Heinz Nixdorf-Lehrstuhl für Medizinische Elektronik

Monitoring multiparametrischer komplexer Mikrosensorarrays für zelluläre Analytik

Eléonore Cabala

Vollsändiger Abdruck der von der Fakultät für Elektrotechnik und Informationstechnik der Technischen Universität München zur Erlangung der akademischen Grades eines

Doktor -Ingenieurs

genehmigten Dissertation.

Vorsitzender:	Univ.-Prof. Dr.-Ing. H.-G. Herzog
Prüfer der Dissertation:	1. Univ.-Prof. Dr. rer. nat. habil. B. Wolf
	2. Univ.-Prof. Dr.-Ing. habil. A. W. Koch

Die Dissertation wurde am 28.09.2006 bei der Technischen Universität München eingereicht und durch die Fakultät für Elektrotechnik und Informationstechnik am 29.01.2007 angenommen.

Danksagung

Ich möchte zuerst Herrn Prof. B. Wolf für die sehr gute Betreuung, für die interessanten wissenschaftlichen Vorträge und Seminare, die er organisiert, und für die entspannte Atmosphäre in seiner Arbeitsgruppe danken.

Herrn Dr. H. Grothe möchte ich für seine Vorschläge zur Erstellung der Software und Datenauswertungen sowie Korrektur der schriftlichen Arbeit danken.

Frau Dr. A. Otto und Herrn Dr. M. Brischwein danke ich für die Unterstützung in der Zellbiologie.

Herrn R. Arbogast und Herrn W. Ruppert sei für die Herstellung der Doppel-Kammer für die Glas-Sensorchips gedankt.

Frau M. Remm danke ich für die Herstellung der Glas-Sensorchips.

Frau I. Szabadoz, Frau G. Teschner und Herrn A. Michelfelder sei für die Unterstützung im Zellkulturlabor gedankt.

Frau M. Brückl bin ich für die Unterstützung bei der Untersuchung der Glas-Sensorchips dankbar.

Frau M. Mayr sei für die Korrektur der schriftlichen Arbeit gedankt.

Herrn A. da Silva verdanke ich viele konstruktive Vorschläge.

Zuletzt spreche ich allen anderen Arbeitskollegen, Freunden, Eltern und meinem Bruder für ihre Unterstützung und wissenschaftlichen Diskussionen meinen Dank aus.

Inhaltsverzeichnis

Kapitel 1

Einführung

1.1 Zell-Chip-Systeme

Die Forschung an Zellen in vitro spielt eine sehr wichtige Rolle in der Medizin und Pharmakologie, um Krankheiten zu verstehen, zu diagnostizieren oder Medikamente zu entdecken. Zudem können Tier-Versuche vermieden werden. Verschiedene in vitro Tests werden von Pharma-Industrien und Forschungslabor benutzt. Verbreitete Methoden sind so genannte Endpunkt-Methoden, die verschiedene Parameter wie Zellzahl, transkribierte RNA oder Proteine analysieren, nachdem die Zellen mit einem Wirkstoff inkubiert wurden. Andere sehr verbreitete Verfahren benutzen Mikrotiterplatten, kombiniert mit optischen Sensoren, um fluoreszente oder lumineszente Signale von Zellen zu detektieren. Diese Messmethoden sind in den 90er Jahren entstanden und haben immer mehr an Bedeutung gewonnen, da 24, 96, 384 bis 1536 parallele Tests durchgeführt werden können, was ein Hochdurchsatzscreening von Wirkstoffen erlaubt (Taylor et al. [37] [42]).

Das Prinzip der Fluoreszenz-Assays ist das folgende: Die Zellen werden in Mikrotiterplatten kultiviert und ein Testprodukt wird ihnen zugegeben. Nach einer gewissen Zeit werden sie mit einem Fluoreszenzfarbstoff inkubiert, der in die Zellen eintreten kann und sich an bestimmten Proteinen oder Ionen bindet. Das Testergebnis wird danach durch ein optisches Messgerät gemessen. Roboter mit Mikrofluidik-Systemen werden benutzt, um den ganzen Prozess zu automatisieren. Durch diese Methode können apoptotische oder lebende Zellen markiert werden, die freie intrazelluläre Calcium-Konzentration kann ermittelt werden, oder Zellorganellen (Cytoskelett, Kern, Mitochondrium) können markiert werden. Ausführliche Beschreibungen von Fluoreszenzfarbstoffen werden z. B. auf der Internetseite der Firma „Invitrogen“ angegeben [79].

Zurzeit sind verschiedene Zell-Screening-Geräte auf dem Markt verfügbar. Die Firma Agilent (US [74]) verkauft zum Beispiel ein „Lab-on-a-Chip“-System (Bioanalyzer 2100), das erlaubt, die Größe und Anzahl von RNA, DNA und Proteinen zu analysieren. Das System erlaubt auch Zell-Fluoreszenz-Assays durchzuführen. Die Firma Evotec (Hamburg [77]) verkauft das Gerät „EVOScreen“, das Fluoreszenz-zelluläre Screenings im 384 Well-Format durchführen kann. Die Firma Molecular Devices (US [82]) verkauft auch mehrere Auslesegeräte für Fluoreszenz-Assays.

Fluoreszenz-Assays erlauben zwar, ein Hochdurchsatzscreening von Wirkstoffen durchzuführen, aber sie haben den Nachteil, dass der Farbstoff Einfluss auf die Zellfunktionen haben kann oder sogar toxisch sein kann. Crawford et al. [4] haben gezeigt, dass mehrere Fluoreszenzfarbstoffe (Dimethylsulfoxid, Rhodamin-123 ...usw.) Einfluss auf das Zellwachstum von Fibroblasten haben. Die Messung kann deshalb nur zeitpunktuell bleiben. Sehr wichtige Parameter wie zelluläre Adhäsion und metabolische Aktivität können durch diese Methoden nicht langzeitig evaluiert werden.

Ein anderes Messprinzip, das diese Nachteile umgeht, wurde von Wolf und Mitarbeitern am Ende der 90er Jahre entwickelt [51],[52]. Das Messprinzip basiert auf der Beobachtung, dass die Zellen hochkomplexe Systeme sind, die „Eingangssignale“ (Hormone, Wachstumsfaktoren) empfangen, verarbeiten und eine entsprechende Anwort auslösen (siehe Abbildung 1.1). Die Zellantworten (pH,

Ionen-Austausch, Adhäsionsänderung) können dann durch Biosensoren detektiert werden, ohne die Zellen zu beeinflussen. Deswegen wurden multiparametrische Biosensorarrays, auf denen Zellkulturen direkt wachsen, entwickelt, um Messungen in der Mikroumgebung von lebenden Zellen mehrere Tage lang in Realzeit durchzuführen. Es wurde bereits in den 90er Jahren ein Messgerät von der Firma Molecular Devices (USA) entwickelt, dass das gleiche Messprinzip benutzt [36] [46]. Das entwickelte Gerät („Cytosensor Microphysiometer“) konnte die Ansäuerungsrate der Zellen mit optoelektronischen Sensoren (LAPS) in Realzeit und über längere Zeiträume messen. Es evaluierte aber nur einen einzigen Parameter und die Interpretation der Messergebnisse war nicht immer eindeutig.

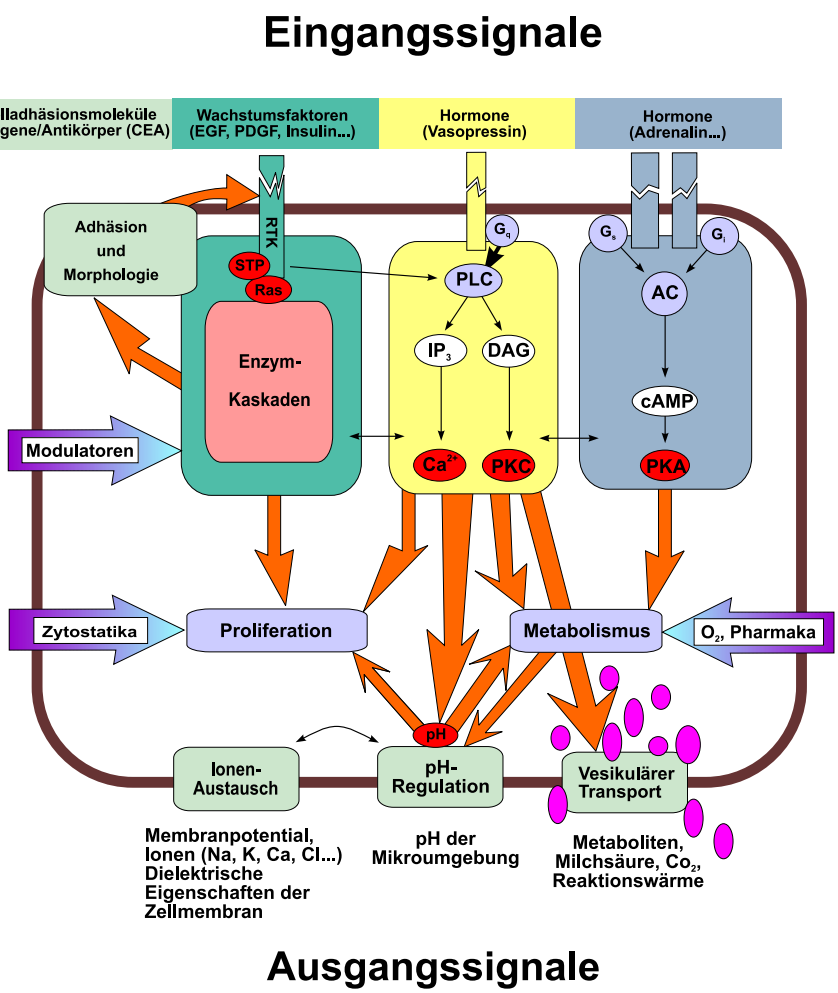


Abbildung 1.1: Schematische Darstellung der Funktionen der Zelle und der Möglichkeit, sie mit Sensoren on-line zu detektieren, nach Wolf et al. aus [12].

Die entwickelten multiparametrischen Biosensorrays, so genannte Zell-Chip-Systeme, messen im Gegensatz dazu mehrere Parameter gleichzeitig. Sie messen die metabolische Aktivität der Zellen, indem sie den extrazellulären pH-Wert und die gelöste Sauerstoff-Konzentration messen, und durch Impedanzsensoren detektieren sie Zellwachstum und Änderungen der zellulären Adhäsion oder Morphologie. In Abbildung 1.2 wird der Aufbau eines Zell-Chip-Systems aufgezeigt. Die Zell-Chip-Systeme bestehen aus biokompatiblen Sensorchips, die verschiedene Mikrosensoren integrieren: Sensoren für pH, gelöste Sauerstoff-Konzentration, Impedanz und Temperatur. Zellen werden in einer Mikrokammer auf den Sensorchips kultiviert, deren Volumen typischerweise 10 Mikroliter ist. Die Zellen verbrauchen die Nährstoffe des Kulturmediums, das periodisch durch ein Fluidik-System gewechselt wird. Die typischen Zeiten des Mediumwechsels sind 10 Minuten. Unter diesen Bedingungen können die Zellen auf dem Chip mehrere Tage lang leben. Zum Kulturmedium werden chemische Wirkstoffe hinzugefügt,

um deren Einfluss auf die Stoffwechselaktiväten der Zellen zu untersuchen. Wie in Abbildung 1.2 dargestellt, ist es bei Glas-Sensorchips noch dazu möglich die Zellen gleichzeitig zu mikroskopieren.

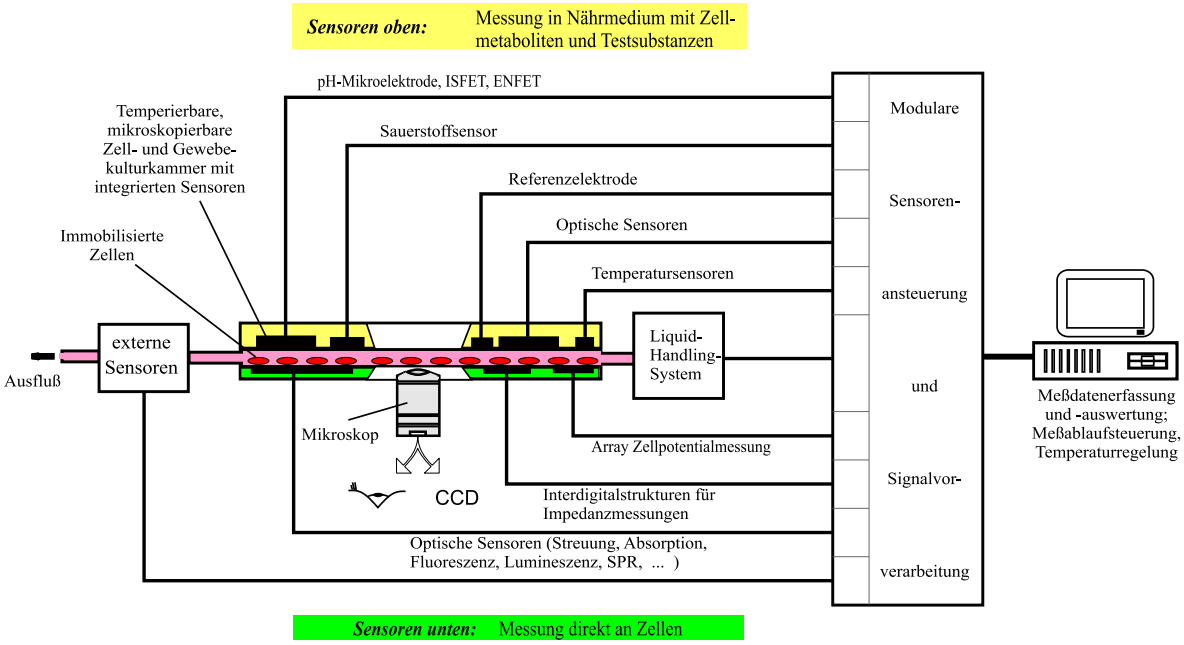


Abbildung 1.2: Schematische Darstellung eines Zell-Chip-Systems: eine Zellkultur wächst auf einem Sensorchip, der pH-, O_2-, Impedanz- und Temperatur-Sensoren integriert. Das Kulturmedium wird durch ein Fluidik-System gewechselt. Wirkstoffe werden ins Medium eingefügt und die Sensoren liefern in Realzeit Informationen über die Zellreaktionen, nach Wolf et al. aus [12].

Die Effekte von Wirkstoffen auf die Stoffwechselaktivität der Zellen werden mehrere Tage lang und in Realzeit registriert. Die Anwendungsbereiche der Zell-Chip-Systeme sind sehr breit. Für die Umweltüberwachung können Algen mit den Sensoren toxische Substanzen im Wasser detektieren. In der Pharmaforschung können Medikamente lange Zeit auf Zellen getestet werden. In der Medizin werden die Effekte von Chemotherapeutika auf Tumor-Zellen untersucht, wie eine Studie von Henning gezeigt hat [38].

1.2 Ziel der Arbeit

Die Realisierung von Zell-Chip-Systemen ist eine interdisziplinäre Arbeit, die aus der Entwicklung biokompatibler Sensoren, von analoger und digitaler Elektronik sowie Software und Datenauswertung besteht. Die Erforschung neuer Sensoren auf Glas-, Silizium- und Keramik-Substraten, sowie die Realisierung der entsprechenden Auswertungselektronik ist das Thema anderer Doktorarbeiten (J. Wiest, J. Ressler). Das Thema dieser Arbeit ist die Entwicklung und Implementierung von Software für Zell-Chip-Systeme.

1.3 Software-Entwicklung

Mit zuhnehmender Computerleistung können immer komplexere Anwendungen realisiert werden. Gleichzeitig wird aber auch die Softwareentwicklung selbst anspruchsvoller und komplexer. Die Software-Entwicklung besteht daher aus mehreren Aktivitäten. Die erste wichtige Aktivität, die zusammen mit den Benutzern realisiert wird, ist die Definition einer Liste der Funktionalitäten und Anforderungen der Software. Wichtig ist zu entscheiden, welche Funktionalitäten die höchste Priorität haben und

unter welchem Betriebsystem die Anwendung laufen soll. Die Zell-Chip-Softwares sollen unter dem Windows-Betriebssystem laufen, über eine „angenehme“ graphische Bedienoberfläche verfügen, und mit verschiedenen Messgeräten kommunizieren, um Messdaten zu erfassen. Eine Datenbankverbindung ist als optionale Funktionalität ebenfalls erwünscht.

Die zweite wichtige Aktivität der Software-Entwicklung ist es, die Software zu strukturieren. Dies kann durch Modellierungsmethoden wie UML (Universal Modelling Language) erfolgen, die in dieser Arbeit teilweise verwendet wird.

Die dritte essentielle Aktivität ist die Implementierung der Software. Für diese Aktivität müssen zuerst verfügbare Programmiersprachen und IDE (Integrated Development Environment) betrachtet und gewählt werden. Zurzeit verfügt man über zahlreiche Möglichkeiten und die Wahl einer Programmiersprache ist nicht trivial. Aktuelle, häufig benutzte Programmiersprachen sind C, C++, C#, Java, Python, Tcl/Tk, Delphi, LabVIEW, Visual Basic. Diese Sprachen sind alle für die Entwicklung von Zell-Chip-Software geeignet und zeigen Vorteile sowie Nachteile. Zum Beispiel hat die LabVIEW-Sprache (Laboratory Virtual Instrument Engineering Workbench) den Vorteil, dass sie relativ einfach zu lernen ist. Sie ist eine visuelle Sprache (G-Sprache: Visual Programming Language), die die Entwicklungszeit von einfacher Datenerfassungssoftware erheblich reduzieren kann. Sie kann von Amateur-Softwareentwicklern benutzt werden, wie Baroth und Hartsough [61] gezeigt haben. Visuelle Programmiersprachen wie LabVIEW sind zwar attraktiv, aber das Lesen und Verstehen des logischen Programmcodes ist schwieriger im Vergleich mit textuellen Programmiersprachen, wie ein Test von Green und Petre [61] und eine kürzlich erfolgte Umfrage von Whitley und Blackwell [71] gezeigt haben. LabVIEW hat auch den Nachteil, dass die Sprache nicht immer genug Flexibilität für die Gestaltung der graphischen Bedienoberfläche und für die Darstellung von Daten bietet.

Java, C++ und C# haben den Vorteil, dass sie so genannte objektorientierte Programmiersprachen sind. Die objektorientierten Programmiersprachen haben seit den 90er Jahren in der Industrie immer mehr an Bedeutung gewonnen, da sie eine gute Strukturierung des Quellcodes erlauben. Durch die Begriffe von Objekt, Klasse und Vererbung wird der Softwarecode als mehrere Objekte und die Beziehung zwischen den Objekten modelliert. Cuadrado et al. [65] erklären, dass objektorientierte Modellierung sehr gut für die Entwicklung von Standard-Software zur Automatisierung und Datenverwaltung in der analytischen Chemie geeignet ist. Objektorientierte Programmiersprachen haben aber auch Nachteile. Eine Studie von Wiedenbeck et al. [72] zeigt zum Beispiel, dass lange objektorientierte Quellcodes für Anfänger weniger verständlich sind als prozedurale Quellcodes (C oder Pascal z. B.) ist. Deshalb fiel in dieser Arbeit die Entscheidung für die C und C++ Programmiersprachen. Für die Herstellung der graphischen Bedienoberfläche wurden C++ Klassen und Objekte benutzt. Für die Kommunikation mit Messgeräten, die Speicherung und das Laden von Messdaten wurden C-Funktionen benutzt.

Für viele Programmiersprachen werden von der Software-Industrie entsprechende IDE entwickelt, wie „Microsoft Visual C++“ oder „Borland C++ Builder“ für die C++ Programmiersprache. Eine IDE integriert einen Compiler oder Interpreter, Debugger, Text-Editor und manchmal auch einen Bedienoberfläche-Editor. Die IDE werden entwickelt, damit die Programmierer sich auf die Funktionalitäten ihrer Software konzentrieren, und weniger Zeit mit administrativen Tätigkeiten (wie die Beschreibung des Kompilierungsprozess) verbringen können. In einer Umfrage von Kline und Seffah [69] wurde „Microsoft Visual C++ 6.0“ als schwierig zu lernen und nicht genug dokumentiert empfunden. Da persönliche Erfahrungen ähnliche Eindrücke ergeben haben, fiel die Entscheidung, mit der „Borland C++ Builder 6.0“ IDE zu arbeiten, die für diese Anwendung besonders geeignet und vorteilhaft erschien. Die IDE ist ein RAD-Werkzeug (Rapid Application Development). Sie erlaubt die graphische Bedienoberfläche mit der Maus zu gestalten statt einen Code zu schreiben, was Entwicklungszeit spart. Sie wird mit einer Bibliothek (VCL-Bibliothek: Visual Component Library) geliefert, die mehrere fertige Komponenten integriert, wie Kommunikation mit SQL-Datenbanken, und die relativ einfach zu benutzen und gut dokumentiert ist, was viel Entwicklungszeit einspart.

1.4 Gliederung der Arbeit

Diese Arbeit ist in sechs Teile gegliedert. Im ersten Teil werden einige Eigenschaften der Zellen in Erinnerung gerufen, die in anderen Teilen dieser Arbeit verwendet werden. Im zweiten Teil werden das benutzte Material und die Methoden beschrieben. Dieser Teil enthält Beschreibungen und Erklärungen der Funktionsprinzipien der Sensoren, sowie Beschreibungen der verschiedenen Chips, die entwickelt worden sind, Beschreibungen von drei Zell-Chip-Messgeräten, die zur Erfassung der gezeigten Messdaten dieser Arbeit benutzt wurden, und zuletzt Beschreibung der verwendeten Zellkultur-Arbeitsmethoden.

Spezielle Datenverarbeitungsmöglichkeiten für Zell-Chip-Messungen werden im dritten Teil präsentiert. Sie hängen von den gemessenen Signalen ab und werden zuerst für Sauerstoffsignale, dann für pH-Signale und zuletzt für die Impedanz-Messsignale erklärt. Es wird schließlich erklärt, wie es möglich ist, einen Vitalitätsindex für Zellkulturen zu definieren und zu berechnen.

Mehrere Software-Programme werden im vierten Teil präsentiert, die in dieser Arbeit entwickelt und verwendet worden sind. Zwei Messprogramme und eine Zell-Chip-Mess/Analyse-Software werden beschrieben. In diesem Kapitel wird erklärt, welche besondere Funktionen die Software für Zell-Chip-Systeme hat.

Die Arbeitsergebnisse werden im fünften Teil diskutiert.

Im letzten Teil dieser Arbeit wird der Einsatz der Software für ein Hochdurchsatz-Zell-Chip-System präsentiert, das gerade entwickelt wird.

Kapitel 2

Über die Zellen

Wie es in der Einführung erklärt wurde, erlauben Zell-Chip-Systeme, mehrere Parameter in der Mikroumgebung von Zellkulturen zu messen, wie pH-Wert oder gelöste O_2-Konzentration. Sie können auch für bestimmte Kulturen (adhärente tierische Kulturen) die Zellmorphologie, die Adhäsion und das Wachstum messen. Deshalb werden allgemeine Merkmale von Zellen in diesem Kapitel beschrieben, wie Zellstruktur, Zellmetabolismus, und Zellwachstum, die wichtig für die folgenden Kapitel dieser Arbeit sind. Es werden auch einige Grundbegriffe über tierische Zellkulturen vorgestellt, die bei der Gestaltung von Zell-Chip-Software und Datenbanken (Kapitel 5) Verwendung finden.

2.1 Zellaufbau

Die einfachsten und ältesten Zellen sind die Prokaryonten, die 1 bis 10 Mikrometer groß sind. Diese Zellen haben einen einzigen zytoplasmischen Bereich mit DNA, RNA, Proteinen und anderen kleinen Molekülen. Sie besitzen keine Zellorganellen, wie Zellkern oder Mitochondrium, und sind alle unizelluläre Lebewesen. Die Bakterien und Archaebakterien gehören zu den Prokaryonten.

Im Gegensatz zu den Prokaryonten haben die Eukaryontenzellen viele Zellorganellen, wie Zellkerne mit Chromosomen, Mitochondrien oder Chloroplasten . Eukaryontenzellen können 10 bis 100 Mikrometer groß sein. Alle Pilze, Tier- und Pflanzenzellen gehören zu den Eukaryonten. Einige Eukaryonten sind unizelluläre Lebewesen, wie die Protisten (Pantoffeltierchen z. B.). Die meisten Eukaryonten bilden aber multizelluläre Lebewesen. Zum Beispiel hat der Mensch mehr als 200 Zelltypen, die spezialisiert sind und zusammenarbeiten, wie die Epithelzellen, Muskelzellen, Sinneszellen, Nervenzellen, Fibroblasten und Blutzellen [1], [10], [13].

In den folgenden Abschnitten werden Zellorganellen, Zellmembran und Cytoskelett der Tier- und Pflanzenzellen kurz beschrieben.

- Zellorganellen
  Die Tier- und Pflanzenzellen haben ähnliche mit Membran umgebene Kompartimente, die Zellorganellen genannt werden. In den Zellorganellen finden spezifische Zellfunktionen statt. Im Zellkern zum Beispiel wird die Proteinsynthese gesteuert. Im Endoplasmatischen Retikulum werden Proteine synthetisiert, während sie im Golgi-Apparat modifiziert werden. Die Mitochondrien und Chloroplasten dienen zur Produktion von biochemischer Energie und sind die einzigen Zellorganellen, die eine eigene DNA haben, und die sich in ähnlicher Weise wie Bakterienzellen teilen können. Der typische Aufbau einer tierischen Zelle ist in der Abbildung 2.1 dargestellt.

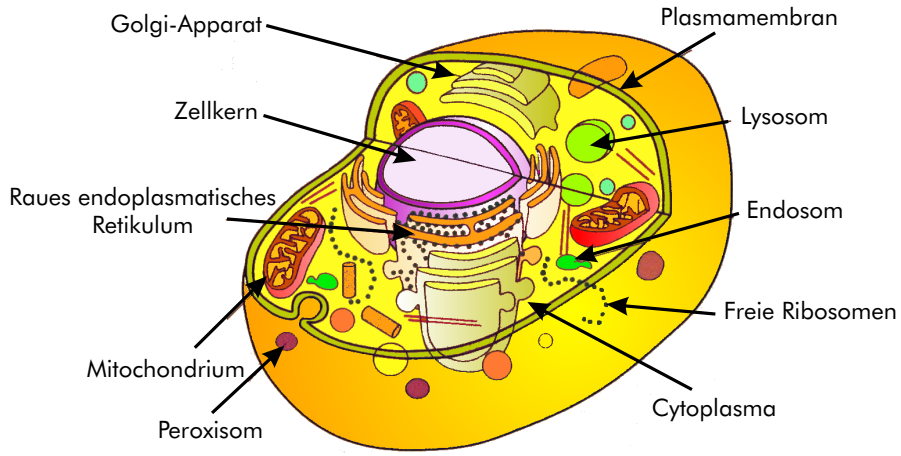


Abbildung 2.1: Aufbau einer tierischen Zelle nach Koolman et al. [10].

- Zellmembran
  Die Zellen sind durch eine Plasmamembran vom extrazellulären Milieu getrennt. Die Pflanzenzellen sind zusätzlich von einer rigiden Zellwand umschlossen, die zu 50% aus Cellulose (Polysaccharide) besteht.

  Die Plasmamembran ist eine komplexe und flexible Struktur, die aus einer Doppelschicht von Lipiden besteht, die Proteine und Kohlenhydrate integriert, wie in Abbildung 2.2 dargestellt. Als Lipide findet man Phospholipide am häufigsten, aber auch Cholesterol und Glycolipide. Die Lipide haben einen polaren, hydrophilen Teil und einen unpolaren, hydrophoben Teil. Die Lipid-Doppelschicht ist deshalb für kleine ungeladene Moleküle, wie Wasser, Sauerstoff, Kohlendioxid durchlässig und für Ionen undurchlässig. Um einen Austausch von Ionen zwischen dem Cytoplasma und dem extrazellulären Raum und den Eintritt großer Moleküle zu erlauben, integriert die Plasmamembran verschiedene Kanäle, wie Natrium-, Kalium-, Chlorid-Ionenkanäle und Glukose-Kanäle.

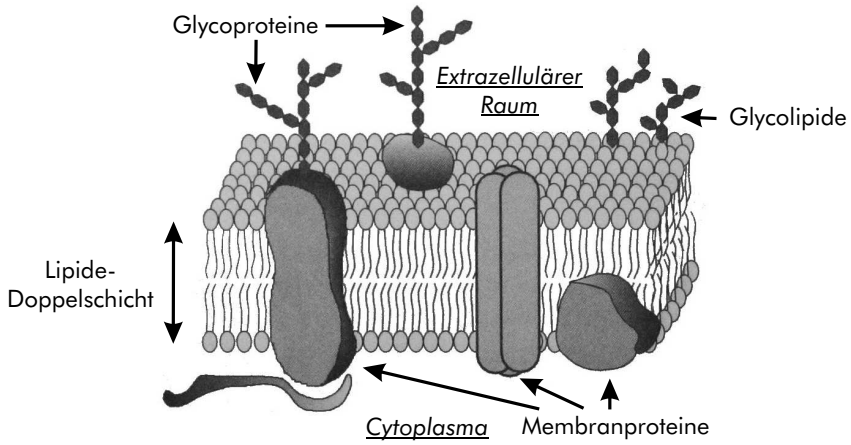


Abbildung 2.2: Schematische Darstellung der Plasmamembran nach Petit et al. [16], deren Dicke 7 nm beträgt.

Die Zellmembran integriert auch andere Proteine, die als Rezeptoren für Signalstoffe fungieren, wie Hormone oder Neurotransmitter. Andere Membranproteine, wie die Glycoproteine Cadherin (Typ P, E oder N), erlauben eine Verbindung der Zellen untereinander. Die Cadherine sind im Cytoplasma mit Aktin-Filamenten verbunden, die die Zellstruktur bilden (siehe nächster Punkt). Alle Glycoproteine und Glycolipide auf der Zelloberfläche bilden die so genannte Glycocalix.

- Cytoskelett
  Die Struktur eukaryontischer Zellen wird durch verschiedene Filamente gebildet: Mikrofilamente aus Actin (6-8 nm), Intermediär-Filamente (10 nm) und Mikrotubuli aus Tubulin (25 nm). Die Filamente sind auch Transportschiene für Proteinkomplexe oder Organellen und Motor für die Bewegung [10].

2.2 Zellmetabolismus

Die Zellen können durch ihren Metabolismus in zwei Gruppen unterteilt werden. Autotrophe Zellen, wie die Pflanzenzellen, verbrauchen anorganische Moleküle, wie Kohlendioxid (CO_2), und Wasser (H_2O) und bilden organische Moleküle. Heterotrophe Organismen verbrauchen Sauerstoff (O_2) und organische Moleküle, wie Glukose, und setzen CO_2 und H_2O frei. Der Zellmetabolismus dieser beiden Arten von Zellen wird in den folgenden Abschnitten kurz erklärt. In allen Zellen wird die biochemische Energie zum größten Teil durch die ATP-Moleküle (Adenosin-Triphosphat) gespeichert, da die Umwandlung von ATP in ADP (Adenosin-Diphosphat) Energie für andere Reaktionen freigeben kann:

$$ATP \rightarrow ADP + P + Energie$$

2.2.1 Tier- und Hefe-Zellen

Der Zellmetabolismus besteht in einer Vernetzung von parallelen und voneinander abhängigen Reaktionen. Das Ziel des Zellmetabolismus ist einerseits, Moleküle abzubauen (katabolische Metabolismuswege), um biochemische Energie zu produzieren, und andererseits, neue Zellelemente zu synthetisieren (anabolische Metabolismuswege). Der Abbau von Glukose, einer der wichtigsten katabolischen Wege, besteht aus mehreren Schritten: der Glykolyse, dem Zitronensäure-Zyklus und der Gärung, die in den folgenden Punkten beschrieben werden.

- Glykolyse
  Im Cytoplasma findet die Glykolyse statt. Die Glykolyse ist eine Reaktion, die Glukose (Moleküle mit 6 Kohlenstoffatomen) in zwei Pyruvat-Moleküle (mit 3 Kohlenstoffatomen) umwandelt. Die Bilanz dieser Reaktion ist:
  $$Glukose \rightarrow 2\ Pyruvate$$
  Diese Reaktion braucht keinen Sauerstoff, sie ist anaerob. Sie produziert 2 ATP-Moleküle pro Glukose.

- Zitronensäure-Zyklus
  Wenn Sauerstoff vorhanden ist, finden nach der Glykolyse weitere Reaktionen statt, um Energie aus dem Pyruvat zu gewinnen. Diese Reaktionen bilden den so genannten Zitronensäure-Zyklus und laufen in Mitochondrien ab. Die Bilanz der zellulären Atmung ist (Glykolyse und Zitronensäure-Zyklus):
  $$6\ O_2 + Glukose \rightarrow 6\ CO_2 + 6\ H_2O$$
  Der Zitronensäure-Zyklus produziert mehr als 35 ATP-Moleküle und ist die größte Energiequelle der Zellen.

- Alkoholische Gärung (Hefen)
  Wenn kein Sauerstoff vorhanden ist, findet die alkoholische Gärung im Cytoplasma von Hefen statt. Die Bilanz der anaeroben Atmung ist:
  $$Glukose \rightarrow 2\ Ethanol + 2\ CO_2 + 2\ H_2O$$

- Gärung (tierische Zellen)
  Wenn kein Sauerstoff vorhanden ist, findet in den tierischen Zellen eine andere Gärung statt, die Laktat (Milchsäure) bildet. Die Bilanz der anaeroben Atmung ist:

$$Glukose \rightarrow 2\ Laktat^- + 2\ H^+ + 2\ H_2O$$

  In Abbildung 2.3 sind diese Reaktionen zusammengefasst. Sauerstoffverbrauch und Säureproduktion wurden blau bzw. rot eingefärbt, da sie mit den Zell-Chip-Biosensoren gemessen werden.

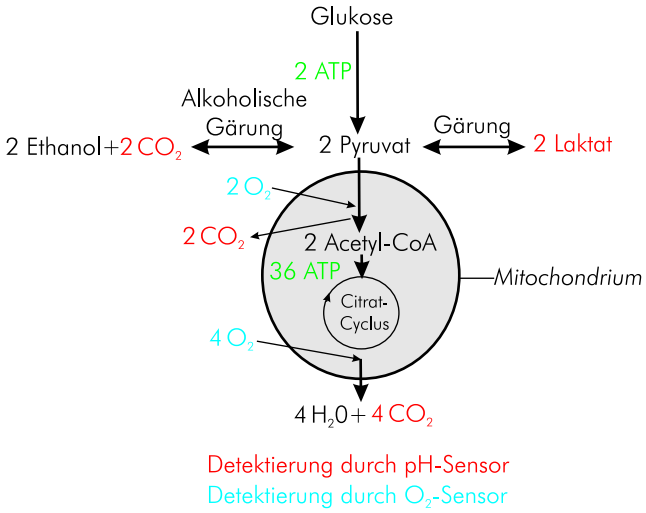


Abbildung 2.3: Glykolyse, Gärung und Atmung

  Der Abbau von Glukose ist nicht der einzige Weg, um ATP zu generieren. Eine andere Energiequelle kommt zum Beispiel vom Abbau von Aminosäuren, wie Glutamin. Glutamin wird zu Glutamat im Cytoplasma transformiert. Glutamat wird danach im Zitronensäure-Zyklus benutzt, um Energie zu erzeugen.

2.2.2 Pflanzen- und Algen-Zellen

Die Pflanzen, Algen und bestimmte Bakterien benutzen Lichtenergie, um organische Moleküle zu synthetisieren: Sie betreiben Photosynthese. In grünen Algen und höheren Pflanzen findet die Photosynthese in den Chloroplasten statt. Die Bilanz der Photosynthese lautet:

$$6\ CO_2 + 6\ H_2O + Licht \rightarrow C_6H_{12}O_6 + 6\ O_2$$

Die Photosynthese funktioniert nur bei Präsenz von Licht (wie ihr Name sagt). Um in der Nacht ATP bilden zu können, haben die Pflanzenzellen Mitochondrien, die unter Präsenz von Sauerstoff erlauben, ATP zu bilden, wie bei den tierischen Zellen. Wenn Zell-Chip-Versuche mit Algen-Zellen unter Lichtanregung durchgeführt werden, wird eine Sauerstoffproduktionsrate gemessen. Das Kulturmedium wird alkalischer werden, da das CO_2 verbraucht wird.

2.2.3 Tumorzellen

In dieser Arbeit werden hauptsächlich Messungen mit Tumorzellen betrachtet. Deshalb wird im folgenden Abschnitt kurz erklärt, was die Tumorzellen von normalen Zellen unterscheidet.

- Vergleich mit normalen Zellen
  Tumorzellen sind entartete Zellen. Sie stammen aus normalen Zellen, deren DNA beschädigt

wurde. Normale differenzierte tierische Zellen können sich in Kultur durch Mitose maximal 30 bis 60 Mal teilen. Die Tumorzellen können sich im Gegensatz dazu beliebig oft teilen. Tumorzellen haben einen veränderten Metabolismus im Vergleich zu normalen Zellen. In Kultur teilen sich normale Zellen bis sie Kontakt zueinander haben. Das Wachstum von Tumorzellen ist dagegen unkontrolliert: in Kultur wachsen Tumorzellen meistens schneller als normale Zellen und auch übereinander. Oberfläche, Morphologie und Kerne von Tumorzellen können ebenfalls deutliche Unterschiede aufweisen [10].

- Tumoren und ihr Metabolismus
  Wenn Tumorzellen im tierischen Körper nicht vom Immunsystem erkannt werden, bilden sie Tumore. Tumore brauchen viel Energie für ihren Metabolismus, aber sie sind meistens schlecht vaskularisiert (mit Blutgefäßen versorgt) und nicht genug mit Sauerstoff versorgt. Für die Energieproduktion wird deswegen die anaerobe Glykolyse verstärkt, wodurch Milchsäure produziert und der extrazelluläre pH-Wert in der Mikroumgebung der Tumorzellen herabgesetzt wird. Es wurde gezeigt, dass der pH-Wert zwischen 5 und 6, statt 7 für normale Gewebe, liegt, und dass er eine essentielle Rolle bei der Malignität der Tumore spielt [20], [21].

2.3 Zellkulturtechnik

Einige Grundbegriffe der Zellkulturmethoden, die in dieser Arbeit verwendet werden, werden in diesem Abschnitt vorgestellt. Ausführliche Informationen über tierische Zellkulturen finden sich im Buch von R. I. Freshney [5].

- Zellkulturpassage
  Im Labor werden die Zellen in Flaschen kultiviert. Wenn eine bestimmte Zellkonzentration erreicht wird, werden die Zellen passagiert: adhärente Zellen werden durch Enzyme (z. B. Trypsin) von der Flasche und voneinander dissoziiert, verdünnt und in neuen Flaschen kultiviert.

- Zelllinie
  Um eine primäre Zellkultur zu erhalten, wird Gewebe zerkleinert und enzymatisch behandelt. Wenn Zellen der erhaltenen Zellkultur eine beliebige Anzahl von Passagen weiterkultiviert werden können, bilden sie eine so genannte Zelllinie. Zellen einer Zelllinie können durch Division von normalen Zellen in Kultur entstehen. Sie können auch in einer Kultur von normalen Zellen durch Beifügen eines onkogenen Virus oder Moleküls entstehen, oder können auch direkt aus Tumorzellen entstehen. Zelllinien werden von speziellen Laboren aufbewahrt und vertrieben, wie z. B. ATCC (American Type Culture Collection) mit 60000 Zelllinien, oder ECAAC (European Collection of Cell Culture) mit 40000 Zelllinien.

- Zellkulturmedium
  Die Zellen werden in bestimmten Medien kultiviert. Das Kulturmedium für tierische Zellen muß Vitamine, essentielle Aminosäuren, Glukose und Salze (Na^+, K^+, Ca^{2+}, Mg^{2+}, Cl^-, SO_4^{2-}, $HCO3^-$, PO_4^{3-}) enthalten. Es existieren verschiedene definierte Medien: Eagle's MEM Medium (Minimum essential medium), das nur die B-Vitamine als Vitamine enthält, DMEM (Dulbecco modifiziertes Eagle MEM), RPMI (Roswell Park Memorial Institute, entwickelt von Morre et al.), und andere. Seren können ins Medium hinzugefügt werden, wie FCS (Fetal Calf Serum) Kälberserum. Die Seren enthalten Proteine, Wachstumsfaktoren, Hormone (Insulin, Hydrocortison), Mineralien und Nährstoffe (Glukose, Aminosäuren). Farbstoffe, wie Phenol-Rot, können dem Medium hinzugefügt werden, um den pH-Wert optisch kontrollieren zu können. Auch Antibiotika, wie z. B. Gentamycin, werden manchmal dem Medium hinzugefügt, um eventuelle bakterielle Kontaminationen zu verhindern.

2.4 Zellwachstum und Zellsterben

Da Zell-Chip-Systeme es erlauben, Messungen mehrere Tage lang durchzuführen, ist es besonders wichtig, das Wachstum von Zellen in Kultur zu betrachten.

2.4.1 Zellwachstum

Wie im letzten Abschnitt erwähnt, werden die Zellkulturen periodisch passagiert. Wenn adhärente Zellkulturen gerade mit Trypsin behandelt wurden, brauchen sie ungefähr 12 Stunden bis sie ihre Adhäsionsproteine wieder aufgebaut haben. Während dieser Zeit bleibt die Zellzahl konstant. Diese Phase wird Latenz- oder Lag-Phase genannt. Nachher teilen sich die Zellen, bis sie die ganze Zellkulturfläche abdecken. Diese Wachstumsphase wird Log-Phase genannt. Tierische Zellen können sich in der Kultur alle 12 Stunden bei schnell wachsenden Zelllinien, bis 60-72 Stunden bei langsam wachsenden Zelllinien teilen. Nach der Log-Phase teilen sich die Zellen viel langsamer oder gar nicht mehr (bei normalen Zellen), die Kultur ist konfluent. Diese Phase heißt Plateau-Phase. In dieser Phase nimmt die Synthese spezialisierter Proteine zu. Epithel- und Endothel-Zellen bilden Monolayer-Kulturen. Tumorzellen können mehrschichtig übereinander wachsen. In der Abbildung 2.4 werden die Lag-, Log- und Plateau-Phasen dargestellt.

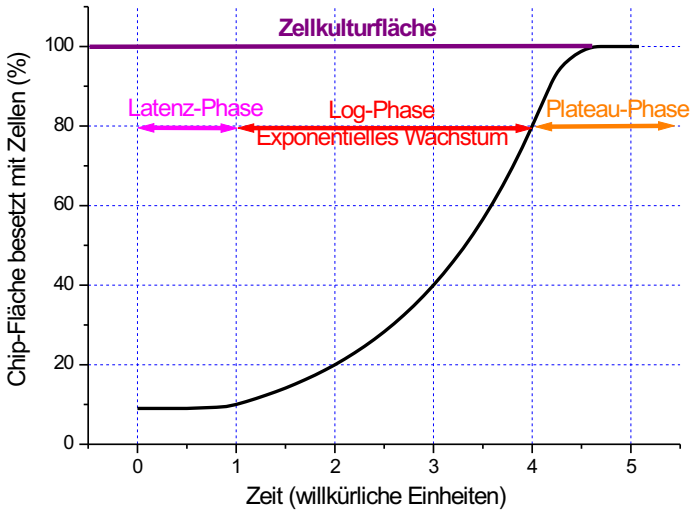


Abbildung 2.4: Zellwachstum: Lag-, Log- und Plateau-Phase

Vor der Teilung einer Zelle wird das genetische Material repliziert, der Inhalt des Cytoplasma wird verdoppelt. Der Zellteilungsprozess ist die Mitose. In Abbildung 2.5 sind mehrere Sequenzen der Mitose einer MCF-7-Zelle aufgezeigt. Die Mitose besteht aus mehreren Phasen: Prophase, Metaphase, Anaphase, Telophase. In der Prophase werden die Chromosomen kondensiert. In der Metaphase werden die Chromosomen aneinandergereiht, wie im 2. Bild der Abbildung 2.5 aufgezeigt. In der Telophase werden die Chromosomen getrennt. Danach werden die Zellen geteilt.

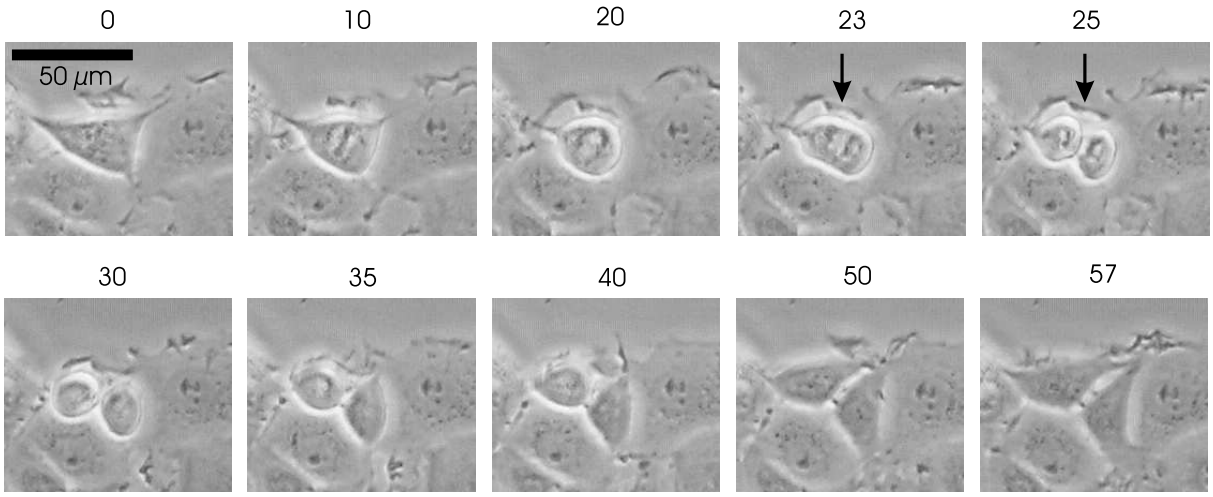


Abbildung 2.5: Phasenkonstrastmikroskopie einer Teilung einer MCF-7-Zelle, Zeit in Minuten

2.4.2 Apoptose und Zellnekrose

Es existieren zwei Arten von Zelltod: Zellnekrose und Apoptose. Die Apoptose ist der genetisch programmierte Zelltod. Die Zellorganellen werden intern zerstört, ihre Destruktion ist intern programmiert. Im Körper erkennen die Makrophagen apoptotische Zellen, absorbieren sie (Phagocytose) und lysieren sie mit Enzymen. Im Gegensatz zur Apoptose führt die Zellnekrose zu einem Platzen der Zelle, die im Körper eine Inflammationsreaktion verursacht.

Für adhärente Zellen wird der Zelltod durch die Impedanz-Sensoren detektiert, da tote Zellen in vitro sich vom Substrat ablösen.

Kapitel 3

Material und Methoden

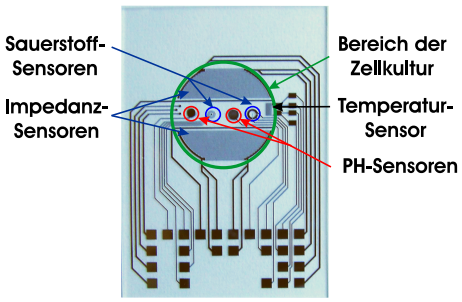
3.1 Sensorchips

Verschiedene Sensorchips wurden am Lehrstuhl in den letzten Jahren entwickelt: Silizium-, Keramik- und Glas-Sensorchips.

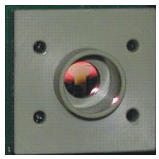
Silizium-Chips werden in Zusammenarbeit mit dem LTE (Lehrstuhl für Technische Elektronik) an der Technischen Universität hergestellt. Ein Silizium-Chip integriert vier pH-Sensoren (ISFETs), einen Sauerstoffsensor und einen Impedanz-Sensor. Ein Silizium-Chip ist $7,4 \times 7,4$ mm groß, die Zellkulturfläche ist 15 mm^2 und das Volumen der Zellkulturkammer beträgt 6 μL. Der Chip wird auf einem PLCC68-Sockel verkapselt.

Keramik-Chips werden von der Firma Heraeus produziert. Sie haben die gleiche Größe wie Silizium-Chips und integrieren zwei pH-Metalloxid-Sensoren, einen Sauerstoffsensor und einen Impedanz-Sensor. Ihre Herstellung ist billiger als die von Silizium-Chips und sie sollen langfristig die Silizium-Chips ersetzen.

Die Glas-Sensorchips werden ebenfalls am Lehrstuhl, bzw. bei Heraeus hergestellt. Sie integrieren zwei IDES, zwei Sauerstoff- und zwei pH-Metalloxid-Sensoren. Ein Glas-Sensorchip ist $24 \times 33,8 \times 0,5$ mm gross. Die Zellkulturfläche ist ungefähr 100 mm^2, das Volumen der Zellkulturkammer beträgt 50 μL. Mit Glas-Sensorchips ist es daher möglich, die Zellen zu mikroskopieren. In Abbildung 3.1 werden ein Glas-Sensorchip und ein verkapselter Silizium-Chip gezeigt.



Multiparametrischer
Biosensorchip auf Glasbasis



Verkapselter multiparametrischer
Biosensorchip auf Siliziumbasis

Abbildung 3.1: Am Lehrstuhl entwickelte und hergestellte Chip-Typen

Die verschiedenen Sensoren werden im nächsten Abschnitt beschrieben.

3.2 Sensoren

3.2.1 PH-Sensoren

3.2.1.1 ISFET-Sensor

Als pH-Sensoren werden ISFET-Sensoren (Ionen Sensitive Feld Effect Transistor) auf den Silizium- und Keramik-Chips integriert. Der erste ISFET wurde von Bergveld im Jahre 1970 erfunden. Nachher wurde dieser Sensortyp weltweit erforscht, wie Lambrechts es erklärt [27]. Der Aufbau des aktuellen Sensors wird in diesem Abschnitt beschrieben und ist in der Abbildung 3.2 gezeigt.

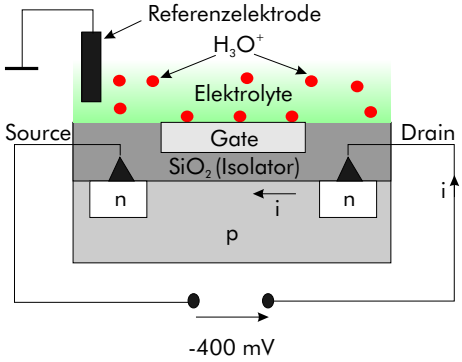


Abbildung 3.2: Aufbau des ISFET-Sensors

Der Sensor ist ein modifizierter MOS-Transistor, dessen Gate nicht metallisiert ist. Das Transistor-Gate und eine Silber/Silberchlorid-Referenzelektrode sind im Kontakt mit der Lösung, deren pH-Wert man bestimmen will. Das Gate besteht aus einer ionensensitiven Schicht und ist ungefähr 100 μm lang und 4 μm breit. Die Protonen binden an das Gate und ändern den Potentialunterschied zwischen Gate und Referenzelektrode. Die Gate-Source-Spannung U_{GS} hängt linear vom pH-Wert ab. Wenn a (Volt/pH) die Sensitivität und b (Volt) den Offset des Sensors darstellen, ergibt sich:

$$U_{GS}(pH) = a \cdot pH + b$$

Der ISFET wird durch 2 Messpunkte kalibriert.

3.2.1.2 PH-Metalloxidsensor

Als pH-Sensoren werden pH-Metalloxidsensoren auf den Glas-Sensorchips integriert. Der Sensor besteht aus einer Referenz- und einer Arbeits-Elektrode. Eine Metalloxid-Schicht, wie Ruthenium-Oxid oder Iridium-Oxid, bedeckt die Arbeitselektrode. Die Referenzelektrode besteht aus Silber/Silberchlorid. Die Referenzelektrode ist direkt in Kontakt mit der Lösung. Die Protonen binden an der Metalloxidschicht und ändern den Potentialunterschied zwischen Referenzelektrode und Arbeitselektrode. Die Spannung zwischen den beiden Elektroden hängt linear vom pH-Wert ab, wie bei den ISFETs und pH-Glaselektroden. Dieser Sensor wurde von unserer Arbeitsgruppe untersucht.

3.2.2 Sauerstoff-Sensoren

Es existieren elektrische und optische Sauerstoff-Sensoren. Für Zell-Chip-Systeme können beide Arten von Sensoren eingesetzt werden. Ihr Funktionsprinzip wird in diesem Abschnitt erklärt.

3.2.2.1 Planarer Sauerstoff-Sensor

Der planare Sauerstoff-Sensor auf den Chips ist ein Drei-Elektroden-Clark-Sauerstoff-Sensor. Der originale Clark-Sauerstoff-Sensor wurde im Jahre 1956 von L. C. Clark erfunden und besteht aus zwei

Elektroden. Die beiden Versionen und der Vorteil des Drei-Elektroden-Sauerstoff-Sensors werden in diesem Abschnitt kurz beschrieben.

- Zwei-Elektroden-Sauerstoff-Sensor
  Der originale Clark-Sensor besteht aus einer Kathode aus Platin oder Gold und einer Anode aus Silber. Eine für Sauerstoff durchlässige Membran bedeckt Elektrolyte (KCl-Lösung) und Elektroden. Es handelt sich um eine amperometrische Messmethode: mit einer konstanten Spannung von -0,7 Volt zwischen Anode und Kathode wird O_2 an der Kathode reduziert. Man misst dann einen Strom, der proportional zum Sauerstoffgehalt in der Lösung ist. In Abbildung 3.3.A werden das Messprinzip und die elektrochemischen Reaktionen dargestellt.

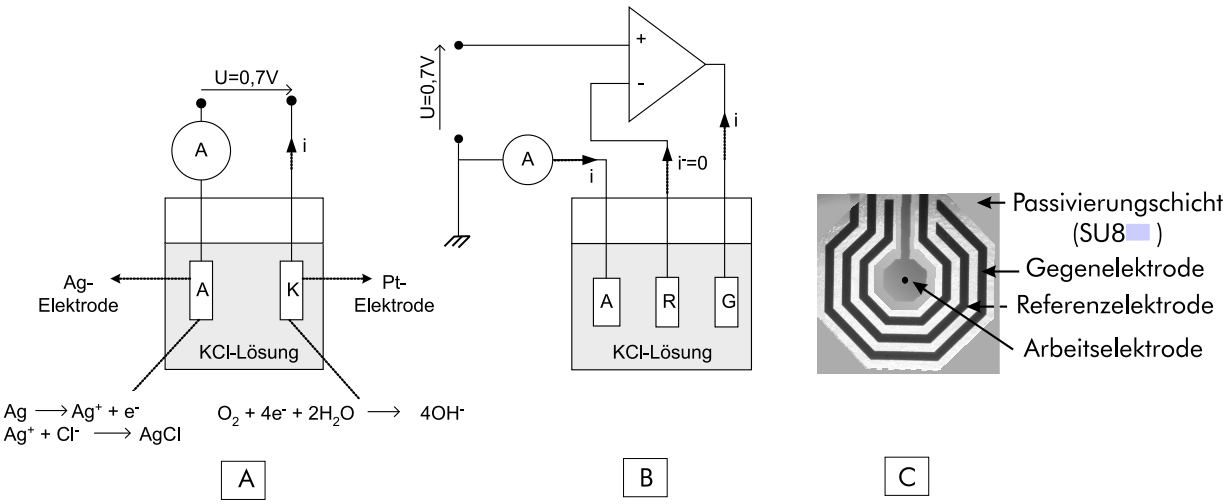


Abbildung 3.3: Abb. A: Zwei-Elektroden-Sauerstoff-Sensor, A: Anode, K: Kathode. Abb. B: Drei-Elektroden-Sauerstoff-Sensor, A: Arbeitselektrode, R: Referenzelektrode, G: Gegenelektrode [27]. Abb. C: Planarer Sauerstoff-Sensor auf einem Glas-Sensorchip. Die Breite der Elektroden ist 50 μm.

Man stellt fest, dass die Anode durch die elektrochemischen Prozesse zerstört wird. Die Lebensdauer des Sensors ist begrenzt. Um diesen Nachteil umzugehen, kann eine „grosse“ Anode benutzt werden. Man kann auch den Aufbau des Sensors ändern, wie im folgenden Punkt erklärt.

- Drei-Elektroden-Sauerstoff-Sensor
  Auf den Sensorchips besteht der planare Sauerstoff-Sensor aus drei Platinelektroden: Arbeitselektrode, Gegenelektrode und Referenzelektrode. Der Strom fließt nur durch Arbeits- und Gegenelektrode, die Referenzelektrode ist hochohmig angeschlossen. Das Prinzip des Drei-Elektroden-Sauerstoff-Sensors ist in Abbildung 3.3.B dargestellt. Da durch die Referenzelektrode kein Strom fließt, wird auch deren chemische Zersetzung verhindet. Eine mikroskopische Aufnahme des Sauerstoff-Sensors auf einem Glas-Sensorchip wird in Abbildung 3.3.C gezeigt.

3.2.2.2 Optischer Sauerstoff-Sensor

Eine andere Methode, die gelöste Sauerstoff-Konzentration zu messen, ist eine optische Methode. Das Prinzip der Messung ist schematisch in Abbildung 3.4 dargestellt, und wurde in [26] und in [25] beschrieben. Der Sensor besteht aus einer Glasfaser, einer Leuchtdiode, und einer Schicht von Fluoreszenz-Molekülen (Luminophoren) am Ende der Glasfaser. Um die gelöste Sauerstoff-Konzentration zu messen, schickt die Leuchtdiode einen Lichtimpuls. Die sensitive Folie absorbiert die Lichtenergie. Wenn keine Sauerstoffmoleküle vorhanden sind, emittieren die Luminophoren das Licht zurück. Wenn Sauerstoffmoleküle vorhanden sind, kommt es zur Kollision zwischen Luminophoren und Sauerstoffmolekülen. Luminophoren transmittieren einen Teil ihrer Energie an die Sauerstoffmoleküle.

Die Amplitude und die Phase des zurückemittierten Lichts sind deshalb von der gelösten Sauerstoff-Konzentration abhängig.

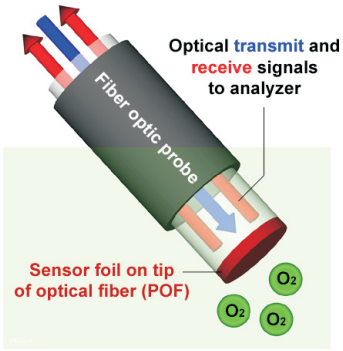


Abbildung 3.4: Prinzip der optischen Messung von gelöster Sauerstoff-Konzentration [83].

3.2.3 Impedanz-Sensor

Wie in der Einführung erklärt, besteht die Zellmembran der tierischen Zellen aus einer Doppelschicht von Lipiden. Die Zellmembran der tierischen Zellen ist bei 10 kHz isolierend. Ein Impedanz-Sensor wird dann benutzt, um Änderungen in Zellmorphologie, Zelladhäsion und des Zellwachstums zu messen. Der Aufbau des Sensors wird im nächsten Abschnitt erklärt.

3.2.3.1 IDES: Inter-Digital Electrode Structure

Der Impedanz-Sensor ist ein Fingerkondensator, der IDES genannt wird: Inter-Digital Electrode Structure. Die Breite der Sensorelektroden beträgt 50 μm, die Höhe 400 nm. Der Abstand zwischen den Elektroden ist 50 μm. Die Glas-Sensorchips integrieren zwei IDES-Sensoren aus Platin, die 12 mm^2 groß sind, und die aus 40 Elektroden, oder 20 Paaren von Elektroden bestehen. Die Elektroden sind 2 bis 4 mm lang, wie in der Abbildung 3.5.A aufgezeigt. Ältere Silizium-Chips integrieren einen IDES-Sensor aus Platin, deren Breite 0,7 mm und Länge 2 mm ist. Der IDES-Sensor besteht aus 20 Elektroden, oder 10 Paaren von Elektroden. In Abbildung 3.5.B wird eine 3D-Darstellung von 4 IDES-Elektroden gezeigt.

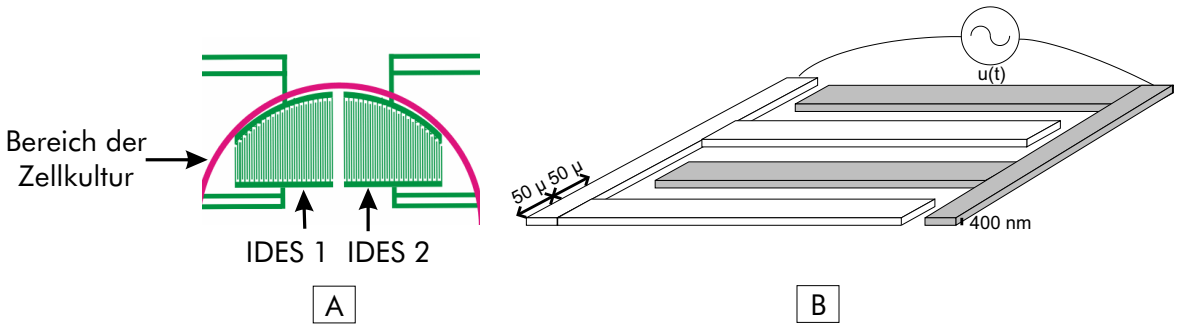


Abbildung 3.5: Abb. A: zwei IDES-Sensoren auf einem Glas-Sensorchip. Abb. B: 3D-Darstellung von 4 IDES-Elektroden.

3.2.3.2 Messprinzip

Um die Impedanz des IDES-Sensors zu messen, produziert ein Gerät eine konstante Wechselspannung am Sensor und misst die Amplitude und die Phase des resultierenden Wechselstroms. Eine Vier-

Punkte-Messung wird benutzt, so dass die Zuleitungswiderstände zwischen dem Messgerät und dem IDES nicht die absoluten Impedanz-Messwerte beeinflussen. Auf den Glas-Sensorchips sind die Zuleitungswiderstände ungefähr 100 Ohm pro Leitung. Die Messfrequenz wurde auf 10 kHz festgelegt, da die Zellen die größten Impedanz-Änderungen bei dieser Frequenz verursachen, wie die Arbeiten von Ehret gezeigt haben [34], [35]. Für die Darstellung der Impedanz werden in dieser Arbeit die Werte einer äquivalenten parallelen Kapazität C (Farad) und eines Widerstands R (Ohm) benutzt, wie in Abbildung 3.6 gezeigt. Diese Darstellung wurde gewählt, weil die Arbeiten von Ehret gezeigt haben, dass der Kapazitätswert bei dieser Darstellung sehr geeignet für die Detektion von Zellwachstum, Zellsterben oder morphologischen Änderungen ist. Die Beziehungen zwischen R und C, der Amplitude und der Phase des Wechselstroms und der Wechselspannung werden in diesem Abschnitt berechnet.

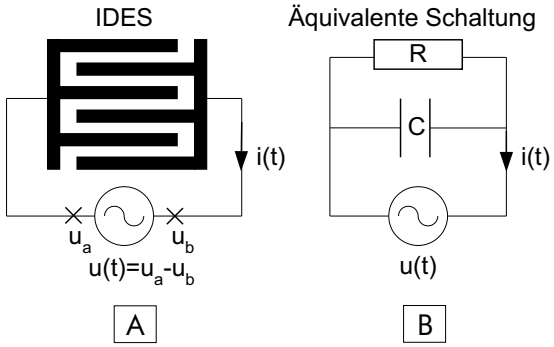


Abbildung 3.6: Darstellung der Impedanz durch eine äquivalente Schaltung mit Widerstand (R) parallel zur Kapazität (C).

- R und C als Funktionen der Amplitude und Phase des Wechselstroms und Wechselspannung:
  Wenn t die Zeit (Sekunden), ω die Kreisfrequenz (Hz), f die Frequenz der Messung (Hz, $\omega = 2\pi f$), i_0 die Amplitude des Stroms (Ampere), u_0 die Amplitude der Spannung (Volt), ϕ der Phasenunterschied zwischen Strom und Spannung (rad.), $\underline{Z}$ die komplexe Impedanz, $\underline{u}$ die komplexe Spannung, $\underline{i}$ der komplexe Strom und j die komplexe Zahl, die $j^2 = -1$ verifiziert, ist, erhält man:

$$\underline{i} = i_0 e^{j(\omega t + \phi)}$$
$$\underline{u} = u_0 e^{j\omega t}$$

  Die Impedanz $\underline{Z}$ ist die Beziehung zwischen Spannung $\underline{u}$ und Strom $\underline{i}$:

$$\underline{Z} = \underline{u}/\underline{i} = (u_0/i_0)e^{-j\phi} \tag{3.1}$$

  Die Impedanz $\underline{Z}$ der äquivalenten Schaltung der Abbildung 3.6.B ergibt sich aus:

$$1/\underline{Z} = 1/R + jC\omega \tag{3.2}$$

  Aus den Gleichungen 3.1 und 3.2 erhalten wir:

$$1/R + jC\omega = (i_0/u_0)(cos(\phi) + jsin(\phi)) \tag{3.3}$$

  Aus der Gleichung 3.3 erhalten wir die gesuchten Beziehungen:

$$R = \frac{u_0}{i_0 cos(\phi)} \quad , \quad C = \frac{i_0 sin(\phi)}{u_0 \omega} \tag{3.4}$$

- Amplitude und Phase von Wechselstrom und Wechselspannung als Funktion von R und C:
  Wenn R und C bekannt sind, können die Phase ϕ und Betrag u_0/i_0 der Impedanz durch die folgenden Gleichungen berechnet werden:

$$\phi = tan^{-1}(RC\omega) \quad , \quad u_0/i_0 = R \cdot cos(\phi) \tag{3.5}$$

3.2.4 Andere Sensoren

Andere wichtige Sensoren für Zellkulturen sind enzymatische Sensoren, wie Glukose- und Laktat-Sensoren. Das Prinzip der Messung von Glukose und Laktat ist das gleiche: es handelt sich wie bei Sauerstoff um eine amperometrische Messung. Für die Glukose-Messung ist die Arbeitselektrode mit einer Schicht vom Enzym Glukoseoxidase bedeckt, das die Transformation von Glukose mit Wasser und Sauerstoff in Glukonsäure und Wasserstoffperoxid H_2O_2 katalysiert. H_2O_2 wird dann an der Elektrode oxidiert, was einen zur Glukosekonzentration proportionalen Strom erzeugt. Für die Messung von Laktat wird Laktatoxidase statt Glukoseoxidase eingesetzt.

3.3 Mikrokammer und Fluidik-System

Wie es in der Einleitung erklärt wurde, werden die Zellen in einer Mikrokammer kultiviert. Um pH- und Sauerstoff-Signale-Änderungen in Bereich von Minuten messen zu können, muss das Volumen der Messkammer klein genug sein. In einem geschlossenen Volumen von 10 Mikroliter und 10^5 tierischen Zellen unter normalen Bedingungen wird der Sauerstoff in 10 Minuten weitgehend verbraucht. Um das Kulturmedium periodisch zu wechseln, wurden in dieser Arbeit peristaltische Pumpen der Firma ISMATEC benutzt. Die Flussrate war typischerweise 100-150 μL/Minute. Die Pumpe war typischerweise 3 Minuten lang eingeschaltet und 8 Minuten ausgeschaltet. In einigen Versuchen war die Pumpe kontinuierlich eingeschaltet und die Flussrate war typischerweise 30 μL/Minute.

3.4 Messgeräte

3.4.1 Einkanal-Messgerät

Das Einkanal-Messgerät ist ein modulares und tragbares Zell-Chip-Messgerät für einen Silizium- oder Keramik-Chip. Das Gerät integriert eine Miniatur-Pumpe mit Behältern, sowie die Elektronik, die aus Verstärkern für den Sauerstoffsensor und den 4 pH-Sensoren und einer Impedanzmesskarte besteht. In einer ersten Version wurden die Messwerte durch eine AD-Wandlerkarte (NI-DAQ PCMCIA 6036 von National Instruments) digitalisiert. In einer neuen Version (IMoLa von J. Wiest) werden die Messwerte durch einen Mikrocontroller mit AD-Wandler digitalisiert und die Verbindung mit dem Computer erfolgt entweder durch die RS232 oder die Bluetooth-Schnittstelle.

3.4.2 Glas-Sensorchip-Messgerät

Das Glas-Sensorchip-Messgerät ist ein Messplatz für zwei Glas-Sensorchips. Die zwei Glas-Sensorchips werden in eine Metall-Doppelkammer eingesetzt, die auf den Tisch eines invertierten Mikroskops gestellt wird, wie in Abbildung 3.7 gezeigt. Der Tisch und die Kulturmediumflaschen werden durch ein 37°C-Wasserbad mit Pumpe erwärmt. Die Doppelkammern werden am Lehrstuhl hergestellt.

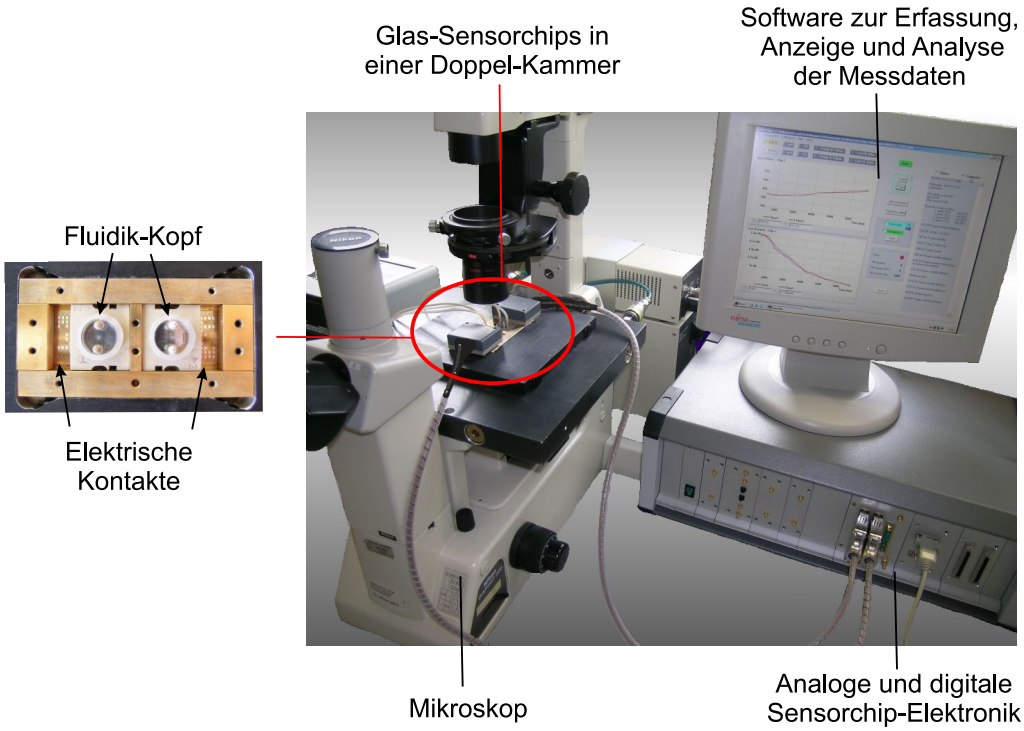


Abbildung 3.7: Aufbau des Glas-Sensorchip-Messplatzes

In dieser Arbeit wurden ein LCR-Meter (SR715 von Stanford Research Systems) und ein Multiplexer benutzt, um die Impedanz der 4 IDES zu messen (siehe 3.4.4). Für die Messung von pH, gelöstem Sauerstoff und Temperatur wurde ein Messgerät benutzt, das die Verstärker für die Sensoren und einen Mikrocontroller mit AD-Wandler enthält. Das Messgerät wurde am Lehrstuhl gebaut. In Abbildung 3.8 wird die digitale Schaltung des Messgeräts aufgezeigt. Das Gerät kommuniziert durch die RS232-Schnittstelle mit dem Computer. Der Mikrocontroller (PIC16F873 von der Firma Microchip) kommuniziert mit 2 AD-Wandler (AD7706BN von Analog Devices) mit dem SPI-Protokoll (Serial Peripheral Interface). Er schickt und empfängt Daten vom Computer durch die serielle Schnittstelle (SUB-D 9-BU). Der Mikrocontroller wurde in C programmiert. Das Programm wurde dann mit der Software „PCW C Compiler“ kompiliert. Der kompilierte Code wurde durch eine PIC-Programmierer-Schaltung (K8048 von VELLEMAN) in das EEPROM des Mikrocontrollers geschrieben. In der digitalen Schaltung wurde ein MAX232-Chip benutzt, um die digitalen Signale des Computers von $\pm$12V auf 0-5 Volt und umgekehrt zu konvertieren.

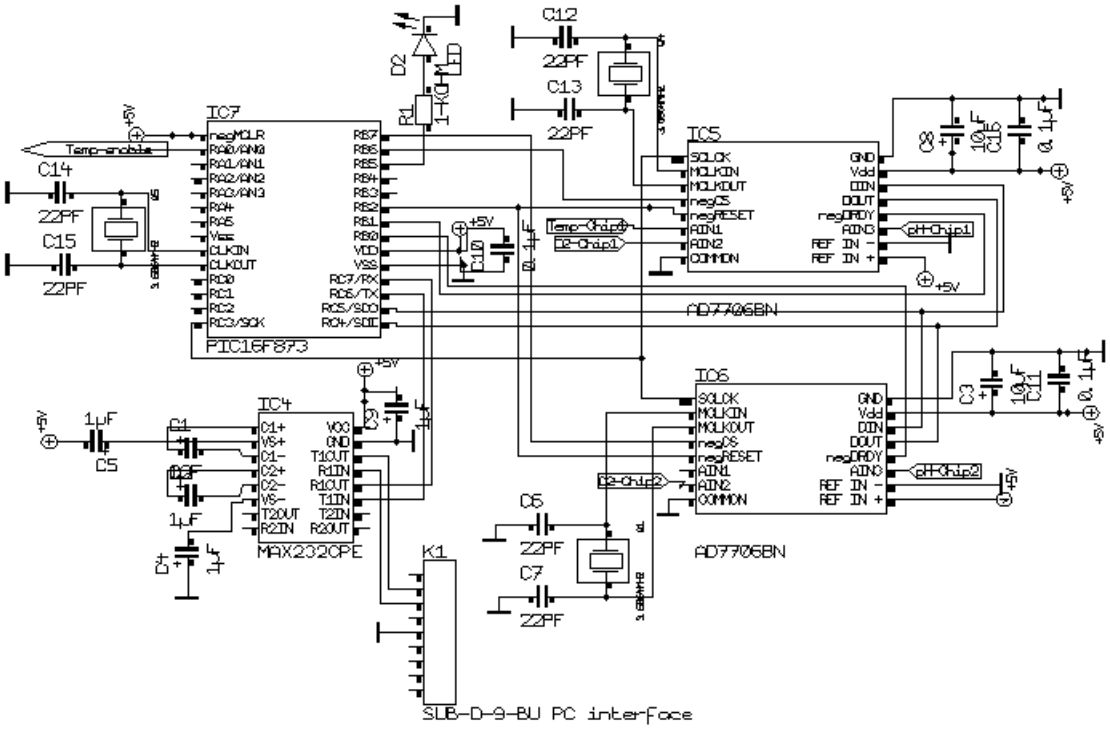


Abbildung 3.8: Digitale Schaltung des Glas-Sensorchip-Messgeräts

3.4.3 Sechsfach-Tester

Als Sechsfach-Tester wurde das „TUM-Screen“-Gerät entwickelt, das in Abbildung 3.9 dargestellt ist. Das Gerät integriert sechs Biomodule für Silizium-Chips, einen Computer mit Touch-Screen und eine peristaltische Pumpe mit mehreren Behältern für das Kulturmedium. Ein LCR-Meter (SR715 der Firma „Stanford Research Systems“) misst die Impedanz der sechs IDES und die sechs Biomodule integrieren die Verstärker für die O_2- und pH-Sensoren. Im Computer werden die Messsignale der Biomodule durch eine AD-Wandlerkarte digitalisiert.

Abbildung 3.9: Sechsfach-Tester „TUM-Screen“ für Silizium-Sensorchips

3.4.4 LCR-Meter

Ein LCR-Meter (SR715 von Stanford Research Systems) wurde zur Messung der Impedanz beim Glas-Sensorchip-Messplatz und beim TUM-Screen benutzt. Dieses Gerät kann konstante Wechselspannungen von 0,1, 1, oder 10 Volt mit einer Frequenz von 10, 100, 1000 oder 10000 Hz produzieren. Für die Messung der Impedanz der IDES mit Zellen wird eine Wechselspannung von 100 mV, 10 kHz produziert. Eine langsame Messrate mit 400 Messperioden wird durchgeführt, eine Messung dauert also $t = 400/10000 = 40\ ms$. Ein Vorwiderstand von 100 kOhm wird in der stromführenden Leitung eingesetzt, so dass die Stromdichte geringer als 20 $\mu A/cm^2$ ist.

3.4.5 Elektronische Zellzähler

Um eine Auswertung der Zellzahl nach einem Versuch zu erhalten, wurde der elektronische Zellzähler Casy®1 der Firma „Schärfe System“ benutzt [76]. Dieses Gerät evaluiert den Durchmesser von Partikeln mit einer Auflösung von 0,1 μm. Eine Bestimmung der Kernzellzahl erhält man durch das folgende Protokoll: Die Zellen werden zuerst mit 900 μL Hypotoner-Puffer eine halbe Stunde inkubiert, was ein Anschwellen der Zellen verursacht. Dann wird 100 μL Lysis-Lösung hinzugefügt, wodurch die Zellen platzen. Schließlich wird die Lösung mit Casy-Puffer (z. B. 20x) verdünnt, um eine Lösung von Zellkernen zu erhalten, deren Durchmesser und Konzentration mit dem Zellzähler evaluiert werden können. Die Zellkerne haben einen typischen Durchmesser von 5 bis 12 μm.

3.5 Zelllinien und Kulturmedium

Für die in dieser Arbeit vorgelegten Ergebnisse, wurden die folgenden Zelllinien benutzt: LS174T, aus einem menschlichen Kolon-Adenokarzinom; HeLa, aus einem menschlichen Cervix-Adenokarzinom; MDA-MB-231 und MCF-7, zwei menschlichen Brust-Tumor-Zelllinien; L929, Maus-Fibrosarcom-Zellen.

Als Kulturmedium wurde ein „Dulbecco's Modified Eagle's“ Medium (Sigma, München) mit Zugabe von 5 bis 10% FCS (Seromed, Biochrom, Berlin) und Gentamycin verwendet. Die Puffer HEPES und Natriumbikarbonat wurden nicht ins Kulturmedium gegeben, um eine minimale Pufferkapazität zu haben (siehe Kapitel 4).

Am Ende der Versuche wurden die Zellen mit 0,1% „Triton X100“ getötet. „Triton X100“ ist ein Detergens, der die Zellmembran ablöst. Er enthält keine Ionen und sein Einfluss auf die Impedanz-Werte ist unwesentlich.

3.6 Mikroskop und Kamera

Für die Versuche mit Glas-Sensorchips wurde die Mikroskope „Nikon Eclipse TS100“ und „Diaphot 200“ benutzt. Mikroskopische Bilder und Videos wurden mit einer digitalen Kamera „Nikon Coolpix 5400“ aufgenommen.

3.7 Software

Für die Herstellung der Software dieser Arbeit wurde der Compiler „Borland C++ Builder 6.0“ verwendet. Für die Herstellung der Graphiken der schriftlichen Arbeit wurde die Software „Microcal Origin 6.0“ benutzt. Für die Herstellung der Zeichnungen wurden die Software „CorelDraw Graphics Suite 12“ und „Microsoft Visio“ verwendet. Für die Herstellung der schriftlichen Arbeit wurde die open-source-Distribution von TEX für Windows „MiXTeX“ verwendet [81]. Als Editor wurde „Vim“ benutzt [85], der frei vertrieben wird.

Kapitel 4

Messsignale und ihre Bearbeitung

4.1 Einführung

Im vorherigen Kapitel wurde erklärt, dass die Zell-Chip-Systeme mehrere Sensoren integrieren, die durch eine Auswertelektronik digitale Messsignale zum Computer ausliefern. Diese digitalen Messsignale werden dann verarbeitet und dem Benutzer gezeigt. Mehrere Algorithmen wurden für die digitale Signalverarbeitung von Zell-Chip-Versuchen implementiert. Ihre Funktionen und Beziehungen werden in Abbildung 4.1 aufgezeigt und hier kurz beschrieben.

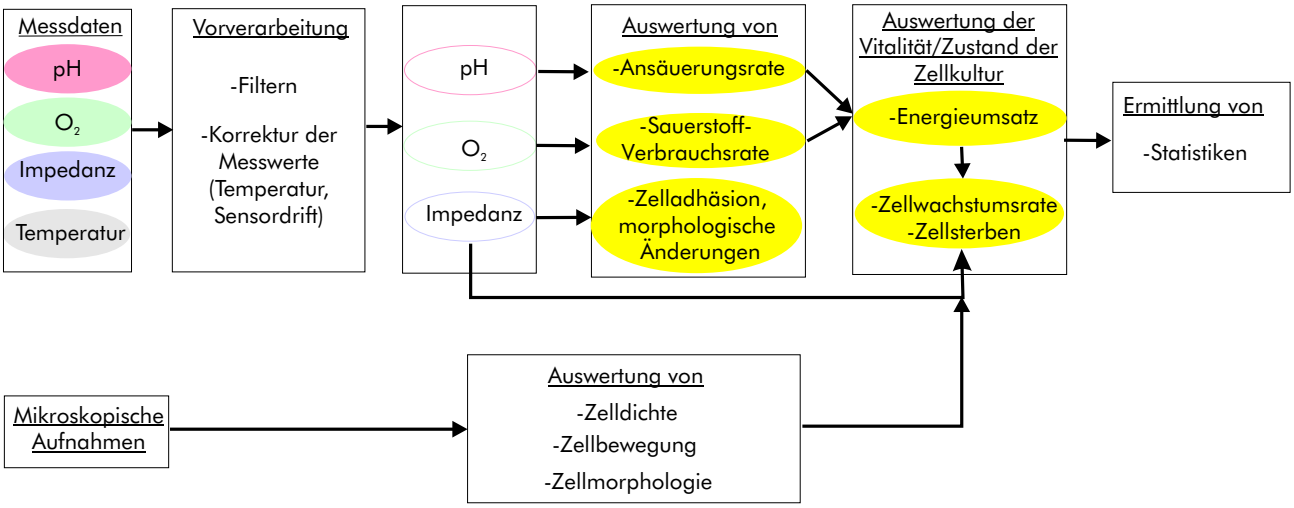


Abbildung 4.1: Digitale Signalverarbeitung für Zell-Chip-Messsignale

In einer ersten Stufe, genannt „Vorverarbeitung“, haben die Algorithmen zum Ziel, die Sensordaten zu filtern, Sensordrifts zu korrigieren, oder Temperaturkorrekturen zu berechnen, wenn es notwendig ist. In einer zweiten Stufe werden verschiedene Eigenschaften der Zellkultur und des Zellmetabolismus ausgewertet: Sauerstoffverbrauchsrate oder Produktionsrate, Ansäuerungsrate, Änderungen in der zellulären Adhäsion oder Morphologie. In einer dritten Stufe werden die Vitalität oder der Zustand der Zellkultur ausgewertet: Zellwachstumsrate, Zellsterben, Energieumsatz werden ermittelt. In einer letzten Stufe werden statistische Auswertungen durchgeführt. In dieser Stufe wird zum Beispiel die Streuung von identischen Messreihen ausgewertet. Es wird auch abgeschätzt, wann und welcher Effekt von einem Wirkstoff auf den Zellmetabolismus ausgeht. In Abbildung 4.1 werden noch dazu Signalverarbeitungen für mikroskopische Aufnahmen dargestellt. Es wurde im vorherigen Kapitel erläutert, dass Zell-Chip-Versuche auf Glas-Sensorchips mit Lichtmikroskopie kombiniert werden können. Dadurch können mikroskopische Bilder und Filme parallel zum Zell-Chip-Versuch aufgenommen werden, die verarbeitet werden, um Auswertungen der Zelldichte, Zellbewegung oder Morphologie zu erlangen.

In diesem Kapitel wird zuerst die Auswertung der Sauerstoff-Verbrauchsrate und der Ansäuerungsrate erklärt. Dann werden Impedanz-Messergebnisse aufgezeigt und digitale Verarbeitungsalgorithmen der Impedanz-Werte präsentiert. In einem dritten Teil werden die statistischen Auswertungen erklärt. In einem vierten Teil wird die Auswertung der Vitalität erklärt. Im Schlussteil wird ein digitaler Bildverarbeitungsalgorithmus kurz präsentiert.

4.2 Gelöste Sauerstoffkonzentration

4.2.1 Einflussfaktoren auf die gelöste Sauerstoffkonzentration

Verschiedene Parameter haben einen Einfluss auf die Konzentration von gelöstem Sauerstoff in einer Flüssigkeit: die Temperatur, der Druck, der Salzgehalt. In diesem Abschnitt werden diese Einflüsse ausgewertet.

- Beziehung zwischen Temperatur, Druck und gelöster Sauerstoffkonzentration:
  Die folgende Gleichung [83] gibt eine Beziehung zwischen diesen Parametern für luftgesättigtes Wasser. Die Bedeutungen und Einheiten der Symbole sind in Abbildung 4.2 angegeben.

$$[O_2](mg/L) = \frac{p_{atm} - p_w(T)}{p_N} \cdot 0,2095 \cdot \alpha(T) \cdot 1000 \cdot \frac{M(O_2)}{V_M} \tag{4.1}$$

Symbol	Einheit	Bedeutung
T	$°C$	Temperatur
$[O_2]$	mg/L	gelöste Sauerstoffkonzentration
p_N	1013 mBar	Standard-Druck: atmosphärischer Druck auf Meereshöhe
p_{atm}	mBar	Atmosphärischer Druck
$p_w(T)$	mBar	Wasserdampf-Druck
$\alpha(T)$		Bunsen-Absorptions-Koeffizient
$M(O_2)$	32g/Mol	Molar-Masse von O_2
V_M	22,414L/Mol	Molar-Volumen

Abbildung 4.2: Bedeutung und Einheiten der Gleichungsymbole aus (4.1).

- Einfluss von T, $p_{atm} = 1020$ mBar
  Je höher die Temperatur ist, desto kleiner ist die gelöste Sauerstoffkonzentration. Eine Zunahme der Temperatur um 5 Grad von $35°C$ ($[O_2] = 7,01$ mg/L) auf $40°C$ ($[O_2] = 6,48$ mg/L), verursacht eine Abnahme der gelösten Sauerstoffkonzentration von 0,53 mg/L.
  $p_w(35°C) = 56,3$ mBar, $\alpha(35°C) = 24,63 \cdot 10^{-3}$, $p_{atm} = 1020$ Bar, $[O_2] = 7,01$ mg/L
  $p_w(40°C) = 73,7$, $\alpha(40°C) = 23,16 \cdot 10^{-3}$, $p_{atm} = 1020$ mBar, $[O_2] = 6,48$ mg/L
  Die Variation der gelösten Sauerstoffkonzentration zwischen $35°C$ und $40°C$ ist über $-\frac{0,53 \cdot 100}{5 \cdot 6,48}$ =-1,63%/°C.

- Einfluss von p_{atm}, $T = 40°C$
  Je größer der atmosphärische Druck ist, desto größer ist die gelöste Sauerstoffkonzentration. Eine Zunahme des atmosphärischen Drucks um 5 mBar ($5 \cdot 10^5$ Pascal) bei $40°C$, verursacht eine Zunahme der gelösten Sauerstoffkonzentration von 0,02 mg/L:
  Mit $p_w(40°C) = 73,7$, $\alpha(40°C) = 23,16 \cdot 10^{-3}$, $p_{atm} = 1025$ mBar, $[O_2] = 6,50$ mg/L
  Die Variation der gelösten Sauerstoffkonzentration zwischen 1020 und 1025 mBar beträgt unter $\frac{0,02 \cdot 100}{5 \cdot 6,48}$ =0,06%/mBar.

- Einfluss des Salzgehalts
  Je salziger das Wasser ist, desto kleiner ist die gelöste Sauerstoffkonzentration. In Zell-Chip-Versuch bleibt der Salzgehalt des Kulturmediums konstant. Man nimmt zum Beispiel 0,9% NaCl für Versuche mit menschlichen Zellen.

Für die Zell-Chip-Messungen wird die Temperatur geregelt. Für Versuche mit menschlichen Zellen werden die Zell-Chip-Systeme in Inkubatoren gestellt, die die Temperatur auf $37 \pm 0,2°C$ regeln können. Der Einfluss des atmosphärischen Drucks ist viel kleiner als der der Temperatur. Die Fehlerquote, die diese Einflüsse bei den Messwerten verursachen, liegt bei insgesamt weniger als 5%. Sie kann deshalb im Folgenden als vernachlässigbar angenommen werden.

4.2.2 Evaluierung der Sauerstoff-Verbrauchsrate oder -Produktionsrate

Sauerstoff wird von tierischen Zellen verbraucht. Die Pflanzen- und Algen-Zellen produzieren Sauerstoff, wenn sie beleuchtet werden, sonst verbrauchen sie Sauerstoff. In diesem Absatz wird erklärt, wie die Sauerstoff-Verbrauchsrate ausgewertet werden kann. Für die Produktionsrate gelten dieselben Auswertungen.

In den Zell-Chip-Versuchen wird das Kulturmedium für die Zellen periodisch durch eine Pumpe erneuert, typischerweise alle 10 Minuten. Wenn das Kulturmedium gerade gewechselt wurde, enthält das Medium die maximale gelöste Sauerstoff-Konzentration, die der Konzentration von Sauerstoff in der Kulturmediumflasche entspricht. Dann wird die Sauerstoff-Konzentration sinken bis das Kulturmedium wieder gewechselt wird. Die Sauerstoffverbrauchsrate ist von der Zellzahl und der metabolischen Aktivität der Zellen abhängig. Abbildung 4.3 zeigt typische Sauerstoff-Messzyklen. In diesem Versuch wurde ein Wirkstoff hinzugefügt (Cytochalasin B), der Einfluss auf die Sauerstoff-Verbrauchsrate der Zellen hat.

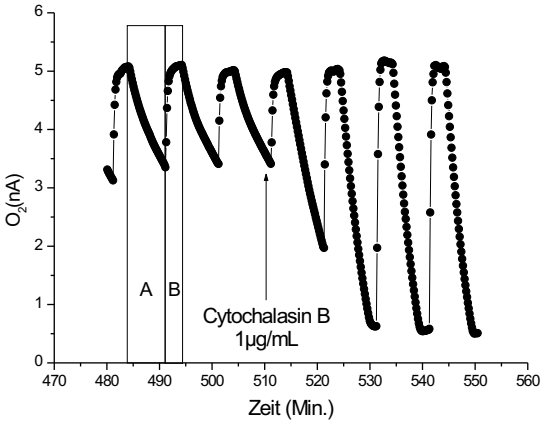


Abbildung 4.3: Verlauf des planaren Clark-Sauerstoff-Sensorstroms. Messung des Einflusses von Cytochalasin B auf die Sauerstoffverbrauchsrate der Tumor-Zellen LS174T. Phase A(Pumpe eingeschaltet) und B(Pumpe ausgeschaltet) definieren ein Messzyklus.

Für die Auswertung der Sauerstoff-Verbrauchsrate werden zwei Regressionsrechnungen präsentiert: lineare und exponentielle Regressionen.

- Regressionsrechnungen:
  Die Regressionsrechnung hat zum Ziel, eine Punktwolke (x_i, y_i) durch eine Funktion $f(x, a, b, ...)$ anzunähern. Um die Parameter (a, b, ...) auszuwerten, wird die Summe der quadratischen Fehler $e(a, b, ...)$ berechnet und minimiert. Mit zwei Parameter (a,b) erhalten wir die folgenden

Gleichungen:

$$e(a,b) = \sum_{i=1}^{n} (y_i - f(x_i, a, b))^2 \tag{4.2}$$

$$\frac{\partial e}{\partial a}(a,b) = 0 \quad und \quad \frac{\partial e}{\partial b}(a,b) = 0 \tag{4.3}$$

Für eine lineare Regression ist $f(x,a,b) = ax + b$. Nachdem die Gleichungen (4.3) gelöst werden, erhält man die Werte von a und b. Wenn $m_x = \sum_{i=1}^{n} x_i/n$ der Mittelwert der X-Koordinaten und $m_y = \sum_{i=1}^{n} y_i/n$ der Mittelwert der Y-Koordinaten sind, ist:

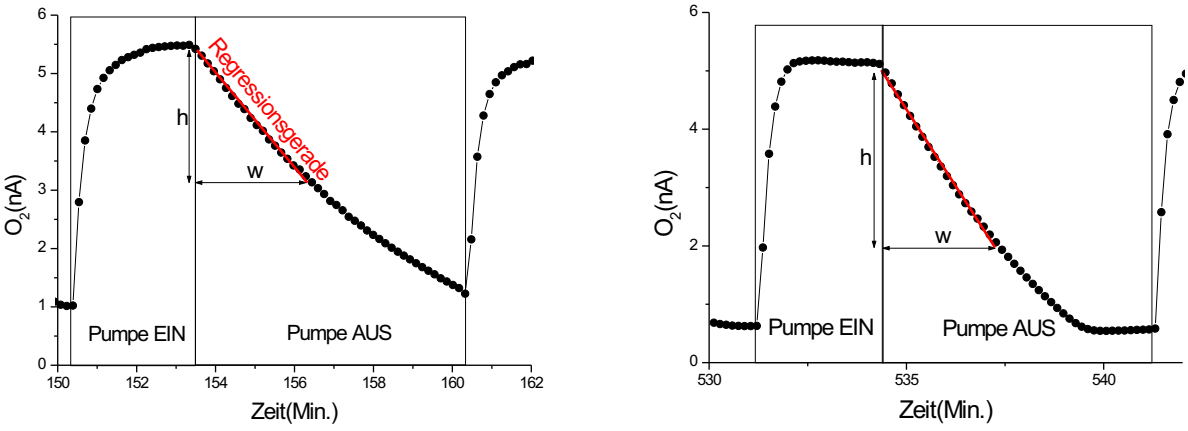
$$a = \frac{\sum_{i=1}^{n} (x_i - m_x)(y_i - m_y)}{\sum_{i=1}^{n} (x_i - m_x)^2} \tag{4.4}$$

und:

$$b = m_y - a \cdot m_x \tag{4.5}$$

Für nicht lineare Regressionen ist das Minimieren des quadratischen Fehlers nicht immer auf analytischem Wege lösbar. Dann werden numerische Methoden eingesetzt.

- Auswertung der Sauerstoffverbrauchsrate durch die Steigung einer Regressionsgerade:
  Eine Möglichkeit zur Bestimmung der Sauerstoffverbrauchsrate ist, die Steigung einer Regressionsgeraden einige Minute lang nach dem Pumpstop zu berechnen. In Abbildung 4.4 sind zwei Vergrößerungen des Sauerstoffsignals aus Versuch 4.3 vor und nach dem Beifügen des Wirkstoffes dargestellt. Die Regressionsgeraden wurden auf den Graphiken eingezeichnet. Man beobachtet, dass die gelöste Sauerstoff-Konzentration nicht ganz linear sinkt. Deshalb wird im folgenden Punkt eine andere Regressionsanalyse der Sauerstoffverbrauchsrate präsentiert.



Sauerstoffsensorstrom
vor dem Hinzufügen von Cytochalasin B

Sauerstoffsensorstrom
nach dem Hinzufügen von Cytochalasin B

Abbildung 4.4: Vergrösserung von zwei Bereichen der Messkurve in Abb. 4.3. Die Sauerstoffverbrauchsrate a wird durch die Steigung einer Regressiongerade ermittelt: a(nA/Min.)$= \frac{h}{w}$

- Auswertung der Sauerstoffverbrauchsrate durch eine exponentielle Regression
  Die Verbrauchsrate von Sauerstoff in den Zellen hängt von der Konzentration mehrerer Moleküle im extrazellulären Raum, wie Sauerstoff und Glukose, und vom Zustand der Zellen ab. Wenn man

den Verbrauch von Sauerstoff als eine chemische Reaktion erster Ordnung modellieren würde, wäre die Geschwindigkeit der Reaktion zu jedem Zeitpunkt proportional zur Sauerstoffkonzentration. In diesem Fall würde die Sauerstoffkonzentration exponentiell sinken. Wenn $O(t)$(Mol/L) die gelöste Sauerstoff-Konzentration, t die Zeit in Minuten, k der Anteil an Sauerstoff, der pro Zeiteinheit verbraucht wird (30 Prozent pro Minute z. B.), α die Konstante gleich $\alpha = \ln(1-k)$, erhalten wir die folgende Gleichung:

$$O(t) = O(0)\exp(\alpha \cdot t)$$

Die Sauerstoffverbrauchsrate k kann dann durch eine exponentielle Regression ausgewertet werden. Berechnungsmethode und -ergebnis werden hier vorgestellt.

– Berechnung
Hier wird angenommen, dass die exponentielle Kurve die Gleichung $f(x) = ae^{bx} + c$ hat (siehe Abbildung 4.5); a, b, c und x sind reelle Zahlen. $x \geq 0$, $b < 0$, $c \geq 0$. Die Konstante c bezeichnet einen Offset des Sauerstoffsensors.
Wenn $(x_i, y_i), i = 1...2n$ die Koordinaten der gemessenen Messpunkte sind, dann sind: $\forall i$: $y_i = ae^{bx_i} + c \Leftrightarrow \ln(y_i - c) = \ln(a) + bx_i$. Der gesuchte Parameter ist $b = \ln(1-k)$. Um b auszuwerten wird die Steigung der Regressionsgerade mit den Punkten $(x_i, \ln(y_i - c)), i = 1...2n$ berechnet.

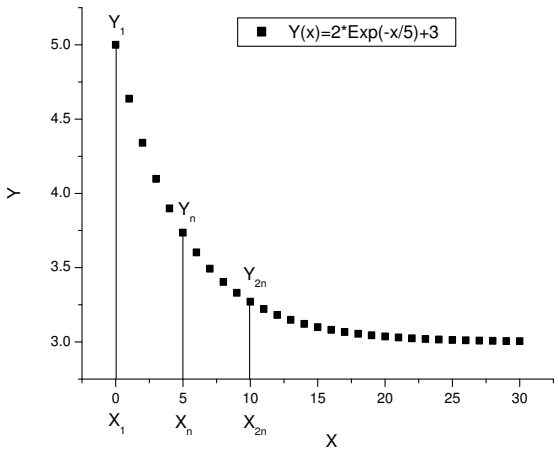


Abbildung 4.5: Exponentielle Funktion, $(x_1, y_1)...(x_{2n}, y_{2n})$ sind die Messpunkte.

Bevor die Steigung der linearen Regression der Punkte $(x_i, \ln(y_i - c)), i = 1...2n$ berechnet werden kann, muß die Konstante c ausgewertet werden. Für die Auswertung von c wird angenommen, dass die Messwerte regelmäßig erfasst wurden und $\forall i$: $x_i = iT$, mit T als reelle Konstante. Wir erhalten dann: $x_{2n} = 2x_n$. Wenn $A = \dfrac{y_{2n} - y_n}{y_n - y_1}$ ist, ist die Konstante c gleich:

$$c = y_1 - \frac{y_n - y_1}{A - 1} \tag{4.6}$$

Man hat tatsächlich:

$$y_1 = Y(0) = a + c$$
$$y_n = Y(x_n) = ae^{bx_n} + c$$
$$y_{2n} = Y(x_{2n}) = ae^{bx_{2n}} + c = ae^{2bx_n} + c$$

Man erhält dann:

$$\frac{y_{2n} - y_n}{y_n - y_1} = \frac{ae^{bx_n}(e^{bx_n} - 1)}{a(e^{bx_n} - 1)} = e^{bx_n} = A$$

Der gesuchte Parameter c ist dann gleich:

$$c = y_1 - a = y_1 - \frac{y_n - y_1}{e^{bx_n} - 1} = y_1 - \frac{y_n - y_1}{A - 1}$$

Das Minimieren der Summe der quadratischen Fehler für eine exponentielle Regression ($f(x) = ae^{bx} + c$) kann nur durch eine numerische Methode erfolgen. Die präsentierte Methode (teilweise aus [70]) wird dagegen analytisch ausgedrückt und ist eine sehr gute Approximation, auch wenn die Summe der quadratischen Fehler nicht minimiert wird.

– Ergebnis
In der linken Abbildung 4.6 werden die Regressionspunkte für drei Messzyklen des Versuchs aus Abb. 4.3 gezeigt. Man kann beobachten, dass die Regressionspunkte mit den Messpunkten vermischt sind: die Regression-Abweichung ist sehr klein. In der rechten Abbildung werden die Regressionspunkte für drei andere Messzyklen aufgezeigt, in denen die Sauerstoffverbrauchsrate größer war. Die Regression-Abweichung ist deutlich größer.

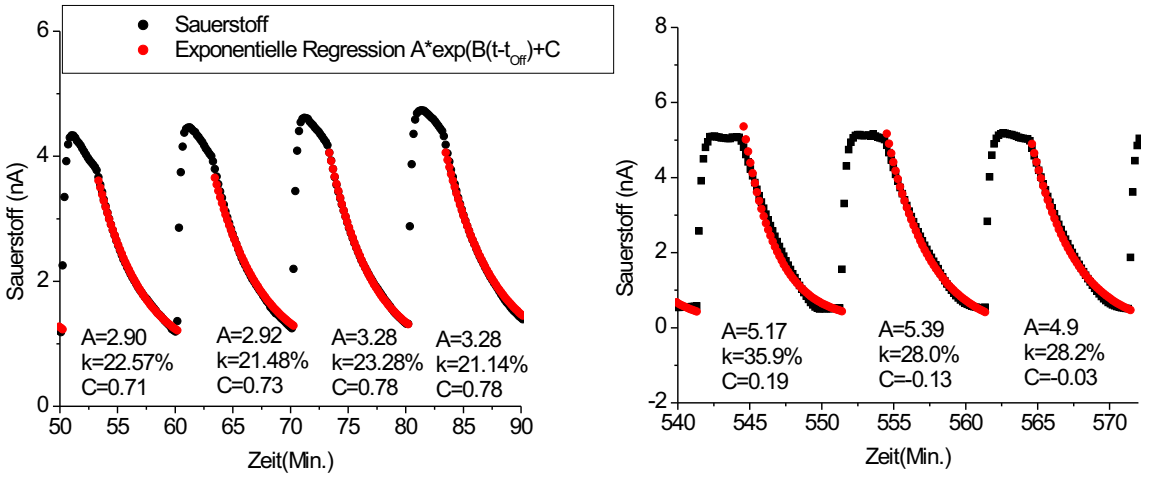


Abbildung 4.6: Sauerstoffmesszyklen vor (links) und nach dem Hinzufügen von Cytochalasin B (rechts) (Versuch 4.10). Die Sauerstoffverbrauchsrate k wird durch eine exponentielle Regression berechnet. $k(\%/\text{Min.}) = 1 - e^{-B}$

In dieser Arbeit wird die Sauerstoffverbrauchsrate aber durch die Berechnung der Steigung der linearen Regression erhalten, wie in Abbildung 4.4 dargestellt, da der Verlauf des Sauerstoffverbauchs von seiner Geschwindigkeit abhängt und die exponentielle Regressions-Abweichung für bestimmte Messzyklen der Versuche zu hoch werden kann.

4.3 Extrazellulärer pH-Wert

Die Zell-Chip-Systeme verfügen über pH-Sensoren, mit denen eine Evaluierung der Ansäuerungsrate der Zellen durchgeführt wird. Der pH-Wert ist als Logarithmus zur Basis 10 der Konzentration von Protonen $[H^+]$ in Mol/L definiert:

$$pH = -\log_{10}([H^+])$$

Da die Zellen Säuren in ein komplexes Kulturmedium auswerfen, das mehrere schwache Säuren und Basen enthält, wird in diesem Kapitel zuerst die Pufferkapazität des Kulturmediums definiert und evaluiert. Danach wird die Berechnung der Ansäuerungsrate erklärt.

4.3.1 Pufferkapazität

In diesem Abschnitt wird zuerst die Pufferkapazität definiert, danach wird die Formel für die Berechnung der Pufferkapazität einer schwachen Säure-Lösung gegeben. Zum Schluss wird die Pufferkapazität verschiedener Kulturmedien evaluiert.

- Definition:
  Wenn n Mol von Protonen H^+ einer Lösung beigefügt werden, die schwache Säuren oder Basen enthält, reagieren einige Protonen mit der schwachen Base. Die Abnahme des pH-Werts ist dabei geringer als die Abnahme des pH-Werts, wenn es keine schwache Säure gibt. Die Pufferkapazität β ist definiert als die Beziehung zwischen der Menge der beigefügten Säure n(Mol/L) und der pH-Wert-Änderung:
  $$\beta = -\frac{dn}{dpH} \tag{4.7}$$

- Pufferkapazität einer Lösung, die eine einzige schwache Säure enthält:
  Bei einer bestimmten Temperatur T ist eine schwache Säure AH durch ihre Dissoziationskonstante $K_a(T)$ charakterisiert. Je stärker die Säure ist, desto größer ist die Dissoziationskonstante.
  $$AH \overset{K_a}{\longleftrightarrow} A^- + H^+ \tag{4.8}$$
  Zum Beispiel ist der pK_a ($pK_a = -\log_{10}(K_a)$) von Bikarbonat (H_2CO_3/HCO_3^-) gleich 6,1. Der pK_a von Phosphat ($H_2PO_4^-/HPO_4^{2-}$) ist gleich 6,8.

  Die Beziehung zwischen den Konzentrationen der verschiedenen Spezies und der Dissoziationskonstante wird durch die Henderson-Hasselbach-Gleichung (4.9) gegeben.
  $$K_a = \frac{[A^-][H^+]}{[AH]} \tag{4.9}$$

  Wenn $A = n_{A^-} + n_{AH}$ die Menge der schwachen Säure (Konstante) ist, gibt Gleichung 4.10 die Pufferkapazität β an (Owicki et al. [46]).
  $$\beta = \frac{A \cdot \ln(10) \cdot 10^{pK_a - pH}}{(1 + 10^{pK_a - pH})^2} \tag{4.10}$$

  Die Gleichung 4.10 zeigt, dass die Pufferkapazität maximal ist, wenn der pH-Wert gleich der pK_a ist. Für eine Lösung, die mehrere schwachen Säuren und Basen enthält, ist die Pufferkapazität gleich der Summe der Kapazitäten aller Säuren und Basen. Da das Kulturmedium eine komplexe Lösung ist, ist eine theoretische Berechnung der Pufferkapazität nicht einfach. Es ist viel einfacher die Pufferkapazität auszuwerten, indem man eine Dosierung durchführt, wie es in Abbildung 4.7 gezeigt wird.

- Evaluierung der Pufferkapazität von DMEM
  Wie im Kapitel 2 erwähnt, wurde das Kulturmedium DMEM mit 5 bis 10 % FCS für die Versuche in dieser Arbeit verwendet. Die Pufferkapazität dieses Mediums mit 5%, 10% und ohne FCS wurde evaluiert. Dafür wurde eine Dosierung mit einer 1-Molar-HCl-Lösung durchgeführt, wie in Abbildung 4.7 dargestellt.

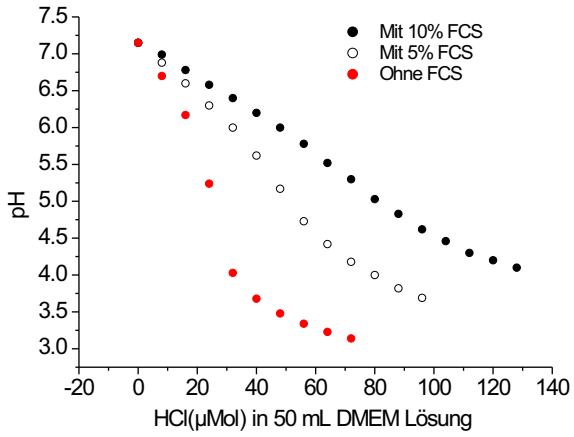


Abbildung 4.7: Dosierungskurve, um die Pufferkapazität von DMEM zu evaluieren: in 50 mL DMEM ohne Bikarbonat, mit Gentamycin, mit 5%, 10% und ohne FCS wurden Tropfen von 1 μL einer 1-Molar-HCl-Lösung beigefügt und der pH-Wert der Lösung wurde gemessen (Raumtemperatur).

Man sieht in Abbildung 4.7, dass die Pufferkapazität von der Konzentration von FCS abhängig ist. Man sieht auch, dass die Pufferkapazität als konstant von pH 7 bis 6 betrachtet werden kann, da der pH-Wert linear mit der beigefügten Säuremenge sinkt. Für das Medium ohne FCS ist sie von pH 7 bis 6 gleich 0,3 mM. Für das Medium mit 5% FCS ist sie von pH 7 bis 6 gleich 0,53 mM. Für das Medium mit 10% FCS ist sie von pH 7 bis 6 gleich 0,78 mM. FCS ist ein Serum und wird aus Blut gewonnen. Da Blut eine Pufferlösung ist (Blut enthält Bikarbonat- und Phosphat-Puffer), steigt die Pufferkapazität des Kulturmediums, wenn die Konzentration von FCS steigt.

Wenn die pH-Variationen nicht eine pH-Einheit überschreiten, kann die Pufferkapazität für die Messungen mit Zellen als konstant betrachtet werden. Für die Evaluierung der Ansäuerungsrate (im nächsten Abschnitt) wird die Pufferkapazität nicht betrachtet. Um die absolute Ansäuerungsrate der Zellen zu erhalten, muss die Rate mit der Pufferkapazität des Kulturmediums multipliziert werden.

4.3.2 Evaluierung der extrazellulären Ansäuerung

Das Cytoplasma der tierischen Zellen hat einen konstanten pH-Wert von 7,2. Wenn der pH-Wert sich ändern würde, würden viele biochemische Reaktionen nicht mehr optimal ablaufen können. Da einige Reaktionen Säuren produzieren, wie Kohlensäure oder Milchsäure, ist die Zelle in der Lage, Protonen und Basen auszuwerfen. Dafür haben die Zellen verschiedene Ionenkanäle, wie Na^+/H^+-Antiporter, $Cl^-/HCO3^-$-Antiporter, $Na^+/HCO3^-$-Symporter, und H^+-Kanäle, die durch ATP kontrolliert werden. Für tierische Zellen wird die Ansäuerungsrate durch die Berechnung der Steigung der Regressionsgerade der pH-Messung berechnet, sobald die Pumpe ausgeschaltet ist, und zwar für jeden Zyklus des Versuchs. In Abbildung 4.8 werden einige pH-Messzyklen eines Versuchs und die Regressionsgerade aufgezeigt. In diesem Versuch wurde der Wirkstoff Cytochalasin B hinzugefügt, was eine kleinere Ansäuerungsrate und Erhöhung des pH-Werts in der Mikroumgebung der Zellen verursacht hat.

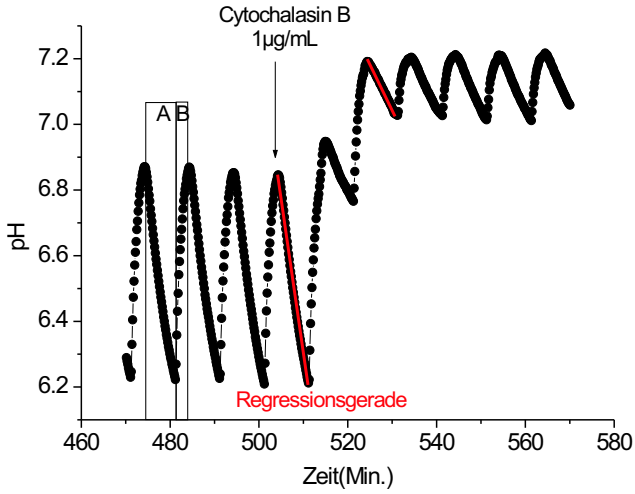


Abbildung 4.8: ISFET pH-Messung. Einfluss von Cytochalasin B auf die Ansäuerungsrate von LS174-T Zellen. Phase A (Pumpe eingeschaltet) und B (Pumpe ausgeschaltet) definieren ein Messzyklus. Die Ansäuerungsrate wird durch die Steigung der Regressionsgerade für jeden Messzyklus evaluiert.

Für Algen-Zellen wird ein Alkalisieren des Mediums statt der Ansäuerung gemessen. Die Berechnung der Metabolismusrate erfolgt aber ebenso wie für tierische Zellen.

4.4 Sauerstoff-Verbrauchsrate und Ansäuerungsrate: Messergebnisse

Zwei exemplarische Messergebnisse werden in diesem Abschnitt aufgezeigt. Für dieerste Messung wurde ein einziger Wirkstoff ins Kulturmedium eingebracht. Bei der zweiten Messung wurde der Effekt von einem Wirkstoff auf den Metabolismus von Zellen unter Nahrungsmangel evaluiert.

- Effekt von Histamin auf HeLa-Zellen
  Histamin ist ein Hormon und Neurotransmitter und bindet an verschiedenen Rezeptoren der Zelle. Im menschlichen Körper ist Histamin in basophilen Granulocyten und in Mastzellen (Abwehrzellen) im Blut gespeichert. Histamin produziert verschiedene Effekte auf den Zielzellen: zum Beispiel verursacht es die Kontraktion der glatten Muskulatur der Bronchien und verlangsamt den Herzschlag [10]. Der Effekt von Histamin auf die Sauerstoff-Verbrauchsrate und Ansäuerungsrate von HeLa-Zellen wurde untersucht. Histamin wurde an einer konfluenten Kultur 16 Stunden lang zugegeben. Das Ergebnis der Messung ist in Abbildung 4.9 dargestellt.

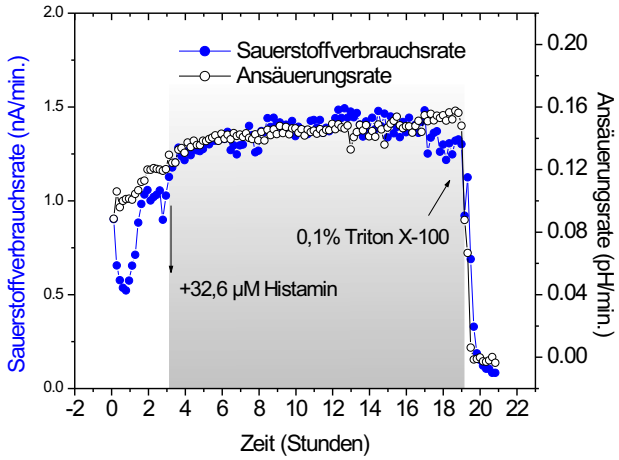


Abbildung 4.9: Effekt von Histamin auf die Ansäuerungsrate und Sauerstoffverbrauchsrate von HeLa-Zellen. Im grauen Bereich enthält das Kulturmedium Histamin.

Man beobachtet keine bedeutende Wirkung von Histamin auf die Ansäuerungsrate und Sauerstoff-Verbrauchsrate. Der Effekt von Histamin auf die Morphologie/Adhäsion von HeLa-Zellen wird in Kapitel 4.5 präsentiert.

- Effekt von Cytochalasin B auf LS174T-Zellen
  Cytochalasine (B,D) sind Gifte aus Schimmelpilzen. Der Effekt von Cytochalasin B auf die Ansäuerungsrate und Sauerstoffverbrauchsrate von LS174T-Zellen unter Nahrungsmangel wurde untersucht. Das Ergebnis der Messung ist in der Abbildung 4.10 dargestellt. Nach ca. 4 Stunden wurde das Kulturmedium durch ein Kulturmedium ohne die Aminosäure Glutamin ersetzt, was einen starken Abfall der Sauerstoffverbrauchsrate (-60%) und Anstieg der Ansäuerungsrate (+50%) zur Folge hatte. Danach wurde der Wirkstoff Cytochalasin B zugegeben, wodurch ein starke Anstieg der Sauerstoffverbrauchsrate und ein Abfall der Ansäuerungsrate verursacht wurde. Als der Wirkstoff abgesetzt wurde, haben die Metabolismus-Raten fast ihre ehemaligen Werte wieder erreicht. Wenn normales Kulturmedium zugegeben wurde, wurde ein kleiner Anstieg der Sauerstoffverbrauchsrate und Abfall der Ansäuerungsrate registriert.

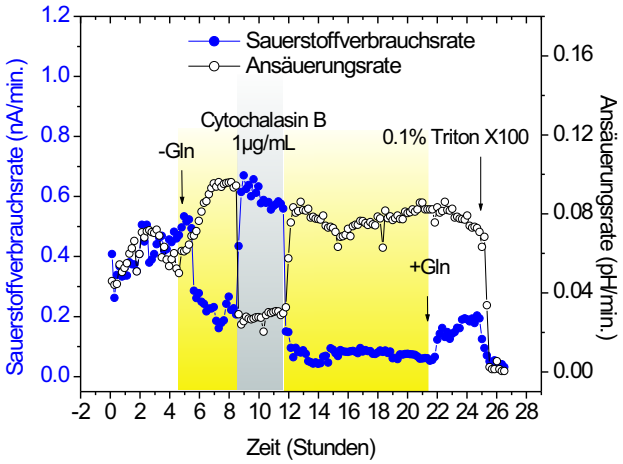


Abbildung 4.10: Effekt von Cytochalasin B auf die Ansäuerungsrate und Sauerstoffverbrauchsrate von LS174T-Zellen, in einem Kulturmedium ohne die Aminosäure Glutamin. Im grauen Bereich enthält das Kulturmedium Cytochalasin B und im gelben Bereich kein Glutamin.

In diesem Messergebnis wird beobachtet, dass die Sauerstoffverbrauchsrate und Ansäuerungsrate gegenläufig sind. Es liegt daran, dass die Metabolismuswege voneinander abhängig sind. Wenn die Ansäuerungsrate als Funktion der Sauerstoffverbrauchsrate dargestellt wird (Abbildung 4.11), wird die Abhängigkeit der zwei Prozesse deutlich.

Für diesen Versuch wurde der lineare Korrelationskoeffizient zwischen Sauerstoffverbrauchsrate und Ansäuerungsrate ausgewertet: er ist gleich -0,77. Der lineare Korrelationskoeffizient $\rho_{x,y}$ ist eine reelle Zahl zwischen -1 und 1, die die lineare Abhängigkeit von zwei Prozessen (X, Y) charakterisiert. Für $\rho_{x,y} = 1$ sind die Prozesse perfekt linear abhängig. Für $\rho_{x,y} = 0$ sind die Prozesse unabhängig. Für $\rho_{x,y} = -1$ sind sie perfekt linear gegenläufig abhängig. Der Koeffizient $\rho_{x,y}$ wird durch die Standardabweichungen (σ_x, σ_y) und Covarianz ($\sigma_{x,y}$) der zwei Prozesse berechnet. Wenn m_x und m_y die Mittelwerte der X- und Y-Punkte sind, ergibt sich:

$$\sigma_x{}^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - m_x)^2 \tag{4.11}$$

$$\sigma_{x,y} = \sum_{i=1}^{n}(x_i - m_x)(y_i - m_y) \tag{4.12}$$

$$\rho_{x,y} = \frac{\sigma_{x,y}}{\sigma_x \sigma_y} \tag{4.13}$$

Der lineare Korrelationskoeffizient kann durch die Steigung s der Regressionsgerade der Punkte (x_i, y_i) ermittelt werden:

$$\rho_{x,y} = \frac{s\sigma_x}{\sigma_y} \tag{4.14}$$

Die Ansäuerungs- und Sauerstoffverbrauchsraten sind aber nicht immer gegenläufig voneinander abhängig. Wenn die Konversion von Glukose zu Pyruvat zum Beispiel gestört wird, können die beiden Raten kleiner werden. Auf der Graphik, die die Ansäuerungsrate als Funktion der Sauerstoffverbrauchsrate darstellt, können Bereiche definiert werden, die den Zustand der Zellen kennzeichnen, wie es in der Abbildung 4.11 aufgezeigt wird. Im Kapitel 4.7 wird diese Datendarstellung auch für andere Versuche verwendet.

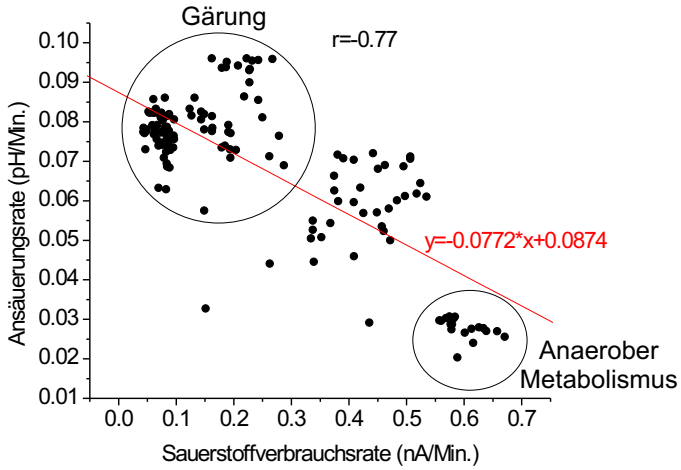


Abbildung 4.11: Ansäuerungsrate als Funktion der Sauerstoffverbrauchsrate für den Versuch gemäß Abb. 4.10.

4.5 Impedanz-Messungen

Mit adhärenten tierischen Zellen können Zellwachstum, Zellsterben, Zelladhäsions- und Morphologie-Änderungen durch Impedanz-Sensoren gemessen werden. In diesem Abschnitt werden verschiedene Impedanz-Messergebnisse betrachtet und eine Methode der Bearbeitung des Impedanz-Signals wird erklärt.

4.5.1 Zellwachstum

Die Zellmembran der tierischen Zellen besteht aus einer Doppelschicht von Lipiden, wie in Kapitel 1 erwähnt. Die geschätzte Kapazität der Zellmembran ist 20 $\mu F/cm^2$. Sie ist frequenz-unabhängig [24]. Wenn Zellen auf dem Sensor wachsen, ändern sich Kapazität und Widerstand des Sensors. Es ist deswegen möglich, das Zellwachstum durch Impedanz-Sensoren zu messen. Messungen des Wachstums von MCF-7- und L929-Zellen auf Glas-Sensorchips wurden einige Tage lang ohne Zugabe von Wirkstoffen durchgeführt. In diesem Abschnitt werden zwei Messergebnisse mit MCF-7-Zellen und ein Messergebnis mit L929-Zellen präsentiert. Danach werden die Messergebnisse interpretiert.

- Erster Versuch mit MCF-7-Zellen
  In einem ersten Versuch wurden Glas-Sensorchips mit $5 \cdot 10^4$ Zellen beimpft. Die Chips wurden 5 Stunden in den Inkubator gestellt und danach ins Testsystem eingesetzt. Die Messung des äquivalenten parallelen Widerstand und der Kapazität wurde 6 Tage lang durchgeführt. Die Ergebnisse sind in Abbildung 4.12 dargestellt.

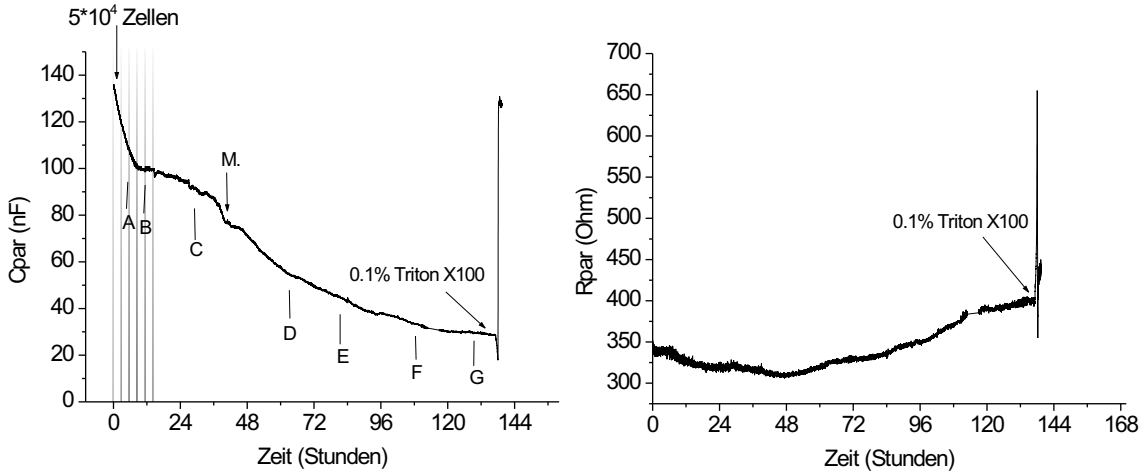


Abbildung 4.12: Messung des Zellwachstums von $5 \cdot 10^4$ MCF-7-Zellen auf einem Impedanz-Sensor, 6 Tage lang. Verlauf von äquivalenter paralleler Kapazität (links) und Widerstand (rechts). Versuch mit 5% FCS und Gentamycin. Zu den Zeiten A, B, C, D, E und F wurden Fotos aufgenommen. Gestrichelter Bereich: Latenz-Phase. M.: Wechsel der Kulturmediumflasche. Volumen der Messkammer: 1 mL. Pumpengeschwindigkeit: 11 $\mu L/Min$.

Während der ersten 10 Stunden war die Zellkultur in der Latenz-Phase: die Kapazität ist schnell gesunken, da die Zellen ihre Glykocalix wieder aufgebaut haben und da die Zelladhäsion am Sensorchip gestiegen ist. Danach war die Zellkultur in der Log- und dann in der Plateau-Phase: die Kapazität ist langsamer gesunken. Am Ende des Versuchs wurde Triton X100 zugegeben: die äquivalente parallele Kapazität ist gestiegen, um den Wert der Sensorkapazität ohne Zellen zu erreichen. Die Widerstandskurve ist während der 48 ersten Stunden leicht gesunken. Danach ist sie gestiegen. Die Interpretation des Widerstandsverlaufs ist im letzten Punkt dieses Abschnitts gegeben. Parallel zu der Messung der Impedanz wurden Fotos aufgenommen und ausgewertet. Die Ergebnisse der Auswertung sind in der Tabelle 4.40 dargestellt. In diesem Versuchergebnis wurde die folgende Beobachtung gemacht: bevor die Kulturmediumflasche gewechselt wurde, ist die Kapazität schneller gesunken und nachher langsamer. Es liegt daran, dass der pH-Wert des Kulturmediums in diesem Messaufbau leicht alkalisch wird (nach 24 Stunden: pH 7.40 statt 7.14), und, dass der pH-Wert Einfluss auf die Zellmorphologie hat. In diesem Abschnitt werden mehrere andere Versuchsergebnisse gezeigt, wo dieses Phänomen ebenfalls zu sehen ist.

- Zweiter Versuch mit MCF-7-Zellen
  In einem anderen Versuch wurden die Chips mit $2 \cdot 10^5$ MCF-7-Zellen beimpft, 5 Stunden in den Inkubator gestellt und danach ins Testsystem eingesetzt. Die Messung des äquivalenten parallelen Widerstands und der Kapazität sind in der Abbildung 4.13 dargestellt.

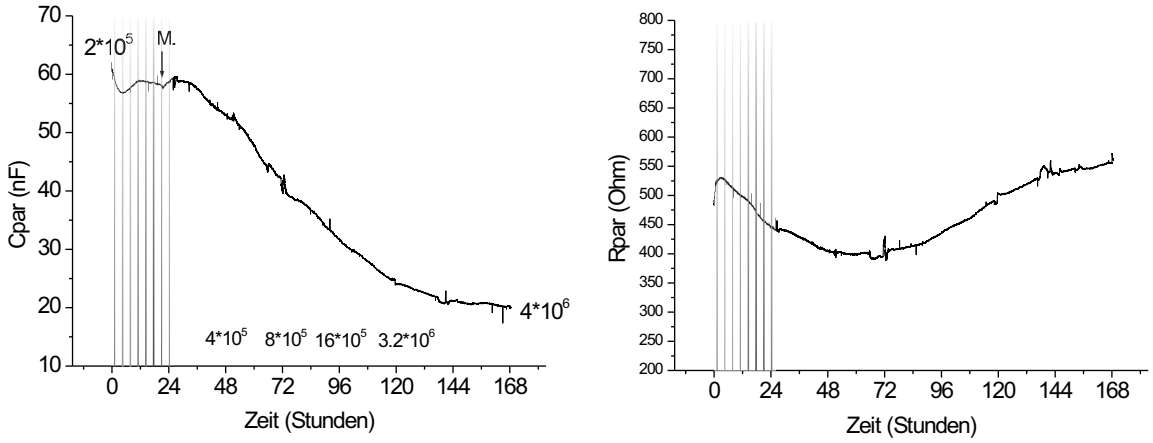


Abbildung 4.13: Messung des Zellwachstums von $2 \cdot 10^5$ MCF-7-Zellen auf einem Impedanz-Sensor, 7 Tage lang. Verlauf von äquivalenter paralleler Kapazität (links) und Widerstand (rechts). Versuch mit 5% FCS und Gentamycin. M.: Wechsel der Kulturmediumflasche. Volumen der Messkammer: 300 μL. Pumpengeschwindigkeit: 11 $\mu L/Min$.

In der Latenz-Phase ist die Kapazität schnell gesunken und danach leicht gestiegen. Als die Zellkultur in der Log-Phase war, ist die Kapazität wie für den Versuch 4.12 gesunken. Die Widerstandskurve ist während der zwei ersten Tage steil gesunken. Danach ist sie gestiegen. Die Interpretation des Widerstandsverlaufs ist im letzten Punkt dieses Abschnitts gegeben.

Am Ende dieses Versuchs wurde eine Schätzung der Zellkernzahl mit dem Casy-Zellzähler vorgenommen. Eine Schätzung von $4{\cdot}10^6$ Zellen wurde erhalten. In Abbildung 4.12 sind Bestimmungen der Zellzahl gegeben, indem die Zellverdoppelungszeit auf 24 Stunden bestimmt wurde. In der Abbildung 4.14 ist die erhaltene Zellkernzählungskurve gezeigt. Ein lokales Maximum von Zellkernen mit einem Durchmesser von 6,7 μm wurde gemessen. Die Kurve zeigt ein Maximum von Zellkernen, die einen Durchmesser von 8,8 μm haben. Diese Zellkerne haben ein Volumen V_2, das fast zweimal so groß wie das Volumen V_1 der 6,7 μm Zellkerne. ($V_1 = 4/3 \cdot \pi \cdot 6,7^3/8 = 157$ $\mu m^3, V_2 = 4/3 \cdot \pi \cdot 8,8^3/8 = 267$ μm^3). Das sind die Zellkerne, die auf die Mitose vorbereitet waren.

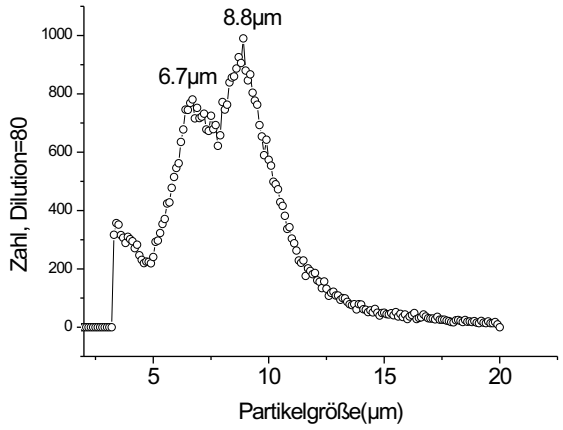


Abbildung 4.14: Ergebnis der Zellkernzählung.

- Versuch mit L929-Zellen
  In diesem Versuch wurden die Glas-Sensorchips mit $1 \cdot 10^5$ und $2 \cdot 10^5$ L929-Zellen beimpft, 5 Stunden in den Inkubator gestellt und danach ins Testsystem eingesetzt. Die Messung des äquivalenten parallelen Widerstands und der Kapazität sind in Abbildung 4.15 dargestellt.

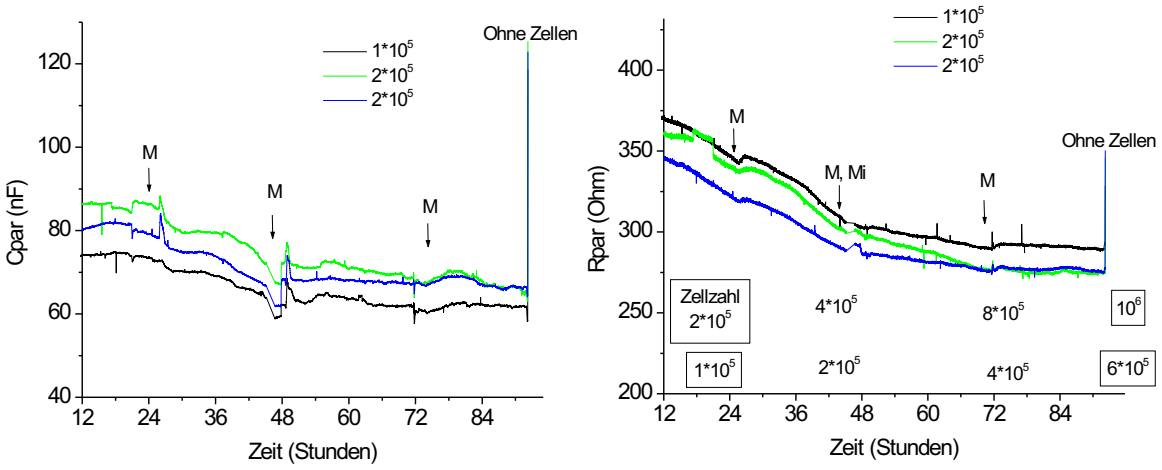


Abbildung 4.15: Verlauf der äquivalenten parallelen Kapazität (links) und des Widerstands (rechts), 3 Tage lang. Versuch mit L929-Zellen, 2 verschiedene Zelldichten, 10% FCS und Gentamycin. M.: Wechsel der Kulturmediumflasche. Volumen der Messkammer: 300 μL. Pumpegeschwindigkeit: 11 $\mu L/Min$.

In der Log-Phase sinken die Kapazitätskurven langsamer als mit MCF-7-Zellen, da die L929-Zellen langsamer und nicht übereinander wachsen, und da sie eine andere Morphologie zeigen. Die Widerstandskurven sind im Gegenteil mit den letzten Versuchen die ganze Zeit gesunken. Am Ende des Versuchs wurden die Zellkerne mit dem Casy-Zellzähler gezählt. Die Ergebnisse der Zellkernzählung und Bestimmungen der Zellzahl zu verschiedenen Zeitpunkten sind in Abbildung 4.15 gegeben.

- Interpretation des Verlaufs der äquivalenten parallelen Kapazität und Widerstand
  Die Verläufe des äquivalenten parallelen Widerstands und der Kapazität dieser Langzeitversuche werden im Folgenden interpretiert. In Abbildung 4.16 wird die Interpretation schematisch dargestellt. Wenn die Zellkultur sich in der Latenz-Phase befindet, haben die Zellen eine kugelförmige Morphologie, wie es auf der mikroskopischen Aufnahme 4.17 gezeigt ist. Nachher ist ihre Morphologie flacher und die Adhäsionskraft größer. Sie besetzen eine größere Fläche auf dem Chip. Deshalb ist die Kapazität kleiner. Der Zellmetabolismus ändert die Leitfähigkeit des Mediums. In der Latenz-Phase wird das Medium leitfähiger, deshalb sinkt der Widerstand. In den Log- und Plateau-Phasen steigt der Widerstand und sinkt die Kapazität, da die Zellen immer mehr Fläche auf dem Sensor besetzen.

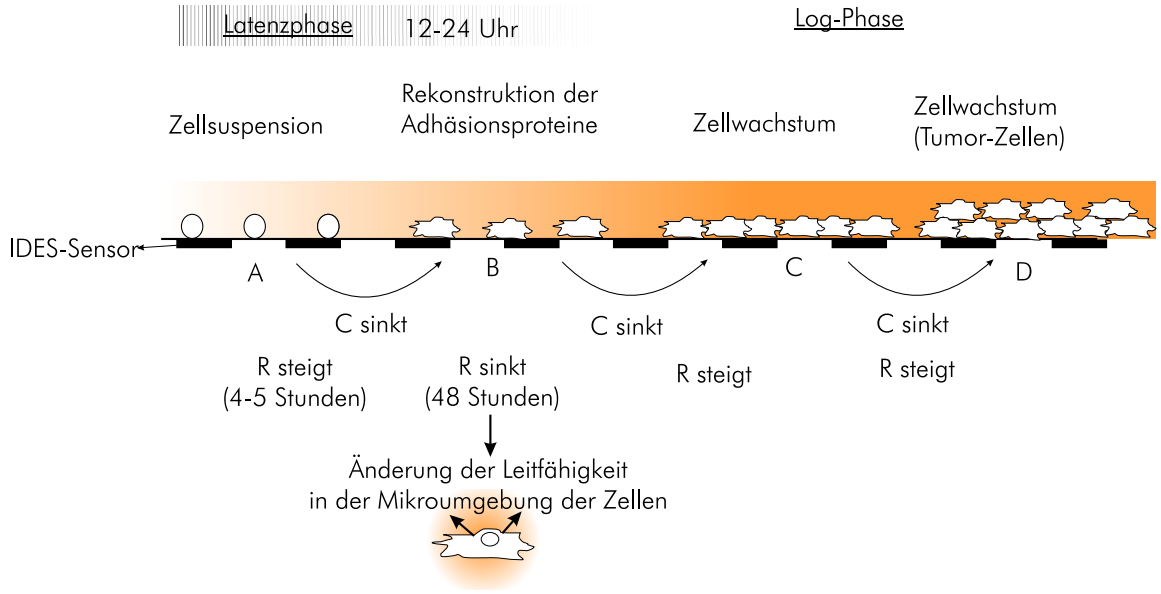


Abbildung 4.16: Erklärung des Verlaufs des äquivalenten parallelen Widerstands R und der Kapazität C.

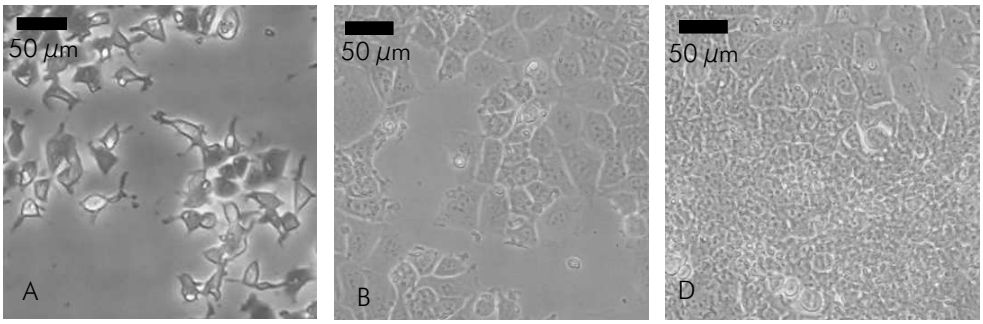


Abbildung 4.17: Phasenkontrastmikroskopie von MCF-7-Zellen zu verschiedenen Zeitpunkten (siehe 4.16): A: Latenz-Phase, B: Log-Phase, D: Plateau-Phase (die Zellen sind übereinander gewachsen)

4.5.2 Zellwachstum und Morphologie/Adhäsions-Änderungen

Mit dem Impedanz-Sensor werden auch Änderungen in der zellulären Adhäsion und Morphologie parallel zum Zellwachstum gemessen. In diesem Abschnitt sind verschiedene Messergebnisse gezeigt, wobei unterschiedliche Wirkstoffe zum Zellkulturmedium zugegeben worden sind.

4.5.2.1 Effekt von Cytochalasin B auf MCF-7-Zellen

Cytochalasin B verhindert die Polymerisation von Aktin-Filamenten. Wie in Kapitel 1 erklärt, besteht das Zytoskelett aus Aktin-Filamenten. Deshalb kann dieser Wirkstoff Zellbewegung und Zellteilung hemmen und ändert Zelladhäsion und Morphologie. Zwei Versuche mit MCF-7-Zellen werden in diesem Abschnitt präsentiert.

- Erster Versuch mit MCF-7-Zellen
  Glas-Sensorchips wurden mit $2 \cdot 10^5$ MCF-7-Zellen beimpft und 5 Stunden nachher ins Testsystem eingesetzt. Die Messergebnisse sind in Abbildung 4.18 gezeigt. Nach einem Tag wurde Cytochalasin B zugegeben. Kurz nach dem Zufügen von Cytochalasin B ist die äquivalente Kapazität während 3-4 Stunden schnell gesunken, der Widerstand dagegen schnell gestiegen, was

eindeutig zeigt, dass Änderungen in der Zellmorphologie und Adhäsion aufgetreten sind. Die Kapazität ist anschließend noch weiter gesunken. Das zeigt, dass es immer noch Zellteilungen auf dem Impedanz-Sensor gab, was mikroskopische Aufnahmen bestätigt haben. 36 Stunden nach dem Messbeginn ist der Widerstand steiler gestiegen und die Kapazität schneller abgefallen. Dies zeigt, dass eine Änderung der zellulären Adhäsion aufgetreten ist.

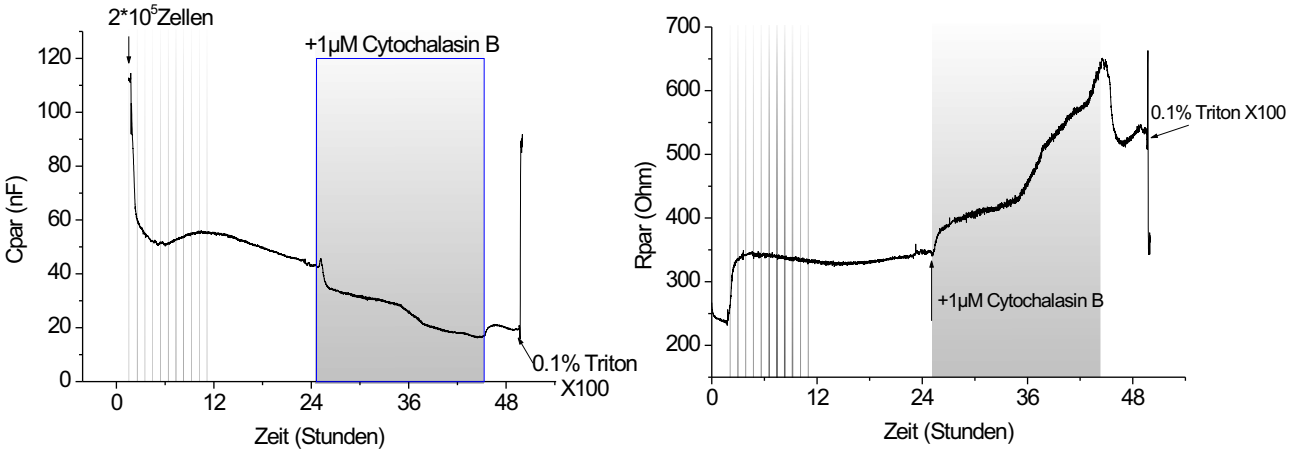


Abbildung 4.18: Effekt von 1 μM Cytochalasin B auf MCF-7-Zellen. Verlauf der äquivalenten parallelen Kapazität(links) und des Widerstands (rechts). Pumpe immer eingeschaltet. Volumen der Messkammer: 1 mL

18 Stunden nach der Zugabe des Wirkstoffes wurde eine mikroskopische Aufnahme aufgenommen, die in Abbildung 4.19 gezeigt ist. Die morphologische Änderung der Gewebe ist deutlich zu sehen. Später wurde ein Kulturmedium ohne Cytochalasin B zugegeben. Da der Effekt des Wirkstoffs reversibel ist, ist die Kapazität wieder gestiegen und der Widerstand gesunken, die Morphologie der Gewebe war wieder normal.

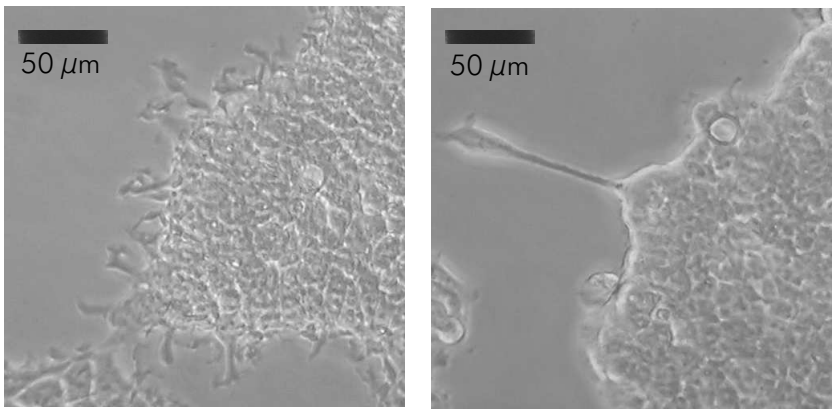


Abbildung 4.19: Mikroskopische Aufnahmen vor (links) und 18 Stunden nach der Zugabe (rechts) von 1μM Cytochalasin B ins Kulturmedium: alle Pseudopoden der Gewebe sind verschwunden.

- Zweiter Versuch mit MCF-7-Zellen
  Glas-Sensorchips wurden mit $2 \cdot 10^5$ MCF-7-Zellen beimpft und nach 5-6 Stunden Inkubation im Testsystem eingesetzt. In diesem Versuch war die Pumpe periodisch eingeschaltet. In Abbildung 4.20 werden die Messergebnisse dargestellt. Die Messzyklen sind in den Messkurven sichtbar.

Nach 18 Stunden wurde 1 μM Cytochalasin B 3 Stunden lang zugegeben. Wie bei den vorherigen Versuche nimmt die Kapazität ab und die Steigung der Widerstandkurve nimmt zu, was zeigt, dass Zellmorphologie/Adhäsions-Änderungen auftreten.

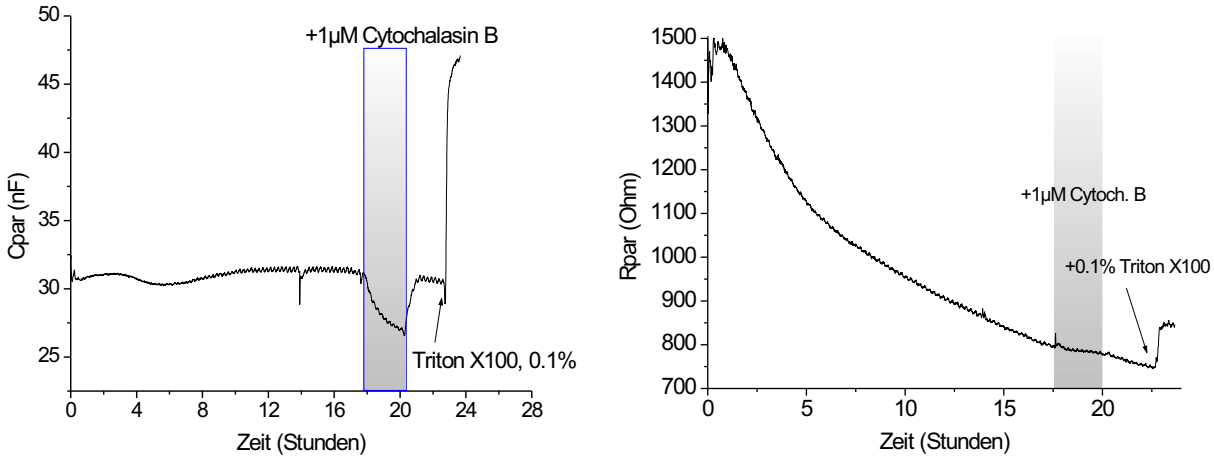


Abbildung 4.20: Effekt von Cytochalasin B auf $2 \cdot 10^5$ MCF-7-Zellen. Verlauf der äquivalenten parallelen Kapazität (links) und des Widerstands (rechts). Pumpe periodisch eingeschaltet. Volumen der Messkammer von 300 μL.

4.5.2.2 Effekt von Histamin auf HeLa- und MDA-MB-231-Zellen

Im Kapitel 4.4 wurde bereits erklärt, dass Histamin ein Hormon ist, das verschiedene Auswirkungen hat, je nach Zelltyp. Die Effekte von Histamin auf die Zellmorphologie wurden auf den Zelllinien MDA-MB-321 und HeLa mit Impedanz-Sensoren registriert und sind in diesem Abschnitt gezeigt. Es wird verdeutlicht, dass die Reaktionen auf Histamin sehr unterschiedlich sein können. Die Zellreaktionen auf Histamin hängen nämlich von der Wirkstoffkonzentration und vom Zelltyp ab.

- Effekt von Histamin auf MDA-MB-231-Zellen
  Ein Glas-Sensorchip wurde mit $1 \cdot 10^5$ MDA-MB-231-Zellen beimpft und in das Testsystem eingesetzt. 2 Stunden später wurde 35 μM Histamin zugegeben. Wie in Abbildung 4.21 gezeigt, ist die Kapazität nach der Zugabe von Histamin 3 Stunden lang gestiegen, was zeigt, dass es Änderungen in der Zellmorphologie gab. Der Widerstandverlauf zeigt keine deutliche Änderung (nur eine sehr kleine Abnahme).

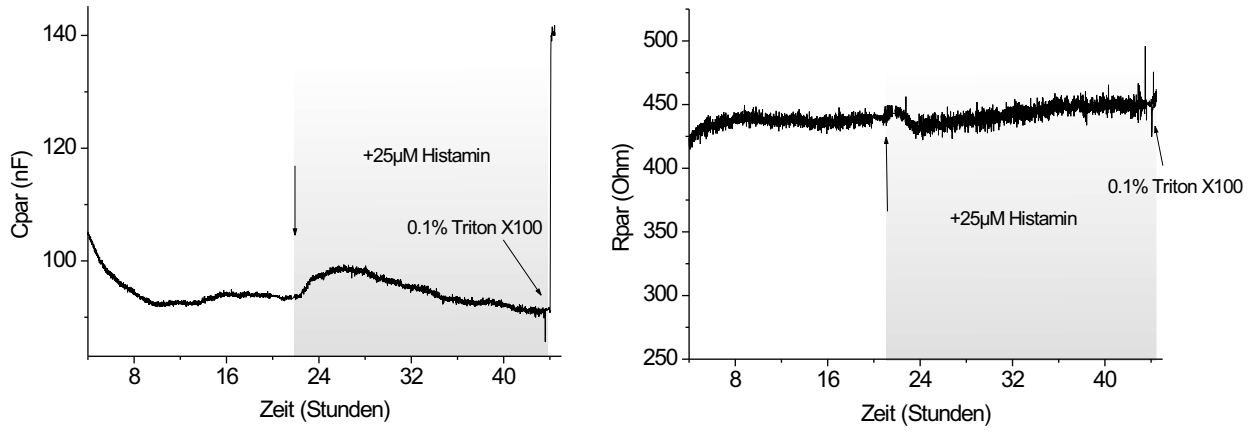


Abbildung 4.21: Effekt von Histamin auf MDA-MB-231-Zellen. Verlauf der äquivalenten parallelen Kapazität (links) und des Widerstands (rechts).

- Effekt von Histamin auf HeLa-Zellen
  Ein Glas-Sensorchip wurde mit $1 \cdot 10^5$ HeLa-Zellen beimpft und ins Testsystem eingesetzt. 3 Tage nachher wurde 50 μM Histamin zugegeben. Wie in Abbildung 4.22 gezeigt, ist die Kapazität nach der Zugabe von Histamin 10 Minuten lang gestiegen und danach tief gesunken. Der Widerstand ist im Gegensatz dazu 10 Minuten lang gesunken und danach gestiegen.

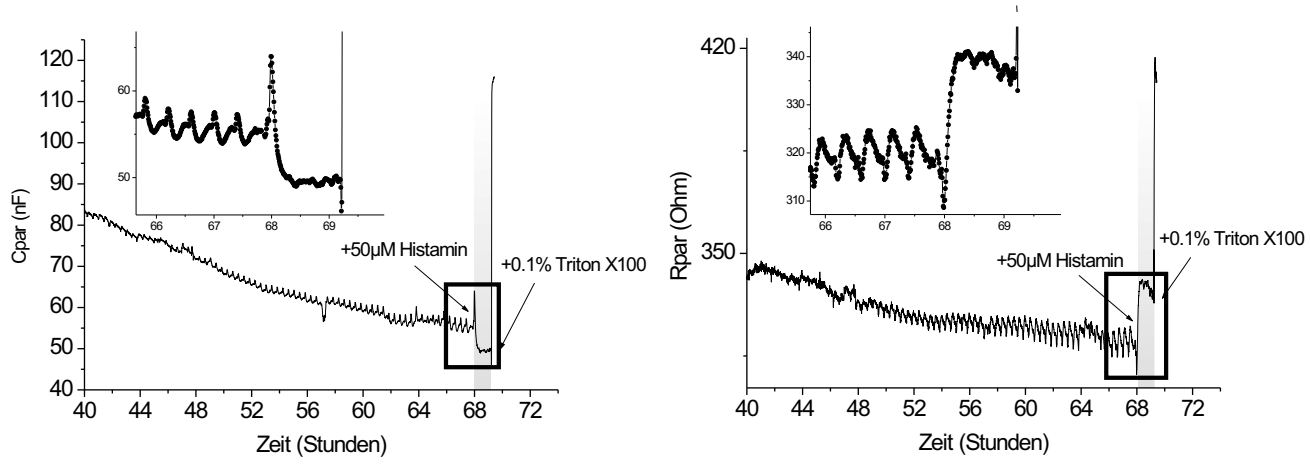


Abbildung 4.22: Effekt von Histamin auf HeLa-Zellen. Verlauf der äquivalenten parallelen Kapazität (links) und des Widerstands (rechts).

4.5.2.3 Effekt von Tamoxifen auf MCF-7-Zellen

Tamoxifen ist ein Chemotherapeutikum, das bei Brust-Tumoren eingesetzt wird. Tamoxifen ist ein Antiöstrogen, hat hohe Affinität zum Östrogen-Rezeptor und konkurriert mit dem Östradiol-Hormon [3] [15]. Drei Tage nach der Zugabe von 2 μM Tamoxifen wird das Zellwachstum gehemmt. Der Wirkstoff hat auch einen Effekt auf die Aktin-Filamente. Um diese Effekte mit Impedanz-Sensoren zu registrieren wurden zwei Versuchsreihen durchgeführt und in diesem Abschnitt vorgestellt.

- Erste Versuchsreihe
  In dieser Versuchsreihe wurden Glas-Sensorchips mit unterschiedlichen Zellzahlen beimpft. Ver-

schiedene Zelldichten wurden gewählt, weil mit hohen und niedrigen Zelldichten unterschiedliche Eigenschaften der Zellkultur registriert werden können. Hohe Zelldichten erlauben nämlich eine bessere Detektierung von zellmorphologischen Änderungen. Mit kleinen Zelldichten kann die Zellwachstumshemmung registriert werden.

Die Glas-Sensorchips wurden mit den folgenden Zellzahlen beimpft: $0,8 \cdot 10^5$, $1 \cdot 10^5$ und $2 \cdot 10^5$. Die Chips wurden 5 Stunden in den Inkubator gestellt und danach ins Testsystem eingesetzt. Ein Tag später wurden 2 μM zum Kulturmedium hinzugegeben. Für einen Versuch mit $0,8 \cdot 10^5$ Zellen wurde kein Wirkstoff zugegeben. Die Messung wurde 4 Tage lang durchgeführt und ist in Abbildung 4.23 dargestellt. Die Kapazitätsverläufe am Anfang der Versuche mit $0,8 \cdot 10^5$ Zellen weisen Unterschiede auf. Es liegt daran, dass die Zellverteilung auf den Sensor-Chips nicht homogen war. Aber die Kapazitätswerte erreichen ein ähnliches Niveau am Ende des Versuchs, was bedeutet, dass keine große Zellwachstumshemmung für den Versuch mit Tamoxifen gemessen wurde.

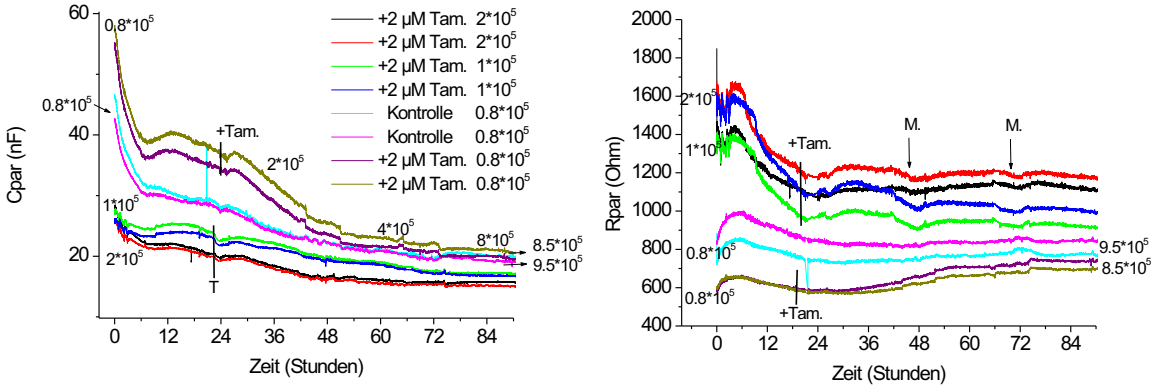


Abbildung 4.23: Effekt von Tamoxifen auf MCF-7-Zellen. Verlauf der äquivalenten parallelen Kapazität (links) und des Widerstands (rechts) mit verschiedenen Zelldichten und einer Kontrolle. Volumen der Messkammer: 1mL. M: Wechsel der Kulturmediumflasche.

Nach den Versuchen wurden die Zellkerne mit dem elektronischen Zellzähler gezählt. Die Ergebnisse und Bestimmungen der Zellzählung werden in Abbildung 4.23 beschrieben. Für eine Anfangs-Zellzahl von $0,8 \cdot 10^5$ wurden für den Kontrollversuch $9,5 \cdot 10^5$ gezählt. Mit Zugabe von Tamoxifen wurden $8,5 \cdot 10^5$ Zellen am Versuchsende gezählt (10% weniger Zellen). Der Zellzahlenunterschied war klein, was kohärent mit dem Kapazitätsverlauf dieser Versuche ist.

- Zweite Versuchsreihe
  In dieser Versuchsreihe wurden 2 Glas-Sensorchips mit $1 \cdot 10^5$ MCF-7-Zellen beimpft, 5 Stunden in den Inkubator gestellt und danach ins Testsystem eingesetzt. Ein Tag später wurde 2 μM Tamoxifen zu einer Kultur zugegeben. Die Messung wurde 5 Tage lang durchgeführt. Die Zellzahl wurde am Ende des Versuchs mit dem Casy-Zellzähler ermittelt. Das Ergebnis wird in Abbildung 4.25 und 4.26 beschrieben. Für die Kontrolle wurden $2,5 \cdot 10^6$ Zellen gezählt. Für die Kultur, die mit dem Wirkstoff behandelt wurde, wurden $2,3 \cdot 10^6$ Zellen gezählt (8% weniger Zellen). Der Zellzahlenunterschied war noch kleiner als bei der ersten Versuchsreihe. Die erhaltenen Zellkernzählungskurven sind in Abbildung 4.24 dargestellt. Die Zellkernzählungskurven zeigen, dass es in beiden Kulturen noch Zellen gab, die sich auf die Mitose vorbereitet hatten.

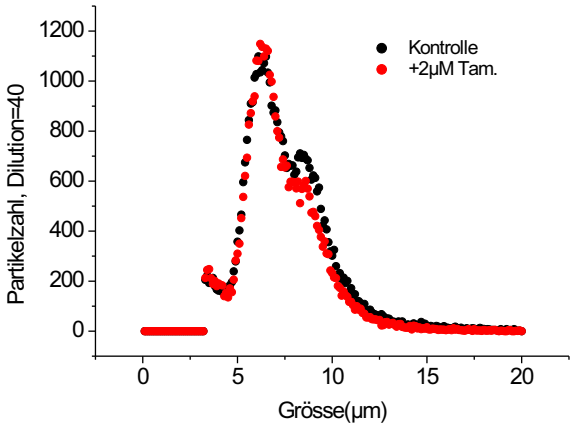


Abbildung 4.24: Zellkernzählung am Ende des Versuchs 4.25.

Die Kapazitätskurven (Abbildung 4.25) der Test- und Kontrollversuche zeigen einige Unterschiede. Am Anfang waren die Kapazitätswerte der Kontrolle deutlich größer als die Werte des Testversuchs. Das liegt daran, dass die Zellen auf dem gesamten Sensorchip nicht gleichmässig verteilt sind. Auf den IDES-Sensoren des Testversuchs lagen mehr Zellen als auf den IDES-Sensoren der Kontrolle. Danach sind die Kapazitätswerte für beide Versuche gesunken. Die Kapazitätswerte der Kontrolle waren jedoch am Ende des Versuchs niedriger, was zeigen könnte, dass die Zellwachstumsrate leicht grösser war.

Die Widerstandskurven (Abbildung 4.26) der Test- und Kontrollversuche zeigen ähnliche Unterschiede. Die Anfangswerte waren grösser bei dem Testversuch, da mehr Zellen auf dem Sensor lagen. Die Widerstandswerte der Kontrolle haben jedoch höhere Werte erreicht als bei dem Testversuch, was zeigen könnte, dass die Wachstumsrate leicht höher war.

Diese Versuchsreihen zeigen, dass 2 μM Tamoxifen keine starke Zellwachstumshemmung von MCF-7-Zellen verursacht.

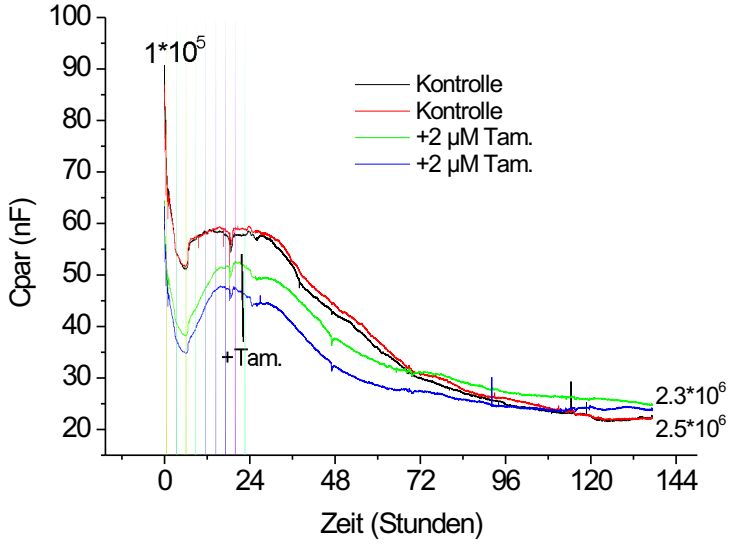


Abbildung 4.25: Effekt von 2 μM Tamoxifen auf $1 \cdot 10^5$ MCF-7-Zellen. Verlauf der äquivalenten parallelen Kapazität. Vergleich mit einer Kontrolle. Messkammer mit einem Volumen von 300 μL.

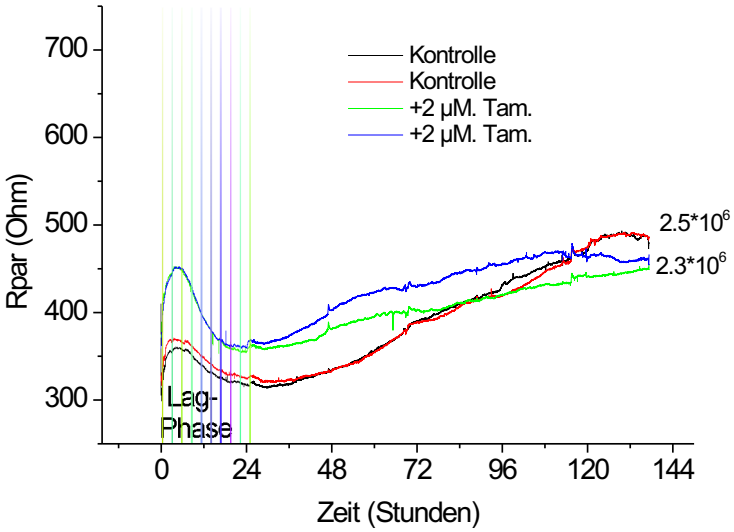


Abbildung 4.26: Effekt von Tamoxifen auf $1 \cdot 10^5$ MCF-7-Zellen. Verlauf des äquivalenten parallelen Widerstandes. Vergleich mit einer Kontrolle. Messkammer mit einem Volumen von 300 μL.

4.5.3 Zellsterben

Es wurde in den letzten Abschnitten gezeigt, dass Zellwachstum und Morphologie-/Adhäsions-Änderungen durch den Impedanz-Sensor detektiert werden können. Wenn Zellen sterben, wird ihre Membran geschädigt, was ebenfalls durch den Impedanz-Sensor festgestellt werden kann. In diesem Abschnitt wird ein Messergebnis gezeigt, in dem Zellen wegen einer bakteriellen Kontamination gestorben sind. Zwei parallele Versuche wurden durchgeführt. In einem Versuch wurde eine bakterielle Kontamination festgestellt. Der Verlauf des äquivalenten parallelen Widerstands und der Kapazität zeigen

große Unterschiede mit dem normalen Versuch und sind in den Abbildungen 4.27 und 4.28 dargestellt. Für den kontaminierten Versuch wurde beobachtet, dass die Kapazität 2 Stunden lang steil abgefallen ist (10 nF/Stunde), und danach stundenlang gestiegen ist (0,3 nF/Stunde), was zeigt, dass es immer weniger lebende Zellen auf dem Sensor gab. Der Widerstand ist 2 Stunden lang steil gestiegen (50 Ohm/Stunde) und danach stundenlang gesunken (3 Ohm/Stunde). Im normalen Verlauf ist der Widerstand nur gestiegen (2-3 Ohm/Stunde).

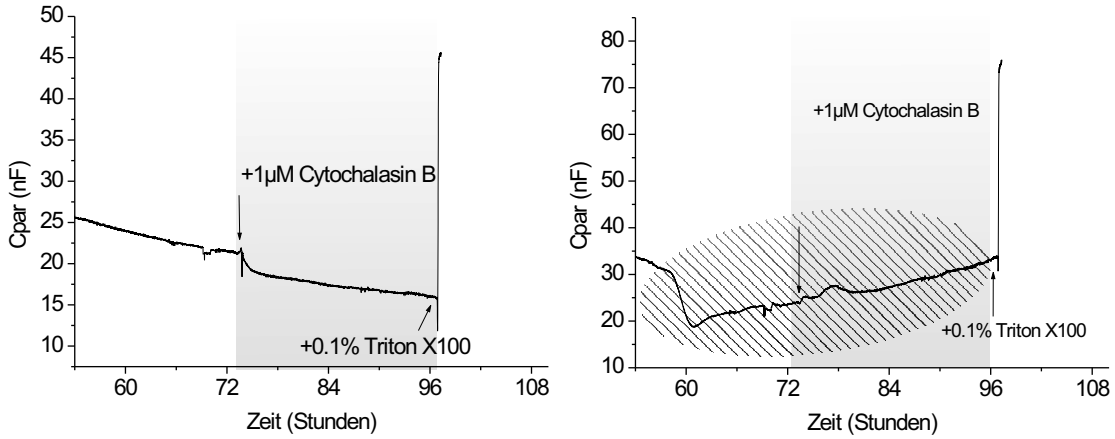


Abbildung 4.27: Verlauf der äquivalenten parallelen Kapazität ohne Kontamination (links) und mit bakterieller Kontamination (rechts).

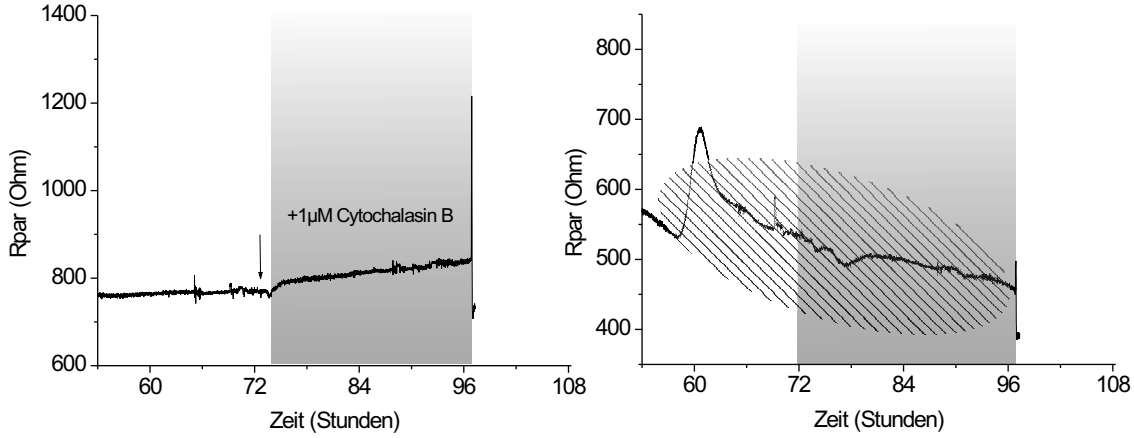


Abbildung 4.28: Verlauf des äquivalenten parallelen Widerstandes ohne Kontamination (links) und mit bakterieller Kontamination (rechts).

4.5.4 Bearbeitung des Impedanz-Signals

- Zusammenfassung der Messergebnisse
  Im vorausgehenden Abschnitt wurden verschiedene Messergebnisse präsentiert und beschrieben. Es wurde gezeigt, dass die Impedanz-Signale vieldeutig sind, da verschiedene Parameter wie Änderungen in Adhäsion/Morphologie, Zellwachstum und Zellsterben gleichzeitig mit den IDES gemessen werden. Während der ersten 12 Stunden nach dem Trypsinieren der Zellkultur werden

hauptsächlich Änderungen in der zellulären Adhäsion gemessen. Danach haben verschiedene Parameter Einfluss auf die Impedanz-Werte. Eine Zusammenfassung der Kapazitätswerte (Werte und Steigung) in der Log-Phase und ohne Wirkstoffzugabe wird in Tabelle 4.29 aufgeführt. Eine Zusammenfassung der Kapazitätswerte, wenn verschiedene Wirkstoffe zugegeben worden sind, ist in Tabelle 4.30 gezeigt.

Zellen	Zellzahl	Min. der Steigung(nF/Std.)	Max.(nF.)	Min.(nF.)	Versuch
MCF-7	50000	-1,25	100	29	4.12
MCF-7	100000	-0,55	58	23	4.25
MCF-7	200000	-0,46	58	20	4.13
L929	100000	-0,16	75	60	4.15
L929	200000	-0,28	80	66	4.15
HeLa	100000	-1,16	95	n.e	4.22

Abbildung 4.29: Zusammenfassung der Kapazitätswerte der präsentierten Versuche, wenn die Zellkultur in der Log-Phase war (ohne Wirkstoffzugabe). n.e.: nicht erreicht.

Mit Tabelle 4.29 kann man verschiedene Eigenschaften der Kapazitätwerte beobachten. Erstens, je höher die Zahl der MCF-7-Zellen ist, desto kleiner ist der absolute Steigungswert der Kapazität. Zweitens ist die Steigung der Kapazität für L929-Zellen viel kleiner als mit MCF-7-Zellen. Die L929-Zellen sind nämlich flacher als MCF-7-Zellen und sie haben daher weniger Einfluss auf die Sensorkapazität, wenn sie wachsen.

Eine Zusammenfassung der Wirkstoffeffekte wird in der Tabelle 4.30 dargestellt.

Wirkstoff	Konz. (μM)	Zellen	Steigung (nF/Std.)	Dauer (Std.)	Effekt (nF.)	Versuch
+CB	1	MCF-7	-10,3	1,1	-11,3	4.18
-CB		MCF-7	+3,4	1,2	+4,1	4.18
+CB	1	MCF-7	-1,3	3,0	-3,9	4.20
-CB		MCF-7	+5,4	0,7	3,8	4.20
+Histamin	25	MDA-MB-231	+1,3	3,5	4,5	4.21
+Histamin	50	HeLa	+82,0	0,1	8,2	4.22
			-37,2	0,4	-14,8	4.22

Abbildung 4.30: Zusammenfassung der Kapazitätswerte der präsentierten Versuche, wenn verschiedene Wirkstoffe zur Zellkultur zugegeben wurden.

Wie die erste Tabelle 4.29 gezeigt hat, können Vergleiche der absoluten Sensorkapazität nur mit gleichen Zelllinien durchgeführt werden. Da relativ viele Versuche mit MCF-7-Zellen durchgeführt wurden, können die Werte der äquivalenten Kapazität für MCF-7-Zellen in der folgenden Tabelle zusammengefasst werden:

Zellkulturzustand	Steigung s(nF/Std.)	Dauer	Erreichte Wert w(nF)
Zellwachstum	$-1,3 < s < 0$	Tagen	$20 - 30$
Zellsterben	$s > 0$	Min.-Tagen	$w > 60$
Morphologie/Adhäsion	$s > 0$	Min.-Tagen	$w < 50$
Morphologie/Adhäsion	$s < -1,5$	Min.-Tagen	$w < 50$

Abbildung 4.31: Auswertung der Kapazität für die Versuche mit MCF-7-Zellen

- Erkennung der Adhäsion/Morphologie-Änderungen oder des Zellwachstums für MCF-7-Zellen
  Nachdem die Tabelle 4.31 erstellt wurde, kann eine Methode implementiert werden, um die Bereiche der Kapazitätskurve zu erkennen, wo es Zellwachstum gibt, und wo es Morphologie-/Adhäsions-Änderungen gibt. Der Algorithmus basiert auf den Messergebnissen der Tabelle 4.31. Wenn Adhäsions-/Morphologie-Änderungen oder Zellsterben auftreten, ist die Steigung der Kapazität entweder positiv oder kleiner als eine bestimmte Schwelle (in der Tabelle $s < -1,5$). Das Ergebnis dieses einfachen Algorithmus mit einer Schwelle gleich $-1,6$ wird für den Versuch nach 4.18 in Abbildung 4.32 gezeigt. Die Ableitung wurde durch die Berechnung der Steigung von Regressionsgeraden ausgewertet.

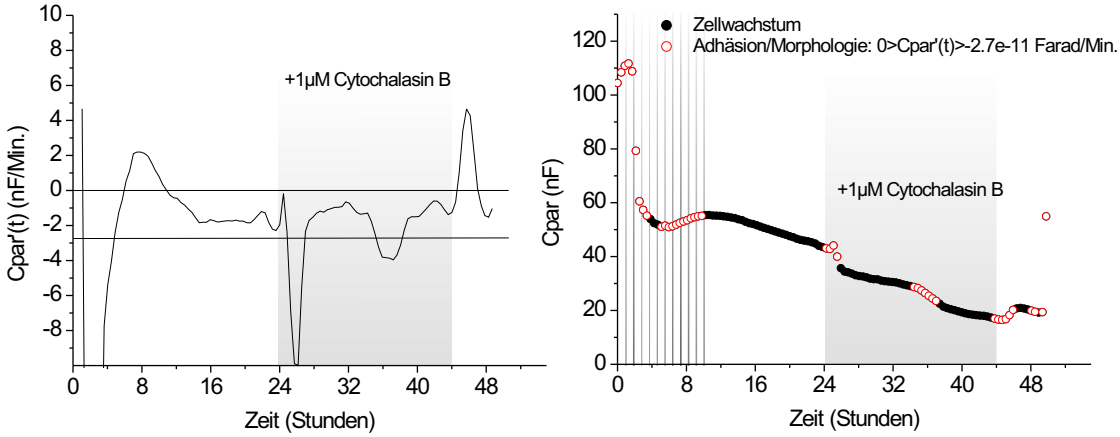


Abbildung 4.32: Ableitung der äquivalenten parallelen Kapazität (links). Bereiche der Messkurve, wo Zelladhäsions-Änderungen detektiert werden (rechts, rote Punkte).

- Vergleich von zwei Messergebnissen: Kontrolle/Test-Versuch
  Es wurde bereits gezeigt, dass die absoluten Kapazitätswerte und ihr Verlauf von den Zellentypen abhängen. In diesem Abschnitt wird erklärt, welche Informationen der Vergleich der Steigung von zwei Kapazitätswerten für 2 Versuche mit identischen Zellen (eine Kontrolle und ein Testversuch) bringt.

  Gegeben ist die Steigung der Kapazitätskurve für die Kontrolle s_1 und die Steigung s_2 der Kapazitätskurve für den Testversuch. Es wird angenommen, dass die Kontrolle eine maximale Wachstumsrate ($s_1 < 0$) zeigt, dass die zwei Versuche mit den gleichen Bedingungen gestartet wurden, und dass sie sich in der Log-Phase befinden. In Tabelle 4.33 wird der Vergleich durchgeführt.

Vergleich der Steigung	Mögliche Interpretationen für den Testversuch
$s_1 = s_2$	-gleiche Wachstumsrate -kleinere Wachstumsrate und Zell-Substrat-Adhäsion größer
$s_2 < 0$, $s_1 < s_2$	-kleinere Wachstumsrate -gleiche Wachstumsrate und Zell-Substrat-Adhäsion kleiner
$s_2 < 0$, $s_1 > s_2$	-größere Zell-Substrat Kontakte.
$s_2 = 0$	-kein Wachstum -Wachstum vorhanden aber Zell-Substrat-Adhäsion kleiner
$s_2 > 0$	-Membranen beschädigt, Zellsterben -Zell-Substrat-Adhäsion wird kleiner

Abbildung 4.33: Vergleich der Kapazitätssteigung zwischen einem Kontroll- (s_1 in nF/Std.) und Testversuch (s_2 in nF/Std.) in der Log-Phase.

Die Tabelle 4.33 zeigt, dass mehrere Interpretationen möglich sind, wenn die Kulturen in der Log-Phase sind. Wenn man zusätzlich die Dauer des Vergleichs und den erreichten Kapazitätswert betrachtet, werden einige Interpretationen ausgeschlossen. Wenn schnelle Änderungen im Kapazitätswert auftreten, bedeutet es, dass Änderungen in der Adhäsion oder Morphologie aufgetreten sind. Wenn langsame Änderungen in der Adhäsion oder Morphologie auftreten, ist die Interpretation schwieriger. Um eine eindeutige Interpretation zu erhalten, können dann mikroskopische Aufnahmen gebraucht werden, um Information über das Zellwachstum zu erhalten, wie es im letzten Abschnitt dieses Kapitels erklärt wird.

4.6 Weiterverarbeitung der Messergebnisse

Nachdem die Evaluierung der Ansäuerungsrate, Sauerstoff-Verbrauchsrate und Impedanz-Auswertungen erklärt wurden, werden Methoden für die Weiterverarbeitung der Ergebnisse in diesem Abschnitt präsentiert. Die Methoden haben zum Ziel, verschiedene wichtige Eigenschaften einer Messreihe auszuwerten. Zuerst wird ein Algorithmus für die Auswertung einer einzigen Messkurve präsentiert. Danach werden statistische Methoden für die Auswertung von parallelen Versuchen vorgestellt.

4.6.1 Auswertung einer einzigen Messkurve

Ziel der Zell-Chip-Versuche ist es, zu determinieren, ob Wirkstoffe einen Effekt auf die Zellmetabolismusraten (Ansäuerungsrate oder Sauerstoffverbrauchsrate) oder Impedanz-Messung verursachen. Wenn ja, muss die Art des Effekts determiniert werden, sowie der Zeitpunkt, wann der Effekt gemessen wurde.

Um automatisch auszuwerten, ob ein Effekt aufgetreten ist, nachdem ein Wirkstoff hinzugefügt wurde, wird eine Schätzung der Messwerte berechnet und mit dem Messsignal verglichen.

In einer ersten Methode kann eine Schätzung der zukünftigen Messwerte durch den Mittelwert der Messpunkte eine Stunde vor dem Einfügen des Wirkstoffs erhalten werden. In den Zell-Chip-Versuchen werden die Metabolismusraten alle 10 Minuten erfasst. Die Messkurve besteht deshalb aus einer Reihe von regelmäßig erfassenden Messpunkten (t_i, y_i) $(i = 1...\infty)$. Zu der Zeit t_6 ist die Schätzung der zukünftigen Messwerten gleich $\hat{y} = \sum_{i=1}^{i=6} y_i/6$. Die Standard-Abweichung beträgt σ_y. Die neuen Messwerte $(t_7, y_7)...(t_j, y_j)$ werden mit $\hat{y}$ verglichen. Wenn es mindestens 3 Messpunkte gibt $(t_n, y_n)...(t_{n+3}, y_{n+3})$, die einen Messwert deutlich grösser oder kleiner als $\hat{y}$ zeigen, wird ein Effekt zu

der Zeit t_n gemessen: $\forall k$: $n \leq k < n+4$; $|y_k - \hat{y}| > 2\sigma_y$. Diese Schätzung hat aber den folgenden Nachteil: wenn die Messkurve nicht konstant vor dem Einfügen des Wirkstoffs ist, wird immer ein Effekt registriert. Um diesen Nachteil auszugleichen, kann eine andere Schätzung berechnet werden.

In einer zweiten Methode kann die Schätzung der Messwerte durch eine lineare Regression $\hat{y}(t) = a_1 t + b_1$ erhalten werden. Eine Stunde vor dem Einfügen des Wirkstoffs wird die Regression und Standard-Abweichung σ_y berechnet werden. Nach dem Einfügen des Wirkstoffs werden die neuen Messwerte zu $\hat{y}(t_i) = a_1 t_i + b_1$ verglichen. Wenn $\forall k$: $n \leq k < n+4$; $|y_k - \hat{y}(t_k)| > 2\sigma_y$, wird ein Effekt zur Zeit t_n gemessen. Eine neue Regressionsgerade nach dem Auftreten des Effekts wird berechnet. Diese Methode ist in Abbildung 4.35 dargestellt.

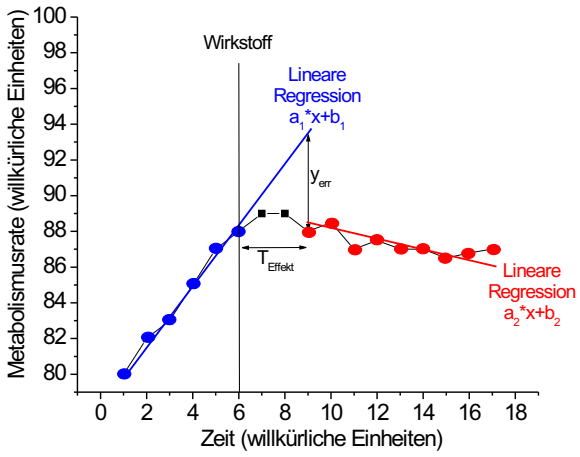


Abbildung 4.34: Evaluation einer Regressionsgerade vor der Zugabe des Wirkstoffs. Evaluation des Effekts des Wirkstoffs.

Mit dieser Methode erhält man mehrere Parameter: die Zeit T_{Effekt} bezeichnet, wann der Effekt aufgetreten ist sowie die Art des Effekts (Steigung oder Abfall der Metabolismusraten). Die Auswertung dieser Parameter wird in Tabelle 4.35 für den Versuch 4.10 dargestellt.

Messparameter	Wirkstoff	Konzentration($\mu M.$)	Effekt?	T_{Effekt}(Std.)	Typ des Effekts
O_2	+CB	1	ja	0,5	Steigung
pH	+CB	1	ja	0,5	Abfall

Abbildung 4.35: Auswertung des Effekts der Zugabe von Cytochalasin B(CB) für den Versuch 4.10.

4.6.2 Auswertung einer Messreihe

Die intrinsische Heterogenität zellulären Materials erfordert umfangreiche Messreihen, mit denen statistische Auswertungen berechnet werden. Wichtige Parameter der statistischen Auswertungen sind die Streuung identischer Messungen und der Unterschied zwischen Kontroll- und Test-Versuchen.

- Streuung einer Datenreihe
  Die Streuung einer Messreihe ist ein Maß für die Reproduzierbarkeit der Messergebnisse. Zu jedem Zeitpunkt der Messung wird sie durch die Standard-Abweichungen der Messpunkte von allen Messreihen charakterisiert.

- Unterschied zwischen Kontroll- und Test-Versuch
  Bei Zell-Chip-Versuchen werden wie bei klassischen Zellversuchen meistens mehrere Kontrollversuche parallel zur Versuchsreihe durchgeführt. Vergleiche zwischen Testversuchen und Kon-

trollversuchen müssen berechnet werden. Zu einem gegeben Zeitpunkt werden sie durch den Vergleich zwischen dem Mittelwert der Kontrollmessreihen und Testversuchsreihen sowie ihrer entsprechenden Streuung errechnet.

4.7 Vitalitätsindex

4.7.1 Bedeutung des Index

Toxizitätsanalyse oder Chemosensitivitätstest sind sehr wichtige Anwendungsbereiche der Zell-Chip-Systeme. Ziel dieser Anwendungen ist es, zu messen, wie schädlich gegebene Wirkstoffe für die Zellkultur sind. Um die Schädlichkeit der Wirkstoffe auszuwerten, wird ein Vitalitätsindex gebraucht. In den Zellkulturarbeitsmethoden wird ein ähnlicher Index (Viabilitätsindex) schon definiert, der bestimmt wird, indem man das Verhältnis zwischen lebenden Zellen und gesamter Zellzahl (lebend und tot) berechnet. Dieser Index kann durch bestimmte Farbstoffe ermittelt werden. Zum Beispiel lagert sich der Farbstoff „Trypan blau“ nur in toten Zellen ein und die Intensität der blauen Farbe ist proportional zu der Anzahl von toten Zellen. Da es mit Zell-Chip-Systemen möglich ist, die metabolische Aktivität der Zellkultur in Realzeit zu messen, kann man einen feinen Index zu jedem Zeitpunkt der Messung und ohne Einfluss auf die Zellkultur definieren.

Die intuitive Bedeutung und die definierte Skala des Vitalitätsindex werden in der Abbildung 4.36 aufgezeigt. Der Index wird als ganze Zahl definiert. 100 ist der Referenzwert, bevor der Wirkstoff zugegeben wird, wenn die Zellen eine normale metabolische Aktivität zeigen. Wenn der Index gleich 0 ist, bedeutet es, dass keine Zellen mehr am Leben sind.

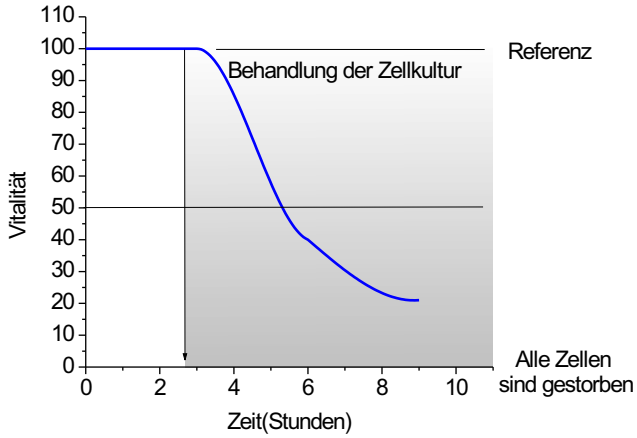


Abbildung 4.36: Definition einer Skala für den Vitalitätsindex.

4.7.2 Bestimmung des Vitalitätsindex für tierische Zellen

Ziel dieses Abschnitts ist es, Methoden vorzuschlagen, um einen Vitalitätsindex durch die Messung von pH-, O_2- und Impedanz-Werten in der Zellenmikroumgebung zu erhalten. Da die Messsignale von den Zellen abhängen, wird in diesem Abschnitt die Bestimmung des Index für tierische Zellen behandelt. Zuerst wird noch einmal gezeigt, welche Bedeutung die Messparameter haben. Danach werden zwei Methoden für die Bestimmung des Vitalitätsindex präsentiert.

- Bedeutung der Sauerstoffverbrauchs- und Ansäuerungsrate und ihrer Variationen für tierische Zellen

In Kapitel 2 (über die Zellen) wurden mehrere wichtige Eigenschaften des tierischen Zellmetabolismus beschrieben. Es wurde erklärt, dass 90% des Sauerstoffs in den Mitochondrien verbraucht wird, um ATP zu generieren. Der Sauerstoffverbrauch der Zelle reflektiert die ATP-Produktion in den Mitochondrien. Es wurde auch erklärt, dass die Zellen durch ihren Metabolismus mehrere Säuren produzieren, wie Milchsäure oder CO_2 ($CO_2 + H_2O \rightarrow H_2CO_3 \rightleftharpoons H^+ + HCO_3^-$). Das CO_2 ist ein Produkt der zellulären Atmung. Milchsäure ist ein Produkt der anaeroben Atmung.

Wenn ein Zell-Chip-Versuch durchgeführt wird, können die Ansäuerungsrate und Sauerstoffverbrauchsrate entweder steigen, konstant bleiben oder sinken. Wenn zum Beispiel die Sauerstoffverbrauchsrate steigt, heißt es, dass die mitochondriale Aktivität steigt. Die Zelle ist sehr aktiv und verbraucht viel ATP. Wenn die Sauerstoffverbrauchsrate sinkt und die Ansäuerungsrate steigt, bedeutet es, dass es vermutlich ein Problem in den Mitochondrien gibt: die Zellen benutzten den anaeroben Metabolismusweg, um ATP zu produzieren, was viel Milchsäure erzeugt. Wenn die Sauerstoffverbrauchsrate und die Ansäuerung sinken, bedeutet es, dass die Zelle weniger aktiv ist. Für die Erhaltung des Vitalitätsindex werden solche Interpretationen gebraucht, um einen Zusammenhang zwischen den Messparametern der Zell-Chip-Systeme und dem Energieverbrauch der Zellen zu erhalten.

- Methoden für die Auswertung des Index
  In einer ersten Methode wird nur die Zelldichte betrachtet, die durch Impedanz-Messung, Mikroskopie oder andere Verfahren bestimmt werden kann. In einer zweiten Methode werden die Sauerstoffverbrauchsrate und die Ansäuerungsrate betrachtet.

  – Erste Methode
    Für die Auswertung des Vitalitätsindex wird zuerst die Zelldichte betrachtet. Es wird angenommen, dass zwei parallele Versuche laufen: ein Kontrollversuch, in dem die Zellen eine maximale Metabolismusrate zeigen, und ein Testversuch, in dem ein toxischer Wirkstoff zugegeben wird. Die Zelldichte des Testversuchs (bzw. Kontrollversuchs) zu der Zeit t ist mit $n(t)$ (bzw. $n_0(t)$) notiert. Der Index ist direkt proportional zur Beziehung der Zelldichte des Testversuchs und der Zelldichte des Kontrollversuchs. Wenn $n(t) = 0$, ist $V(t) = 0$. Wenn $n(t) = n_0(t)$, ist $V(t) = 100$. Wenn $E[]$ die Funktion darstellt, die den ganzen Teil einer reellen Zahl wiedergibt, ist der Index gleich:

$$V(t) = E\left[\frac{100 \cdot n(t)}{n_0(t)}\right] \tag{4.15}$$

  – Zweite Methode
    In dieser Methode wird die Energieproduktion durch die Messung der Metabolismusraten bestimmt. Der Vitalitätsindex wird durch die relative Energieproduktion ermittelt. Wenn der Versuch gestartet wird, haben die Zellen einen „normalen“ Metabolismus. Die Startwerte der Sauerstoffverbrauchsrate (O_0) und Ansäuerungsrate (A_0) dienen als Referenz. Wenn ein Wirkstoff hinzugefügt wird, können die Raten sich ändern. Ihre Variation charakterisiert den Zustand der Zellkultur. Abbildung 4.37 zeigt die Variationen der Metabolismusraten für acht durchgeführte Zell-Chip-Versuche.

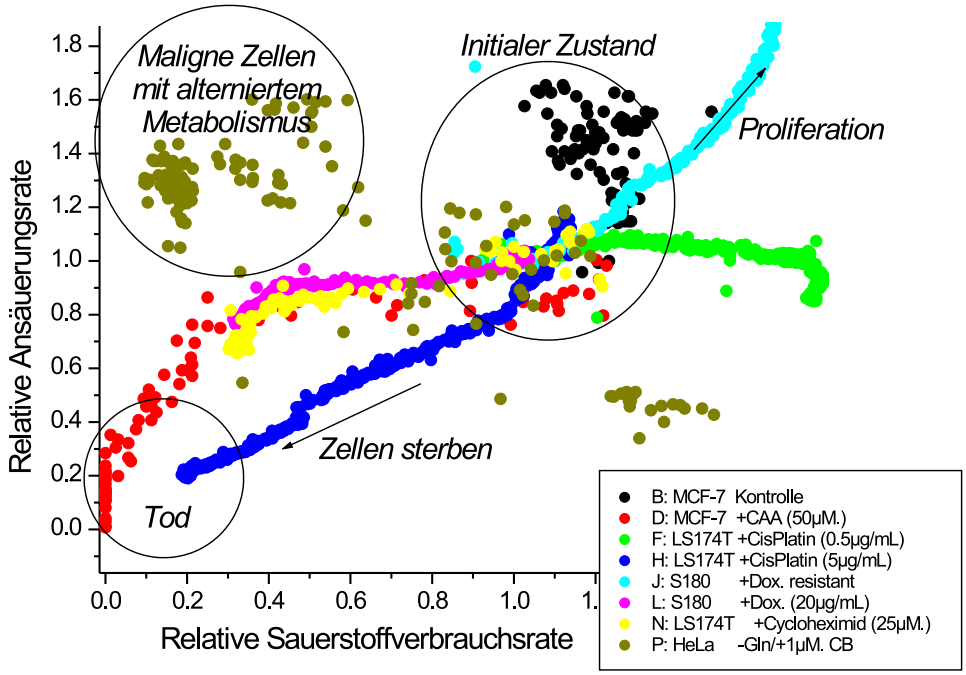


Abbildung 4.37: Variationen der Metabolismusraten für acht verschiedene Zell-Chip-Versuche.

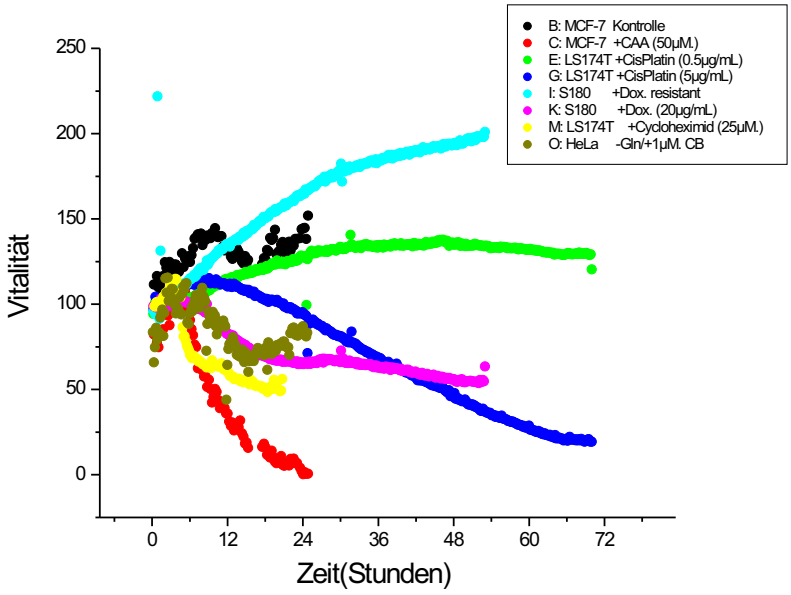


Abbildung 4.38: Ermittlung des Vitalitätsindex für die Versuche 4.37

Der Vitalitätsindex V kann durch eine Funktion der Metabolismusraten definiert werden: $V(t) = f(O(t)/O_0, A(t)/A_0)$. Wenn $O(t) = 0$ und $A(t) = 0$, ist $V(t) = 0$. Wenn $O(t)$ oder $A(t)$ steigen, soll $V(t)$ steigen. Man kann zum Beispiel die Funktion $f(x, y) = 50x + 50y$ für die Auswertung des Index benutzen. Wenn $f(x, y) = 50x + 50y$, werden die Vitalitätswerte

für die Versuche 4.37 durch die Kurven der Abbildung 4.38 dargestellt.

4.8 Verarbeitung von mikroskopischen Aufnahmen

Als die Impedanz-Ergebnisse und ihre Verarbeitung präsentiert wurden, wurde festgestellt, dass eine parallele Auswertung des Zellwachstums sehr nützlich wäre. Da mikroskopische Aufnahmen parallel zu den Versuchen mit Glas-Sensorchips durchgeführt werden können, können diese Aufnahmen benutzt werden, um das Zellwachstum auszuwerten. Um eine objektive Auswertung zu erhalten, wurde ein Algorithmus implementiert, der im nächsten Abschnitt beschrieben ist.

4.8.1 Auswertung des mit Zellen besetzten Anteils des Sensorchips

Um den Anteil des mit Zellen besetzten Sensorchips auswerten zu können, wurde ein Bildverarbeitungsalgorithmus implementiert. Dieser Algorithmus wurde mit der Borland C++ Builder 6.0 IDE entwickelt. Der Algorithmus verarbeitet graue Phasenkontrast-Mikroskopie-Aufnahmen, in denen die Zellen eine Fläche von ungefähr 20×20 Pixel besetzten. Jeder Punkt des Bildes (i, j) hat einen Grauwert G (ganze Zahl zwischen 0 und 255: $0 \leq G(i, j) \leq 255$). Der Algorithmus besteht aus mehreren Schritten, die in den nächsten Punkten beschrieben werden.

- IDES-Elektroden
  Zuerst wird erkannt, wo die IDES-Elektroden sind. Die Elektroden haben immer eine sehr dunkle Farbe, deshalb werden alle Bereiche des Fotos, die eine Graustufe unter einer bestimmten Schwelle S_G ($S_G = 90$) haben, als IDES-Elektrode erkannt. Diese Bereiche werden für die Auswertung nicht betrachtet.

- Ableitung
  Danach berechnet der Algorithmus eine Ableitung des Bildes in der Richtung X und Y. Es seien A_i und A_j die Werte der Ableitung in der Richtung X und Y. Die Ableitung wird durch die Berechnung der Steigung einer Regressionsgerade über 4 Punkte ermittelt: $A_i = Reg((i, G(i, j)), ...(i+4, G(i+4, j)))$, $A_j = Reg((j, G(i, j)), ...(j+4, G(i, j+4)))$.

- Erkennung der Zellen
  Gegeben sei eine reelle positive Zahl S_1. Wenn $|A_i| > S_1$ oder $|A_j| > S_1$, wird der Punkt als Grenze einer Zelle erkannt. S_1 wurde gleich 5 gewählt.

  Man erhält dann ein neues Bild: $H(i, j)$. Wenn $|A_i| < S_1$ und $|A_j| < S_1$, ist $H(i, j) = 0$. Sonst ist $H(i, j) = G(i, j)$.

  Danach wird ein Mittelwertfilter berechnet, um ein neues Bild $I(i, j)$ zu erhalten. Mit diesem Filter wird der Mittelwert der Nachbarpunkte für jeden Punkt berechnet. Die Anzahl der Nachbarpunkte wurde gleich 5 gewählt: $I(i, j) = \sum_{k,l=-5}^{k,l=5} G(i + k, j + l)/121$.

  Wenn ein Punkt des erhaltenen Bildes $I(i, j)$ einen grauen Wert größer als eine bestimmte Schwelle S_2 ($S_2 = 60$) hat, wird er als Bereich einer Zelle erkannt. Man erhält dann ein binäres Bild: $Z(i, j)$. Wenn $I(i, j) < S_2$ ist $Z(i, j) = 0$. Sonst ist $Z(i, j) = 1$. Der Teil des Sensorchips (T in %) besetzt mit Zellen wird dann gleich $T = 100 \cdot \sum_{i,j=0}^{i,j=n} Z(i, j)/n^2$ ausgewertet.

  Das Ergebnis dieser Verarbeitungen wird in Abbildung 4.39 dargestellt.

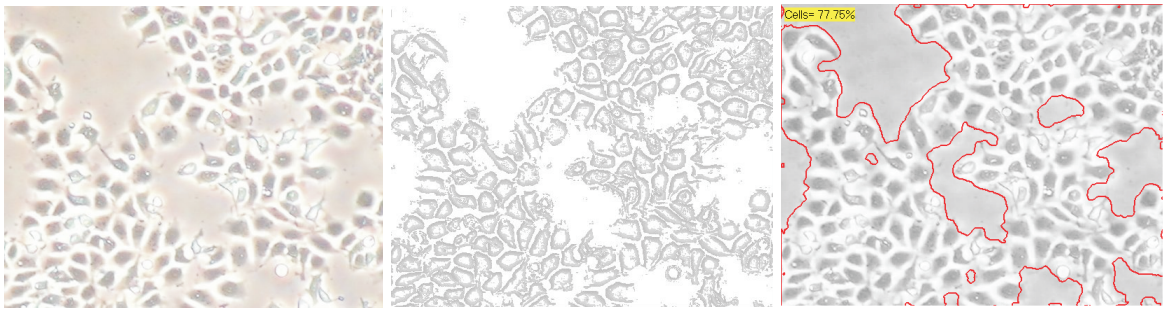
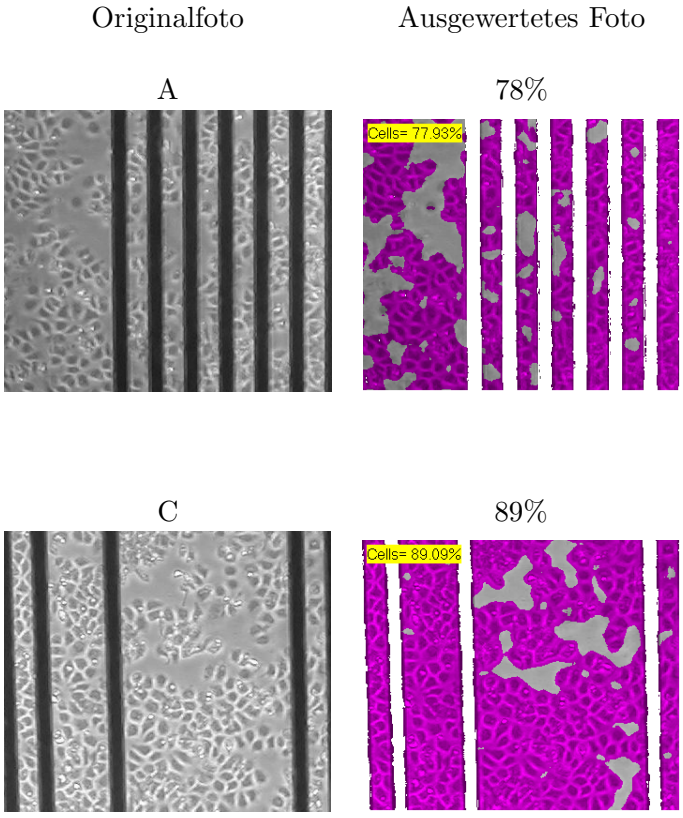


Abbildung 4.39: Zellfotoverarbeitung. Linkes Bild: Originalfoto. Mittleres Bild: Ableitung des Bildes mit Schwellenwert ($H(i,j)$). Rechtes Bild: Umrisse

Als der Impedanz-Versuch gemäß Abb. 4.12 in dieser Arbeit vorgestellt wurde, wurde erwähnt, dass mikroskopische Aufnahmen parallel zu diesem Versuch durchgeführt und ausgewertet wurden. Das Ergebnis der Auswertungen zu verschiedenen Zeitpunkten des Versuchs ist in Abbildung 4.40 dargestellt.



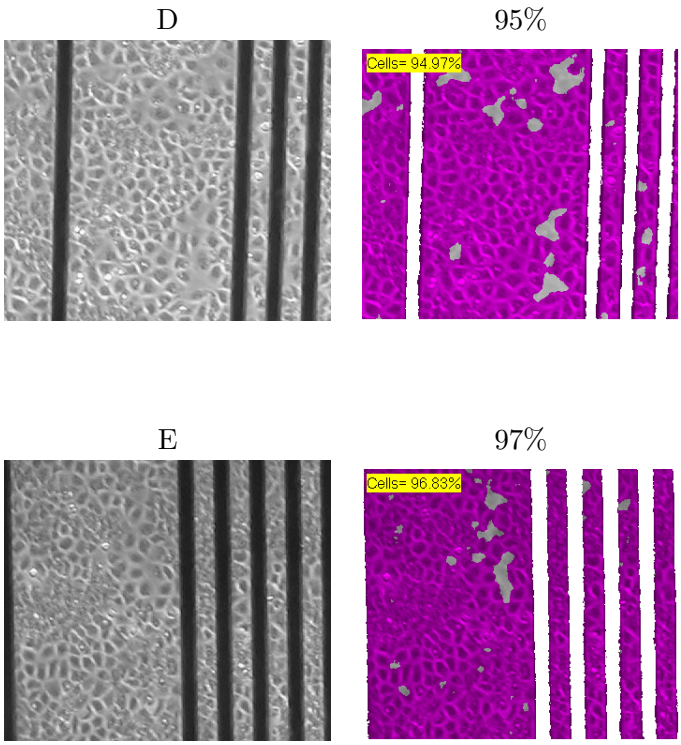


Abbildung 4.40: Auswertung des mit MCF-7-Zellen besetzten Anteils vom Sensorchip für den Versuch nach 4.12 zu verschiedenen Zeitpunkten.

Durch die Auswertung von Abb. 4.40 kann eine Beziehung zwischen Kapazitätswert und mit Zellen besetzter Sensorfläche errechnet werden. Sie wird in Abbildung 4.41 gezeigt. Die mikroskopische Bildverarbeitung ist viel versprechend, um absolute Impedanz-Werte besser auszuwerten.

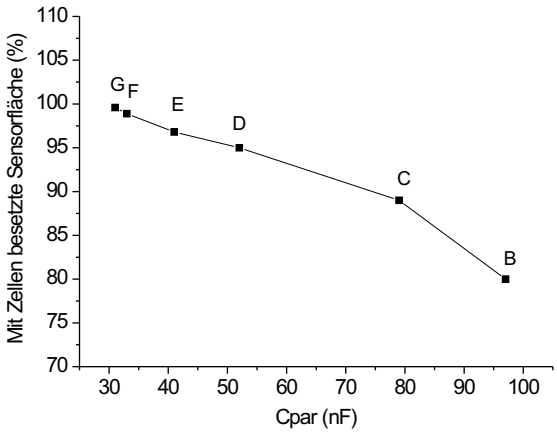


Abbildung 4.41: Beziehung zwischen Kapazitätswert und mit MCF-7-Zellen besetzter Sensorfläche, erhalten durch den Versuch nach Abb. 4.12 und die Auswertungen nach Abb. 4.40.

Kapitel 5

Software

5.1 Einführung

Die Software ist eine essentielle Komponente der Zell-Chip-Systeme, da sie die Schnittstelle zwischen dem Meßsystem und dem Benutzer bildet. Dabei ist es erstens wichtig, dass die Software sehr benutzerfreundlich ist. Benutzerfreundlich bedeutet, dass die Software eine intuitive Bedienoberfläche mit genügend Hilfe, eine klare Darstellung der Messdaten, eine sichere Speicherung der Daten und Erkennung von Hardware- und Benutzer-Eingabefehlern bietet. Zweitens ist es unverzichtbar, dass der Softwarecode sehr gut strukturiert ist, so dass Änderungen der Software relativ leicht durchgeführt werden können. In diesem Kapitel werden zuerst allgemeine Eigenschaften der Software für Zell-Chip-Systeme angeführt. Danach werden ein Zell-Chip-Mess-/Analyse-Programm und zwei Messprogramme präsentiert. Anschließend wird die gewählte Entwicklungsumgebung vorgestellt. Zum Schluss wird die Implementierung der Software erklärt.

5.2 Eigenschaften der Zell-Chip-Software

In diesem Abschnitt werden mehrere Eigenschaften der Zell-Chip-Software präsentiert. Zuerst wird eine allgemeine Software-Architektur beschrieben. Danach wird berechnet, wieviel Daten die Software verwalten muss. Zum Schluss wird die Beziehung zwischen den entwickelten Programmen dieser Arbeit dargestellt.

5.2.1 Allgemeine Architektur

In Kapitel 2 (Material und Methoden) wurde erklärt, dass mehrere Zell-Chip-Messsysteme entwickelt worden sind, die sich hauptsächlich durch die Anzahl der Sensor-Chips, die Zahl der Sensoren pro Chip und die Art der Sensoren unterscheiden. Die Software für alle Zell-Chip-Systeme hat die gleiche Architektur, die in Abbildung 5.1 gezeigt wird.

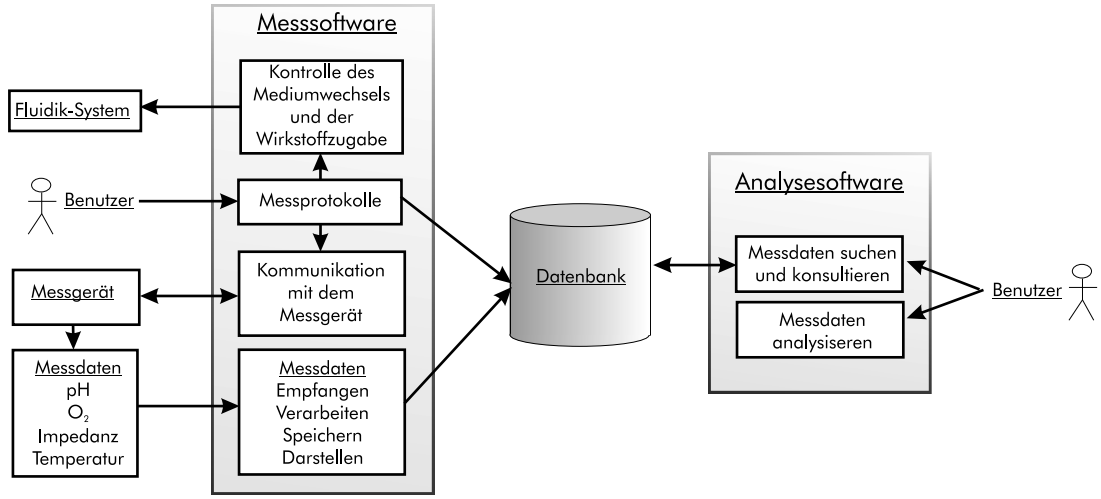


Abbildung 5.1: Architektur und Funktionalitäten der Software für Zell-Chip-Systeme.

Alle Zell-Chip-Meßsysteme brauchen ihre eigene Messsoftware, die mehrere Aufgaben erfüllen muß. Eine wichtige Aufgabe ist die Kommunikation mit dem Benutzer, um das Messprotokoll zu definieren. Eine weitere Aufgabe der Messsoftware ist die Kontrolle des Fluidik-Systems, um Nährmedien periodisch zu wechseln und Wirkstoffe zuzugeben. Andere wichtige Aufgaben sind die Kommunikation mit dem Messgerät, um die Messdaten zu empfangen, die Verarbeitung, die Darstellung und die zuverlässige Speicherung der Messdaten. Eine optionale Aufgabe ist die Übertragung der Messdaten in eine Datenbank.

5.2.2 Datenmenge

In diesem Abschnitt wird berechnet, wieviel Speicherplatz Zell-Chip-Messdaten brauchen. Dafür werden zuerst Computerspeicherplatz-Maßeinheiten definiert. Danach werden theoretische Berechnungen durchgeführt. Zuletzt werden Beispiele gegeben.

- Definitionen
  Alle Daten sind im Computer binär gespeichert. Das Bit (0 oder 1) ist die Speichereinheit. Ein Byte ist eine Gruppierung von 8 Bit, ein KByte von $2^{10} = 1024$ Bytes, ein MByte von 1024 KByte und ein GByte von 1024 MByte. Eine Gleitkomma-Zahl („doppelte“ Genauigkeit) braucht im Arbeitsspeicher 8×8 Bit (8 Byte).

  Der ASCII-Code (American Standard Code For Information Interchange) assoziiert für jeden Tastaturcharakter eine 8-Bit-Zahl. Zum Beispiel hat der Buchstabe A den Code 65.

- Berechnung
  Wenn m die Zahl der Messungen pro Sekunde pro Well, n die Zahl der Sensoren und t die Dauer des Versuches in Tagen sind, ist die Zahl der produzierten Daten $D(m, n, t)$ gleich:

$$D(m, n, t) = m \cdot n \cdot t \cdot 60 \cdot 60 \cdot 24 = m \cdot n \cdot t \cdot 86400 \tag{5.1}$$

  Wenn ein Datum eine Gleitkomma-Zahl ist, und wenn das Datum binär gespeichert wird, braucht es 8 Byte. Der benötigte Speicherplatz $S_1(m, n, t)$ in MByte ist:

$$S_1(m, n, t) = D(m, n, t) \cdot 8/(1024 \cdot 1024) = 0,66 \cdot m \cdot n \cdot t \tag{5.2}$$

  Wenn ein Datum eine Gleitkomma-Zahl ist, und wenn es in ASCII gespeichert wird, kann es 15 Byte brauchen. Der benötigte Speicherplatz $S_2(m, n, t)$ in MByte ist in diesem Fall:

$$S_2(m, n, t) = D(m, n, t) \cdot 15/(1024 \cdot 1024) = 1,23 \cdot m \cdot n \cdot t \tag{5.3}$$

- Beispiele
  Zwei Beispiele sind hier gegeben.

  1. $m = 1/5$, $n = 3$, $t = 1$ und 1 Sensor-Chip
     Mit 1 Messung alle 5 Sekunden, 3 Sensoren pro Chip und 1 Tag Messdauer sind S_1 und S_2 gleich: $S_1(1/5, 3, 1) = 0,66 \cdot 3/5 = 0,396$ MByte, $S_2(1/5, 3, 1) = 1,23 \cdot 3/5 = 0,74$ MByte.

  2. $m = 1/5$, $n = 3$, $t = 7$ und 24 Sensor-Chips
     Mit 1 Messung alle 5 Sekunden, 3 Sensoren pro Chip, 7 Tage Messdauer und 24 Sensor-Chips ist der benötigte Speicherplatz, wenn die Daten in ASCII gespeichert werden, gleich: $24 \cdot S_2(1/5, 3, 7) = 124,6$ MByte.

Zurzeit verfügen die Computer über Festplatten, die eine Kapazität von mindestens 40 GByte haben. Im zweiten Fall können mindestens 300 Versuchergebnisse ohne Komprimierung gespeichert werden.

5.2.3 Zusammenhang der entwickelten Softwareprogramme

In Kapitel 3 (Material und Methoden) wurde erklärt, dass mehrere Zell-Chip-Systeme entwickelt wurden, die ihre entsprechende Messsoftware haben. Die ersten entwickelten Messprogramme integrieren keine Datenverarbeitung. Deshalb wurde eine Auswertungssoftware gebraucht, um die Messdaten von allen Messsystemen analysieren zu können. Diese Software stellt wiederum die Messdaten dar und kann sie mit den Algorithmen verarbeiten, die im Kapitel 4 (Messsignale und ihre Bearbeitung) präsentiert wurden. Sie kann auch als Messsoftware für 1 bis 24 Chips benutzt werden. Diese Software wird im Abschnitt „Mess-/Analyse-Software“ dieses Kapitels im Detail beschrieben. In der Abbildung 5.2 werden den Zusammenhang zwischen den älteren Messprogramme und der Mess-/Analyse-Software dargestellt.

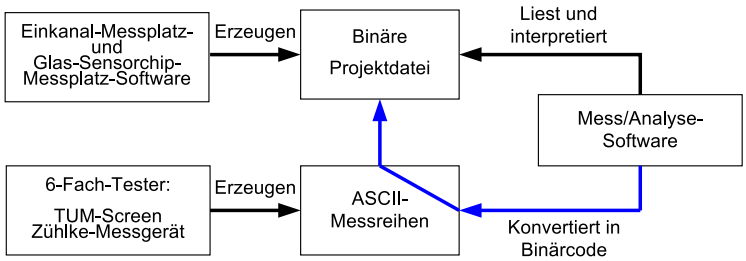


Abbildung 5.2: Zusammenhang zwischen der Mess/Analyse-Software und den Messprogrammen.

Am Ende einer Messung erzeugen die Glas-Sensorchip- und die Einkanal-Messplatz-Software eine binäre Projekt-Datei, die direkt von der Analyse-Software dargestellt und ausgewertet werden kann. Die ältere Programme vom TUM-Screen und Zühlke-Gerät erzeugen ASCII-Messreihen, die in die Analyse-Software importiert, und danach ausgewertet werden können.

5.3 Mess-/Analyse-Software

Wie bereits erklärt, wurden mehrere Zell-Chip-Systeme entwickelt, die über Messsoftwares verfügen, die jedoch keine Datenauswertungen integrieren. Eine Mess-/Analyse-Software wurde deshalb implementiert, um Datenauswertungen von allen Zell-Chip-Messsystemen erhalten zu können. Diese Software bietet auch die Möglichkeit, als Datenerfassungssoftware für einen bis 24 Chips benutzt zu werden. In Abbildung 5.3 wird eine Bildschirmkopie der Mess/Analyse-Software gezeigt. In diesem Abschnitt werden zuerst die Funktionalitäten der Software beschrieben. Danach wird ihre Implementierung erklärt.

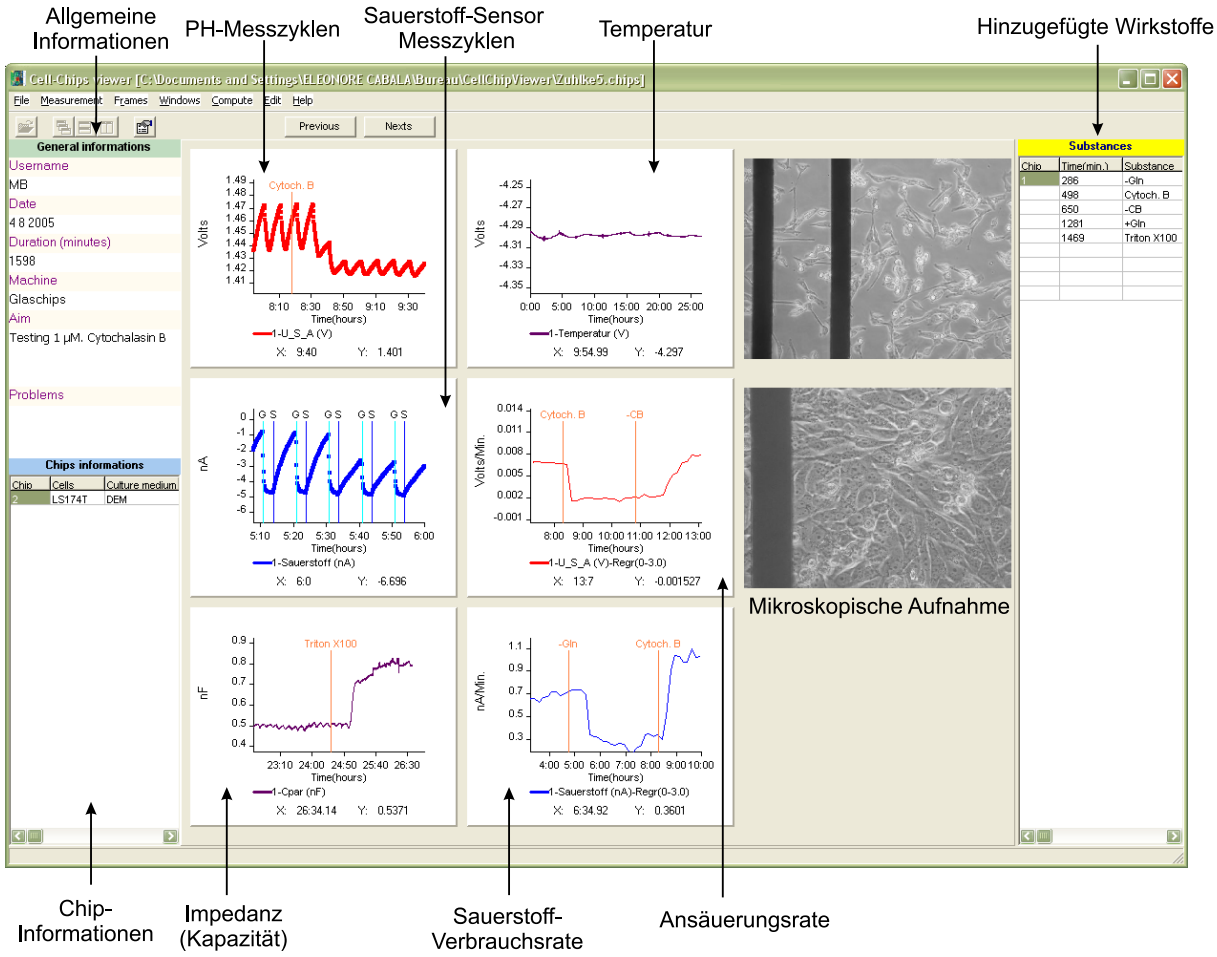


Abbildung 5.3: Bildschirmkopie der Mess-/Analyse-Software. Die Messergebnisse für einen Sensor-Chip und die Versuchsinformationen werden dargestellt. Wenn Versuche mit Glas-Sensorchips durchgeführt werden, können auch mikroskopische Aufnahmen dargestellt werden.

Die Bedienoberfläche der Mess-/Analyse-Software besteht aus einem Hauptfenster mit Menüleiste, die die sieben Hauptmenüs „File“, „Measurement“, „Frames“, „Window“, „Compute“, „Edit“ und „Help“ integriert. Das Hauptfenster kann bis zu vier Rahmen („Frames“) zeigen. Ein erster „Frame“ enthält generelle Informationen über den Versuch, ein anderer enthält Informationen über die Chips, ein dritter, der eine aktive Baumstruktur enthält, erlaubt Messkurven, Bilder und Videos darzustellen, der letzte zeigt Informationen über das Messprotokoll.

Im folgenden wird erklärt, wie die Software installiert wird und wie Messungen mit dieser Software durchgeführt werden können. Danach werden die Analyse-Funktionen der Software beschrieben.

5.3.1 Installation der Software

Die Software besteht aus einer einzigen EXE-Datei: „CellChip.exe“. Um die Software zu installieren, muss die EXE-Datei einfach auf die Festplatte kopiert werden.

5.3.2 Messungen starten und stoppen

Die Software kann als Messsoftware für alle Zell-Chip-Systeme benutzt werden, die über 1 bis 24 Chips verfügen. Um eine Messung zu starten, muss man das Menü „Measurement→Start“ selektieren. Danach öffnen sich mehrere Konfigurationsfenster, die in Abbildung 5.4 gezeigt werden. Es werden

Fenster geöffnet, die erlauben, biologische Versuchsinformationen sowie Messkonfigurationen einzutragen. Wenn man im letzten Konfigurationsfenster auf „Ok“ drückt, wird die Messung gestartet. Alle Messwerte und Versuchsinformationen werden dann automatisch in einem Unterverzeichnis des Programmverzeichnisses gespeichert.

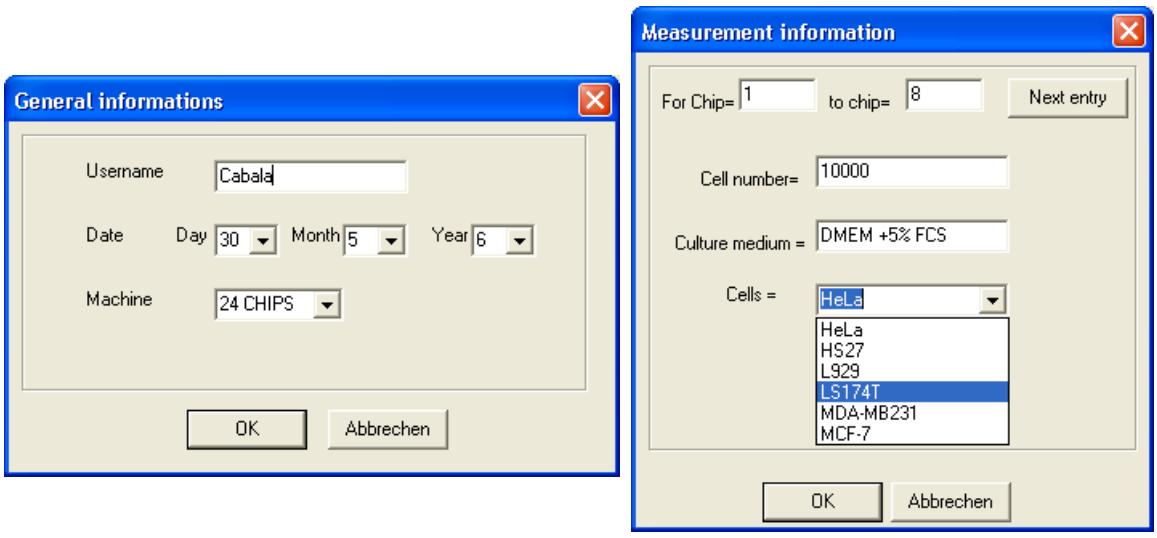


Abbildung 5.4: Dialogfenster für die Eingabe von Versuchsinformationen: allgemeine Informationen, Zellen und Kulturmedium.

Um die eingegebene Messinformationen zu ändern, werden die Menüs „Edit → Cells...“ und „Edit → Edit the general Infos“ benutzt. Die Messwerte und Informationen werden danach in Graphiken und „Frames“ dargestellt.

- Darstellung der Versuchsinformationen
  Die Versuchsinformationen werden in vier verschiedene „Frames“ dargestellt, die durch das Hauptmenü „Frames → ...“ gezeigt werden können. Diese „Frames“ werden in Abbildung 5.5 dargestellt. Der „General Informations Frame“ enthält allgemeine Informationen über den Versuch, wie Datum, Benutzername, Ziel des Versuchs. Der „Graphs Frame“ enthält eine Baumstruktur, die die Graphiken der Versuche in verschiedenen Kategorien (z.B. pH, O_2, Impedanz) gruppiert. Der „Chips Informations Frame“ enthält Informationen über die Chips, die verwendeten Zellen und das Kulturmedium. Zuletzt enthält der „Substances Frame“ Informationen über zugegebene Wirkstoffe.

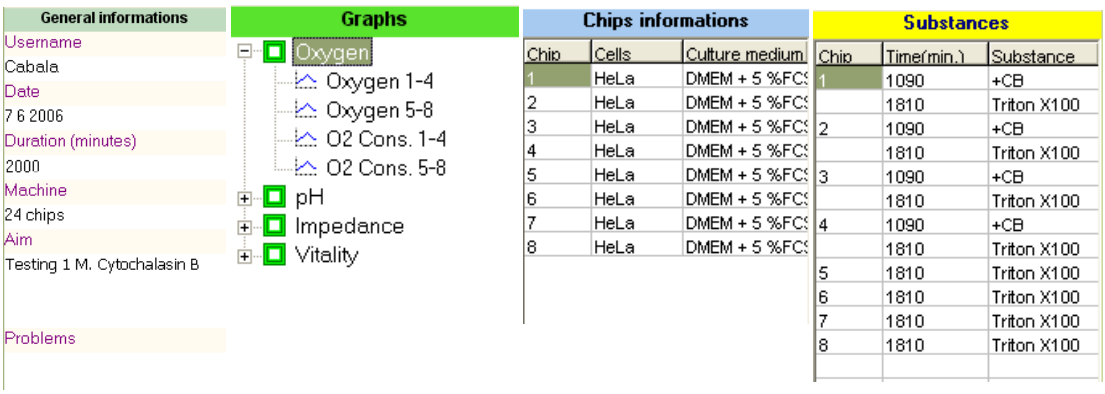


Abbildung 5.5: Verschiedene Informationen werden in Rahmen („Frames“) dargestellt. Von links nach rechts: Generelle Informationen, Graphiken-Baumstruktur, Chip-Informationen und Wirkstoff-Informationen

Der Inhalt der „Frames“ und Kommentare können durch das Menü „Edit → ...“ ausgegeben werden.

- Darstellung der Messkurven
  In Abbildung 5.3 werden die Messkurven für einen Sensor-Chip gezeigt. Um die anderen Messkurven zu sehen, werden die Knöpfe „Previous“ und „Next“ in der Software benutzt, die unter der Menüleiste der Software stehen (siehe Abbildung 5.8). Eine oder mehrere Messkurven mit Versuchsinformationen werden in einer kleinen Graphik dargestellt, die in Abbildung 5.6 gezeigt wird.

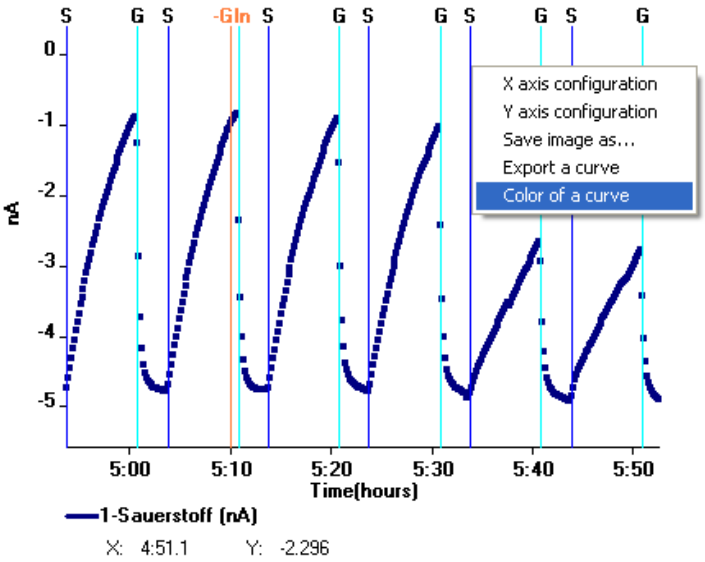


Abbildung 5.6: Fenster, in dem eine annotierte Messkurve (hier: Sauerstoff-Sensorstrom) dargestellt wird.

Verschiedene Graphik-Funktionalitäten erhält man durch Maus und Tastatur-Betätigung. Diese Funktionalitäten sind in Tabelle 5.7 angegeben. Wenn man auf der Graphik rechts klickt, wird ein Menü geöffnet, das verschiedene Funktionen bietet: zum Beispiel können die Farben der Kurven geändert werden, oder eine Kurve kann als ASCII-Messreihe (Textdatei) exportiert werden. Durch das Graphik-Menü „X axis configuration“ können die Messzeiten in Minuten oder in Stunden gezeigt werden. Mit bestimmten Tasten können die Pumpzustände und andere Informationen, wie das Einfügen von Wirkstoffen auf den Graphiken eingezeichnet werden. Mit dem Mausrad kann man Bereiche der Messkurven vergrößern oder verkleinern. Mit den Pfeiltasten kann man die Grenzen der Kurvendarstellung ändern.

Objekt	Betätigung	Effekt
Maus	Rechter-Knopf-Klick	Öffnen des Menüs
	Linker-Knopf-Klick	Die Position des Mauskursors in Messkurven-Koordinaten wird gegeben
	Mausrad, 1. Klick	Beginn des Zeichnens eines Rechtecks
	Mausrad, 2. Klick	Vergrößerung der Messkurve im Bereich des Rechtecks
	Mausrad ↑	Vergrößerung der Messkurve
	Mausrad ↓	Verkleinerung der Messkurve
Tastatur	S	Zeigt/versteckt die Wirkstoffe
	P	Zeigt/versteckt die Pumpzustände
	* oder G	X-Achsenskala wird größer
	/ oder J	X-Achsenskala wird kleiner
	+ oder H	Y-Achsenskala wird größer
	- oder K	Y-Achsenskala wird kleiner
	U	Die Messpunkte werden als einzelne Rechtecke dargestellt
	I	Die Messpunkte werden als einzelne Punkte dargestellt oder durch eine Linie verbunden
	O	Die Linie $y = 0$ wird dargestellt oder versteckt
	←	Neue X-Achsengrenzen (mit $a = (x_{max} - x_{min})/4$): $x_{min} = x_{min} - a$, $x_{max} = x_{max} - a$
	→	Neue X-Achsengrenzen: $x_{min} = x_{min} + a$, $x_{max} = x_{max} + a$
	↑	Neue Y-Achsengrenzen (mit $b = (y_{max} - y_{min})/4$): $y_{min} = y_{min} + b$, $y_{max} = y_{max} + b$
	↓	Neue Y-Achsengrenzen: $y_{min} = y_{min} - b$, $y_{max} = y_{max} - b$

Abbildung 5.7: Graphik-Funktionalitäten, die man durch Maus und Tastatur-Ereignisse erhält.

- Darstellung von Multimedia-Daten
  Wenn Versuche mit Glas-Sensorchips durchgeführt werden, besteht die Möglichkeit mikroskopische Abbildungen aufzunehmen oder Versuche mit einem Fluoreszenz-Mikroskop parallel durchzuführen. Deshalb wurde in der Software die Möglichkeit gegeben, Videos und Abbildungen einzufügen und darzustellen (Menü „Edit→Insert ...“). Die Videos müssen im AVI-Format importiert werden. Jedes Bild wird neben den Graphiken dargestellt. Jedes Video wird in einem eigenen Fenster dargestellt.

- Drucken von Messkurven
  Durch das Menü „File→Print...“ können alle Messkurven zum einem gewählten Drucker geschickt werden.

- Wirkstoffzugabe
  Wenn Wirkstoffe ins Kulturmedium eingefügt werden, wird der Knopf „Substances“ benutzt, um Informationen über die Wirkstoffe einzutragen. Dieser Knopf steht unter der Menüleiste der Software, wie in Abbildung 5.8 gezeigt.

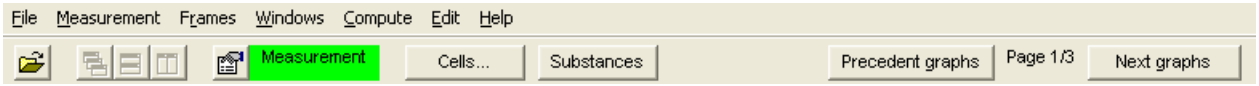


Abbildung 5.8: Menüleiste und Knöpfe der Software

Informationen über die Wirkstoffe (Name, Konzentration) werden im Fenster 5.9 eingetragen.

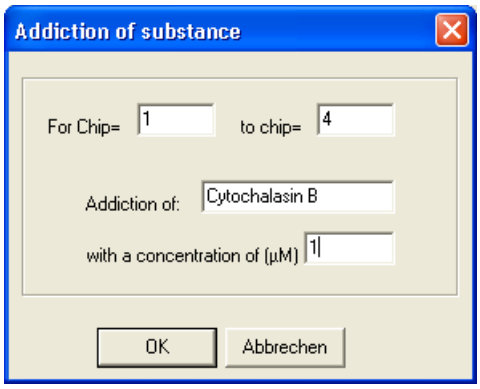


Abbildung 5.9: Dialogfenster für die Eingabe von Wirkstoffinformationen

- Versuch stoppen
  Um den Versuch zu stoppen, muss man auf das Menü „Measurement− >Stop“ klicken.

5.3.3 Messungen importieren und darstellen

Im ersten Teil dieses Abschnitts wird erklärt, wie man Messungen in die Software importieren kann. Danach wird erklärt, wie die Daten dargestellt werden.

- Import von Messreihen
  Das Programm liest binäre Dateien, die ein bestimmtes Format haben. Die Dateinamen der binären Dateien enden mit „.chips“. Die Software kann ASCII-Messreihen zum binären Format konvertieren. Sie enthält eine automatische Konvertierung der ASCII-Messdaten des TUM-Screen- und des Zühlke-Geräts, die durch das Menü „File→Import TUM-Screen Files“ oder „File→Import Zühlke Files“ erhalten werden können. Durch das Menü „File→Import ASCII Files” kann die Software auch beliebige ASCII-Reihen konvertieren, deren Werte mit Tabulatoren getrennt sind, und in denen die Messzeit als erste Spalte in Minuten angegeben wird. Die Pumpzustände müssen als „0“(AUS) und „1“(EIN) in einer Spalte genannt „Pump“ geschrieben werden, wie in den folgenden Zeilen gezeigt wird. Die Zeilen werden mit dem Zeichen „Zeilenende“ abgeschlossen („$\backslash n$“: ASCII Code 10).

```
Time(min)   1-Rpar1(Ohm)   1-Rpar2(Ohm)   1-Cpar1(nF)      1-Cpar2(nF)      Pump
0           702.361        608.546        5.23554e-08      5.00347e-08      0
0.50005     571.468        608.319        5.33002e-08      4.99893e-08      0
1.0001      574.027        607.709        5.33249e-08      4.99942e-08      0
1.50015     574.104        607.929        5.33556e-08      5.00724e-08      1
2.00022     572.35         606.533        5.33323e-08      5.01077e-08      1
2.50027     572.732        605.532        5.34487e-08      5.00729e-08      1
3.00032     573.466        604.816        5.33975e-08      5.00787e-08      1
```

Um eine oder mehrere ASCII-Messreihen zu importierem, muss man auf das Menü „File→Import ASCII Files“ drücken. Danach öffnet sich ein Dialogfenster, das in Abbildung 5.10 dargestellt ist. Zuerst muss der Name einer Projekt-Datei („.chips“) angegeben werden, in der die binären Daten gespeichert werden. Man muss dafür auf den Knopf „Output File“ drücken und einen Dateinamen eintragen. Danach können einige Informationen über den Versuch eingetragen werden, wie Datum, Benutzername, Zellen ...usw. Danach wird die erste ASCII-Datei durch den Knopf

„Import file chip 1“ ausgewählt. Bis zu 6 Dateien können mit diesem Knopf importiert werden. Zuletzt kann man die neue erstellte Projektdatei durch das Hauptmenü „File→Open“ öffnen.

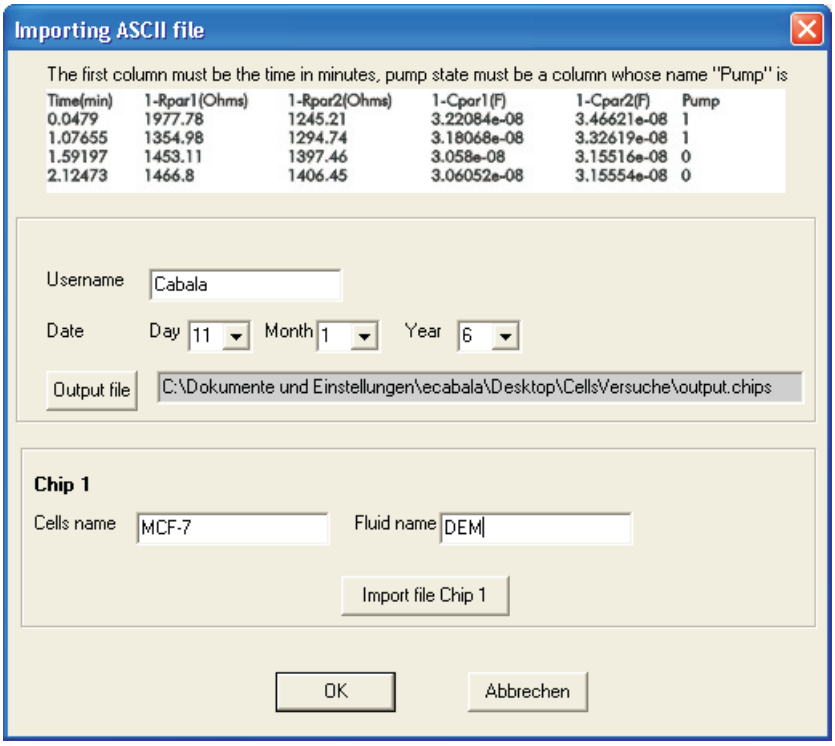


Abbildung 5.10: Dialog-Fenster für den Import von ASCII-Messreihen

Wenn ein Projekt geöffnet wird, oder wenn eine Messung läuft, werden die Versuchsinformationen und die Messkurven gezeigt, wie im vorherigen Abschnitt beschrieben.

5.3.4 Messungen verarbeiten

Nachdem ein Projekt geöffnet wurde, können mehrere Verarbeitungen durchgeführt werden, die in diesem Abschnitt präsentiert werden. Die Datenauswertungen werden im Hauptmenü „Compute“ ausgewählt. Wenn man eine Verarbeitung selektiert, wird ein Dialogfenster geöffnet, damit man die Kurven auswählen kann, die verarbeitet werden sollen, wie in Abbildung 5.11 dargestellt. Mehrere Kurven können durch das gleichzeitiges Betätigen der Taste „Strg“ und der linken Maustaste selektiert werden.

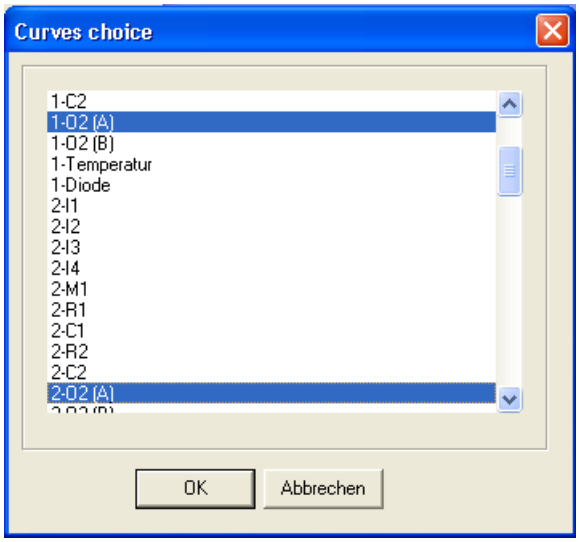


Abbildung 5.11: Wahl einer oder mehrerer Kurven („Strg“ Taste + Linker Mausknopf), die verarbeitet werden sollen.

Wenn die Berechnung durchgeführt wurde, wird eine neue Graphik automatisch erstellt und in der Graphiken-Baumstruktur eingetragen. Die angebotenen Verarbeitungen werden in den folgenden Punkten beschrieben.

- *Linear regression*: Berechnung der Steigung von Regressionsgeraden für jeden Messzyklus einer Messkurve (Bestimmung der Ansäuerungsrate und der Sauerstoffverbrauchsrate)
  Das Programm berechnet für alle Messzyklen die Steigung der Regressionsgeraden während einer beliebigen Zeitspanne (t_{calc}) nach dem Pumpstop. Die Berechnung für einen Zyklus wurde in Kapitel 4 (Messsignale und ihre Bearbeitung) erklärt. Wenn diese Verarbeitung selektiert wird, öffnet sich ein Fensterdialog (siehe Abbildung 5.12), damit man die Zeit t_{calc} eingeben kann.

- *Exponential regression*: Berechnung von exponentiellen Regressionen für jeden Messzyklus einer Messkurve ($a \cdot e^{b(t-t_{OFF})} + c$)
  Wie in Kapitel 4 erklärt, können für bestimmte Sauerstoff-Messzyklen exponentielle Regressionen berechnet werden. Das Programm berechnet eine exponentielle Regression während einer beliebigen Zeitspanne nach dem Pumpstop (t_{calc}) und für alle Zyklen der Messkurve. Zwei Dialogfenster werden gezeigt, damit man die Zeit t_{calc} eingeben und die berechneten Parameter (a, b, c) auswählen kann, die darzustellen sind (siehe Abbildung 5.12).

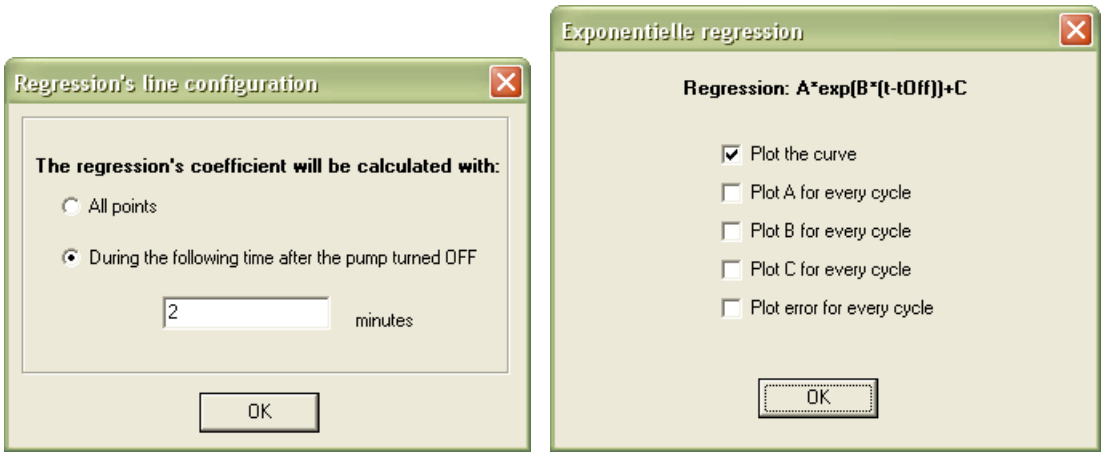


Abbildung 5.12: Dialogfenster für die Berechnung von exponentiellen Regressionen

- *Amplitude* ($y_{ON} - y_{OFF}$): Berechnung der verbrauchten Sauerstoff-Menge oder der produzierten Säure für jeden Messzyklus einer Messkurve
  Das Programm berechnet für jeden Messzyklus den Unterschied zwischen dem y-Wert (y_{ON}), wenn die Pumpe gerade gestoppt wurde, und dem y-Wert, bevor sie wiedereingeschaltet wird (y_{OFF}).

- *Derive*: Berechnung der Ableitung einer Messkurve
  Die Ableitung wird durch eine Berechnung der Steigung der Regressionsgerade alle n Punkte ($n > 2$) bestimmt. Wenn diese Verarbeitung selektiert wird, öffnet sich ein Dialogfenster, damit man die Anzahl der Punkte (n) eingeben kann.

- *Drift correction for* O_2: Berechnung einer Drift-Korrektur für den Sauerstoff-Sensor
  Das Programm berechnet eine Drift-Korrektur für O_2-Messkurven. Für die Berechnung wird angenommen, dass die Sensitivität des Sensors sich mit der Zeit ändert, und dass die gelöste Sauerstoff-Konzentration der Kulturmediumflasche konstant bleibt. Wenn die Pumpe gerade ausgeschaltet wird, soll der gemessene O_2-Wert immer das gleiche Niveau halten (y_{ON}). Wenn $y_{0\text{-}ON}$ das erreichte Niveau für den ersten Messzyklus des Versuchs und $y_{n\text{-}ON}$ für den n^{ten} Messzyklus ist, dann wird der gemessene O_2-Wert (y) für den n^{ten} Messzyklus wie folgt korrigiert: $y' = y_{0\text{-}ON} \cdot y / y_{n\text{-}ON}$

- 10^{-y}: Umwandlung des pH-Werts in H_3O^+-Konzentration
  Das Programm berechnet für alle y-Messwerte einer Kurve den entsprechenden Wert $z = 10^{-y}$.

- *Average between similar curves*: Mittelwert und Standard-Abweichungen von Kurven-Reihen
  Wie wir im Kapitel „Material und Methoden“ gesehen haben, verfügen einige Chips über mehrere pH-, Sauerstoff- oder IDES-Sensoren. Dazu werden immer mehrere parallele Versuche durchgeführt. Es ist deshalb notwendig, den Mittelwert und die Standard-Abweichung von den erhaltenen Kurven-Reihen zu berechnen, was als weitere Datenauswertungsoption in dem Programm angeboten wird.

- *Smoothing*: Berechnung des Mittelwerts von mehreren Messpunkten
  Das Programm berechnet für jeden Messpunkt (x_i, y_i) der Messkurve den Mittelpunkt (x_i, z_i) von den $(2n-1)$ Nachbarpunkten: $z_i = \sum_{k=-n+1}^{k=n-1} y_{k+i}/(2n-1)$

- *Smoothing*(2): Berechnung des Mittelwerts von mehreren Messpunkten
  Das Programm berechnet für alle p Messpunkte $(x_i, y_i)...(x_{i+p}, y_{i+p})$ der Messkurve den Mittelpunkt (X_i, Y_i) dieser p Punkte: $X_i = \sum_{k=i}^{k=i+p} x_k/p$, $Y_i = \sum_{k=i}^{k=i+p} y_k/p$. Wenn die Messkurve n Messpunkte hat, besteht die erhaltene Mittelwert-Kurve aus n/p Punkten.

5.4 Messsoftware des Einkanal-Messgeräts

In diesem Abschnitt werden die Bedienoberfläche und Funktionalitäten der Messsoftware für die erste Version des Einkanal-Messgeräts beschrieben. Die Bedienoberfläche der Messsoftware des Einkanal-Messgeräts besteht aus einem Fenster mit Menüleiste, Graphiken, Knöpfen und Anzeigeelementen, wie in Abbildung 5.13 gezeigt. In der Menüleiste werden verschiedene Funktionalitäten angeboten, wie die Öffnung und Darstellung von gespeicherten Messungen, die Kalibrierung der Sensoren, die Konfiguration der Messung und die Darstellung einer Programmhilfe. Diese Funktionalitäten werden in diesem Abschnitt beschrieben.

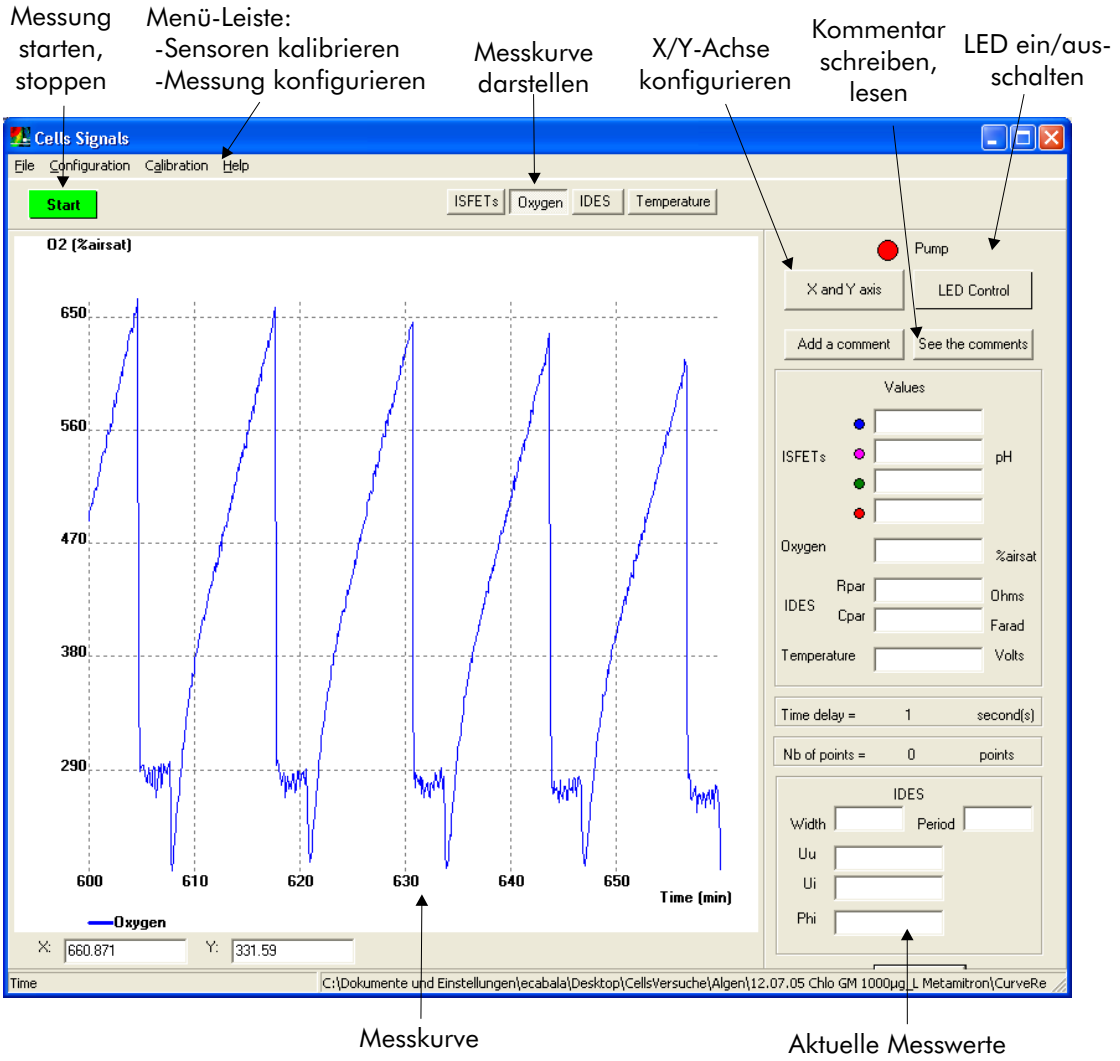


Abbildung 5.13: Bildschirmkopie der Messsoftware des Einkanal-Messgeräts. Darstellung des planaren Clark-Sauerstoffsensorstroms für einen Versuch mit Algen.

Um eine neue Messung zu starten, muss man auf den grünen „Start“-Knopf drücken. Danach werden Fensterdialoge angezeigt, um die Konfiguration einer neuen Messung einzugeben. Ein erstes Fenster dient dazu, Kalibrierungswerte der Sensoren einzugeben. Wenn der Benutzer die Sensoren mit der Software schon kalibriert hat, werden die Werte automatisch in den Fensterfeldern eingetragen. Ein zweites Fenster dient dazu, allgemeine Konfigurationen einzutragen, wie Benutzername, Messfrequenz oder benutzte Sensoren.

Nach dem Start der Messung werden die Messwerte periodisch empfangen und angezeigt. Kommentare können geschrieben werden, und die LED zur Lichterzeugung bei Pflanzenzellen kann ein/ausgeschaltet werden. Die empfangenen Daten werden in der ASCII-Datei *Datas.txt* unter dem Verzeichnis *Messungen\jjmmdd_n* gespeichert (jj:Jahr, mm:Monat, tt:Tag, n:Nummer). Die Benutzerkommentare werden in der Datei *jjmmdd_n.txt* gespeichert. Die Sensormesswerte und Messzeiten werden zusätzlich in den folgenden binären Dateien gespeichert: *ISFET1, ISFET2, ISFET3, ISFET4, Oxygen, Cpar, Rpar, Temperature, SensorsTime*. Diese binären Dateien und die binäre Datei *Calib* werden benutzt, um eine gespeicherte Messung darzustellen. In der Datei *Calib* werden zwei Integer-Zahlen (p und o) nacheinander gespeichert: wenn $p = 1$ (bzw. $o = 1$), bedeutet das, dass die ISFETs-Werte (bzw. Sauerstoff-Werte) in pH-Einheiten (bzw. %airsat) gespeichert wurden. Bei $p = 0$ (bzw. $o = 0$) wurden sie in Volt gespeichert.

Der Benutzer kann eine Benutzerhilfe erhalten, indem er auf das „Help“-Menü klickt. Die Hilfe ist in einer PDF-Datei enthalten, die mit der „Acrobat Reader“-Software dargestellt wird.

5.5 Messsoftware der Glas-Sensorchip-Messgeräte

Die Bedienoberfläche der Glas-Sensorchip-Software besteht aus einem Fenster mit Menüleiste, mehreren Graphiken, Knöpfen und Anzeigenelementen, wie in Abbildung 5.14 gezeigt. Die Installation der Software und ihre Funktionalitäten werden in diesem Abschnitt beschrieben.

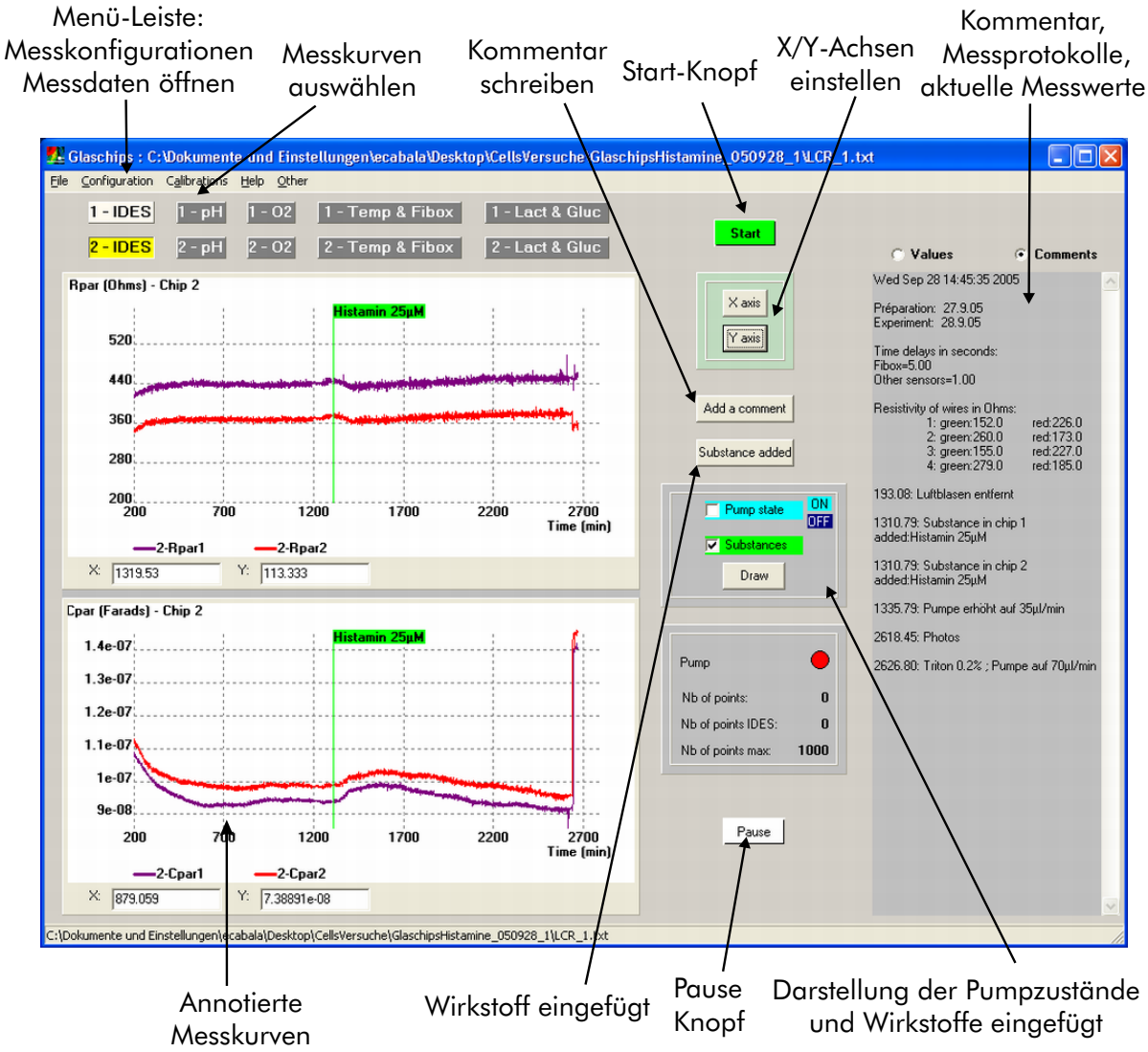


Abbildung 5.14: Bildschirmkopie der Messsoftware des Glas-Sensorchip-Messplatzes. Darstellung des Impedanz-Signals für einen Versuch mit Tumor-Zellen und Zugabe von Histamin.

- Installation der Software
  Die Software besteht aus einer einzigen EXE-Datei: „Glaschip.exe“. Um die Software zu installieren, muss die EXE-Datei einfach auf die Festplatte kopiert werden.

- Menüleiste
  In der Menüleiste werden verschiedene Funktionalitäten angeboten, wie die Öffnung und Darstellung von gespeicherten Messungen, die Kalibrierung der Sensoren, die Konfiguration der Messung und die Darstellung einer Programmhilfe.

Um eine gespeicherte Messung darzustellen, muss man das „File->Open“-Menü anklicken. Dann wird ein Dialogfenster aufgezeigt, um eine Kommentar-Datei auszuwählen. Nachdem eine Kommentardatei selektiert wurde, wird die Software alle gespeicherten Messkurven und die Kommentare dazu darstellen. Die Software liest alle binären Dateien, die sich im gewählten Verzeichnis befinden, und deren Namen dem Namen einer Messkurve entspricht (*1-O2-1, 1-O2-2, 1-Pt100 ...*).

Um die Messintervalle zu konfigurieren, muss man das „Configuration->Time delays“-Menü anklicken. Ein Dialogfenster wird geöffnet, um die Zeiten zu definieren, wie in Abbildung 5.15 gezeigt. Um die serielle Schnittstellennummer zu konfigurieren, muss man das „Configuration->Serial ports“-Menü anklicken. Ein Dialogfenster wird geöffnet, um die Nummer zu definieren, wie in Abbildung 5.15 gezeigt. Diese Messkonfigurationen werden automatisch in der „measure.conf“-Datei im Programmverzeichnis gespeichert. Wenn das Programm startet, werden die letzten Konfigurationsparameter automatisch geladen.

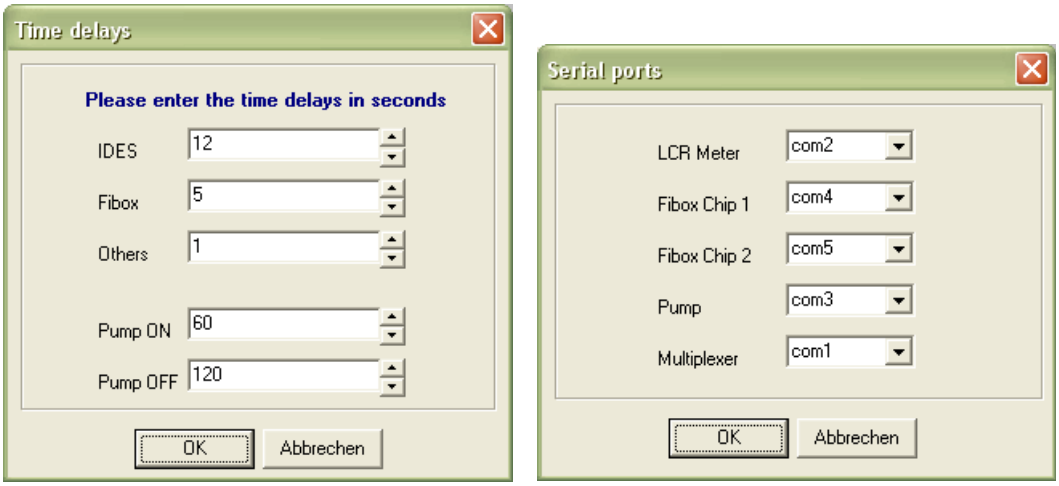


Abbildung 5.15: Fensterdialoge für die Konfiguration der Messintervalle und Messgeräte.

Eine Benutzerhilfe bekommt man, indem man das „Help“-Menü anklickt. Die Hilfe ist eine PDF-Datei, die mit der „Acrobat Reader“-Software dargestellt wird.

- „Start“- und „Pause“-Knöpfe
  Um eine neue Messung zu starten, muss man auf den grünen „Start“ Knopf drücken. Danach werden Dialogfenster gezeigt, um die Konfiguration einer neuen Messung einzugeben. Ein erstes Dialogfenster dient dazu, allgemeine Informationen über den Versuch einzutragen, wie den Namen der Zelllinie, das Kulturmedium, den Benutzer. Ein zweites Dialogfenster dient dazu, Sensoren oder Messgeräte für die Messung auszuwählen.

- „X Axis“- und „Y Axis“-Knöpfe
  Um die X-Achse der Graphiken zu konfigurieren, muss man auf den Knopf „XAxis“ drücken, wodurch sich ein Dialogfenster öffnet. Man kann entweder die 1000 letzten gemessenen Werte in Realzeit darstellen oder die ganze Messkurve. Um die Y-Achsen der Graphiken zu konfigurieren, muss man auf den Knopf „Y Axis“ drücken, wodurch sich ein Dialogfenster öffnet, in dem die minimalen und maximalen Werte der Y-Achse eingegeben werden können. Diese Dialogfenster besitzen die Checkbox „Best Fit“, die es erlaubt, die minimalen und maximalen Werte der Kurven als Grenzen der Achsen zu definieren.

- „Substance added“- und „Add a comment“-Knöpfe
  Während einer laufenden Messung kann der Benutzer Kommentare schreiben, indem er auf den

Knopf „Add a comment“ drückt. Der Kommentar und die aktuelle Messzeit werden dann in die Datei „comments.txt“ geschrieben. Wenn der Benutzer Wirkstoffe ins Kulturmedium einfügt, kann er auf den Knopf „Substance added“ drücken, um den Wirkstoffnamen einzutragen. Die Wirkstoffnamen werden im Kommentar eingetragen und können zusätzlich in die Graphiken eingezeichnet werden (siehe nächster Punkt).

- „Pump“- und „Substances“-Checkbox
  Wenn der Benutzer die Zustände der Pumpe auf den Graphiken darstellen will, kann er die Checkbox „Pump“ anwählen und dann auf den Knopf „Draw“ drücken. Dunkle blaue Linien zeigen, wann die Pumpe ausgeschaltet wurde, helle blaue Linien zeigen, wann sie eingeschaltet wurde. Wenn der Benutzer die Zeiten sehen will, bei denen Wirkstoffe eingefügt wurden, muss er die Checkbox „Substances“ anklicken und dann auf den Knopf „Draw“ drücken.

5.6 Implementierung

5.6.1 „Borland C++ Builder“-Entwicklungsumgebung

„Borland C++ Builder“ ist eine Entwicklungsumgebung (Integrated Development Environment: IDE) für C++ Anwendungen unter dem Windows-Betriebsystem. Die Version 6.0 dieses Kompilers wurde benutzt, um die Software dieser Arbeit zu entwickeln. In diesem Abschnitt werden einige Eigenschaften der Entwicklungsumgebung beschrieben, die nützlich sind, um die Struktur der entwickelten Software zu verstehen. Detaillierte Eigenschaften der IDE sind in ihrer Dokumentation gegeben. Die Wahl der C++ Programmiersprache und der Borland IDE für die Entwicklung der Zell-Chip-Software wurde bereits in Kapitel 1 (Einführung) begründet.

5.6.1.1 VCL-Komponentenbibliothek

Die „Borland C++ Builder“-Entwicklungsumgebung integriert mehrere C++ Bibliotheken. Sie integriert zwei Bibliotheken von graphischen Elementen, die CLX (Component Library for Cross Plattform) und VCL (Visual Component Library) genannt werden. Die CLX-Bibliothek ist eine Multiplatform-Bibliothek, die jedoch weniger Funktionalitäten als VCL integriert. In dieser Arbeit wurde deshalb die VCL-Bibliothek benutzt. Einige wichtige Eigenschaften der VCL-Bibliothek werden in diesem Abschnitt gegeben.

Die VCL-Bibliothek bietet verschiedene graphische Elemente, wie Knöpfe, Feld-Editoren, Frames, Listen, Scrollbar...usw. Zu jedem Element gehört eine C++ Klasse, deren Name mit „T“ anfängt. Zum Beispiel gehört zu dem Fenster-Element die Klasse „TForm“, zu dem Knopf-Element die Klasse „TButton“, zu dem Scrollbar-Element die Klasse „TScrollBar“. Die VCL-Bibliothek bietet auch zahlreiche nicht-graphische Elemente, wie Kommunikationsfunktionen mit SQL-Datenbanken, oder Kommunikationsfunktionen mit Office-Programmen.

Im „C++ Builder“ sind alle VCL-Komponenten in einem Menü dargestellt. Um eine Komponente zu benutzen, wird die Komponente mit der Maus selektiert und auf das Anwendungsfenster transportiert. Der Quellcode der Anwendung wird dann automatisch von der IDE auf den neuesten Stand gebracht. Erstellung von neuen Klassen, Deklaration von neuen Objekt-Instanzen und „include“-Befehle im Quellcode oder Link-Befehle in der Makefile-Datei werden automatisch realisiert.

Beispiele der Verwendung der VCL-Bibliothek sind im nächsten Abschnitt gegeben.

5.6.1.2 Entwicklung von SDI- und MDI-Anwendungen

Unter Windows können mehrere Arten von Anwendungen entwickelt werden: Konsole-Anwendungen, Fenster-Anwendungen ohne untergeordnete Fenster (SDI: Single Document Interface) oder Hauptfenster-Anwendung mit untergeordneten Fenster (MDI: Multiple Document Interface). Wenn man

unter Borland C++ ein neues Projekt startet, wird zuerst gewählt, was für eine Anwendung erstellt wird. Im folgenden Abschnitt wird ein Beispiel einer SDI-Anwendung erklärt. Danach werden einige Eigenschaften der MDI-Anwendungen erklärt, da die präsentierte Mess-/Analyse-Software im letzten Teil dieses Kapitels eine MDI-Anwendung ist.

- SDI-Anwendung
  Für eine SDI-Anwendung erstellt die IDE fünf Dateien, genannt „Unit1.cpp“, „Unit1.h“, „Unit1.dfm“, „Project.bpr“ und „Project1.cpp“. „Unit1.dfm“ ist eine so genannte „Formular-Datei“ und wird von der IDE komplett administriert. Sie enthält Beschreibungen der graphischen Bedienoberfläche: wo und welche graphischen Elemente sich auf dem Hauptfenster befinden, welche Funktionen aufgerufen werden, wenn ein Ereignis auf dem Element auftritt. „Project.bpr“ ist eine XML-Datei, die ähnliche Funktionalitäten wie eine „Makefile“-Datei hat. Sie beschreibt zum Beispiel, welche Dateien zu dem Projekt gehören, welche Packages und Bibliotheken für die Kompilierung des Projektes gebraucht werden, usw... In diesem Abschnitt werden die Inhalte der „Unit1.cpp“-, „Unit1.h“-, „Unit1.dfm“-Dateien für eine sehr einfache Anwendung gezeigt. Die Anwendung ist in Abbildung 5.16 dargestellt. Sie besteht aus einem einzigen Fenster, das einen Knopf und einen Feld-Editor enthält. Jedes Mal wenn der Benutzer auf den Knopf drückt, wird eine interne Variable inkrementiert und ihr Wert wird in das Textfeld geschrieben.

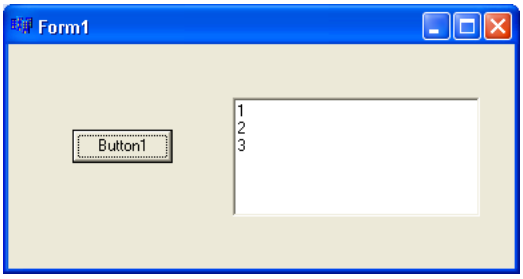


Abbildung 5.16: SDI-Anwendung, die aus einem Hauptfenster mit Knopf und Feld-Editor besteht.

Die Headerdatei „Unit1.h“ enthält die Deklaration der Klasse „TForm1“, die die Deklaration der Objekte und Methoden des Fensters integriert. Der Programmcode dieser Datei ist in den folgenden Zeilen gegeben. „__fastcall“ und „__published“ sind Schlüsselwörter von „C++ Builder“.

```
#ifndef Unit1H
#define Unit1H
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
#include <ComCtrls.hpp>

class TForm1 : public TForm
{
__published: // Von der IDE verwaltete Komponenten
        TButton *Button1;
        TRichEdit *RichEdit1;
        void __fastcall Button1Click(TObject *Sender);
private: // Anwender-Deklarationen
```

```
public: // Anwender-Deklarationen
        __fastcall TForm1(TComponent* Owner);
};

extern PACKAGE TForm1 *Form1;
#endif
```

Die Datei „Unit1.cpp“ enthält den Code der Methoden, die in „Unit1.h“ deklariert wurden. Jedes Mal, wenn der Benutzer auf den Knopf drückt, wird die Methode „Button1Click“ aufgerufen. Der Programmcode dieser Datei ist in den folgenden Zeilen gegeben.

```
#include "Unit1.h"

#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;

__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{}

void __fastcall TForm1::Button1Click(TObject *Sender)
{
        static int i=0;
        i+=1;
        RichEdit1->Lines->Add(i);
}
```

Die Datei „Unit1.dfm“ enthält die Beschreibung aller graphischen Elemente der Anwendung: die Eigenschaften des Hauptfensters „Form1“, des Knopfs „Button1“ und des Textfeldes „RichEdit1“ werden in dieser Datei automatisch von der IDE beschrieben. Die Eigenschaften der graphischen Elemente können auch im C++ Quellcode eingegeben werden. Zum Beispiel kann die Position des Knopfes „Button1“ durch sein Attribut „Left“ geändert werden: `Button1->Left=100`. Der Programmcode der „Unit1.dfm“-Datei ist in den folgenden Zeilen gegeben.

```
object Form1: TForm1
  Left = 363
  Top = 145
  Width = 387
  Height = 202
  Caption = 'Form1'
  Color = clBtnFace
  Font.Charset = DEFAULT_CHARSET
  Font.Color = clWindowText
  Font.Height = -11
  Font.Name = 'MS Sans Serif'
  Font.Style = []
  OldCreateOrder = False
  PixelsPerInch = 96
  TextHeight = 13
  object Button1: TButton
```

```
    Left = 48
    Top = 64
    Width = 75
    Height = 25
    Caption = 'Button1'
    TabOrder = 0
    OnClick = Button1Click
  end
  object RichEdit1: TRichEdit
    Left = 168
    Top = 40
    Width = 185
    Height = 89
    TabOrder = 1
  end
end
```

Weitere Informationen über SDI-Anwendungen unter Borland C++ sind in der Dokumentation der IDE enthalten.

- MDI-Anwendung
  Für eine neue MDI-Anwendung erstellt die IDE acht Dateien, genannt „mdiapp.cpp“, „main.cpp“, „main.h“, „main.dfm“, „ChildWin.cpp“, „ChildWin.h“, „ChildWin.dfm“ und „mdiapp.bpr“. „mdiapp.bpr“ ist die Borland-Projektdatei und hat den gleichen Zweck wie „Projekt1.bpr“ für eine SDI-Anwendung. „main.cpp“, „main.h“ und „main.dfm“ beschreiben die Methoden und Objekte des Hauptfensters. „ChildWin.cpp“, „ChildWin.h“, „ChildWin.dfm“ beschreiben die Methoden und Objekte eines Child-Fensters. In Abbildung 5.17 wird die hergestellte Anwendung gezeigt, nachdem der Benutzer viermal die Taste „Neu“ angeklickt hat.

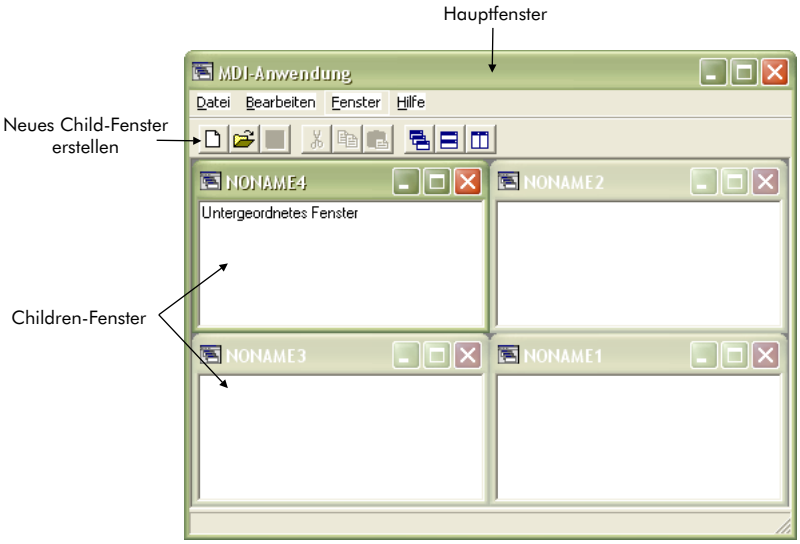


Abbildung 5.17: MDI-Anwendung, die aus einem Hauptfenster und einer beliebigen Anzahl von Children-Fenstern besteht.

Das Child-Fenster und das Hauptfenster sind Instanzen einer Klasse, die von der Klasse „TForm“ abgeleitet sind. In der Klasse „TForm“ gibt es die Eigenschaft „FormStyle“, die die Art des Fensters (Haupt-, Kind-, normal) beschreibt. In einer MDI-Anwendung ist die Eigenschaft „FormStyle“ für das Hauptfenster gleich „fsMDIForm“. Für das Child-Fenster ist sie gleich „fsMDIChild“.

Weitere Informationen über MDI-Anwendungen unter Borland C++ sind in der Dokumentation der IDE enthalten.

5.6.2 Allgemeine Funktionalitäten

Alle Zell-Chip-Messsoftwares haben gemeinsame Funktionalitäten, die in diesem Abschnitt zuerst präsentiert werden, und deren Implementierung erklärt wird. Die wichtigsten Funktionalitäten der Messsoftware wurden in Abbildung 5.1 aufgezeigt. Die Implementierung gemeinsamer Funktionalitäten der Software des Glas-Sensorchip-Messplatzes und des Einkanal-Messgeräts wird in den nächsten Abschnitten erklärt. Zuerst werden wichtige Messkonfigurationen vorgestellt. Danach werden die Kommunikation durch die serielle Schnittstelle und mit NI-DAQ Karten erläutert. Danach werden einige Eigenschaften der Speicherung der Messdaten erklärt. Zum Schluss wird die Implementierung einer C++ Komponente für die Darstellung der Messkurven beschrieben.

5.6.2.1 Verwaltung der Messkonfigurationen und Messinformationen

Die Zell-Chip-Programme müssen mehrere Arten von Versuchsinformationen verwalten. Erstens werden für die Kommunikation mit den Messgeräten mehrere Konfigurationen benötigt und für die Messung werden Informationen über die Sensorchips gebraucht. Zweitens muss das biologische Versuchsprotokoll für eine sichere Dokumentation der Versuche und für die Konfiguration des Fluidik-Systems verwaltet werden. In diesem Abschnitt werden diese Informationen und die Verwaltung der Messkonfigurationen beschrieben.

- Messgeräte-Konfigurationen und Sensorchip-Informationen
  Messgeräte werden mit mehreren Parametern konfiguriert. Ein wichtiger Konfigurationsparameter kann zum Beispiel die Konfiguration der Verbindung zwischen Computer und Messgerät sein, wie die Nummer einer seriellen Schnittstelle. Ein anderer Konfigurationsparameter für die Messung ist das Zeitintervall zwischen zwei Datenerfassungen. Andere wichtige Konfigurationsparameter sind die Bezeichnungen der verwendeten Sensorchips und die Kalibrierungswerte der Sensoren für jeden Chip.

- Allgemeine Informationen
  Für die Dokumentation des Versuchs werden mehrere allgemeine Informationen gebraucht, wie das Datum des Versuchs oder der Benutzername.

- Biologisches Versuchsprotokoll
  Für die Dokumentation des Versuchs und die Kontrolle des Fluidik-Systems wird ein biologisches Versuchsprotokoll benötigt. In diesem Protokoll wird angegeben, wann das Medium gewechselt wird, wie lang, und mit welcher Flussrate. Die Protokolle müssen zusätzlich für jeden Chip die folgenden Informationen enthalten:

  – Zelllinie (String)
  – Zellzahl (Integer)
  – Inkubationszeit (Integer)
  – Zellkulturmedium
  – Wirkstoffe im Kulturmedium zu- oder abgeführt (String), und Zeit der Zu/Abführung (Integer)

  Das Format des Felds „Zellkulturmedium“ ist komplexer als das Format der anderen Felder. Es existieren tatsächlich verschiedene Kulturmedien, in denen mehrere Stoffe addiert werden können (z. B. FCS oder Gentamycin). Im Prinzip sollen alle Komponenten des Mediums und ihrer Konzentration angegeben werden.

Nachdem die verschiedenen Konfigurationen aufgelistet wurden, wurde ein UML-Klassendiagram realisiert, das in Abbildung 5.18 dargestellt wird. Die Klasse „Messkonfiguration“ enthält Hardware-Parameter für die Messgeräte und das Fluidik-System, sowie verschiedene Zeitstempelwerte. Die Klasse „Messkonfiguration für 1 Chip“ enthält die Eigenschaften jedes Chips (Kalibrierungswerte der Sensoren, biologische Informationen) sowie eine Liste von 0 bis n Wirkstoffen, die zu der Zellkultur des Chips zugegeben werden.

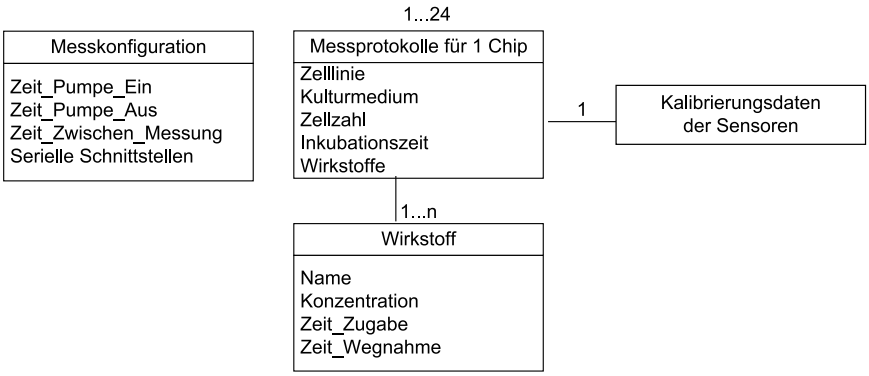


Abbildung 5.18: UML-Klassendiagramm für Messkonfiguration und Messprotokoll

5.6.2.2 Kommunikation durch die serielle Schnittstelle

Für die Kommunikation durch die serielle Schnittstelle wurden Funktionen der „Win32 API“ benutzt, die auf den Internetseiten von Microsoft [80] beschrieben sind. In diesem Abschnitt wird die Verwendung dieser Kommunikationsfunktionen erklärt.

Zuerst muss die Kommunikation durch den COM-Port konfiguriert werden. Die Struktur „DCB“ (Device Control Block), die alle Konfigurationsparameter der seriellen Schnittstelle (Baudrate, Zahl der Stopbits, Paritätsbit...usw.) enthält, wird dafür benutzt. Eine Funktion („initDcb“) wurde geschrieben, um die Werte der Konfigurationsparameter zu setzen. Diese Funktion ist in den folgenden Zeilen gegeben. Sie verwendet als Parameter die wichtigsten Konfigurationsparameter der Schnittstelle.

```
#include <stdio.h>
#include <stdlib.h>
#include <wtypes.h>

static DCB initDcb (DCB dcb,int baudrate,int stopbits, int dtr_control) {
         dcb.DCBlength         = sizeof(dcb);
         dcb.BaudRate          = baudrate;
         dcb.fParity           = FALSE;
         dcb.fOutxCtsFlow      = FALSE;
         dcb.fOutxDsrFlow      = FALSE;
         dcb.fDtrControl       = dtr_control;
         dcb.fDsrSensitivity   = FALSE;
         dcb.fTXContinueOnXoff = TRUE;
         dcb.fOutX             = FALSE;
         dcb.fInX              = FALSE;
         dcb.fNull             = FALSE;
         dcb.fRtsControl       = RTS_CONTROL_DISABLE;
         dcb.fAbortOnError     = FALSE;
         dcb.ByteSize          = 8;
         dcb.Parity            = NOPARITY;
```

```
        dcb.StopBits          = stopbits;
        return dcb;
}
```

Die Funktion „rs232_init“ wurde geschrieben, um einen COM-Port zu öffnen. Sie verwendet als Parameter die wichtigsten Konfigurationsparameter der Schnittstelle. Sie ruft die Funktion „CreateFile“ der Win32-API auf, um die Schnittstelle zu öffnen. Danach ruft sie die Funktion „initDcb“ und die Win32-API-Funktionen „GetCommState“ und „SetCommState“ auf, um die Konfigurationsparameter zu setzen. Zusätzlich definiert sie „Timeouts“. Zum Schluss gibt sie die COM-Port-Handle-Nummer zurück. Diese Funktion ist in den folgenden Zeilen angegeben.

```
COMMTIMEOUTS ComTimeouts;
HANDLE rs232_init(char *portName,int baudrate,int stopbits, int dtr_control){
//portName="com1" or "com2"...,
//baudrate=CBR_2400 or CBR_9600 ...
//stopbits= ONESTOPBIT or TWOSTOPBITS
//dtr_control= DTR_CONTROL_ENABLE or DTR_CONTROL_DISABLE
        HANDLE hCom;
        DCB dcb;
        hCom=CreateFile(portName,GENERIC_READ | GENERIC_WRITE,0,
                        NULL,OPEN_EXISTING,0,NULL);
        GetCommState(hCom,&dcb);
        dcb=initDcb(dcb,baudrate,stopbits,dtr_control);
        SetCommState(hCom,&dcb);

        ComTimeouts.ReadIntervalTimeout=MAXDWORD; //in milliseconds
        ComTimeouts.ReadTotalTimeoutMultiplier=MAXDWORD;
        ComTimeouts.ReadTotalTimeoutConstant=MAXDWORD;
        ComTimeouts.WriteTotalTimeoutMultiplier=MAXDWORD;
        ComTimeouts.WriteTotalTimeoutConstant=50;
        if (SetCommTimeouts(hCom,&ComTimeouts)) printf ("time out ok\n");
        else printf("Time out not ok\n");

        return hCom;
}
```

Wenn ein Befehl durch die serielle Schnittstelle geschickt wird, wird die Funktion „WriteFile“ benutzt. Als Parameter nimmt diese Funktion die Handle-Nummer des COM-Ports, die Charakterfolge, die geschickt werden soll, die Zahl der Charaktere, die geschickt werden sollen, und die Adresse einer (unsigned long) Variablen. Ein Beispiel der Verwendung dieser Funktion ist in den folgenden Zeilen angegeben.

```
ULONG nBytesWritten=0;
WriteFile(hCom,"FREQ3\r",6,&nBytesWritten,NULL);
```

Um empfangene Daten zu lesen, werden die Funktionen „ClearCommError“ und „ReadFile“ benutzt. Zuerst wird „ClearCommError“ aufgerufen, um den Zustand des COM-Ports zu ermitteln. Als Parameter nimmt „ClearCommError“ die Handle Nummer des COM-Ports, die Adresse einer (unsigned long) Variablen und die Adresse einer (COMSTAT) Struktur, die den aktuellen Zustand des COM-Ports beschreibt. Mit dem Feld „cbInQue“ dieser Struktur kann man erfahren, wieviel Charaktere im Empfangbuffer gespeichert sind. Wenn ein oder mehrere Charaktere empfangen wurden, wird die Funktion „ReadFile“ aufgerufen, um die Charakterfolge zu lesen. Als Parameter nimmt „ReadFile“

die Handle-Nummer des COM-Ports, eine Arbeitsspeicheradresse für die Kopie der Charakterfolge, die Anzahl der empfangenen Charaktere, und die Adresse einer (unsigned long) Variablen. Ein Beispiel ist in den folgenden Zeilen angegeben.

```
ULONG nBytesRead=0;
char buffer[100];
ULONG ret;
COMSTAT stat;

ClearCommError(hCom, &ret, &stat);
if (stat.cbInQue!=0) ReadFile(hCom,buffer,stat.cbInQue,&nBytesRead,NULL);
```

5.6.2.3 Kommunikation mit NI-DAQ AD-Wandlerkarten

Die Funktionen der Bibliothek „nidaq32.lib“ wurden für die Kommunikation mit verschiedenen AD-Wandlerkarten benutzt (PCI 6013 und PCMCIA 6035). Diese AD-Wandlerkarten haben mehrere analoge Eingänge und digitale Ein/Ausgänge. Die „nidaq32“-Bibliothek integriert zahlreiche Funktionen, um die Digitalisierung der analogen Spannungen oder digitale Ausgänge zu steuern. Mit der Funktion „*AI_VRead(int Device, int Chan, int Gain, float *Volt)*“ zum Beispiel wird die Spannung des analogen Eingangs „*Chan*“ in die Variable „*Volt*“ geschrieben. Mehr Informationen finden sich in der Dokumentation der Bibliothek.

5.6.2.4 Speicherung der Messdaten und Erfassung der Messzeit

Die sichere Speicherung der Messdaten ist eine sehr wichtige Aufgabe der Messsoftware. In den entwickelten Messprogramme wurde eine automatische Speicherung der Daten implementiert. Jede Messung ist nummeriert und automatisch in einem Unterverzeichnis des Programmverzeichnisses gespeichert. Die Erstellung des Verzeichnisnamens wird im ersten Teil dieses Abschnittes erklärt. Die Speicherung der empfangenen Messdaten in Textdateien, die auf die Festplatte transferiert werden, wird im zweiten Teil dieses Abschnitts erklärt.

- Automatische Erstellung eines Messverzeichnisses
  Alle Messprogramme dieser Arbeit speichern automatisch alle Messdaten in einem Verzeichnis, genannt „$Messungen/jjjj\_mm\_tt\_n$“, wo $jjjj$ das Jahr mit vier Ziffern, mm den Monat, tt den Tag und n die Nummer des Messverzeichnisses darstellen. Die Strukturen „*struct tm*“ und „*time_t*“ und die Funktionen „$time(time\_t *)$“ und „$localtime(time\_t *)$“ der C-Bibliothek „time“ wurden benutzt, um das aktuelle Datum zu erhalten. Die Verwendung dieser Funktionen, um die Werte der Integer-Variablen „*day*“, „*month*“ und „*year*“ zu erhalten, wird in den folgenden Zeilen gezeigt.

  ```
  #include<time.h>
  int day,month,year;  //0<=day<=31, 1<=month<=12, 0<=year<=99
  time_t t0;
  struct tm *tp;

  time(&t0);
  tp=localtime(&t0);
  day=tp->tm_mday;
  month=tp->tm_mon+1;
  year=tp->tm_year-100;
  ```

  Danach wird der Verzeichnisname „$year\_Month\_Day\_1$“ erstellt (z. B. 2003_02_11_1), wie in den folgenden Zeilen gezeigt.

```
char directory[400];
int file_number=1;
//48: ASCII code of 0
directory[0]='2'; directory[1]='0'; directory[2]=year/10+48;
directory[3]=year-10*(year/10)+48; directory[4]='_'; directory[5]=month/10+48;
directory[6]=month-10*(month/10)+48; directory[7]='_'; directory[8]=day/10+48;
directory[9]=48+day-10*(day/10); directory[10]='_';
directory[11]=file_number+48; directory[12]='\0';
```

Danach wird überprüft, ob der Verzeichnisname „$year\_Month\_Day\_1$“ schon existiert. Wenn ja, wird die Variable „$file\_number$“ inkrementiert, bis ein Verzeichnisname kommt, der noch nicht erstellt wurde. Danach wird das Verzeichnis erstellt ($mkdir(directory)$) und man ändert das laufende Verzeichnis ($chdir(directory)$). Der Algorithmus ist in den folgenden Zeilen gegeben.

```
//I verify if the directories 2003_02_11_1,2003_02_11_2...2003_02_11_10...
//already exist until I find one that doesn't exist:
int ciphers=1;      //number of ciphers of the variable "file_number"
while ((chdir(directory)==0)&&(ciphers<=9)){
        int c=10;int d=1;
        int rest=0;
        int i,k;
        for (i=1;i<ciphers;i++) c*=10;
        chdir("../");
        file_number++;
        if ((file_number-c)==0) ciphers+=1;
        for (i=ciphers-1;i>=0;i--){
                d=1;
                for (k=0;k<i;k++) {d*=10;}
                directory[11+ciphers-i-1]=file_number/d+48-rest*10;
                rest=file_number/d;
        }
        directory[ciphers+11]='\0';
}
mkdir(directory);
chdir(directory);
```

- Speicherung der Messdaten in Textdateien (ASCII-Dateien)
  Für die Speicherung der Messdaten wurden Funktionen der C-Bibliothek „stdio“ benutzt. Um eine Textdatei zu öffnen, wird die Funktion „$fopen()$“ benutzt. Das folgende Beispiel zeigt, wie die Datei „Messdaten.txt“ erstellt wird.

  ```
  FILE *file_Datas_txt;
  file_Datas_txt=fopen("Messdaten.txt","w");
  ```

  Danach werden die Messdaten mit der C-Funktion „$fprintf()$“ in die Datei geschrieben. Im folgenden Beispiel werden die Inhalte der Variablen „$double\ theTime$“ und „$double\ y\_value$“ in die Textdatei „Messdaten.txt“ geschrieben.

  ```
  fprintf(file_Datas_txt,"%g",theTime);
  fprintf(file_Datas_txt,"\t%g\n",y_value);
  ```

Um den Datentransfer auf die Festplatte zu eröffnen, wird die Funktion „$fflush(FILE *)$“ aufgerufen. Nachdem die Messdaten geschrieben wurden, wird die Messdatei mit der Funktion „*fclose*“ geschlossen.

```
fclose(file_Datas_txt);
```

- Erfassung der Messzeit
  Die Erfassung der Messzeit seit dem Beginn einer Messung wird durch die Funktion „*clock_t clock()*“ der C-Bibliothek „time“ durchgeführt. Diese Funktion berechnet die seit dem Programmaufruf verbrauchte „CPU-Zeit“. Mehrere Variablen werden für die Erhaltung der Messzeit benutzt, sie müssen am Beginn einer Messung initialisiert werden. Diese Variablen mit ihren Initialisierungswerten sind in den folgenden Zeilen angegeben:

```
float clocksPerMin=60*CLOCKS_PER_SEC;
double clock0;              //clock0 is initialised when a measurement begins
int firstTime=1;            //true when it's the first time that the getTime()
                            //function is called
double timeSpent=0;         //time spent in minutes
```

  In den Messprogrammen wird die Funktion „*getTime()*“ aufgerufen, jedesmal wenn eine Messung durchgeführt wird. Nach dem Aufruf dieser Funktion hat die Variable „*timeSpent*“ den Wert der abgelaufenen Zeit seit dem Beginn der Messung in Minuten. Die Funktion „*getTime()*“ ist in den folgenden Zeilen angegeben:

```
void getTime(){
  if (!firstTime){
    timeSpent=(clock()-clock0)/clocksPerMin;
  }else {
    clock0=clock();
    timeSpent=0;
    firstTime=0;
  }
}
```

5.6.2.5 Darstellung von annotierten Messkurven

Ein „Borland C++“-Package wurde programmiert, um Messkurven darzustellen und um über farbige Knöpfe zu verfügen. Das Package wird von der Glas-Sensorchip-Messsoftware und der Einkanal-Messgerät-Software benutzt. Die Funktionalitäten und die Implementierung des Package werden in diesem Abschnitt erklärt.

- Funktionalitäten
  Das Package integriert zwei Komponenten: einen farbigen Knopf und eine Komponente für die Darstellung von Messkurven. Diese letztere Komponente kann entweder die letzten tausend Messpunkte in Realzeit darstellen oder den ganzen Verlauf der Messung darstellen. Die Zeiten des Mediumwechsels und der Zugabe von Wirkstoffen werden durch vertikale Linien auf der Messkurve gezeichnet. Die Wirkstoffnamen werden über die vertikalen Linien geschrieben. In den Abbildungen 5.13 und 5.14 sind Software-Bedienoberflächen mit dieser Komponente (weiße Quadrate mit Messkurven) gezeigt.

- Installation und Verwendung der Komponente in „C++ Builder“
  Das Package besteht aus mehreren Dateien, die in der folgenden Tabelle angegeben sind.

Komponente	Dateien
	GraphButtonPakage.bpk, GraphButtonPakage.cpp
Farbiger Knopf	SpeedButtonColored.dcr SpeedButtonColored.h, SpeedButtonColored.cpp
Graphik	GraphPanel.dcr Curve.h, Graph.h, Graph.cpp, GraphPanel.h, GraphPanel.cpp

Die „.dcr“ Dateien sind so genannten Komponenten-Resource-Dateien. Sie wurden mit der Borland-Software „Bildeditor“ erstellt. Sie werden für die Darstellung von Ikons für jede Komponente auf der „C++ Builder“-VCL-Komponenten-Menüleiste benutzt. Die „GraphButtonPakage.bpk“-Datei (Borland Package) ist die Makefile-Datei. Für die Installation des Packages muss diese Datei geladen werden, danach muss das Projekt kompiliert werden und dann kann das Package installiert werden.

Die Verwendung der Komponenten erfolgt wie die Verwendung anderer VCL-Komponenten. Die Komponente wird mit der Maus in der VCL-Menüleiste selektiert und auf das Fenster der entwickelten Anwendung transportiert. Im Quellcode müssen dann die folgenden Befehle geschrieben werden:

```
#include "Curve.h"
#include "Graph.h"
#include "GraphPanel.h"
#include "SpeedButtonColored.h"
```

Nachdem die Komponente „GraphPanel“ in das Anwendungsfenster eingefügt wurde, wird eine Instanz der Klasse „TGraphPanel“ automatisch deklariert: „*TGraphPanel *GraphPanel1*“, die benutzt werden kann, um Messkurven darzustellen. Die Komponente kann eine beliebige Anzahl von „Curve“-Objekten darstellen. Ein „Curve“-Objekt hat mehrere Eigenschaften, wie Farbe (TColor color), Name (char *name), Visibilität (bool draw), X-Werte (double *tx) oder Y-Werte (double *ty). Ein Beispiel der Erstellung einer Tabelle von zwei „Curve“-Objekten ist in den folgenden Zeilen gegeben. In diesem Beispiel werden die X- und Y-Werte der Kurven initialisiert.

```
Curve theCurve[2];
theCurve[0].color=clRed;
theCurve[0].draw=1;
strcpy(theCurve[0].name,"Curve 1");
theCurve[1].color=clBlue;
theCurve[1].draw=1;
strcpy(theCurve[1].name,"Curve 2");
int i;
for(i=0;i<GRAPH_SIZE;i++){
        theCurve[0].tx[i]=i;
        theCurve[0].ty[i]=cos((float)i/10);
        theCurve[1].tx[i]=i;
        theCurve[1].ty[i]=cos((float)i/10)+sqrt((float)i);
}
```

Nachdem die Kurven erstellt wurden, muss die Methode „*createGraphPanel(Curve *theCurve,int nbOfCurves,char *XAxisName,char *YAxisName)*“ aufgerufen werden, um die Graphik auszugeben.

```
GraphPanel1->createGraphPanel(theCurve,2,"Time (min)","Values (Volt)");
```

Die „Curve“-Objekte enthalten eine begrenzte Anzahl von Messpunkten (GRAPH_SIZE). Um den ganzen Verlauf einer gemessenen Kurve sehen zu können, wurde die Funktion „*ReadPlotFile_readAndPlot(FILE *fx, ListOfGraphs *list, double x_min, double y_min, bool allFile, bool forWMF)* “ implementiert. Diese Funktion nimmt als Parameter die binäre Datei „*fx*“, in der die X-Koordinaten (double) nacheinander gespeichert sind. Sie nimmt als Parameter eine Liste von „GraphPanel“-Objekten, den minimalen und maximalen X-Wert und den Boolean-Parameter „allFile“, der zeigt, ob die ganzen Messdateien gelesen werden müssen. Diese Funktion liest die gespeicherten X-Koordinate der Messpunkten in der binären Datei „*fx*“ und die Y-Koordinate der Messkurven in den binären Dateien „$theCurve[i].fy\_read$“. Um die Grenzen der Y-Achsen zu definieren werden Methoden der Klasse „GraphPanel“ benutzt, wie es in den folgenden Zeilen gezeigt wird.

```
GraphPanel1->setReadPlotValueLimits(-30,20);  //-30<y<20
GraphPanel1->setReadPlotBestFitValues();      //ymin<y<ymax
```

Ein Beispiel der Verwendung der Funktion „*ReadPlotFile_readAndPlot*“ ist in den folgenden Zeilen gegeben.

```
struct ListOfGraphs *list=0;
FILE *fx; FILE *fy; FILE *fz;
list=ListOfGraphs_add(GraphPanel1,list);
fx=fopen("fx.b","rb");
fy=fopen("fy.b","rb");
fz=fopen("fz.b","rb");
theCurve[0].fy_read=fy;
theCurve[1].fy_read=fz;
ReadPlotFile_readAndPlot(fx,list,0,100,1,0);
ListOfGraphs_free(list);list=0;
fclose(fx);fclose(fy);fclose(fz);
```

- Implementierung
  Mehrere C++ Klassen wurden erstellt, um die Komponente zu implementieren. Die Hauptklasse ist „TGraphPanel“, die von der VCL-Klasse „TPanel“ abgeleitet ist. Diese Klasse benutzt eine Instanz der Klasse „Graph“. Die Klasse „Graph“ benutzt eine bis n Instanzen der Klasse „Curve“. In der Abbildung 5.19 werden diese Beziehungen dargestellt.

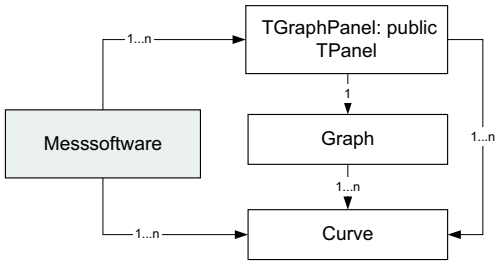


Abbildung 5.19: C++ Klassen für die Implementierung einer Komponente für die Darstellung von Messkurven und ihre Beziehung mit der Messsoftware.

Diese Methoden und Eigenschaften der erstellten C++ Klassen werden in den folgenden Punkten beschrieben.

– „Curve“-Klasse
Die „Curve“-Klasse enthält verschiedene Eigenschaften einer Kurve, wie Farbe, Punktanzahl, X-Werte, Y-Werte, Namen. Die Deklaration dieser Klasse ist in den folgenden Zeilen gegeben.

```
#define GRAPH_SIZE 1000

class Curve {
 public:
  char name[100];
  TColor color;
  FILE *fx;      //file in which are saved the X coordinates
  FILE *fy;      //file in which are saved the Y coordinates
  FILE *fx_read; //file used to read the X coordinates
  FILE *fy_read; //file used to read the Y coordinates
  double tx[GRAPH_SIZE];   //table of the GRAPH_SIZE last X values
  double ty[GRAPH_SIZE];   //table of the GRAPH_SIZE last Y values
  int begin;     //from which case the tables are drawn
  bool draw;     //to know if the curve has to be drawn
  int option;    //serial port number, or chanel number for the ADC card
  int nbPoints;  //number of points measured
} ;
```

– „Graph“-Klasse
Die „Graph“-Klasse integriert die Funktionen, um Tabellen von X- und Y-Werten darzustellen, Legenden zu schreiben und vertikale Linien zu zeichnen. Die Zeichnungen werden in einem Objekt „*TCanvas*“ realisiert, das das „*TGraphPanel*“-Objekt enthält. Die Deklarationen einiger Methoden und Objekte dieser Klasse sind in den folgenden Zeilen gegeben.

```
class Graph {
 private:...
 public:
  Graph(TImage *theImage, char *legend_x,        //constructor
        char *legend_y,int nbLines,
        Curve *theCurves);
  Graphics::TBitmap *backDrawing;                //canvas shown
  void drawAxis(TCanvas *ca,...);                //draws the grid
  void drawLegend(TCanvas *ca);                  //draws the legend
  void plotTables(TCanvas *ca,double *tx,        //draws tables
                  double *ty,...,
                  int lineNumber);
  void drawVerticalLine(TCanvas *ca,TColor col,  //draws a vertical line
                        ...);
  void drawText(TCanvas *ca,TColor col,...);     //writes a text
  ~Graph();                                      //destructor
   ...
};
```

Die Funktion „*drawXAxis(...)*“ wird aufgerufen, um ein Gitter zu zeichnen. Die Implementierung dieser Funktion wird hier beschrieben. Verschiedene Variablen werden gebraucht, um das Gitter zu zeichnen: der minimale und maximale X-Wert der Kurve (*Table_tmin, Table_tmax*) und die Grenzen des sichtbaren Fensters (*Time_min, Time_max*), wie in der

Abbildung 5.20 gezeigt. Die Funktion „*calcAxis(...)*“ wurde implementiert, um drei Zahlen (n, puis und D) für die Zeichnung des Gitters zu berechnen. Die Bedeutung dieser Zahlen ist folgende: $E[]$ sei die Funktion, die den ganzen Teil einer reellen Zahl wiedergibt, v eine reelle Zahl mit $1 \leq v < 10$ und n eine ganze Zahl:

$$val = (Time\_max - Time\_min)/5 = v \cdot 10^n \tag{5.4}$$

$$puis = 10^n \tag{5.5}$$

$$D = E[v] \tag{5.6}$$

Die Distanz zwischen zwei Gitterlinien wird gleich $D \cdot puis$ definiert. Danach wird die Position der ersten Gitterlinie berechnet. Sie wird gleich: $puis \cdot E[Table\_tmin/puis]$ definiert.

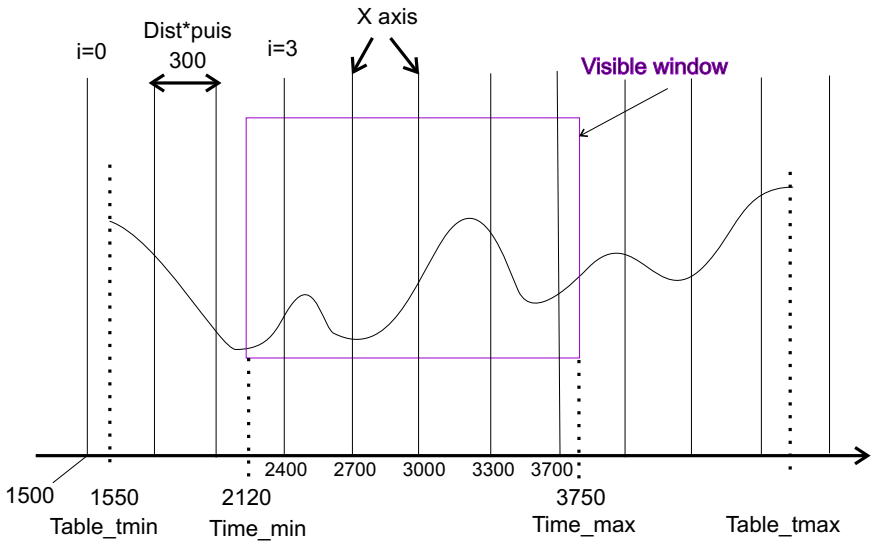


Abbildung 5.20: Zeichnung des Gitters.

Die Funktion „*calcAxis(...)*“ ist in den folgenden Zeilen angegeben. Diese Funktion prüft, ob die Zahl „*double val*“ kleiner als 1 ist. Wenn ja, wird sie n Mal mal 10 multipliziert, bis sie grösser als 1 ist. Wenn die Zahl grösser als 10 ist, wird sie n Mal mal 0,1 multipliziert, bis sie kleiner als 10 ist.

```
void calcAxis(double val,int *n,double *puis,int *D){
        double v,mult;
        int direction,i;
        *n=0;
        *puis=1;
        v=val;direction=-1;mult=0.1;
        if(v<1) {direction=1;mult=10;}
        while((v<1)||(v>=10)) {
                v=v*mult;
                *n+=-direction;
                *puis/=mult;
        }
        *D=(int)v;
}
```

– „TGraphPanel“-Klasse
Die „TGraphPanel“-Klasse ist von der „TPanel“-Klasse abgeleitet. Sie integriert ein

„TImage“-Objekt, in dessen Canvas (Graphik-Ausgabe Feld) die Graphik gezeichnet wird. Sie enthält auch zwei „TLabel“- und „TEdit“-Objekte, die dazu dienen, die Position der Maus in den Messkurven-Koordinaten zu schreiben. Einige wichtige Methoden der Klasse „TGraphPanel“ wurden bereits in diesem Abschnitt präsentiert. Eine andere wichtige Methode dieser Klasse ist „*blitBackDrawing()*“. Die Zeichnung der Messkurven werden in einer versteckten Canvas (*theGraph* → *backDrawing*) durchgeführt. Danach wird die Funktion „*blitBackDrawing();*“ aufgerufen, um die versteckte Canvas in die Canvas des „TImage“-Objektes zu kopieren. Dieses Verfahren vermeidet, dass die Graphiken blinken, wenn sie neu aufgezeigt werden.

Erläuterungen der wichtigsten Methoden und Objekten der Klasse "TGraphPanel" sind in den folgenden Zeilen angegeben.

```
class PACKAGE TGraphPanel : public TPanel
{protected:
  Graph *theGraph;
  void blitBackDrawing();
  TMetafileCanvas *WMFCanvas;
  TMetafile *theMetafile;
 public:
  __fastcall TGraphPanel(TComponent* Owner);
  char nameOfGraph[200];
  TImage *theImage;              //TImage containing the graphic
  TLabel *label_x;               //TLabel included in the panel
  TEdit *edit_x;
  TLabel *label_y;
  TEdit *edit_y;
  void createGraphPanel(Curve *theCurves,    //initialization
                        int curvesNb,
                        char *leg_x,
                        char *leg_y);
  void redrawTables();                       //redraws the tables
  void setTablesTimeLimits(double t0,double t1);
  void setTablesValueLimits(double v0,double v1);
  void setTablesBestFit_y();
  void setBackgroundColor(TColor c);
  void setGridColor(TColor c);
  void createWMF(char *filename);  //creates an EMF image of the graphic
  void createBMP(char *filename);  //creates a BMP image of the graphic
  void drawVerticalLines(char *filename, TColor col);
  int  ReadPlotFile_readAndPlot(FILE *fx,...);   //plots measurement files
  void setReadPlotValueLimits(double v1,double v2);
  void setReadPlotBestFitValues();
  void free_TGraphPanel();        //destruction of all objects
}
```

Die Funktionalitäten und die Implementierung dieser Komponente wurden verbessert, wie im Abschnitt „Mess-/Analyse-Software“ dieses Kapitels gezeigt wird.

5.6.3 Messsoftware des Einkanal-Messgeräts

- Programmstruktur
  Der Programmcode wurde in mehrere Dateien unterteilt. Die Dateinamen und ihre allgemeinen Funktionalitäten sind in der folgenden Tabelle zusammengefasst.

Funktionalität	Dateien
Kommunikationscode mit AD-Wandlerkarte	ADConverter.cpp, ADConverter.h
Erstellung des Messverzeichnisses und Speicherung der Daten	File.cpp, File.h
Projekt-Strukturen	ChipsProject_struct.h
Projekt-Dateien	Project1.cpp,Project1.bpr
Formulare (Fensterdialog) Jeweils „.cpp“-, „.h“- und „.dfm“-Datei	Unit1 (Hauptfenster) ADCardDial, Calib_O2Dial, Calib_pHDial, Comment, DialogStopConfirm, ExitConfirm, graphSelect, OpenFromTo, Help, SeeComments, SensorsChoice, TimeDelay, Wait, WaitExport, XYaxisDialog, UsingCalibrationsValuesDial

  Für die Darstellung der Messkurven benutzt das Programm zusätzlich die Komponente „*TGraphPanel*“, die im vorherigen Abschnitt beschrieben wurde.

- Algorithmen
  Die Speicherung der Messdaten und die Erstellung des Messverzeichnisses wurden bereits im vorherigen Abschnitt beschrieben. Für die Darstellung der Messdaten wurden 5 „*TGraphPanel*“-Objekte (*GraphPanel1...GraphPanel5*) und eine Tabelle von 8 „*Curve*“-Objekten (*Curve curveTable*[*8*]) erstellt. Um die Messspannungen auszulesen und einen digitalen Ausgang (LED) zu steuern, kommuniziert die Software mit der AD-Wandlerkarte „NiDAQ 6036“, wie in Abbildung 5.21 aufgezeigt.

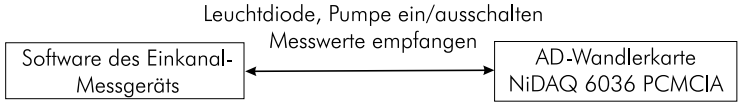


Abbildung 5.21: Software des Einkanal-Messgeräts: Kommunikation mit einer AD-Wandlerkarte

  Für die periodische Messung der Sensorspannungen wird das „*TTimer Timer1*“-Objekt benutzt. Wenn eine Messung gestartet wird, wird die Eigenschaft „*Timer1->Enabled*“ auf 1 (true) gesetzt. Die Methode „*Timer1->OnTimer*“ wird dann periodisch (alle *Timer1->Interval* mS.) aufgerufen.

5.6.4 Messsoftware des Glas-Sensorchip-Messplatzes

5.6.4.1 Kommunikation mit Geräten und Kommunikationsprotokoll

Wie bereits in Kapitel 3 (Material und Methoden) erklärt wurde, kann die Software mit verschiedenen Messgeräten kommunizieren: pH/O_2/Temperatur-Messgerät; Fibox 2, LCR-Meter, Multiplexer und ISMATEC-Pumpe. Diese Geräte sind alle durch die serielle Schnittstelle mit dem Computer verbunden. In Abbildung 5.22 werden die Software-Kommunikationen graphisch dargestellt.

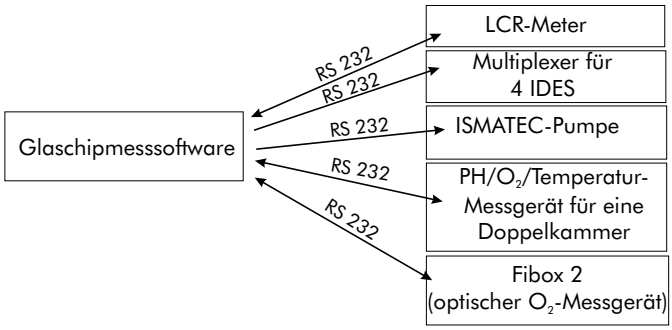


Abbildung 5.22: Software des Glas-Sensorchip-Messplatzes: Kommunikation mit 5 Geräten durch die RS232-Schnittstelle

Jedes Gerät hat sein eigenes Kommunikationsprotokoll, das in diesem Abschnitt beschrieben wird.

- pH/O_2/Temperatur-Messgerät
  Wie bereits in Kapitel 3 erklärt wurde, integriert jeder Glas-Sensorchip zwei pH-, zwei O_2-Sensoren und einen Temperatur-Sensor. Die aktuelle Software steuert die Messung dieser Sensoren für zwei Glas-Chips. Dafür schickt und empfängt sie eine bestimmte Charakterfolge, die in Tabelle 5.23 angegeben ist. Der erste gesendete Charakter ist die Nummer der Messplatine. Eine Messplatine beinhaltet die Messschaltungen für zwei Glas-Sensorchips. Der zweite geschickte Charakter (T, O oder P) zeigt, welche Sensorspannung gemessen werden soll. Nach dem Empfang einer Charakterfolge schickt das Messgerät den Wert der Sensor-Messspannung(en) zurück. Der Spannungswert ist auf 16 Bit kodiert. Der Wert 65536 (2^{16}) entspricht 5 Volt.

Messparameter	Messplatinen-Nummer	Software-Befehl	Messgerät-Antwort
Temperatur	1	1T	2×16 Bit
Sauerstoff	1	1O	4×16 Bit
pH	1	1P	4×16 Bit
Alle (T.,S.,pH)	1	1A	10×16 Bit
Temperatur	2	2T	2×16 Bit
Sauerstoff	2	2O	4×16 Bit
pH	2	2P	4×16 Bit

Abbildung 5.23: Befehle zur Steuerung und Antwort des pH/O_2/Temperatur-Messgeräts

- „Fibox 2“-Messgerät
  Das „Fibox 2“-Messgerät wird benutzt, um optisch die gelöste Sauerstoffkonzentration zu messen. Das Kommunikationsprotokoll des Geräts befindet sich in der Software-Dokumentation des Geräts. Die wichtigsten Eigenschaften des Kommunikationsprotokolls, die von der Glas-Sensorchip-Messsoftware benutzt werden, werden hier angegeben. Einige Befehle, die das Gerät versteht, sind in Tabelle 5.24 angegeben. Wenn ein Befehl geschickt wird, muss beachtet werden, dass die Charaktere nacheinander mit einem Abstand von mindestens 1 ms. zum Messgerät geschickt werden müssen.

  Das Gerät kann entweder dauernd Messdaten an den PC schicken („Continuous Mode“) oder nur Messdaten schicken, wenn der PC ihm den Befehl dazu gibt („Single Mode“). Im Programmcode der Glas-Sensorchip-Messsoftware wurde der „Single Mode“ benutzt.

  Wenn das Gerät den Befehl „Get Data“ bekommt, schickt es die Sensormesswerte. Die empfangenen Daten haben das folgende Format: AX..X;PX..X;TX..X;OX..X; Dabei stellt A die

Amplitude, P die Phase, T die Temperatur, O den Sauerstoffgehalt (%airsat), und X..X eine Gleitkomma-Zahl dar.

	Aktion	Software-Befehl
Konfiguration:	Continuous Mode	g0\r
	Single Mode	g1\r
	No temperature compensation	tx\r
	Get Data	d\r

Abbildung 5.24: Einige Befehle zur Steuerung des „Fibox 2“-Geräts

- LCR-Meter
  Wie in Kapitel 3 erklärt wurde, wurde ein LCR-Meter benutzt, um die Impedanz der IDES-Sensoren zu messen. Die benutzten Befehle für die Steuerung dieses Geräts werden hier beschrieben.

  Zuerst wird das Gerät konfiguriert, indem der folgende Befehlsatz geschickt wird:

```
WriteFile(hCom," *RST; PMOD4; VOLT0.1; RATE2; MMOD1; CIRC1; OUTF0; FREQ3\r",
          57,&nBytesWritten,NULL);
```

  Die Bedeutung der benutzten Konfigurationsbefehle sind in Tabelle 5.25 angegeben.

Konfiguration	Software-Befehl
Reset	*RST
Messspannung:100 mV	VOLT0.1
Messfrequenz:10 kHZ	FREQ3
400 Messperioden	RATE2
Äquivalente Schaltung: Rpar,Cpar	PMOD4

Abbildung 5.25: Benutzte Befehle zur Konfiguration des LCR-Meters

Danach wird eine Messung durch den folgenden Befehlsatz gestartet:

```
WriteFile(hCom,"STRT;*WAI\r",10,&nBytesWritten,NULL);
```

Die gemessenen Impedanz-Werte, die aus zwei Parametern bestehen (major parameter: Kapazität, minor parameter: Widerstand), werden durch den folgenden Befehlsatz zum Computer geschickt:

```
WriteFile(hCom,"XALL?\r",6,&nBytesWritten,NULL);
```

Die empfangene Charakterfolge hat das folgende Format:

X	X	X..X	X	X..X	X
Good(G)/Bad	Range(0,1,2,3)	Major Parameter	,	Minor Parameter	;
G	1	1.234e-8	,	580	;

Der erste Charakter zeigt, ob der Messwert korrekt ist (nicht „over range“). Der dritte Charakter bis zum „,“ ist der gemessene „Major Parameter“ (Kapazität). Danach kommt der „Minor Parameter“ (Widerstand), und zum Schluss der Charakter „;“. Als Beispiel kann die Charakterfolge „G11.234e-8,580;“ empfangen werden. Der Kapazitätswert ist dann gleich $1.234 \cdot 10^{-8}$ Farad, und der Widerstand gleich 580 Ohm.
Mehr Informationen über das Kommunikationsprotokoll des LCR-Meters finden sich im Handbuch des Geräts.

- Multiplexer
  Jeder Chip integriert zwei Impedanz-Sensoren. Wie in Kapitel 3 erwähnt, wird ein Multiplexer benutzt, um vier IDES-Sensoren zu messen. Der Multiplexer wird durch die folgenden Befehle gesteuert:

Chip-Nummer	IDES-Nummer	Software-Befehl
1	1	1\r
1	2	2\r
2	1	3\r
2	2	4\r

Abbildung 5.26: Befehle zur Steuerung des Multiplexers

- ISMATEC-Pumpe
  Das benutzte Kommunikationsprotokoll mit der ISMATEC-Pumpe ist hier angegeben. Es können mehrere peristaltische Pumpen durch eine einzige serielle Schnittstelle gesteuert werden. Deshalb braucht jede Pumpe eine Adresse. Die Adresse „1“ wird einer Pumpe zugewiesen, indem man ihr den Befehl „@1\r“ schickt:

```
WriteFile(hCom,"@1\r",3,&nbBytesWritten,NULL);
```

  Danach wird die Pumpe mit den folgenden Befehlen ein- und ausgeschaltet:

Adresse	Aktion	Software Befehl
1	Einschalten	1H\r
1	Ausschalten	1I\r

Abbildung 5.27: Befehle zur Steuerung der Pumpe

Mehr Informationen finden sich in der Dokumentation der Pumpe.

5.6.4.2 Programmstruktur und Algorithmen

- Programmstruktur
  Der Programmcode wurde in mehrere Dateien unterteilt. Die Dateinamen und ihre allgemeinen Funktionalitäten sind in der folgenden Tabelle zusammengefasst.

Funktionalität	Dateien
Kommunikationscode mit AD-Wandlerkarte	ADConverter.cpp, ADConverter.h
Kommunikationscode Jeweils „.cpp“ und „.h“-Datei	FIBOX, IDES_LCRCorrection, LCRMeter Micro, multiplexer-serial, PUMP, RS232
Erstellung des Messverzeichnisses und Speicherung der Daten	file.cpp, file.h
Projekt-Strukturen	ChipsProject_struct.h, Sensor.h
Projekt-Dateien	Project1.cpp,Project1.bpr
Formulare (Fensterdialog) Jeweils „.cpp“-, „.h“- und „.dfm“-Datei	Unit1 (Hauptfenster) AddSubstanceDial, AxisDial, ChipsCellDial, CommentDial, ResistivityDial, SerialPorts, StartDial, TimeDelays, Confirmation, WaitDial, Xaxis, Yaxis

Für die Darstellung der Daten benutzt das Programm zusätzlich die Komponente „*TGraphPanel*“, die in Kapitel 5.4.1 beschrieben wurde.

- Algorithmen
  Die Speicherung der Messdaten und Erstellung des Messverzeichnisses wurden bereits im Abschnitt 5.4.1 beschrieben.

  Für die Darstellung der Messdaten werden 20 „*TGraphPanel*“-Objekte erstellt (*GraphPanel1 ... GraphPanel20*). Diese Objekte sind paarweise in „*TPanel*“-Objekten gruppiert (*Panel1 ... Panel10*), die gezeigt oder nicht gezeigt werden, je nach Zustand der Knöpfe „*IDES1Button ... Lact2Button*“. Für die Darstellung der Messpunkte werden drei Tabellen von „*Curve*“-Objekten benutzt, eine für die „Fibox-2“-Messdaten, eine für die IDES-Messdaten und eine letzte für alle anderen Messdaten. Die Deklaration dieser Tabelle ist die folgende:

```
Curve FiboxSensorsTable[1*2];
Curve IDESSensorsTable[4*2];//1-Rpar1 1-Rpar2 1-Cpar1 1-Cpar2 2-Rpar1 ...aso
Curve ADCSensorsTable[12*2];
```

  Die erste Hälfte der Tabellen wird für den ersten Chip benutzt, die zweite für den zweiten Chip.

  Für die periodische Messung der Sensordaten werden mehrere „*TTimer*“-Objekte benutzt: „*Timer1*“ für die Messung mit dem O_2/Temperatur/pH-Messgerät, „*TimerPump*“ für die Steuerung der Pumpe, „*TimerFibox*“ für die Messung mit dem „Fibox 2“-Gerät, „*TimerLCRMeter*“ für die Messung der Impedanz. Wenn ein Messgerät benutzt wird, wird die Eigenschaft „*Enabled*“ des entsprechenden „Timer“ auf 1 (true) gesetzt. Ihre Methode „*OnTimer*“ wird dann periodisch aufgerufen.

5.6.5 Mess-/Analyse-Software

Nachdem die Funktionalitäten der Software beschrieben wurden, wird in diesem Abschnitt ihre Implementierung erklärt. Zuerst wird die Struktur des Programms durch sein UML-Klassendiagramm präsentiert. Danach wird die Darstellung der Messkurven erklärt. Zum Schluss wird das Format der Projektdatei erläutert.

5.6.5.1 UML-Klassendiagramm

UML (Universal Modelling Language) ist eine Methode, um objektorientierte Programmcodes graphisch zu modellieren. Die UML-Modellierung besteht hauptsächlich aus Klassendiagrammen und

„Use Case“-Diagrammen. Das UML-Klassendiagramm der Analyse-Software wird in Abbildung 5.28 gezeigt.

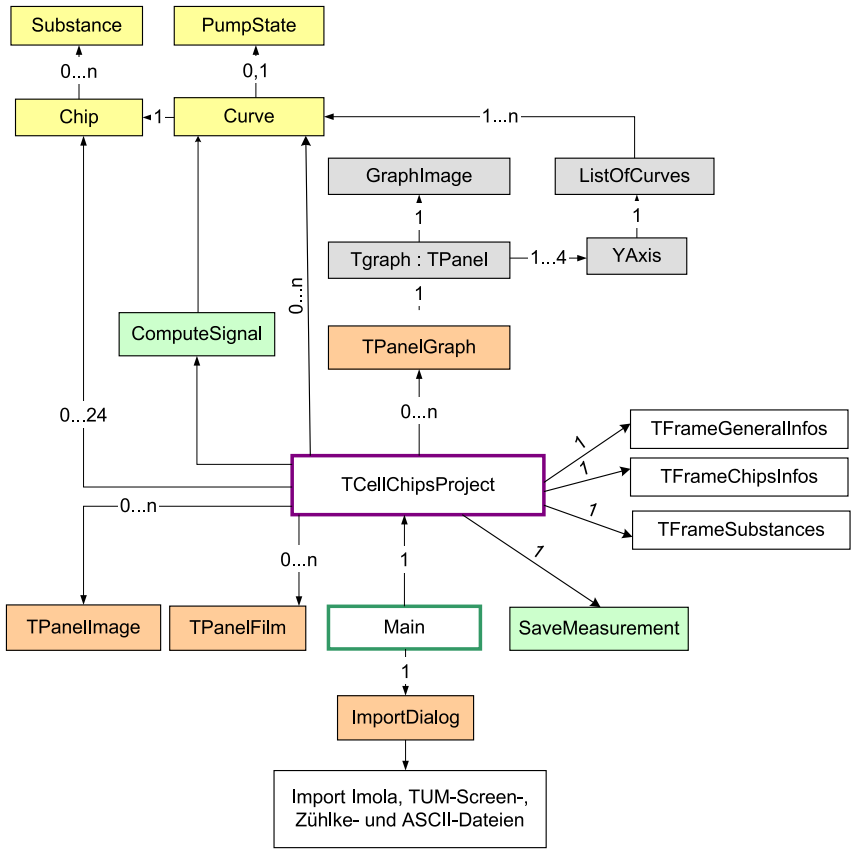


Abbildung 5.28: UML-Klassendiagramm

Die Funktionalitäten jeder Klasse werden in den folgenden Punkten beschrieben.

- „*Main*“-Klasse
  Die „*Main*“-Klasse ist das Hauptformular des Programms. Sie enthält die Deklarationen von allen Kontroll- und Anzeigeelementen des Hauptfensters (Menüleiste, ToolButtons, Frames), sowie die Funktionen, die aufgerufen werden, wenn Ereignisse auf diesen Elementen auftreten. In den folgenden Zeilen sind Deklarationen von einigen Methoden und Attributen der Klasse angegeben. Wenn der Benutzer zum Beispiel das Menü „*File ->Open*“ anklickt (Ereignis auf dem „*FileOpenItem*“-Objekt), wird die Funktion „*FileOpen*1*Execute*“ aufgerufen.

```
class TMainForm : public TForm
{
 __published:
        TMainMenu *MainMenu1;
        TMenuItem *File1;
        TMenuItem *FileOpenItem;
        ...
        void __fastcall FileOpen1Execute(TObject *Sender);
        ...
 public:
        TCellChipsProject theCellChipsProject;
        bool measure;                //true when a measurement is started
```

```
        bool opened;                //true when a project file is opened
}
```

- „*TCellChipProject*“-Klasse
  Die „*TCellChipProject*“-Klasse enthält die Methoden, um eine Projektdatei zu lesen, zu verarbeiten und zu speichern. Sie enthält auch die Methoden, um eine Messung zu starten. Einige Methoden und Objekte dieser Klasse sind in den folgenden Zeilen angegeben.

```
class TCellChipsProject{
 protected:
        FILE *theOriginalFile;     //File used to read the project data
        FILE *theSwapFile;         //File used to write temporary data

        Chip theChips[24];
        Curve *theCurves[1000];
        TPanelGraph *thePanelGraphs[100];
        TPanelImage *thePanelImages[100];

        int nbCurves;               //number of Curves objects
        int nbChips;                //number of Chip objects
        int nbXValues;              //number of different X values
        int TFormGraphNb;           //number of TPanelGraph objects
        int ImageNb;                //number of PanelImage objects

        TScrollBox *thePanels[30];  //a scrollBox contains up to 6 "TPanelGraph"
        int visiblePanel;           //number of the scrollBox shown
        int PanelsNb;               //number of scrollBox objects

        PumpONOFF thePumpONOFF;
        TFrameGraphTreeView *theGraphTreeView;
        TFrameGeneralInfos *theGeneralInfos;
        TFrameChipsInfos *theChipsInfos;
        TFrameSubstances *theSubstancesInfos;

        //to read and plot a saved project:
        void readCurve(FILE *fin);
        void readChip(FILE *fin,int chipNb );
        void readGraph(FILE *fin);
        void readSubstances(FILE *fin);
        void readPumpONOFF(FILE *fin);
        void readComments(FILE *fin);
        int readImage(FILE *fin);

        //creates a new PanelGraph and places it in an ScrollBox:
        void insertPanelGraph( YAxis *theYax,int YAxisNb,char *theName,
                               char *theUnits_x);
        void addCurve(char *Curvename,int chip,int XValueNum,bool pump);
        void addGraph(int *curveNum, int nbCurves, char *nameOfGraph,
                      char *unitsY);
```

```
        //variables and methods used to plot and save a measurement:
        TTimer *timerMeasure;
        void __fastcall TimerClicked(TObject *Sender);
        FILE *fx[10];
        FILE *fy[20];
        FILE *fx_read[10];
        FILE *fy_read[20];
        int FileYvalueNb[10];
        SaveMeasurement theSaveMeasurementFiles;

        void Compute_derive(int curveNum);
        ...

public:
        //to open and plot a project:
        int open(char *filename,TFrameGraphTreeView *FTreeView,
                 TFrameGeneralInfos *f,TFrameChipsInfos *theChipsInf,
                 TFrameSubstances *theSubstancesInf);
        void close();

        //methods to process a saved project:
        //they open a dialog box to choose the curves, that must be
        //processed and process it:
        void compute_DeriveAllCurves();
        void compute_DriftCorrectionAllCurves();
        ...

        //to save the project
        void save(char *filename);

        //methods and variables used to start and end a measurement:
        char *programDirectory;
        int startMeasurement(TFrameGeneralInfos *f,
                             TFrameChipsInfos *theChipsInf,
                             TFrameSubstances *theSubstancesInf);
        void setTimeDelay(double delay);
        void endMeasurement();

        //methods that open a dialog box to set the name of cells
        //and culture medium or substances added:
        void addSubstance();
        void cellsDialOpen();

        //method used to send the graphs to a printer:
        void print_project();

        void seePrecedentGraphs();  //to see the previous scrollBox
                                    //which contains many graphs
        void seeNextGraphs();
};
```

Wenn eine Messung gestartet wird, werden die X-Koordinaten in die binären Dateien „*fx[0]*“, „*fx[1]*“, .., und die Y-Koordinaten in die Dateien „*fy[0]*“, „*fy[1]*“... geschrieben. Im Attribut „*lastXValue*“ und „*lastYValue*“ der Klasse „*Curve*“ wird der letzte Messpunkt gespeichert, um danach in einer ASCII-Textdatei abgelegt zu werden. In den folgenden Zeilen wird gezeigt, wie neue Messpunkte in die binären Dateien „*fx[0]*“ und „*fy[0]*“ geschrieben werden.

```
double tx,ty;   //time of measurement and value measured
int i;
tx=10;ty=0.8

fwrite(&tx,sizeof(double),1,fx[0]);

for(i=0;i<nbCurves;i++){
        theCurves[i]->lastXValue=tx;
        theCurves[i]->lastYValue=ty;
        fwrite(&ty,sizeof(double),1,fy[0]);
        theCurves[i]->nbPoints++;
}

fflush(fy[0]);
fflush(fx[0]);
```

Um die Graphik neu zu zeichnen, wird die Methode „*redraw*“ der „*PanelGraph*“-Klasse verwendet. Zum Beispiel wird für die Zeichnung der ersten Graphik folgender Aufruf benutzt:

```
thePanelGraphs[0]->redraw();
```

- „*SaveMeasurement*“-Klasse
  Für die Erstellung eines Messverzeichnisses und die Speicherung der Daten werden Funktionen der Klasse „*SaveMeasurement*“ benutzt. Die Implementierung dieser Funktionen wurde bereits in Kapitel 5.4 erklärt. Die Deklaration einiger Methoden und Attribute dieser Klasse sind in den folgenden Zeilen angegeben.

```
class SaveMeasurement{
 protected:...
 public:
        char *programDirectory;
        Curve **theCurves;
        int curveNb;
        int nbChips;
        int nbImages;
        int TFormGraphNb;
        TPanelGraph **theGraphs;
        TPanelImage **theImages;
        struct ProjectFileHeader *theHeader;
        PumpONOFF *thePumpONOFF;
        Chip *theChips;
        int nbXvalues;

        void CreateFilenames(); //creates a directory for the measurement
```

```
        //the method "saveMeasurement" must be called at the end of a measurement
        //to write the binary project file:
        void saveMeasurement(TFrameGeneralInfos *theGeneralInfos, char *filename);

        //to write the measurements data in ASCII files:
        void createASCII();//must be called one time at the begin
                           //of a new measurement:
                           //it creates the files and writes the legend
        void saveASCII(int nXValue,int pumpState);//it saves the last measurements
                                                  //of the curves, whose X values
                                                  //number is "nXvalue"
                                                  //in a ASCII file

};
```

- „*Substance*“, „*Chip*“, „*PumpONOFF*“
  „*Substance*“ ist eine Struktur, die Information über den Namen und die Zeit der Zugabe eines Wirkstoffs ins Kulturmedium enthält.

```
typedef struct Substance{
        char name[50];
        double time;
        double concentration;
} Substance;
```

Die Klasse „*Chip*“ enthält alle Informationen über den Chip während des Versuches, wie Zellname, Kulturmedium, Zugabe von Wirkstoffen.

```
class Chip{
 public:
        char Celltype[50];
        char FluidName[50];
        int numberOfCells;        //cells number at the begin of the experiment
        Substance *theSubstances; //pointer on a table of substances
        int nbSubstances;         //how many substances habe been added or removed
                                  //from the culture medium
        int nbSubstancesMax;      //how big is the table of substances

        void addSubstance(char *name,double time,double concentration);
        void freeSubstances();
};
```

„*PumpONOFF*“ ist eine Struktur, die die Zeiten des Ein/Ausschaltens der Pumpe enthält. Diese Zeiten sind in der Tabelle „*times*“ gespeichert. Der erste Wert der Tabelle ist der erste Zeitpunkt, an dem die Pumpe eingeschaltet wurde, der zweite Wert gibt an, wann sie ausgeschaltet wurde, ...usw. Diese Struktur ist hier angegeben:

```
typedef struct PumpONOFF{
        int number;    //number of values in the "times" table
        double *times; //times[0]:ON, times[1]:OFF, times[2]:ON ...
} PumpONOFF;
```

- Darstellung der Messkurven und Bilddaten: „*YAxis*“, „*TPanelGraph*“, „*TPanelImage*“
  „*YAxis*“ ist eine Struktur, die die Konfigurationen der Y-Achse speichert.

```
typedef struct YAxis{
        double valueMin;  //minimal y-value shown
        double valueMax;  //maximal y-value shown
        char units[50];   //units of the axis
        int nbOfCurves;   //number of curves that belong
                          //to the y-axis
        bool bestFitY;
        ListOfCurves *theCurves;
} YAxis;
```

„*TPanelGraph*“ und „*TPanelImage*“ sind Klassen, die von der Klasse „*TPanel*“ abgeleitet sind.

```
class TPanelGraph : public TPanel
{
private:
       TPopupMenu *PopupMenu;    //popup menu shown when the user clicks
                                 //on the graph
       TMenuItem *Xaxisconfiguration1;
       void __fastcall Xaxisconfiguration1Click(TObject *Sender);
       ...
       TGraph *theGraph;
public:
        __fastcall TPanelGraph(TComponent* Owner,YAxis *theYax,int YAxisNb,
                               char *name, char *units_x, int PosX, int PosY,
                               TPanel *panParent);

        YAxis *theYAxis[4];    //up to 4 y-axis can be defined
        int nbYax;             //number of different y-axis
        char name[200];        //name of the graphic
        char units_x[100];     //units of the x-axis

        void TPanelGraph_free();
        void redraw();
        void print_graph(TRect *theRect);
        void save(FILE *fout); //saves the graphic in a binary file
};

class TPanelImage : public TPanel
{
 private:
 public:
       __fastcall TPanelImage(TComponent* Owner, int PosX, int PosY,
                              TPanel *panParent, char *filename,
                              char *theName,int ch, double ti);
       TImage *theImage;
       char name[200];
```

```
        int chip;
        bool deleted;
        double time;
        char *filename;

        void TPanelImage_free();
        void save(FILE *fout, int number);
        void load(FILE *fout);
};
```

- Zeichnung der Messkurven: „*Curve*“, „*GraphImage*“, „*TGraph*“, „*ListOfCurves*“
  Die Klassen, die für die Zeichnung der Messkurve benutzt werden, sind in Abbildung 5.28 grau gefärbt. Die Implementierung der Funktionen für die Zeichnung der Messkurven wurde bereits in Kapitel 5.4 erklärt.

  Die Klasse „*Curve*“ enthält alle Informationen über die Messkurve, wie Namen, minimaler Wert, maximaler Wert, Position in der Datei der X-Werte und Y-Werte. Die Methode „*allocPoints*()“ erlaubt es, die Messdaten in den binären Dateien „*fPointsX*“ und „*fPointsY*“ zu lesen. Die Messwerte werden dann in die Tabellen „*PointsX*“ und „*PointsY*“ geschrieben. Die X-Werte müssen sequentiell in der Datei „*fPointsX*“ gespeichert werden. In der Datei „*fPointsY*“ können auch Y-Werte von anderen Messkurven gespeichert werden. Diese Datei hat das folgende Format ($y_i(t)$: Messwert für die Kurve i, zur Zeit t):

$y_0(0)$	$y_1(0)$	...	...	$y_n(0)$	$y_0(1)$	$y_1(1)$	...	...	$y_n(t)$

  Das Attribut „*YValuePosition*“ der „*Curve*“-Klasse zeigt, welche Kurvennummer (i) die Kurve hat. Das Attribut „*File_YValuesNb*“ zeigt, wieviel Y-Werte in derselben Datei gespeichert sind (n). Die wichtigsten Attribute der „*Curve*“-Klasse sind hier angegeben.

```
class Curve{
 public:
        char name[50];                  //name of the curve
        char filename[50];
        int curveType;                  //O2, pH, Temp, Rpar or Cpar
        int nbPoints;                   //number of points in the curve
        double *PointsX;                //table of the X coordinates
        double *PointsY;                //table of the Y coordinates
        FILE *fPointsX;                 //File with the X coordinates
        FILE *fPointsY;                 //File with the Y coordinates
        long filePosition_XValues;      //Position in the file of the X values
        long filePosition_YValues;      //Position in the file of the Y values
        int File_YValuesNb;             //how many Y values from other curves
                                        //are saved in the same file
        int YValuePosition;             //position of the Y value in the file
        double vmin, vmax;              //minimal und maximal Y values
        double tmin,tmax;               //minimal und maximal X values
        int ChipNb;
        PumpONOFF *thePumpONOFF;
        Chip *theChip;
        void exportCurve(char *filename, bool all, //export all the curve or
                         double t1, double t2);     //between the times t1 and t2
```

```
                                                    //in an ASCII file
        void allocPoints();             //reads the points in the binary files
                                        //and writes it in the tables PointsX and
                                        //PointsY
        void freePoints();

        double lastYValue;              //variables used to save the last points
        double lastXValue;              //measured

        ...
} ;
```

Die Klasse „*GraphImage*“ enthält alle Funktionen, um die Graphiken zu zeichnen, wie „*drawXAxis*()“ oder „*drawLegend*()“.

Die Klasse „*TGraph*“ implementiert alle Funktionen, die aufgerufen werden, wenn ein Ereignis auf der Graphik auftritt (Maus- oder Tastatur-Ereignisse). Zum Beispiel wird die folgende Funktion aufgerufen, wenn der Benutzer eine Graphik selektiert hat, und wenn er auf eine Taste drückt:

```
 void __fastcall keyDown(TObject *Sender,Word &Key, Classes::TShiftState Shift);
```

Die „*ListOfCurves*“-Struktur enthält eine Liste von „*Curve*“- und „*Color*“-Objekten, die in einer Graphik dargestellt werden. Die Funktion „*ListOfCurves_add*()“ erlaubt, ein Element in die Liste einzufügen. Die Funktion „*ListOfCurves_free*()“ löscht die Liste. Die Deklaration der Struktur und Funktionen sind hier angegeben:

```
struct ListOfCurves{
        int YaxisNb;
        TColor color;               //color of the curve
        Curve *theCurve;
        struct ListOfCurves *tail;
};

struct ListOfCurves *ListOfCurves_add(Curve *c,TColor col,struct ListOfCurves *l);

void ListOfCurves_free(struct ListOfCurves *l);
```

- „*ComputeSignal*“
  Die „*ComputeSignal*“-Datei enthält die Messkurvenverarbeitungen, die in diesem Kapitel bereits beschrieben wurden. Die Berechnung der Metabolismusraten erfolgt zum Beispiel durch die folgenden Funktionen:

```
void ComputeSignal_AcidificationRate(Curve *theCurve,Curve *theCurveResult);
void ComputeSignal_O2ConsumptionRate(Curve *theCurve,Curve *theCurveResult);
};
```

Der erste Parameter dieser Funktionen ist die gemessene Messkurve. In den zweiten Parameter wird das Ergebnis der Berechnungen geschrieben. Die „*ComputeSignal*“-Datei integriert andere Funktionen, die mit anderen Parametern aufgerufen werden, wie die folgende Funktion, die Mittelwerte von Messpunkten berechnet:

```
void ComputeSignal_smoothing(double *Xval, double *Yval, int l,
                             int smooth, double **Xval2,
                             double **Yval2, int *l2);
};
```

Die ersten zwei Parameter „$Xval$“ und „$Yval$“ sind die Tabelle der Messwerte. Der dritte Parameter ist die Länge der Tabellen. „*smooth*“ ist die Anzahl der Punkte, die gemittelt werden. „$Xval2$“ und „$Yval2$“ sind Zeiger auf die Tabelle, in die die Ergebnisse der Berechnungen geschrieben werden. „$l2$“ ist ein Zeiger auf die Länge der „$Xval2$“- oder „$Yval2$“-Tabelle. Der benötigte Speicherplatz für die Ergebnisse („$Xval2$“ und „$Yval2$“) wird von der Funktion „*ComputeSignal_smoothing*“ gewährt.

- „$TFrameGeneralInfos$“, „$TFrameChipsInfos$“ und „$TFrameSubstances$“
  Diese Klasse beschreiben die drei Frames, die im Hauptfenster dargestellt werden können. Die Klasse „$TFrameGeneralInfos$“ implementiert die Methode „$writeInfos$“ und „$clearInfos$“:

```
class TFrameGeneralInfos : public TFrame
{
 __published:...

 public:
        __fastcall TFrameGeneralInfos(TComponent* Owner);

        struct ProjectFileHeader *theProjectHeader;
        char *Aim;
        char *Problems;

        void writeInfos();
        void clearInfos();
};
```

Die Klassen „$TFrameGeneralInfos$“ und „$TFrameSubstances$“ implementieren die Methode „$writeInfos$“, „$clearInfos$“ und „*copyToClipboard*“ :

```
class TFrameSubstances : public TFrame
{
__published:...
public:
        __fastcall TFrameSubstances(TComponent* Owner);
        void writeInfos(Chip *theChips,int nbChips);
        void clearInfos();
        void copyToClipboard();
};

class TFrameChipsInfos : public TFrame
{
__published:...
public:
        __fastcall TFrameChipsInfos(TComponent* Owner);
        void writeInfos(Chip *theChips,int nbChips);
        void clearInfos();
```

```
        void copyToClipboard();
};
```

Der Programmcode wurde in mehrere Dateien unterteilt. Die Dateinamen und ihre allgemeine Funktionalitäten sind in der folgenden Tabelle aufgelistet.

Funktionalität	**Dateiname**
Import von ASCII-Messreihen	ASCIIImport.cpp, ASCIIImport.h TUMScreenImport.cpp, TUMScreenImport.h ZuhlkeImport.cpp, ZuhlkeImport.h
Klassen Jeweils „.cpp“- und „.h“-Datei	Chip, Curve, Graph TPanelGraph, TPanelImage, FilmInWindow ListOfCurves, GraphImage, ComputeSignal TCellChipMeasurement, TCellChipProject
Speicherung der Daten	SaveMeasurement.cpp SaveMeasurement.h
Projekt-Strukturen	ChipsProject_struct.h, YAxis.h
Projekt-Dateien	MDIAPP.bpr, mdiapp.cpp
Formulare: Jeweils „.cpp“-, „.h“- und „.dfm“-Dateien	
Hauptfenster	Main
Frames	TFrameInfosAndGraphs, TFrameChipsInfos TFrameGeneralInfos,TFrameSubstances
Import von ASCII-Messreihen	ASCIIImportDial, TUMScreenImportDial, ZuhlkeImportDial
Konfiguration der Datenverarbeitungen	ComputeSmoothingDial, CparDerDial, DerivationOptionsDial DivisionDial, ExponentielleDial, RegressionDial
Achsen-Konfiguration	XAxisDial, YAxisDial
Fehler	ErrorDial, OverwritingFileDial, DeletingGraphicDial
Sonstige Zwecke	ABOUT, AddSubstanceDial, ChoosingChipDial CommentsDial, CurveChoiceDial, CurvesColorDial, CurvesDial EditInfosDial ExportCurveDial,GeneralInfosDial, CellsDial SubstancesDial, NameOfGraphDial, MeasurementConfigDial

5.6.5.2 Format der Projektdatei

Es wurde bereits erklärt, dass die Software binäre Projektdateien öffnen und darstellen kann, deren Namen auf ”.chips” enden. Diese Dateien haben ein bestimmtes Format, das in Abbildung 5.29 dargestellt ist. Zuerst wird in der Datei die Struktur „*ProjectFileHeader* “ gespeichert. Diese Struktur enthält allgemeine Informationen, wie den Benutzernamen, das Datum des Versuchs, das Messgerät. Sie wird in den folgenden Zeilen angegeben:

```
struct ProjectFileHeader{
  double SoftwareVersion;
  char username[50];
```

```
  int day;
  int month;
  int year;
  int duration;
  int machine;  //0: IMola, 1:Glaschips ...aso.
};
```

Danach werden alle Objekte des Projektes gespeichert (Chips, Graphiken, Kurven ...usw). Jedes Objekt wird durch eine Nummer gekennzeichnet (*int ObjectType*), die in die Datei geschrieben wird. Danach wird eingetragen, welche Position (*long Position_Following_object*) das nächste Objekt in der Datei besetzt. Danach werden die Objektdaten geschrieben.

struct ProjectFileHeader	int ObjectType	long Position_Following_object	Datas...	int ObjectType	aso...

Abbildung 5.29: Format der Projektdatei

Die Nummer des Feldes „*ObjectType*“ ist eine Zahl zwischen 1 und 12. Sie ist für alle verfügbaren Objekte des Projektes in den folgenden Zeilen angegeben:

```
#define CHIP 1
#define GRAPH 2
#define CURVE 3
#define SUBSTANCES 4
#define PUMPONOFF 5
#define YAXIS 6
#define XVALUES 7
#define YVALUES 8
#define COMMENTS 9
#define IMAGE 10
#define FILM 11
#define GRAPH_POSITION 12
```

Das Format des „*Datas*“-Feldes hängt von dem gespeicherten Objekt ab. In Abbildung 5.30 wird dieses Format für alle Objekte des Projektes angegeben.

Objekt	Maximale Anzahl	Format des „*Datas*“-Feldes
Chip	24	struct CHIP_struct
Curve	1000	struct CURVES_struct
YAxis	n	struct YAXIS_struct;[int CurveNumber, int color]
Graph	100	struct GRAPH_struct;[long YAxisPosition]
PumpONOFF	1	int Nb, [double times]
Substances	24	int nbOfSubstances,[Substance]
XValues/Yvalues	n	[double values]
Comment	1	int nbChars; [char]

Abbildung 5.30: Format des Feldes „Datas“

Für das „*Chip*“-Objekt wird eine „Chip“-Struktur gespeichert, die in den folgenden Zeilen angegeben ist:

```
struct CHIP_struct{
  int number;
  char cells_name[100];
  char fluid_name[100];
  int cellsNumber;
  int nbOfCurves;
};
```

Für das „*Curve*“-Objekt wird eine „Curve“-Struktur gespeichert, die in den folgenden Zeilen angegeben ist:

```
struct CURVES_struct{
  char name[100];
  long valuesXPos;
  long valuesYPos;
  int valuesXNum;
  int nbPoints;
  int ChipNumber;
};
```

Für das „*YAxis*“-Objekt werden eine „YAxis“-Struktur und eine Tabelle von Kurvennummern und Farben gespeichert. Die „YAxis“-Struktur ist in den folgenden Zeilen angegeben:

```
struct YAXIS_struct{
  int number;
  int bestFitY;
  double min_y;
  double max_y;
  char units_y[100];
  int curvesNb;
};
```

Für das „*Graph*“-Objekt werden eine „Graph“-Struktur und die Tabelle der Position in der Datei der benutzten Y-Achsen gespeichert. Die „Graph“-Struktur ist in den folgenden Zeilen angegeben:

```
struct GRAPH_struct{
  int chip;
  char name[100];
  char units_x[100];
  int nbYaxis;
  int allPoints;
  double min_x;
  double max_x;
};
```

Für das „*PumpONOFF*“-Objekt wird die Zahl der Pumpzustände und eine Tabelle von Zeiten gespeichert.

Für das „*Substances*“-Objekt werden die Zahl der Wirkstoffe und eine Tabelle von „*Substances*“-Strukturen gespeichert. Die „Substances“-Struktur ist in den folgenden Zeilen angegeben:

```
typedef struct Substance{
  char name[50];
  double time;
} Substance;
```

Für die „*XValues*“- oder „*YValues*“-Objekte wird eine Tabelle von Messwerten (double) gespeichert. Für das „*Comments*“-Objekt werden die Anzahl der Charaktere und die Tabelle der Charaktere gespeichert.

5.7 Datenbank

Für die Archivierung der Zell-Chip-Daten wurde eine Datenbank entwickelt. Die Zell-Chip-Systeme sind Meßsysteme für Zellkulturen. Wenn Zellkulturversuche durchgeführt werden, müssen die Arbeitsbedingungen zusammen mit den Messergebnissen dokumentiert und archiviert werden. Eine manuelle Archivierung solcher Daten ist schnell sehr aufwendig und nicht leicht zugänglich. Deshalb ist die Realisierung einer Datenbank für Zellkulturversuche in einem Forschungslabor essentiell.

Die entwickelte Datenbank dient zur Archivierung der Zell-Chip-Daten und der Daten von anderen Zellkultur-Messgeräten, wie des Casy-Zellzählers oder des Fluoreszenz-Messgeräts. Die Datenbank wurde von A. Veit mit mySQL und dem Apache Web-Server programmiert. Eine Internet-Schnittstelle wurde mit der PHP-Skriptsprache implementiert. Das Menü des Web-Interface der Datenbank ist in Abbildung 5.31 dargestellt.

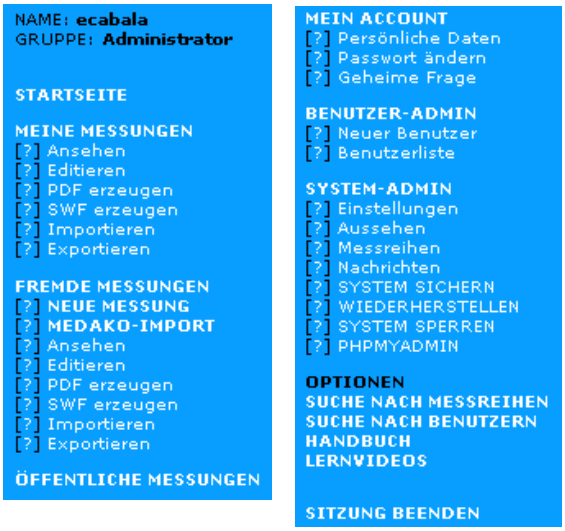


Abbildung 5.31: Menü des Web-Interface der entwickelten Datenbank für Zellkultur-Messungen.

Die entwickelte Datenbank mit Web-Interface bietet die folgenden Funktionalitäten:

- Benutzergruppen
  Mehrere Benutzergruppen mit verschiedenen Rechten sind definiert: Administrator, Betreuer von Messungen und einfache Benutzer ohne Schreibrechte. Diese Benutzergruppen erlauben eine sichere Verwendung der Datenbank, da nur die autorisierten Benutzer Datenbank-Daten schreiben, löschen oder konsultieren können.

- Import von Messdaten
  Das Web-Interface bietet die Möglichkeit, Daten in die Datenbank zu importieren. Verschiedene Informationen über den Versuch werden in der Datenbank archiviert. Das sind zum Beispiel Informationen über:

  – Das benutzte Zellkulturgerät (z. B.: Einkanal-Messgerät, TUM-Screen, Glas-Sensorchip-Messplatz, oder Mikroskop)
  – Sensorchips (Herstellungsnummer, Art)
  – Zellen, Zellzahl, Inkubationszeit

  - Kulturmedium

- Suche in der Datenbank
  Das Web-Interface bietet die Möglichkeit, in der Datenbank nach Daten zu suchen. Die Suche erfolgt nach verschiedenen Kriterien, wie benutzte Zellen, Messgerät oder Sensor-Chips.

Kapitel 6

Zusammenfassung der Ergebnisse und Diskussion

6.1 Zu Kapitel 4

Im Kapitel 4 wurden mehrere Versuchsergebnisse und ihre Verarbeitung präsentiert. In diesem Kapitel werden die vorgelegten Ergebnisse diskutiert.

- Metabolismusraten
  Zuerst wurde erklärt, wie die Metabolismusraten evaluiert werden. Für die Sauerstoffverbrauchsrate wurde die Möglichkeit aufgezeigt, sie durch exponentielle Regressionen auszuwerten. Es wurde aber erklärt, dass die Regressionsabweichung für einige Messzyklen der Versuche höher als die lineare Regressionsabweichung wird, da der Verlauf des Sauerstoffverbauchs von seiner Geschwindigkeit abhängt. Für die Auswertung der Sauerstoffverbauchsrate wurde deshalb die Steigung einer Regressionsgerade nach dem Pumpstop einige Minute lang berechnet. Die Ansäuerungsrate wurde mittels Berechnung der Steigung einer Regressionsgerade ausgewertet, wenn das Fluidik-System ausgeschaltet ist.

- Impedanz-Messergebnisse
  Impedanz-Ergebnisse wurden vorgestellt. Zum ersten Mal wurden langzeitige Versuche mit MCF-7- und L929-Zellen präsentiert, die parallel mit Phasenkontrast-Mikroskopie beobachtet wurden, und deren Zellzahl am Ende ausgewertet wurde. Ein Vergleich zwischen den Verläufen dieser beiden Zelllinien hat gezeigt, dass die Impedanz-Verläufe von den Zelltypen abhängen. Verschiedene Zelltypen haben tatsächlich verschiedene Morphologien und verschiedene Zell/Zell- oder Zell/Substrat-Kontakte. Sie zeigen auch ein anderes Wachstumverhalten, denn einige Zelllinien (wie z. B. die MCF-7-Zellen) wachsen übereinander, andere bleiben als Monolayer-Kultur erhalten.

  Versuche mit Histamin und Cytochalasin B haben deutlich gezeigt, dass schnelle Zellmorphologie/Adhäsions-Änderungen sehr leicht mit Impedanz-Sensoren zu detektieren sind. Die Impedanz-Messung ist deshalb eine einfache Methode, um solche Effekte zu messen. Dieser Sensor kann daher z. B. benutzt werden, um ein automatisches Mikroskop zu steuern. Wenn Änderungen in der Steigung des Impedanz-Signals detektiert werden, wird ein Befehl zum Mikroskop geschickt, damit eine Aufnahme erfolgt. Die Integration der Zell-Chip-Systeme mit einem automatisierten Mikroskop wird zurzeit realisiert und ist im nächsten Kapitel (7) ausgeführt.

  Es wurde erklärt, dass die Impedanz-Signale vieldeutig sind. Es wurde deshalb eine Methode dargelegt, um Zellwachstum, Zellsterben und Adhäsions-Änderungen getrennt auswerten zu können. Die Methode hat gezeigt, dass schnelle Änderungen in der Zelladhäsion oder Morphologie eindeutig ausgewertet werden können. Langsame Änderungen sind im Gegensatz dazu schwieriger

auszuwerten. Um eine genaue Aussage der Impedanz-Daten zu erhalten, könnten deshalb mikroskopische Aufnahmen benutzt werden. Ein Algorithmus für die Auswertung der Aufnahmen wurde am Ende des Kapitels dargestellt.

- Weiterverarbeitung der Messergebnisse
  Zum Schluss wurden Weiterverarbeitungsmöglichkeiten der Zell-Chip-Signale gezeigt. Es wurde erklärt, dass statistische Analyse der Daten sehr wichtig sind, um genaue Aussagen über Wirkstoffeffekte auf die Zellmetabolismusraten durchzuführen. Es wurde auch gezeigt, dass ein Vitalitätsindex sehr nützlich ist, um eine intuitive Interpretation der multiparametrischen Messdaten zu erhalten. Eine Methode für die Auswertungen des Index wurde vorgeschlagen.

6.2 Zu Kapitel 5

In Kapitel 5 wurden Software-Programme präsentiert, die mit der IDE „Borland C++ Builder 6.0“ realisiert wurden. In der Einleitung dieser Arbeit wurde bereits die Wahl dieser IDE diskutiert. In diesem Kapitel werden die Leistungen der entwickelten Software diskutiert.

Drei Zell-Chip-Programme wurden in dieser Arbeit beschrieben und ihre Implementierung wurde erklärt. Die Funktionalitäten einer Software für die Datenerfassung und Analyse von 1 bis 24 Sensorchips wurde zuerst präsentiert. Die Analyse-Funktionen bestehen hauptsächlich aus Regressionsanalysen, um Metabolismusraten zu erhalten, sowie einigen statistischen Auswertungen. Die Funktionalitäten einer Software für ein Einkanal-Zellchip-System und schließlich einer Software für das Glas-Sensorchip-Messsystem wurden beschrieben.

Vor ihrer Implementierung wurde die Software durch UML-Klassendiagrammen strukturiert. Ihre Struktur ist deswegen klar und die Programme sind relativ leicht zu ändern. Eine „C++ Builder“-Komponente wurde für die Darstellung der Messkurven programmiert, damit man über eine flexible Darstellung der Zell-Chip-Daten verfügt. Mit dieser Komponente können die Messkurven annotiert werden, um mehrere versuchszugehörige Informationen wie Pumpzustände oder eingefügte Wirkstoffe darzustellen. Das Verhalten und die Funktionalitäten dieser Komponente können leicht geändert werden, um sie an die Benutzeranforderungen anzupassen.

Die Entwicklung einer Datenbank mit Web-Interface wurde begonnen, um Messergebnisse von Zell-Chip-Systemen sowie von anderen Zellkultur-Messgeräten zu archivieren. Sie wird eine entscheidende Erleichterung für die Suche nach Messdaten und deren Auswertung bringen.

Die vorgestellten Programme haben ihre Aufgabe erfolgreich erfüllt. Die Messsoftware für das Einkanal-Messgerät wurde erfolgreich im Institut für Limnologie in Iffeldorf (von T. Stadthagen und seinen Studenten) benutzt, um Messungen mit Algen-Zellen durchzuführen. Die Mess-/Analyse-Software wurde im Institut ebenfalls erfolgreich eingesetzt, um Datenauswertungen zu erhalten. Was die Messsoftware des Glas-Sensorchips-Messsystems betrifft, wurde sie an unserem Institut bei mehrtägigen Messungen mit Glas-Sensorchips benutzt. Die Software wurde dabei auch erfolgreich von Biologen (E. Motrescu, M. Brischwein, M. Brückl) und Studenten verwendet.

Kapitel 7

Ausblick

Um eine zuverlässige statistische Analyse der Zell-Chip-Daten und ein Hochdurchsatz-Screening von Wirkstoffen zu erlauben, wird am Heinz Nixdorf-Lehrstuhl für Medizinische Elektronik derzeit ein neues Zell-Chip-System mit 24 parallelen Sensorchips auf Glas-Substrat entwickelt. Das neue System besteht aus einer 24-Multiwellplatte mit optischen pH- und O_2-Sensoren und einem IDES-Sensor, deren Größe die Standardgröße der 24-Wellplatten für Zellkulturen hat. Die optischen Sensoren und ihre Ausleseelektronik werden von der Firma „PreSens“ entwickelt [83]. Für den Wechsel des Kulturmediums und die Zugabe von Wirkstoffen wird ein Pipettier-Roboter benutzt. Der Roboter wird von der Firma H+P gebaut und seine Elektronik wird am Lehrstuhl für Medizinische Elektronik entwickelt (V. Lob). Zur Visualisierung wird das automatische „iMic“-Mikroskop der Firma „Till Photonics“ eingesetzt [84]. Das vollständige System ist in Abbildung 7.1 dargestellt.

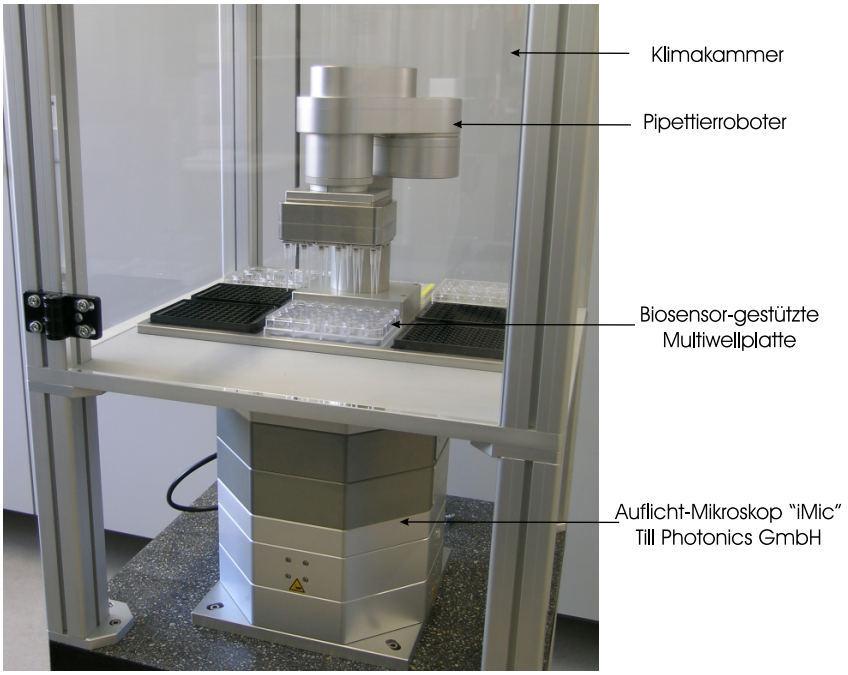


Abbildung 7.1: Hochdurchsatz-Zell-Chip-System mit automatisiertem Fluoreszenz-Mikroskop und Pipettier-Roboter, das zurzeit am Heinz Nixdorf-Lehrstuhl für Medizinische Elektronik der Technischen Universität München entwickelt wird (V. Lob [41]).

Auch für dieses komplexe System kann die entwickelte Mess-/Analyse-Software dieser Arbeit für die Darstellung und Auswertung der Multiwellplatten-Daten eingesetzt werden. Dafür sollen die Kommunikationsfunktionen mit dem Pipettierroboter und der Multiwellplatten-Elektronik noch implementiert

werden. Für das „iMic“-Mikroskop wurden bereits eine SDK mit ActiveX-Kontrollelementen von „Till Photonics“ entwickelt, die in die gesamte Software integriert werden soll.

Dieses Hochdurchsatz-Zell-Chip-System ermöglicht essentielle Anwendungen, wie Chemosensitiväts-Tests. Die Datenauswertungen, die in dieser Arbeit aufgezeigt wurden, werden mit zunehmenden Datenmenge optimiert werden können, um in Realzeit den Zustand der Zellkultur genau zu ermitteln.

Kapitel 8

Zusammenfassung

8.1 Zusammenfassung

Zell-Chip-Systeme sind multiparametrische Messsyteme, die erlauben, die metabolische Aktivität von Zellkulturen in Realzeit und über längere Zeiträume auszuwerten. In dieser Dissertation werden zunächst verschiedene Zell-Chip-Systeme und deren Elektronik präsentiert und beschrieben. Danach werden Messergebnisse und die Probleme ihrer Verarbeitung vorgestellt und diskutiert. Zuletzt wird die Entwicklung von einer speziellen Mess- und Analyse-Software für diese Messsysteme erklärt und diskutiert. Dabei wird ein elektronisch ermittelter Vitalitätsindex für Zellen und Gewebe errechnet, der für die systemische Wirkstoff-Forschung von großer Bedeutung ist. Anhand dieses Indexes kann die Sensitivität und Resistenz von Zellen gegenüber Therapeutika online und real time ermittelt werden.

8.2 Abstract

Cell Chip Systems are multiparametric measuring systems which allow to record the metabolic activity of cell cultures in real-time and over long periods of times. In this thesis, different cell chip systems and their electronics are presented. Measurement curves and problems in their processing are described and discussed. Finally, the development of data acquisition and analysis software for cell chip systems is explained and discussed. With this software a vitality index for cells and tissue, which has a significant role for screening agents, is evaluated. By means of this index, the sensitivity and resistance of cells to therapeutika can be evaluated online and in realtime .

8.3 Résumé

Les systèmes "Cell on chips" sont des systèmes de mesure multiparamétriques qui permettent d'évaluer l'activité métabolique de cultures cellulaires en temps réel et pendant plusieurs jours. Dans cette thèse, différents systèmes "Cell on chips" et leur électronique sont tout d'abord présentés et décrits. Ensuite, des résultats typiques de mesures et les problèmes relatifs à leur traitement sont présentés et discutés. Enfin le développement de logiciels de mesure et d'analyse pour ces systèmes est expliqué et discuté. Avec ces logiciels, un index de vitalité pour les cellules et les tissus, qui joue un rôle significatif dans la recherche de substances actives, est évalué. A l'aide de cet index, la sensibilité et la résistance de cellules à des médicaments peuvent être évaluées en temps réel.

Abkürzungen

Abkürzung	Bedeutung
AD	Analog/Digital
ADP	Adenosin-Diphosphat
ASCII	American Standard Code for Information Interchange
ATP	Adenosin-Triphosphat
AVI	Audio Video Interleaved
BMP	Bitmap
CB	Cytochalasin B
DMEM	Dulbecco modifiziertes Eagle-MEM
DNA	Desoxyribonucleinsäure
FCS	Fetal Calb Serum
GUI	Graphical User Interface
IDE	Integrated Development Environment
IDES	Inter-Digital Electrode Structure
ISFET	Ions Sensitive Field Effect Transistor
Konz.	Konzentration
$\mu M.$	Mikro-Molar
MDI	Multiple Document Interface
MEM	Minimum Essential Medium
MOS	Metaloxid-Sensor
OOA	Object Oriented Analysis
OOD	Object Oriented Design
RAD	Rapid Application Development
RNA	Ribonucleinsäure
SDI	Single Document Interface
SDK	Software Development Kit
Std.	Stunde
SQL	Structured Query Language (Datenbanksprache)
UML	Unified Modelling Language
VCL	Visual Component Library (Borland)
XML	Extensible Markup Language

Literaturverzeichnis

Biophysik

[1] B. Albert, A. Johnson, J. Lewis, M. Raff, K. Roberts. P. Walter: „Molekular Biologie der Zelle“, Garland Science, 2002

[2] B. Bassea, B. C. Baguleyb, E. S. Marshallb, G. C. Wakea, D. J.N. Walla: „Modelling cell population growth with applications to cancer therapy in human tumour cell lines“, Progress in Biophysics & Molecular Biology 85 353-368, 2004

[3] W. Bursch, A. Ellingert, L. Török, W. Parzefall,S. Coulibaly, K. Hochegger, M. Schorkhuber, G. Partik, B. Marian, R. Walkers, M. Sikorska, R. Schulte-Hermann: „In vitro studies on subtypes and regulation of active cell death“, Toxicology in vitro 11, 579-588, 1997

[4] JM. Crawford, NS. Braunwald: „Toxicity in vital fluorescence microscopy: effect of dimethylsulfoxide, rhodamine-123, and DiI-low density lipoprotein on fibroblast growth in vitro“, In Vitro Cell Dev Biol. 27A(8):633-8., 1991

[5] R. I. Freshney: “Tierische Zellkulturen, ein Methoden-Handbuch“, Walter de Gruyter, 1990

[6] B. Gabler, C. Winkler, A. K. Dreiss, J. Marshall, C. P. Lohmann: „Vitality of epithelial cells after alcohol exposure during laser-assisted subepithelial keratectomy flap preparation“, J Cataract Refract Surg 28 1841-1846, 2002

[7] R. Glaser: „Biophysics“, Springer Verlag, 2001

[8] N. G. Greena, A. Ramosa, H. Morga: „Numerical solution of the dielectrophoretic and travelling wave forces for interdigitated electrode arrays using the finite element method“, Journal of Electrostatics 56, 2002

[9] R. King: „Are biological processes too complex to model?“, Mathematics and Computers in Simulation 39, 583-588, 1995

[10] J. Koolman, K.-H. Röhm: „Taschenatlas der Biochemie“, Thieme, 2003

[11] M. Kraus, B. Wolf: „Emergence of Self-Organisation in Tumor Cells: Relevance for Diagnosis and Therapy“, Tumor Biology 14, 338-353, 1993

[12] M. Kraus, B. Wolf: „Structured Biological Modelling - A New Approach to Biophysical Cell Biology“, CRC Press, Inc., Boca Raton, 1995

[13] A. L. Lehninger: „Prinzipien der Biochemie“, Walter de Gruyter, 1987

[14] J. C. Mellor, C. DeLisi: „Circuits of the cell“, BIOSILICO Vol. 2, No. 3, 2004

[15] K. Moser, A. Stacher: „Chemotherapie maligner Erkrankungen, Leitfaden für Klinik und Praxis“, Deutscher Ärzte-Verlag, 1989

[16] J. M. Petit, A. Maftah, R. Julien: „Biologie cellulaire“, Dunod, 2002

[17] R. D. Schmidt: „Taschenatlas der Biotechnologie und Gentechnik “, Wiley-VCH, 2002

[18] C. Sorokin, J. Myers: „The course of respiration during the life cycle of chlorella cells“, J. Oen. Physxoi., Vol. 40, No. 4, 1957

[19] H. Stöcker, F. Jundt, G. Guillaume: „Toute la physique“, Dunod, 1999

[20] B. Wolf, M. Kraus: „Bedeutung der zellulären Selbstorganisation für die Tumorbiologie“, Naturwissenschaften 80, 343-352, 1993

[21] B. Wolf, M. Kraus: „pH-abhängige Selbstorganisation von Tumorwachstum und Invasion“, Dtsch. Zschr. Onkol. 27,3 , 1995

[22] B. Wolf, V. Dinger, C. Weiler, A. Schwinde, P. Scheipers, M. Kraus: „Drug targeting and metabolic investigations of cryoprepared tumor cells with analytical electron energy loss spectroscopy“, Tumor Biol. 17 234-250, 1996

Sensoren

[23] J. W. Deitmer, D. Schild: „Ca^{2+} und pH, Ionenmessungen in Zellen und Geweben“, Spektrum Akademischer Verlag GmbH, 2000

[24] S. Grimmes and O. G. Martinsen: „Bioimpedance & Bioelectricity Basics“, Academic Press, 2000

[25] P. Gründler: „Chemische Sensoren, Eine Einführung für Naturwissenschaftler und Ingenieure“, Springer Verlag, 2004

[26] I.Klimant, M.Kühl, R.N.Glud, G.Holst: „Optical measurement of oxygen and temperature in microscale: strategies and biological applications“, Sensors and Actuators B, 38-39, 29-37, 1997

[27] M. Lambrechts, W. Sansen: „Biosensors: microelectrochemical devices“, Institute of Physics Publishing, 1992

[28] J.A. Mihell, J.K. Atkinson: „Planar thick-film pH electrodes based on ruthenium dioxide hydrate“, Sensors and Actuators B 48 505-511, 1998

[29] W.-D. Schmidt: „Sensor-Schaltungstechnik“, Vogel, 2002

Lab-on-Chip Systeme

[30] S. Arndt, J. Seebach, K. Psathaki, H.-J. Galla, J. Wegener: „Bioelectrical impedance assay to monitor changes in cell shape during apoptosis“, Biosensors & Bioelectronics 19 583-594, 2004

[31] M. Brischwein, E. R. Motrescu, A. M. Otto, E. Cabala, H. Grothe, B. Wolf: „Functional Cellular Assays with Multiparametric Silicon Sensor Chips“, Lab on a Chip 3 (4) 234-240, 2003

[32] C. J. Cao, R. J. Mioduszewski, D. E. Menking, J. J. Valdesp, V. I. Cortes, M. E. Eldefrawi and A. T. Eldefrawi: „Validation of the Cytosensor for In Vitro Cytotoxicity Studies“, Toxicology in Vitro 11 285-293, 1997

[33] B.-W. Changa, C.-H. Chena, S.-J. Ding, D. Chan-Hen Chend, H.-C. Changa: „Impedimetric monitoring of cell attachment on interdigitated microelectrodes“, Sensors and Actuators B, 2004

[34] R. Ehret: „Zelluläre Sensorik mit Interdigitalen Elektrodenstrukturen für die biomedizinische Forschung“, 1996

[35] R. Ehret, W. Baumann, M. Brischwein, A. Schwinde, K. Stegbauer, B. Wolf: „Monitoring of cellular behaviour by impedance measurements on interdigitated electrode structures“, Biosensors & Bioelectronics 12 29-41, 1997

[36] S. E. Eklund, D. E. Cliffel, E. Kozlov, A. Prokop, J. Wikswo, F. Baudenbacher: „Modification of the Cytosensor^{TM} microphysiometer to simultaneously measure extracellular acidification and oxygen consumption rates“, Analytica Chimica Acta 496 93-101, 2003

[37] K. A. Giuliano, D. L. Taylor: „Fluorescent-protein biosensors: new tools for drug discovery“, Tibtech, Vol. 16, 1998

[38] T. Henning, M. Brischwein, W. Baumann, R. Ehret, I. Freund, R. Lammerer, M. Lehmann, A. Schwinde, B. Wolf: „Approach to a multiparametric sensor-chip-based tumor chensitivity assay“, Anti-Cancer Drugs 12, pp. 21-32, 2001

[39] S. Kintzios, I. Marinopoulou, G. Moschopoulou, O. Mangana, K. Nomikou, K. Endo, I. Papanastasiou, A. Simonian: „Development of a novel, multi-analyte biosensor for assaying cell division: Identification of cell proliferation/death precursor events“, Biosensors & Bioelectronics, 2005

[40] E. Landwojtowicz, P. Nervi, A. Seelig: „Real-Time Monitoring of P-Glycoprotein Activation in Living Cells“, Biochemistry 41, 8050-8057, 2002

[41] V. Lob, M. Brischwein, H. Grothe, J. Ressler, K. Kaufmann, B. Wolf: „Cell-based assays: Mikrosensorarray-basiertes Screening an lebenden Zellen und Geweben“, BIOSpektrum Sonderausgabe, 11. Jahrgang, 2005

[42] D. L. Taylor, E. S. Woo, K. A. Giuliano: „Real-time molecular and cellular analysis: the new frontier of drug discovery“, Current Opinion in Biotechnology 12, 2001

[43] M. Lehman, W. Baumann, M. Brischwein, R. Ehret, M. Kraus, A. Schwinde, M. Bitzenhofer, I. Freund, B. Wolf: „Non-invasive measurement of cell membrane associated proton gradients by ion-sensitive field effext transistor arrays for microphysiological and bioelectronical applications“, Biosensors & Bioelctronics 15 117-124, 2000

[44] E. Motrescu: „Analysis of Biological Signals with Multifunctional Bioelectronics Sensor Chips on Living Cells“, 2004

[45] E. R. Motrescu, A. M. Otto, M. Brischwein, S. Zahler, B. Wolf: „Dynamic analysis of metabolic effects of chloroacetaldehyde and cytochalasin B on tumor cells using bioelectronic sensor chip“, J. Cancer Res. Clin. Oncol., 2005

[46] J. Owicki, J. Wallace Parce: „Biosensors based on the energy metabolism of living cells: The physical chemistry and cell biology of extracellular acidification“, Biosensors & Bioelectronics 7 255-272, 1992

[47] M.L. Pourciel-Gouzy, W. Sant, I. Humenyuk, L. Malaquin, X. Dollat, P. Temple-Boyer: „Development of pH-ISFET sensors for the detection of bacterial activity“, Sensors and Actuators B 103 247-251, 2004

[48] J. Ressler, H. Grothe, M. Brischwein, B. Wolf: „Sensor-supported 24-Well plates: multifunctional tools for biological and biomedical HTS applications“, DrugPlus International 19-21, 2004

[49] A.-K. Souida, K. A. Tackaa, K. A. Galvana, H. S. Penefsky: „Immediate effects of anticancer drugs on mitochondrial oxygen consumption“, Biochemical Pharmacology 66 977-987, 2003

[50] J. Wegener, C. R. Keese, I. Giaever: „Electric Cell-Substrate Impedance Sensing (ECIS) as a Noninvasive Means to Monitor the Kinetics of Cell Spreading to Artificial Surfaces“, Experimental Cell Research 259, 158-166, 2000

[51] B. Wolf, M. Brischwein, W. Baumann, R. Ehret, M. Kraus: „Monitoring of cellular signalling and metabolism with modular sensor-technique: The PhysioControl-Microsystem (PCM)“, Biosensors & Bioelectronics 13 501-509, 1998

[52] B. Wolf, M. Kraus, M. Brischwein, R. Ehret, W. Baumann, M. Lehmann: „Biofunctional hybrid structures-cell-silicon hybrids for applications in biomedicine an bioinformatics“, Bioelectrochemistry and Bioenergetics 46 215-225, 1998

Datenverarbeitung

[53] S.Charbonnier: „On line extraction of temporal episodes from ICU high-frequency data: A visual support for signal interpretation“, Computer Methods and Programs in Biomedicine 78, 115-132, 2005

[54] U. Fayyad, G. G. Grinstein, A. Wierse: „Information visualization in data mining and knowledge discovery“, Academic Press, 2002

[55] K. Kroschel: „Statistische Informationstechnik, Signal- und Mustererkennung, Parameter- und Signalschätzung“, Springer Verlag, 2004

[56] S. Mahmoodia, B. S. Sharif: „Signal segmentation and denoising algorithm based on energy optimisation“, Signal Processing 85, 1845-1851, 2005

[57] M. Petrou: „Image Processing. The Fundamentals“, John Wiley & Sons Ltd, 1999

[58] W. Voß: „Taschenbuch der Statistik“, Carl Hanser Verlag, 2004

[59] C. Weigand: „Statistik mit und ohne Zufall“, Physica-Verlag, 2006

[60] Xiaobo Zhou, Xinhua Cao, Zach Perlman, Stephen T.C. Wong a: „A computerized cellular imaging system for high content analysis in Monastrol suppressor screens“, Journal of Biomedical Informatics 39, 115-125, 2006

Software

[61] E. Baroth, C. Hartsough: „Visual programming in the real world“, Visual Object-Oriented Programming: Concepts and Environments (M. Burnett, A. Goldberg, T. Lewis, eds), chapter 2, Manning Publications Co., Greenwich, CT, pp. 21-42, 1995

[62] G. Booch: „Object-Oriented Analysis and Design with Applications“, Benjamin/Cummings, Redwood City, CA, 1994

[63] B. Bruegge, A. H. Dutoit: „Object-oriented software engineering, Conquering complex and changing systems“, Prentice Hall, 2000

[64] P. Coad, J. Nicola: „Objekt-orientierte Programmierung“, Prentice Hall, 1994

[65] M. Urbano Cuadrado, M.D. Luque de Castro, M.A. Gomez-Nieto: „Object-oriented techniques for design and development of standard software solutions in automation and data management in analytical chemistry“, Trends in Analytical Chemistry, Vol. 25, No. 1, 2006

[66] S. Dupin: „Le Langage C++“, CampusPress France, 2000

[67] B. W. Kernighan, D. M. Ritchie: „Le Langage C Norme ANSI“, Masson, 1999

[68] T. R. G. Green, M. Petre: „When visual programs are harder to read than textual programs“. Proceedings of the 6th European Conference on Cognitive Ergonomics (ECCE 6), pp. 167-180, 1992

[69] R. B. Klinea, A. Seffah: „Evaluation of integrated software development environments: Challenges and results from three empirical studies“, Int. J. Human-Computer Studies 63 607-627, 2005

[70] W. H. Press, S. A. Teukolsky, W. T. Vetterling, B. P. Flannery: „Numerical recipes in C: the art of scientific computing“, Cambridge University Press, 1988

[71] K. N. Whitley, A. F. Blackwell: „Visual Programming in the Wild: A Survey of LabVIEW Programmers“, Journal of Visual Languages and Computing, 435-472, 2001

[72] S. Wiedenbecka, V. Ramalingama, S. Sarasammaa, C. L. Corritoreb: „A comparison of the comprehension of object-oriented and procedural programs by novice programmers“, Interacting with Computers 11 255-282, 1999

Internetseiten

[73] „Acea Biosciences“-Firma:
http://www.aceabio.com

[74] „Agilent“-Firma:
http://www.agilent.com

[75] Borland®:
http://www.borland.com

[76] Casy®-Zellzähler:
www.casy-technology.com

[77] „Evotec“-Firma:
http://www.evotec.com

[78] GNU-Projekt:
http://www.gnu.org/home.html

[79] „The handbook: A guide to fluorescent probes and labeling technologies“:
http://probes.invitrogen.com/handbook/

[80] Microsoft Development pages:
http://msdn.microsoft.com

[81] LaTeX-Distribution (TeX-Kompiler und andere Komponenten):
http://www.miktex.org

[82] „Molecular Devices“-Firma:
http://www.moleculardevices.com

[83] „PreSens“-Firma:
http://www.presens.de/html/start.html

[84] „TILL Photonics GmbH“:
http://www.till-photonics.com/Company/imprint.php

[85] „Vim“, Text-Editor:
http://www.vim.org/index.php