universität wien

MAGISTERARBEIT

Titel der Magisterarbeit

„Die Integration des ANT Protokolls in LabVIEW zur Erstellung drahtloser Messsysteme im Sport, dargestellt am Beispiel eines Feedbacksystems zur Optimierung des Zielverhaltens im Biathlonsport.“

Verfasser

Martin Böcskör, Bakk. rer. nat.

angestrebter akademischer Grad

Magister der Naturwissenschaft (Mag. rer. nat.)

Wien, 2009

Studienkennzahl lt. Studienblatt:	A 066 826
Studienrichtung lt. Studienblatt:	Magisterstudium Sportwissenschaft
Betreuerin / Betreuer:	Univ.-Prof. Dipl.-Ing. Dr. Arnold Baca

Danksagung

An dieser Stelle möchte ich mich bei all jenen bedanken, die mir beim Erstellen dieser Arbeit durch ihre Unterstützung zur Seite gestanden sind. Mein besonderer Dank gilt Dipl. Ing. Philipp Kornfeind für die technische Unterstützung, Dipl.-Sporting. Dr. Mario Heller, der immer mit Rat und Tat zur Seite stand und Gerald Sonnberger, Bakk. rer. nat., der mich überhaupt erst auf die Idee brachte, diese Magisterarbeit zu schreiben.

Vielen Dank an Mag. Brian Horsak, Sebastian Bichler, M.A., Mag[a]. Eva Maria Karall, Bakk. rer. nat., für die hilfreichen Anmerkungen, Hinweise und Ratschläge.

Besonders Bedanken möchte ich mich bei Mag. Oliver Strubreither, für die Hilfe, die vielen Tipps und Ratschläge und die Motivation – Danke Olli!

Bedanken möchte ich mich auch bei dem Support Team der Firma ANT Wireless und den Nutzern von labviewforum.de für alle Tipps zur Programmierung.

Dem gesamten Team der Abteilung Biomechanik/Bewegungswissenschaft und Sportinformatik danke ich für die schöne und lehrreiche Zeit. Danke an meinen Betreuer Univ. Prof. Dipl. Ing. Dr. techn. Arnold Baca für das Vertrauen und die Unterstützung.

Christina Fuith danke ich für Ausdauer, Geduld, für das Verständnis und das Aufmuntern, wenn es mal nicht wie gewollt lief.

Ganz besonders bedanken möchte ich mich bei meiner Mutter Hildegard, meinem verstorbenen Vater Josef und meiner Familie, die mein Studium erst ermöglicht haben, mich all die Jahre hinweg unterstützt haben. Danke dafür, dass ihr nie an mir gezweifelt und mir all das ermöglicht habt.

Inhaltsverzeichnis

1. Einleitung

Eines der zentralen Themen in der sportwissenschaftlichen Forschung ist das Analysieren, Verstehen und Optimieren von sportlichen Bewegungsabläufen. Sei es darum, die Grenzen der Belastbarkeit des Systems „Körper“ zu kennen, um innerhalb dieser Reize zu setzen und so ein gewünschtes Ergebnis (Verbesserung des Gesundheitszustandes oder auch die Erbringung sportlicher Höchstleistung) zu erzielen, oder um zu wissen, wieso eine Bewegungsausführung im Vergleich zu einer anderen eher zum Erreichen eines definierten Ziels führt. In diesem Kontext ist das Messen und Erheben von empirischen Daten Mittel zum Zweck, um Abläufe und Vorgänge besser zu verstehen und zu optimieren.

Die Sportwissenschaftlerin bzw. der Sportwissenschaftler, aber auch die Trainerin und der Trainer sehen sich mit vielen Systemen zur Datenakquise und Datenverarbeitung konfrontiert. Diese helfen, relevante Parameter zu messen und in einer geeigneten, verständlichen Form aufzubereiten. Das gewonnene Feedback soll einer Leistungsverbesserung dienen. Doch gerade im Bereich des Sports ist oftmals die fehlende Transparenz solcher Messsysteme zu beklagen. Sie berechnen Parameter und Indizes aus einer Fülle verschiedener Daten. Der Benutzer weiß dabei oft nicht, wie die Berechnung zu Stande kommt. Messsysteme können auf Gütekriterien hin überprüft werden um die Ergebnisse zu validieren. Doch was sollen Forscherinnen und Forscher tun, wenn es kein System für ein definierte Aufgabenstellung gibt, oder bestehende Systeme durch Kabel, Gewicht oder andere systemimmanente Eigenschaften dazu führen, dass die Messsituation und somit auch die Messergebnisse nicht der sportlichen Realität entsprechen? Zu Lösen ist diese Schwierigkeit durch das Erstellen eigener, auf die konkrete Messsituation angepasster Messsysteme. Dabei können diese Systeme mit aktuellen Technologien drahtlos, klein und leicht gehalten werden. Verfälschungen der Messsituation können so weitgehend vermieden bzw. vernachlässigbar gering gehalten werden.

1.1 Ziele und Gliederung der Arbeit

Ziel dieser Magisterarbeit ist die Entwicklung eines transparenten und skalierbaren Systems zur Datenerhebung beim Schussvorgang im Biathlon. Dieses System ist quelloffen und wurde bewusst mit generischen Eigenschaften entwickelt, um Entwicklerinnen bzw. Entwicklern als Schablone für eigene Systeme zu dienen. Der Umgang mit der SensRcore Technologie von ANT, die Nutzung und Nutzbarmachung des ANT Protokolls in LabVIEW und die Umsetzung einer Applikation werden detailliert beschrieben, um es Leserinnen und Leser dieser Ar-

beit zu ermöglichen, eigene drahtlose Mess- und Feedbacksysteme zu planen und zu implementieren.

Zu Beginn zeigt ein kurzer historischer Überblick der elektronischen Datenverarbeitung Entwicklungen und Trends auf, die die Grundvoraussetzungen für die Erstellung von drahtlosen Feedbacksystemen bilden. Die verwendeten Technologien, Schnittstellen und Programmiersprachen, die im Rahmen dieser Magisterarbeit zum Einsatz kamen bzw. programmiert und entworfen wurden, werden detailliert beschrieben. Abschließend beschäftigt sich diese Arbeit mir der Programmierung eines Feedbacksystems für den Biathlonsport und der Konfiguration der dabei eingesetzten Hardware. Im Detail wird der Umgang mit der ANT SensRcore Technologie zur Erfassung von Daten eines Flexiforce Drucksensors beschrieben. Die Datenübertragung wird mit dem ANT Protokoll realisiert. Die Darstellung und Speicherung der gewonnenen Messwerte erfolgt in einer LabVIEW Applikation. Die Programmierung der Schnittstelle zwischen der ANT Hardware und LabVIEW bildet einen zentralen Punkt dieser Arbeit.

Abbildung 1 zeigt die Struktur eines generischen Systems zur Datenerfassung mit Hilfe der ANT Hardware, der SensRcore Technologie, des ANT Protokolls und LabVIEW.

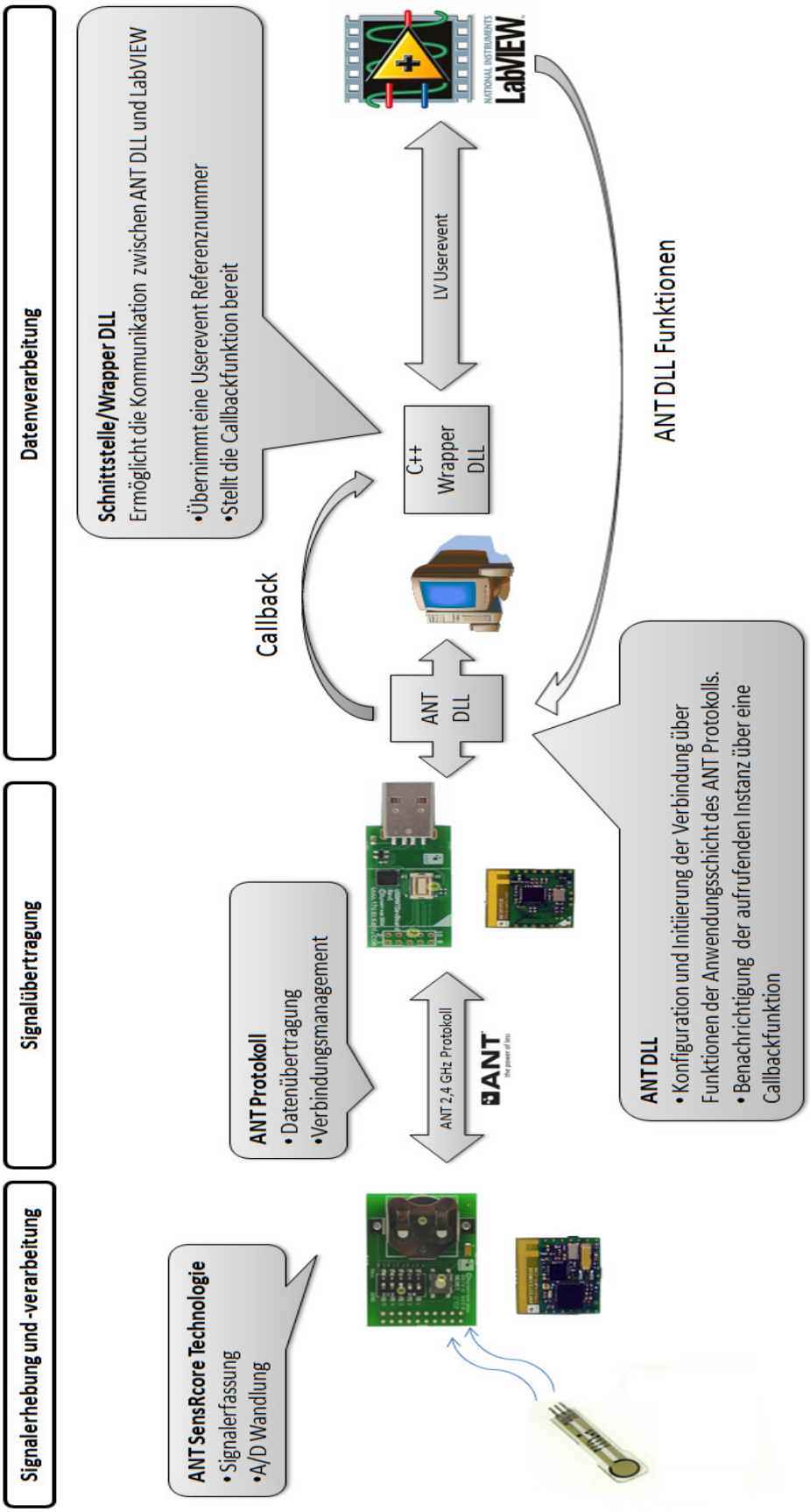


Abbildung 1: Überblick eines generischen Systems basierend auf der ANT Technologie und LabVIEW

Ein Hauptaugenmerk dieser Magisterarbeit liegt in der Entwicklung eines Referenzsystems zur Verwendung der ANT Technologie in LabVIEW. Abbildung 2 gibt einen Überblick des Systems aus der Sicht des Anwenders in Form eines Anwendungsfalldiagramms.

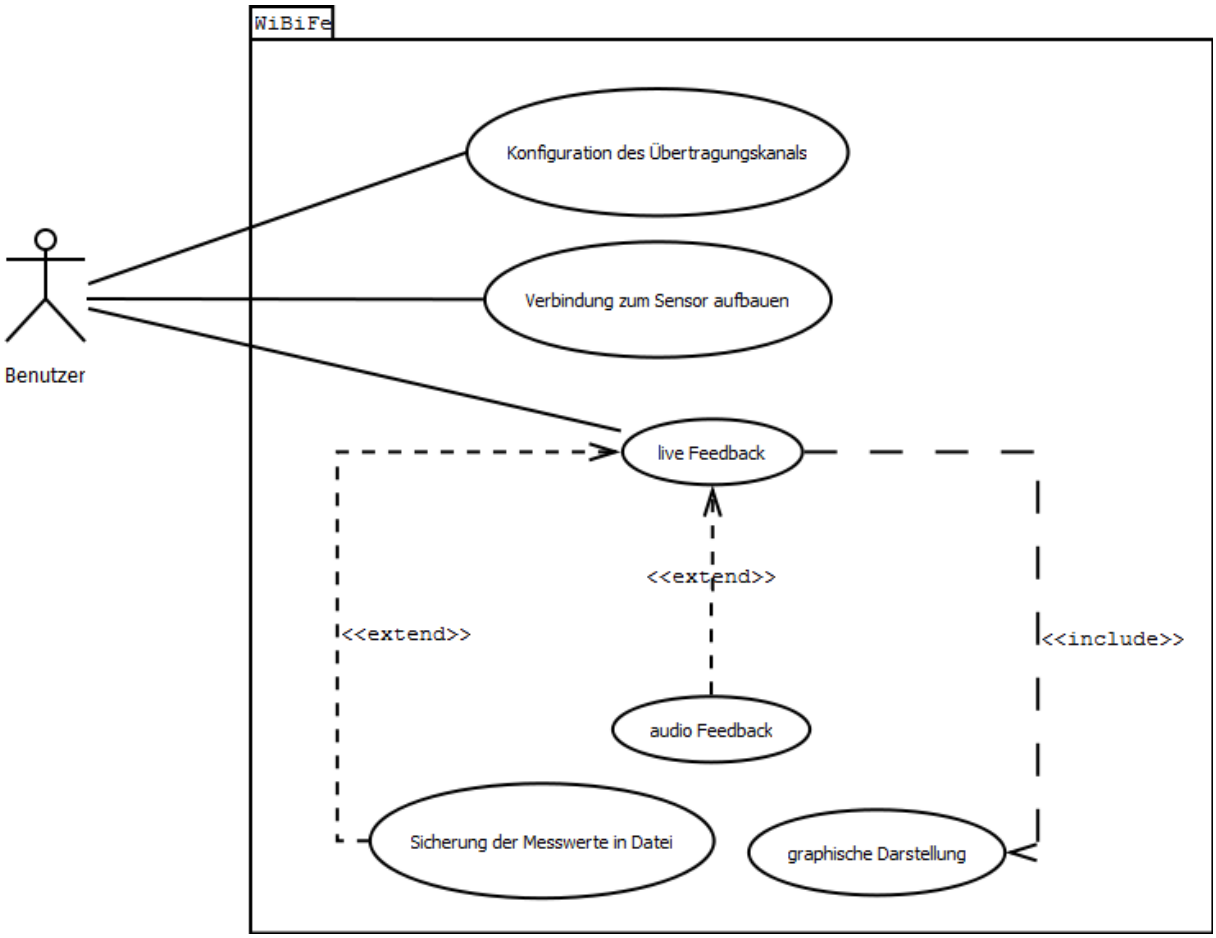


Abbildung 2: WiBiFe Anwendungsfalldiagramm

Eine genaue Beschreibung und Anleitung der vorgestellten Hard- und Software und der Programmierung sollen der Leserin bzw. dem Leser dieser Arbeit als Leitfaden zur Verwendung der beschriebenen Technologien dienen. Insbesondere die Programmierung und Verwendung der Schnittstelle (Wrapper[1] DLL[2]) zwischen dem ANT Protokoll und LabVIEW war ein Ziel dieser Arbeit. Folgendes Sequenzdiagramm (Abbildung 3) zeigt, wie der Nachrichtenfluss zwischen der ANT DLL und LabVIEW mit Hilfe der Wrapper DLL ermöglicht wird.

[1] Wrapper: Der Begriff Wrapper bezeichnet in der EDV eine Schnittstelle zwischen zwei Programmen.

[2] Dynamic Link Library: DLLs sind Befehls- bzw. Funktionssammlungen, die von mehreren Applikationen gemeinsam genutzt werden können.

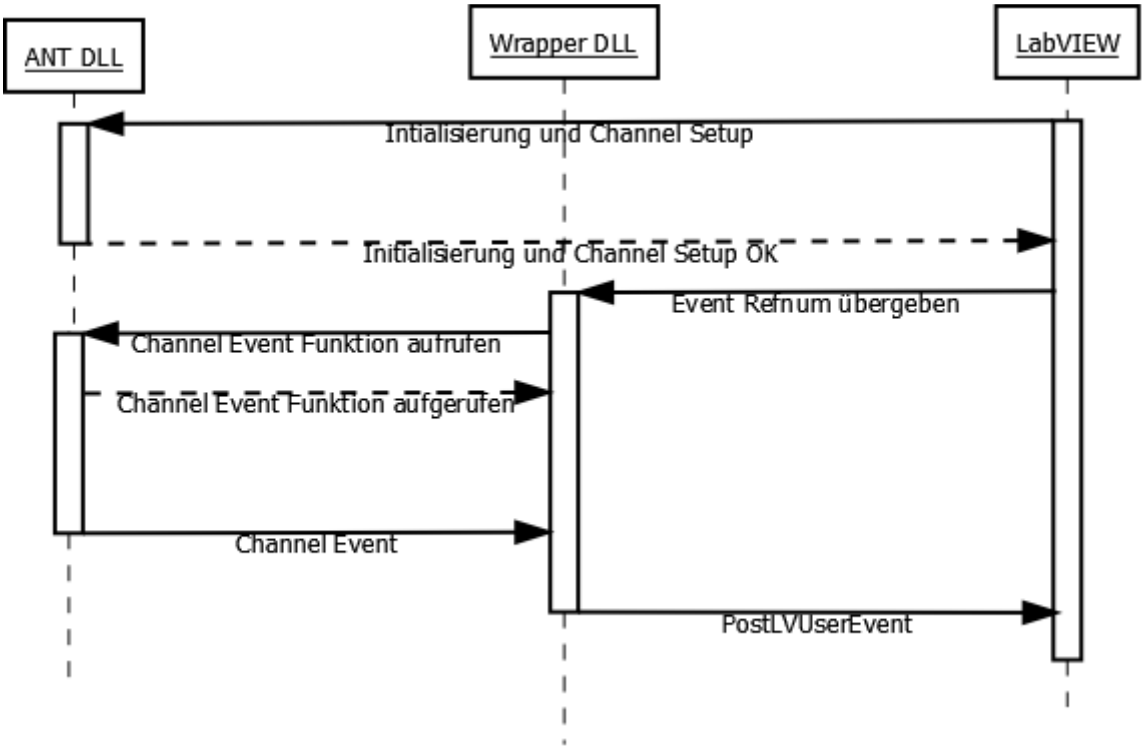


Abbildung 3: Nachrichtenfluss von der ANT DLL über die Wrapper DLL hin zu LabVIEW

Es besteht die Möglichkeit von der speziellen Implementierung auf ein allgemeines Niveau zu abstrahieren, und so die Programmierung von Schnittstellen zwischen einer DLL und LabVIEW wie hier beschrieben (bzw. in Abbildung 3 dargestellt), umzusetzen.

Die vom Autor programmierte **antwrapper.dll** (Wrapper DLL), **Eusociality_Control_Center.vi** und **Eusociality_Control_Connect.vi** unterliegen der GNU General Public License, das Copyright liegt bei Martin Böcskör. Die Nutzung und Weiterentwicklung der Software ist ausdrücklich erlaubt und erwünscht, mit der Bedingung, dass dieses Produkt weiterhin offen und frei zugänglich ist.

Zur besseren Lesbarkeit des Textes wurden **Dateien fett**, *Funktionsnamen kursiv* und `Parameter mit Courier New` geschrieben.

2. Evolution und Möglichkeiten von Computersystemen

In unserem modernen, hochtechnisierten Leben ist die elektronische Datenverarbeitung nicht mehr wegzudenken. Beinahe überall finden sich unauffällige elektronische Helfer, die den Alltag regeln, steuern und überwachen. Sei es nun im Auto, Handy, in digitalen Uhren oder Ampeln. Vor rund 70 Jahren war es undenkbar, dass in beinahe jedem Haushalt ein Computer steht, und weltübergreifend Daten über das Internet, ausgetauscht werden.

Diese Möglichkeiten verdanken wir durch zwei wesentliche Entwicklungen:

1. Der Miniaturisierung der Hardwarearchitektur
2. Die im gleichen Maße zunehmende Leistungssteigerung der Mikroprozessoren

Prozessoren werden je nach Anforderung durch folgende Faktoren optimiert:

1. Leistungsfähigkeit
2. Gewicht
3. Leistungsaufnahme

Der Fortschritt auf diesen Gebieten wird durch Abbildung 4 veranschaulicht:

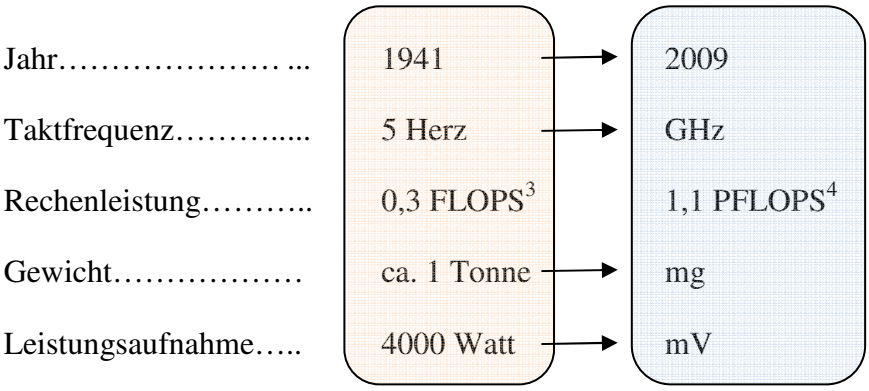


Abbildung 4: Kennzahlen von Computern 1941 - 2009

Durch diesen Fortschritt, verbunden mit der Entwicklung neuer Technologien, wie zum Beispiel die drahtlose Übertragung von Daten, konnte sich die EDV auch in Bereichen etablieren,

[3] FLOPS: Fließkomma Operationen pro Sekunde. Maßzahl für die Leistungsfähigkeit von Prozessoren.

[4] PFLOPS: Peta Flops = 10^{15} Flops

in denen miniaturisierte Elektronik eine Grundvoraussetzung ist (Medizin, Sportwissenschaft etc.).

Ein gutes Beispiel dafür ist das von IBM entwickelte System zur drahtlosen Überwachung von Herzpatienten. Abbildung 5 zeigt einen Überblick der Funktionalität des Systems. Sie zeigt auch die Interdisziplinarität zwischen der EDV und anderen Fachrichtungen wie Medizin und Sportwissenschaft.

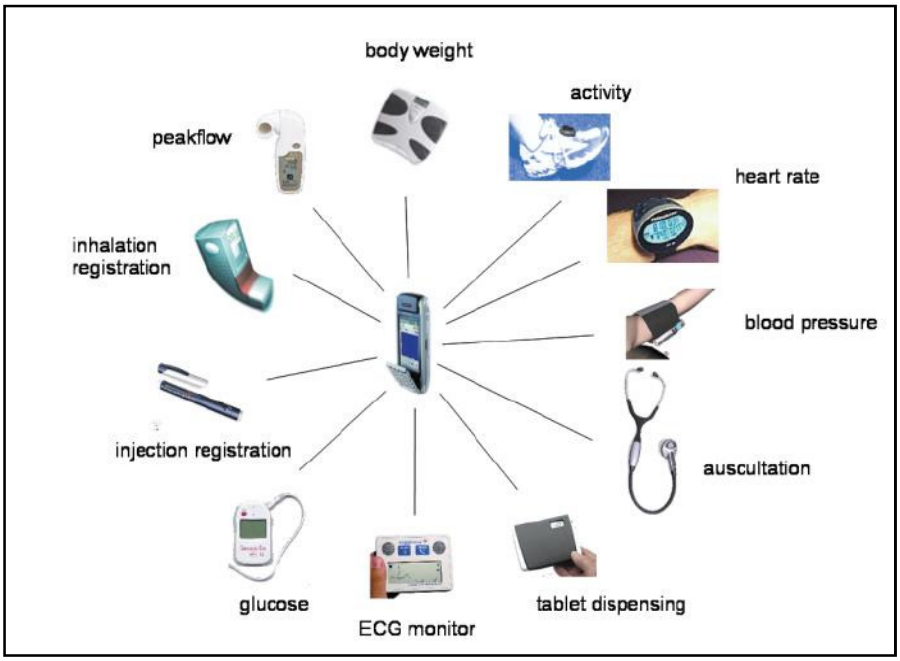


Abbildung 5: IBM mobile healthcare toolkit (Husemann & Nidd, 2005)

Im Bereich des Sports steckt ein enormes Potential für Mess- und Feedbacksysteme. In vielen Bewegungshandlungen stecken zahlreiche, im ersten Moment nicht sichtbare Informationen. Diese können zur Optimierung dieser Bewegungen beitragen.

Betrachtet man dabei zum Beispiel eine Läuferin oder einen Läufer, so kann die gemessene Zeit für eine bestimmte Strecke als ergebnisbezogenes Feedback[5] dienen.

Doch im Prozess des Laufens stecken wesentlich mehr Informationen, die sich ebenfalls als Feedback eignen. Die Fragestellung kann lauten: „Wie reagiert der Körper auf die Belas-

[5] Ergebnisbezogenes Feedback: In der Sportwissenschaft auch „Knowledge of Result (KR)“ Marschall & Daugs, 2003).

tung?“. Diese verlaufsbezogenen Informationen[6] sind für Feedback sehr interessant (Marschall & Daugs, 2003).

Abbildung 6 zeigt Beispielhaft, welche Feedbackinformationen sich für Läuferinnen und Läufer eignen.

Abbildung 6: Im Prozess des Laufens enthaltene Informationen

Einige Firmen bieten Systeme, die einige der genannten Parameter messen und aufzeichnen. Die gewonnenen Informationen werden der Läuferin oder dem Läufer während und nach dem Laufen als Feedback dargeboten.

Des Weiteren muss darauf geachtet werden, dass relevante Parameter auch in der ausreichenden Präzision bezüglich Messgenauigkeit vorhanden sind.

[6] Verlaufsbezogenes Feedback: In der Sportwissenschaft auch als „Knowledge of Performance (KP)“ bezeichnet (Marschall et al., 2003).

Eine nicht adäquate Messung der Herzfrequenz oder Laufgeschwindigkeit kann in unserem Beispiel für einen vorzeitigen Abbruch des Rennens entscheidend sein (=fehlerhaftes Feedback).

Konsequent weitergedacht, gehören zu den Anforderungen an ein Feedbacksystem unter anderem das präzise Messen der für die sportliche Technik benötigten Parameter ohne dabei die Athletin bzw. den Athleten bei der Ausübung zu stören (Baca, 2008, S. 44).

Abbildung 6 zeigt eines deutlich. Das verwendete Feedbacksystem (Pulsuhr gekoppelt mit einem Geschwindigkeitsmessgerät) ist klein und unauffällig. Das System behindert die Athletin nicht beim Laufen. Es kann als rückwirkungsfrei bezeichnet werden. Die Rückwirkungsfreiheit eines Mess- oder Feedbacksystems ist maßgeblich für die Nutzung der gemessenen Daten in Training und Wettkampf. Die Athletin bzw. der Athlet darf nicht vom Feedbacksystem beeinflusst oder gar gestört werden. Systemimmanente Eigenschaften wie die Masse oder Größe müssen diesen Anforderungen entsprechen. Eine besondere Herausforderung stellen Feedbacksysteme in Präzisionssportarten dar, in denen Sportgeräte wie z.B.: Gewehre, Schläger o.ä. eingesetzt werden.

Hier muss der Begriff Rückwirkungsfreiheit ausgedehnt werden. Das von Baca & Kornfeind (2006) für den Biathlonsport entwickelte Feedbacksystem MoTrack ist im Vergleich zu anderen Systemen rückwirkungsfrei. Einige Systeme erfordern die Anbringung zusätzlicher Gegenstände an der Waffe der Athletin bzw. des Athleten. So misst z.B.: das Feebacksystem der Firma Noptel die Trajektorie des Gewehrlaufs mittels eines Lasers. Dieser muss am Gewehrlauf befestigt werden. MoTrack benötigt keine Modifikationen am Gewehr und beeinflusst den Zielvorgang dadurch nur unwesentlich. Die Rückwirkungsfreiheit beruht darauf, dass MoTrack auf einer videobasierten Lösung zur Erfassung der Bewegung der Laufmündung beruht. Abbildung 7 die Funktionsweise von MoTrack.

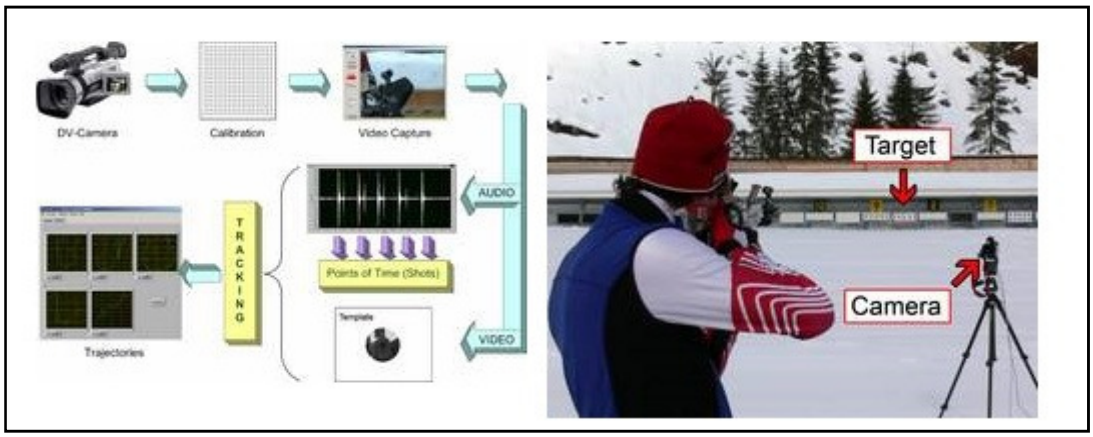


Abbildung 7: Konzept von MoTrack. Ein Feedbacksystem für den Biathlonsport (Baca et al., 2008)

Die Laufmündung des Gewehres der Athletin bzw. des Athleten wird mit einer Videokamera während des Schussvorganges (inklusive Audio) aufgenommen. MoTrack, eine LabVIEW Applikation, unterteilt das aufgenommene Video in fünf Sequenzen. Eine Sequenz repräsentiert einen Schuss. Zur Unterteilung des Videos wird die Audiospur genutzt. Überschreitet das Audiosignal einen bestimmten Pegel gilt das als Indikation für einen Schuss. Die Software analysiert mit Verfahren zur Bilderkennung, die LabVIEW bereitstellt, eine bestimmte Anzahl von Frames vor dem Triggersignal (Schuss). Mustererkennungsalgorithmen durchsuchen die Sequenzen nach einem vorher definierten Bild der Laufmündung. Danach können die X und Y Koordinaten der Laufmündung berechnet werden. An der Laufmündung selbst befindet sich eine Markierung[7] (Punkt). Dieser Referenzpunkt ermöglicht die Berechnung des Winkels der Verdrehung der Laufmündung.

[7] Die Laufmündung wird dabei mit einem wasserfesten Marker gekennzeichnet, ohne dabei die Eigenschaften der Waffe zu ändern.

In der Praxis brachte die Methode der Schusserkennung den Nachteil, dass ein Schießstand während einer Messung mit MoTrack nur von einer Person genutzt werden kann. Audioinformationen durch Schüsse anderer Athletinnen bzw. des Athleten beeinflussen den Messvorgang negativ.

Rückwirkungsfrei muss somit auch bedeuten, dass nicht nur die Athletin bzw. der Athlet, vom Feedbacksystem unbeeinflusst bleibt. Auch Umweltbedingungen dürfen das System nicht stören.

Zusätzlich zu all den genannten Kriterien ist auch die Zeit ein kritischer Parameter. Die gewonnenen Informationen müssen innerhalb definierter zeitlicher Grenzen zur Verfügung stehen und auch für spätere Analysen verfügbar sein. Der Athletin bzw. dem Athleten muss vor der Rückmeldungsinformation soviel Zeit eingeräumt werden, um das bewegungsinduzierte Feedback zu verarbeiten. Ein zu langes Intervall, welches diese intrinsischen Informationen verblassen lässt ist zu vermeiden. Eine Intervalldauer von 5 bis 10 Sekunden kann als Richtwert angenommen werden (Marschall et al., 2003). Bewegungslernen ist ein in hohem Maße individueller Prozess. Aufgrund dessen ist es sinnvoll, Feedbacksysteme flexibel zu gestalten. Trainerin bzw. der Trainer müssen über den Informationsgehalt, die Häufigkeit und Verteilung und die zeitliche Platzierung der Feedbackinformation individuell entscheiden können.

Der Informationsgehalt des Systems kann optimiert werden, indem fachkundige Trainerinnen und Trainer in die Entwicklung eines Systems mit einbezogen werden. Sie verfügen über das Wissen, welche Informationen relevant sind, und wie komplex die Darstellung erfolgen soll. Die zeitliche Unabhängigkeit wird durch eine Online Darstellung, Pufferung und Speicherung der Daten erreicht. Die Nutzerin oder der Nutzer des Mess- bzw. Feedbacksystems kann damit frei über die zeitliche Platzierung des Feedbacks entscheiden und ist unabhängig von starren, vorgegebenen Strukturen.

Um ein Mess- bzw. Feedbacksystems, das mit all den genannten Kriterien konform geht umzusetzen, ist die Verwendung einer drahtlosen, leichten Hardware notwendig. Diese muss Anschlussmöglichkeiten für geeignete Sensoren bieten und die nötigen Ressourcen[8] zur Verfügung stellen. Die Hardware wird in eine einfache Entwicklungsumgebung eingebunden, damit die tatsächliche Implementierung für Sportwissenschaftlerinnen und Sportwissenschaftler keine zusätzliche technische Herausforderung darstellt.

[8] Analoge Sensoren müssen mit einer Spannung bzw. Strom versorgt werden. Die Hardware wandelt das analoge in ein digitales Signal um (A/D Wandlung).

Die im nächsten Kapitel vorgestellten Technologien und Entwicklungswerkzeuge wurden bei dem in dieser Arbeit vorgestellten System verwendet. Sie haben sich nach sorgfältiger Analyse des Problemfeldes als interessant und auch als geeignet herausgestellt.

3. Die ANT Technologie

Wie in Kapitel 2 bereits erwähnt, werden immer leistungsfähigere, kleinere und stromsparendere Systeme auf den Markt gebracht. Doch eine andere Entwicklung hat die Verbreitung der Personal Computer maßgeblich beeinflusst. Es war die Fähigkeit mit anderen Systemen zu kommunizieren und Daten auszutauschen. Diese Eigenschaft ist eine Grundvoraussetzung für die Funktionsweise drahtloser Messsysteme. Die Art und Weise Systeme zu vernetzen wurde maßgeblich von einer Erfindung beeinflusst.

Im Oktober 1972 präsentierte Robert Kahn eine Technologie die die Kommunikation revolutionierte – das Internet, eine wichtige Entwicklung im Bereich der Informatik (Leiner, Cerf, Clark, Kahn, Kleinrock, Lynch, Postel, Roberts & Wolf, 2003). Das Internet kann als Netzwerk von Netzen bezeichnet werden. Viele unterschiedliche Systeme, die in Netzten verschiedenster Architekturen miteinander verbunden sind, können miteinander kommunizieren. Ermöglicht wird diese Eigenschaft durch die leistungsfähige TCP/IP Protokollfamilie.

Protokolle ermöglichen es, Systeme miteinander zu verbinden. Im folgenden Kapitel wird geklärt, was Protokolle sind, welche Aufgaben sie erfüllen und wie sie aufgebaut sind.

3.1 Protokolle in der Informatik

Protokolle in der Informatik definieren Regeln für bestimmte Aufgaben. „Ein Protokoll ist die Menge aller Regeln, die zu beachten sind, wenn Daten auf zuverlässige Weise vom Sender zum Empfänger übertragen werden sollen“ (vgl. Georgi et al., 2007).

An dieser Stelle sei erwähnt, dass der Begriff Protokoll oft auch als ein Synonym für einen Protokoll-Stack[9] verwendet wird. Beim ANT Protokoll handelt es sich auch um einen Protokoll-Stack, wobei die unteren Schichten vom Entwickler verborgen bleiben.

Die Vernetzung von Systemen ist aber keine triviale Aufgabe. So müssen z.B.: Regeln definiert werden, wie die Information physikalisch übertragen wird, wie und ob sichergestellt werden soll, ob die Übertragung erfolgreich war usw. Um die Komplexität dieser Herausforderung zu verringern wird das Problem in mehrere Bereiche (Schichten) unterteilt.

Ein weiteres Problem ist die Tatsache, dass sich die vernetzten Systeme in ihrer Hard- und Softwarearchitektur und der Netztopologie unterscheiden können. Aufgrund dessen wurde die

[9] Protokoll-Stack: Hierarchisch aufgebaute Kommunikationsstruktur

Kommunikation von Systemen durch die ISO[10] standardisiert. Diese Standardisierung ist im OSI[11]-Referenzmodell definiert (Cisco Systems Inc., 2000).

3.2 Das OSI-Referenzmodell

Das OSI-Referenzmodell wurde entworfen um die Kommunikation zwischen Systemen zu regeln. Das Modell unterteilt die Kommunikation in Teilbereiche. Diese Teilbereiche werden durch die sieben Sichten des OSI-Referenzmodells repräsentiert. Die einzelnen Schichten sind beschrieben, und es ist definiert, was sie zu leisten haben. Hunt (2003) hielt fest, dass das OSI-Referenzmodell eine gemeinsame Basis für die Diskussion über Kommunikation darstellt.

Abbildung 8 zeigt die Schichten des OSI-Referenzmodells und die Kommunikationsstruktur.

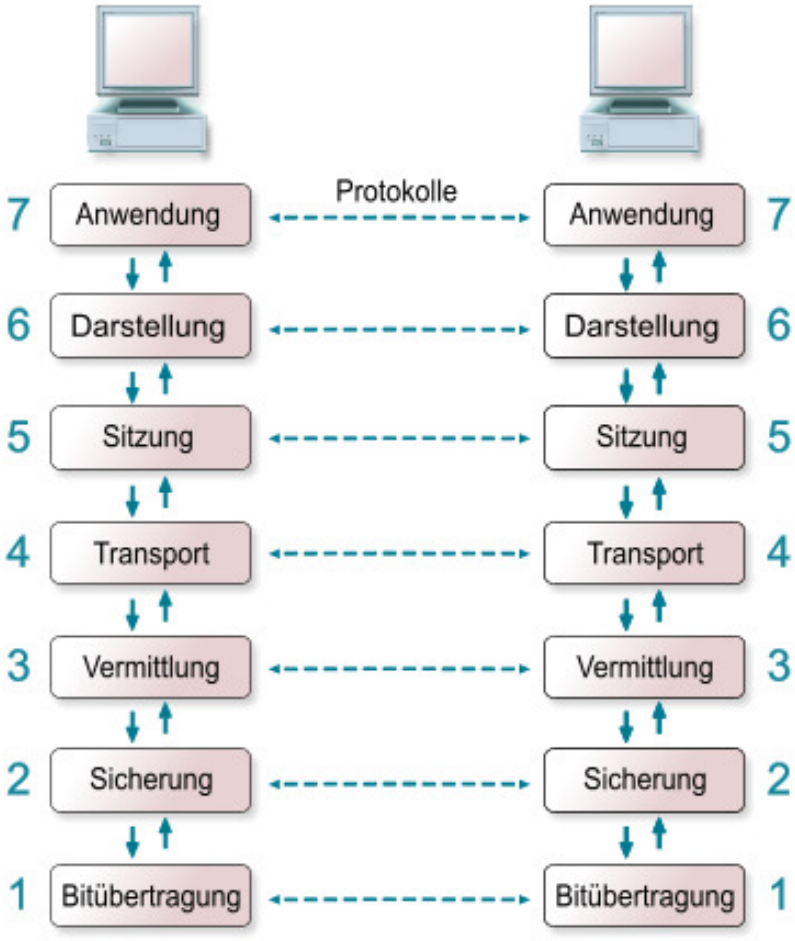


Abbildung 8: OSI Referenzmodell Schichten (Kleine, 2007)

[10] ISO: International Organization for Standardization

[11] OSI: Open System Interconnection

3.2.1 Beschreibung der Schichten des OSI Modells

Wie schon vorher erwähnt, definiert das OSI- Referenzmodell die Aufgaben der sieben Schichten. Diese Aufgaben werden nun erläutert.

Die Hardwareeigenschaften, welche zur physikalischen Übertragung des Datenstroms erforderlich sind werden in der Bitübertragungsschicht (1) definiert. Ein bekannter Standard dieser Schicht ist IEEE 802.3 (Ethernet). Dieser regelt die Datenübertragung über ein kabelgebundenes Netzwerk (LAN[12]) (Hunt, 2003).

Die Sicherungsschicht (2) kümmert sich darum, dass der Datenstrom fehlerfrei beim Empfänger ankommt. Des Weiteren gibt es in dieser Schicht Mechanismen, welche eine Überlastung des Empfängers durch zu hohe Senderaten verhindern (vgl. Holzinger, 2002).

Die Vermittlungsschicht (3) dient hauptsächlich der Übertragung von Datenpaketen. Hier erfolgt unter anderem die Wahl der Datenwege, Fehlerbehandlung und Flusskontrolle zwischen den Endpunkten einer Verbindung. Die Transportschicht (4) kümmert sich um den Transport der Daten und stellt sicher, dass diese den richtigen Empfänger erreichen (Plate, 2007).

Die Kommunikationsschicht (5) verwaltet die Verbindungen bzw. Sessions[13] zwischen zwei Kommunikationspartnern. Die Darstellungsschicht (6) definiert Regeln, wie die Kodierung und Darstellung der Informationen erfolgt (Hunt, 2003).

Die Anwendungsschicht (7) stellt Dienste bereit, die es Anwendungen ermöglichen untereinander zu kommunizieren. Mit diesen Prozessen bzw. Anwendungen kann der Benutzer auch interagieren (z.B.: ein Internetbrowser der auf den HTTP[14] Dienst zugreift) (Hunt, 2003).

3.2.2 Informationsaustausch zwischen den Schichten

Die genannten Schichten können miteinander interagieren. Dabei gibt es zwei Mechanismen. Jede Schicht stellt der darüberlegenden Operationen bzw. Dienste zur Verfügung. Der Datenaustausch von zwei gleichen Schichten wird über Protokolle geregelt (siehe Abbildung 8).

Die physikalische Verbindung zweier Systeme erfolgt ausschließlich über die Bitübertragungsschicht. Zwischen zwei gleichen Schichten besteht eine logische Verbindung.

[12] LAN: Local Area Network

[13] Session: Länger andauernde (logische) Kommunikationsverbindung zweier Partner.

[14] Hyper Text Transport Protokoll: Protokoll zur Übertragung von Daten z.B.: Webseiten.

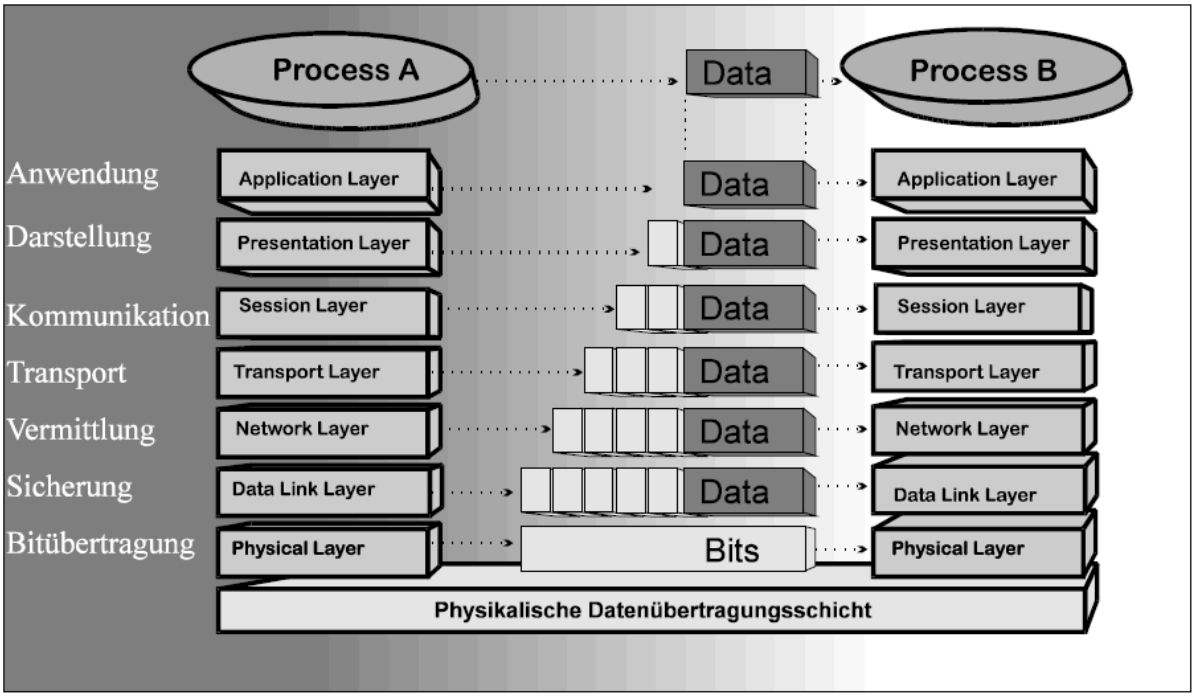


Abbildung 9: Datenaustausch mit dem OSI-Referenzmodell (Haeberlin, 1999)

Abbildung 9 zeigt wie Daten zwischen zwei Prozessen über das OSI-Referenzmodell ausgetauscht werden. Zur Erläuterung dieses Vorganges nehmen wir an, dass Prozess A ein Webserver, Prozess B ein Browser ist. Zweck der Kommunikation ist die Übertragung einer Webseite. Der Webserver gibt die Daten über HTTP an die Anwendungsschicht. Die Daten werden durch die Schichten von oben nach unten durchgereicht. Dabei versieht jede Schicht die Daten mit zusätzlichen Informationen, die für die Schicht relevant sind. Sie werden gekapselt. Bei diesem Vorgang entsteht ein zusätzlicher Overhead[15] für jede Schicht. Sind die Daten an der Bitübertragungsschicht angekommen, werden sie über ein physikalisches Medium von einem System zum anderen übertragen. Dort angekommen, werden die Daten von unten nach oben durchgereicht. Dabei werden die Daten „entpackt", bis sie über die Anwendungsschicht an die Applikation gelangen. Hier wird z.B.: die Webseite im Browser angezeigt.

3.3 Aktuell verwendete Protokoll-Stacks zur drahtlosen Datenübertragung

Der Datenaustausch zwischen Systemen erfolgt zunehmend kabellos. Technologien wie WLAN, Bluetooth oder ZigBee – um nur einige zu nennen – vernetzen Systeme indem sie de-

[15] Overhead: Informationen, die zusätzlich zu den Daten benötigt werden, um diese z.B.: wieder herzustellen usw. Ein Beispiel hierfür wären Steuerinformationen.

finierte Übertragungsfrequenzen nutzen. Im Bereich des Sports bieten solche Übertragungsmöglichkeiten viele Vorteile. Sie erlauben es, flexiblere Systeme zur Datenerhebung zu entwickeln, und reduzieren störende Einflüsse durch Verkabelung.

Diese Protokolle können teilweise durch das OSI-Referenzmodell beschrieben werden. Abbildung 10 zeigt den Aufbau des Bluetooth Protokoll-Stacks, und welche Schichten dieses Stacks denen des OSI Modells entsprechen.

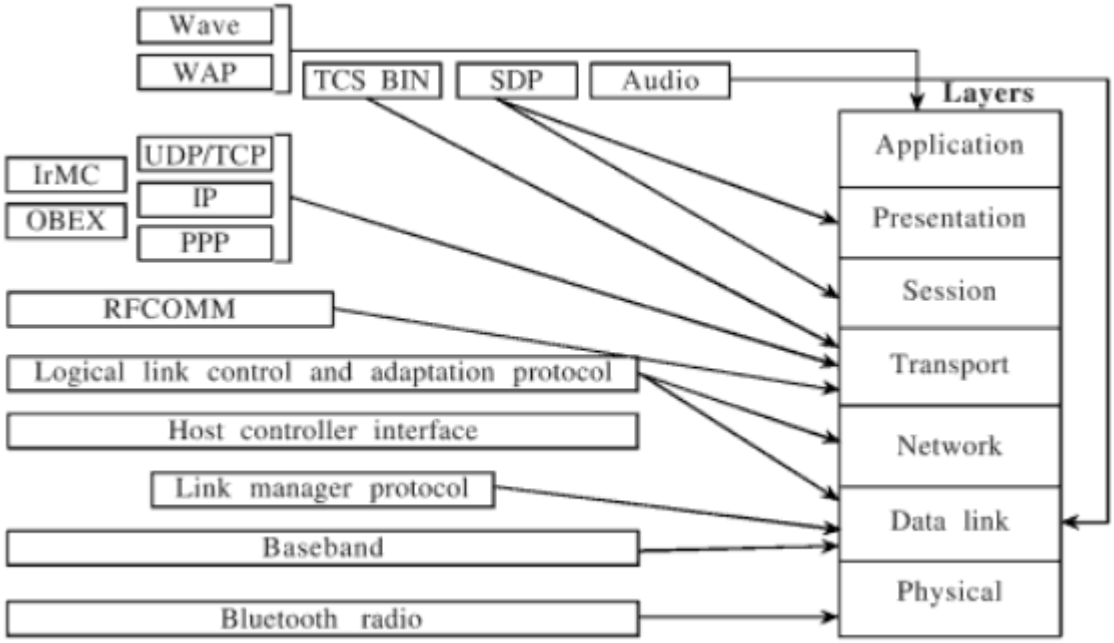


Abbildung 10: Bluetooth Stack OSI Schichten (Prabhu & Prathap Reddi, 2006)

Die grundlegenden Prinzipien des Datenflusses sind aber dieselben. Die Anwendungszwecke unterscheiden sich hingegen. Übertragungstechnologien bieten Lösungen für definierte Aufgabenbereiche. So verfügt WLAN im Vergleich zu Bluetooth oder ZigBee höhere Datenraten. ZigBee und Bluetooth bieten hier jedoch deutliche Vorteile durch niedrigeren Stromverbrauch, und geringere Hardwarekosten (Zhao & Guibas, 2004).

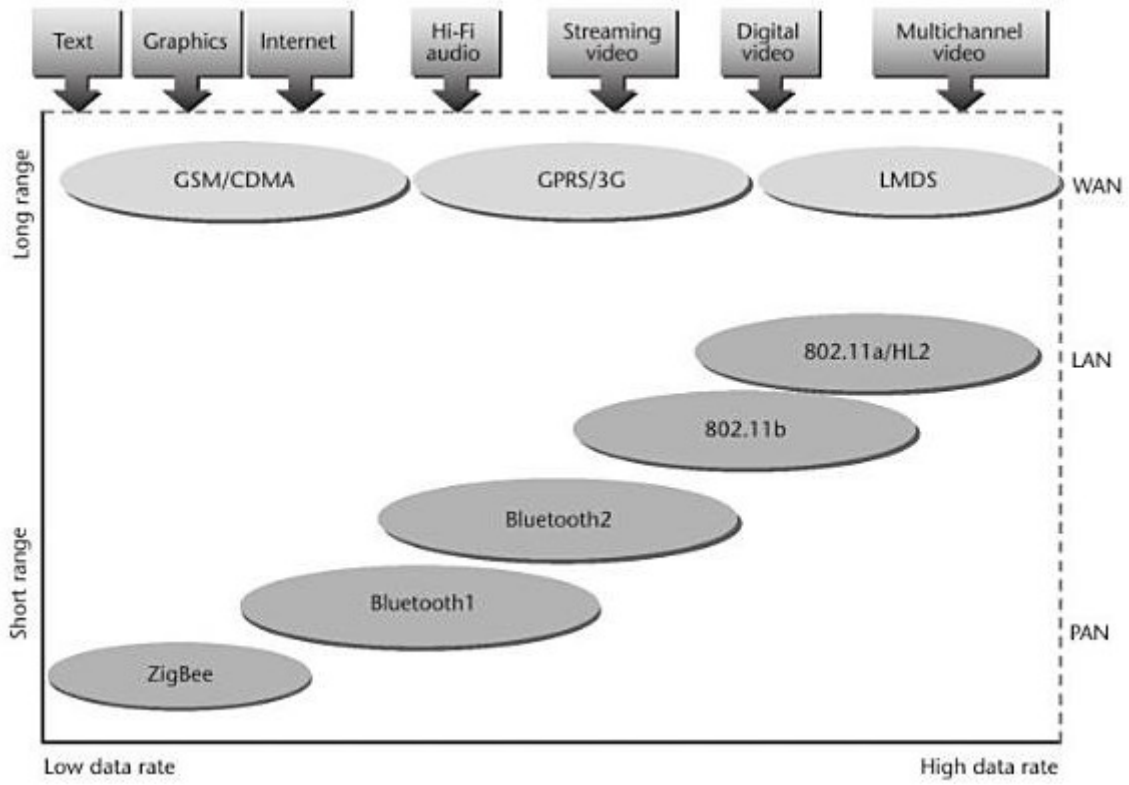


Abbildung 11: Übertragungsraten und Einsatzbereiche der verschiedenen Übertragungstechnologien (Zhao & Guibas, 2004)

Der Anwendungszweck ergibt sich aus den Eigenschaften der Technologien, wie Abbildung 11 darstellt. ZigBee und Bluetooth kommen bei Personal Area Networks zur Anwendung. Der Zweck von Personal Area Networks ist die Vernetzung von Geräten über kurze Distanzen (etwa 30 Meter) hinweg. Diese Technologien werden für Sensorlösungen auch im Anwendungsfeld des Sports eingesetzt. Ein Beispiel für eine solche Sensorlösung wäre eine Pulsuhr, welche mit einem Herzfrequenzsensor vernetzt ist. Eine Übertragungstechnologie gewinnt für sportbezogene Messsysteme zunehmend an Bedeutung – das ANT Protokoll.

3.4 Details zur ANT Technologie

> "ANT is a 2.4GHz practical wireless networking protocol and embedded system solution specifically designed for wireless sensor networks (WSN) that require:
>
> - ultra low power - runs on a coin cell for years of operation;
> - highly resource optimized - fits into a compact sized memory;
> - network flexibility and scalability - self-adaptive and able to do practical mesh,

- easy to use with low system cost - operates independently with a single chip" (ANT Wireless, 2008)

ANT ist ein Protokoll speziell für Drahtlossensor-Netzwerke. Analog zum OSI-Referenzmodell sieht das ANT Protokoll verschiedene Aufgabenbereiche vor, die nötig sind um die Daten zu übertragen.

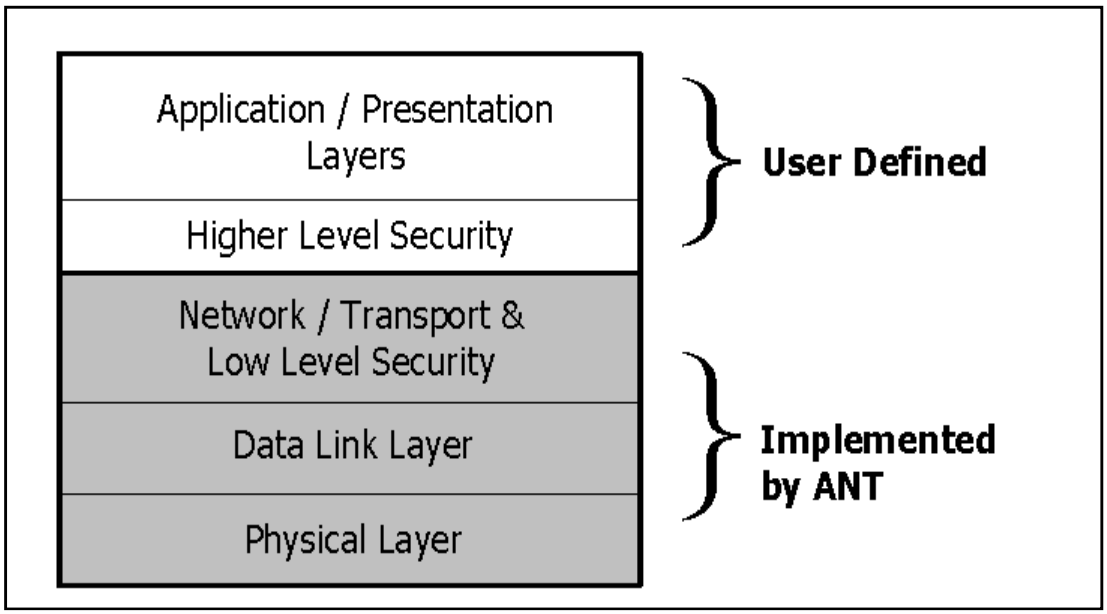


Abbildung 12: Schichten des ANT Protokolls (Dynastrean Innovations, 2008a)

ANT verbirgt die unteren Schichten vor dem Entwickler. Dieser muss sich nur um die Schnittstelle mit der Anwendung kümmern (Abbildung 12). Dies nimmt der Anwendungsentwicklerin bzw. dem Anwendungsentwickler Arbeit ab. Der große Nachteil ist die fehlende Transparenz dieses Bereiches.

Die ANT Technologie bietet dennoch für den Anwendungsentwickler eine einfache Lösung, um Daten von einem Sender zu einem Empfänger zu übertragen.

3.5 ANT Development Kit Hardware

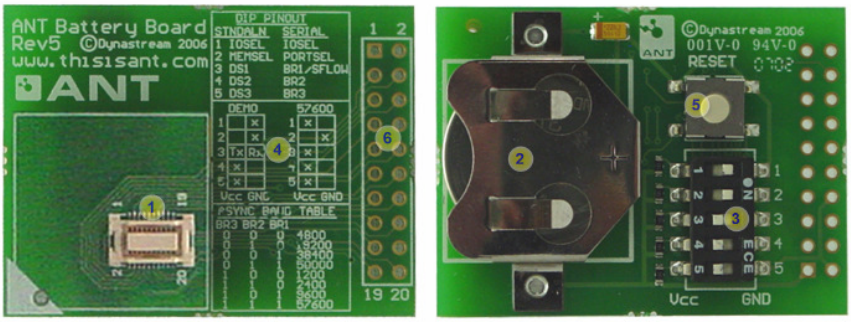
Die Firma ANT bietet zur Erstellung von Sensorlösungen ein so genanntes Development Kit an. Das Development Kit beinhaltet Hardware, die über das ANT Protokoll vernetzt werden kann. Die Kernstücke des Development Kits sind Bausteine bzw. Module die als Sender/Empfänger in einem ANT Netzwerk dienen können. Diese Bausteine verfügen über

ROM[16] Speicher. Auf diesem ist das ANT Protokoll gespeichert. Die Bausteine enthalten auch alle weiteren Bestandteile zum Aufbereiten und Übertragen der Daten. Das ANT Development Kit beinhaltet zusätzliche Hardware, um die Komponenten an den PC anzuschließen bzw. das Modul mit Strom zu versorgen, wenn es unabhängig von einem PC betrieben wird.

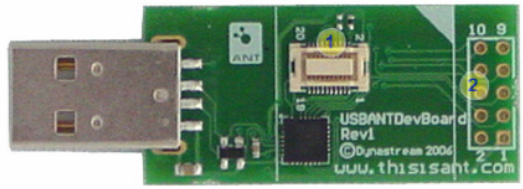
Folgende Komponenten sind im Development Kit enthalten:

- 2xANT Battery Board
- 2xUSB Interface Board
- 2xANT AP1M5IB Modul
- 2xANT 11TS33M5IB (SensRcore) Modul
- Treiber, Benutzerhandbuch und Applikations CD

ANT Battery Board



USB Interface Board



[16] ROM: Read only memory – Speicher auf/in den einmal Informationen geschrieben werden können. Danach ist nur noch ein Lesen der Information möglich z.B.: CDs oder DVDs.

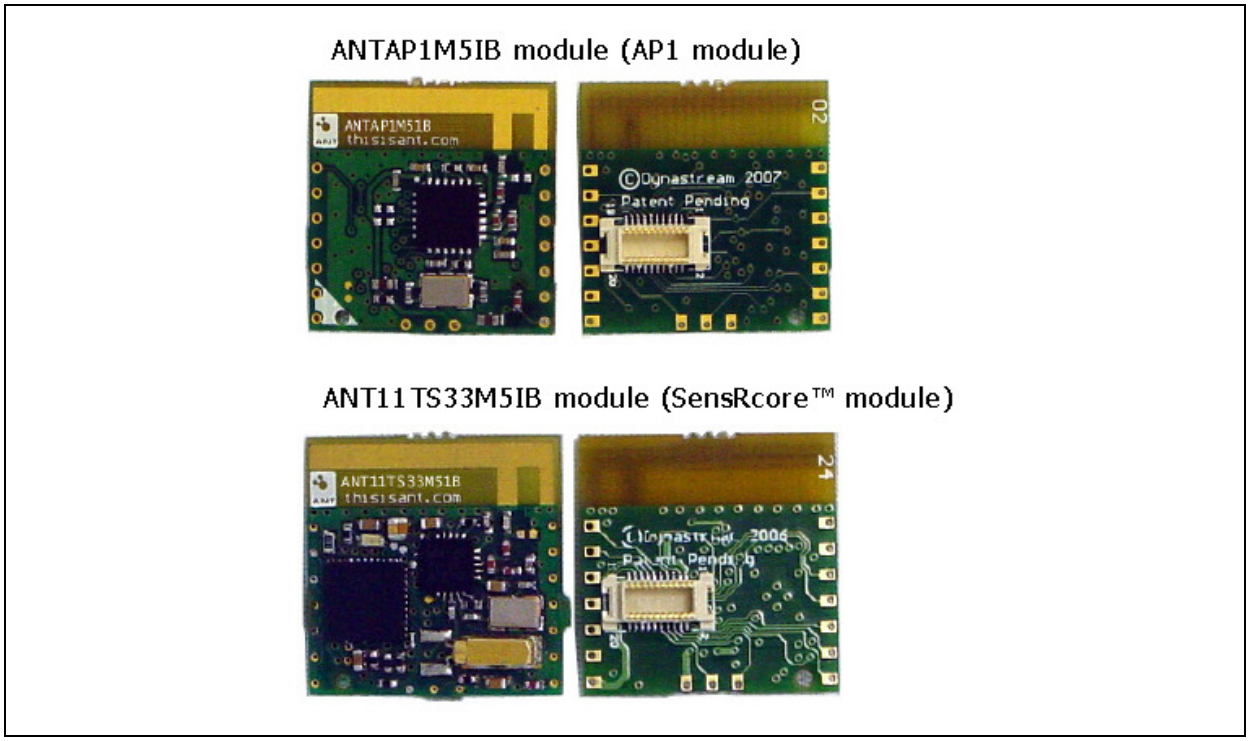


Abbildung 13: Hardwarebestandteile des ANT Development Kits, bestehend aus dem Battery Board, dem USB Interface und den ANT Modulen (Dynastream Innovations Inc., 2008c)

3.6 Spezifikationen des ANT SensRcore Moduls

Die Umsetzung einer Idee in Form eines Programms ist an Voraussetzungen gekoppelt. Die eingesetzte Hardware muss diese erfüllen. Die wichtigsten dieser Voraussetzungen werden in diesem Kapitel erörtert.

Viele Sensoren benötigen, wie schon erwähnt Ressourcen. Das Battery Board kann diese in Form von 3 Volt Versorgungsspannung bereitstellen (CR 2032 Knopfzelle). Das ANT11TS33M5IB Modul arbeitet innerhalb eines Spannungsbereichs von 1.8 Volt bis 3.6 Volt. Als Referenzspannung[17] kann zwischen 2,5 Volt oder 1,5 Volt gewählt werden (Dynastream Innovations Inc., 2008b).

Die maximale Abtastfrequenz liegt bei 500 Hz. Analoge Daten werden mit einer Genauigkeit von 10 Bit aufgelöst. Die Repräsentation erfolgt als 16 Bit Wert. Zur Datenübertragung steht eine Bandbreite von 20kbps zur Verfügung.

[17] Referenzspannung: Die Referenzspannung legt die maximale Spannung für das System fest. Wird 1,5 Volt gewählt, kann der Bereich von 0 bis 1,5 Volt mit der Auflösung von 10 Bit digitalisiert werden. Das Spannungssignal des analogen Sensors soll diesen Bereich bestmöglich angepasst werden.

3.7 Erklärung der SensRcore Technologie

Neben der Übertragungsmöglichkeit bietet ANT eine weitere innovative Technologie namens SensRcore. Sie ermöglicht es, Signale von Sensoren mit einem Minimum an Aufwand zu messen und aufzubereiten. Die SensRcore Technologie ist direkt in bestimmten ANT Modulen integriert.

Abbildung 13 zeigt die unterschiedlichen ANT Module. Das SensRcore Modul verfügt über einen Mikrocontroller[18], der es möglich macht, sowohl analoge als auch digitale Sensoren anzuschließen (siehe Kapitel 6.2.4).

Dieser Mikrocontroller ist programmierbar, somit können z.B. Abtastraten oder Details zur A/D-Wandlung entsprechend konfiguriert werden. Die Software des ANT Development Kits macht diesen Schritt sehr einfach, Kapitel 3.6 befasst sich detailliert mit diesem Vorgang. Auf der, dieser Arbeit beiliegenden CD befindet sich eine Textdatei, welche die Konfiguration enthält. Diese Datei ist für die später vorgestellte Applikation geeignet. Die PDF Datei **ANTDKT3 Development Kit User Manual Rev1.3.pdf**, welche ebenfalls auf der beigelegten CD zu finden ist, bietet eine detaillierte Anleitung des Herstellers.

3.8 Konfiguration der ANT Module im Überblick

Damit ein Modul im SensRcore Modus betrieben werden kann, muss es über eine Konfiguration verfügen. Diese Konfiguration kann über das Programm SensRware.exe erstellt werden. Anschließend wird die erstellte Datei mit dem Programm Antware.exe in den NVM[19] Speicher des ANT Moduls geschrieben. Anschließend ist das SensRcore Modul betriebsbereit. Die Konfiguration wird automatisch geladen, sobald das Modul mit Strom versorgt wird (z.B.: mit dem Batteryboard). Abbildung 14 Zeigt schemenhaft sowohl das Vorgehen, als auch die am Prozess beteiligten Programme.

In den folgenden Kapiteln wird das Konfigurieren des ANT SensRcore Modul erläutert.

[18] Mikrocontroller: Ein Prozessor mit Schnittstellen zur Kommunikation (Schwarz, 2008).

[19] NVM (Non Volatile Memory): NVM ist ein nichtflüchtiger Speicher. Die Informationen bleiben auch erhalten, wenn der Speicher nicht mit Strom versorg wird. z.B.: Flash Memory (USB Stick) (Datacom, 2008).

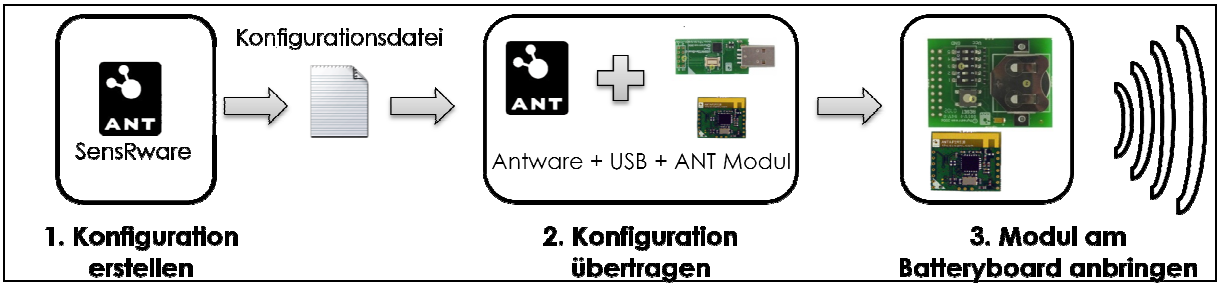


Abbildung 14: Konfiguration des ANT SensRcore Moduls im Überblick

3.9 Erstellen einer SensRcore Konfiguration

Die Konfigurationsdatei wird mit Hilfe des Programms **Sensrware.exe** erstellt. Diese Datei besteht aus Nachrichten, wie sie im ANT Protokoll definiert sind. Im linken Fensterbereich des Programms **Sensrware.exe** (Abbildung 15) werden alle Einstellungen bezüglich der Signalerfassung getätigt.

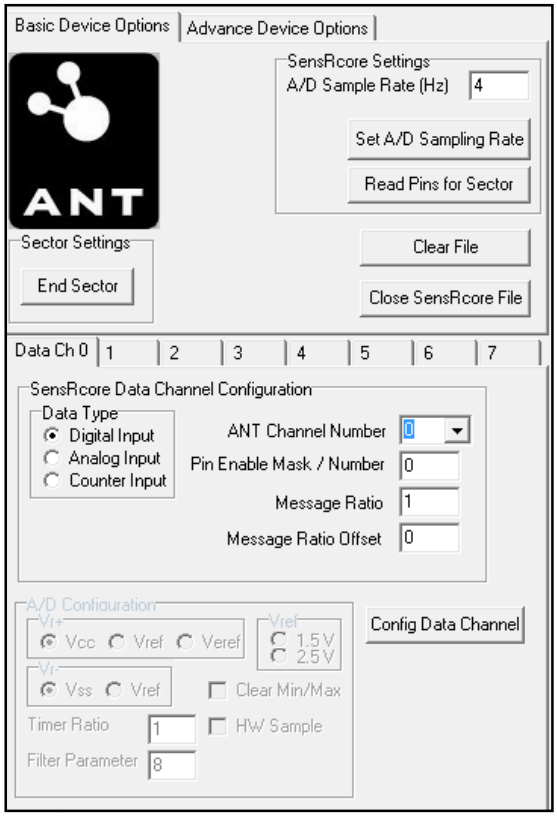


Abbildung 15: Einstellung der Parameter zur Signalerfassung mit SensRware

Im rechten Fensterbereich (Abbildung 16) wird der bzw. werden die Übertragungskanäle konfiguriert. Der Empfänger muss die gleiche „Mandatory Channel Configuration[20]“ und die gleiche Übertragungsfrequenz wie der Sender besitzen. Der Kanal muss zum Schluss des Vorganges geöffnet werden.

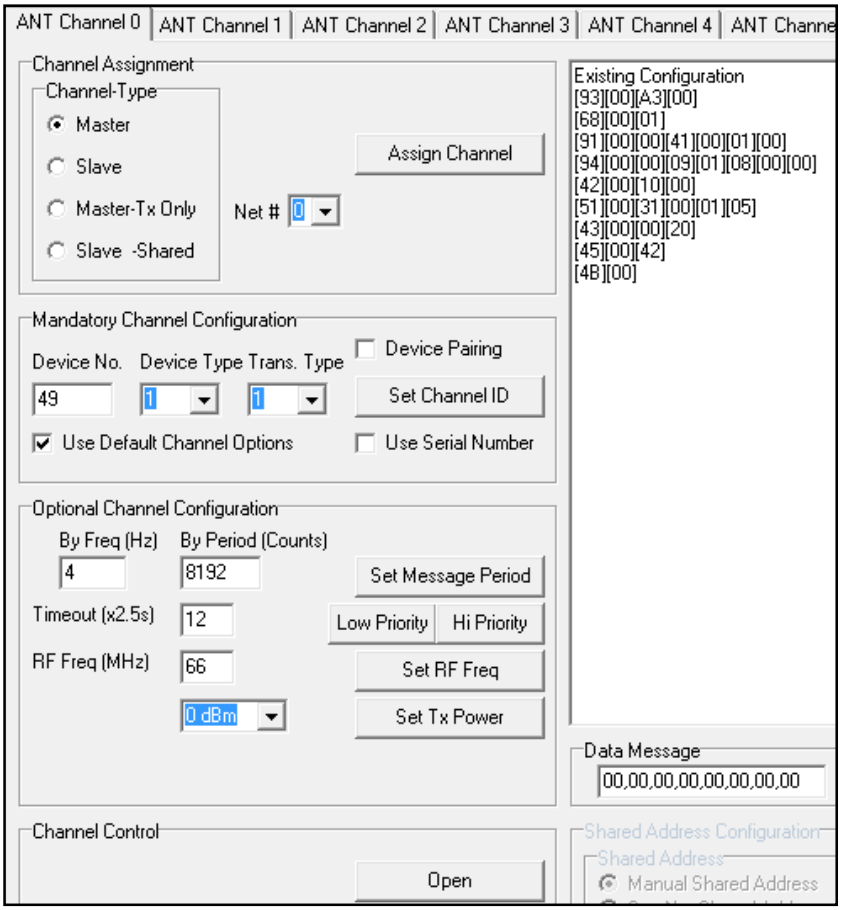


Abbildung 16: Einstellung und Konfiguration des Übertragungskanals mit SensRware

SensRware erzeugt eine Textdatei (.txt), welche die eingegebenen Parameter in eine Reihe von Befehlen enthält, wie sie im ANT Protokoll definiert sind.

C [93][00][A3][00] // Setting A/D Sample Rate at 200 Hz

C [68][00][01] // Enabling LED on RF events

…

[20] Mandatory Channel Configuration: Für die Receiver Node können alle Parameter auf 0 gesetzt werden. Somit werden alle ANT Geräte in Reichweite gesucht. Zu Verwendung des ANT Sport+ Netzwerks gibt es vordefinierte Gerätetypen. Weitere Informationen unter www.thisisant.com

Diese Konfigurationsdatei muss abschließend auf das SensRcore Modul geladen werden. Das Modul wird dazu auf das USB[21] Interface Board gesteckt und per USB mit dem PC verbunden. Mit Hilfe der Applikation **ANTware.exe** wird eine Verbindung zum Modul hergestellt. Die Baudrate[22] ist für das SensRcore Modul 57600 bps. Bei nicht SensRcore Modulen maximal 50000 bps. Nach dem Herstellen der Verbindung, wird der Flash Speicher des Moduls mit Clear NVM[23] gelöscht. Mit Load NVM wird das Konfigurationsfile hochgeladen (Abbildung 17).

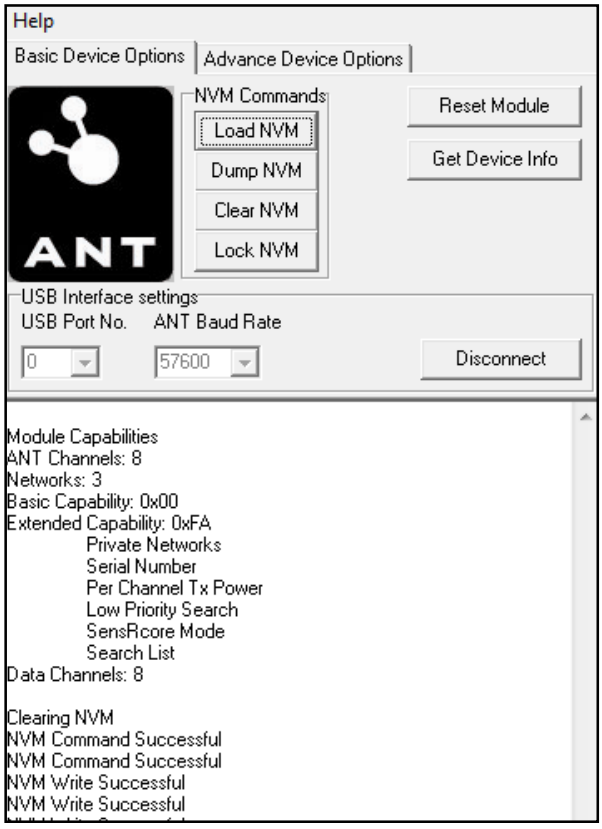


Abbildung 17: Hochladen einer Konfigurationsdatei mit dem Programm Antware

[21] USB: Universal Serial Bus: Schnittstelle zum Verbinden von externen Geräten mit dem Computer.

[22] Baudrate: Die Baudrate oder Symbolrate gibt an, wie oft ein Signal pro Zeiteinheit seinen Zustand ändern kann. Die Übertragungsgeschwindigkeit kann wie folgt errechnet werden: Baudrate*Bits pro Baud.

[23] NVM (Non Volatile Memory): NVM ist ein nichtflüchtiger Speicher. Die Informationen bleiben auch erhalten, wenn der Speicher nicht mit Strom versorg wird. z.B.: Flash Memory (USB Stick) (Datacom, 2008).

Das ANT Modul ist nach der Konfiguration einsatzbereit und kann auf das ANT Battery Board gesteckt werden. Hier übernimmt es die vorgesehene Aufgabe entsprechend der Konfiguration.

4. Die Entwicklungsumgebung LabVIEW

Die ANT Technologie erleichtert die Entwicklungen von Messsystemen. Die Verwendung der ANT Technologie beseitigt viele Schwierigkeiten im Bereich des Hardwaredesigns. Das Messsystem muss aber auch eine Verbindung zwischen der Hardware und der Benutzerin bzw. dem Benutzer herstellen. Diese muss so gestaltet werden, dass sie für die Anwenderin bzw. dem Anwender einfach und intuitiv bedienbar ist. Die Schnittstelle muss alle Anwendungsfälle beinhalten, die für die Benutzerin bzw. dem Benutzer notwendig sind sowie alle vereinbarten Anwendungsszenarien abdecken (siehe Abbildung 2). Dazu muss die Entwicklerin bzw. der Entwickler diese Aufgabenstellungen in einer Sprache formulieren, die der Computer verstehen kann. Durch eine Programmiersprache kann eine solche Aufgabenstellung in einer für Mensch und Maschine verständlichen Sprache formuliert und realisiert werden kann.

Das Softwarepaket LabVIEW[24] stell der Entwicklerin bzw. dem Entwickler eine graphische Programmiersprache zu Verfügung, welche es ermöglicht, Anwendungen durch das Platzieren von Elementen auf einer Oberfläche zu erzeugen. Das Erstellen von Programmen basiert auf Programmablaufplänen oder Flussdiagrammen. Programme werden vielmehr gezeichnet als geschrieben (vgl. Georgi et al., 2007).

[24] Laboratory Virtual Instrument Engeneering Workbench. Graphische Programmiersprache von National Instruments.

Um den Unterschied zwischen einer textbasierende Programmiersprache und LabVIEW zu veranschaulichen, wird im Folgenden der Code eines einfachen Programms zur Addition zweier Zahlen gezeigt. Abbildung 18 zeigt die Programmierung des Beispiels in C++.

```
1   #inlcude <iostream>
2   using namespace std;
3
4   int main(){
5       int a, b;
6       cout << "Bitte geben Sie Zahl 1 ein: ";
7       cin >> a;
8       cout << "Bitte geben Sie Zahl 2 ein:";
9       cin >> b;
10      cout << "Das Ergebnis ist: " << a+b << endl;
11  return 0;
12  }
```

Abbildung 18: Der Code eines Programms zur Addition zweier Zahlen in C++

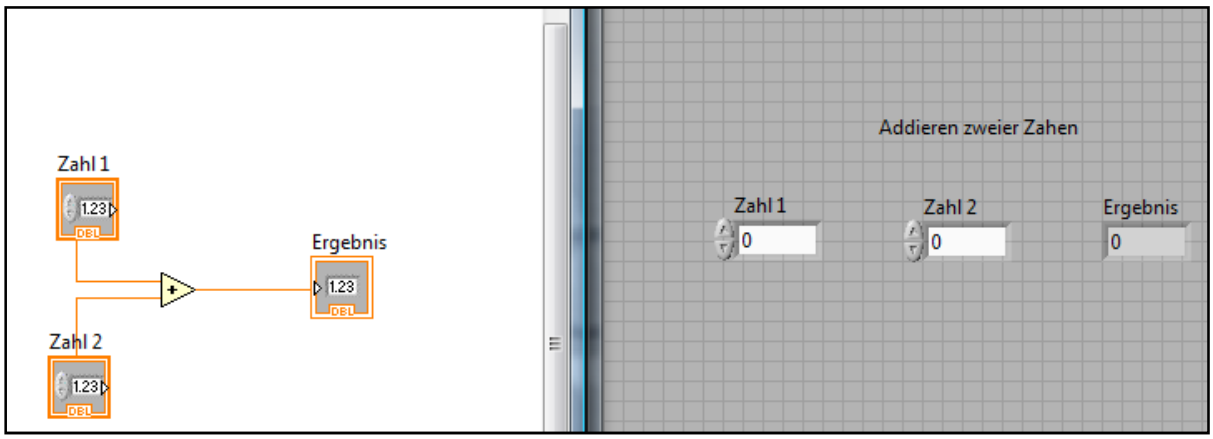


Abbildung 19: Der Code eines Programms zur Addition zweier Zahlen in LabVIEW

Die beiden vorangegangenen Illustrationen lassen gut erahnen, wie sehr sich die Herangehensweisen unterscheiden. Der C++ Code ist eine Applikation für die Kommandozeile. Ein GUI[25] ist nicht implementiert. Es gibt hierfür sehr gute Tools wie Visual Studio C++ für Windows, oder auch KDevelop für Linux. Mit ihnen lassen sich Anwendungen in gewohnter „Fenster“ Form entwickeln. Die Programmlogik ist bei C++ Schreibarbeit, in einer Sprache, die gelernt werden muss. Im Falle von LabVIEW beschränkt sich die Programmierung der

[25] GUI: Graphical User Interface - Graphische Benutzerschnittstelle wie z.B.: in Windows geläufig.

Addition zweier Zahlen auf das Platzieren von Elementen auf dem Frontpanel[26] (Abbildung 19 rechts). Die Umsetzung der Programmlogik erfolgt im Blockdiagramm[27] durch das Verbinden der Zahlen mit dem Additionsoperator (Abbildung 19 links).

LabVIEW bietet im Vergleich zu „textbasierenden" Programmiersprachen Vorteile. Die graphische Programmierung gestaltet sich intuitiv. LabVIEW macht das Programmieren einfacher, ersetzt es jedoch nicht. LabVIEW Programme (als exe) benötigen eine Run Time Engine (RTE)[28] um zu funktionieren. Die LabVIEW RTE wird kostenlos angeboten. Dennoch darf nicht außer Acht gelassen werden, dass LabVIEW eine proprietäre[29] Programmiersprache ist. C++ hingegen ist eine standardisierte Sprache, für die es freie Compiler[30] gibt.

Es kann festgehalten werden, dass Sportwissenschaftlerinnen bzw. Sportwissenschaftler mit LabVIEW eine geeignete Entwicklungsumgebung zur Verfügung steht, mit der sich Anwendungen im Bereich der Messtechnik realisieren lassen.

[26] Frontpanel in LabVIEW: Für den Anwender sichtbarer Bereich des Programms.

[27] Blockdiagramm in LabVIEW: Entspricht dem Quellcode des Programms. Hier erfolgt die Umsetzung der Algorithmen und der Programmlogik.

[28] Run Time Engine: Eine RTE stellt Bibliotheken und Funktionen für Programme bereit. Die LabVIEW RTE erlaubt es Anwendungen ohne einer LabVIEW Installation auszuführen.

[29] Proprietär: Der Begriff proprietär wird in der Informatik für nicht freie Software oder Protokolle verwendet. Als Beispiel hierfür seien OpenOffice und Microsoft Office genannt. OpenOffice ist frei. Microsoft Office ist proprietäre Software.

[30] Compiler: Ein Compiler übersetzt ein Programm der Sprache X in eines der Sprache Y. Beide Programme haben das Selbe Ein/Ausgabeverhalten. Meistens werden Compiler genutzt um Programme einer Hochsprache wie C++ in Maschinensprache zu übersetzen.

4.1 Nutzung des ANT Protokolls in LabVIEW

In den vorangegangenen Kapiteln wurde erläutert, wie das ANT Protokoll und allgemeine Protokoll-Stacks aufgebaut sind. Sie besitzen Schichten, welche miteinander kommunizieren können. Ein Programm kann dabei die Anwendungsschicht nutzen, um Verbindungen zu initiieren oder zu beenden, und Daten zu senden oder zu empfangen.

Das ANT Development Kit beinhaltet eine Datei (DLL), um die Anwendungsschicht zu nutzen. Diese Befehlssammlung ermöglicht die Kommunikation zwischen der Applikation, dem ANT Protokoll und der ANT Hardware. Folgend wird beschrieben, wie die Funktionen der ANT DLL in LabVIEW nutzbar gemacht werden können.

Die Verwendung der DLL in LabVIEW erfolgt über den „Call Library Node“ (Abbildung 20). Diese Routine lädt Funktionen, die die DLL exportiert und übergibt gegebenenfalls Parameter an die Funktion. Der „Call Library Node“ ist der Werkzeugpalette des Blockdiagramms, unter Konnektivität – Knoten zum Aufruf externer Bibliotheken zu finden.

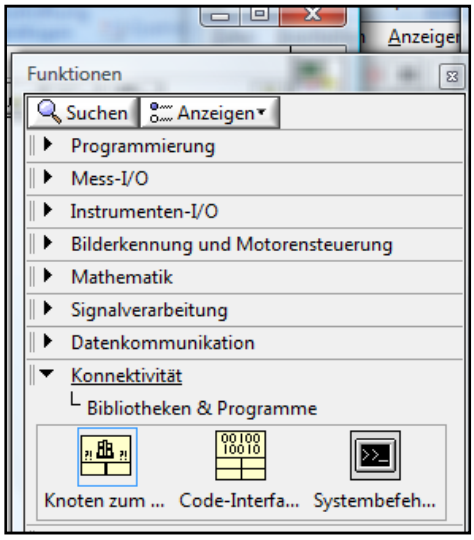


Abbildung 20: LabVIEW Werkzeugpalette - Call Library Node

Dieses Element kann per Drag & Drop auf das Blockdiagramm platziert werden. Durch einen Doppelklick auf dieses Element öffnet sich ein Konfigurationsdialog, wie Abbildung 21 zeigt.

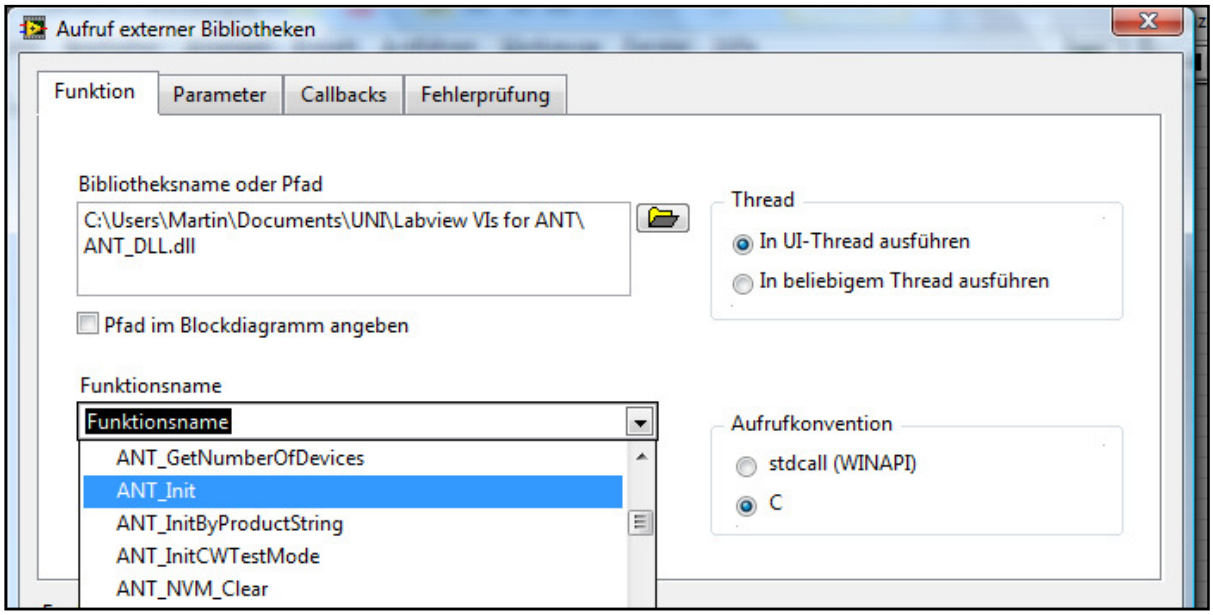


Abbildung 21: LabVIEW Call Library Node Konfigurationsdialog

Funktion, Parameter und Aufrufkonvention[31] können hier angegeben werden.

LabVIEW kann Funktionen aus DLLs importieren. Die importierten Funktionen stehen der Entwicklerin bzw. dem Entwickler als fertige VIs[32] zur Verfügung. Für diesen Vorgang wird zusätzlich zur DLL die Header Datei[33] benötigt.

Um die Bibliothek zu importieren, kann in LabVIEW im Menü „Werkzeuge – Importieren – Shared Library (dll)“ (Abbildung 22) ein Dialog aufgerufen werden. Dieser führt Schritt für Schritt den Import durch.

[31] Aufrufkonvention: Die Aufrufkonvention definiert Eigenschaften zum Funktionsaufruf. So regelt die Konvention in welcher Reihenfolge Funktionsparameter übergeben werden, und wer die Aufräumarbeiten am Stack übernimmt.

[32] VI (Virtual Instrument): LabVIEW Programme werden als VI bezeichnet. Sie stellen den Quellcode dar. Durch das Kompilieren sind diese als eine ausführbare Datei (Exe oder Binary) nutzbar.

[33] Header Datei: In der Header Datei sind die Funktionsprototypen definiert. Die geläufige Dateiendung ist .h

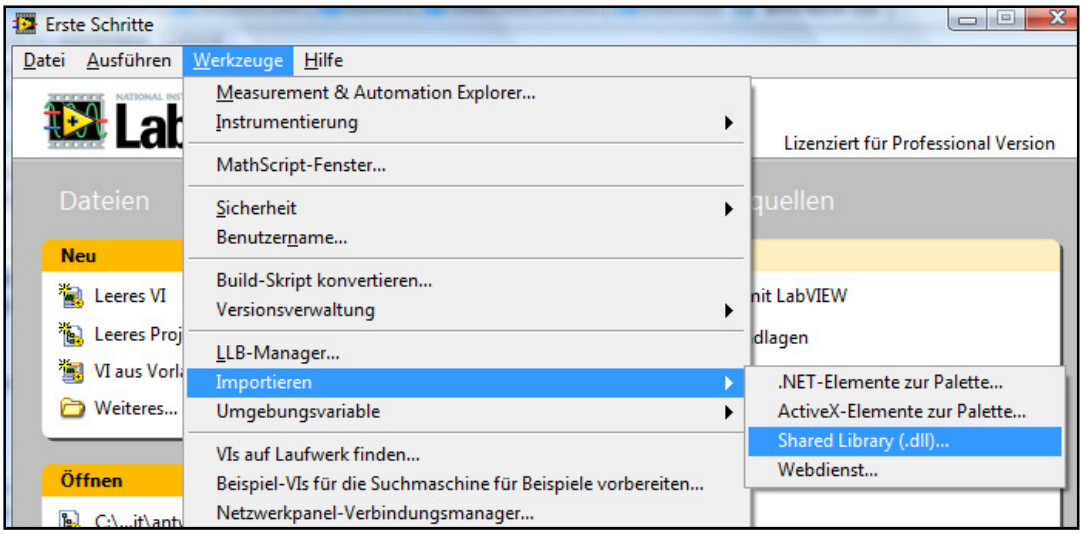


Abbildung 22: Importieren einer DLL in LabVIEW - Schritt 1

LabVIEW muss angewiesen werden, aus den Funktionen, die die DLL exportiert, VIs zu erzeugen (Abbildung 23).

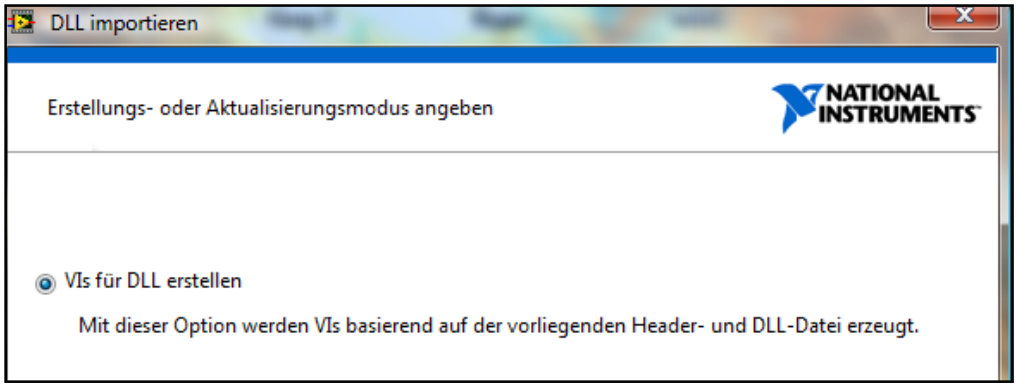


Abbildung 23: Importieren einer DLL in LabVIEW - Schritt 2

Der Speicherort der DLL und der Header Datei kann wie in Abbildung 24 definiert werden.

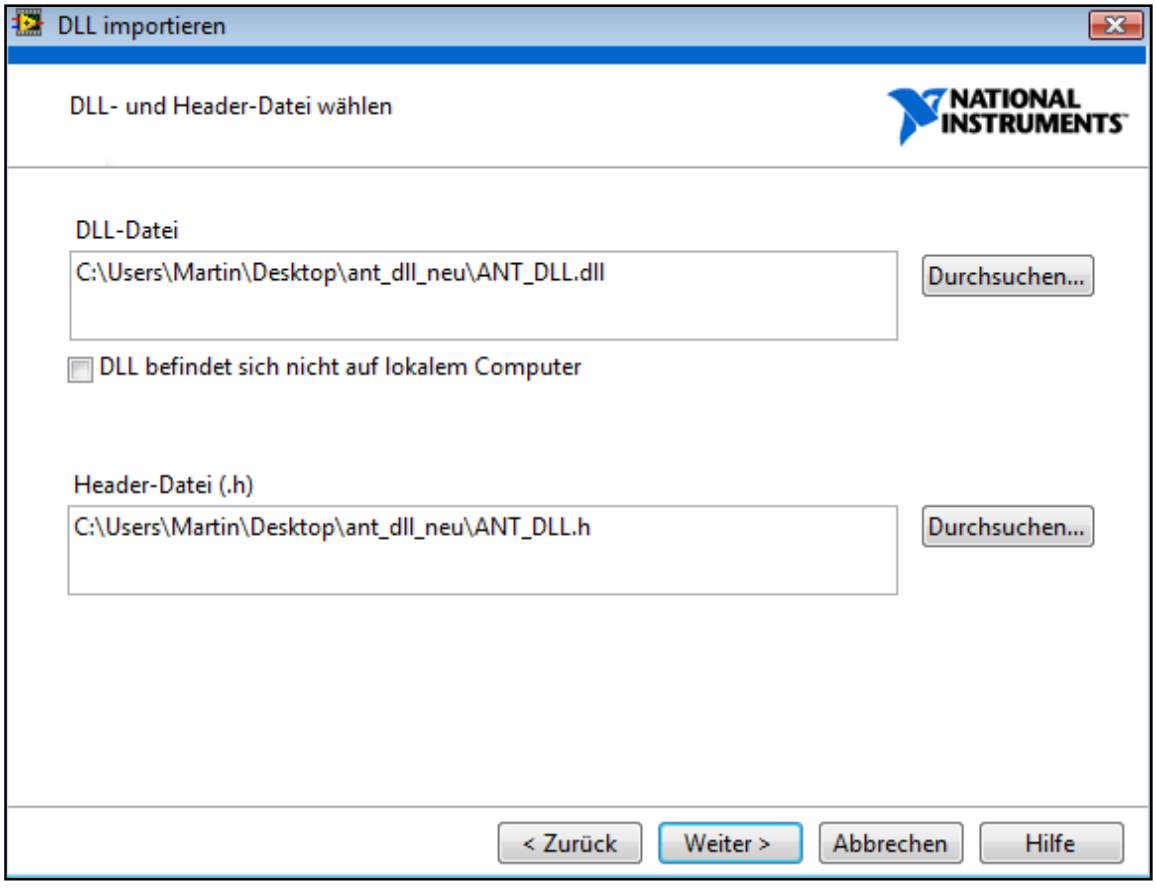


Abbildung 24: Importieren einer DLL in LabVIEW - Schritt 3

Einzuschließende Pfade müssen nicht extra angegeben werden. Das Ändern der Standardeinstellungen ist nicht nötig.

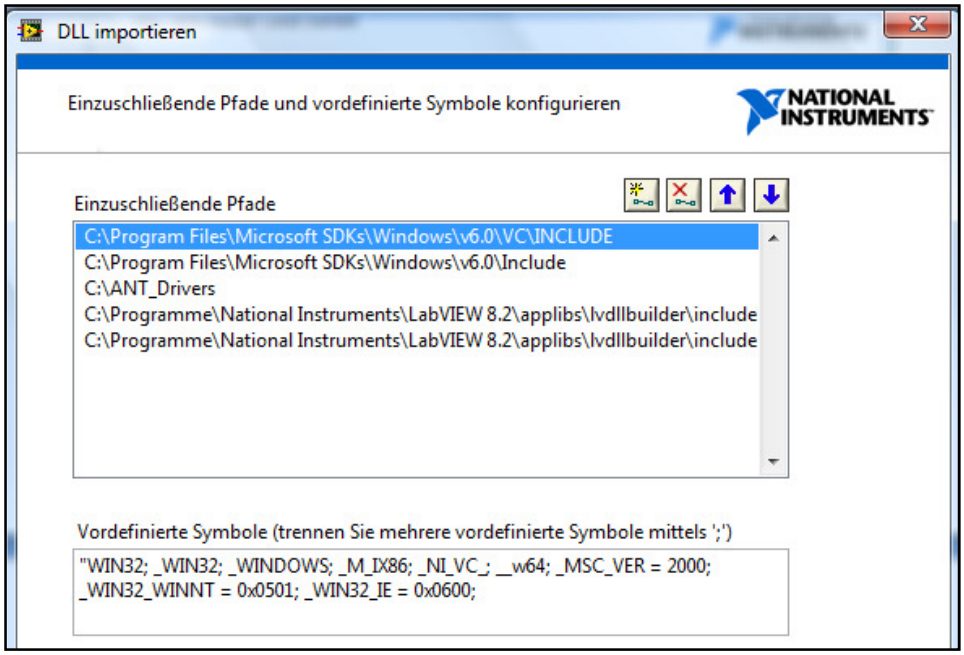


Abbildung 25: Importieren einer DLL in LabVIEW - Schritt 4

Nach der Analyse der DLL meldet LabVIEW aufgetretene Fehler. Wie Abbildung 26 zeigt, gab es in der Header Datei keinen passenden Deklarationen für 2 Funktionen. Diese Funktionen werden im weiteren Verlauf nicht benötigt.

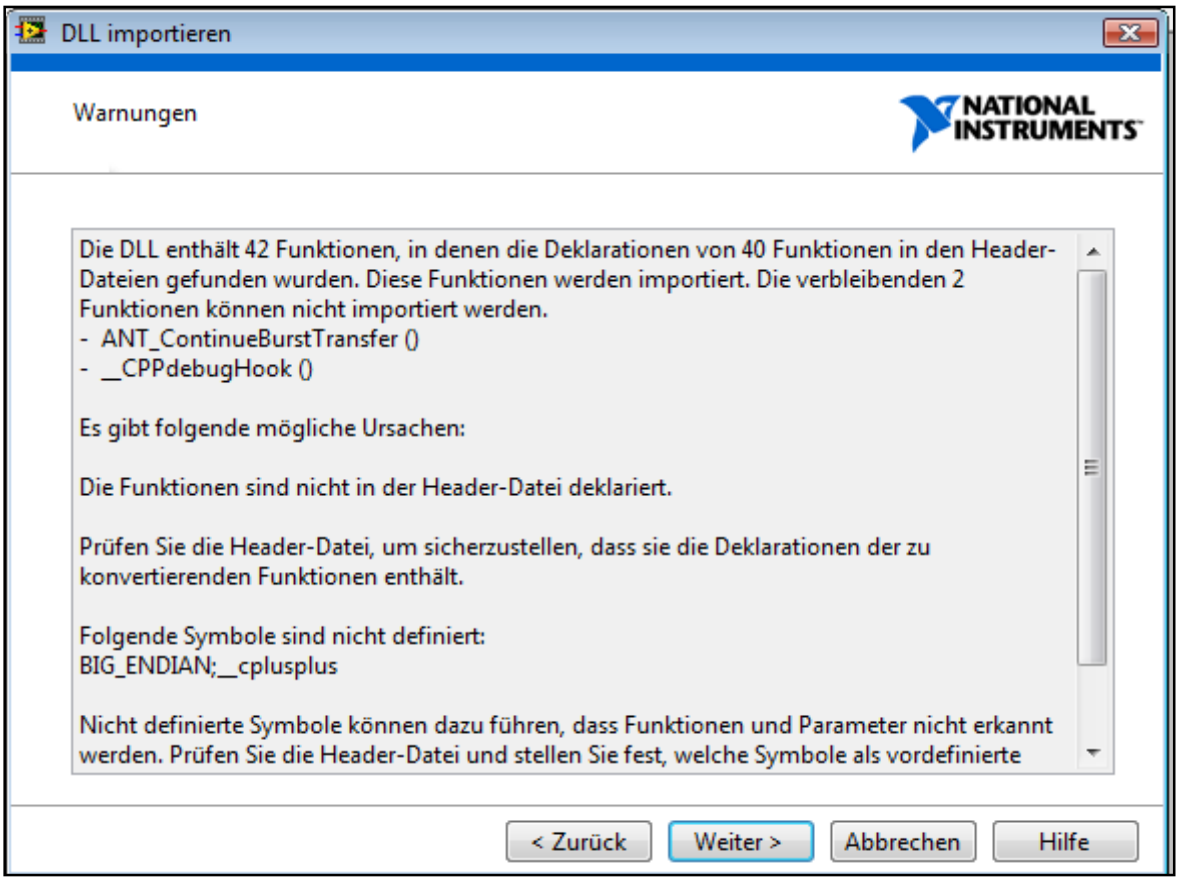


Abbildung 26: Importieren einer DLL in LabVIEW - Schritt 5

Abschließend werden, wie in Abbildung 27 dargestellt, die zu importierenden Funktionen ausgewählt und der Speicherort für die erzeugten VIs angegeben. Die Dateien **ANT_DLL.dll**, **SiUSBXp.dll** und **SiUSB.dll** müssen im gleichen Verzeichnis wie die VIs liegen. Diese DLLs liegen dem ANT Development Kit bei. Diese Dateien sind zwingend nötig, und müssen, sollte das Programm (z.B.: als Binary/Exe) weitergegeben werden, ebenfalls im Programmordner enthalten sein.

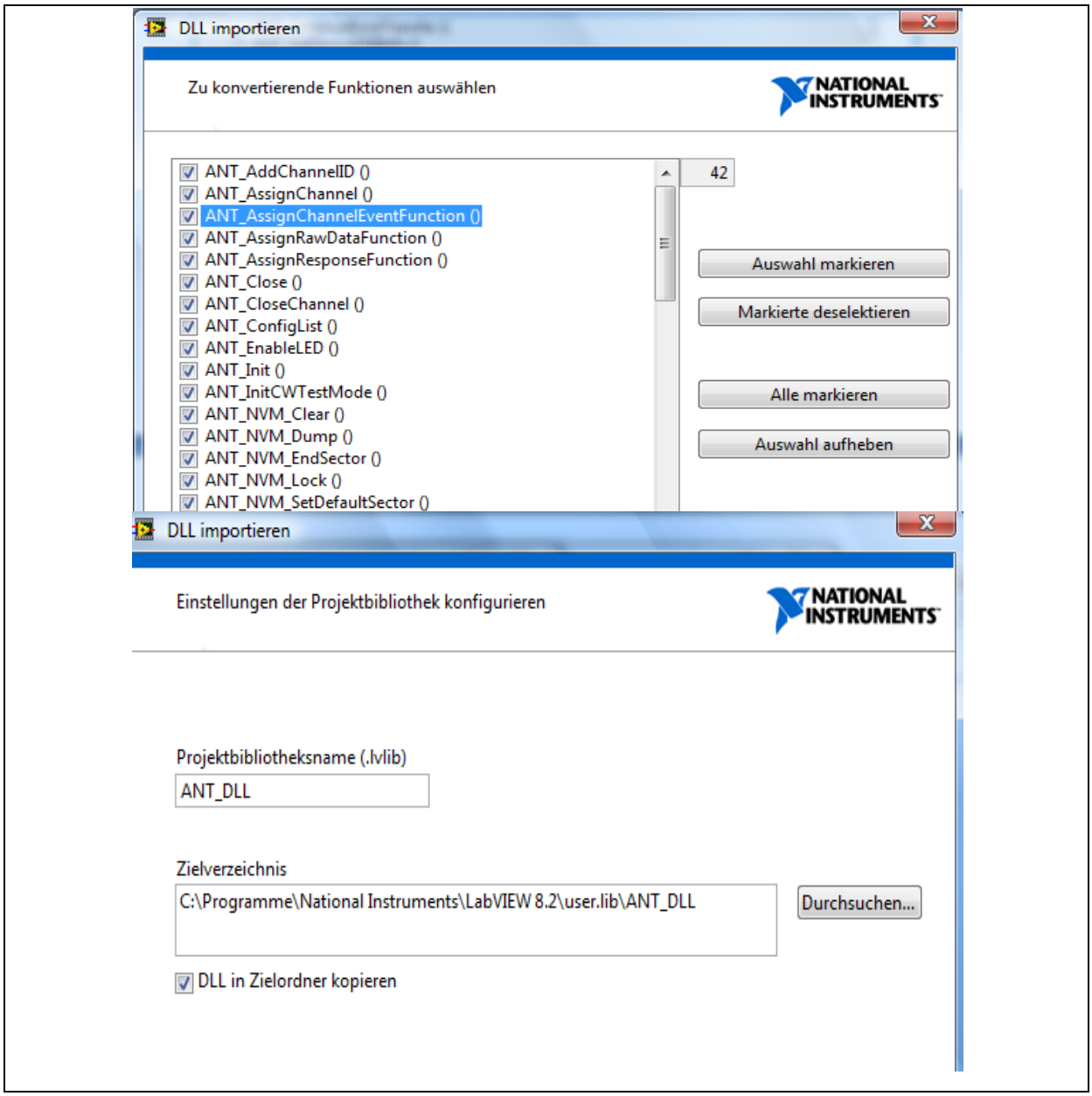


Abbildung 27: Importieren einer DLL in LabVIEW - Schritt 6

Abbildung 28 zeigt das Konfigurieren der Fehlerhandhabung. Die einfache Fehlerhandhabung bietet genug Raum zur Implementierung von Fehlercodeausgaben.

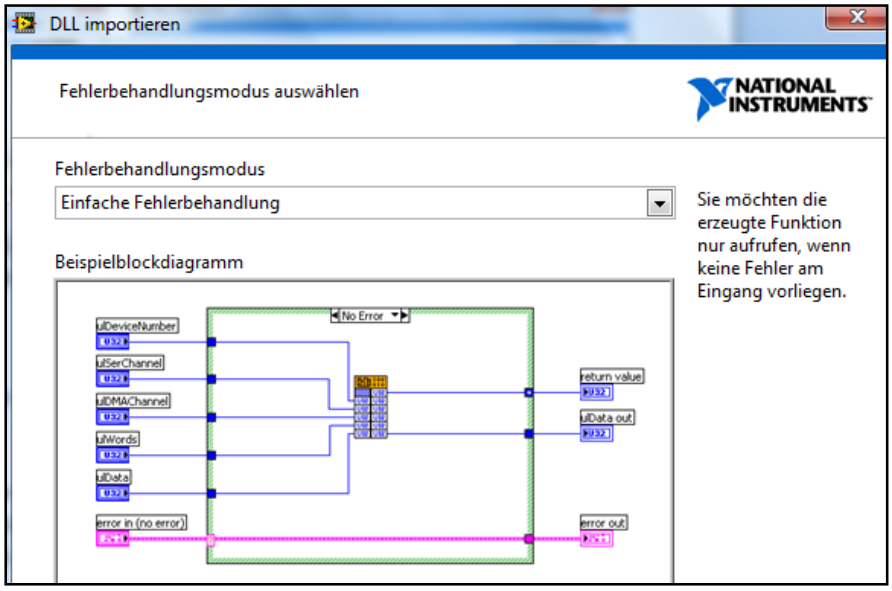


Abbildung 28: Importieren einer DLL in LabVIEW - Schritt 7

Der folgende Dialog dient zur Konfiguration der erzeugten VIs. Namen und Datentypen der Funktionsparameter können hier geändert werden. Das ist aber nicht notwendig, da LabVIEW diese der Funktionsdeklaration entnimmt. Danach ist der Import abgeschlossen und die Funktionen stehen als VIs zur Verfügung.

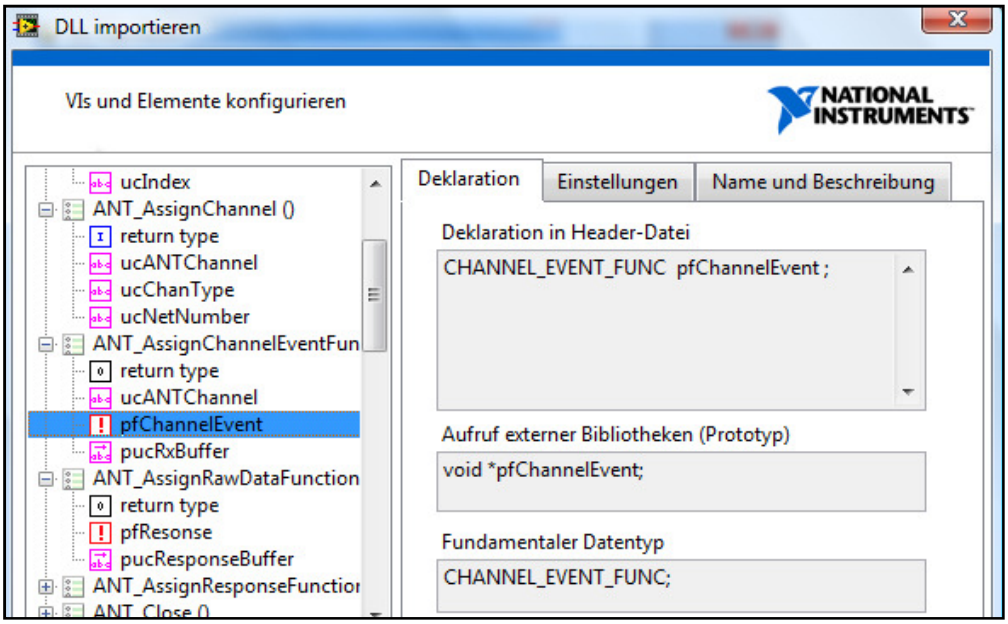


Abbildung 29: Importieren einer DLL in LabVIEW - Schritt 8

In LabVIEW ist es nicht möglich alle Funktionen der ANT DLL zu importieren. Das Rufzeichen vor dem Parameter `pfChannelEvent` (Abbildung 29) weist auf einen Importfehler hin. LabVIEW konnte keinen passenden Datentypen für dieses Element finden. Auch in anderen Funktionen ist das gleiche Problem erkennbar.

Um zu verstehen wie es dazu kommt, ist ein Blick auf die Beschreibung der betroffenen Funktionen notwendig. Diese Funktionen sind sich ähnlich, deshalb wird beispielhaft die Funktion *ANT_AssignChannelEventFunction* erläutert.

> "ANT_AssignChannelEventFunction sets the channel event function and the return data buffer. The callback function and data buffer are used whenever an event message is received from ANT for the given channel. The response buffer needs to be large enough to hold an incoming response which is of size MESG_DATA_SIZE. This function must be called to set up a given channel before other ANT functions that use this channel are called.
>
> The channel event callback function must be a C function. Each channel can have its own event callback function along with a unique data buffer or they can both be shared or any combination thereof that best suits the application. "(Dynastream Innovations Inc, 2008a, S. 67)

Der Prototyp der Funktion sieht wie folgt aus:

`void` *ANT_AssignChannelEventFunction*(UCHAR `ucChannel`,

CHANNEL_EVENT_FUNC `pfChannelEvent`, UCHAR `*pucRxBuffer`);

Parameters	Type	Description
ucChannel	UCHAR	Channel Number
pfChannelEvent	CHANNEL_EVENT_FUNC	Pointer to the function that will be called whenever an event for this channel occurs.
pucResponseBuffer	UCHAR*	Pointer to the buffer where the data of the response/event message is written. This buffer should be sized to MESG_DATA_SIZE.

```
// Example Usage
BOOL ANT_ChannelEventFunction(UCHAR ucChannel, UCHAR ucEvent);
UCHAR aucChannelEventBuffer[MESG_DATA_SIZE];
.
.
ANT_AssignChannelEventFunction(channel_0, &ANT_ChannelEventFunction, aucChannelEventBuffer);
```

Abbildung 30: Definition "ANT_AssignChannelEventFunction" (modifiziert nach Dynastream Innovations Inc., 2008a)

Die Funktion liefert void[34] als Ergebnis zurück. Danach folgt der Name der Funktion: „*ANT_AssignChannelEventFunction*“. Die Funktion hat 3 Parameter (Abbildung 30). Zwei davon sind vom Datentyp[35] UCHAR[36]. Eine Variable hat einen besonderen Datentyp (CHANNEL_EVENT_FUNC). Die Namen der Parameter sind variabel. Anstelle von `ucChannel` kann auch eine Variable mit dem Namen `ANTKanal` an die Funktion übergeben werden. Der Datentyp und die Reihenfolge müssen übereinstimmen.

Die Funktion übernimmt mehrere Aufgaben. *ANT_AssignChannelEventFunction* gibt einen Bereich im Arbeitsspeicher bekannt. Der Parameter `pucRxBuffer` ist ein Zeiger[37] auf diesen Bereich. `ucChannel` gibt an, für welchen Übertragungskanal die Funktion zuständig ist. Werden Daten von verschiedenen Sensoren empfangen, benötigt jeder Datenstrom seinen eigenen Übertragungskanal.

Beim Import der DLL ist im Zuge der Arbeit ein Problem mit dem Parameter `pfChannelEvent` aufgetaucht.

`pfChannelEvent` ist ein Zeiger auf eine Funktion, die immer dann aufgerufen werden soll, wenn Daten auf dem Kanal anliegen, der über `ucChannel` definiert wurde. *ANT_AssignChannelEventFunction* benötigt einen Zeiger auf eine C Funktion als Argument. LabVIEW kann jedoch keine C Funktionen bereitstellen.

Diese Funktion wird immer bei einem bestimmten Ereignis aufgerufen. Deshalb auch der Name C*hannelEventFunction*.

[34] Void: Void ist ein Datentyp. Dieser Datentyp wird als Rückgabewert bei Funktionen eingesetzt, die keinen Wert als Ergebnis liefern.

[35] Datentyp: Der Datentyp definiert wie den Wertebereich eine Variable (Böhm, 2006).

[36] UCHAR: UCHAR ist ein 8 Bit Datentyp. Der Wertebereich ist 0 – 255.

[37] Zeiger: „Ein Zeiger ist nichts anderes als eine Variable, die die Adresse einer anderen Variable enthält.“ (Böhm, 2006, S. 140)

5. Programmierung der Schnittstelle zwischen der ANT DLL und LabVIEW

Die *ANT_AssingChannelEventFunction* benötigt demnach eine C Funktion, die definiert, was sie zu tun hat. (Abbildung 31)

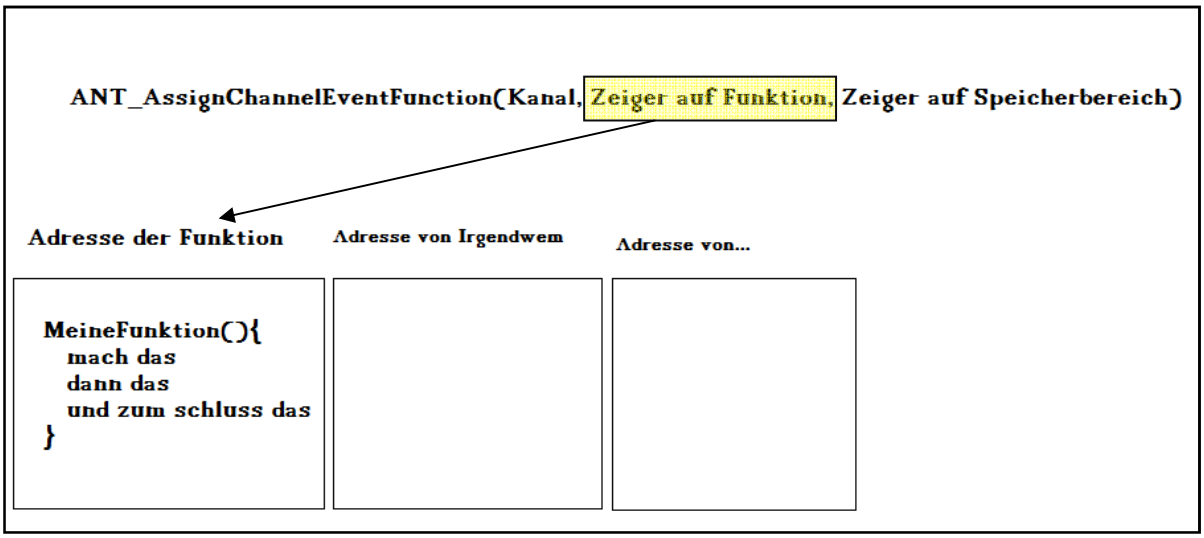


Abbildung 31: Schemenhafte Darstellung der Funtkion ANT_AssignChannelEventFunction

LabVIEW kann diese aber nicht bereitstellen. Ohne diese Funktion ist nicht definiert, was mit den Daten, welche empfangen (bzw. auch gesendet) werden passieren soll, wie Abbildung 32 zeigt.

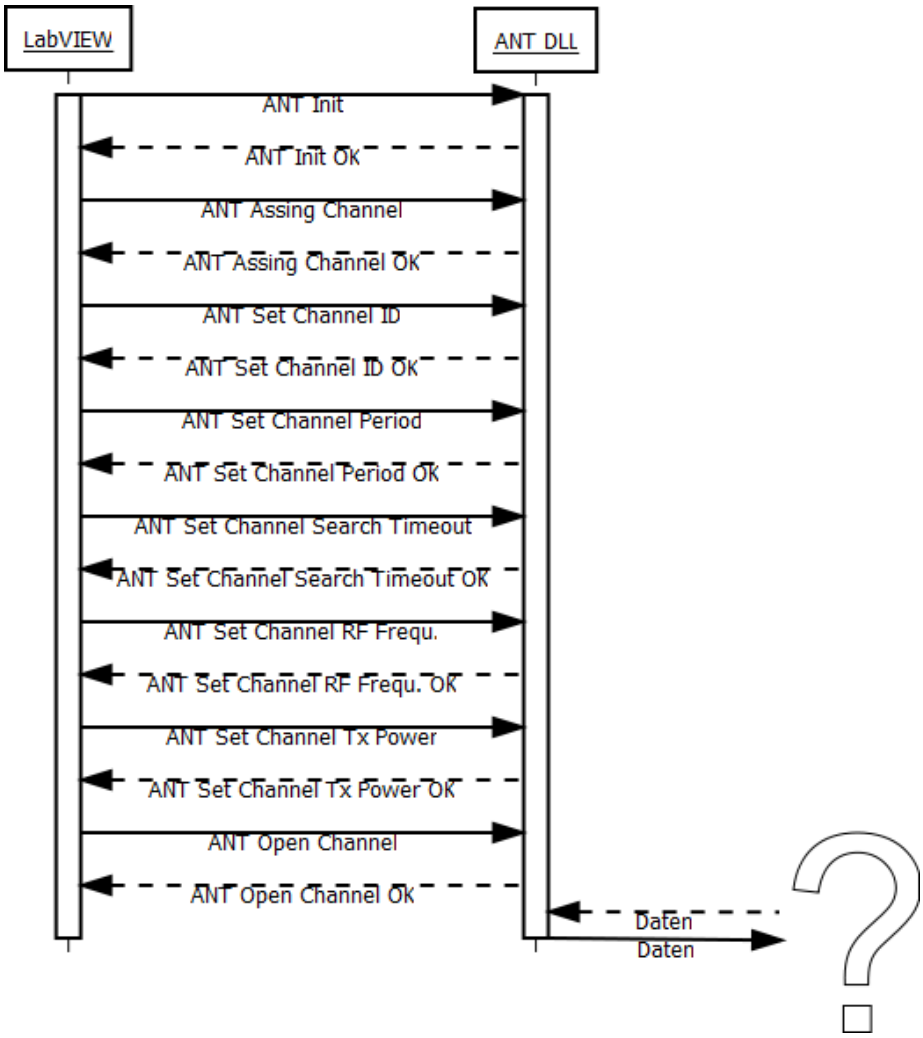


Abbildung 32: Kommunikation zwischen LabVIEW und der ANT DLL ohne Wrapper

Es ist daher notwendig, eine DLL zu schreiben, um *ANT_AssingChannelEventFunction* mit diesen Informationen zu versorgen, damit sie ihrer Arbeit nachgehen kann. Diese DLL (bzw. ihre Funktionen) muss dafür sorgen, dass LabVIEW immer dann informiert wird, wenn etwas am entsprechenden Übertragungskanal passiert. Diese DLL ist eine Schnittstelle zwischen der ANT DLL und LabVIEW. Abbildung 33 zeigt die Kommunikation zwischen der ANT DLL und LabVIEW mit Hilfe dieser Wrapper[38] DLL.

[38] Wrapper: Der Begriff Wrapper bezeichnet in der EDV eine Schnittstelle zwischen zwei Programmen.

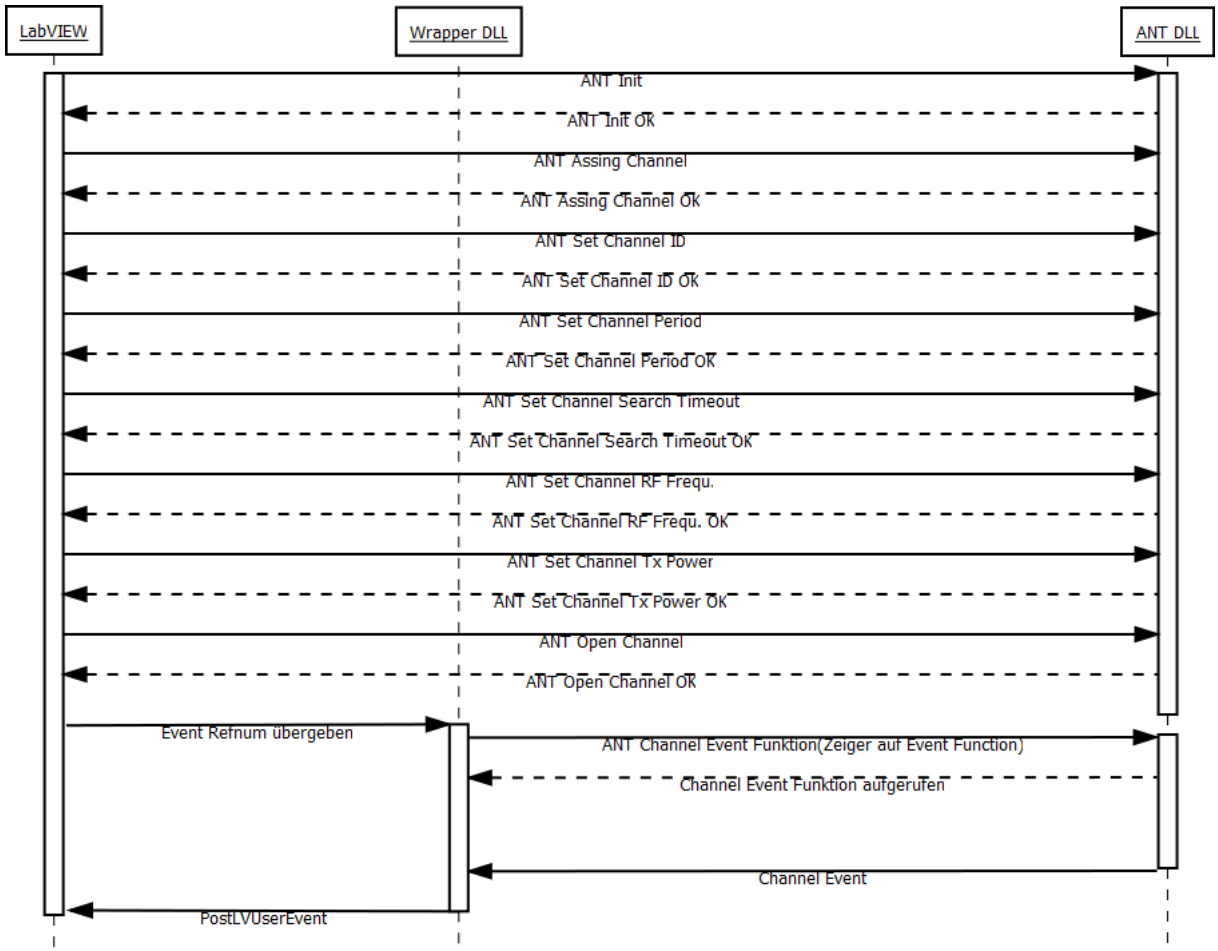


Abbildung 33: Kommunikation zwischen LabVIEW und der ANT DLL mit Wrapper

Zur Programmierung der Wrapper DLL wurde die Entwicklungsumgebung Visual C++ eingesetzt. Der gesamte Code der DLL liegt dieser Arbeit in digitaler Form, als Visual C++ Projekt bei. Die Funktionen werden im Folgenden erörtert.

Die DLL besteht aus sechs Funktionen wovon *ANT_LabVIEW_Schnittstelle* exportiert wird d.h. dem Benutzer bzw. der Anwendung zur Verfügung steht.

5.1 Funktion: ANT_LabVIEW_Schnittstelle

Prototyp: int *ANT_LabVIEW_Schnittstelle*(UCHAR `antChannel`, LVUserEventRef `*eventrefLabVIEW`, LVUserEventRef `*responserefLabVIEW`)

ANT_LabVIEW_Schnittstelle dient als Wrapper zwischen der LabVIEW Anwendung und der ANT DLL. Ihr wird als Parameter der jeweilige ANT Kanal, von welchem die Daten empfan-

gen oder gesendet werden übergeben. Sie übernimmt durch LabVIEW generierte Ereignis Referenz Nummern, die die Kommunikation zwischen der DLL und LabVIEW ermöglichen. Diese Referenznummern werden in globale Variablen geschrieben, damit die anderen Funktionen der DLL darauf zugreifen können.

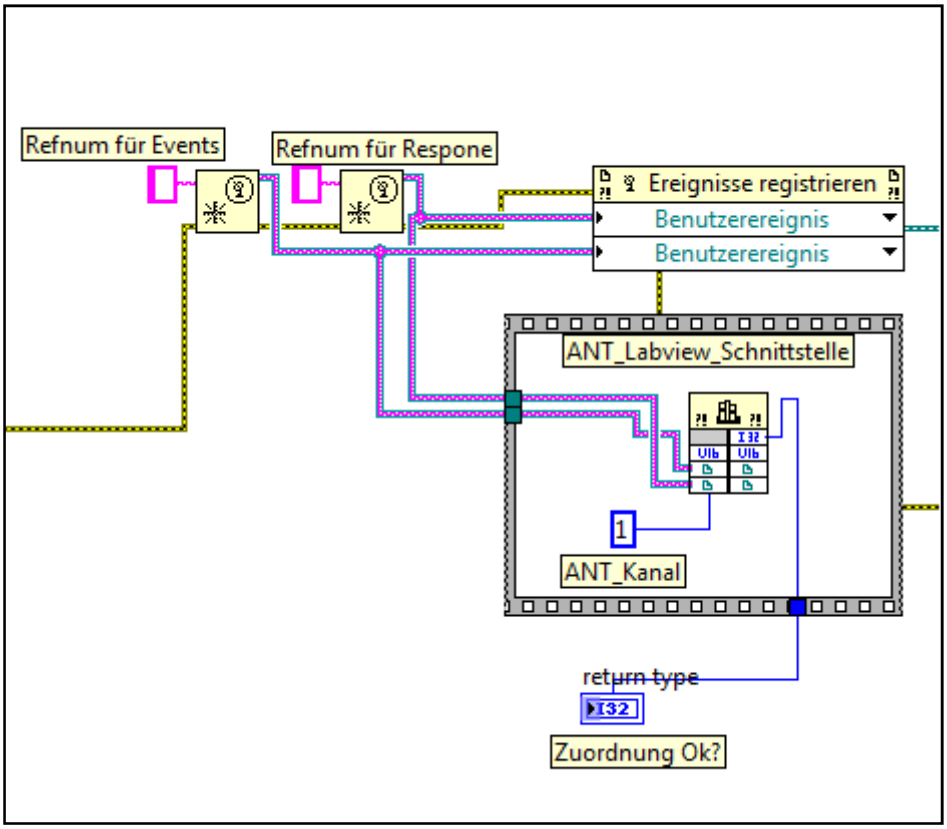


Abbildung 34: Die Verwendung der Funktion ANT_LabVIEW_Schnittstelle in LabVIEW

Die Referenznummern werden wie aus Abbildung 34 ersichtlich übergeben. Die Funktion alloziert den nötigen Speicher für die Buffer. *ANT_LabVIEW_Schnittstelle* lädt die **ANT_DLL.dll** und bezieht die Adresse der ANT Funktionen (*ANT_AssignChannelEventFunction* und *ANT_AssignResponseFunction*). Diese beiden ANT Funktionen werden im Anschluss mit den benötigten Parametern aufgerufen. Abschließend wird die **ANT_DLL.dll** freigegeben. Wenn keine Fehler aufgetreten sind, wird 1 zurückgegeben, anderenfalls 0. Dieser Returnwert wurde bewusst nicht als boolesche Variable[39] definiert, um Raum zur Definition von diversen Fehlercodes zu lassen.

[39] Boolesche Variable: Eine Variable die nur zwei Zustände annehmen kann. Entweder True oder False

5.2 Funktion: ANT_ChannelEventFunction/ANT_ResponseFunction

Die Funktionen, die *ANT_LabVIEW_Schnittstelle* zum Handeln der Events nutzt, werden nachfolgend erklärt. Da diese Funktionen sehr ähnlich ablaufen wird hier die Prozedur anhand von *ANT_ChannelEventFuction* erklärt.

Prototyp: BOOL *ANT_ChannelEventFunction*(UCHAR `ucChannel`, UCHAR `ucEvent`)

BOOL *ANT_ResponseFunction*(UCHAR `ucChannel`, UCHAR `ucMsgId`)

In *ANT_ChannelEventFuction* wird das Verhalten definiert, welches beim Aufrufen von *ANT_AssignChannelEventFunction* (bzw. *ANT_AssignResponseFunction)* ausgeführt werden soll. In diesem Fall muss *ANT_ChannelEventFuction* die erhaltenen Daten an LabVIEW senden.

Als Parameter besitzt sie `uChannel` und `ucEvent`. Diese Parameter werden durch das ANT Modul übergeben, und spezifizieren den Kanal und die Art des Events. Sie können wie in Abbildung 35 zur weiteren Ablaufsteuerung verwendet werden.

```
1  switch(ucEvent){
2      case EVENT_RX_BROADCAST:
3              mache was bestimmtes…
4              break;
5  }
```

Abbildung 35: Switch Statement zur differenzierten Behandlung der Events in der ANT_ChannelEventFunction

Alle ANT Events die in **antmessage.h** definiert sind können genutzt werden. Diese Datei ist als Include[40] anzugeben.

In der *ANT_ChannelEventFunction* muss die Funktion *SendEvent* aufgerufen werden. *SendEvent* leitet die Daten an LabVIEW weiter.

[40] Include: Include ist ein Schlüsselwort in vielen Programmiersprachen. Es dient zur Integration von Dateien in den Quellcode.

5.3 Funktion: SendEvent/SendResponse

Prototyp: void *SendEvent*(UCHAR `ucChannel`, UCHAR `ucEvent`)

void *SendResponse*(UCHAR `ucChannel`, UCHAR `ucMsgId`)

Diese Funktionen sind einander ähnlich, weshalb die Erklärung am Beispiel von *SendEvent* erfolgt.

Der Funktion werden als Aufrufparameter der Kanal, und der Event übergeben. *SendEvent* schreibt diese Daten und den Inhalt des ANT Buffers in den vorher allozierten Platz für den LabVIEW Datenstring. Die Übermittlung dieser Daten an die LabVIEW Anwendung erfolgt über *PostLVUserEvent*[41]. LabVIEW wird so über die Event Referenznummer (Abbildung 34) darüber informiert, dass Daten anliegen und kann diese verarbeiten.

[41] PostLVUserEvent: Diese Funktion wird von LabVIEW exportiert. Um Sie zu nutzen muss die Datei „extcode.h“ als Include angegeben werden.

5.4 Verwendung der Wrapper DLL in LabVIEW

Nachdem der Aufbau und die Interaktion der ANT DLL und der Wrapper DLL geklärt wurde, soll dieses Kapitel die Verwendung der Wrapper DLL in LabVIEW darlegen.

Die Wrapper DLL kommuniziert mit LabVIEW über einen Userevent. Die Verarbeitung solcher in LabVIEW erfolgt innerhalb einer Ereignisstruktur. Sobald ein Ereignis registriert wird, wird die Struktur abgearbeitet. Abbildung 36 zeigt den zugehörigen Abschnitt im Blockdiagramm.

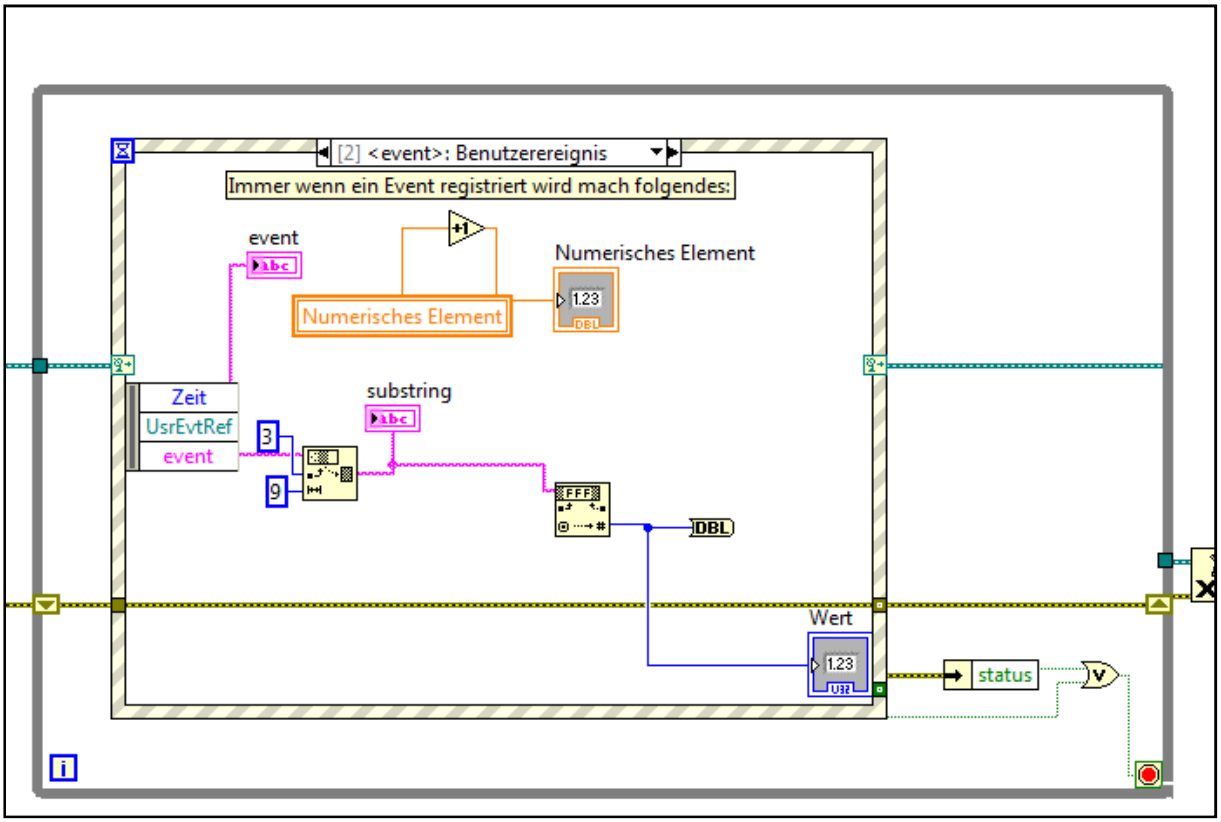


Abbildung 36: LabVIEW Event Struktur

Beim Stoppen der LabVIEW Applikation, wird die Struktur beendet und **ANT_DLL.dll** vom Speicher entfernt. Dieser Vorgang ist in Abbildung 37 dargestellt.

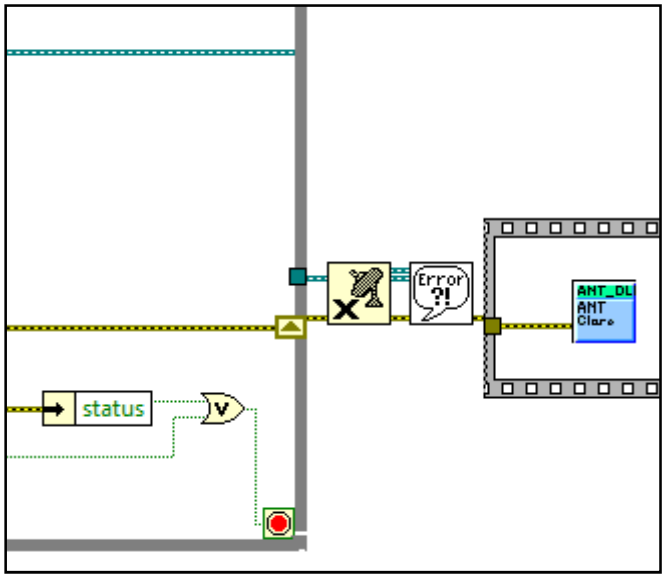


Abbildung 37: Freigeben des Events und die nötigen Aufräumarbeiten nach Beenden der Applikation in LabVIEW

6. Eusociality_Control_Center.vi

All diese notwendigen Schritte (Konfigurieren und Öffnen eines ANT Kanals, und zum Empfangen der Daten), wurden vom Autor in einem LabVIEW VI zusammengefasst. Die Benutzerin bzw. der Benutzer kann die Konfiguration des Empfängers mit einigen wenigen Mausklicks durchführen. Die empfangenen Daten werden angepasst nach ihrem Inhalt aus Datenstring (`eventStringHandle bzw. responseStringHandle`[42]) gelesen und als Integer Zahlenwert ausgegeben. Dieses VI dient als Referenz für den Umgang der **ANT_DLL.dll** und der **antwrapper.dll** in LabVIEW.

[42] eventStringhandle bzw. responseStringhandle: Diese beiden Variablen vom Datentyp LStrHandle zeigen auf die Informationen, welche mit PostLVUserEvent and LabVIEW gesendet werden.

Abbildung 38 zeigt das Frontpanel von **Eusociality_Control_Center.vi**. Auf der linken Seite des Frontpanels befinden sich alle Elemente, die zur Konfiguration des ANT USB Interface Boards und Aufbau des Datenübertragungskanals zum entfernten ANT Modul erforderlich sind.

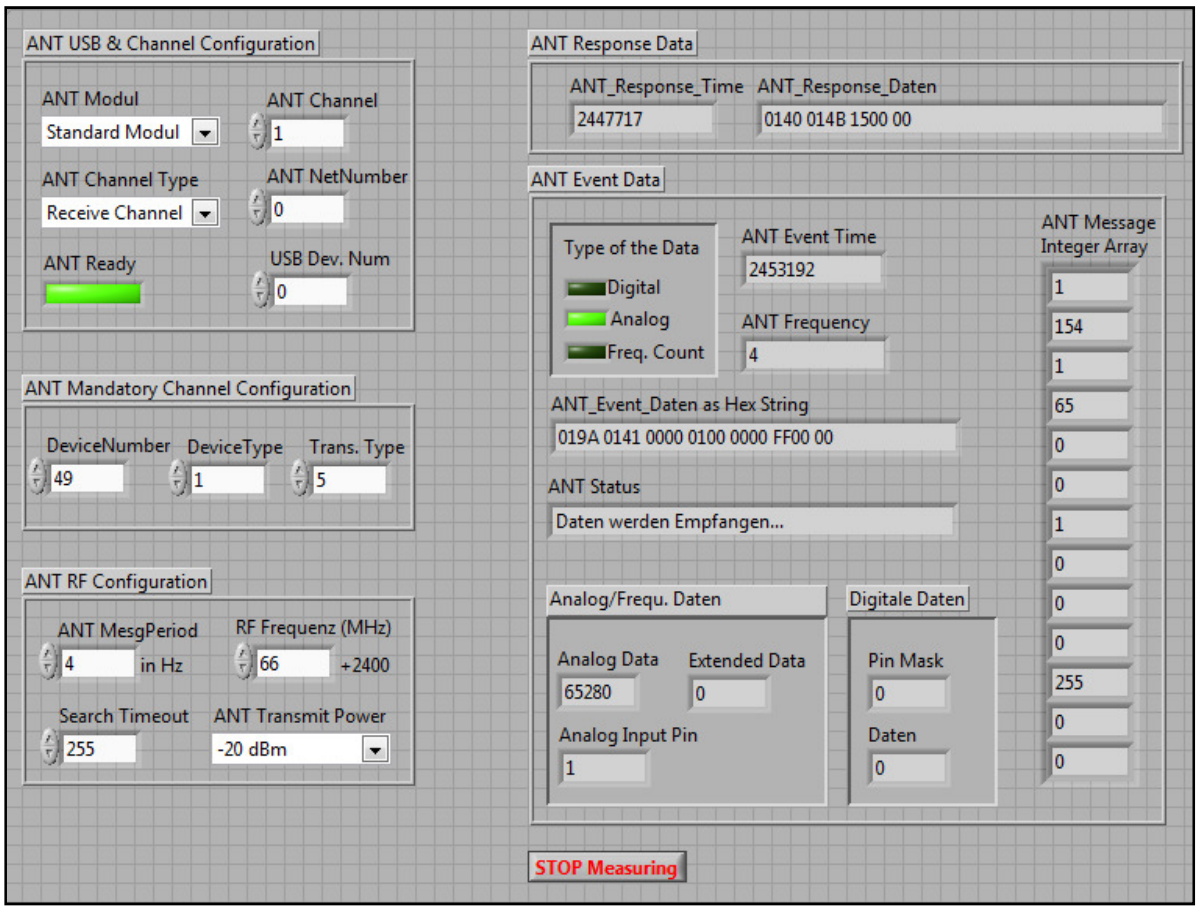


Abbildung 38: Frontpanel des Eusociality_Control_Center.vi

Die rechte Seite stellt die empfangenen Daten dar. Weiters wird die Art der Daten, welche das ANT Modul sendet über ein boolesches Element (Type of the Data) repräsentiert. Die Response Daten, und der Zeitpunkt zu dem sie empfangen wurden, werden hier nicht weiter verarbeitet. Bei Bedarf kann dies implementiert werden.

Die Event Daten gilt es differenzierter zu betrachten. Diese Daten werden als UCHAR Array[43] mit der Länge von 13 empfangen. Das bedeutet, dass das Zeichen 41 dem Zahlenwert 65 entspricht. Im **Eusociality_Control_Center.vi** liegen diese Daten in ihrer ursprünglichen Form aber auch als UINT8[44] Array vor.

Ungeachtet der Darstellungsform, enthalten die Daten alle wichtigen Metainformationen und auch die konkreten Messwerte.

Index	0	1	2	3	4	5	6	7	8	9	10	11	12
Daten	01	9A	01	41	00	00	01	00	00	FF	FF	00	00
	Chan.	Event	Chan.	Msg.ID	Data		Pin	Data		Messwert		Ext.	

Abbildung 39: Aufschlüsselung des ANT Datenstring in die einzelnen Elemente

Abbildung 39 zeigt die Daten als UCHAR Array mit ihrer zugehörigen Indexposition. Indexposition 0 (01) gibt an von welchem ANT Kanal die Daten empfangen wurden. Indexposition 1 (9A) kennzeichnet das Ereignis am Kanal. Definiert sind Events in der Datei **antdefines.h**.

```
153 ////////////////////////////////////////////
154 // PC Application Event Codes
155 ////////////////////////////////////////////
156 //NOTE: These events are not generated by the embedded ANT module
157
158 #define EVENT_RX_BROADCAST              ((UCHAR)0x9A)    // returned when module receives broadcast data
159 #define EVENT_RX_ACKNOWLEDGED           ((UCHAR)0x9B)    // returned when module receives acknowledged data
160 #define EVENT_RX_BURST_PACKET           ((UCHAR)0x9C)    // returned when module receives burst data
161
162 #define EVENT_RX_EXT_BROADCAST          ((UCHAR)0x9D)    // returned when module receives broadcast data
163 #define EVENT_RX_EXT_ACKNOWLEDGED       ((UCHAR)0x9E)    // returned when module receives acknowledged data
164 #define EVENT_RX_EXT_BURST_PACKET       ((UCHAR)0x9F)    // returned when module receives burst data
```

Abbildung 40: Definition der ANT Event Message Codes aus antdefines.h

[43] Array: Ein Array ist eine Datenstruktur, die mehrere Elemente eines bestimmten Datentyps unter einen Namen zusammenfasst (Georgi et al., 2007).

[44] UINT8: UNIT8 ist ein 4 Bit Integer Datentyp.

Diese Codes können dazu genutzt werden, um bestimmte Datenoperationen beim Auftreten eines Ereignisses durchzuführen, wie in Abbildung 41 dargestellt wird.

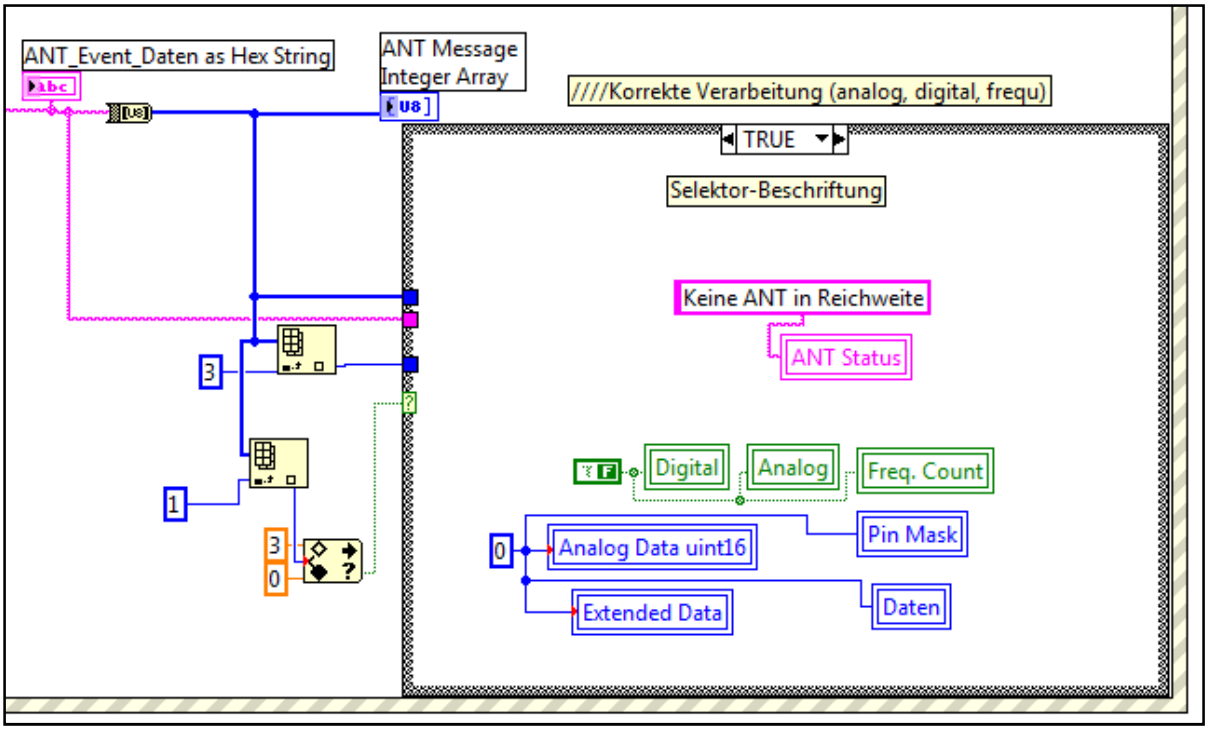


Abbildung 41: Verarbeitung eines Events in Eusociality_Control_Center.vi

Abbildung 42 zeigt die Nutzung des Eventcodes. Der Wert an Indexposition 1 des Arrays wird geprüft. Zahlen zwischen 0 und 3 (1=Searchtimeout, 2=RX Fail) dienen als Fehlerindikator. Im Fehlerfall zeigen alle Variablen 0 bzw. False, und `ANT Status` die Meldung „Keine ANT in Reichweite“ an. Bei einem anderen Event werden die Daten verarbeitet.

Bei diesem Schritt wird der Wert an Indexposition 3 herangezogen. Dieser enthält Informationen zur Identifizierung der empfangenen Daten. Dabei spezifiziert

- 40 UCHAR bzw. 64 Integer Digitale Daten,
- 41 UCHAR bzw. 65 Integer Analoge Daten,
- 42 UCHAR bzw. 66 Integer Frequenz Counter Daten.

In **Eusociality_Control_Center.vi** wurden die nötigen Identifikationswerte zur Verwendung eines SensRcore Moduls als Transmitter implementiert (Dynastream Innovations Inc., 2008c).

Die Metainformationen werden in einer Case Struktur[45] genutzt um verschiedene Rechenoperationen auszuführen (Abbildung 42).

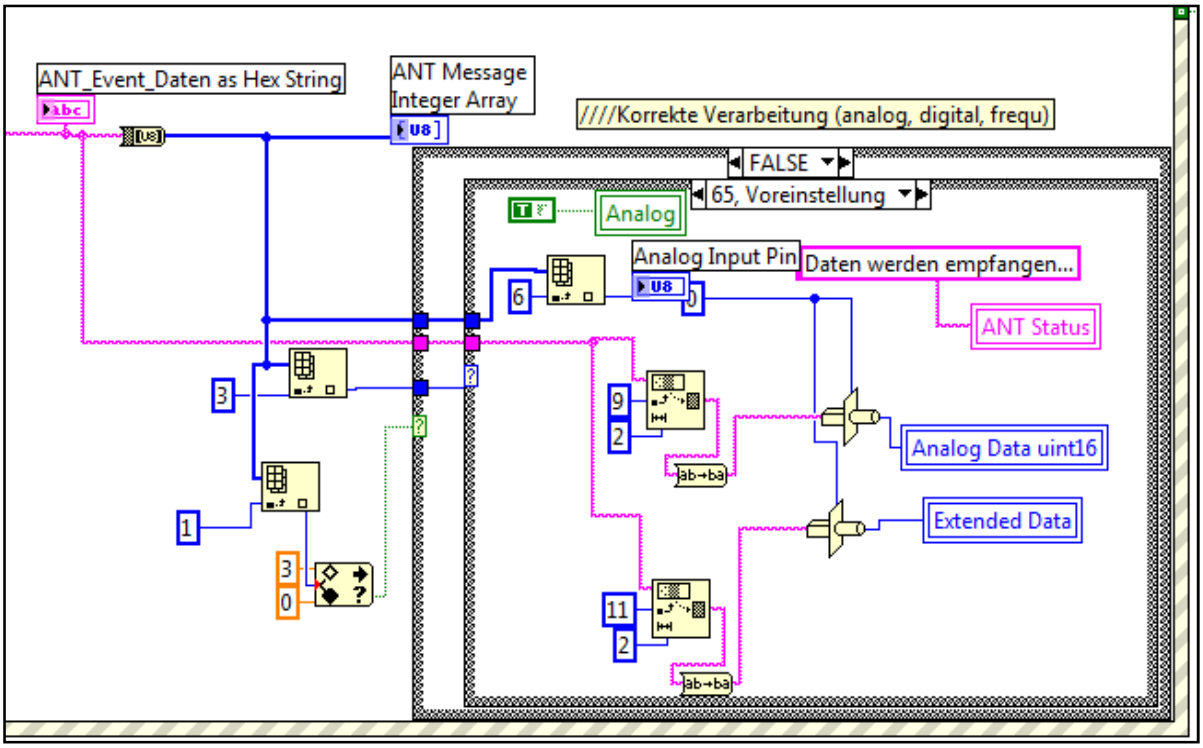


Abbildung 42: Datenverabeitung in der Eventstruktur nach der Datenart

Werden Daten empfangen, die als analoge Daten[46] gekennzeichnet sind, soll angezeigt werden, dass analoge Daten vorliegen. Dazu wird der Wert der Variable `Analog` auf True gesetzt. Die Messwerte an Indexposition 9 und 10 des Datenarrays werden in 16 Bit Werte umgewandelt. In die Variable `ANT Status` wird geschrieben, dass Daten empfangen werden.

[45] Case Struktur: Case Strukturen werden eingesetzt um ein definiertes Verhalten auszuführen, wenn eine Variable eine bestimmten Zustand annimmt. Der Aufbau ähnelt einem Konditionalsatz. Bsp.: Die zu prüfende Variable ist „Wetter“. Wenn es regnet (Prüfung), nehme ich einen Schirm (Verhalten). Wenn die Sonne scheint, bin ich fröhlich.

[46] Analoge Daten werden durch den UINT8 Zahlenwert 65 identifiziert.

Am Frontpanel sieht das Ganze, wie in Abbildung 43 ersichtlich, aus.

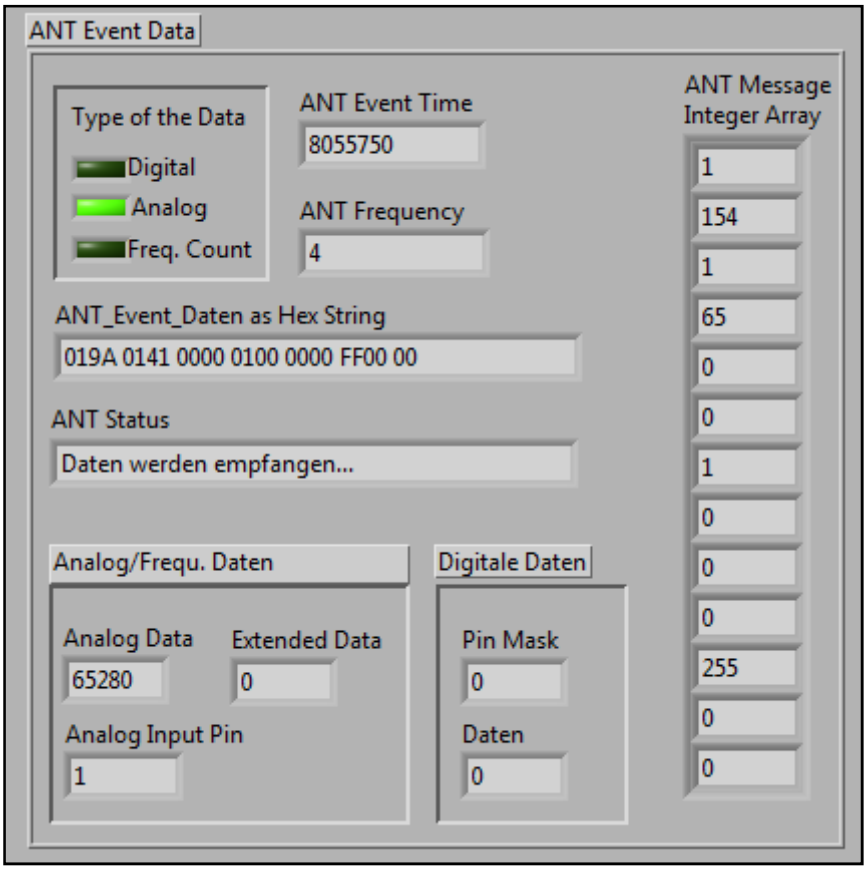


Abbildung 43: Datenverarbeitung in der Eventstruktur nach der Datenart Frontpanel

Bei Verarbeitung der analogen Messwerte ist darauf zu achten, dass diese als Little Endian[47] gesendet werden (LSB[48] zuerst). Darum wird vor der Umwandlung in einen Integer Zahlenwert die Reihenfolge umgekehrt.

Ist der Messwert an Indexposition 9 und 10: 01 CC mit der Basis 16, so muss zuerst die Position der Elemente vertauscht werden. Aus dem 01 CC wird CC 01. Dieser Wert kann in einem vorzeichenlosen 16 Bit Integer Wert umgerechnet werden. Daraus ergibt sich die Zahl 52225.

[47] Little Endian: Die Endianess gibt die Reihenfolge der Bytes von Daten an. Im Fall von Little Endian steht das LSB an erster Stelle (und somit an der niedrigsten Adresse im Speicher). Werden die z.B.: 0xFF 0F C0 als Little Endian dargestellt ergibt sich daraus: 0xC0 0F FF.

[48] LSB: Least significant bit

Die Periodendauer und die Frequenz wird, wie in Abbildung 44 und in Abbildung 45 dargestellt, errechnet.

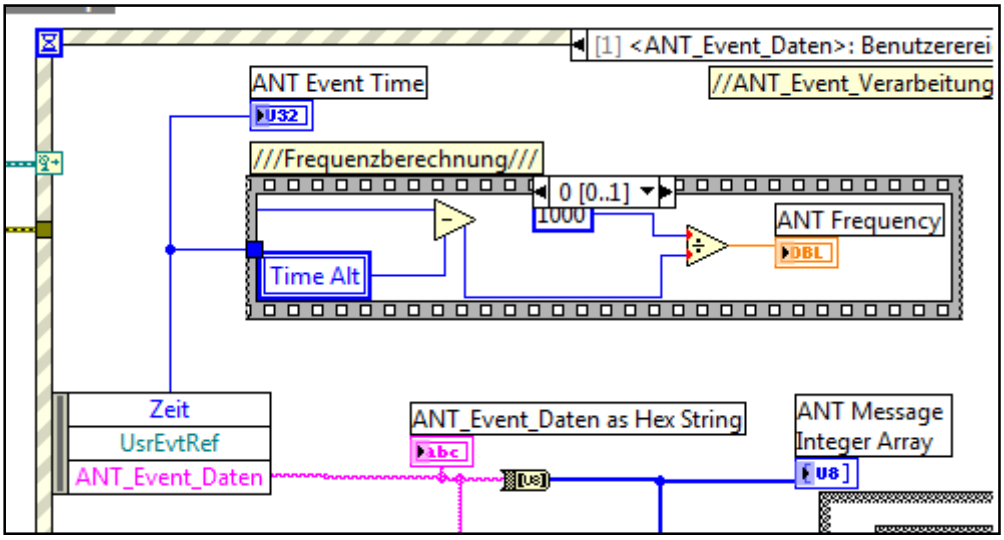


Abbildung 44: Berechnung der Frequenz in Eusociality_Control_Center.iv – Schritt 1

Der Wert der Variable Time Alt wird von dem Zeitwert, den die Eventstruktur beim Auftreten eines Events liefert abgezogen. Damit wird die Periodendauer berechnet. Durch eine Division dieses Werts mit 1000 ergibt sich die Frequenz.

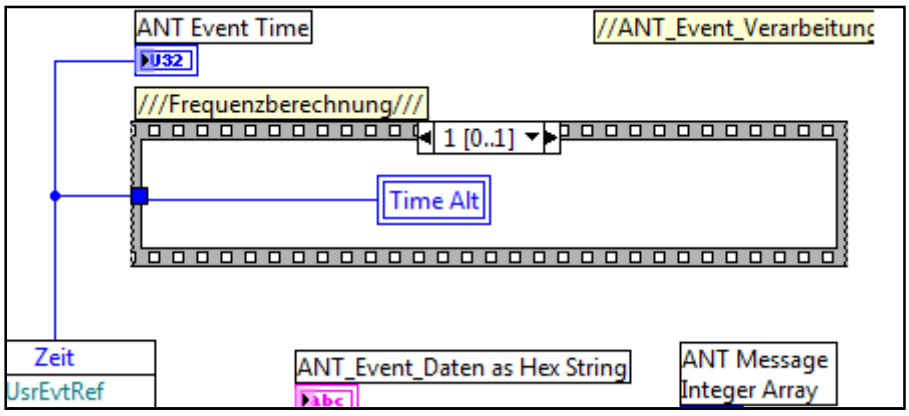


Abbildung 45: Berechnung der Frequenz in Eusociality_Control_Center.iv - Schritt 2

Im Anschluss daran wird die aktuelle Zeit des Events in die Variable `Time Alt` geschrieben, um für den nächsten Event t-1 zu erhalten.

Details zur Programmierung können dem Blockdiagramm von **Eusociality_Control_Center.vi** entnommen werden. Die VIs sind mit Kommentaren für das bessere Verständnis versehen. Gleiches gilt auch für den Quellcode der Wrapper DLL.

Wird **Eusociality_Control_Center.vi** oder **Eusociality_Control_Connect.vi** nicht als Teil der vom Autor bereitgestellten LabVIEW Bibliothek verwendet, ist sicherzustellen, dass die Dateien **ANT_DLL.dll**, **SiUSBXp.dll**, **SiUSB.dll** und **antwrapper.dll** im gleichen Verzeichnis wie das **Eusociality_Control_Center.vi** bzw. **Eusociality_Control_Connect.vi** liegen. Die **ANT_DLL.dll** legt bei erfolgreicher Initialisierung der Komponenten immer eine Debugdatei (**deviceX.txt**) im gleichen Verzeichnis an, welches zur Fehleridentifikation genutzt werden kann.

6.1 Eusociality_Control_Connect.vi

Eusociality_Control_Center.vi dient als Referenzapplikation zum Verwenden des ANT Protokolls und der Wrapper DLL. Es zeigt der Programmiererin bzw. dem Programmierer, wie mit Hilfe dieser Werkzeuge eine Applikation zum Einstellen und Auslesen von Daten des ANT Moduls einzusetzen sind.

Anwendungen, die auf ein bestimmtes Problem hin programmiert werden, müssen Daten anhand der Problemdefinition verarbeiten. Die Schritte bis zur Verarbeitung des Events bleiben jedoch gleich. Die Initialisierung des ANT USB Stick, das Konfigurieren und Öffnen des ANT Kanals sind in **Eusociality_Control_Connect.vi** zusammengefasst. Dieses VI wird, wie in Abbildung 46 abgebildet, mit den nötigen Parametern versorgt, und liefert eine Event Referenz Nummer zurück. Diese wird im Haupt VI der Applikation in einer Event Case Struktur abgearbeitet.

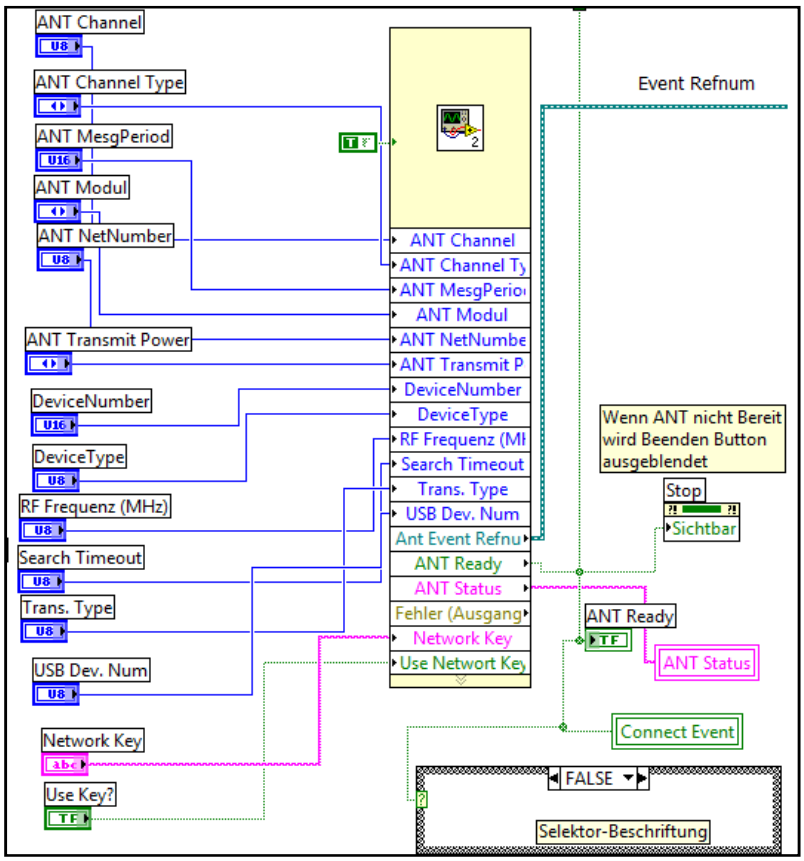


Abbildung 46: Die Verwendung und Konfiguration des Eusociality_Control_Connect.vi im Blockdiagramm

Bei der Abarbeitung des Events muss darauf geachtet werden, dass das Warten auf ein Ereignis nicht den restlichen Programmablauf blockiert.

7. WiBiFe – Wireless Biathlon Feedbacksystem

In den vorangegangenen Kapiteln dieser Arbeit wurde das ANT Protokoll, die ANT Hardware und LabVIEW verdeutlicht. Nun zeigt das folgende Kapitel anhand einer konkreten LabVIEW Anwendung den Einsatz der bereitgestellten Werkzeuge in der Praxis. Das vorgestellte System wird bis zum Status eines Prototyps entwickelt. Die verwendeten Sensoren sind Standardausführungen. Für eine professionelle Umsetzung müssen Sensoren produziert werden, die an die Anforderungen des Systems angepasst sind.

7.1 Konzept des Messsystems

Die ANT Hardware ermöglich es, wie im Verlauf dieser Arbeit erklärt wurde, Sensoren drahtlos zu betreiben. Dieses Kapitel beschäftigt sich mit der Programmierung eines Feedbacksystems für den Biathlonsport. Abbildung 47 skizziert die Funktionsweise des Systems.

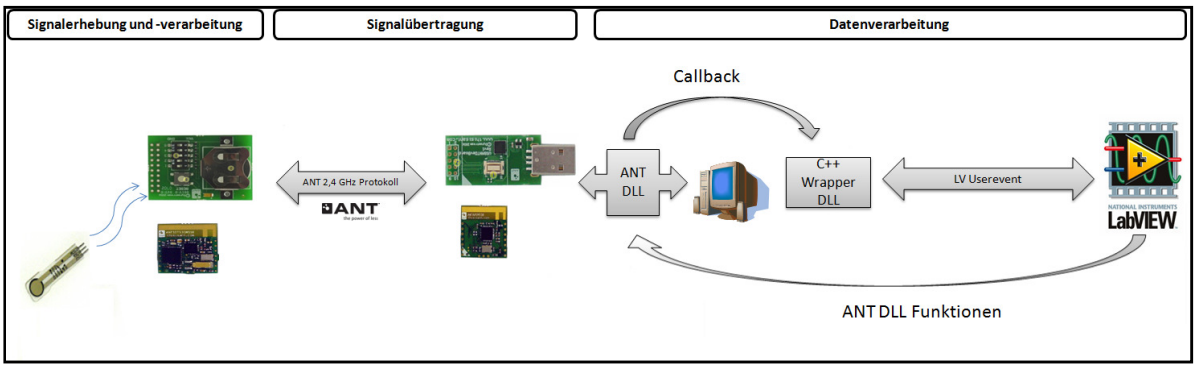


Abbildung 47: Schemenhafte Darstellung der Funktionsweise von WiBiFe – einem Feedbacksystem für den Biathlonsport

Ein flexibler, leichter Kraftsensor wird am Abzugsbügel eines Biathlongewehrs angebracht. Dieser Sensor misst den Kraftverlauf während des Schießvorganges. Diese so erhobenen Daten werden mit Hilfe der ANT Hardware drahtlos zu einem Computer übertragen. Eine LabVIEW Applikation stellt diese in Echtzeit dar, und speichert sie für spätere Analysen.

7.2 Details zum verwendeten Flexiforce Sensor

Die verwendete ANT Hardware wurde in den Kapiteln 3.4 – 3.7 vorgestellt. In diesem Kapitel wird auf die Spezifikationen des verwendeten Kraftsensors eingegangen, und alle nötigen Schritte erläutert, um mit diesem Sensor Daten zu erheben.

7.2.1 Eigenschaften und Aufbau der Flexiforce Sensoren

> "The FlexiForce sensor is an ultra-thin and flexible printed circuit, which can be easily integrated into most applications. With its paper-thin construction, flexibility and force measurement ability, the FlexiForce force sensor can measure force between almost any two surfaces and is durable enough to stand up to most environments. FlexiForce has better force sensing properties, linearity, hysteresis, drift, and temperature sensitivity than any other thin-film force sensors. The "active sensing area" is a 0.375" diameter circle at the end of the sensor." (Tekscan, 2005, S. 4)

Die Sensoren bestehen, wie Abbildung 48 zeigt, aus zwei Schichten Polyester, auf denen leitendes Material (Silber) aufgebracht ist. Zwischen diesen beiden Schichten befindet sich, eine drucksensitive Tinte. Das leitende Material ist in Spalten und Reihen angeordnet. Deren Überschneidungsflächen bilden die Messpunkte (Tekscan, 2005).

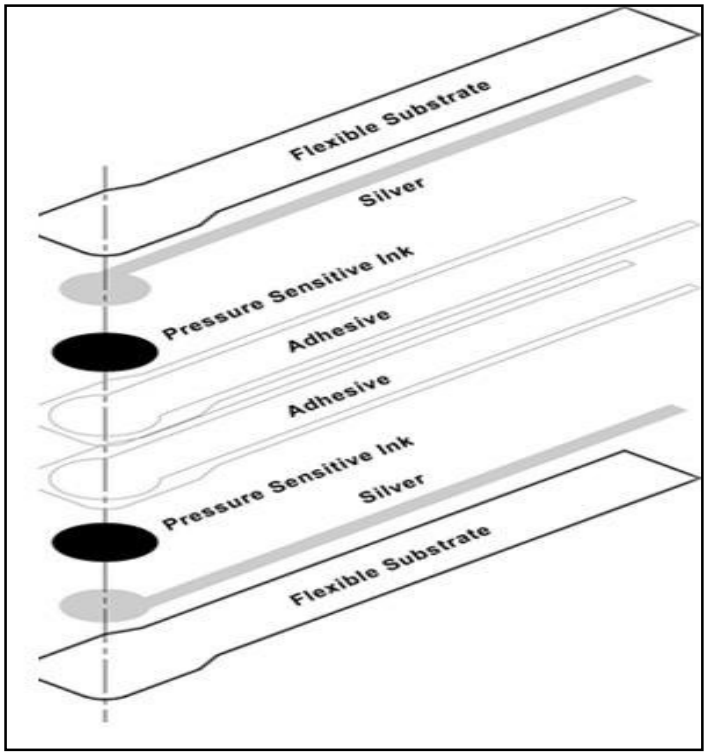


Abbildung 48: Bestandteile und Aufbau von Flexiforce Sensoren (Tekscan, 2007)

Der Flexiforce Sensor arbeitet wie ein variabler elektrischer Widerstand. Je höher die aufgebrachte Kraft bzw. der Druck, desto geringer ist der Widerstand.

7.2.2 Input/Output Verhalten

Damit WiBiFe funktioniert, verlässliche und richtige Werte liefert, muss der Sensor bestimmte Auflagen erfüllen. Neben dem geringen Eigengewicht und dem dünnen Aufbau, muss ein lineares oder berechenbares Verhalten zwischen der, auf den Sensor wirkenden Kraft bzw. wirkender Druck und dem Spannungsoutput bestehen. Tekscan bietet mit den Flexiforce Sensoren eine Plattform, die diesen Anforderungen gerecht wird. Abbildung 49 zeigt ein Kraft/Widerstands- und ein Kraft/Leitwertdiagramm eines Flexiforce Sensors.

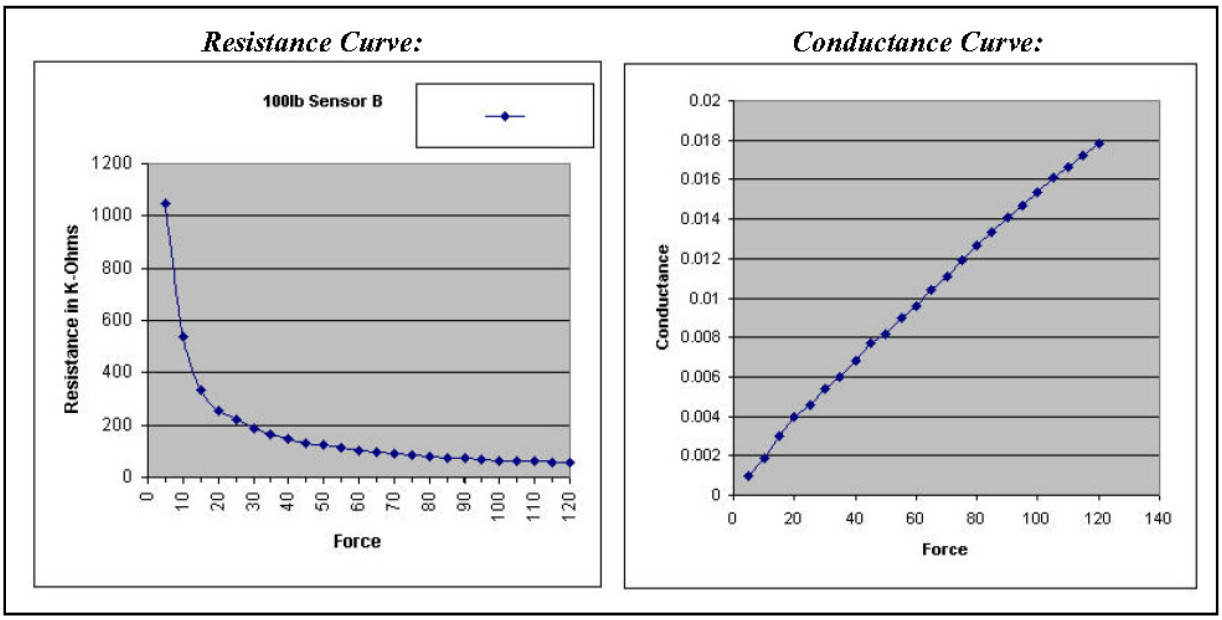


Abbildung 49: Kraft/Widerstands- und Kraft/Leitwertdiagramm eines Flexiforce Sensors (Tekscan, 2005)

Die Widerstandskurve ist nicht linear, aber dennoch berechenbar. Das Verhalten gilt es in künftigen Experimenten herauszufinden, und die Messwerte dementsprechend zu korrigieren.

Der benötigte Messbereich für den Biathlonsport[49] wird von den Standard Flexiforce Sensoren nicht genau abgedeckt. Bei Bedarf können Sensoren mit dem gewünschten Messbereich produziert werden.

Für die Erstellung des WiBiFe Prototypen wird der Flexiforce A201-2 Sensor verwendet. Dieser deckt den Messbereich von 0 – 110N ab.

7.2.3 Design der Schaltung zum Betreiben des Sensors

Um den Flexiforce Sensor nutzbar zu machen muss die Schaltung aus Abbildung 50 realisiert werden. Diese versorgt den Sensor mit einer Spannung und normiert den Spannungsoutput V_{out} auf einen gewünschten Bereich. Durch diese Normierung wird sichergestellt, dass V_{out} den Messbereich des ANT Moduls optimal nutzt. Durch verändern von R_F wird V_{out} verändert.

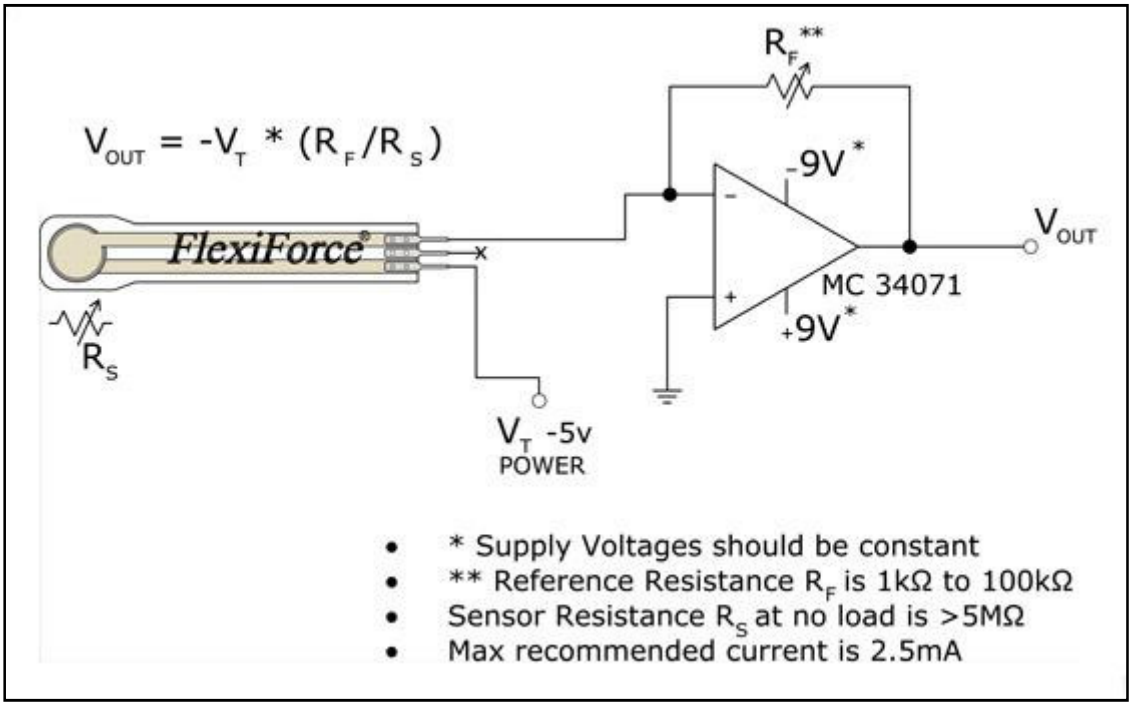


Abbildung 50: Elektrische Schaltung zum Betreiben eines Flexiforce Sensors (Tekscan, 2005)

[49] Die minimale Gewichtskraft zum Auslösen eines Schusses beträgt 500g. Das entspricht rund 5N (International Biathlon Union, 1998).

Eine kritische Eigenschaft des Flexiforce ist dessen hoher Widerstand im unbelasteten Zustand. Abbildung 51 zeigt, dass eine Versorgungspannung von 9V verwendet wird. Eine solche Spannung kann das ANT Battery Board nicht liefern. Für den Prototyp wurde die Originalschaltung (vgl. Abbildung 50) realisiert, d.h. zur Verwendung des Feedbacksystems in der Praxis muss diese Schaltung adaptiert werden, damit die Spannungsversorgung über das Battery Board erfolgen kann.

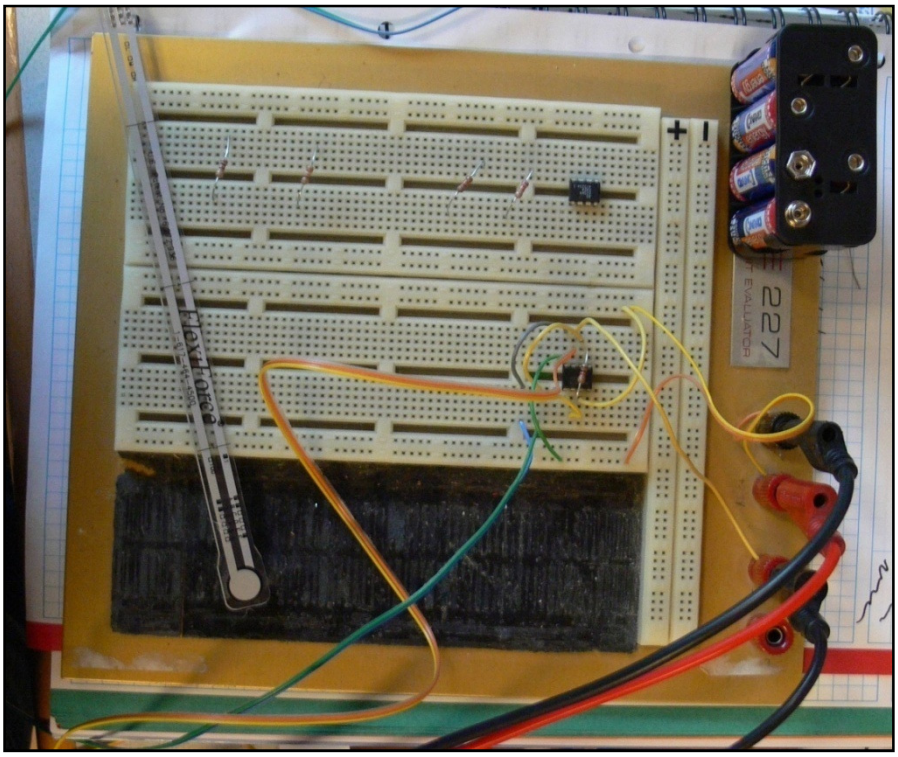


Abbildung 51: Schaltung zur Verwendung des Flexiforcesensors in der Praxis

Abbildung 52: Netzteil und Messgerät für die Flexiforceschaltung

Abbildung 51 und Abbildung 52 zeigen den Schaltungsaufbau für den Prototyp des Feedbacksystems.

7.2.4 Anbringung des Sensors an das ANT Battery Board

Die Anbringung des Sensors am ANT Battery Board wird in Abbildung 53 dargestellt. Das Battery Board bietet Anschlussmöglichkeiten für den Sensor und versorgt das ANT SensRcore Modul mit Strom. AIO0 ist der Eingang für die analoge Spannung. Vcc liefert die Versorgungsspannung. Der Anschluss Gnd ist die Masse.

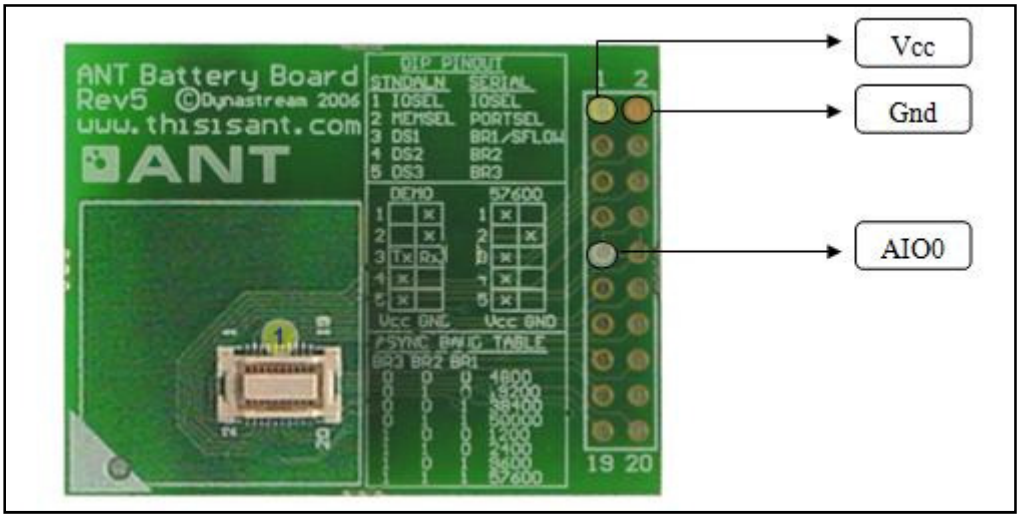


Abbildung 53: Verbindungsstellen zum Anbringen eines Sensors an das ANT Battery Board

7.3 Konfiguration des SensRcore Moduls

Abschließend muss das ANT SensRcore Modul konfiguriert werden. Dieser Vorgang wird wie im Kapitel 3.6 beschrieben, durchgeführt. Einzig die Abtastrate und die „Message Rate“ müssen der Applikation angepasst werden. WiBiFe nutzt eine Abtast- und Übertragungsrate von 100Hz. Es empfiehlt sich, das LED des ANT SensRcore Moduls anzustellen. Diese LED blinkt bei jeder Datentransmission. Damit wird bei Problemen die Anzahl möglicher Fehlerquellen reduziert. Das LED benötigt zwar Strom, die Menge ist jedoch gering und somit vernachlässigbar. Der Konfigurationsvorgang ist in Abbildung 54 und Abbildung 55 dargestellt.

Das ANT SensRcore Modul kann in dieser Konfiguration bei einer Nutzungsdauer von 1 Stunde am Tag mit einer vollen 2023 Knopfbatterie 80 Tage lang betrieben werden. Der „Power Estimator“ zur Strombedarfsberechnung liegt dem ANT SDK bei.

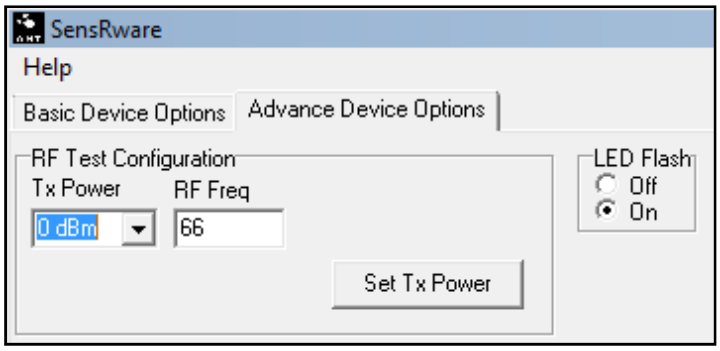


Abbildung 54: Aktivieren des LEDs des ANT Moduls in der SensRcore Configuration

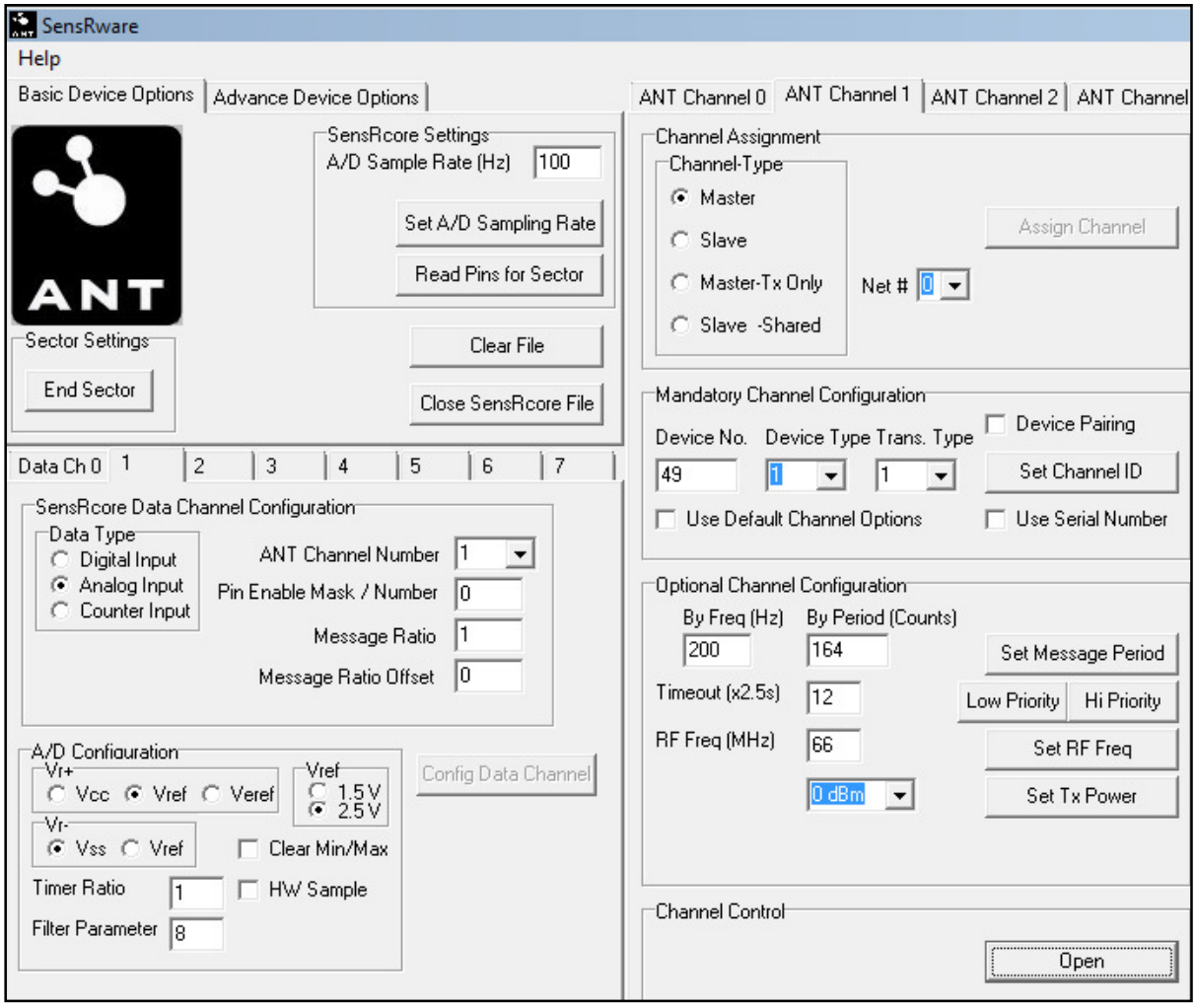


Abbildung 55: Einstellung zur SensRcore Konfiguration für WiBiFe

SensRware generiert eine Konfigurationsdatei, die alle Einstellungen enthält. Abbildung 56 zeigt den Inhalt der Konfigurationsdatei.

```
Existing Configuration
[68][00][01]                          // Enabling LED on RF events
[93][00][47][01]                      // Setting A/D Sample Rate at 100 Hz
[91][01][01][41][00][01][00]          // Configuring Data Channel 1 on ANT Channel 1
                                         as Analog Input using pin(s) 00 reporting on 100.00% of messages
[94][00][01][09][01][08][00][00]      // Configuring Data Channel 1 A/D with reference setting of 9,
                                          sampling at 1.00 of sample rate, filter N value of 8
[42][01][10][00]                      // Assign Channel Master on ANT chanel 1 on network 0
[51][01][31][00][01][01]              // Assign Channel ID 1, Dev. Num = 49, Dev. Type = 1, Trans. Type = 1
[43][01][A4][00]                      // Set ANT Channel 1 Message Period to 200.00 Hz
[63][01][0C]                          // Set ANT Channel 1 low priority search timeout to  30.0 s
[45][01][42]                          // Set ANT Channel 1 RF Frequency to 2466 MHz
[4B][01]                              // Open ANT chanel 1
```

Abbildung 56: SensRcore Konfigurationsdatei für WiBiFe - wibife.txt

Diese Datei kann gemäß der Anleitung aus Kapitel 3.6 hochgeladen werden. Die Hardware ist somit entsprechend den Anforderungen konfiguriert, und einsatzbereit.

7.4 WiBiFe – Umsetzung und Programmierung in LabVIEW

Zur Umsetzung des Programms in LabVIEW wird **Eusosiality_Control_Connect.vi** genutzt. Das VI übernimmt das Setup der ANT Hardware und des ANT Kanals. Der Programmablauf unterteilt sich in folgende Aufgaben:

- Standard State herstellen
- Eine Verbindung mit Hilfe von Eusociality Control Connect herstellen
- Verarbeitung der Daten des ANT Event

Abbildung 57 zeigt schematisch den oben beschriebenen Programmablauf als UML Zustandsdiagramm. Da diese Graphik als Überblick dient, wurden die Zustände dabei nicht bis ins letzte Detail verfeinert.

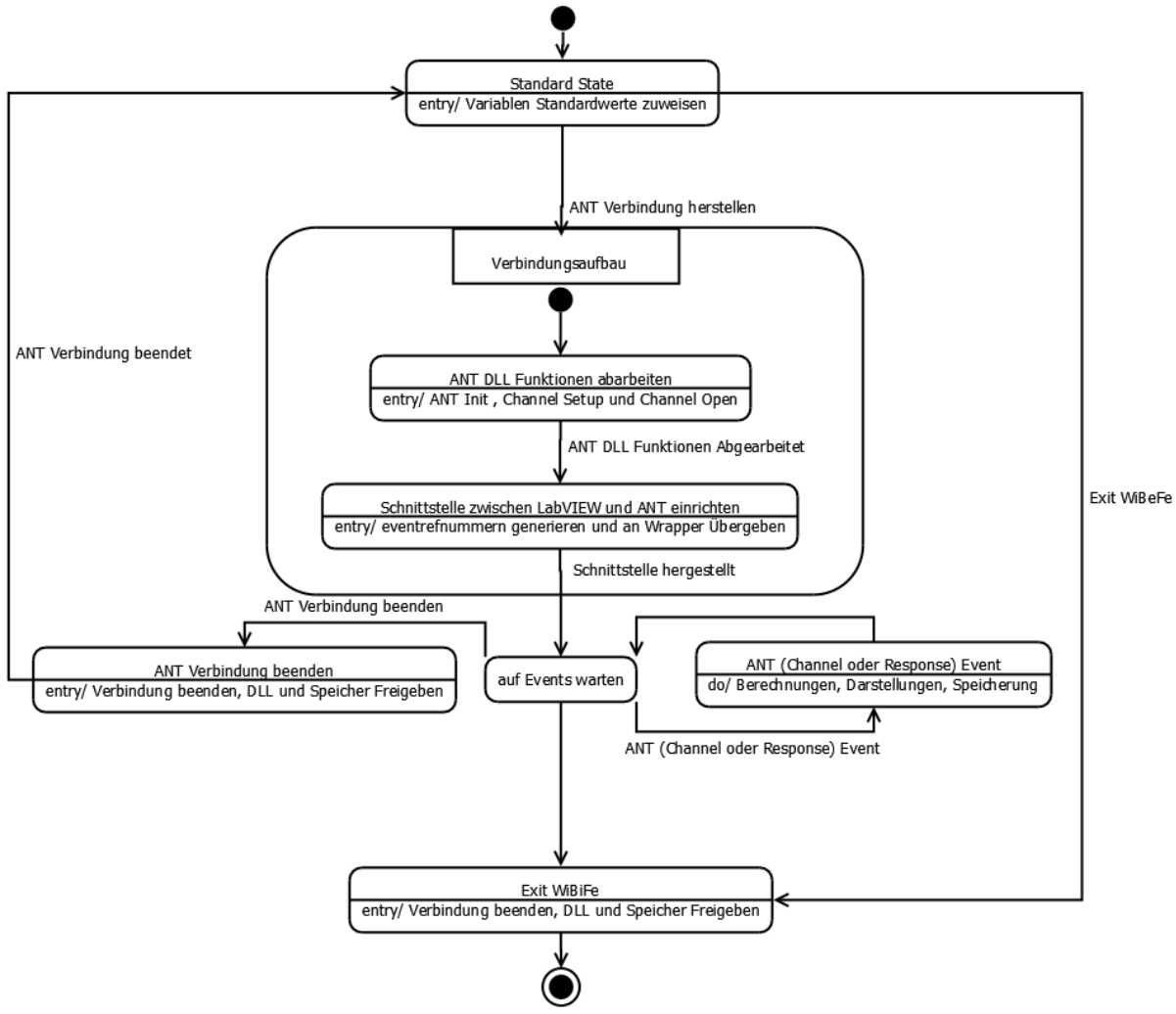


Abbildung 57: Programmkonzept von WiBiFe

7.4.1 Standard State

Beim Programmstart und beim Beenden der ANT Verbindung werden alle Variablen mit ihren Standardwerten initialisiert, um die Applikation in einen definierten Zustand zu überführen. Das ist notwendig, um die Korrektheit der Berechnungen und Programmbedingungen zu garantieren.

7.4.2 WiBiFe ANT Konfiguration - Setup und Verbindung

Nach Herstellen des Standard States wartet WiBiFe auf ein Benutzerereignis. Zur Auswahl stehen „Exit WiBiFe“ oder „ANT Verbindung Herstellen“. Der Sinn besteht darin, dass der Benutzerin bzw. dem Benutzer durch dieses Warten die Möglichkeit gegeben wird, Parameter zum Konfigurieren des ANT USB Interface Boards bzw. dem ANT Kanal zu ändern. Der rot markierte Bereich aus Abbildung 58 zeigt die veränderbaren Parameter. Von der Benutzerin bzw. dem Benutzer geänderte Werte können als Standard definiert, bzw. die voreingestellten Standardwerte können wieder hergestellt werden. Der gelb umrandete Bereich aus Abbildung 58 markiert das zugehörige Menü. Die Konfiguration des Soundmoduls findet im grün umrandeten Bereich statt.

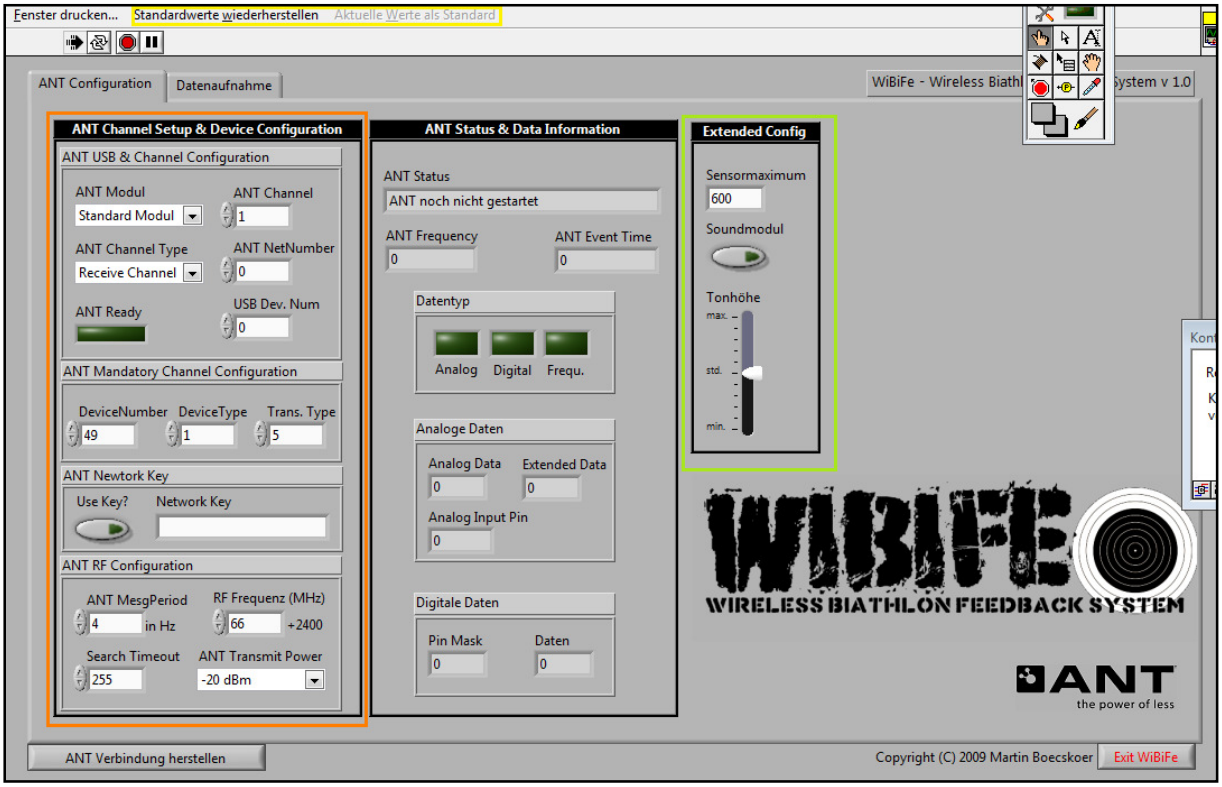


Abbildung 58: ANT Konfiguration am Frontpanel in WiBiFe

ANT Status & Data Information dienen zur Darstellung. Diese Felder können zur Überprüfung genutzt werden, ob und welche Daten nach Herstellen der Verbindung die Applikation empfängt.

7.4.3 Die Datenaufnahme in WiBiFe

War die Initialisierung der ANT Hardware erfolgreich, werden die Buttons „ANT Verbindung herstellen“ und „Exit WiBiFe“ ausgegraut und deaktiviert. Der Button „ANT Verbindung beenden“ wird eingeblendet. Die vom ANT Modul empfangenen Daten werden aufbereitet und dargestellt (Abbildung 59).

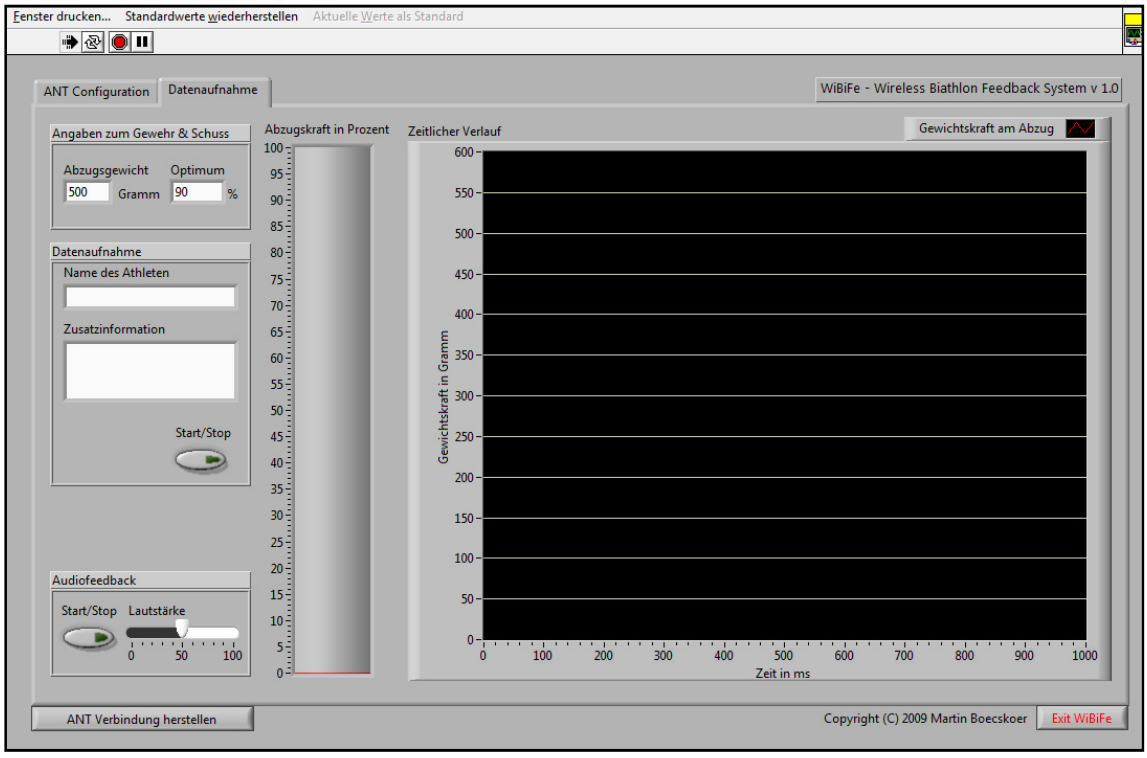


Abbildung 59: Datenaufnahme und –darstellung in WiBiFe

7.4.3.1 Angaben zum Gewehr und Schuss

Unter dem Abschnitt „Angaben zum Gewehr & Schuss“ kann die Benutzerin bzw. der Benutzer das Abzugsgewicht[50] einstellen. Das Optimum gibt an, wie viel des Abzugsgewichts vor dem Zielvorgang vom Schützen am Abzugsbügel anzulegen ist. Wird dieses Optimum erreicht, ändert sich die Farbe der beiden Diagramme von Rot nach Grün.

7.4.3.2 Datenaufnahme

Die Elementgruppe „Datenaufnahme“ steuert das Schreiben der Messwerte in eine Datei. Angaben zum Namen der Sportlerin bzw. des Sportlers und Zusatzinformationen sind optional. Diese Angaben bilden die Header der Datei. Der Start/Stop Button startet oder beendet den Schreibvorgang.

Die Messwertdatei wird im Verzeichnis, in welchem sich das Programm befindet erstellt und trägt den Namen **wibife_<Datum>_<Uhrzeit>.txt**. In der Datei stehen von links nach rechts Zeitwert, Schussnummer, Prozent der Abzugskraft und das Abzugsgewicht in Gramm.

Diese Datei kann zur weiteren Analyse verwendet werden. Das Trennzeichen zwischen den Spalten ist ein Tabulator.

7.4.3.3 Audiofeedback

Des Weiteren bietet WiBiFe die Möglichkeit, neben der visuellen auch einer akustischen Rückmeldung. Der Ton wird bis zum Erreichen des eingestellten Optimums tiefer, danach bleibt er gleich.

7.4.4 Weitere Informationen zu WiBiFe

Details zur Programmierung und Installation können der Magisterarbeit beiliegenden CD entnommen werden.

[50] Das Abzugsgewicht muss mindestens 500 Gramm betragen (International Biathlon Union, 1998).

8. Zusammenfassung und Ausblick

Im Laufe dieser Magisterarbeit wurde gezeigt, wie das ANT Protokoll und die Hardware des ANT Development Kits erfolgreich in LabVIEW implementiert werden können. Die nötigen Schnittstellen und Werkzeuge wurden vom Autor programmiert und bereitgestellt. Das Feedbacksystem WiBiFe zeigt den Einsatz dieser Werkzeuge in der Praxis.

Die Nutzung des ANT Protokolls und der Hardware des Development Kits erweitert das Spektrum an möglichen Applikationen im Bereich des Sports enorm. Gerade im Bereich des Leistungssports dürfen Feedback- und Messsysteme nicht durch Gewicht, Kabel oder andere Eigenschaften störend wirken. Mit der ANT Technologie ist es möglich solche Systeme zu entwickeln.

Der technische Fortschritt im Bereich der Datenverarbeitung und Sensortechnologien wird in Zukunft die Möglichkeiten zur Verbesserung des Trainings erweitern und so Sportlerinnen und Sportlern helfen, ihr Potential auszuschöpfen. Sportwissenschaftlerinnen und Sportwissenschaftler sollten in der Lage sein, diese Entwicklung zu verfolgen und deren Potential zu nutzen. Denn sie sind es, die die Grundlagen für den Erfolg von den Athletinnen und Athleten beisteuern.

Das System WiBiFe ist ein Schritt in diese Richtung und soll den Athletinnen und Athleten helfen, ihren Zielvorgang zu optimieren. Das System WiBiFe ist zum Zeitpunkt der Fertigstellung dieser Magisterarbeit im Stadium eines Prototyps. Um das System in der Praxis einzusetzen, muss die Sensorschaltung optimiert werden. Bei der Programmierung in LabVIEW wurde zugunsten der Lesbarkeit des Codes auf das direkte Verdrahten der Elemente verzichtet. Somit können „Race Conditions“ und auch Performanceeinbußen entstehen. Die Integration weiterer Sensoren, Laufzeit- und Programmanalysen der Software und deren Weiterentwicklung können als Ziele für die Zukunft genannt werden.

Schlagwortverzeichnis

Abbildungsverzeichnis

Literaturverzeichnis

ANT Wireless. (2008). *Technology*. Zugriff am 22. Dezember 2008 unter http://www.thisisant.com/technology

Baca, A. & Kornfeind, P. (2006). Rapid feedback systems for elite sports training. *IEEE Pervasive Computing*, *5*(3), S. 70-76.

Baca, A. (2008). Feedback systems. In P. Dabinichki & A.Baca (Hrsg.) *Computers in Sport*. (43-65).Southampton: WIT Press.

Böhm, O. (2006). *C++ für Schnelleinsteiger*. Poing: Franzis Verlag GmbH.

Cisco Systems Inc. (2000). *Internetworking Technologies Handbook. Third Edition*. Indianapolis: Cisco Press.

Datacom. (2008). *ITWissen. NVM (non-volatile memory).* Zugriff am 28.02.2009 unter http://www.itwissen.info/definition/lexikon/Nichtfluechtiger-Speicher-NVM-non-volatile-memory.html

Dynastream Innovations Inc. (2008a). *ANT Message and ProtokollUsage*. Zugriff am 22. Dezember 2008 unter http://www.thisisant.com/images/Resources/PDF/1204662412_ant%20message%20protocol%20and%20usage%20rev%202.12.pdf

Dynastream Innovations Inc. (2008b). *AT3 RF Transceiver Module*. Zugriff am 25. Dezember 2008 unter http://www.thisisant.com/images/Resources/PDF/1211572185_at3_rf_transceiver_module_datasheet_rev%202.2.pdf

Dynastream Innovations Inc. (2008c). *SensRcore™ Messaging and Usage*. Zugriff am 25. Dezember 2008 unter http://www.thisisant.com/images/Resources/PDF/1186013376_ant%20sensrcore%20messaging%20and%20usage%20rev1.4.pdf

Georgi, W. & Metin, E. (2007). *Einführung in LabVIEW*. Leipzig Carl Hanser Verlag.

Haeberlin, V. (1999). *Schnittstellenkonzepte in Tumordokumentationssystemen.* Dissertation, Justus-Liebig-Universität Giessen.

Holzinger, A. (2002). *Basiswissen Multimedia. Band 1: Technik*. Würzburg: Vogel Verlag und Druck GmbH.

Hunt, C. (2003). *TCP/IP Netzwerk-Administration. 3. Auflage.* Köln: O'Reilly Verlag GmbH & Co. KG.

Husemann, D. & Nidd, M. (2005). Pervasive Patient Monitoring – Take Two at Bedtime. In European Reasearch Consortium for Informatics an Mathematics (Hrsg.) *ERICIM News – Special: Biomedical Informatics, 2005*(60), S. 70-71.

International Biathlon Union. (1998). *Event & Competition Rules*. Zugriff am 01. März .2009 unter http://www.ibu.at/rules/rules.pdf

Kleine, M. (2007). *SelfLinux – Das OSI Referenzmodell.* Zugriff am 07. Juli 2009 unter http://www.selflinux.org/selflinux/html/osi.html

Leiner, M. B., Cerf, G. V., Clark, D. D., Kahn, E. R., Kleinrock, L., Lynch, D.C., Postel, J., Roberts, L.G. & Wolf, S. (2003). *A Brief History of the Internet.* Zugriff am 07. Juli 2009 unter http://www.isoc.org/internet/history/brief.shtml

Marschall, F. & Daugs, R. (2003). Feedback. In H. Mechling & J. Munzert (Hrsg.), *Handbuch der Bewegungswissenschaft – Bewegungslehre* (S. 281 – 295). Schorndorf: Verlag Hofmann.

Plate, J. (2007). *Grundlagen Computernetze*. Zugriff am 22.Dezember 2008 unter http://www.netzmafia.de/skripten/netze/netz0.html#0.1

Prabhu, C. S. R & Prathap Reddi, A. (2004). *Buches Bluetooth Technology and Its Applications with JAVA and J2ME*. New Delhi: Prentice-Hall of India Pvt.Ltd.

Schwarz, A. (2008). *Mikrokontroller*. Zugriff am 28. Februar 209 unter http://www.mikrocontroller.net/articles/Mikrocontroller

Tekscan. (2005). *Flexiforce Sensor User Manual*. Zugriff am 08. Februar 2008 unter http://www.tekscan.com/pdfs/FlexiforceUserManual.pdf

Tekscan. (2007). Flexiforce Sensors. Zugriff am 25. August 2009 unter http://www.tekscan.com/flexiforce/flexiforce.html

Zhao, F. & Guibas, L. (2004). *Wireless Sensor Networks: An Information Processing Approach.* San Fransisco: Morgan Kaufman

Anhang

Kurzfassung

Aktuelle Entwicklungen im Bereich der Computertechnik ermöglichen es, drahtlose Feedback- und Messsysteme im Bereich des Sports zu entwickeln. Das ANT Protokoll und die SensRcore Technologie Produkte dieses Trends. Diese Magisterarbeit zeigt die Integration des ANT Protokolls und der Hardware in die Entwicklungsumgebung LabVIEW. Die Programmierung und technische Details zur Bereitstellung der Schnittstellen werden erörtert. Abschließend werden diese Technologien zum Erstellen eines drahtlosen Feedbacksystems zur Optimierung des Zielvorganges im Biathlonsport genutzt. Die zur Programmierung verwendeten Werkzeuge sind LabVIEW und Visual C++. Der Leserin bzw. dem Leser wird der Umgang mit diesen Technologien und Werkzeugen erläutert um so eigene Feedbacksysteme erstellen zu können.

Abstract

Current trend in hardware and computer technology allow the development of wireless feedback and measurement systems designed to improve athletic performance. The ANT network protocol and the SensRcore technology reflect this trend. This paper shows the integration of the ANT network protocol and SensRcore Hardware in the development environment LabVIEW. The programming process and technical details to create the necessary interfaces between those technologies are described in this thesis. The usage of the created programs is described within the development of a feedback system to optimize the target activity in biathlon. LabVIEW and Visual C++ were used to program those interfaces. The reader should be enabled to create own applications using the technologies described within this paper.

Curriculum Vitæ
Martin Böcskör, Bakk. rer. nat.

Persönliche Daten

Anschrift	Wienerstraße 84a, 7400 Oberwart
Telefon	+43660/6867970
Geburtsdaten	06.09.1980 in Oberwart
Familienstand	ledig

Schulische Ausbildung

1994 – 1995	HTBL Pinkafeld (EDV)
1996 – 1999	Handelsschule Oberwart
2001 – 2003	(berufsbegleitend) Berufsreifeprüfung

Studium

2003 – 2007	Bakkalaureatsstudium Gesundheitssport
seit SS2007	Magisterstudium Sportwissenschaft
seit WS2008	Bakk. Informatik (Universität Wien)

Berufliche Tätigkeiten

1999 – 2000	Bundesheer
2000 – 2001	Techniker bei Apollo Media AG
2001 – 2003	Premiere GmbH
seit 01.03.06	Studienassistent am ZSU (Abteilung Biomechanik/Bewegungswissenschaft und Sportinformatik)
seit 06.2008 – 12.2008	Systemadministrator am Zentrum für Sportwissenschaft/Universität Wien

Quellcode antwrapper.dll

```
/*
antwrapper.dll - A wrapper DLL that provides LabVIEW an Interface to the ANT_DLL.Dll, to receive
    Data (eventdriven) from a ANT Module.

Copyright (C) 2008 Martin Boecskoer

This program is free software; you can redistribute it and/or modify it under the terms of the GNU
    General Public License as published by the Free Software Foundation; either version 3 of the Li-
    cense, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY;
    without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR
    PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not,
    see <http://www.gnu.org/licenses/>.
*/

// antwrapper.cpp : Definiert die exportierten Funktionen für die DLL-Anwendung.
//

#include "stdafx.h"
#include "stdio.h"
#include "extcode.h" //labview inlcude
#include "antmessage.h" //ant inlcude

//Definitionen der Laenge für die Buffer
#define RESPONSE_LAENGE 5
#define ANT_RESPONSE 7
#define EVENT_LAENGE 11
#define ANT_MESSAGE 13 //max. Länge Ext.Message - channel und id

//Typendifinitionen für die ANT Funktionen
typedef BOOL (*ptreventfunction)(UCHAR ucChannel, UCHAR ucEvent);
typedef BOOL (*ptrresponsefunction)(UCHAR ucChannel, UCHAR ucMsgId);

typedef void (*ANT_RESPONSE_POINTER)(ptrresponsefunction, UCHAR*);
typedef void (*ANT_EVENT_POINTER)(UCHAR, ptreventfunction, UCHAR*);

//ANT Buffer & Globals

LVUserEventRef *eventref;
LVUserEventRef *responseref;
```

```
//Buffer der Länge ANT_MESSAGE werden leer initialisiert
UCHAR ResponseBuffer[RESPONSE_LAENGE]={};
UCHAR ChannelEventBuffer[EVENT_LAENGE]={}; //max Länge Response = 11

LStrHandle eventStringHandle;
LStrHandle responseStringHandle;

//Funktionsdeklarationen
BOOL ANT_ChannelEventFunction(UCHAR ucChannel, UCHAR ucEvent);
BOOL ANT_ResponseFunction(UCHAR ucChannel, UCHAR ucMsgId);
void SendResponse(UCHAR ucCHannel, UCHAR ucMsgId);
void SendEvent(UCHAR ucChannel, UCHAR ucEvent);



extern "C" __declspec(dllexport) int ANT_LabView_Schnittstelle(UCHAR antChannel, LVUserE-
    ventRef *eventreflabview, LVUserEventRef *responsereflabview)
{
    int ok = 0;

    //Übergaben von den LabviewRefs an Globale Variable
    eventref=eventreflabview;
    responseref=responsereflabview;

  //Eventstringhandle Speicher zuweisen
    responseStringHan-
    dle=(LStrHandle)DSNewHandle(sizeof(int32)+ANT_RESPONSE*sizeof(UCHAR));
    LStrLen(*responseStringHandle)=ANT_RESPONSE;
    eventStringHan-
    dle=(LStrHandle)DSNewHandle(sizeof(int32)+ANT_MESSAGE*sizeof(UCHAR));
    LStrLen(*eventStringHandle)=ANT_MESSAGE;

  //Zeiger auf die ANT_Funktionen definieren
  ANT_RESPONSE_POINTER ANT_AssignResponseFunction;
  ANT_EVENT_POINTER ANT_AssignChannelEventFunction;

  //ANT_DLL laden
  HINSTANCE hANTdll = LoadLibraryA("ANT_DLL.dll");
  if (hANTdll != NULL){

      //Pointer auf die Funktionen setzen
      ANT_AssignResponseFunction = (ANT_RESPONSE_POINTER) GetProcAddress(hANTdll,
    "_ANT_AssignResponseFunction");
      ANT_AssignChannelEventFunction = (ANT_EVENT_POINTER) GetProcAddress(hANTdll,
    "_ANT_AssignChannelEventFunction");
  }
```

```
      //Funktionen aufrufen
      if(ANT_AssignResponseFunction != NULL  && ANT_AssignChannelEventFunction != NULL)
          {
                  ANT_AssignResponseFunction((ptrresponsefunction)ANT_ResponseFunction, Res-
    ponseBuffer);
                  ANT_AssignChannelEventFunction(antChannel, (ptreventfunc-
    tion)ANT_ChannelEventFunction, ChannelEventBuffer);
                  ok=1;
          }

      else
           ok=0;


  //ANT_DLL freigeben
  FreeLibrary(hANTdll);

    return ok;
}

//ChannelEventFunction
BOOL ANT_ChannelEventFunction(UCHAR ucChannel, UCHAR ucEvent)
{

    SendEvent(ucChannel, ucEvent);
    return TRUE;
}

//ResponseFunction
BOOL ANT_ResponseFunction(UCHAR ucChannel, UCHAR ucMsgId)
{

    SendResponse(ucChannel, ucMsgId);
    return TRUE;
}

void SendEvent(UCHAR ucChannel, UCHAR ucEvent){
    //aucChannelEventbuffer --> LStrBuf
    memcpy((char*)LStrBuf(*eventStringHandle), &ucChannel, 1);
    memcpy((char*)LStrBuf(*eventStringHandle)+1, &ucEvent, 1);
    memcpy((char*)LStrBuf(*eventStringHandle)+2, ChannelEventBuffer,ANT_MESSAGE);

    //UserEvent+EventDaten an Labview
    PostLVUserEvent(*eventref, (void*)&eventStringHandle);
}
```

```
void SendResponse(UCHAR ucChannel, UCHAR ucMsgId){

    //Daten der ResponseFunction --> LStrBuf
    memcpy((char*)LStrBuf(*responseStringHandle), &ucChannel, 1);
    memcpy((char*)LStrBuf(*responseStringHandle)+1, &ucMsgId, 1);
    memcpy((char*)LStrBuf(*responseStringHandle)+2, ResponseBuffer,ANT_RESPONSE);

    //LStHandle Response an Labview
    PostLVUserEvent(*responseref, (void*)&responseStringHandle);
}
```

SensRcore Konfigurationsdatei – wibife.txt

```
C [68][00][01]                          // Enabling LED on RF events
C [93][00][47][01]                      // Setting A/D Sample Rate at 100 Hz
C [91][01][00][41][00][01][00]          // Configuring Data Channel 0 on ANT Channel 1 as Analog
                                           Input using pin(s) 00 reporting on 100.00% of messages
C [94][00][00][01][01][08][00][00]      // Configuring Data Channel 0 A/D with reference setting of
                                           1, sampling at 1.00 of sample rate, filter N value of 8
C [42][01][10][00]                      // Assign Channel Master on ANT chanel 1 on network 0
C [51][01][31][00][01][05]              // Assign Channel ID 1, Dev. Num = 49, Dev. Type = 1,
                                           Trans. Type = 5
C [43][01][48][01]                      // Set ANT Channel 1 Message Period to 100.00 Hz
C [45][01][42]                          // Set ANT Channel 1 RF Frequency to 2466 MHz
C [4B][01]                              // Open ANT chanel 1
C [91][00][00][40][00][01][00]          // Configuring Data Channel 0 on ANT Channel 0 as Digital
                                           Input using pin(s) 00 reporting on 100.00% of messages
C [93][00][00][20]                      // Setting A/D Sample Rate at 4 Hz
```

CD Ordnerstruktur

Ordner	Inhalt
\Sources\LabVIEW \Sources\Wrapper \Sources\ANT_conifg	Quellcodes WiBiFi, Eusociality Control Center Tools, Antwrapper.dll sowie das SensRcore Konfigurationsfile.
\DLL\	Hier finden Sie die kompilierte Form der Antwrapper.dll, sowie die Hersteller (ANT) DLL's
\ANT_Docs\	Dieser Ordner beinhaltet alle Dokumente zum Umgang mit der ANT Technologie als PDF.
\Anleitungen\	Hier finden Sie Videoanleitungen, die die Installation von Eusosiality Control Center und der nötigen Bibliotheken in LabVIEW beschreiben.