Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Seminararbeit

Sebastian Gregor

Vision einer grafisch unterstützten Entwicklungsumgebung für Wearable Computing

Fakultät Technik und Informatik
Studiendepartment Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Sebastian Gregor

Vision einer grafisch unterstützten Entwicklungsumgebung für Wearable Computing

Seminararbeit eingereicht im Rahmen des Masterstudiums
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 30. März 2009

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einleitung

Diese Seminararbeit steht im engen Zusammenhang mit den Arbeiten des Autors im Rahmen der Veranstaltungen „Anwendungen 2“ und „Projekt“ des Masterstudiums im Studiengang Informatik der Hochschule für Angewandte Wissenschaften Hamburg. Diese Veranstaltungen thematisieren Teilaspekte einer Idee für eine Masterarbeit. In der Ausarbeitung zu der Veranstaltung „Anwendungen 2“ [5] hat der Autor mehrere Phyiscal Computing und Physical Interaction Design Tools und Plattformen analysiert. Der Projektbericht [4] des Autors beschreibt die Erfahrungen, die während eines Kunstprojektes mit einer dieser Plattformen (Arduino) gemachten wurden. Diese Arduino-Plattform bildet die Grundlage für die in dieser Arbeit beschriebenen Vision einer grafisch unterstützten Entwicklungsumgebung.

1.1 Motivation

Elektronische Textilien (e-textiles) nehmen eine immer wichtigere Rolle im Wearable Computing ein. Tragbare, aus Stoff und leitbarem Garn hergestellte Computer können nützliche Funktionen anbieten und „verschwinden“ dabei fast vollständig in Kleidungsstücken. Sie verkörpern somit hervorragend Mark Weisers Idee vom Ubiquitous Computing [17].

E-textiles erlauben es Künstlern und Modedesignern, neue und interessante Stoffe, Kleidungsstücke oder künstlerische Artefakte zu kreieren. Dabei entstehen viele neue Ansätze und Sichtweisen bei der Verwendung und der Interaktion mit Technologie. Obwohl sich diese vielen neuen Möglichkeiten bieten, bestehen bei Kunstschaffenden und Designern große Hemmnisse bei der Verwendung von neuen Technologien. Diese basieren vor allem auf mangelndem Verständnis und fehlendem Wissen über die Kontrollmöglichkeiten der Technologie. Hat der Modedesigner oder Künstler keine ausreichenden Informatikkenntnisse, so ist er auf die Zusammenarbeit mit Informatikern oder Ingenieuren angewiesen. Mamykina u. a. teilen in [7] diese Zusammenarbeit in drei Kategorien auf: der Technologist als Assistent, eine Partnerschaft mit der Kontrolle durch den Künstler und eine gleichberechtigte Partnerschaft zwischen beiden. Die Rolle des Technologen in allen Formen dieser Zusammenarbeit sehen Greg Turner und Ernest Edmonds in [15] im Schaffen und Erweitern von Umgebungen, die es dem Künstler ermöglichen seine Ideen auszudrücken. Mit Hilfe dieser Umgebungen lassen sich Ideen des Künstlers in etwas transformieren, das mit einem Computer verarbeitet

werden kann. Durch diese Arbeit werden dieser Prozess und die verwendeten Technologien zu etwas für den Künstler greif- und erfassbarem.

Nach [16] folgt die Entwicklung eines Systems für interaktiver Kunst oder interaktivem Design häufig einem festen Muster. Zuerst erfolgt eine grobe Beschreibung durch den Künstler auf einer abstrakten Ebene. Hierbei präsentiert der Künstler seine Ideen in einer vagen Beschreibung mittels metaphorischen Ausdrücken. Diese Beschreibung wird durch den Programmierer in maschinennahe Konstrukte herunter gebrochen. Diese werden dann abstrahiert, angepasst und zu „high-level“-Strukturen zusammengefügt. Um einen genauen Überblick über die notwendigen Techniken und Technologien zu erhalten, stellt der Programmierer dem Künstler während dieses Prozesses viele Fragen und „Was-Wenn“-Szenarien. In [16] wird für diesen Prozess der Begriff des „Anpassens“ geprägt. Der Programmierer versucht auf diese Art und Weise den Computer für die Benutzung durch den Künstler anzupassen. Idealerweise führt das dazu, dass der Künstler den Programmierer nicht mehr benötigt.

1.2 Ziel und Gliederung

Ziel dieser Arbeit ist es, die Vision einer grafisch unterstützte Entwicklungsumgebung für Wearable Computing zu beschreiben. Diese soll den Designern einen intuitiveren Einstieg in die Verwendung von Wearable Computing Technologien ermöglichen. Dabei soll keine neue Umgebung geschaffen, sondern vielmehr die bereits vorhandene Wearable Computing Plattform LilyPad erweitert werden. Im nachfolgenden Kapitel wird diese Vision erläutert und Designkriterien, die der Entwicklungsumgebung zugrunde liegen, beschrieben. Im Kapitel 2.3 wird die verwendete LilyPad-Plattform und Technologien, die bei der Umsetzung der Vision hilfreich sein können, vorgestellt.

2 Vision

Abbildung 2.1: eingenähter LilyPad

Die Vision dieser Arbeit folgt der in Kapitel 1.1 beschriebenen Denkweise des „Anpassens“. Aus den Entwicklungen von Leah Buechley ist bereits das LilyPad [3] Toolkit (siehe 2.3.2) für den Einstieg in Wearable Computing und e-textiles entstanden. Programmiert wird dieses Toolkit über die Arduino Entwicklungsumgebung (siehe 2.3.1). Das Ziel ist es, die Arduino Entwicklungsumgebung für die LilyPad-Plattform so anzupassen, dass Designer beim Entwurf und Implementatierung von e-textiles nicht mehr auf die Hilfe eines Programmierers angewiesen sind. Zu diesem Zweck soll ausgehend von der Core Library der Arduino IDE eine grafisch unterstützten Entwicklungsumgebung für LilyPad entstehen.

Im folgenden Kapitel werden die der Entwicklung zugrundeliegenden Kriterien erläutert. Anschließend werden die Funktionen der beiden Teile der Entwicklungsumgebung, eine grafisch unterstützte Programmier- und eine grafisch unterstützte Simulationsumgebung, beschrieben. Nach der Beschreibung von für die Entwicklung relevanten Technologien werden zum Abschluss die bisher vom Autor identifizierten Risiken dargestellt.

2.1 Entwicklungskriterien

In [15] wird von Turner u. a. die unterstützende Arbeit von Informatikern in interaktiven Kunstprojekten untersucht. Dabei wurden Arbeiten der Informatiker identifiziert, die für das kreative Arbeiten von Künstlern hilfreich sind. Aus diesen Arbeiten lassen sich Anforderungen an Software Tools in der interaktiven Kunst Domäne ableiten, so dass Künstler bei ihrer kreativen Arbeit unterstützt werden. Hierzu werden in der Arbeit von Schneiderman u. a. in [11] zwölf Designprinzipien vorgestellt. Die resultierenden Anforderungen lassen sich auf die

Werkzeuge zum Designen von e-textiles übertragen. Sie werden von André Knörig in [6] zu folgenden Punkten zusammengefasst:

Erforschen und Experimentieren Designer wollen mit alle Ausdrucksfreiheiten, die ihnen die Technik zur Verfügung stellt, vertraut sein. Zu diesem Zweck ist es wichtig Experimentiermöglichkeiten zur Verfügung zu stellen. Turner u. a. beschreiben in [15], wie wichtig in diesem Zusammenhang undo- (engl. für rückgängigmachen) und redo-Funktionen (engl. für wiederherstellen), sowie keep, recover und move previous versions (engl. für übernehmen, wiederherstellen und auf ältere Version zurückgehen)-Funktionen beim Experimentieren sind. Um das Verhalten bei Veränderung von einzelnen Parameter besser vergleichen zu können, ist eine Möglichkeit der Gegenüberstellung unterschiedlicher Tests und Ergebnisse vorteilhaft. Wie Terry u. a. in [14] darstellen, kann die Entwicklung unterschiedlicher Experimente durch die gleichzeitige und aktive Entwicklung von Alternativen und Variationen unterstützt werden.

Low threshold, high ceiling, wide walls Mit dieser metaphorischen Beschreibung soll das große Spektrum an Einsatzmöglichkeiten der Software beschrieben werden. Low treshold (engl. für niedrige Schwelle) bezieht sich auf ein niedriges Einstiegshemmnis bei der Verwendung des Tools. Nach dem Start soll der Benutzer sofort in der Lage sein mit dem Tool zu arbeiten, ohne Einstellungen vornehmen zu müssen. High ceiling (engl. für hohe Decke) beschreibt die Möglichkeit, dass Experten anspruchsvolle Projekte umzusetzen, was jedoch in Konflikt zu den vorher beschriebenen niedrigen Einstiegshemmnis stehen kann. Wide walls (engl. für weite Wände) steht Metaphorisch für die Offenheit der Umgebung, die unterschiedlichsten Design Varianten zu erlauben und den Benutzer nicht in seiner Kreativität zu beschränken.

Informalität Das Tool soll einen informellen Einsatz ermöglichen. Das heißt, dass auch nicht vordefinierte Aktionen ausgeführt werden können. Knörig spricht in diesem Zusammenhang davon, lose Verknüpfungsrichtlinien oder formlose Annotation zu ermöglichen.

Community - Unterstützung und Zusammenarbeit Kreativität und kreatives Arbeiten ist ein sozialer Prozess, in dem der Austausch von Ideen wichtig ist. Mit einer Unterstützung der Community ist nicht nur die Möglichkeit zum Austausch von Sourcecode - Dateien, sondern vielmehr die Möglichkeit das Projekt gut zu präsentieren und zu veröffentlichen gemeint.

Detailkontrolle Ein erleichterter Einstieg, wie in „Low threshold, high ceiling, wide walls“ beschrieben, soll trotzdem nicht die Kontrolle über Details einschränken. Nicht nur für das Experimentieren, sondern auch bei der Feinjustierung von gewünschten Aktionen ist die Kontrolle über alle Parameter wichtig.

Personalisierung Benutzer mit unterschiedlichen Fertigkeiten bei der Verwendung eines Tools haben, was die Kontrolle und den Funktionsumfang angeht, unterschiedliche Bedürfnisse. Eine Anpassungsmöglichkeit des Tools soll dem Rechnung tragen.

Fokus Die Kernfunktionalitäten sollen bei der Entwicklung immer im Vordergrund stehen. Die Einsatzmöglichkeiten und die Akzeptanz des Tools hängen direkt von der Zuverlässigkeit der Kernfunktionen ab.

Ästhetik Ästhetik ist eines der Hauptaugenmerke von Designern und Künstlern bei ihrer Arbeit. Look-and-Feel sollten die Zielgruppe und das Einsatzgebiet wiederspiegeln. Als Beispiel für die Unterschiede werden von Knörig Processing [10] und Eclipse (siehe 2.3.3) angeführt. Das für Informatiker entwickelte Werkzeug Eclipse besitzt einegroßen Vielfalt von Funktionen, sowie Eigenschafts- und Ausgabefenstern. Processing hingegen spricht mit der Schlichtheit der Oberfläche und dem Fokus auf wenige Kernfunktion eher Designer an.

Kosten Nicht nur die Anschaffungskosten sind ein wichtiges Kriterium, welches vor allem bei Designprojekten mit kleinem Budget eine große Rolle spielt. Auch Kosten die durch Zeitersparnisse verringert werden oder die Möglichkeit etwas „überhaupt“ tun zu können sind wichtige Kriterien bei der Entwicklung.

Flexibilität Die Umgebung sollte die Umsetzung einer Vielzahl von unterschiedlichen Applikationen erlauben. Des weiteren ist dieser Punkt für die oben angesprochene Personalisierung wichtig. Aus diesem Grund rät Knörig zum Einsatz von offenen Schnittstellen und Standards, was entscheidend für die Möglichkeit des Austausches von oder zwischen Projekten ist.

Peripherie In seiner Analyse (vgl. [6]) beschreibt Knörig, das der Designprozess eine Menge periphere Aktivitäten umfasst. Zu diesen Aktivitäten gehören z. B. das Recherchieren, das erstellen von Sketches (engl. für Skizzen) und das Dokumentieren. Um diese Aktivitäten zu Unterstützen, sollte es Möglich sein, unterschiedliche Typen von Dokumenten, Dateien und Notizen an ein Projekt anzufügen.

2.2 Vision der Entwicklungsumgebung

Die Vision einer grafisch unterstützten Entwicklungsumgebung soll den oben beschriebenen Kriterien gerecht werden. Die sich nach Ansicht des Autors hieraus ergebende Funktionen und Bestandteile der Umgebung, die sich aus einer grafisch unterstützten Programmier- und einer grafisch unterstützten Simulationsumgebung zusammensetzt, werden in den beiden nachfolgenden Abschnitten beschrieben.

2.2.1 grafisch unterstützte Programmierumgebung

Um einen einfachen Einstieg zu ermöglichen, soll es mit der Programmierumgebung möglich sein, ein Programm für die LilyPad-Plattform mittels grafischer Programmierung zu erstellen. Über eine Auswahlfenster sollen die Elemente der grafischen Programmierung, Grafiken von LilyPad, LilyPad-Sensoren und -Aktoren, durch einfaches Drag-and-Drop auf einer Arbeitsfläche platziert werden können. Es ist vorgesehen die Entwicklung ausgehend von eine festen Anzahl von Sensoren und Aktoren zu beginnen. Dem Benutzer soll es darüber hinaus möglich sein, weiter Elemente einzubinden. Hierzu muss der Benutzer die Aktionsmöglichkeiten und Funktionen, sowie ein grafische Symbol für jedes neue Element definieren können.

Die Sensoren und Aktoren können mittels frei platzierbaren Verbindungen, die leitbares Garn symbolisieren, mit dem LilyPad-Anschlüssen symbolisch verbunden werden. Als Hintergrund kann im Arbeitsbereich eine Grafik, z. B. eine Skizze des Zuschnittes eines zu erstellenden Kleidungsstückes, eingefügt werden. Diese soll eine bessere Orientierung beim Experimentieren mit der Anordnung der Elemente erleichtern. Über ein Eigenschaftsfenster soll das Verhalten der einzelnen Elemente editiert werden können. Über die Verbindungslinien und das Eigenschaftsfenster des LilyPad kann das Verhalten des Gesamtsystems gesteuert werden.

Eine frei definierbare Zeitachse soll dem Benutzer chronologisch gesteuerte Aktionen ermöglichen. Dieser Zeitstrahl kann ähnlich einer Bildlaufleiste, wie aus Browsern oder Dokumentenbearbeitungsprogrammen bekannt, bedient werden.

Der vom System erstellte Sourcecode soll jederzeit über eine Ansicht im System manuell editierbar sein. Diese Funktion soll es erfahrenen Benutzern möglich sein, den Sourcecode mit eigenen Programmzeilen zu erweitern. Unerfahrenen Nutzern wird gleichzeitig die Möglichkeit gegeben, durch ein direktes Vergleichen von Quellcode, welcher für unterschiedliche Elemente mit unterschiedlichen Eigenschaften erzeugt wurde, die Syntax und Semantik der Programmiersprache für die LilyPads erlernen.

Durch die Integration der Arduino Core Library und dem AVR-GCC (vergleiche 2.3.1) ist es möglich den erzeugten Sourcecode per klick auf einen „Upload“-Knopf zu kompilieren und auf einen angeschlossenen LilyPad zu laden. Um die Veröffentlichung, Dokumentation und den Austausch von Projekten mit anderen Designern zu erleichtern, sollen unterschiedliche Dokumente an ein Projekt angefügt werden können. Innerhalb der einzelnen Arbeitsflächen sollen Notizfelder eine bessere Dokumentation ermöglichen. Der Export aller erstellten Arbeitsflächen, Dokumente sowie des Sourcecodes wird durch einen „Veröffentlichen“-Button realisiert.

2.2.2 grafisch unterstützte Simulationsumgebung

Beim Design von e-textiles werden elektronische Komponenten verbaut und gleichzeitig Software für die Steuerung dieser implementiert. Bei fehlerhaften Verhalten der Konstruktion ist es für unerfahrene Nutzer nur schwer ersichtlich, wo ein Fehler zu suchen ist. Erschwerend kommt hinzu, dass die LilyPads über keine Debug-Schnittstelle verfügen und die Auslöser von Laufzeitfehlern nur schwer zu ermitteln sind. Des weiteren ist das Vernähen von elektronischen Komponenten und das Heraustrennen dieser bei Fehlern ein kosten- und zeitintensiver Prozess. Aus diesen Gründen ist es sinnvoll eine Simulationsmöglichkeit innerhalb der Entwicklungsumgebung anzubieten.

Ziel der Umgebung ist die Simulation der vorher erstellten grafischen Programmnotation anzubieten. Dabei soll es möglich sein, den Zustand des Gesamtsystems sowohl über das manuelle Triggern von Sensoren, als auch über ein vorher chronologisch definierten Ablauf von Sensoreingaben zu verändern. Beim manuellen Triggern der Eingaben soll eine quantitative Eingabe von Sensorwerten mittels Zahlen, als auch durch einen grafische Schieberegler möglich sein. Bei einer vordefinierten zeitlichen Abfolge von Sensorwerten soll die Simulation über den gesamten Zeitraum der Sensoreingaben verfolgt werden können. Um Beispielsweise Eingabewerde zu variieren, soll zu jedem Zeitpunkt eine Unterbrechung der Simulation möglich sein.

2.3 Technologien

In diesem Abschnitt wird die, der Entwicklungsplattform zugrundeliegende, Zielplattform LilyPad beschrieben. Zuerst wird eine kurze Übersicht über die Physical Computing Plattform Arduino, aus der LilyPad hervorgegangen ist, gegeben. Im zweiten Abschnitt des Kapitels wird Eclipse beschrieben, das sich als Grundlage für die Entwicklung einer grafischen unterstützten Entwicklungsumgebung für LilyPad anbietet.

2.3.1 Arduino

Arduino ist eine Mikrokontroller gesteuerte Physical Computing Plattform [8] bestehen aus Open-Source Hardware und einer Open-Source Entwicklungsumgebung. Die Hardwareplattform verfügt über 8-Bit RISC Mikrokontroller und ist in unterschiedlichen Formfaktoren verfügbar. Sie lässt sich über USB, Funk oder RS232 an einen Computer anschließen. Die Entwicklungsumgebung (siehe Abbildung 2.2) setzt sich aus einem Editor und der Core-Bibliothek zusammen. Programmiert wird der Arduino in einer der Programmiersprache C

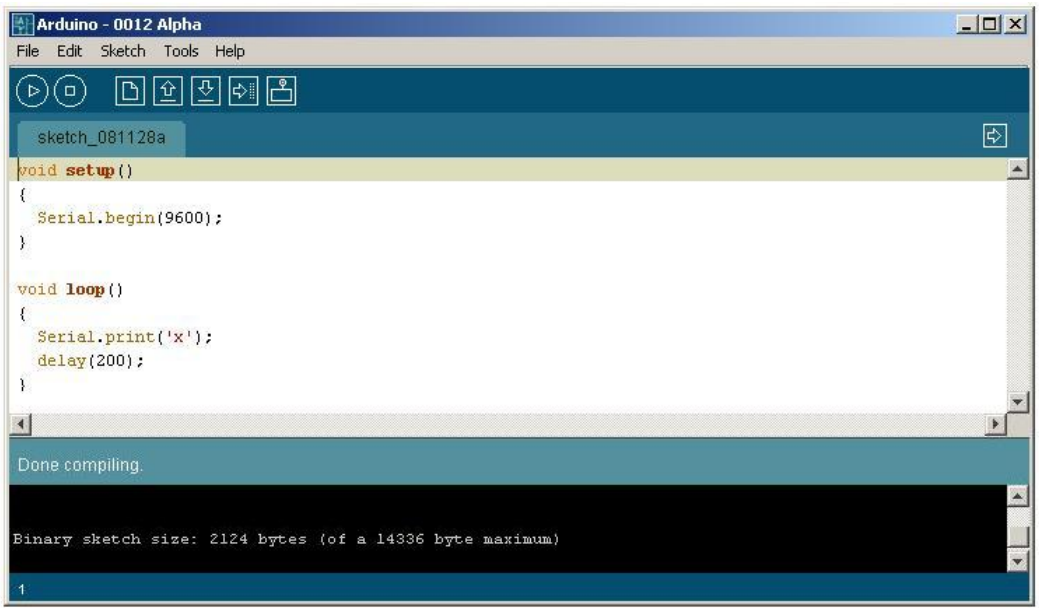

Abbildung 2.2: Arduino IDE

ähnlichen Sprache, die auf Wiring [1] basiert. Nachdem der Arduino programmiert ist, kann er mit einer externen Stromquelle autonom, ohne Verbindung zum Computer arbeiten.

Die Core Library besteht aus C/C++ Klassen und Funktionen, die die hardwarenahen und Mikrokontroller-spezifische Programmierelemente kapseln. Mithilfe dieser Bibliothek und dem integrierten AVR-GCC[1]. Dadurch können Benutzer auch ohne Wissen über die Programmierung von Mikrokontrollern Programme für den Arduino schreiben. Die erstellten Programme können ohne eine spezielle Programmierhardware auf den Arduino geladen werden. Möglich macht dies ein Bootloader[2], der auf jedem Arduino Mikrokontroller aufgespielt ist.

Die Arduino IDE ist nach dem Entpacken der Installationsdatei sofort funktionsfähig, ohne das weitere Komponenten installiert werden müssen. Trotz der beschreibenen Vorteile gegenüber anderne Mikrokontroller-Plattformen muss der Benutzer zumindest über Grundkenntnisse in der Programmierung verfügen. Ein weiterer negativer Punkt und ein großes Hemmnis für Designer beim Einstieg in dieses Thema ist die Notwendigkeit, die C ähnliche Programmiersprache der Arduino Plattform erlernen zu müssen.

[1] AVR-GCC: C-Compiler für AVR-Mikrokontroller

[2] Bootloader: spezielle Software zum Laden eines Programmes in den Speicher eines Mikrokontrollers während der Initialisierungsphase

2.3.2 LilyPad

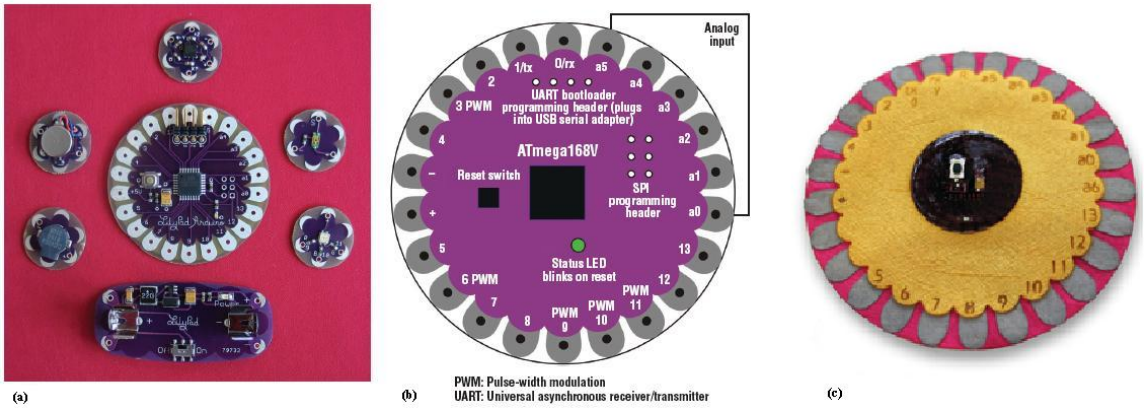

Abbildung 2.3: Arduino LilyPad (a)Starterkit (b)schematische Darstellung (c)Textilbasierter LilyPad

LilyPad [3] ist eine Arduino (vgl. Kapitel 2.3.1) kompatibles Mikrokontroller gesteuerte Platine für das Design von e-textiles [2]. Entwickelt wurde LilyPad 2007 von Leah Buechley an der University of Colorado. In der ersten Ausführungen handelt sich um eine auf leitbaren Textilien basierende Plattform (siehe Abbildung 2.3(c)). Diese konnte mit leitfähigem Garn zusammen mit Sensoren und Aktoren in die Kleidung eingenäht werden. Aufgrund des komplizierten und zeitaufwendigen Herstellungsprozesses wurde eine PCB-Version (siehe Abbildung 2.3) entwickelt. Bei dieser wurden die Ein- und Ausgabekontakte des Mikrokontrollers sternförmig an den Rand der Platine geführt und zum vernähen mit ausreichen großen Löchern versehen. Über einen USB-Adapter kann man einen LilyPad an einen Computer anschließen. Über diesen Adapter ist es so möglich den LilyPad mit der Arduino IDE (siehe Abbildung 2.2) zu programmieren.

LilyPad soll als Zielplattform für die Entwicklung verwendet werden. Dies bietet den Vorteil, dass mit Arduino Core Library und den in der Arduino integrierte AVR-GCC bereits eine funktionierende Umgebung zum kompilieren des Quellcodes vorhanden ist. Dadurch kann ein in der Programmierumgebung mit Hilfe des grafischen Editors oder manuell erstellter Sourcecode zu einem für den LilyPad ausführbaren Programmcode kompiliert werden.

2.3.3 Eclipse

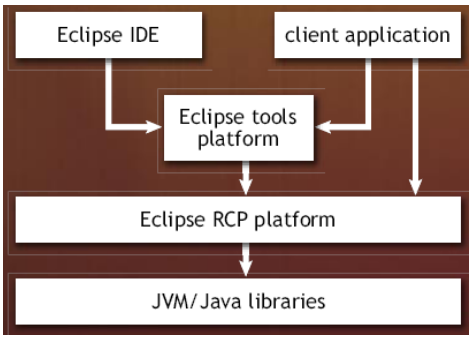

Abbildung 2.4: Eclipse RCP

Eclipse ist eine offene Entwicklungsplattform und ein Applikations - Framework zur Entwicklung von Software. Rubel beschreibt in [12] den Aufbau von Eclipse und deren Erweiterbarkeit auf Basis von Plug-ins. Diese werden gegen die portable API von Eclipse programmiert und laufen so auf allen von Eclipse unterstützten Betriebssystemen unverändert. Aufgrund dieser Flexibilität und Erweiterbarkeit, sowie der in 2.3.3 beschriebenen Rich Client Platform (RCP) Framework und dem Graphical Modeling Framework (GMF) bietet sich Eclipse als Grundlage für die in Kapitel 2 beschriebene Entwicklungsumgebung an.

RCP

Aufgrund des Plug-in basierten Aufbaus von Eclipse ist es möglich Anwendungen auf Basis des Eclipse Frameworks zu schreiben. Diese Technik nennt sich Rich Client Platform und beinhaltet eine minimale Anzahl von Plug-ins. Abbildung 2.4 gibt einen Überblick, wie eine Applikation auf Basis von RCP erstellt werden kann. Die mit RCP erstellen Anwendungen laufen unabhängig von der Eclipse IDE. Dies hat den Vorteil, dass eine eigene, gegenüber der Eclipse Benutzeroberfläche, leichtgewichtigere Grafische Benutzeroberfläche verwendet werden kann. Gleichzeitig ist es jedoch möglich die Flexibilität und Portierbarkeit von Eclipse für die zu entwickelnde Entwicklungsumgebung zu nutzen.

GMF

Das Eclipse Graphical Modeling Framework beinhaltet eine Laufzeitumgebung und Komponenten um einen grafischen Editor zu erstellen. Es basiert auf dem Eclipse Modeling Framework (EMF) und dem Graphical Editing Framework (GEF):

EMF Das EMF ist ein Framework zum Entwickeln von Applikationen basierend auf einem einfachen Klassenmodell. Dazu bietet EMF einen Editor zum Erstellen und Manipulieren eines strukturierten Modells an. Dieses Modell wird in XML[3] abgespeichert. Aus diesen XML Beschreibung kann EMF Java-Klassen generieren mit deren Hilfe eine Instanz des Modells erzeugt werden kann.

[3] XML: Extensible Markup Language - Beschreibung zur hierarchischen Darstellung von Daten in einem Textdokument

GEF Das Eclipse Grafical Editing Framework erlaubt es, aus einem bestehenden Modell heraus, einen grafischen Editor zu erzeugen. Ein mit dem GEF erstellter Editor wird nach dem MVC[4] - Paradigma erstellt und ermöglicht Domänen-spezifische Grafiken zu verwenden.

Elipse GMF soll verwendet werden um einen grafischen Editor für die Entwicklungsumgebung zu erzeugen. Mit diesem sollen die im Kapitel 2.2 beschriebenen grafischen Programmierelemente dargestellt werden. Diese können dann mit Hilfe vom in GMF integrierten EMF und eines vorher vom Autor erstellten Modells zu Arduino Sourcecode interpretiert werden.

2.4 Risiken

Ein Risiko bei der Entwicklung der Entwicklungsumgebung liegt in der Ausgewogenheit zwischen Flexibilität und Simplizität. Eine Unausgewogenheit zwischen beiden kann zu einer mangelnden Akzeptanz bei der Zielgruppe führen. Auch durch die Verwendung von visuellen Repräsentationen und grafischer Programmierung kann es zu Schwierigkeiten kommen. Carsten Schmidt zeigt in [13], dass visuelle Repräsentationen die Programme ab einem gewissen Umfang unübersichtlich werden lassen können. Die ebenfalls angestrebte Möglichkeit der manuellen Anreicherung des generierten Sourcecodes durch den Benutzer kann die Simulationsmöglichkeiten enorm erschweren. Hier müssen weitere Recherchen zeigen, ob und inwieweit sich diese Schwierigkeiten lösen lassen.
Offen ist auch, ob und in welchem Umfang eine grafische Simulation bestimmter Aktoren umsetzbar ist. Aufgrund der Zweidimensionalität des Bildschirms lassen sich z. B. Bewegungen nur schlecht darstellen und simulieren. Auch RGB LEDs lassen sich nur schwer farbecht auf dem Bildschirm darstellen.

[4] MVC: Model-View-Control - Architekturparadigma zur Strukturierung von Programmen

3 Zusammenfassung und Ausblick

Diese Arbeit beschreibt die Vision einer grafisch unterstützten Entwicklungsumgebung für Wearable Computing Projekte. Aufgrund bestehender Entwicklungen und der großen Verbreitung wird die auf Arduino basierende LilyPad als Hardwareplattform verwendet. Dies bietet den Vorteil, dass mit Arduino Core Library und den in der Arduino integrierte AVR-GCC bereits eine funktionierende Umgebung zum kompilieren des Quellcodes vorhanden ist. Dadurch kann ein in der Programmierumgebung mit Hilfe des grafischen Editors oder manuelle erstelltem Sourcecode zu einem für den LilyPad ausführbaren Programmcode kompiliert werden.

Die Entwicklungsumgebung soll sich aus einer grafisch unterstützten Programmier- und Simulationsumgebung zusammensetzen. Mit Hilfe der grafischen Programmierumgebung soll Designern der Einstieg in die Programmierung des LilyPad erleichtert und mit Hilfe der Simulationsumgebung das Experimentieren mit der Technik unterstützt werden.

Umgesetzt werden soll die Applikation mit Hilfe von Eclipse RCP und GMF. Die Verwendung von Eclipse RCP hat den Vorteil, dass die erstellte Applikation auf allen von Eclipse unterstützten Betriebsystemen lauffähig ist, ohne das Änderungen am Programm vorgenommen werden müssen. Dies erleichtert die geplante Implementierung für die drei Betriebssystemen Windows, Linux und MacOS X. Gerade die Umsetzung auf MacOS X ist für die Akzeptanz wichtig, da es häufig von Designern verwendet wird.

Ein erster Schritt der Entwicklung wird das Bereitstellen von verschiedenen Prototypen sein. Diese Prototypen ermöglichen Usability-Tests in einem frühen Stadium der Entwicklung. Dadurch soll die Zielgruppe möglichst frühzeitig in die Entwicklung mit eingebunden und die Akzeptanz und Einsetzbarkeit der Entwicklungsumgebung erhöht werden.

Literaturverzeichnis

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

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

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

[4] GREGOR, Sebastian: *Masterprojektbericht - Emotional Tent*, Hochschule für Angewandte Wissenschaften Hamburg, Projektbericht, 2009

[5] GREGOR, Sebastian: *Physical Interaction Design*, Hochschule für Angewandte Wissenschaften Hamburg, Seminararbeit, 2009

[6] KNÖRIG, André: *Design Tools Design*, University of Applied Sciences Potsdam, Diplomarbeit, 2008

[7] MAMYKINA, Lena ; CANDY, Linda ; EDMONDS, Ernest: Collaborative creativity. In: *Commun. ACM* 45 (2002), Nr. 10, S. 96–99. http://dx.doi.org/http://doi.acm.org/10.1145/570907.570940. – DOI http://doi.acm.org/10.1145/570907.570940. – ISSN 0001–0782

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

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

[10] 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

[11] RESNICK, Mitch ; MYERS, Brad ; NAKAKOJI, Kumiyo ; SHNEIDERMAN, Ben ; PAUSCH, Randy ; SELKER, Ted ; EISENBERG, Mike: Design Principles for Tools to Support Creative Thinking. In: *NSF Workshop Report on Creativity Support Tools*, 2005, S. 25–36

[12] RUBEL, Dan: The Heart of Eclipse. In: *Queue* 4 (2006), Nr. 8, S. 36–44. http://dx.doi.org/http://doi.acm.org/10.1145/1165754.1165767. – DOI http://doi.acm.org/10.1145/1165754.1165767. – ISSN 1542–7730

[13] SCHMIDT, Carsten: *Generierung von Struktureditoren für anspruchsvolle visuelle Sprachen*, Universität Paderborn, Diss., 2006

[14] TERRY, Michael ; MYNATT, Elizabeth D. ; NAKAKOJI, Kumiyo ; YAMAMOTO, Yasuhiro: Variation in element and action: supporting simultaneous development of alternative solutions. In: *CHI '04: Proceedings of the SIGCHI conference on Human factors in computing systems*. New York, NY, USA : ACM, 2004. – ISBN 1–58113–702–8, S. 711–718

[15] TURNER, Greg ; EDMONDS, Ernest: Towards a Supportive Technological Environment for Digital Art. In: *Proceedings of OzCHI2003*, 2003, S. 44–51

[16] TURNER, Greg ; EDMONDS, Ernest ; WEAKLEY, Alastair: Seeing Eye-to-Eye: Supportive Transdisciplinary Environments for Interactive Art. In: *IV '05: Proceedings of the Ninth International Conference on Information Visualisation*. Washington, DC, USA : IEEE Computer Society, 2005. – ISBN 0–7695–2397–8, S. 912–919

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

Bildnachweise

Abbildung 2.1:
http://farm3.static.flickr.com/2028/2274045715_cb6838a14f.jpg
(Zugriffsdatum: 22.03.2009)

Abbildung 2.2:
Wurde vom Autor erstellt.

Abbildung 2.3:
[3] Seite 13
und
http://web.media.mit.edu/~leah/grad_work/index.html
(Zugriffsdatum: 22.03.2009)

Abbildung 2.4:
[12] Seite 39