Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Seminararbeit

Sebastian Gregor

Physical Interaction Design

Fakultät Technik und Informatik
Studiendepartment Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Sebastian Gregor

Physical Interaction Design

Seminararbeitarbeit eingereicht im Rahmen der Seminararbeitprüfung
im Studiengang Informatik
am Studiendepartment Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer : Prof. Dr. Kai v. Luck

Abgegeben am 28. Februar 2009

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einleitung

1.1 Motivation

Das Schaffen von dynamischen, mit den Betrachtern interagierenden Kunstobjekten und das Ausprobieren von neuen Techniken und Ausdrucksmöglichkeiten üben einen starken Anreiz auf Künstler aus. Der Autor hat während des Masterprojektes [15], sowieder Teilnahme an einem Wearable Computing Workshop im Rahmen von Pentiment 2008 [21] mit Designern und Künstlern zusammen gearbeitet. Dabei wurde das große Interesse von Kunstschaffenden an interaktiven Kunstinstallationen deutlich. Gleichzeitig hat sich während der Zusammenarbeit gezeigt, dass Künstler über wenige bis keine Erfahrungen im Umgang mit Elektronik, sowie Programmierung verfügen und technische Schaltungen schlecht abstrahieren können. Um dynamisches Verhalten bei Kunstobjekten zu ermöglichen und die gewünschte Interaktivität zu erreichen ist der Einsatz von unterschiedlichen elektronischen Bauteilen und Computern oder Mikroprozessorsystemen nötig.
Bei der Modellierung von Interaktionen zwischen Mensch und Computer spricht man in der Informatik von Interaction Design. Während Interaction Design vor allem für die Interaktionen im Desktop/PC Umfeld steht, haben sich für die Modellierung von Interaktionen zwischen Menschen und kleinen (eingebetteten) Computerplattformen die Begriffe Physical Computing und Physical Interaction Design (s. Kapitel 2) etabliert. Hier findet die Interaktion nicht über Monitor, Maus und Tastatur, sondern mit Hilfe von Sensoren, wie z. B. Gyroskope, Drucksensoren oder Abstandssensoren, und Aktoren statt.

1.2 Zielsetzung und Gliederung

Ziel dieser Arbeit ist es, die verwendeten Techniken und Technologien einiger ausgewählter Physical Interaction Design Toolkit's und Projekte zu betrachten, welche das Potential haben Kunstschaffenden den Einstieg in die Verwendung von Sensoren und Physical Interaction Design zu erleichtern. Zu diesem Zweck wird in Kapitel 2 der Begriff Physical Interaction Design definiert. In Kapitel 3 werden dann die Toolkits und Projekte näher betrachtet. Den Abschluss bildet Kapitel 4 mit einer Zusammenfassung.

2 Physical Interaction Design

In diesem Kapitel wird der Begriff Physical Interaction Design definiert. Dazu wird im ersten Teil das Forschungsgebiet Interaction Design beschrieben, um dann mit Hilfe der Definition von Physical Computing den Schritt zum Physical Interaction Design zu machen.

2.1 Interaction Design

„In the next fifty years, the increasing importance of designing spaces for human communication and interaction will lead to expansion in those aspects of computing that are focused on people, rather than machinery. The methods, skills, and techniques concerning these human aspects are generally foreign to those of mainstream computer science, and it is likely that they will detach (at least partially) from their historical roots to create a new field of ‚interaction design.'" [27]

Interaction Design ist eine Teildisziplin der Informatik, die sich mit der Beziehung zwischen Benutzern und interaktiven Produkten beschäftigt. Wie Terry Winograd in dem Zitat am Anfang des Kapitels beschreibt, stellt Interaction Design den Benutzer und die Benutzerszenarien in den Mittelpunkt der Entwicklung. In der Literatur gibt es eine Vielzahl von unterschiedlichen Definitionen von Interaction Design. Jonas Löwgren geht in seinem Artikel [14] von zwei unterschiedliche Ansichtsweisen von Interactive Design aus, die in der Forschung und praktischen Anwendung zusammenfließen. Auf der einen Seite spricht Löwgren von Sicht auf Interaction Design als Design Disziplin, auf der anderen Seite als eine Erweiterung von Mensch-Maschinen Interaktion oder Human-Computer Interaction (HCI) zu sehen. HCI ist eine Disziplin deren Ursprung bereits auf das Jahr 1960 zurückgeht und die sich mit unterschiedlichen Themen aus Ingenieurwissenschaften und Verhaltensforschung sowie deren Zusammenwirken beschäftigt [1]. In der Informatik liegt der Fokus auf der Interaktion zwischen Menschen und Computer. Nach [14] liegt das Hauptaugenmerk von HCI auf den instrumentellen Qualitäten, wie Benutzbarkeit (engl. Usability) und Nützlichkeit von digitalen Produkten und Diensten. Dabei werden überwiegend berufsbedingte oder aufgabenorientierte Gebrauchssituationen, mit dem Fokus auf individuelle Benutzer und ihren Absichten, betrachtet. Fragen, wie ein Mensch Informationen aufnimmt und wie er diese verarbeitet,

nehmen eine zentrale Rolle ein. Bei Interaction Design als Erweiterung von HCI, werden aus der HCI-Forschung gewonnenen Erkenntnisse und Werkzeuge bereits im Designprozess mit eingebunden, anstatt nach dem Designentwurf auf Usability-Probleme hinzuweisen.
Die Benutzung von digitalen Produkten und Diensten hat sich stark verändert. Von einem reinen arbeitsrelevanten Einsatz erfolgt ein immer stärker werdender Wandel hin zu einem Einsatz in der Informations- und Unterhaltungsindustrie. Dadurch fliessen in die Betrachtungen immer mehr nicht technische, ästhetischen und ethische Qualitäten aus der Sichtweise Interaction Design als Design Disziplin mit ein und werden zu gleichen Teilen betrachtet.

2.2 Physical Computing

Durch Ubiquitous Computing [26] hat sich der Umgang mit interaktiven Computersystemen drastisch verändert. Neue technische Möglichkeiten führen dazu, dass Computer kleiner, verfügbarer und vernetzter werden. In diesem Zusammenhang sind Begriffe wie Ambient, Tangible, Pervasiv, Embedded, Spatial oder Physical Computing aufgekommen [20]. Physical Computing steht hier für das Erfassen von Daten der physischen Welt und deren Verarbeitung durch Computer [19]. Zum Erfassen der Daten beschäftigt sich Physical Computing mit dem Auslesen und Verarbeiten der unterschiedlichsten Sensordaten wie z. B. Gyroskope, Kameras, einfache Schalter oder Bewegungssensoren. Zur Verarbeitung der Daten werden häufig mikrokontrollergesteuerte Hardware - Plattformen, PDA's oder Handy's eingesetzt, die als Ausgabegeräte unterschiedliche Aktoren wie z. B. Servomotoren, Vibrationsmotoren oder Lautsprecher ansprechen. Durch Verknüpfung von Sensoren und Aktoren mit physischen Objekten und die dadurch entstehenden Interaktionsmöglichkeiten wird der physische Raum zu einem Ein- und Ausgabemedium für die Informationsverarbeitung. Dies führt zu einer Verbindung des medialen Raumes mit dem physischen Raum [23].

2.3 Physical Interaction Design

Interaction Design beschäftigt sich vor allem mit dem Modellieren von Interaktionen in der Desktop/PC - Umgebung. In dieser Umgebung werden hauptsächlich der Maus und der Tastatur als Eingabe- und dem Monitor als Ausgabegerät genutzt. Physical Interaction Design hingegen ist Interaction Design aus der Perspektive von Physical Computing.

„...physical objects have a sensory richness of meaning that screen-based elements do not. When we see, hear and feel real-world objects we are enabled to train both cognitive and perceptual skills in combination.“ [20]

Der Einsatz und die Techniken des Physical Computing ermöglichen es, wie Panagis Papadimatos in seinem Zitat schreibt, mehr Sinne eines Menschen anzusprechen als beim interagieren mit Desktop/PC - ähnlichen Umgebungen. Dadurch wird es möglich, neue Dimensonen von Interaktionsdesign zu erforschen.

3 Tools

Es gibt eine Vielzahl unterschiedlicher Tools und Projekte im Bereich Physical Interaction Design. Dabei werden viele verschiedene Strategien und Implementierungsansätze verfolgt. Im folgenden Kapitel werden exemplarisch für die unterschiedlichen Herangehensweisen einige Projekte genauer vorgestellt.

3.1 Phidgets

Phidgets wurde 2001 von Chester Fitchett and Saul Greenberg [9] als ein Forschungsprojekt der University of Calgary gestartet. Der Name des Projektes setzt sich aus den beiden englischen Wörtern Physical und Widget[1] zusammen. Diese Namenswahl lässt bereits das Ziel des Projektes erkennen, kleine technische Bauteile zur Erstellung von physischen Benutzungsschnittstellen zu entwickeln.

„... just as widgets make GUIs easy to develop, so could phidgets make the new generation of physical user interfaces easy to develop.“ [9]

Abbildung 3.1: Phidget Servo Kit

Bei der Entwicklung der Hardwarebestandteile wurde das Hauptaugenmerk auf einen einfachen Aufbau gelegt. Dadurch können sich Entwickler auf die Modifikation und die Orchestrierung der Bauteile zu physischen Benutzungsschnittstellen konzentrieren, anstatt sich mit der Konstruktion der Bauteile beschäftigen zu müssen. Jedes der Hardwarebauteile (Abbildung 3.1 zeigt exemplarisch ein Phidget zur Steuerung eines Servomotors) ist ein eigenständiges Modul, das über eine Schnittstelle nach dem USB[2]-Standard mit einem PC verbunden wird. Der Benutzer benötigt aufgrund dieses Aufbaus nur ein geringes Vorwissen im Bereich der Elektronik, um die einzelnen

[1] Interaktionselement einer grafischen Benutzeroberfläche
[2] USB: Abkürzung für Universal Serial Bus, einem seriellen Bussystem

Module verwenden zu können. Für die Ansteuerung eines Moduls wurde eine API entwickelt. Abbildung 3.2 zeigt den schematischen Aufbau der Phidget Architektur. Der schwarz umrandete Bereich auf der linken Seite symbolisiert den Aufbau der Hardwarekomponenten, mit deren Hilfe der Entwickler eine physischen Benutzungsschnittstellen aus Sensoren und Aktoren aufbauen kann. Der schwarz umrandete Bereich auf der rechten Seite stellt den Aufbau der API dar, mit deren Hilfe der Entwickler die Software zum Verarbeiten und Steuern der Phidgets erstellen kann. Die API ist in vier Schichten gegliedert. Der Communication-Layer ermöglicht den anderen Schichten eine Kommunikation mit den Phidgets per USB. Der Phidget Manager beinhaltet eine API, mit deren Hilfe die Präsenz von Phidgets und der Typ des verbundenen Phidgets erkannt wird. Der Phidget Manager generiert ein für den angeschlossenen Phidget spezifisches COM[3] Objekt. Dieses Objekt kommuniziert mit dem

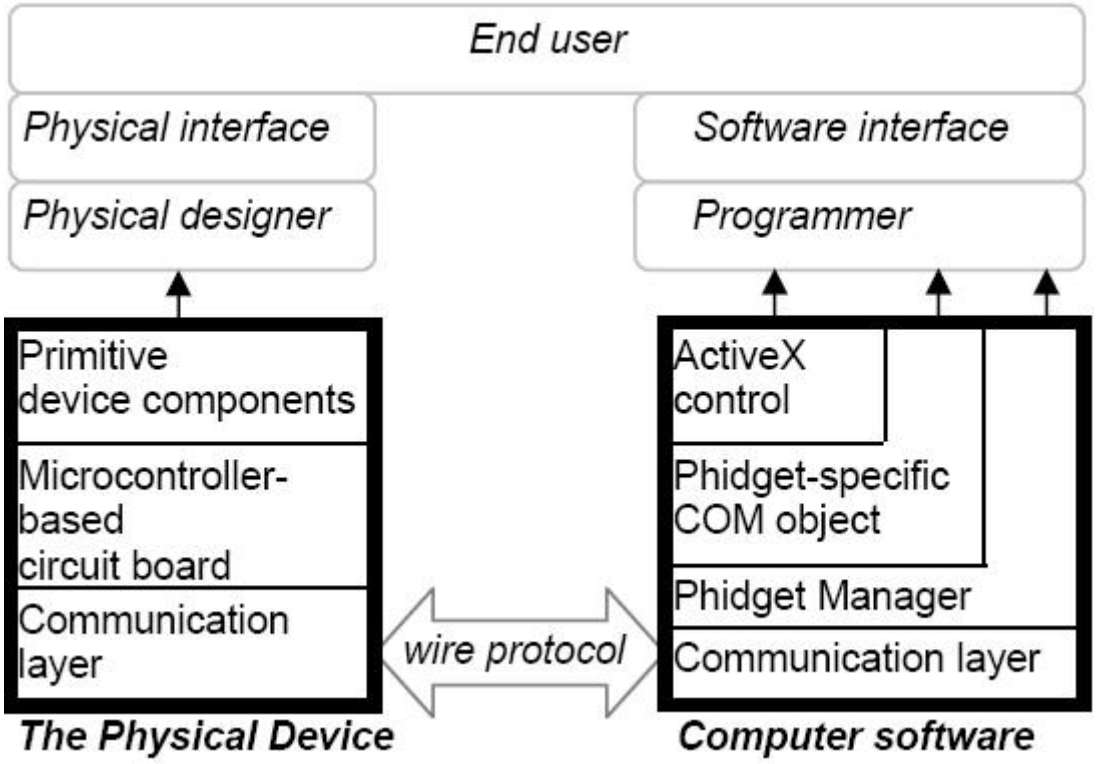

Abbildung 3.2: Phidget Architektur

physischen Phidget über den Phidget Manager. Über das ActiveX[4] Control lassen sich die einzelnen Phidgets auf dem Bildschirm visualisieren. Gleichzeitig wird dem Entwickler die Möglichkeit gegeben, die einzelnen Phidgets zu simulieren. Es ist dem Entwickler überlas-

[3] COM: Von Microsoft entwickleltes Component Object Model zur Interprozesskommunikation und dynamische Objekterzeugung.
[4] ActiveX: Erweiterung des COM zur Darstellung aktiver Inhalte.

sen, ob die physischen Phidgets durch die COM- oder durch die ActiveX-Objekte gesteuert werden.

3.2 d.tools

D.tools ist ein Toolkit, das 2005 von Hartmann u. a. [12][13] von der HCI Group der Stanford Universität entwickelt wurde. Ziel des Projektes ist die Entwicklung eines Toolkits zum Entwerfen, Programmieren und Herstellen von eigenen elektronischen Geräten. Zielgruppe der Entwicklung sind Produkt- und Interaktionsdesigner. Diese Gruppe verfügt über das Wissen des Herstellungsprozesses von Geräten, die Erstellung von Inhalten und den Entwurf von Interaktionsmöglichkeiten des Menschen mit einem Gerät. Das Wissen, die benötigten elektronischen Komponenten zu verbauen und zu programmieren, fehlt jedoch meistens. Als Entwicklungsergebnis steht ein Toolkit aus Hardwarekomponenten und eine grafischen Entwicklungsumgebung zur Verfügung. Die Entwicklungsumgebung unterstützt neben dem Design auch das Testen und Analysieren einer physischen Benutzungsschnittstelle. Die Hardware

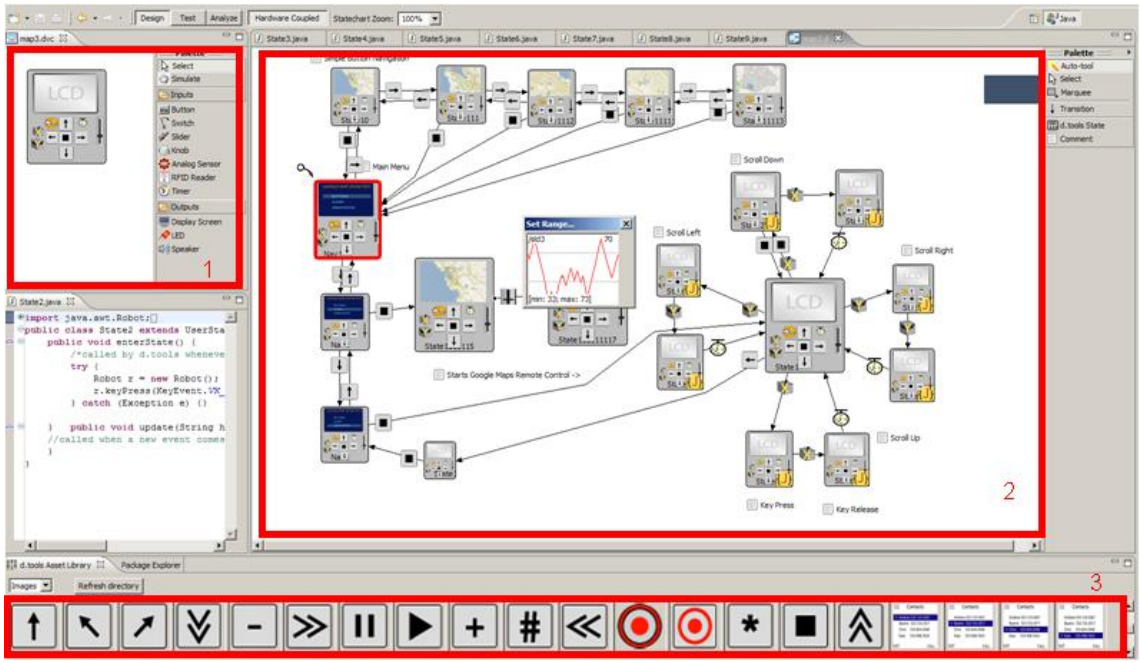

Abbildung 3.3: d.tools - Entwicklungsumgebung

des Toolkits besteht aus einer Reihe von Aktoren und Sensoren. Jeder verwendete Aktor oder Sensor besitzt einen kleinen Mikrokontroller vom Typ ATtiny der Firma Atmel [3] und

wird über ein Kabel an den I2C - Bus[5] der Hauptplatine angeschlossen. Für die Verarbeitung auf der Hauptplatine wird ein ATmega128 Mikrokontroller der Firma Atmel verwendet. Dieser Mikrokontroller realisiert ebenfalls die Kommunikation der Hauptplatine mit der d.tools - Entwicklungsumgebung auf dem PC. Zur Übertragung der Daten vom Computer zur Hauptplatine wird das Open Sound Control Protokoll[6] verwendet. Die d.tools Hardware kann nicht käuflich erworben werden. Die d.tools Entwicklungsumgebung kann aber mit auch mit der Hardware von Phidget (s. Kapitel 3.1), Wiring und Arduino (s. Kapitel 3.3) verwendet werden. Dafür muss die Hardware mit einer entsprechenden Firmware ausgestattet werden, um mit der d.tools Entwicklungsumgebung kommunizieren zu können. Dieses Vorgehen erfordert jedoch zustätzliche Hardware zum Schreiben der Frimware auf den jeweilige Mikroprozessoren.
Die Entwicklungsumgebung (siehe Abbildung 3.3) ist als Eclipse[7] Plugin realisiert und gliedert sich in drei Hauptbestandteile:

- Device Designer:
  Der Device Designer (in Abbildung 3.3 mit '1' gekennzeichnet) erkennt automatisch die an der Hauptplatine angeschlossenen Sensoren, Aktoren und Ausgabegeräte. Diese werden als Icons dargestellt und können zu einer Art Bauplan des Devices zusammengefügt werden. Darüberhinaus können weitere virtuellen Komponenten hinzugefügt und später mit der Hardware synchronisiert werden.

- Statechart Editor:
  Der Statechart Editor (in Abbildung 3.3 mit '2' gekennzeichnet) dient der Modellierung des Verhaltens eines Prototypen mittels Interaktionsgraphen. Dazu werden die vorher im Device Designer erstellten grafischen Instanzen als Zustände in einem Zustandsautomaten verwendet. Die Instanzen werden per Drag and Drop aus dem Device Designer in den Statechart Editor eingefügt und mittels Transitionen zu einem Graphen verbunden. Diesen Transitionen werden Hardwareevents zugeordnet und mit Icons der eventauslösenden Hardware markiert. Zur besseren Übersichtlichkeit lassen sich Teilgraphen im Statechart Editor definieren.

- Asset Library:
  Die Asset Library ist eine Bibliothek des verfügbaren Contents, die in einem separaten Fenster (in Abbildung 3.3 mit '3' gekennzeichnet) dargestellt wird. Der Content kann einem Zustand und damit den im Device Designer definierten Aktor oder Ausgabegeräte zugewiesen werden. Dazu muss das entsprechende Symbol aus dem Asset Library Fenster auf den Zustand im Statechart Editor gezogen wird.

[5] serieller Datenbus - http://www.i2c-bus.org/
[6] Nachrichten-basiertes Kommunikationsprotokoll - http://archive.cnmat.berkeley.edu/OpenSoundControl/
[7] Open-Source-Entwicklungsumgebung - http://www.eclipse.org

Nachdem mit Hilfe der beschriebenen Komponenten das Verhalten des elektronischen Gerätes als Zustandsautomat modelliert wurde, kann das Verhalten getestet und analysiert werden. Zum Testen bietet d.tools die Möglichkeit an, alle Hardwareevents eines Testlaufes aufzuzeichnen. Diese werden dann automatisch mit den Transitionen verknüpft. Auf diese Weise erhält man nach mehreren Testläufen einen Überblick ob und welche unterschiedlichen Hardwareevents evtl. die gleichen Transitionen angestoßen haben. Darüber hinaus werden bei einer angeschlossenen Videokamera die Bildabschnitte, welche während der Test mit der Videokamera aufgenommen worden sind, automatisch den unterschiedlichen Transitionen und Hardwareevents chronologisch zugeordnet. Dadurch bietet d.tools die Möglichkeit Usability - Tests durchzuführen und die Interaktionen der Benutzer mit dem Geräte zu analysieren.

3.3 Arduino

Arduino ist ein Open-Source-Projekt auf der Basis von Wiring [4]. Wiring wurde 2003 von Hernando Barragán als Prototyping - Tool zum Einstieg von Künstlern in das Themengebiet elektronische Kunst entwickelt. Es basiert auf einer Mikroprozessorplatform mit einem AVR - Mikrokontroller und eine Entwicklungsumgebung auf Basis von Processing [22].

„It has always been our aim to encourage the traditional design disciplines to go beyond the screen, trying to give meaning to humanmachine interactivity through actually designing the machine itself.“ [16]

Im Jahr 2005 entstand auf Basis von Wiring das interanationale Arduino Projekt. Das Ziel des Projektes ist es, wie im Zitat aus [16] beschrieben, Menschen aus unterschiedlichen Designdisziplinen eine Plattform für den Einstieg in das Themengebiet Physical Interaction Design zur Verfügung zu stellen. Zu diesem Zweck wurde eine Open - Source Hardwareplattform definiert und eine dazu passende Open - Source Enwicklungsumgebung entwickelt. Durch die enge Verzahnung von Hardware und Software ist es möglich, den Benutzern die meisten Konfigurationsschritte bei der Inbetriebnahme und Programmierung zu ersparen. Dadurch lassen sich kleine Programme für die Arduino Hardware schreiben ohne das der Entwickler spezielle Kentnisse über den verwendeten Mikrokontroller oder fundierte Erfahrungen bei der Programmierung von Mikrokontrollern verfügen muss.

Als Hardware kommen bei Arduino kleine 8 Bit-Mikroprozessorsysteme zum Einsatz. Zum Zeitpunkt der Erstellung dieser Arbeit existiert eine Vielzahl unterschiedlicher Arduino - kompatibler Hardware, wie beispielsweise der in Abbildung 3.4 gezeigten Arduino Basishardware. Über eine RS232[8]- oder USB-Schnittstelle kann die Arduino Hardware an einen PC angeschlossen werden. Damit ein Mikroprozessorsystem Arduino kompatibel ist, muss es über einen speziellen Bootloader[9] verfügen. Dieser Bootloader ermöglicht das Programmieren des Mikroprozessorsystems durch die Arduino Entwicklungsumgebung ohne das eine spezielle Programmierhardware benötigt wird. Die Arduino Hardware benötigt, nachdem sie programmiert wurde, keine Verbindung zu einem PC um zu funktionieren.

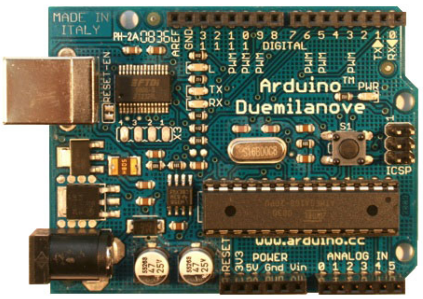

Abbildung 3.4: Arduino Duemilanove

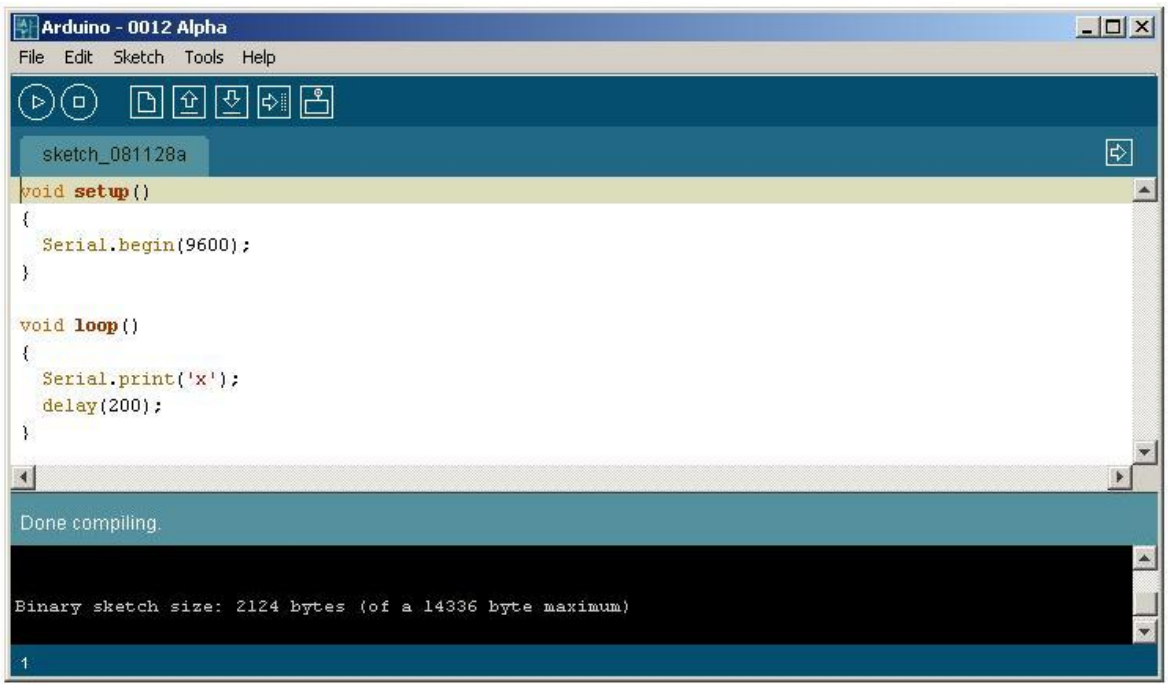

Abbildung 3.5: Arduino - Entwicklungsumgebung

[8] RS232: Bezeichnet einen Standard für eine serielle Schnittstelle.

[9] spezielle Software zum Laden eines Programmes in den Speicher eines Mikrokontrollers während der Initialisierungsphase

Die Arduino Entwicklungsumgebung (siehe Abbildung 3.5) ist in Java[10] geschrieben und besteht aus zwei Hauptelementen:

- Source Code Editor:
  Mit dem auf der Processing Entwicklungsumgebung basierenden Source Code Editor kann der Benutzer Programme, die sog. Sketche editieren, kompilieren und auf die Arduino Hardware laden. Alle notwendigen Einstellungen, wie die verwendete Hardware und Schnittstelle, lassen sich vom Benutzer einstellen. Dies hat den Vorteil, dass der Benutzer weder die bei der Mikrokontrollerprogrammierung üblichen Makefiles, noch Command - Line Argumente für den Compiler bearbeiten bzw. editieren muss. Zum Kompilieren der erstellten Sketche durch den AVR-GCC[11] - Compiler reicht ein Klick auf die entsprechende Schaltfläche. Der Source Code Editor enthält desweiteren einen Serial Monitor mit dem eine serielle Verbindung zwischen Hardware und dem Editor hergestellt werden kann. Damit können Ausgaben von der Hardware empfangen und Befehle zur Hardware gesendet werden. Diese Funktionalität erlaubt ein einfaches Debuggen des Programmes auf Hardwareebene mittels Ausgabebefehlen im Programmcode.
- Core Library:
  Die Core Library besteht aus einer Reihe von AVR C/C++ - Funktionen. Diese Funktionen kapseln die mikrokontrollerspezifischen Punkte wie z. B. das Manipulieren von Registern bei der Programmierung. Die Core Library hat die Möglichkeit andere C/C++ - Bibliotheken einzubinden. Mit dieser Funktion lassen sich anwendungsspezifische Bibliotheken wie z. B. für die Ansteuerung eines Servomotors schreiben und der Arduino Entwicklungsumgebung hinzufügen. Diese Bibliotheken lassen sich über ein einfaches Menu im Source Code Editor in ein Programm einbinden und verwenden.

Die Core Library und der Source Code Editor stehen auf der Arduino Webseite für Windows, MacOs X und Linux zum Download zur Verfügung. Nach dem Entpacken ist die Entwicklungsumgebung sofort funktionsfähig, ohne das weitere Komponenten installiert werden müssen.

[10] objektorientierte Programmiersprache der Firma Sun Microsystems - http://java.sun.com
[11] AVR-GCC ist der Name der Compiler-Suite des GNU-Projekts - http://gcc.gnu.org/

4 Zusammenfassung und Ausblick

In diese Arbeit wurden drei unterschiedliche Physical Interaction Design - Projekte vorgestellt. Während des Betriebes benötigen die Hardwarekomponenten von Phidgets (s. Kapitel 3.1) und d.tools (s. Kapitel 3.2) immer eine Verbindung zu einem PC. Im Gegensatz dazu kann die Hardware des Arduino Projekt (s. Kapitel 3.3) auch autonom ohne einen PC betrieben werden. Bei der Programmierung sticht vor allem das d.tools - Projekt mit seiner grafischen Entwicklungsumgebung hervor. Diese erlaubt es Benutzern ohne Programmierkenntnisse die gewünschte Funktionalität zu modellieren. Obwohl die beiden anderen Projekte die hardwarenahen Aspekte der Programmierung mittels APIs kapseln und für den Benutzer transparent machen, sind grundlegende Programmierkenntnisse notwendig. Hier bietet Arduino allerdings den Vorteil einer Entwicklungsumgebung, die auf die verwendete Hardware zugeschnitten und ohne eine Installation von weiteren Komponenten funktionsfähig ist. Dies erleichtert den Einstieg in die Programmierung. Ein weiterer Vorteil bei Arduino ist die Open-Source Hardware. Diese Erlaubt es Entwicklern eigene Arduino Plattformen und Erweiterungen für bereits existierende Hardware zu entwickeln. Dies hat dazu geführt, dass eine sehr große Entwicklergemeinde mit einer umfangreichen Wissensbasis um das Andruino Projekt entstanden ist. Bei den anderen beiden Projekten ist dies nicht der Fall. Die Hardware des d.tools Projekt ist sogar überhaupt nicht käuflich zu erwerben.

Für den Einsatz in der interaktiver Kunst kommt von diesen drei Projekten am ehesten das Arduino Projekt in Frage. Es ist das Einzige, bei dem die Hardware sowohl mit, als auch ohne angeschlossenen PC funktioniert. Allerdings stellt die Programmierung eine sehr große Hürde für Künstler dar. Hier würde eine grafische Entwicklungs- und Testumgebung eine enorme Erleichterung bringen. In [11] beschreibt der Autor wie eine solche Entwicklungsumgebung aussehen könnte und mit welchen Techniken sie sich realisieren lassen würde.

Literaturverzeichnis

[1] ACM: *ACM SIGCHI Curricula for Human-Computer Interaction*. http://sigchi.org/cdg/cdg2.html. – (letztes Zugriffsdatum: 13.02.2009)

[2] ANDERSON, Gretchen: Let's get physical. In: *interactions* 15 (2008), Nr. 5, S. 68–72. – ISSN 1072–5520

[3] ATMEL: *Atmel Corporation - Homepage*. http://www.atmel.com/. – (letztes Zugriffsdatum: 22.02.2009)

[4] BARAGAN, Hernando: *Wiring: Prototyping Physical Interaction Design*, Interaction Design Institute Ivrea, Mastersthesis, 2004

[5] BUECHLEY, Leah: A Construction Kit for Electronic Textiles. In: *10th IEEE International Symposium on Wearable Computers* (2006), Oktober, S. 83 – 90

[6] BUECHLEY, Leah ; EISENBERG, Michael: The LilyPad Arduino: Toward Wearable Engineering for Everyone. In: *Pervasive Computing, IEEE* (2008), April, S. 12 – 15

[7] BUECHLEY, Leah ; ELUMEZE, Nwanua ; DODSON, Camille ; EISENBERG, Michael: Quilt Snaps: A Fabric Based Computational Construction Kit. In: *Proceedings of the 2005 IEEE International Workshop on Wireless and Mobile Technologies in Education*, 2005

[8] EISENBERG, Michael ; EISENBERG, Ann ; BUECHLEY, Leah ; ELUMEZE, Nwanua: Invisibility Considered Harmful: Revisiting Traditional Principles of Ubiquitous Computing in the Context of Education. In: *Wireless, Mobile and Ubiquitous Technology in Education* (2006), November, S. 103 – 110

[9] FITCHETT, Chester ; GREENBERG, Saul: The Phidget Architecture: Rapid Development of Physical User Interfaces. In: *Workshop ApplicationModels and Programming Tools for Ubiquitous Computing*, 2001

[10] GREENWOLD, Simon: *Spatial Computing*, Massachusetts Institute of Technology, Mastersthesis, 2003

[11] GREGOR, Sebastian: *Physical Interaction Design - Vision for a visual programming and simulation environment*, Hochschule für Angewandte Wissenschaften Hamburg, Seminararbeit, 2009

[12] HARTMANN, Björn ; KLEMMER, Scott R. ; BERNSTEIN, Michael ; ABDULLA, Leith ; BURR, Brandon ; ROBINSON-MOSHER, Avi ; GEE, Jennifer: Reflective Physical Prototyping through Integrated Design, Test, and Analysis. In: *Proceedings of UIST*, 2006

[13] KLEMMER, Scott R. ; HARTMANN, Bjoern ; TAKAYAMA, Leila: How Bodies Matter: Five Themes for Interaction Design. In: *Proceedings of DIS*, 2006

[14] LÖWGREN, Jonas: *Interaction Design*. http://www.interaction-design.org/encyclopedia/interaction_design.html. – (letztes Zugriffsdatum: 21.02.2009)

[15] MASTERPROJEKT: *Ambient Awareness - Homepage*. http://ambientawareness.org/. – (letztes Zugriffsdatum: 22.02.2009)

[16] MELLIS, David A. ; BANZI, Massimo ; CUARTIELLES, David ; IGOE, Tom: Arduino: An Open Electronics Prototyping Platform, 2007

[17] MOGGRIDGE, Bill: *Designing Interactions*. The MIT Press, 2007. – ISBN 978–0–26213–474–3

[18] NETZ, Medien K.: *Experiments in Art and Technology*. http://www.medienkunstnetz.de/kuenstler/eat/biografie/. – (letztes Zugriffsdatum: 21.02.2009)

[19] O'SULLIVAN, Dan ; IGOE, Tom: *Physical Computing*. Thomson Course Technology, 2004

[20] PAPADIMATOS, Panagis: *Physical Computing*, University College London, Masterstheses, 2005

[21] PENTIMENT: *Pentiment Homepage*. http://www.pentiment.de. – (letztes Zugriffsdatum: 21.02.2009)

[22] REAS, Casey ; FRY, Benjamin: Processing.org: a networked context for learning computer programming. In: *SIGGRAPH '05: ACM SIGGRAPH 2005 Web program*. New York, NY, USA : ACM, 2005, S. 14

[23] SATO, Keiichi ; VERPLANK, William: Panel: Teaching Tangible Interaction Design. In: *Proceedings of DIS'00: Designing Interactive Systems 2000*, 2000, S. 444–445

[24] SUKALE, Martin: *Computergestützte Kunstprojekte - Neuere technologische Entwicklungen*, HAW Hamburg, Studienarbeit, 2008

[25] SUKALE, Martin: *Konstruktion eines Netzwerkes eingebetteter Systeme für interaktives Design*, HAW Hamburg, Diplomarbeit, 2008

[26] WEISER, Mark: The Computer for the Twenty-First Century. In: *Scientific American 265*, 1991, S. 94–104

[27] WINOGRAD, Terry: From Computing Machinery to Interaction Design. In: DENNING, Peter (Hrsg.) ; METCALFE, Robert (Hrsg.): *Beyond Calculation: The Next Fifty Years of Computing*, Springer-Verlag, 1997, S. 149–162

[28] WRIGHT, Matthew ; FREED, Adrian ; MOMENI, Ali: OpenSound Control: State of the Art 2003. In: *Proceedings of the 2003 Conference on New Interfaces for Musical Expression (NIME-03)*, 2003, S. 153–159

[29] ZHANG, Yun ; CANDY, Linda: An in-depth case study of art-technology collaboration. In: *C&C '07: Proceedings of the 6th ACM SIGCHI conference on Creativity & cognition*. New York, NY, USA : ACM, 2007. – ISBN 978–1–59593–712–4, S. 53–62

Bildnachweise

Abbildung 3.1: http://www.phidgets.com/images/2000.jpg
(Zugriffsdatum: 21.02.2009)

Abbilgung 3.2: [9]

Abbilgung 3.3: [12] (Das Bild wurde vom Autor bearbeitet.)

Abbilgung 3.4: http://arduino.cc/en/uploads/Main/ArduinoDuemilanove.jpg
(Zugriffsdatum: 25.02.2009)

Abbilgung 3.5: Das Bild wurde vom Autor erstellt.