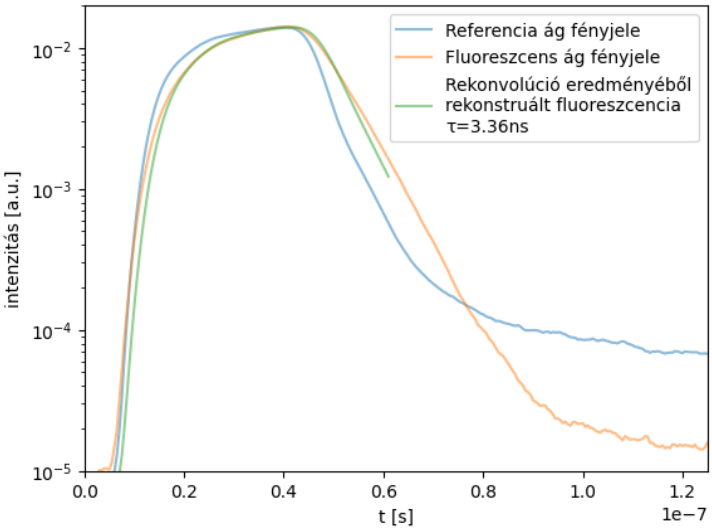
A dekonvolúciónál a Wiener-szűrés nem eredményezett érdemi javulást, ezért mellőztük



6.3. ábra. A dekonvolúció eredménye



6.4. ábra. Dekonvolúció eredménye alapján rekonstruált fluoreszcens jel és mért fényjelek

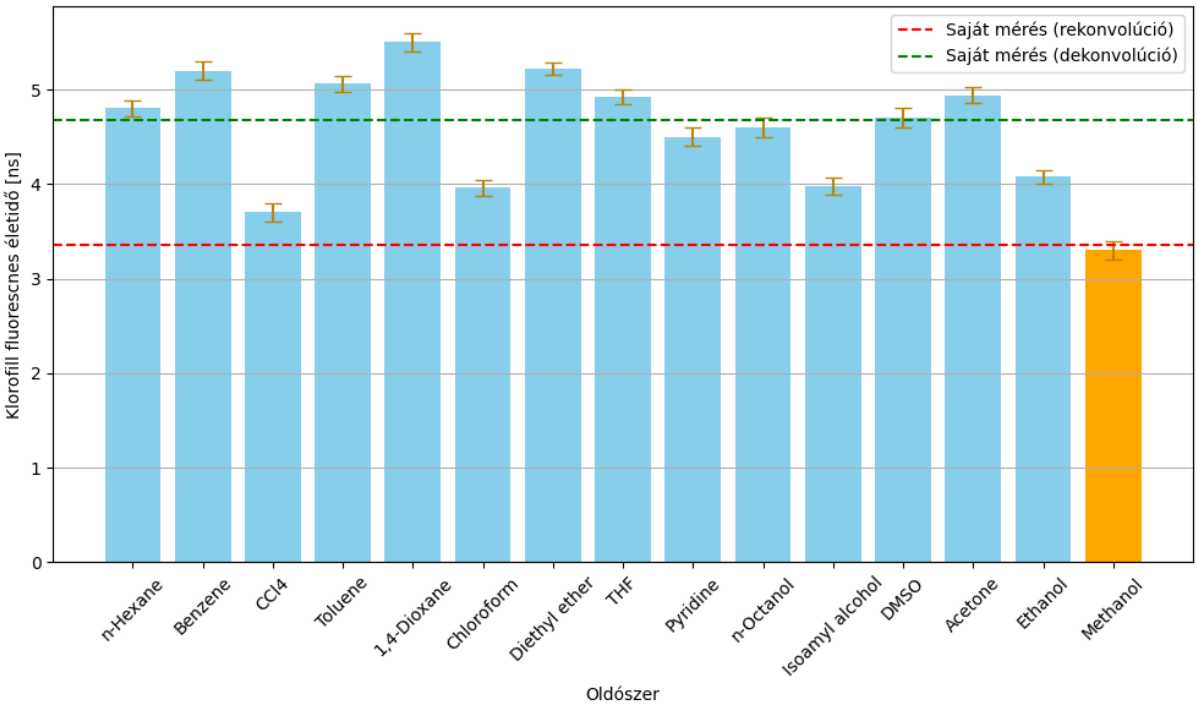


6.5. ábra. A rekonvolúció eredményével rekonstruált fluoreszcens jel és mért fényjelek

alkalmazását. Az így kapott dekonvoluált jel nagyon zajos volt, de így is megkíséreltünk exponenciális illesztést végezni egy $A\exp(-\frac{t}{\tau}) + B$ alakú modellfüggvénnyel (6.3. ábra). A kapott élettartam habár a várható nagyságrendi tartományba esik, nem tartjuk túl megbízhatónak a dekonvoluált jel zajossága és a modellfüggvény erőltetettsége (B paraméter) miatt.

A rekonvolúciót a 4. fejezetben bemutatott módszerrel végeztük el, a 6.5. ábrán látható eredménnyel.

Az eredményeket összehasonlítva a szakirodalomban található értékekkel jó tartományban vagyunk, a klorofill fluoreszcens élettartama $3-7ns$ között mozog függően a koncentrációtól[8] az oldószertől[10], és a klorofill típusától (a vagy b). Az eredmények jó nagyságrendűek úgy a dekonvolúció ($\tau_D = 4.68ns$), mint a rekonvolúció ($\tau_R = 3.36ns$) módszerrel, de a rekonvolúció eredménye megbízhatóbb és jó egyezést mutat a szakirodalomban található



6.6. ábra. A saját mérés (metanolban oldott klorofill) eredményének összehasonlítása szakirodalommal[10]

$3.3 \pm 0.1ns$ értékkel[8, 10] metilalkoholban.

7. Összegzés és jövőbeli kilátások

A dolgozatban egy időkorrelált egyfotonszámláló (TCSPC) berendezés működését, megépítését, hozzá társuló adatfeldolgozási módszereket és működésének demonstrálását mutattuk be. A rendszer tervezése során elsődleges tervezési szempontként jelent meg a gyenge, alacsony koncentrációjú mintákból származó fluoreszcens jelek detektálása, és a berendezés modularitása, hogy a jövőbeli fejlesztések során könnyen bővíthető legyen.

A megépített berendezés három fő részből áll: a gerjesztő fényforrásból, a detektorokból és az jelszámláló elektronikából. A megfelelő gerjesző fényforrás kialakításához több megoldást is kipróbáltunk (UV LED, lézerdióda), végül a $405nm$-es ézerdióda megfelelő válaszidőt és intenzitást biztosított. Detektoraink az ID Quantique ID120-STD SPAD típusai voltak, ezek beállításakor fölöttébb nagy figyelmet fordítottunk a zajszint minimalizálására, a detektorokat hűtve $-40°C$-ra és egy saját tervezésű árnyékolóval. A detektorok által küldött TTL szintű digitális impulzusok feldolgozására egy Cronologic TimeTagger4 számlálót használtunk. Ennek vezérlésére a gyártó által biztosított példaprogramból kiindulva saját C/C++ programot írtunk, amely a számláló által küldött adatokat feldolgozta, és a megfelelő formátumban elmentette.

A mért statisztikák kiértékelését de- és rekonvolúciós módszerekkel végeztük el. Tapasztalatunk alapján a rekonvolúció megbízhatóbb eredményeket ad, mint a dekonvolúció, mivel az utóbbi során a mérési zaj jelentősen torzította az eredményt.

A berendezés és adatfeldolgozó programunk működését klorofilloldat fluoreszcens élettartamának meghatározásával demonstráltuk. A kapott eredményeket a szakirodalomban található értékekkel összehasonlítva nagyságrendileg helyesek, a klorofill fluoreszcens élettartama $3-7ns$ között mozog, a kapott eredmények pedig $4.68ns$ dekonvolúcióval és $3.36ns$ rekonvolúcióval. A rekonvolúció eredménye megbízhatóbb, és talál a szakirodalomban található $3.3 \pm 0.1ns$ értékkel metil-alkoholban.

A berendezésünk és a hozzá tartozó adatfeldolgozó programunk jól működik, de a jövőbeli fejlesztések során érdemes lenne figyelembe venni a referenciaágon fellépő fotoncsapdázás miatti jeltorzulást, ami megoldható ebben az ág optikai cage-ében levő lencsék cseréjével. Továbbá a berendezés használatát lehetővé tenni azáltal, hogy a felhasználóbarátabb, könnyebben konfigurálható, első lépésben konzolos, később grafikus felhasználói felületet készítünk a vezérlőprogramhoz. Érdemes lenne a fluoreszcens minta kémcsövének rögzítését szabványosítani, megelőzve a minták elmozdulását, esetleg berendezésre való kiömlését figyelmetlenség miatt.

8. Függelék

8.1. Alkatrészlista

Alkatrész	**Megj.**
Tektronix AFG 3252 Arbitrary function generator Tektronix DPO 3032 Oscilloscope	
Ampul laser diode, 405nm, 20mW, diameter 5.6mm Thorlabs Kinematic Mirror Mount for 1" Optics with Post-Centered Front Plate, M4 Taps (KM100CP/M)	lézer rögzítéséhez + 3DP adapter
Thorlabs EBP1 - Economy 30:70 Beamsplitter, Ø1", AOI: 45° Thorlabs LMR1.5/M - Lens Mount with Retaining Ring for Ø1.5" Optics, M4 Tap	
LEYBOLD 472 401 Polarization filter	2 pcs
Band color filter from a projector disassembly	
Thorlabs M42L02 - Ø50 µm, 0.22 NA, Low OH, FC/PC-FC/PC Fiber Patch Cable, 2 m Long	2 pcs
Thorlabs Cage Assembly Rod, 8" Long, Ø6 mm, 4 Pack (ER8-P4) Thorlabs Ø25.0 mm Pedestal Pillar Post, M6 Taps, L = 50 mm (RS2P/M) Thorlabs Clamping Fork, 1.24" Counterbored Slot, Universal (CF125)	1 pcs/SPAD
Thorlabs Kinematic, SM1-Threaded, 30 mm-Cage-Compatible Mount for Ø1" Optic, Metric (KC1T/M) Thorlabs SM1-Threaded Adapter for Ø11 mm, >0.35" (8.9 mm) Long Cylindrical Components (AD11F) Thorlabs 780 nm, f = 11.07 mm, NA = 0.26 FC/PC Fiber Collimation Pkg	1 pcs/SPAD (összesen 3)
Thorlabs Z-Axis Translation Mount, 30 mm Cage Compatible (SM1ZA) Thorlabs SM1 to M9 x 0.5 Lens Cell Adapter (S1TM09) Thorlabs f = 11.00 mm, NA = 0.26, WD = 6.91 mm, Mounted Aspheric Lens, ARC: 650 - 1050 nm (A220TM-B)	1 pcs/SPAD
ID Qunatique ID120 Single Photon Avalanche Diode (SPAD)	2 pcs
Cronologic Timetagger4 Time-to-Digital converter	

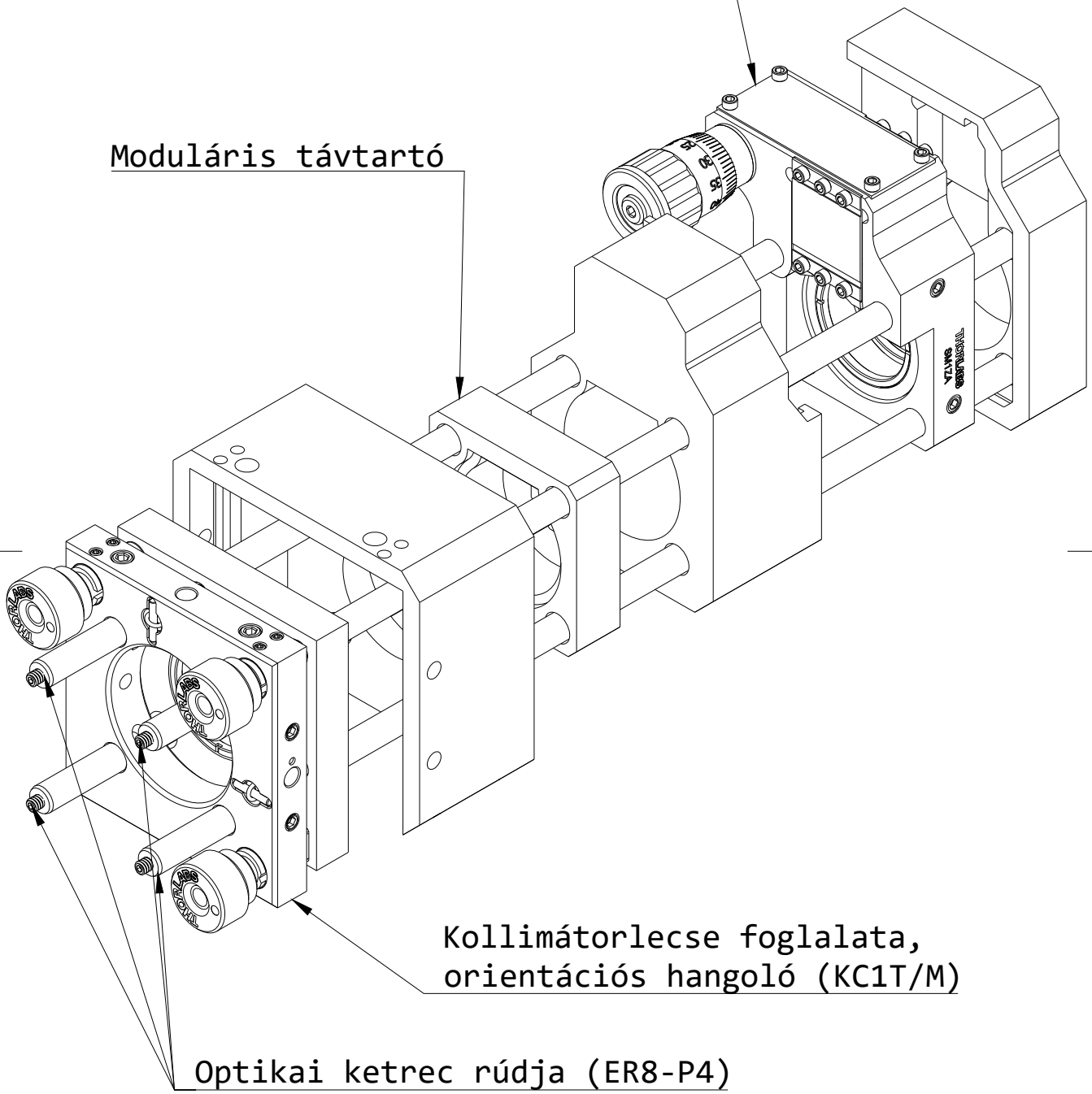
1. táblázat. Főbb alkatrészek és mérőeszközök listája

8.2. Tervrajzok

A következő oldalakon megtalálható az optikai cage-et árnyékoló alkatelemek rajzai:

Aszférikus lencse foglalata,
transzlációs finomhangoló (SM1ZA)

Moduláris távtartó



Kollimátorlecse foglalata,
orientációs hangoló (KC1T/M)

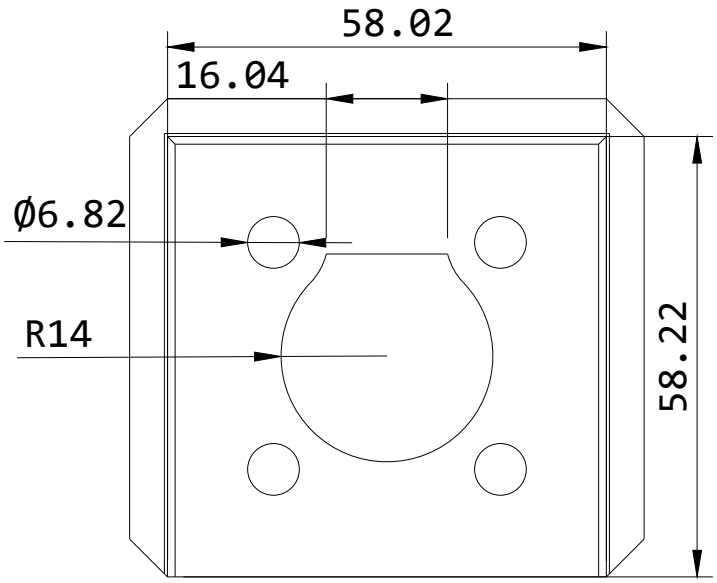
Optikai ketrec rúdja (ER8-P4)

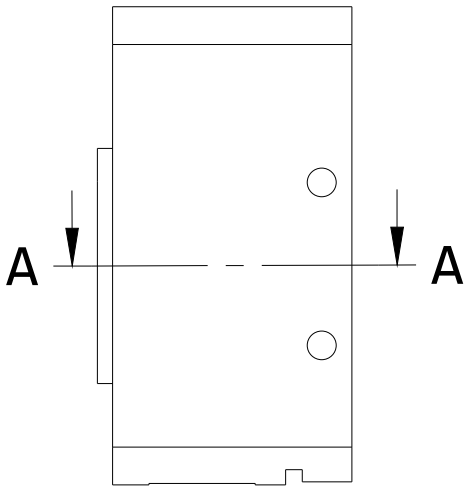
PHYSICA
Universitas Napocensis Babeș-Bolyai

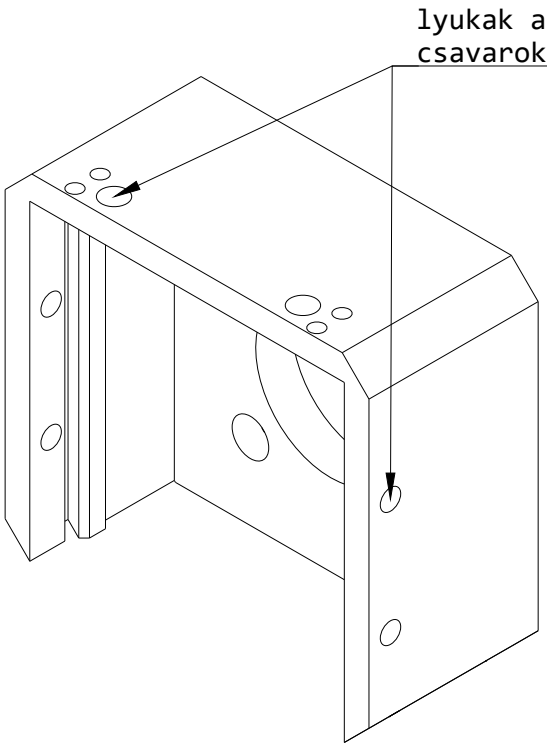
Alkatelem

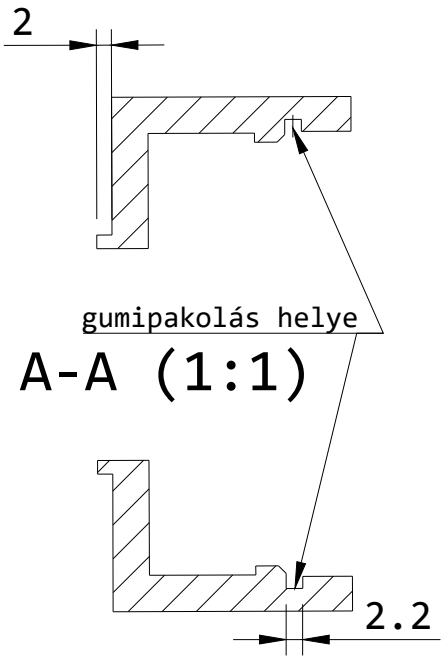
SPAD optical cage árnyékolás

	mm	Oldal 1/5	Tervezte Csiszér Csanád







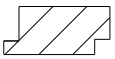


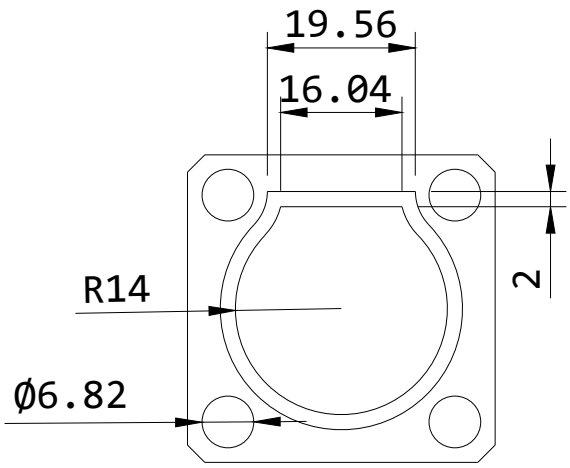
Alkatelem

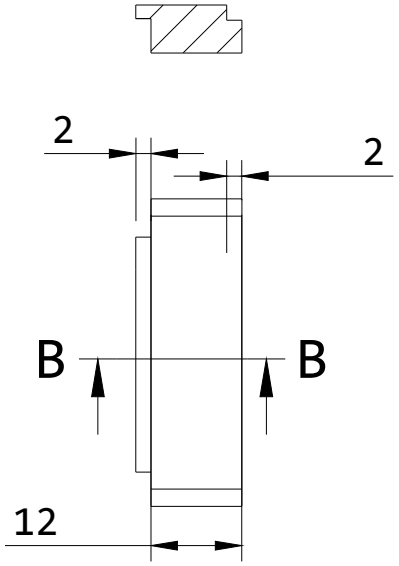
SPAD optical cage árnyékolás

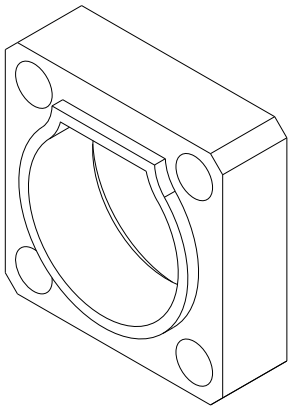
	mm	Oldal 2/5	Tervezte Csiszér Csanád

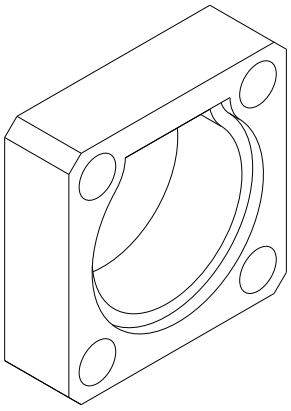
B-B
(1:1)







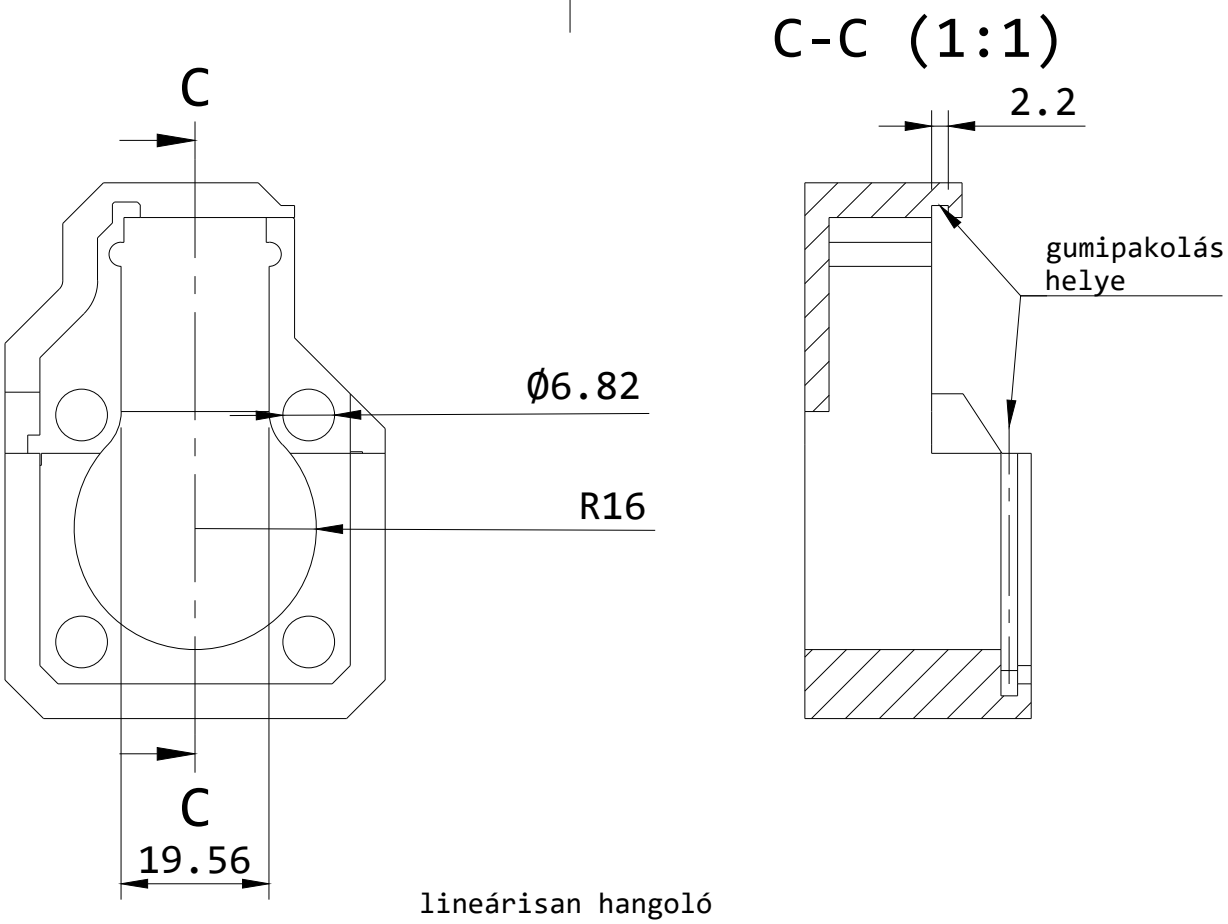


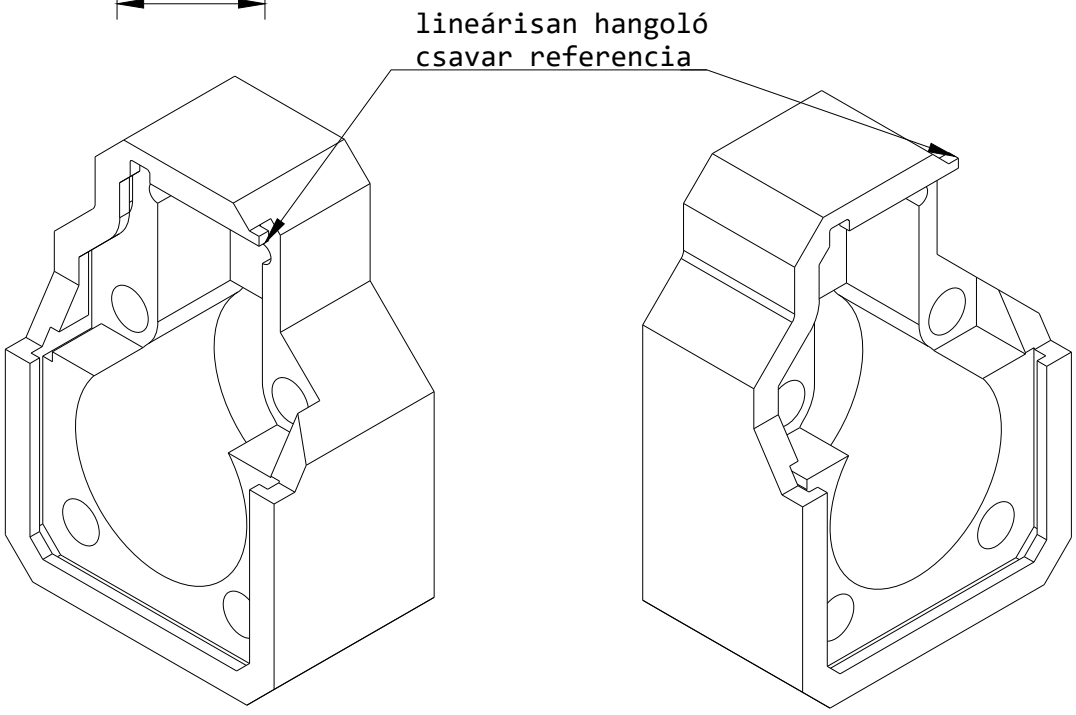


Alkatelem

SPAD optical cage árnyékolás

	mm	Oldal 3/5	Tervezte Csiszér Csanád

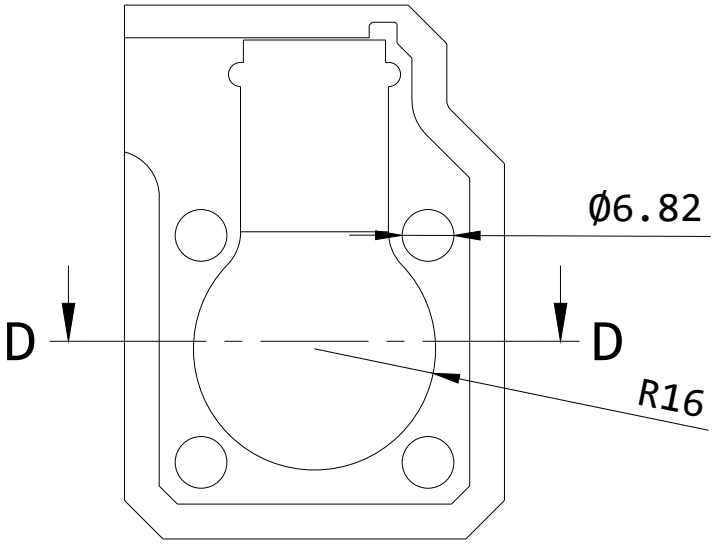


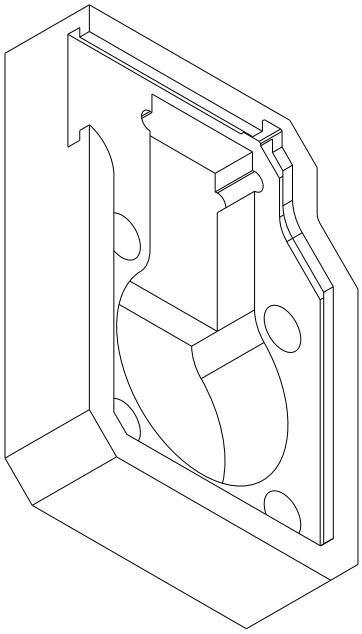


Alkatelem

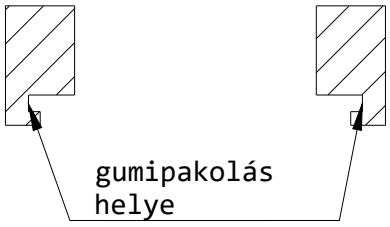
SPAD optical cage árnyékolás

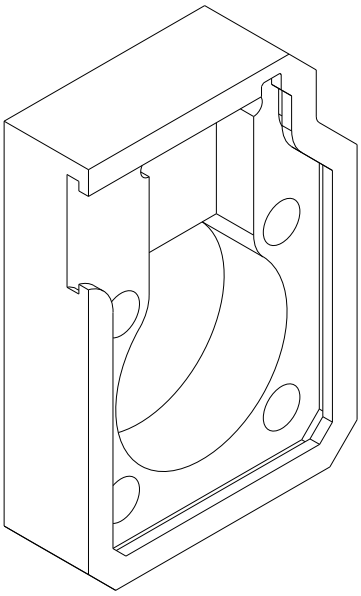
	mm	Oldal 4/5	Tervezte Csiszér Csanád





D-D (1:1)





Alkatelem

SPAD optical cage árnyékolás

	mm	Oldal 5/5	Tervezte Csiszér Csanád

8.3. Illesztőprogram

Az alábbiakban a *Timetagger4* számlálót vezérlő a gyártó példaprogramjából[5] kiinduló C++ kód látható:

```
// timetagger4_user_guide_example.cpp : Example application for the
↪  TimeTagger4
#include <stdio.h>
#include <stdlib.h>
#include <chrono>
#include <thread>
#include <fstream>
#include "TimeTagger4_interface.h"

// If true the time tagger triggers a start periodically
// The time difference of signals on channel A are measured
// else start signal either from input or tiger is used (see below)
// frequency of  start signal is printed and the hits are sampled
const bool USE_CONTINUOUS_MODE = false;
const bool USE_TIGER_START = false;        // if false, external signal
↪  must be provided on start; not applicable if continuous mode is
↪  enabled
const bool USE_TIGER_STOPS = true;         // if false please connect
↪  signals to some of channels A-D

#define MAX_BIN 250000 // divide by two to get the period in us

timetagger4_device * initialize_timetagger(int buffer_size, int
↪  board_id, int card_index) {
        // prepare initialization
        timetagger4_init_parameters params;

        timetagger4_get_default_init_parameters(&params);
        params.buffer_size[0] =
        ↪  buffer_size;                                  // size of the
        ↪  packet buffer
        params.board_id =
        ↪  board_id;                                                    //
        ↪  value copied to "card" field of every packet, allowed range
        ↪  0..255
        params.card_index =
        ↪  card_index;                                                 //
        ↪  which of the TimeTagger4 board found in the system to be
        ↪  used

        int error_code;
        const char * err_message;
```

```
        timetagger4_device * device = timetagger4_init(&params,
        ↪  &error_code, &err_message);
        if (error_code != CRONO_OK) {
                printf("Could not init TimeTagger4 compatible board:
                ↪  %s\n", err_message);
                return nullptr;
        }
        timetagger4_static_info static_info;
        timetagger4_get_static_info(device, &static_info);
        bool timeTaggerNG = static_info.board_revision >= 7;
        if (USE_CONTINUOUS_MODE && !timeTaggerNG) {
                printf("Cannot use continuous mode with TimeTagger
                ↪  1G/2G: %s\n", err_message);
                timetagger4_close(device);
                return nullptr;
        }

        return device;
}

int configure_timetagger(timetagger4_device * device) {
        // prepare configuration
        timetagger4_static_info static_info;
        timetagger4_get_static_info(device, &static_info);
        timetagger4_configuration config;
        // fill configuration data structure with default values
        // so that the configuration is valid and only parameters
        // of interest have to be set explicitly
        timetagger4_get_default_configuration(device, &config);

        // set config of the 4 TDC channels
        for (int i = 0; i < 1; i++)
        // for (int i = 0; i < TIMETAGGER4_TDC_CHANNEL_COUNT; i++)
        {
                // enable recording hits on TDC channel
                config.channel[i].enabled = true;

                // define range of the group
                config.channel[i].start = 0;        // range begins
                ↪  right after start pulse
                if (!USE_CONTINUOUS_MODE) {
                        config.channel[i].stop = MAX_BIN;        //
                        ↪  recording window stops after ~15 us
                }
                else {
                        config.channel[i].stop = 0x7fffffff; // trigger
                        ↪  is independent of stops
                        // set to maximal value
```

```
        }

        // measure only rising edge for tiger (positive) pulse
        ↪  or falling for user (negative) pulse
        // config.trigger[TIMETAGGER4_TRIGGER_A + i].falling =
        ↪  USE_TIGER_STOPS ? 0 : 1;
        // config.trigger[TIMETAGGER4_TRIGGER_A + i].rising =
        ↪  USE_TIGER_STOPS ? 1 : 0;
        config.trigger[TIMETAGGER4_TRIGGER_A + i].falling = 1;
        config.trigger[TIMETAGGER4_TRIGGER_A + i].rising =  1;
}
// config.channel[1].enabled= true;
// config.channel[1].start=0;
// config.channel[1].stop = 300000;
// config.trigger[1].falling = true;
// config.trigger[1].rising = true;

// generate an internal 25 kHz trigger, used for tiger and
↪  continuous mode
config.auto_trigger_period =
↪  (int)(static_info.auto_trigger_ref_clock / 5000);
config.auto_trigger_random_exponent = 0;

// setup TiGeR
// sending a signal to the LEMO outputs (and to the TDC on the
↪  same channel)
// requires proper 50 Ohm termination on the LEMO output to work
↪  reliably

// width of the 12ns pulse in the auto_trigger clock periods
int pulse_width = (int) (12e-9 *
↪  static_info.auto_trigger_ref_clock);

if (!USE_CONTINUOUS_MODE) {
        // use 200 kHz auto trigger to generate

        // generate above configured auto trigger to generate a
        // signal with 12 ns pulse width on LEMO output Start
        config.tiger_block[0].enable = 0; //USE_TIGER_START ? 1
        ↪  : 0;
        config.tiger_block[0].start = 0;
        config.tiger_block[0].stop = config.tiger_block[0].start
        ↪  + pulse_width;
        config.tiger_block[0].negate = 0;
        config.tiger_block[0].retrigger = 0;
        config.tiger_block[0].extend = 0;
        config.tiger_block[0].enable_lemo_output = 1;
```

```
        config.tiger_block[0].sources =
        ↪  TIMETAGGER4_TRIGGER_SOURCE_AUTO;
        // if TiGeR is used for triggering with positive pulses
        if (USE_TIGER_START)
                config.dc_offset[0] =
                ↪  TIMETAGGER4_DC_OFFSET_P_LVCMOS_18;
        else // user input expect NIM signal
                config.dc_offset[0] =
                ↪  TIMETAGGER4_DC_OFFSET_P_TTL;

        // start group on falling edges on the start channel 0
        config.trigger[TIMETAGGER4_TRIGGER_S].falling =
        ↪  USE_TIGER_START ? 0 : 1;
        config.trigger[TIMETAGGER4_TRIGGER_S].rising =
        ↪  USE_TIGER_START ? 1 : 0;
} else {
        // Auto trigger is used as a start signal
        config.tdc_mode = TIMETAGGER4_TDC_MODE_CONTINUOUS;
}

for (int i = 1; i < TDC4_TIGER_COUNT; i++) {
        config.tiger_block[i].enable = 0; // USE_TIGER_STOPS ? 1
        ↪  : 0;
        config.tiger_block[i].start = i * 100;
        config.tiger_block[i].stop = config.tiger_block[i].start
        ↪  + pulse_width;
        config.tiger_block[i].negate = 0;
        config.tiger_block[i].retrigger = 0;
        config.tiger_block[i].extend = 0;
        config.tiger_block[i].enable_lemo_output =
        ↪  USE_TIGER_STOPS ? 1 : 0;
        config.tiger_block[i].sources =
        ↪  TIMETAGGER4_TRIGGER_SOURCE_AUTO;


        if (USE_TIGER_STOPS)
                config.dc_offset[i] =
                ↪  TIMETAGGER4_DC_OFFSET_P_LVCMOS_18;
        else // user input expect NIM signal
                config.dc_offset[i] =
                ↪  TIMETAGGER4_DC_OFFSET_P_TTL;

        // this is not related to the tigers, but uses the same
        ↪  indexing (0 is start)
        // optionally increase input delay by 10 * 200 ps for
        ↪  each channel on new TT
        // config.delay_config[i].delay = i * 10;
}
```

```
        // write configuration to board
        return timetagger4_configure(device, &config);
}


void print_device_information(timetagger4_device * device,
↪  timetagger4_static_info * si, timetagger4_param_info * pi) {
        // print board information
        printf("Board Serial        : %d.%d\n", si->board_serial >> 24,
        ↪  si->board_serial & 0xffffff);
        printf("Board Configuration : %s\n",
        ↪  timetagger4_get_device_name(device));
        printf("Board Revision      : %d\n", si->board_revision);
        printf("Firmware Revision   : %d.%d\n", si->firmware_revision,
        ↪  si->subversion_revision);
        printf("Driver Revision     : %d.%d.%d\n", ((si->driver_revision
        ↪  >> 16) & 255), ((si->driver_revision >> 8) & 255),
        ↪  (si->driver_revision & 255));
        printf("Driver SVN Revision : %d\n", si->driver_build_revision);
        printf("\nTDC binsize         : %0.2f ps\n", pi->binsize);
}

double last_abs_ts_on_a = 0;
int64_t last_group_abs_time = 0;



int64_t processPacket(volatile crono_packet *p, bool print,
↪  timetagger4_static_info *si, timetagger4_param_info *pi, uint16_t
↪  *stats_rising, uint16_t *stats_falling, uint16_t
↪  *stats_rising_first, uint16_t *stats_falling_first) {
        // do something with the data, e.g. calculate current rate
        int64_t group_abs_time = p->timestamp;
        if (!USE_CONTINUOUS_MODE) {
                // group timestamp increments at binsize, but we see
                ↪  only a fraction of the packets (every update_count)
                double rate = 1e12 / ((double)(group_abs_time -
                ↪  last_group_abs_time) * pi->packet_binsize);
                if (print && last_group_abs_time > 0 && p->length>0) {
                        printf("\r%.6f kHz", rate / 1000.0);
                        // ...or print hits (not a good idea at high
                        ↪  data rates,
                        printf("Card %d - flags %d - length %d - type %d
                        ↪  - TS %lu\n", p->card, p->flags, p->length,
                        ↪  p->type, p->timestamp);
                }
                last_group_abs_time = group_abs_time;
```

```
}

int hit_count = 2 * (p->length);
// Two hits fit into every 64 bit word. The second in the last
 ↪  word might be empty
// This flag  tells us, whether the number of hits in the packet
 ↪  is odd
if ((p->flags & TIMETAGGER4_PACKET_FLAG_ODD_HITS) != 0)
        hit_count -= 1;

uint32_t* packet_data = (uint32_t*)(p->data);
uint32_t rollover_count = 0;
//
uint64_t rollover_period_bins = si->rollover_period;

for (int i = 0; i < hit_count; i++)
{
        uint32_t hit = packet_data[i];
        uint32_t channel = hit & 0xf;
        // extract hit flags
        uint32_t flags = hit >> 4 & 0xf;


        if ((flags & TIMETAGGER4_HIT_FLAG_TIME_OVERFLOW) != 0) {
                // this is a overflow of the 23/24 bit counter)
                rollover_count++;
        }
        else {
                // extract channel number (A-D)
                char channel_letter = 65 + channel;

                // extract hit timestamp
                uint32_t ts_offset = hit >> 8 & 0xffffff;

                // Convert timestamp to ns, this is relative to
                 ↪  the start of the group
                uint64_t ts_offset_index = ts_offset +
                 ↪  rollover_count * rollover_period_bins;
                if(ts_offset_index<MAX_BIN){
                        if(flags & TIMETAGGER4_HIT_FLAG_RISING)
                         ↪  stats_rising[ts_offset_index]++;
                        else stats_falling[ts_offset_index]++;
                        if(i == 0){
                                if(flags &
                                 ↪  TIMETAGGER4_HIT_FLAG_RISING)
                                 ↪  stats_rising_first[ts_offset_index]++;
                                else
                                 ↪  stats_falling_first[ts_offset_index]++;
```

```
                        }
                }
                else{
                        printf("ERROR: bigger binsize needed");
                }

                double ts_offset_ns = (ts_offset_index) *
                ↪  pi->binsize / 1000.0;

                if (USE_CONTINUOUS_MODE) {
                        if (channel == 0)
                        {
                                // compute the absolute time by
                                ↪  adding the group time in ns
                                double abs_ts_on_a =
                                ↪  (group_abs_time *
                                ↪  pi->packet_binsize) / 1000 +
                                ↪  ts_offset_ns;
                                double diff = abs_ts_on_a -
                                ↪  last_abs_ts_on_a;
                                if (last_abs_ts_on_a > 0 &&
                                ↪  print) {
                                    printf("Time difference
                                    ↪  between hits on A
                                    ↪  %.1f ns\n", diff);
                                }

                                last_abs_ts_on_a = abs_ts_on_a;
                        }
                }
                else {
                        if (print)
                                printf("Hit  on channel %c -
                                ↪  flags %d - offset %u (raw) /
                                ↪  %.1f ns\n", channel_letter,
                                ↪  flags, ts_offset,
                                ↪  ts_offset_ns);
                }
            }
        }
        return group_abs_time;
}

int main(int argc, char* argv[]) {
        printf("cronologic timetagger4: %s\n",
        ↪  timetagger4_get_driver_revision_str());
        timetagger4_device* device = initialize_timetagger(12 * 1024 *
        ↪  1024, 0, 0);
```

```
if (device == nullptr) {
        exit(1);
}
int status = configure_timetagger(device);
if (status != CRONO_OK) {
        printf("Could not configure TimeTagger4: %s",
        ↪  timetagger4_get_last_error_message(device));
        timetagger4_close(device);
        return status;
}
timetagger4_static_info static_info;
timetagger4_get_static_info(device, &static_info);

timetagger4_param_info parinfo;
timetagger4_get_param_info(device, &parinfo);

print_device_information(device, &static_info, &parinfo);

// configure readout behaviour
timetagger4_read_in read_config;
// automatically acknowledge all data as processed
// on the next call to timetagger4_read()
// old packet pointers are invalid after calling
↪  timetagger4_read()
read_config.acknowledge_last_read = 1;

// structure with packet pointers for read data
timetagger4_read_out read_data;

// start data capture
status = timetagger4_start_capture(device);
if (status != CRONO_OK) {
        printf("Could not start capturing %s",
        ↪  timetagger4_get_last_error_message(device));
        timetagger4_close(device);
        return status;
}

// start timing generator
timetagger4_start_tiger(device);

// some book keeping
int packet_count = 0;
int empty_packets = 0;
int packets_with_errors = 0;
int get_packet_count = 200;
bool last_read_no_data = false;
```

```
uint16_t stat_rising[MAX_BIN];
uint16_t stat_falling[MAX_BIN];
uint16_t stat_rising_first[MAX_BIN];
uint16_t stat_falling_first[MAX_BIN];

for(int i = 0; i<MAX_BIN; i++) {
        stat_rising[i]=0;
        stat_falling[i]=0;
        stat_rising_first[i]=0;
        stat_falling_first[i]=0;
}

std::ofstream of;

int64_t group_abs_time = 0;
int64_t group_abs_time_old = 0;
int update_count = 100;
int save_count = 100;

bool no_data_printed = false;

if (argc>1){
        get_packet_count = atoi(argv[1]);
}

save_count = get_packet_count/10;
update_count = get_packet_count/10;

// read 10000 packets
printf("Reading %d packets\n", get_packet_count);
// printf("Press Ctrl-C to stop\n");
// loop until all packets are read
while (packet_count < get_packet_count)
{
        // get pointers to acquired packets
        status = timetagger4_read(device, &read_config,
        ↪  &read_data);
        // printf("Status: %d\n", status);
        if (status != CRONO_OK) {
                std::this_thread::sleep_for(std::chrono::milliseconds(10));
                //to avoid a lot of lines with no data
                if (!no_data_printed) {
                        printf(" - No data! -\n");
                        no_data_printed = true;
                }
        }
        else
        {
```

```
// iterate over all packets received with the
↪  last read
volatile crono_packet* p =
↪  read_data.first_packet;
while (p <= read_data.last_packet)
{
        // printf is slow, so this demo only
        ↪  processes every nth packet
        // your application would of course
        ↪  collect every packet
        bool print = packet_count % update_count
        ↪  == 0;
        bool save = packet_count % save_count ==
        ↪  0;
        processPacket( p, print, &static_info,
        ↪  &parinfo, stat_rising, stat_falling,
        ↪  stat_rising_first,
        ↪  stat_falling_first);
        no_data_printed = false;
        p = crono_next_packet(p);
        // printf("\n%d\n", packet_count);
        packet_count++;
        if (save){
                of.open("out_rising.txt");
                // of.open("out_rising.txt",
                ↪  std::ios_base::app);
                of<<packet_count;
                for(int i=0; i<MAX_BIN; i++){
                        of<<"\t"<<stat_rising[i];
                }
                of<<"\n";
                of.close();

                of.open("out_rising_first.txt");
                // of.open("out_rising.txt",
                ↪  std::ios_base::app);
                of<<packet_count;
                for(int i=0; i<MAX_BIN; i++){
                        of<<"\t"<<stat_rising_first[i];
                }
                of<<"\n";
                of.close();

                of.open("out_falling.txt");
                // of.open("out_falling.txt",
                ↪  std::ios_base::app);
                of<<packet_count;
                for(int i=0; i<MAX_BIN; i++){
```

```
                                        of<<"\t"<<stat_falling[i];
                                    }
                                    of<<"\n";
                                    of.close();

                                    of.open("out_falling_first.txt");
                                    // of.open("out_falling.txt",
                                    ↪  std::ios_base::app);
                                    of<<packet_count;
                                    for(int i=0; i<MAX_BIN; i++){
                                        of<<"\t"<<stat_falling_first[i];
                                    }
                                    of<<"\n";
                                    of.close();
                                }
                            }
                        }
                    }

                    // shut down packet generation and DMA transfers
                    timetagger4_stop_capture(device);

                    // deactivate timetagger4
                    timetagger4_close(device);
                    return 0;
}
```

8.4. Kiértékelő program

```
import numpy as np
import scipy.signal as signal
import scipy.optimize as opt
from scipy.stats import linregress
import warnings

def load_data(
    dir,
    log=False,
    exc_freq=1e3,
    bin_size=0.5e-9,
    noise_level=0
):
    """
    Loads and processes photon arrival histogram data from measurement
    ↪  files.

    Parameters:
        dir (str): Directory containing measurement files.
```

```
    log (bool): If True, prints summary information.
    exc_freq (float): Excitation frequency in Hz.
    bin_size (float): Time bin size in seconds.
    noise_level (float): Noise level to subtract from data.

Returns:
    time (np.ndarray): Time array for one excitation period.
    normalized_rising_all (np.ndarray): Normalized photon counts per
     ↪ bin.
"""
# nyers adatok betoltese fajlokbol
rising_first_raw = np.loadtxt(dir + '/out_rising_first.txt')
falling_first_raw = np.loadtxt(dir + '/out_falling_first.txt')
rising_all_raw = np.loadtxt(dir + '/out_rising.txt')
falling_all_raw = np.loadtxt(dir + '/out_falling.txt')

# hisztogram adatok kinyerese (elso elem kihagyasa, ami a
↪ ciklusszam)
rising_first = rising_first_raw[1:]
falling_first = falling_first_raw[1:]
rising_all = rising_all_raw[1:]
falling_all = falling_all_raw[1:]

# gerjesztesi ciklusok szama
excitation_cycles = int(rising_first_raw[0])

# meresi konzisztencia ellenorzese
if not (
    rising_first_raw[0] == falling_first_raw[0] ==
    rising_all_raw[0] == falling_all_raw[0]
):
    raise ValueError("Gerjesztesi ciklusok szama nem egyezik meg!")

if rising_all.sum() != falling_all.sum():
    warnings.warn("Csomagvesztesi kockazat az 'all' blokkban!",
     ↪ RuntimeWarning)
if rising_first.sum() != falling_first.sum():
    warnings.warn("Csomagvesztesi kockazat az 'first' blokkban!",
     ↪ RuntimeWarning)

reldiff = (rising_all.sum() - rising_first.sum()) / rising_all.sum()
if reldiff > 0.01:
    warnings.warn(
        "Tulszaturacio kockazatos, az 'first' es 'all' kozti elteres
         ↪ meghaladja 1%-t",
        RuntimeWarning
    )
```

```
    # normalizalas ciklusonkent
    normalized_rising_all = rising_all / excitation_cycles

    # csak a relevans [0, periodus] intervallum kivalasztasa
    bin_number = normalized_rising_all.shape[0]
    period = 1 / exc_freq
    time = np.arange(bin_number) * bin_size

    normalized_rising_all = normalized_rising_all[time <= period]
    time = time[time <= period]

    # zajszint kivonasa, ha meg van adva
    if noise_level > 0:
        normalized_rising_all = np.maximum(normalized_rising_all -
        ↪  noise_level, 0)

    if log:
        print()
        print(f"Dir: {dir}")
        print(f"Gerjesztesi ciklusok szama: {excitation_cycles:.2e}")
        print(f"Regisztralt fotonok szama 'all' blokkokban:
        ↪  {rising_all.sum():.2e}")
        print(f"Regisztralt fotonok szama 'first' blokkokban:
        ↪  {rising_first.sum():.2e}")
        print(f"'first' es 'all' blokkok kozti elteres:
        ↪  {reldiff*100:.2f} %")
        print()

    return time, normalized_rising_all

def deconvolution(
    f_m,
    irf,
    ignore=None,
    ignore_value=1e-9,
    wiener=None
):
    """
    Performs deconvolution of a measured signal with an instrument
    ↪  response function (IRF).

    Parameters:
        f_m (np.ndarray): Measured signal.
        irf (np.ndarray): Instrument response function.
        ignore (int or None): If set, replaces a region in the frequency
        ↪  domain with ignore_value.
        ignore_value (float): Value to use in the ignored region.
        wiener (float or None): Wiener filter parameter.
```

```
    Returns:
        np.ndarray: Deconvoluted signal.
    """
    fft_f_m = np.fft.fft(f_m)
    fft_irf = np.fft.fft(irf)
    tort = fft_f_m / fft_irf

    if wiener is not None:
        irf_abs = np.abs(fft_irf)
        # wiener szuro
        return np.abs(np.fft.ifft(tort * (irf_abs / (irf_abs +
        ↪  wiener))))
    if ignore is not None:
        tort[ignore:-ignore] = ignore_value
    res = np.abs(np.fft.ifft(tort))
    return res

def rolling_average(time, data, window_size):
    """
    Computes a rolling average (moving average) of the data.

    Parameters:
        time (np.ndarray): Time array.
        data (np.ndarray): Data array.
        window_size (int): Window size for averaging.

    Returns:
        (np.ndarray, np.ndarray): (smoothed_time, smoothed_data)
    """
    smooth_data = np.convolve(data, np.ones(window_size) / window_size,
    ↪  mode='valid')
    if window_size > 1:
        # simitott idoablak szamolasa
        smooth_time = time[(window_size-1)//2:-(window_size//2)]
    else:
        smooth_time = time
    return smooth_time, smooth_data

if __name__ == "__main__":
    # parameterek
    bin_size = 0.5e-9  # 1 bin = 0.5 ns
    exc_freq = 150e3   # gerjesztesi frekvencia
    log = True

    # zajszint meghatarozasa sötét meresekbol (nincs gerjesztes, nincs
    ↪  fluoreszcencia)
    _, noise_exc = load_data(
```

```
    'meresek/tcspc/uj/exc/dark_cunt',
    log=log,
    exc_freq=exc_freq,
    bin_size=bin_size
)
_, noise_flu = load_data(
    'meresek/tcspc/uj/fluo/dark_cunt',
    log=log,
    exc_freq=exc_freq,
    bin_size=bin_size
)
exc_noise_level = noise_exc.mean()
flu_noise_level = noise_flu.mean()

# foton eloszlasok betoltese meresekbol
time, exc = load_data(
    'meresek/tcspc/uj/exc/1e8',
    log=log,
    exc_freq=exc_freq,
    bin_size=bin_size,
    noise_level=exc_noise_level
)
time, flu = load_data(
    'meresek/tcspc/uj/fluo/1e9',
    log=log,
    exc_freq=exc_freq,
    bin_size=bin_size,
    noise_level=flu_noise_level
)

# normalizalas valoszinusegi eloszlassal
exc /= exc.sum()
flu /= flu.sum()

# simitas mozgóatlaggal
w = 20
_, exc = rolling_average(time, exc, w)
time, flu = rolling_average(time, flu, w)

# --- dekonvolucio ---

# csak az elso szelet elemzese (t_max-ig)
t_max = 1e-6
i_max = int(t_max / bin_size)
local_time_array = time[:i_max]
local_exc = exc[:i_max]
local_flu = flu[:i_max]
```

```
deconvoluted_signal = deconvolution(
    local_flu,
    local_exc,
    ignore=170,
    ignore_value=1e-4
)

# exponencialis lecsenges illesztese a dekonvolvalt jelre
def exp_decay(t, A, tau, C):
    return A * np.exp(-t / tau) + C

popt, pcov = opt.curve_fit(
    exp_decay,
    local_time_array,
    deconvoluted_signal,
    p0=[1, 4e-9, 0]
)
fit_curve = exp_decay(local_time_array, *popt)

print("Dekonvolúció eredménye:")
print(f"Jellemző idő (tau): {popt[1]*1e9:.2f} ns")
perr = np.sqrt(np.diag(pcov))
for i, (param, err) in enumerate(zip(popt, perr)):
    print(f"Parameter {i}: {param:.3e} ± {err:.3e}
    ↪  ({err/abs(param)*100:.2f}%)")

# --- rekonvolucio (modell ellenorzes) ---

t_test_interval = 0.6e-7
t_test = time[:int(t_test_interval / bin_size)]

def model_convoluted_decay(t, tau, c):
    """
    Model: convolution of excitation with exponential decay.
    """
    # konvolucio az exc jellel es exponencialis lecsengessel
    f_m = signal.convolve(
        exc,
        c * np.exp(-t / tau),
        mode='full'
    )[:len(t_test)]
    return f_m

popt, pcov = opt.curve_fit(
    model_convoluted_decay,
    t_test,
    flu[:len(t_test)],
    p0=[1e-7, 1]
```

```
)
print("Rekonvolúció eredménye:")
print(f"Fitted tau: {popt[0]:.2e} s ± {np.sqrt(pcov[0,0]):.2e}")
print(f"Fitted c: {popt[1]:.2e} ± {np.sqrt(pcov[1,1]):.2e}")
```

Hivatkozások

[1] W. Becker. *The bh TCSPC handbook*. http://www.becker-hickl.com/literature/documents/flim/the-bh-tcspc-handbook/. 2023.

[2] Burle Industries. *Photomultiplier Handbook*. Essentially the Engstrom-RCA Handbook reprinted.

[3] Becker & Hickl GmbH. *TCSPC Devices for Ultra-High Time Resolution Single-Photon Counting*. https://www.becker-hickl.com/products/category/tscpc-flim-time-tagging/.

[4] Cronologic GmbH. *TimeTagger: The low-cost, mid-resolution time-to-digital converter*. https://www.cronologic.de/product/timetagger.

[5] Cronologic GmbH. *TimeTagger4 User Guide*.

[6] PicoQuant GmbH. *TCSPC and Time Tagging Electronics*. https://www.picoquant.com/products/category/tcspc-and-time-tagging-modules.

[7] HORIBA. *Fluorescence Lifetimes with TCSPC*. https://www.horiba.com/usa/scientific/products/fluorescence-spect

[8] Marie Kaplanová és Karel Čermák. „Effect of reabsorption on the concentration dependence of fluorescence lifetimes of chlorophyll a". *Journal of Photochemistry* 15.4 (1981), 313–319. oldal. ISSN: 0047-2670. DOI: https://doi.org/10.1016/0047-2670(81)80005-6. URL: https://www.sciencedirect.com/science/article/pii/0047267081800056.

[9] William Lawrence és mások. „A comparison of avalanche photodiode and photomultiplier tube detectors for flow cytometry - art. no. 68590M". 6859. kötet. 2008. márc., 68590M–68590M. oldal. DOI: 10.1117/12.758958.

[10] Beata Myśliwa-Kurdziel és mások. „Solvent effects on fluorescence properties of protochlorophyll and its derivatives with various porphyrin side chains". *European Biophysics Journal : EBJ* 37.7 (2008), 1185–1193. oldal. DOI: 10.1007/s00249-008-0288-x. URL: https://doi.org/10.1007/s00249-008-0288-x.

[11] ID Quantique. *ID120 brochure*. https://marketing.idquantique.com/acton/attachment/11868/f-0238/1/-/-/-/-/ID120_Brochure.pdf.

[12] ID Quantique. *Photon Counting for Brainies*. 2019.

[13] J. Večeř és mások. „Reconvolution analysis in time-resolved fluorescence experiments—an alternative approach: Reference-to-excitation-to-fluorescence reconvolution". *Review of Scientific Instruments* 64.12 (1993. dec.), 3413–3424. oldal. ISSN: 0034-6748. DOI: 10.1063/1.1144312. eprint: https://pubs.aip.org/aip/rsi/article-pdf/64/12/3413/19024859/3413_1_online.pdf. URL: https://doi.org/10.1063/1.1144312.

[14] M. Wahl. *Time-Correlated Single Photon Counting*. Technical Note. PicoQuant GmbH.