Entwicklung und Evaluation einer gendergerechten Lerneinheit zur Bluetooth-Kommunikation unter Einsatz von Android Smartphones, Lego Mindstorms NXT und Arduino Mikrocontrollern

Schriftliche Hausarbeit im Rahmen der Ersten Staatsprüfung, dem Landesprüfungsamt für Erste Staatsprüfungen für Lehrämter an Schulen vorgelegt von:

Jan Tobias Holz

31. März 2011, Aachen

Gutachter: Prof. Dr. Ulrik Schroeder
Betreuer: Dipl.-Gyml. Thiemo Leonhardt
Lehr- und Forschungsgebiet Informatik 9
RWTH Aachen

Inhaltsverzeichnis

1. Motivation **3**

2. Theoretischer Hintergrund: Gendergerechtes Motivieren **5**

2.1. Ausgangslage 5

2.2. Gendereffekte der Medien 7

2.3. Motivation für verstärkt technische Themen 12

2.4. Zusammenfassung der Thesen 13

3. Entwicklung des Moduls **15**

3.1. Aufbau des Moduls 15

3.2. Entwicklung der Software 18

3.2.1. Entwicklung der Android App 18

3.2.2. Entwicklung des Arduino Sketches 24

3.3. Entwicklung der Selbstlerneinheit 26

3.3.1. Strukturierung des Inhalts 26

3.3.2. Didaktische Überlegungen 27

4. Evaluation **33**

4.1. Methodik 33

4.1.1. Problemzentriertes Interview 33

4.1.2. Interviewleitfaden 35

4.2. Durchführung 36
4.3. Auswertung 38
4.3.1. Gender Gap 38
4.3.2. Erste These 40
4.3.3. Zweite These 42
4.3.4. Feedback zum Modul 43

5. Fazit **45**

6. Ausblick **46**

7. Quellenverzeichnis **48**

A. Anhang **51**
A.1. Lerneinheiten 52
A.1.1. Vorbereitungen zum Einsatz 52
A.1.2. Materialien 53
A.2. Quelltexte der Android Apps 54
A.2.1. NXT Remote 54
A.2.2. Arduino Remote 72
A.3. Der fahrbare Arduino Mikrocontroller 88
A.3.1. Materialliste 88
A.3.2. Bau 89
A.3.3. Quelltext 90
A.4. Interviewleitfaden 93
A.4.1. Vor der Lerneinheit 93
A.4.2. Nach der Lerneinheit 93

B. Erklärung **95**

1. Motivation

„Everything is better with Bluetooth!“

Dr. Dr. Sheldon Cooper
The Big Bang Theory

Das Schülerlabor Informatik der RWTH Aachen bietet Schülerinnen und Schülern (Abk.: SuS) die Möglichkeit anhand spezieller Module unterschiedliche Zugänge zur Informatik selbst zu erkunden. Die SuS haben die Möglichkeit in halb- bis mehrtägigen Projekten die Inhalte handlungsorientiert selbst zu erarbeiten (SLI, 2011). Im Rahmen dieser Staatsexamensarbeit wurde ein Modul zur Bluetooth-Kommunikation rund um Android Smartphones, Lego Mindstorms NXT und Arduino Mikrocontroller entwickelt.

Die Bluetooth-Technologie ist aus mobilen Geräten des Alltags inzwischen nicht mehr wegzudenken. Egal ob Laptop, Headset oder Smartphone, nirgends darf es fehlen. Doch wie läuft so eine Bluetooth-Kommunikation zwischen den Geräten ab? Die Antwort auf diese Frage wird von den SuS in der Lernumgebung dieses Moduls selbst erarbeitet. Das neu erworbene Wissen wird unmittelbar in die Praxis umgesetzt, indem mit Java eine eigene App für aktuelle Android Smartphones entwickelt wird: Eine bluetoothbasierte Fernsteuerung,

die den Lagesensor des Smartphones ausliest. Mit der Bluetooth-Kommunikation, als einer Form der Datenkommunikation, wird hier ein zentraler Aspekt der Informatik, praxisnah aufgegriffen.

Das Modul besteht aus einer halbtägigen Selbstlerneinheit, die von den SuS in Zweiergruppen bearbeitet wird. Es wurden zwei verschiedene Versionen der Lerneinheit erstellt: In der einen Variante werden einsatzbereite Lego Mindstorms NXT ferngesteuert, in der anderen Variante fahrbare Arduino Mikrocontroller, welche zuvor noch selbst programmiert werden müssen.

Die Praxisnähe, der Einsatz aktueller Technologien, sowie der Alltagsbezug sollen auf die SuS genderübergreifend motivierend wirken. Dabei soll der Einsatz von Smartphones insbesondere Mädchen ansprechen. Die Auswirkung von Smartphones auf die Motivation der SuS wurde im Rahmen dieser Staatsexamensarbeit qualitativ evaluiert. Vor diesem Hintergrund wurde zudem untersucht, ob die Motivation, die möglicher Weise durch Smartphones hervorgerufen wird, auch auf technischere Themen, wie den Einsatz von Arduino Mikrocontroller übertragen werden kann. Die Evaluation des entwickelten Moduls fand am Ernst-Moritz-Arndt-Gymnasium in Bonn statt.

2. Theoretischer Hintergrund: Gendergerechtes Motivieren

2.1. Ausgangslage

Informatikunterricht hat, so wie auch die Informatik selbst, mit einer stark ausgeprägten Gender Gap zu kämpfen. Konkrete Zahlen lassen sich dazu nur schwer und vereinzelt finden, wie auch bereits Humbert und Panske angemerkt haben. Doch die Lage wird schnell deutlich, wenn die Geschlechterverteilung und die geringe Anzahl Teilnehmerinnen in Wahlkursen, Wettbewerben und später dem Informatikstudium betrachtet wird (Abbildung 2.1) (Humbert/Panske, 2010, S. 25). Die Gründe hierfür sind mannigfaltig und werden an dieser Stelle nicht im Detail erörtert. Stattdessen werden einige Thesen aufgestellt und betrachtet, die bei der Gestaltung von gendergerechten Lerneinheiten für die Informatik helfen sollen.

Neben den genderspezifischen Aspekten wird stets auch berücksichtigt, dass das Betrachten der Didaktik ebenso wichtig ist, wie das Betrachten von Genderfragen. Beides wirkt sich für SuS maßgeblich auf das Erleben der Lerneinheit aus, wobei der Einfluss der didaktischen

Abbildung 2.1.: Studienanfänger in der Informatik (Destatis, 2011)

Aspekte unter Umständen sogar überwiegen kann (Petersen et al., 2007, S. 150-153).

Zudem wurde bei der gesamten Gestaltung berücksichtigt, dass der Einsatz aktueller Medien und Technologien aus der Alltagswelt der SuS genderübergreifend besonders motivierend wirken kann und daher dem Einsatz alter oder altbekannter Medien und Technologien vielfach vorzuziehen ist (Deinmann, 2002, S. 65).

Mit diesen Erkenntnissen als Grundlage wurde nun recherchiert und überlegt, welche Maßnahmen unternommen werden können um der Gender Gap entgegen zu wirken und SuS gleichermaßen zu erreichen.

2.2. Gendereffekte der Medien

In ihrem Artikel „Informatik und Gender – nehmt die Forschungsergebnisse ernst!“ fordern Humbert und Panske eine Umorientierung des Informatikunterrichts um SuS genderübergreifend anzusprechen. Die Nutzung neuer Medien beschränkt sich im Rahmen des Informatikunterrichts häufig nach wie vor auf den Einsatz von Computern. Dies wird der Alltagswelt der Schüler und vor allem der Schülerinnen jedoch nicht gerecht. Die Autoren des Artikels berufen sich dazu auf die JIM-Studie 2008. Diese Studie zeigt auch, dass 95% der Jugendlichen ein Handy besitzen, wobei Mädchen mit 96% sogar einen leichten Vorsprung gegenüber Jungen mit 94% haben. Handys sind damit genderübergreifend das verbreitetste Gerät bei den 12- bis 19-jährigen (Humbert/Panske, 2010, S. 27)(JIM 2008, S. 10). Aus den Daten der Mediennutzung und Medienbindung ergibt sich zudem, dass Handys insgesamt für die Jugendlichen einen hohen Stellenwert haben. Dieser ist für Mädchen jedoch noch ein Stück höher, als für Jungen. Von den Autoren wird deshalb vorgeschlagen Handys schwerpunktmäßig in den Informatikunterricht einzubinden.

> „Ein von den Interessen und den technischen Präferenzen der Mädchen und jungen Frauen ausgehender Informatikunterricht arbeitet in der technischen Umsetzung (inklusiv Programmierung) mit Mobiltelefonen. PDAs, Laptops oder Desktop-Computer kommen in diesem Unterricht nicht oder nur selten zum Einsatz.“ (Humbert/Panske, 2010, S. 29)

Seit Humberts und Panskes Veröffentlichung 2009 kam es zu einem Smartphone-Boom, der klassische PDAs nahezu vollständig und herkömmliche Handys teilweise verdrängt hat (Statista, 2011a)(Statista, 2011b). Die Grenze zwischen dem Einsatz von Handys und PDAs kann heute also nicht mehr so klar gezogen werden, wie es im vorigen Zitat verlangt wird. Auch werden in der aktuellen JIM-Studie 2010 erstmals Smartphones separat als Medium aufgeführt. Dass bereits schon rund 14% der Jugendlichen zwischen 12 und 19 Jahren über ein Smartphone verfügen zeugt von der raschen Verbreitung. Insbesondere ältere SuS verfügen bereits über Smartphones (JIM 2010, S. 54). Die Verbreitung von Handys hat sich gegenüber 2008 nochmals leicht gesteigert und beträgt nun 97% (siehe Abbildung 2.2) (JIM 2010, S. 10).

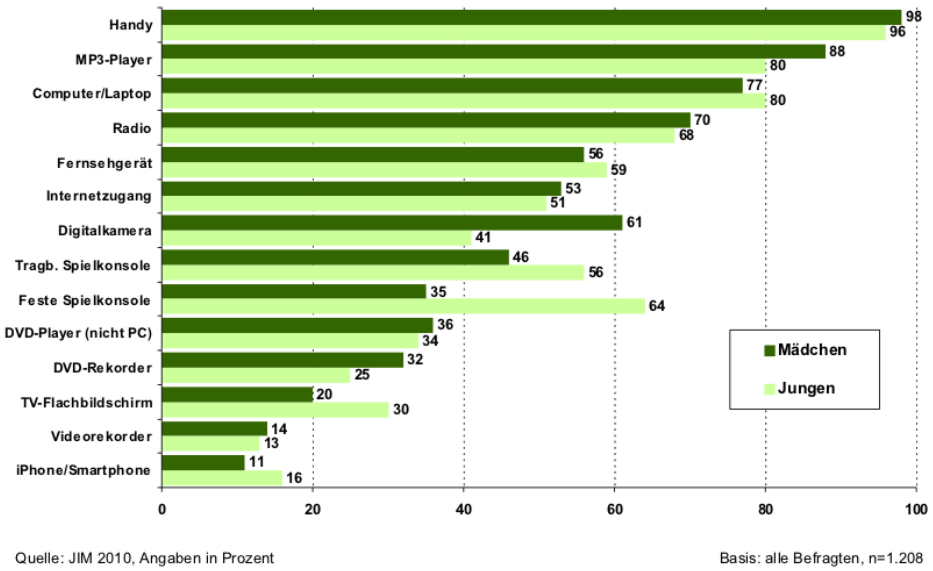

Abbildung 2.2.: Gerätebesitz Jugendlicher 2010 (JIM 2010, S. 10)

Humberts und Panskes Argumentation, dass der Einsatz von Handys im Informatikunterricht aufgrund der gleichmäßigen Verbreitung

und Nutzung bei beiden Geschlechtern sinnvoll ist, kann anhand der JIM-Studie 2010 auch auf Smartphones übertragen werden, da für die SuS bei der Nutzung die klassischen Handy-Funktionen wie SMS verschicken und Telefonieren deutlich überwiegen. Zudem scheint die Grenze zwischen Smartphone und Handy vielfach nicht bewusst wahrgenommen zu werden, sodass SuS ein Smartphone mitunter für ein gewöhnliches Handy halten. Dies spiegelt sich auch darin wieder, dass mit durchschnittlich 9% nur ein vergleichsweise geringer Anteil des SuS Apps nutzt. Hierbei lässt sich in der Nutzung allerdings ein deutlicherer Unterschied zwischen Jungen (12%) und Mädchen (5%) festzustellen (JIM 2010, S. 57-58).

Gendergerechter Unterricht soll nun die Interessen von Mädchen und Jungen gleichermaßen aufgreifen, ohne eine der beiden Gruppen zu bevorzugen. In den Thesen von Humbert und Panske finden sich zwei Punkte die im Folgenden mit den bisherigen Erkenntnissen zu einer These zur Gestaltung von gendergerechtem Informatikunterricht kombiniert werden sollen. Humbert und Panske schreiben:

> „Informatikunterricht auf Computer und ihre (Be-)Nutzung zu reduzieren, kommt den Interessen der Jungen entgegen. [...] Um die tatsächlichen Lernvoraussetzungen von Mädchen und jungen Frauen zu berücksichtigen, findet Informatikunterricht in normalen Klassen- oder Kursräumen ohne Computer statt.“ (Humbert/Panske, 2010, S. 28)

Um einen zukunftsweisenden, gendergerechten Unterricht zu gestalten setzen Humbert und Panske nun vollständig auf den Einsatz von Handys:

> „Um diese Zukunft vorwegzunehmen, wird zur Zeit ein ganz normaler Informatikkurs in der gymnasialen Oberstufe mit Mobiltelefonen durchgeführt – einzig diese Geräte werden zur Programmierung genutzt – keine weitere Technik steht den Schülerinnen und Schülern im Unterricht zur Verfügung.“ (Humbert/Panske, 2010, S. 28)

Dieser Ansatz ist jedoch kritisch zu betrachten, da er zwar an der Alltagswelt der SuS anknüpft, die Anwendung des informatischen Wissens jedoch recht realitätsfern geschieht. Die Programmierung von Handys an Handys ist außerhalb dieser speziellen Lernumgebung nicht zu finden. Dadurch läuft dieser Ansatz Gefahr träges Wissen zu erzeugen. Um Wissen auch außerhalb des Unterrichts nutzbar zu machen, sollte dieser an realistische Szenarien anknüpfen.

In dieser Arbeit wird als gemeinsamer Nenner der beiden zitierten Thesen daher eine Lernumgebung mit Notebooks und Smartphones gewählt. Der Einsatz ist somit nicht an Fachräume gebunden und kann im normalen Klassenraum stattfinden. Die Notebooks werden genutzt um Apps für Smartphones zu programmieren, die daraufhin direkt von den SuS eingesetzt werden können. Smartphones sind an dieser Stelle Handys vorzuziehen, da sie wesentlich vielfältiger eingesetzt werden können und die aktuellere und daher spannendere Technologie sind. Der relativ geringen Nutzung von Apps durch Mädchen wird versucht entgegenzuwirken, indem die von Mädchen bevorzugten Funktionen des Smartphones aufgegriffen werden. Beispiele hierfür sind das Erstellen von Fotos und Filmen oder das Verschicken von Daten per Bluetooth (siehe Abbildung 2.3).

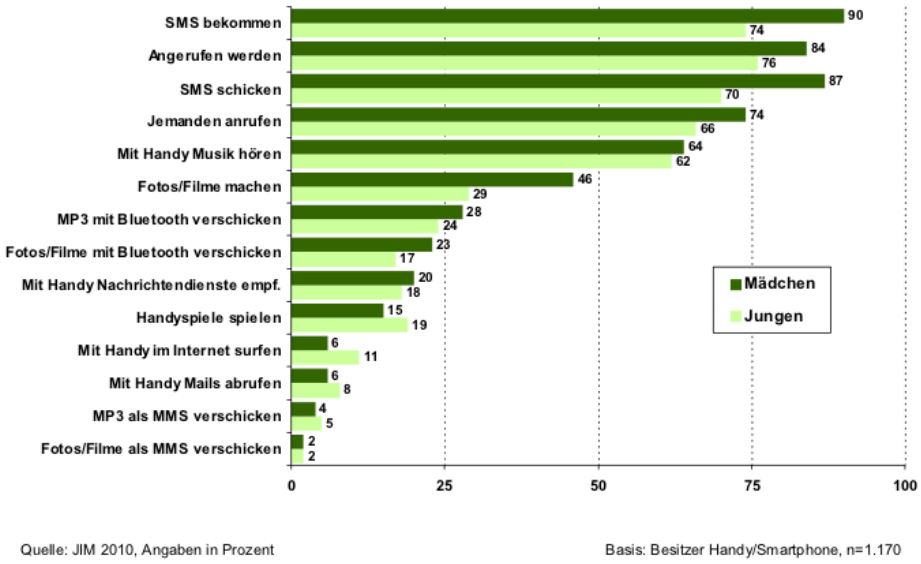

Abbildung 2.3.: Nutzung verschiedener Handy-Funktionen (täglich bzw. mehrmals pro Woche) (JIM 2010, S. 57-58)

Aus den vorgestellten Informationen ergibt sich somit einer erste These zur Gestaltung eines gendergerechten Moduls im Rahmen dieser Staatsexamensarbeit:

1. Der Einsatz von Smartphones, die über den Laptop programmiert werden, ermöglicht einen genderübergreifend motivierenden Unterricht.

Im Folgenden wird betrachtet, welche Rolle Arduino Mikrocontroller und Lego Mindstorms NXT bei der Gestaltung des Moduls spielen.

2.3. Motivation für verstärkt technische Themen

In Hinblick auf die eingangs erwähnte Gender Gap (siehe Abschnitt 2.1) wird Schülerinnen oftmals eine große Technikdistanz nachgesagt, während Schülern eher ein größeres Technikinteresse zugeschrieben wird. Der Roberta Abschlussbericht äußert sich dazu wie folgt:

> „Die Problematik im Technikverhältnis liegt weniger in der Nutzung von Technik – dies haben Mädchen und Frauen (wenn bisweilen im Freizeitbereich auch zeitlich verzögert) in ähnlichem Maße getan wie Jungen und Männer. Sie liegt vielmehr in der Frage technischer Konstruktion. Dies gilt zunächst für ‚harte' Technik und Hardware [...]" (Petersen et al., 2007, S. 170)

Nachdem in Abschnitt 2.2 mit Smartphones und Laptops die grundlegenden Medien für einen gendergerechten Informatikunterricht ausgewählt wurden, stellt sich die Frage, ob Schülerinnen über den Einsatz von Smartphones leichter für „harte" technische Themen zu begeistern sind, als dies im herkömmlichen Unterricht oder bei herkömmlicher Computernutzung der Fall ist. Im Rahmen des Roberta-Projekts wurde darüber berichtet, dass Mädchen über ansprechende Zugangsmöglichkeiten für technische Tätigkeiten begeistert werden konnten (Petersen et al., 2007, S. 169-172). Es kann nun versucht werden bei der Entwicklung der Lerneinheit auf diese Art und Weise genderübergreifend ebenfalls Motivation für verstärkt technische Themen und Tätigkeiten zu schaffen.

Sofern die These aus Abschnitt 2.2 gültig ist und der Einsatz von Smartphones genderübergreifend motivierend wirkt, könnte diese Motivation genutzt werden, um andere, technischere Themen erfolgreich aufzugreifen. Innerhalb dieses Moduls handelt bei dem verstärkt technischen Thema um den Einsatz von programmierbaren Arduino Mikrocontrollern. Dem gegenüber steht der Einsatz des Lego Mindstorms NXT als weniger technisches Gerät in der zweiten Variante des Moduls. Die SuS können vor Beginn des Moduls frei wählen, ob sie lieber mit dem Arduino oder dem NXT arbeiten. Eine Überbrückung von größerer Technikdistanz könnte sich darin äußern, dass sich viele Schüler und vor allem Schülerinnen für die Arbeit mit dem Arduino entscheiden.

Aus diesen Überlegungen ergibt sich somit eine zweite These zur Gestaltung einer gendergerechten Lerneinheit:

2. Durch den motivierenden Einsatz von Smartphones können SuS auch für verstärkt technische Themen (z.B.: Mikrocontroller) motiviert werden.

Die beiden Thesen dieser Staatsexamensarbeit, die in diesem Abschnitt und Abschnitt 2.2 aufgestellt worden sind, werden im Folgenden noch einmal kurz zusammengefasst, bevor auf die Entwicklung des Moduls eingegangen wird.

2.4. Zusammenfassung der Thesen

Aus den Inhalten, die in den vorigen Abschnitten vorgestellt wurden, haben sich nun zwei Thesen für die Entwicklung eines gendergerechten

Moduls für das Schülerlabor Informatik ergeben:

1. Der Einsatz von Smartphones, die über den Laptop programmiert werden, ermöglicht einen genderübergreifend motivierenden Unterricht.

2. Durch den motivierenden Einsatz von Smartphones können SuS auch für verstärkt technische Themen (z.B.: Mikrocontroller) motiviert werden.

Gemäß der ersten These spielen Smartphones und ihre Programmierung über Laptops bei der Entwicklung des Moduls die zentrale Rolle. Die zweite These wird im Modul durch den kombinierten Einsatz von NXTs beziehungsweise Arduinos mit Smartphones umgesetzt. Die beiden aufgestellten Thesen werden im Rahmen der qualitativen Evaluation untersucht.

Im Folgenden wird die Entwicklung der Selbstlerneinheit und der Software unter Berücksichtigung der aufgestellten Thesen erläutert.

3. Entwicklung des Moduls

3.1. Aufbau des Moduls

Das Modul für das Schülerlabor Informatik besteht neben den eingesetzten Geräten in erster Linie aus einer Selbstlerneinheit (Abbildung 3.1), welche von den SuS im Zweiergruppen an Laptops im Browser und in Eclipse bearbeitet wird. Im Idealfall gibt es pro Person einen Laptop, sodass alleine gelesen werden kann, während die Programmierung gemeinsam zu zweit an einem Laptop erfolgt. Als weitere Bestandteile kommen noch das bereitgestellte Android-Projekt für Eclipse, welches das Grundgerüst für die spätere App liefert, sowie natürlich die App selbst, die von den SuS am Ende des Moduls auch verwendet werden kann, hinzu. Zu Beginn der Lerneinheit befindet sich das Grundgerüst des Android-Projekts bereits in Eclipse, um unmittelbar in das eigentlich Thema einsteigen zu können.

Aus Hardware-Sicht kommen neben Android Smartphones und Laptops noch Lego Mindstorms NXT (Abbildung 3.2) beziehungsweise Arduino Mikrocontroller (Abbildung 3.3) zu Einsatz. Auf den Laptops muss zusätzlich zu Eclipse auch das Android SDK installiert sein. Bei den Smartphones muss im Vorfeld lediglich die Installation von Nicht-

Abbildung 3.1.: Screenshot der NXT-Version der Selbstlerneinheit

Market-Anwendungen erlaubt werden; die NXT können sogar gänzlich ohne Vorbereitung eingesetzt werden. Lego liefert bei seiner Firmware bereits eine gut dokumentierte API zur Bluetooth-Kommunikation mit (The Lego Group, 2006). Der Einsatz des Arduino Mikrocontrollers gestaltet sich jedoch komplizierter. Arduino bietet „nur“ eine Open Source Hardware-Plattform für Entwickler. Somit können in der Regel keine einsatzbereiten Produkte erworben werden, sondern lediglich die einzelnen Komponenten. Daher handelt es sich bei dem fahrbaren Arduino Mikrocontroller, der in diesem Modul eingesetzt wird, um einen Eigenbau. Die hauptsächlichen Bestandteile sind ein

Arduino Duemilanove, ein Bluetooth-Modul und zwei Motoren mit zugehörigem Controller. Details zum Aufbau und zu den Komponenten finden sich im Anhang A.3.

Abbildung 3.2.: Lego Mindstorms NXT

Die Bearbeitungszeit des gesamten Moduls muss je nach Leistungsniveau mit vier bis fünf Zeitstunden veranschlagt werden. Da es sich um eine Selbstlerneinheit handelt, ist keine intensive Betreuung der SuS erforderlich. Trotzdem sollte bei der Durchführung im Schülerlabor Informatik immer mindestens ein Tutor anwesend sein, um bei besonderen Fragen oder Problemen helfen zu können. Das Modul richtet sich an SuS der Oberstufe, da fundierte Java-Kenntnisse erforderlich sind und das Konzept der Objektorientierung bekannt sein muss.

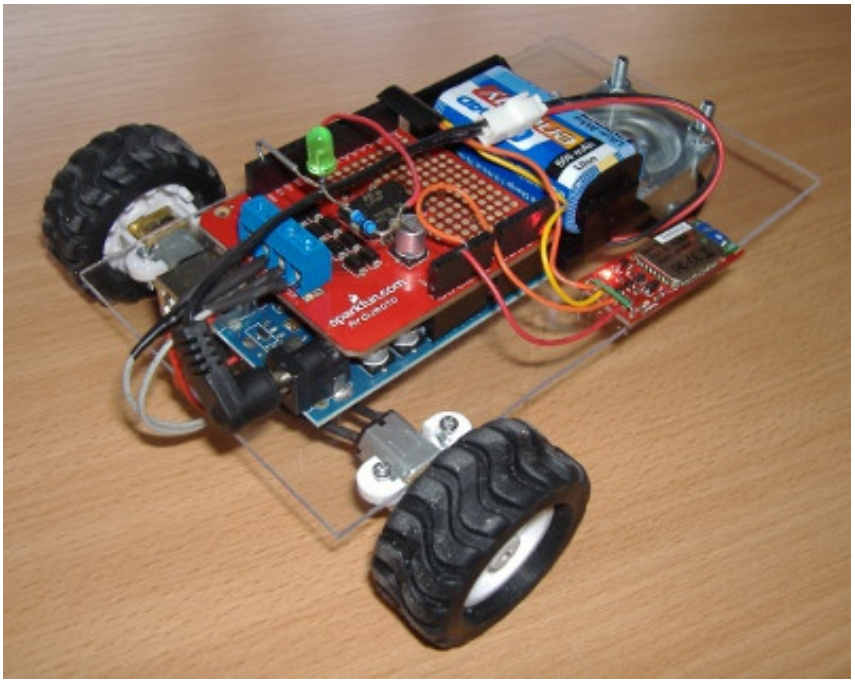

Abbildung 3.3.: Eigenbau eines fahrbaren Arduino Mikrocontrollers

3.2. Entwicklung der Software

3.2.1. Entwicklung der Android App

Wie eingangs erwähnt entwickeln die SuS im Laufe des Moduls eine Android App, mit der sich entweder ein NXT oder ein Arduino durch Auslesen des Lagesensors über Bluetooth fernsteuern lassen. Dementsprechend gibt es zwei Varianten der Android App: NXT Remote und Arduino Remote. Beide Apps sind nahezu identisch; der größte Unterschied besteht darin, dass für die Kommunikation mit dem NXT beziehungsweise dem Arduino unterschiedliche Bytes per Bluetooth verschickt werden. Da der NXT über einen rotierenden Arm verfügt,

der ein- und ausgeschaltet werden kann, findet sich in der NXT Remote zudem ein Button mehr (Abbildung 3.4).

Abbildung 3.4.: GUI der NXT Remote

Wie bereits herausgearbeitet wurde, kommt die Programmierung der App am Laptop eher den Interessen der Schüler entgegen. Im Gegenzug ist die App inhaltlich deshalb an die Interessen der Schülerinnen angelehnt (siehe Abschnitt 2.2). Den Kern der App bildet daher die Bluetooth-Kommunikation, wie sie auch bei der Übertragung von MP3s und Fotos genutzt wird. Ergänzend dazu werden die elementaren Aspekte der Android-Programmierung aufgegriffen, die nötig sind, um eine sinnvoll einsetzbare App zu erhalten. Beispiele hierfür

sind Activities, Interaktionen mit dem Nutzer, Beeinflussung der GUI und der Zugriff auf den Bewegungssensor.

Die Bluetooth-Kommunikation wird auf Basis der von Android zur Verfügung gestellten Schnittstellen komplett in der App implementiert. Dabei werden verschiedene allgemeingültige und bluetoothspezifische Elemente der Datenkommunikation genutzt, wie beispielsweise Bluetooth-Geräteklassen, Sockets und Streams.

Die Funktionalitäten der App wurden klar strukturiert und in vier Klassen aufgeteilt (siehe Abbildung 3.5):

- NXTremote.java bzw. ArduinoRemote.java enthält die Haupt-Activity, mit der der User interagiert
- CheckSensor.java ist für die Abfrage des Lagesensors des Smartphones zuständig
- ConnectBluetooth.java managt die Bluetooth-Kommunikation
- SelectDevice.java verwaltet die Dropdown-Auswahl der Bluetooth-Geräte

Von den SuS werden keine speziellen Vorkenntnisse in der Android-Programmierung benötigt, um die App im Rahmen der Lerneinheit selbst zu programmieren. Sie müssen lediglich fundierte Kenntnisse der Objektorientierung und Java-Programmierung mitbringen.

Exemplarische Erläuterung des Quelltextes

An dieser Stelle sollen entscheidende Stellen des Quelltextes zur Bluetooth-Kommunikation kurz aufgezeigt und erläutert werden, um ein

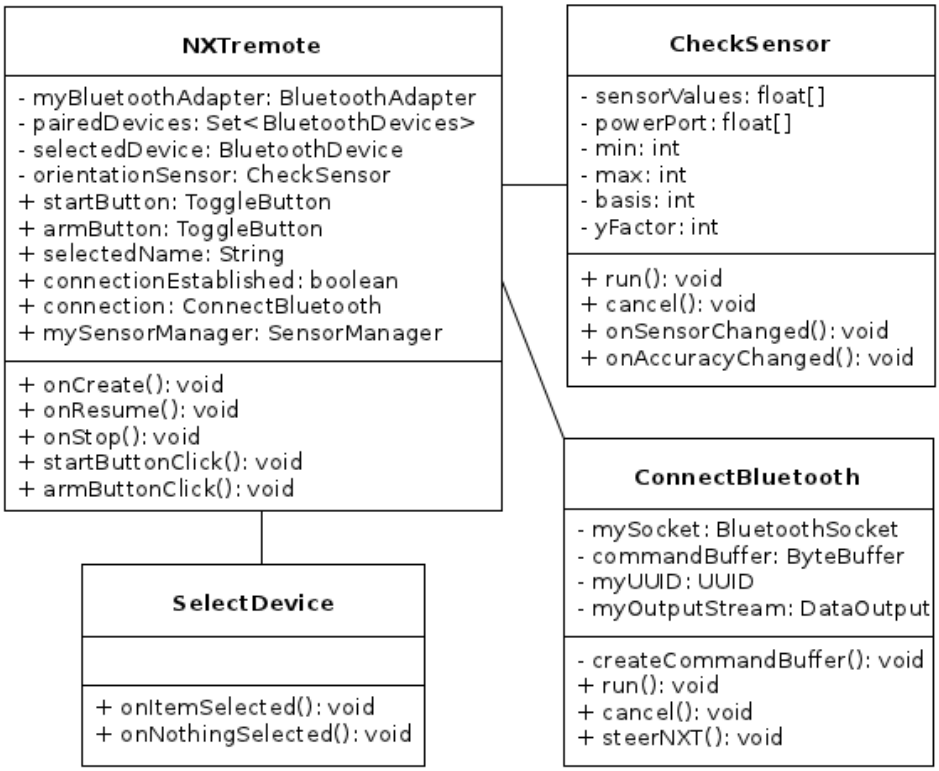

Abbildung 3.5.: UML-Diagramm der NXT Remote

grundlegendes Verständnis für die Funktionsweise und die Inhalte der App zu schaffen. Sämtliche Details, wie die umfassenden Erklärungen der Lerneinheit oder die vollständigen Quelltexte, können im Anhang gefunden werden. Die folgenden Beispiele sind der NXT-Version entnommen und sind analog auch in der Arduino-Version zu finden (siehe Anhang A.2).

In der App gilt es zunächst den Bluetooth-Adapter des Smartphones mit Hilfe des Betriebssystems zu identifizieren und ihn als Objekt in das Programm einzubinden. Anschließend muss die Bluetooth-Funktion des Smartphones von der App gegebenenfalls erst aktiviert

werden, bevor das Programm weiterarbeiten kann:

```
BluetoothAdapter myBluetoothAdapter;
myBluetoothAdapter = BluetoothAdapter.
   getDefaultAdapter();

if (!myBluetoothAdapter.isEnabled()) {
   Intent enableBtIntent = new Intent(
      BluetoothAdapter.ACTION_REQUEST_ENABLE);
   int requestBt = 0;
   startActivityForResult(enableBtIntent,
      requestBt);
}
```

Im nächsten Schritt müssen die gepaarten Bluetooth-Geräte abgefragt werden. Die Liste der Geräte wird zusätzlich nach einer bestimmten Bluetooth-Geräteklasse gefiltert, um beispielsweise nur NXTs (Geräteklasse 2052) auszuwählen:

```
pairedDevices = myBluetoothAdapter.
   getBondedDevices();
ArrayAdapter<String> pairedDevicesArray = new
   ArrayAdapter<String>(this, android.R.layout.
   simple_spinner_item);
pairedDevicesArray.setDropDownViewResource(
   android.R.layout.simple_spinner_dropdown_item
   );
if (pairedDevices.size() > 0) {
   for (BluetoothDevice device : pairedDevices)
```

```
        {
6     if ((myBluetoothAdapter.isEnabled()) && (
         device.getBluetoothClass().getDeviceClass
         () == 2052))
7       pairedDevicesArray.add(device.getName());
8     }
9 }
```

Anschließend wird ein Socket geöffnet, eine Verbindung zu einem ausgewählten NXT aufgebaut und ein `OutputStream` zur Übertragung der Daten erstellt:

```
1 BluetoothSocket mySocket;
2 mySocket = device.
     createRfcommSocketToServiceRecord(myUUID);
3 mySocket.connect();
4 DataOutput myOutputStream;
5 myOutputStream = new DataOutputStream(mySocket.
     getOutputStream());
```

Mit der Funktion `myOutputStream.write(byte[] buffer)` können nun Daten vom Smartphone über den Stream zum NXT geschickt werden. Die exakten Byte-Folgen zur Steuerung des NXT über Bluetooth lassen sich der offiziellen Dokumentation entnehmen (The Lego Group, 2006).

3.2.2. Entwicklung des Arduino Sketches

Die Programmierung des Arduino Mikrocontrollers geschieht über die sehr einfache Arduino IDE, die in der Lerneinheit kaum einer Einführung bedarf (Abbildung 3.6). Der Sketch – so wird ein Programm für den Arduino genannt – kann unkompliziert per USB auf den Mikrocontroller übertragen werden. Zwar kann der Ardunio nicht in Java programmiert werden, jedoch erfolgt die Programmierung in einem C-Derivat, das Java sehr stark ähnelt. Somit entstehen für die SuS an dieser Stelle keine Schwierigkeiten durch den Wechsel.

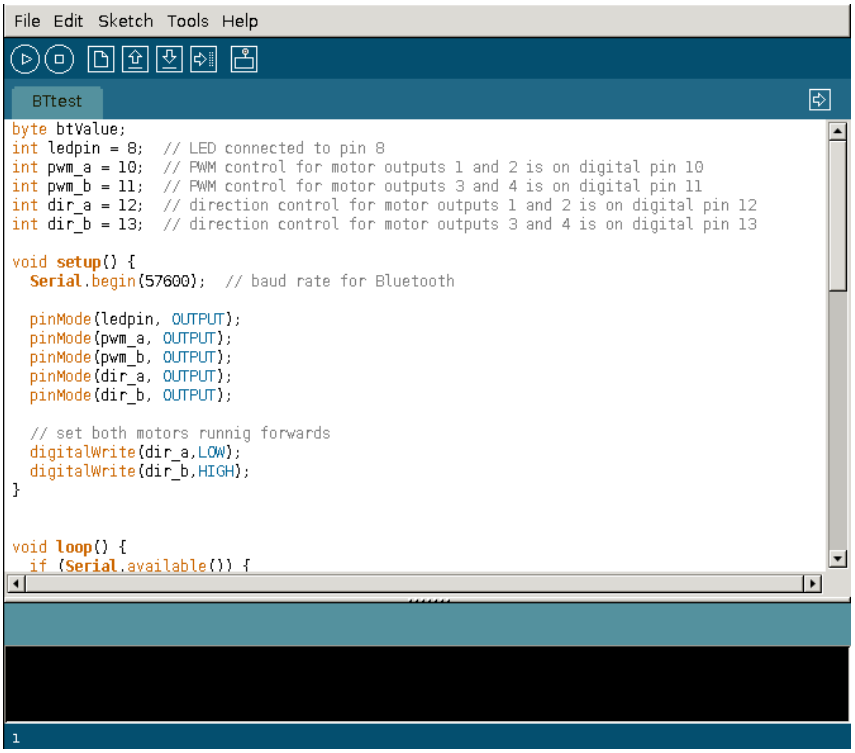

Abbildung 3.6.: Arduino IDE

Das Programm des Mikrocontrollers ist vergleichsweise banal und

besteht hauptsächlich aus dem Einlesen des Bluetooth-Streams und mehreren if-Abfragen zum gezielten Verarbeiten der Daten. Je nach Wert der Bytes werden die beiden Motoren des fahrbaren Arduino mit unterschiedlicher Leistung und Drehrichtung betrieben.

Im Folgenden wird exemplarisch ein kurzer Auszug aus dem Quelltext wiedergegeben:

```
// Daten des Bluetooth-Streams lesen
if (Serial.available()) {
   btValue = Serial.read();
}

// Daten an die beiden Motoren weitergeben
if (btValue == 251) {
   digitalWrite(dir_a,LOW);
   digitalWrite(dir_b,HIGH);
}
```

Der vollständige Quelltext kann in Abschnitt A.3.3 des Anhangs gefunden werden. Die einzelnen Elemente der Bluetooth-Kommunikation müssen für den Mikrocontroller nicht selbst implementiert werden.

3.3. Entwicklung der Selbstlerneinheit

3.3.1. Strukturierung des Inhalts

In der, mit HTML und jQuery gestalteten, Lernumgebung können die SuS selbstständig lernen, wie sie mit Java erstmals eine Android App in Eclipse entwickeln. Dabei sind, wie bereits in Abschnitt 3.2.1 erwähnt, Vorkenntnisse in Java und bezüglich der Objektorientierung erforderlich, sodass das Modul für SuS der Oberstufe gedacht ist. Der Aufbau der Lerneinheit ist an das Leitprogramm-Prinzip angelehnt und die SuS werden im Rahmen der Lerneinheit relativ stark geleitet.

Der inhaltliche Aufbau der Selbstlerneinheit orientiert sich in erster Linie an der Struktur der App und den Komponenten, die für die Bluetooth-Kommunikation implementiert werden müssen (siehe Tabelle 3.1). Die ersten vier Kapitel sind somit für die NXT- und die Arduino-Version identisch aufgebaut. Erst gegen Ende der Lerneinheit unterscheidet sich die Struktur. In der NXT-Version wurden nach der Programmierung der App bereits alle obligatorischen Funktionen implementiert. Abschließend kann optional der zusätzliche Button zur Steuerung des Roboterarms programmiert werden. Bei der Arduino-Version hingegen muss nach Fertigstellung der App noch der Mikrocontroller programmiert werden, bevor er ferngesteuert werden kann.

Lerneinheit NXT Remote	Lerneinheit Arduino Remote
1. Einstieg in die Android-Programmierung	1. Einstieg in die Android-Programmierung
2. Das Grundgerüst der App	2. Das Grundgerüst der App
3. Die verschiedenen Zustände der App	3. Die verschiedenen Zustände der App
4. Eine Bluetooth-Verbindung aufbauen	4. Eine Bluetooth-Verbindung aufbauen
5. Der Roboter bewegt sich!	5. Der Lagesensor
6. Bonus: Der Arm	6. Der Arduino bewegt sich!

Tabelle 3.1.: Kapitelübersicht der Lerneinheiten

3.3.2. Didaktische Überlegungen

Die Lerneinheit wurde in Anlehnung an das Leitprogramm-Prinzip entwickelt (Kirchgraber et al., 1999). Die SuS können sich die Inhalte selbstständig aneignen und finden alle relevanten Informationen didaktisch aufbereitet in den entsprechenden Kapiteln.

Bei der Bearbeitung werden den SuS diverse Orientierungshilfen geboten: Zunächst wird das Modul kurz vorgestellt und eine Übersicht über die kommenden Kapitel gegeben. Es wird zudem ein Farbcode eingeführt, mit dessen Hilfe die SuS Inhalte sofort zuordnen können: Grün steht für Fragen, Blau für Informationen rund um Bluetooth

und Schwarz für Quelltextausschnitte. Diese Inhalte befinden sich in farblich gekennzeichneten Boxen und lockern so den Text der Lerneinheit auf (siehe Abbildung 3.7). Gemäß dem Leitprogramm-Prinzip wechseln sich neue Inhalte häufig mit Aufgaben für die SuS ab. Die Lösungen können von den SuS bei Bedarf direkt unter den Fragen als Selbstkontrolle eingeblendet werden. Auch finden sich am Ende jedes Kapitels Hilfestellungen in Form größerer, kommentierter Quelltextpassagen, die eingeblendet werden können und verhindern sollen, dass SuS an einem Detail scheitern und dadurch nicht weiter kommen.

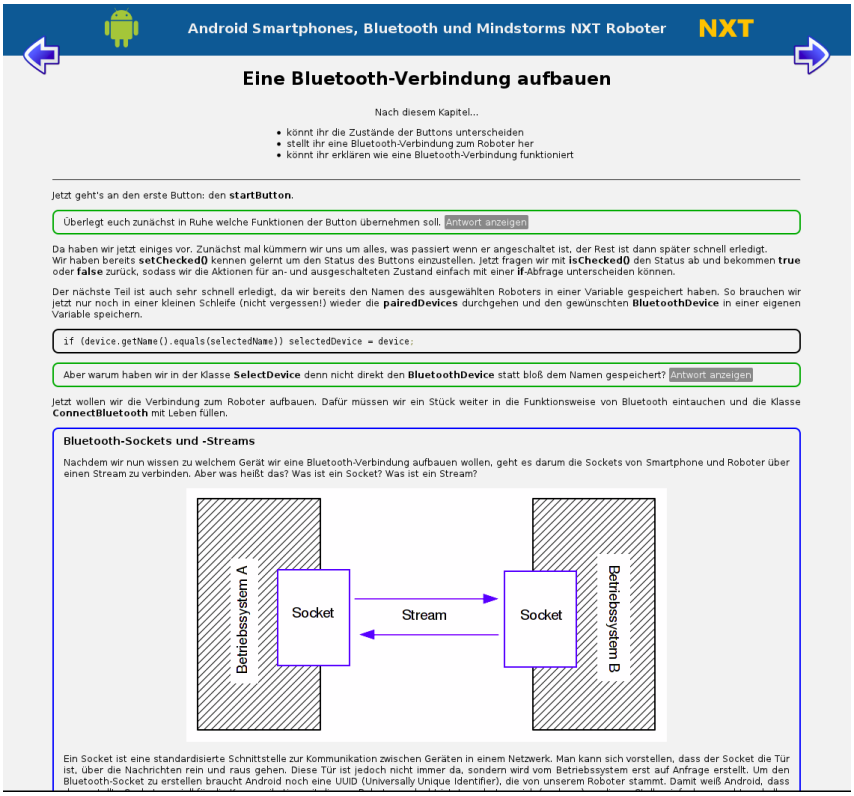

Abbildung 3.7.: Screenshot der Lerneinheit mit verschiedenfarbigen Boxen

Im Unterschied zu Leitprogrammen wird in diesem Modul das Mastery-Learning-Prinzip jedoch nicht in der klassischen Form als obligatorischer Test am Kapitelende umgesetzt (Krapp/Weidenmann, 2006, S. 620). Stattdessen steht es den SuS zunächst frei durch die komplette Lerneinheit vor- und zurück zu navigieren. Jedes Kapitel erweitert die Android App jedoch um bestimmte Funktionen, die in dem darauf folgenden Kapitel zwingend benötigt werden. Daraus ergibt sich ganz automatisch, dass die SuS ein Kapitel zuerst erfolgreich absolvieren müssen, bevor sie zum nächsten Kapitel übergehen können. Im Gegensatz zu Blooms klassischem Mastery Learning entsteht dabei jedoch nicht der Eindruck eines Prüfungszwangs. Da das erfolgreiche abschließen eines Kapitels zum normalen Entwicklungsprozess der App gehört, sind die SuS eher von sich aus daran interessiert und somit auch eher intrinsisch motiviert erfolgreich zu sein.

Während der Bearbeitung der einzelnen Kapitel werden die SuS inhaltlich sehr stark geleitet. Diese Herangehensweise wurde aus mehreren Gründen gewählt: Ein wichtiger Punkt betrifft den Unterschied zwischen leistungsstarken und leistungsschwachen SuS in Kombination mit dem häufig sehr heterogenem Vorwissen, das besonders in der Informatik ein ausgeprägtes Problem ist. Je offener eine Lerneinheit gestaltet ist, desto eher fühlen sich leistungsschwächere SuS verloren oder überfordert.

> „Deshalb ist auch mit der Gefahr eines Schereneffektes zu rechnen: Da Leistungsstarke in der Regel von situierten Lernumgebungen sehr viel stärker profitieren als Leistungsschwache, besteht die Gefahr, dass sich die Kluft

> zwischen ‚guten‘ und ‚schlechten‘ Schülern [...] noch vergrößert.“ (Krapp/Weidenmann, 2006, S. 635)

Dies ist selbstverständlich zu vermeiden, insbesondere wenn Begeisterung für die Informatik geweckt werden soll.

Es kann zwar zur Zeit davon ausgegangen werden, dass die SuS wenig bis kein Vorwissen in der Android-Programmierung mitbringen, da dieses Arbeitsgebiet noch vergleichsweise jung ist, aber dennoch bietet sich ein geleiteter Einstieg in das Themenfeld an, um aus den zuvor genannten Gründen „Desorientierung und Überforderung“ (Krapp/Weidenmann, 2006, S. 635) zu vermeiden. Zudem ist es auch gut möglich, dass das Vorwissen zur Android-Programmierung in Zukunft mit der Verbreitung von Smartphones im privaten Bereich und im Informatikunterricht weitaus heterogener werden wird.

Das geleitete Arbeiten ermöglicht es auch, dass im Rahmen des Moduls nicht systematisch sämtliche Grundlagen der Android-Programmierung besprochen werden müssen, sondern schnellstmöglich von der Theorie zur Entwicklung einer sinnvollen und motivierenden App übergegangen werden kann.

Als weiteres Argument kann die Zeiteffizienz beim Einsatz genannt werden (Krapp/Weidenmann, 2006, S. 634-636). Die Gruppen der SuS befinden sich häufig für einen Tag oder weniger im Schülerlabor Informatik, sodass eine konstruktivistische, situierte Lerneinheit mit sehr freien Arbeits- und Gestaltungsmöglichkeiten leicht den Zeitrahmen sprengen kann. Mit der Anlehnung an das Leitprogramm-Prinzip wird für das Modul eine Bearbeitungszeit von etwa vier bis fünf Zeitstunden benötigt. Die SuS können ihr Arbeitstempo in den Zweiergruppen da-

bei selbst bestimmen, wodurch Rücksicht auf das individuelle Lernen genommen wird. Langsamere SuS haben so genügend Zeit zur Bearbeitung und können in Ruhe Hilfestellungen lesen, während schnellere SuS die Möglichkeit haben die App nach eigenen Wünschen weiter zu modifizieren oder einfach das Ergebnis ihrer Arbeit zu genießen und mit der App ihren NXT oder Arduino fern zu steuern.

Ausgleichend zum geleiteten Vorgehen sind die Inhalte des Moduls zudem stark handlungsorientiert, sodass die SuS sich aktiv und in verschiedenen Umgebungen (Browser, Eclipse, Smartphone, NXT, Arduino) mit den Aufgaben auseinander setzen können. Die vorstrukturierten Inhalte werden dadurch abwechslungsreich und motivierend aufgelockert.

Ausgehend vom vorliegenden Modul ließe sich zudem eine umfangreichere Lerneinheit entwickeln, die den SuS später freieres Arbeiten in einer eher kognitivistisch-konstruktivistisch gestalteten Lernumgebung – ähnlich dem Cognitive Apprenticeship – ermöglicht. Im Ausblick in Kapitel 6 werden einige weiterführende Möglichkeiten besprochen.

Thematische Einordnung

Obwohl das Modul nicht für den Einsatz im regulären Informatikunterricht konzipiert wurde, lässt es sich thematisch in den Lehrplan Informatik der Sekundarstufe II für Gymnasium und Gesamtschulen in NRW einordnen. Die Inhalte können dem Themenfeld „Technische, funktionale und organisatorische Prinzipien von Hard- und Softwaresystemen kennen lernen und einordnen“ zugeordnet werden, da neben

Aspekten der Datenkommunikation und der Vernetzung auch das Zusammenspiel mehrerer Systemkomponenten und ihrer Funktionen unter Android behandelt wird (Lehrplan Informatik, 1999, S. 15-16).

4. Evaluation

4.1. Methodik

4.1.1. Problemzentriertes Interview

Das entwickelte Modul wurde hinsichtlich der beiden, in Kapitel 2 aufgestellten, Thesen qualitativ evaluiert. Es wurde eine qualitative Evaluation gewählt, da die subjektiven Eindrücke und die Meinungen der SuS eingefangen werden sollten. Es ging nicht, wie bei einer quantitativen Evaluation, darum exakt definierte, standardisierte Ergebnisse zu erhalten, „sondern [...] Motivation zu erfragen, etwas über den Lern- und Arbeitsstil, die persönlichen Ambitionen und Aspirationen in Erfahrung zu bringen.“ (Kuckart et al., 2008, S. 12) Die Evaluation wurde als problemzentriertes Interview durchgeführt, um ein möglichst unvoreingenommenes Bild von der individuellen Wahrnehmung der Teilnehmerinnen und Teilnehmer zu erhalten.

Das Interview wurde somit als Leitfadeninterview durchgeführt und den SuS wurden keine Antwortmöglichkeiten vorgegeben. Stattdessen wurden sie zum freien Erzählen angeleitet und der Interviewverlauf wurde, soweit das Thema es zuließ, an ihre Äußerungen angepasst.

Die Instrumente dieser Interviewmethode sind:

- ein Kurzfragebogen
- Audioaufzeichnungen
- ein Leitfaden
- Postskripte

Der Kurzfragebogen zur Erhebung sozialer Daten war im Rahmen dieser Evaluation nicht nötig, da die relevanten Informationen bereits im Vorfeld bereitgestellt worden waren. Von den Interviews wurden Audioaufzeichnungen erstellt, um sich während der Interviews vollständig auf das Gespräch und die Situation konzentrieren zu können. Der Leitfaden der Interviews diente als „Gedächtnisstütze und Orientierungsrahmen zur Sicherung der Vergleichbarkeit der Interviews“ (Witzel, 2000). Im Anschluss an die Interviews wurden Postskripte erstellt, in denen spontane Eindrücke, Auffälligkeiten oder nonverbale Aspekte kurz notiert wurden.

Obwohl es eigentlich ein theoriegenerierendes Verfahren ist, wurde das problemzentrierte Interview in diesem Fall gezielt als Möglichkeit eingesetzt, um die durchgeführten Maßnahmen unmittelbar zu untersuchen. Für die Durchführung war vor allem wichtig, dass den SuS vorab gesagt wurde, zu welchem Zweck und vor welchem theoretischen Hintergrund die Interviews geführt wurden und dass man als Fragender „sein Erkenntnisinteresse offen legt und deutlich macht, dass die Explikationen der Interviewten nicht als Ausdruck von intellektuellen Leistungen bewertet, sondern als individuelle Vorstellungen und Meinungen akzeptiert werden.“ (Witzel, 2000)

Eine anschließende Transkription der Interviews und Typenbildung anhand der Daten, wie von Witzel vorgeschlagen, wurde aufgrund der geringen Anzahl der durchgeführten Interviews nicht vorgenommen. Die Ergebnisse wurden direkt aus den Einzelfallanalysen der Interviews abgeleitet (Witzel, 2000).

4.1.2. Interviewleitfaden

Besonderes Augenmerk lag im Vorfeld auf der theoriegeleiteten Gestaltung des Interviewleitfadens, welcher vollständig im Anhang A.4 zu finden ist. Der Leitfaden ist zweiteilig, da sowohl vor, als auch nach der Durchführung der Lerneinheit ein kurzes Interview mit den SuS geführt wurde. Der Ablauf der beiden Interviewphasen wird im Folgenden zusammenfassend dargestellt.

Vor der Durchführung wird zunächst das Vorhaben der Staatsexamensarbeit kurz vorgestellt und der theoretische Hintergrund erläutert. Anfangs werden die SuS zu ihren Einstellungen zur Informatik befragt. Anschließend sollen sie sowohl ihre eigenen Vermutungen zur Gender Gap in der Informatik äußern, als auch ihren eigenen Umgang mit Technik einschätzen. Dadurch wird ein Eindruck von der Technikdistanz der SuS gewonnen. Zum Abschluss des ersten Interviewteils wird konkret auf die thesengestützte Medienwahl eingegangen und nach dem Interesse an der Arbeit mit Smartphones, Lego Mindstorms NXT und Arduino Mikrocontrollern gefragt.

Nach der Durchführung des Moduls wird den SuS zunächst die Möglichkeit für freies Feedback gegeben, um auf diese Weise bereits Im-

pulse für das weitere Interviewgespräch sammeln zu können. Anschließend wird stärker auf die eingesetzten Geräte eingegangen und geschaut, wie die Nutzung der Smartphones, NXTs, Arduinos und Laptops empfunden wurde. Desweiteren wird überprüft, ob sich die Selbsteinschätzung der SuS bezüglich ihres eigenen Umgangs mit Technik durch die Lerneinheit verändert hat. Weitere Fragen beziehen sich auf den Aufbau der Lerneinheit, um sicher zu stellen, dass die Motivation an dieser Stelle nicht auf eine Art und Weise beeinflusst wurde, die die Rückschlüsse auf die potentiell motivierende Wirkung von Smartphones verfälschen würde. Exemplarisch wird hier auch eine These aus dem Roberta-Abschlussbericht aufgegriffen, nach der Fahrzeuge mit Rädern statt Beinen – was sowohl auf den NXT, als auch den eingesetzten Arduino zutrifft – eher Schülern entgegen kommen (Petersen et al., 2007, S. 166). Anhand des Beispiels wird nach den Vorlieben, Interessen und Wünschen der SuS gefragt.

Abgeschlossen werden die Interviews mit kurzen, allgemeinen Fragen zum Interviewablauf selbst. Es wird gefragt ob die Interviews als unangenehm empfunden wurden oder nach Meinung der SuS noch etwas fehlt. Zuletzt wird den SuS Angeboten ihnen die anonymisierten Ergebnisse nach Abschluss der Auswertung zukommen zu lassen.

4.2. Durchführung

Die Durchführung und Evaluation der Lerneinheit fand am 02.02.2011 und 03.02.2011 am Ernst-Moritz-Arndt-Gymnasium in Bonn statt. Ansprechpartner dort war Nils van den Boom, aus dessen Informatik-

kursen die teilnehmenden SuS stammten.

An jedem der beiden Tage wurde die Lerneinheit mit einer Gruppe von SuS durchgeführt. Dabei sollten geschlechtergetrennt je vier Schülerinnen und vier Schüler die Lerneinheit durchführen. Die Gruppengröße war durch das zur Verfügung stehende Material begrenzt. Die Geschlechtertrennung wurde vorgenommen, damit die Äußerungen in den Gruppeninterviews nicht durch Vorurteile über die Rollenverteilung der Geschlechter geprägt werden würden. Stattdessen sollten die individuellen Meinungen durch die Trennung möglichst unverfälscht und offen eingefangen werden können.

Unabhängig von der Planung ergaben sich jedoch kurzfristig Gruppengrößen von fünf Schülerinnen und drei Schülern. Dies hat jedoch weder bei der Durchführung, noch bei der Evaluation zu Problemen geführt. Die Gruppe der Schülerinnen setzte sich aus drei Schülerinnen des Informatik Grundkurses der Stufe 11 und zwei Schülerinnen des Informatik Leistungskurses der Stufe 13 zusammen; die Gruppe der Schüler kam komplett aus dem Informatik Grundkurs der Stufe 11. Die SuS konnten sich selbst aussuchen, ob sie lieber mit dem Lego Mindstorms NXT oder dem Arduino Mikrocontroller arbeiteten.

Der Einsatz des Moduls war mit einem halben Tag veranschlagt worden. An beiden Tagen wurde morgens um 8 Uhr mit dem Aufbau begonnen, sodass die Durchführung für die SuS im Rahmen eines normalen Schulvormittags stattfand. Nach dem einleitenden Gruppeninterview von circa 15 Minuten bearbeiteten die SuS die Lerneinheit jeweils selbstständig für vier bis fünf Stunden. Danach wurde das abschließende Gruppeninterview durchgeführt, welches wieder etwa 15

Minuten dauerte.

Am ersten Tag wurde der Ablauf zwischenzeitlich durch verschiedene Probleme bei der App-Entwicklung gestört, die entweder durch kleinere (Tipp-)Fehler in der Lerneinheit oder durch mangelnde Vorkenntnisse zur Objektorientierung auftraten. Dies führte dazu, dass am ersten Tag in der Gruppe der Schülerinnen zunächst keine lauffähige App erstellt werden konnte. Nachdem die Lerneinheit beendet worden war, wurde der entscheidende, kleine Fehler jedoch noch entdeckt und behoben, sodass die fertige App trotzdem kurz eingesetzt werden konnten. Die Fehler in der Lerneinheit wurden anschließend korrigiert. Am zweiten Tag wurde das Modul mit der Gruppe der Schüler durchgeführt. Dabei gab es keine nennenswerten Probleme mehr.

Aus den Interviews ging bereits sehr früh hervor, dass das Modul trotz der teilweise aufgetretenen Probleme von beiden Gruppen als positiv und motivierend empfunden worden war. So sagte eine Schülerin beispielsweise zum Einsatz der Smartphones: „Das ist voll begeisternd!“

Im Folgenden werden die Interviews in Hinblick auf die aufgestellten Thesen ausgewertet.

4.3. Auswertung

4.3.1. Gender Gap

Anhand des Interviewleitfadens wurden die Sichtweisen und Einstellungen der SuS zu den einzelnen Punkte der beiden Thesen erfragt.

Zuvor sollten die SuS jedoch von ihren eigenen Ansichten zur Gender Gap im Informatikunterricht erzählen. Dabei wurde von ihnen immer direkt hervorgehoben, dass dieses Phänomen nicht bloß in der Informatik auftritt, sondern allgemein im MINT-Bereich, wobei sie ihre Aussagen auf die Geschlechterverhältnisse in ihren eigenen Kursen stützten. Als Ursachen für die geringe Anzahl Schülerinnen in der Informatik wurden hauptsächlich Angst vor Mathematik, Einschüchterung durch das große Vorwissen von „Freaks“ und das größere Interesse an kreativen Aufgaben angeführt:

> „Viele Mädchen kommen ja auch schon mit Mathe nicht so gut klar. Wenn man dann hört ‚Informatik‘, sind vielleicht viele auch erstmal abgeschreckt.“ — Schülerin 1

> „Mich schreckt das so ein bisschen ab, wenn ich sehe welche Freaks andere Leute sind und was die dann alles können.“ — Schülerin 2

> „Sie [Anm.: Schülerinnen] mögen dann wahrscheinlich eher kreative Aufgabe.“ — Schüler 6

An dieser Stelle ist es bemerkenswert, dass die Informatik von einigen der befragten SuS nicht als eine kreative Disziplin empfunden wird.

Desweiteren war die vorherrschende Meinung, dass Schülerinnen insgesamt zu oft im Vorfeld von der Informatik abgeschreckt werden, bevor sie sich überhaupt darauf einlassen und es dadurch zur Gender Gap kommt.

4.3.2. Erste These

Zu Beginn soll kurz die erste These wiederholt werden:

1. Der Einsatz von Smartphones, die über den Laptop programmiert werden, ermöglicht einen genderübergreifend motivierenden Unterricht.

Im Vorfeld und im Nachhinein wurden die SuS zu ihrem Interesse an der Arbeit mit Smartphones befragt. Vor der Bearbeitung der Lerneinheit war bereits bei allen SuS verhaltenes bis großes Interesse vorhanden:

> „Das [Anm.: Motivation durch Smartphones] ist so eine Sache, ob man sich jetzt dafür interessiert oder nicht. Ich glaube das liegt jetzt nicht nur daran ob man etwas mit einem Smartphone macht. [...] Es macht es insgesamt ein bisschen interessanter, wenn man etwas mit einem Smartphone macht statt mit einem Laptop.“ — Schülerin 2

> „Ist halt schon immer praktisch, wenn das [Anm.: die Lerneinheit] dann so praxisorientiert ist.“ — Schüler 6

> „Also ich finde das [Anm.: Motivation durch Smartphones] eine super Idee!“ — Schüler 7

Kurz nachdem die SuS mit der Bearbeitung des Moduls begonnen hatten, konnte bereits eine merkliche Steigerung des Interesses beobachtet werden. Zunächst stand bei den einzelnen Gruppen das Smartphone im Zentrum der Aufmerksamkeit, nach relativ kurzer Zeit ging das Interesse dann auch auf die Lernumgebung am Laptop über, da die

SuS wissen wollten, was genau sie mit dem Smartphone im Rahmen des Moduls machen können.

Nach Abschluss der Moduls waren die SuS einstimmig vom Einsatz der Smartphones begeistert. Auch SuS, die zuvor eher zurückhaltend gewesen waren, äußerten sich im Interview sehr positiv:

> „Was mir auf jeden Fall gefallen hat ist halt diese Praxisnähe, [...] wenn man am Ende dann mit seinem Handy einfach auf 100 Meter Entfernung einen Roboter fernsteuern kann ist das schon cool.“ — Schüler 6

> „Das ist ja quasi theoretisch und praktisch in einem, sodass man das dann auch direkt anwenden kann.“ — Schülerin 3

> „Es ist viel cooler wenn man sieht, was man gemacht hat und dass es theoretisch auch das eigene Handy kann.“ — Schülerin 1

> „Das ist voll begeisternd!“ — Schülerin 2

Die SuS betonten auch, dass sie die Arbeit ohne Smartphone weitaus weniger reizvoll und interessant gefunden hätten. Die Frage nach dem ausschließlichen Einsatz von Laptops stieß auf geschlossene Ablehnung. Auch der Vorschlag Android-Tablets anstatt Smartphones einzusetzen rief lediglich gemischte Reaktionen hervor.

Das genderübergreifende motivieren mit Smartphones hat somit in beiden Gruppen sehr gut funktioniert und bestätigt die erste These für diese Fälle vollkommen.

4.3.3. Zweite These

Zunächst soll auch die zweite These kurz wiederholt werden:

2. Durch den motivierenden Einsatz von Smartphones können SuS auch für verstärkt technische Themen (z.B.: Mikrocontroller) motiviert werden.

Im Zuge der Evaluation konnte nicht festgestellt werden, dass die Motivation mit Mikrocontrollern zu arbeiten durch den Einsatz von Smartphones beeinflusst wurde. Trotz des einheitlichen Interesses an der Arbeit mit den Smartphones, wollte nur ein Teil der SuS mit dem Arduino arbeiten. Von den acht SuS entschieden sich drei Schülerinnen und zwei Schüler für die Arbeit mit dem Mikrocontroller.

Der Unterschied im Interesse rührte jedoch lediglich – gender- und altersunabhängig – von den individuellen Vorkenntnissen der SuS her. SuS mit mehr Vorkenntnissen bevorzugten den Arduino, während SuS mit weniger Vorkenntnissen den NXT bevorzugten. Ihre Vorkenntnisse schätzten die SuS in den Interviews selbst ein. Die SuS, die am Arduino interessiert waren, äußerten sich alle in etwa wie folgt:

> „Ich guck mir dieses Ding [Anm.: Arduino] an und denke: Hey, was kann man da alles dranbauen?“ — Schülerin 1

Zwar wollten die SuS, die weniger Vorkenntnissen hatten, lieber mit dem NXT arbeiten, jedoch fanden sie den fahrbaren Arduino nach eigenen Aussagen prinzipiell auch interessant. Die Arbeit mit dem NXT trauten sie sich jedoch selbst eher zu.

Durch das Bearbeiten des Moduls stieg in den Arduino-Gruppen das Interesse am Mikrocontroller noch weiter und die SuS würden nach eigenen Aussagen gerne auch in Zukunft mit solchen Mikrocontrollern arbeiten. Nach der Lerneinheit hatten die SuS das Gefühl eine konkretere Vorstellung davon zu haben, wie Mikrocontroller eingesetzt werden können:

> „Macht Spass [Anm.: das Programmieren des Mikrocontrollers] und ist auf jeden Fall einfacher als so, wie ich das bis jetzt immer gesehen habe.“ — Schüler 6

Die NXT-Gruppen waren mit ihrer Wahl ebenfalls zufrieden, die Einstellung zur Arbeit mit dem NXT änderte sich jedoch nicht. Dies kann darauf zurück geführt werden, dass der NXT auch lediglich genutzt wurde, während mit dem Arduino aktiv gearbeitet wurde.

Obwohl also bei allen SuS ein grundlegendes Interesse am Mikrocontroller vorhanden war und der Einsatz von Smartphones motivierend wirkte, konnte die zweite These im Rahmen dieser Evaluation nicht bestätigt werden. Die Motivation für die Arbeit mit den Mikrocontrollern wurde durch die Smartphones nicht merklich beeinflusst. Sie war aber genderübergreifend vom Vorwissen der SuS abhängig.

4.3.4. Feedback zum Modul

Unabhängig von der Überprüfung der Thesen haben die SuS weiteres Feedback zum Aufbau und zu den Inhalten des Moduls gegeben. So wurde das Thema Bluetooth von den SuS als interessant und relevant empfunden und auch der Umfang der Moduls hat den SuS gefallen:

> „Für das Pensum, was wir hatten um die Dinge zu programmieren, war das [Anm.: die Zeit] eigentlich ziemlich perfekt.“ — Schüler 1

Es wurde jedoch auch mehrfach der Wunsch geäußert freier arbeiten zu können und den NXT und auch den Arduino selbst zusammen zu bauen. Die SuS wären bereit gewesen sich dafür auch länger mit einer entsprechenden Lerneinheit zu diesem Thema zu befassen. Auch der Vorschlag das Modul dazu auf zwei Tage auszudehnen stieß auf Zuspruch. Gleichzeitig war den SuS aber auch bewusst, dass eine relativ stark leitende Lernumgebung nötig ist, um das Thema in kürzerer Zeit zu erarbeiten.

Im Ausblick in Kapitel 6 wurden einige Ideen zur Umgestaltung und weiteren Ausgestaltung der Lerneinheit aufgegriffen und fortgeführt.

5. Fazit

Es ist gelungen im Rahmen dieser Staatsexamensarbeit ein genderübergreifend motivierendes Modul für das Schülerlabor Informatik zu entwickeln.

Im Rahmen der durchgeführten, qualitativen Evaluation mit acht SuS kann die erste aufgestellte These als zutreffend angesehen werden: Durch den gezielten Einsatz von Smartphones als „Alltagsgegenstand“ können SuS in der Informatik genderübergreifend motiviert werden.

Auch der Einsatz der NXTs und der Arduinos gefiel den SuS sehr gut. Dabei waren jedoch keine zusätzlichen Auswirkungen der Smartphones auf das Interesse an verstärkt technischen Themen feststellbar, sodass die zweite aufgestellte These nicht greift. Das Interesse am Mikrocontroller war stattdessen vom Vorwissen der einzelnen SuS abhängig.

Die Daten der Evaluation sind aufgrund der sehr kleinen Testgruppe und der qualitative Erhebung nicht objektiv vergleichbar und möglicherweise nicht repräsentativ. Für weiterführende Aussagen müsste eine umfassendere, quantitative Evaluation durchgeführt werden.

6. Ausblick

Mit dieser Staatsexamensarbeit wurde ein vollständiges Modul für das Schülerlabor Informatik erstellt, das in der vorliegenden Form einsatzbereit ist. Durch Äußerungen in den Interviews und weitere, eigene Ideen ergeben sich jedoch noch einige zusätzliche Möglichkeiten, wie das Modul in Zukunft umgestaltet oder erweitert werden könnte.

Von den SuS wurde in den Interviews vorgeschlagen die Lerneinheit freier zu gestalten. Dies ist zwar möglich, jedoch muss dann wesentlich mehr Zeit für die Bearbeitung des Themas eingeplant werden. Es ist zudem sinnvoll einen geleiteten Einstieg in das Themengebiet anzubieten, um auch leistungsschwächere SuS zu unterstützen (siehe Abschnitt 3.3.2) (Krapp/Weidenmann, 2006, S. 634-636).

Mit geringen Veränderungen könnte die bestehende Lerneinheit jedoch als geleiteter Einstieg genutzt werden, um anschließend Schritt für Schritt in eine offenere und situierte Lernumgebung überzugehen. Nachdem mit der Lerneinheit eine gemeinsame Grundlage in Form einer funktionierenden Android App geschaffen wurde, könnten anschließend Dokumentationen erläutert und zu Verfügung gestellt werden. Mit diesen könnten die SuS ihre App dann gemäß ihren eigenen Vorstellungen erweitern. Die Gestaltung eines solchen Moduls ent-

spräche dann in etwa dem Cognitive-Apprenticeship-Ansatz (Krapp/ Weidenmann, 2006, S. 631-632).

Desweiteren wäre es denkbar eine Bauphase vor der Bearbeitung des Lerneinheit durchzuführen, in der die SuS ihren NXT oder fahrbaren Arduino komplett selbst zusammenbauen. Im weiteren Verlauf könnte dann auch die Möglichkeit bestehen den NXT oder den Arduino individuell um weitere Elemente zu erweitern.

Würde das Modul um all diese Möglichkeiten erweitert werden, würde der Zeitaufwand für die Vorbereitung, Durchführung und Betreuung jedoch überproportional steigen. Mit den Inhalten ließen sich dann schätzungsweise mindestens zwei Tage füllen.

Eine ganz andere Möglichkeit bestünde darin die Inhalte der Lerneinheit für niedrigere Jahrgangsstufen herunter zu brechen. Dazu könnte der App Inventor von Google eingesetzt werden, der es ermöglicht erste Android Apps ohne spezielle Programmierkenntnisse per Drag and Drop zu erstellen. Kurz nach der Fertigstellung des Moduls wurden im App Inventor Bausteine für die Bluetooth-Kommunikation mit Lego Mindstorms NXT in einer Beta-Version freigeschaltet. Wenn diese Software ausgereift genug ist, um verlässlich eingesetzt zu werden, könnten die Inhalte der Lerneinheit damit für jüngere SuS aufbereitet werden.

Neben den beschriebenen Ideen sind Dank der vielfältigen Einsatzmöglichkeiten von Android Smartphones, Lego Mindstorms NXT, Arduino Mikrocontrollern und Bluetooth noch zahllose weitere Szenarien denkbar.

7. Quellenverzeichnis

Medienpädagogischer Forschungsverbund Südwest: JIM 2008: Jugend, Information, (Mulit-)Media. Basisstudie zum Medienumgang 12- bis 19-Jähriger in Deutschland. ⟨URL: `http://www.mpfs.de/index.php?id=117`⟩ – Zugriff am 17.03.2011

Medienpädagogischer Forschungsverbund Südwest: JIM 2010: Jugend, Information, (Mulit-)Media. Basisstudie zum Medienumgang 12- bis 19-Jähriger in Deutschland. ⟨URL: `http://www.mpfs.de/index.php?id=181`⟩ – Zugriff am 17.03.2011

Ministerium für Schule und Weiterbildung, Wissenschaft und Forschung des Landes Nordrhein-Westfalen: Richtlinien und Lehrpläne für die Sekundarstufe II – Gymnasium/-Gesamtschule in Nordrhein-Westfalen. Informatik. 1. Auflage. 1999

The Lego Group: Lego Mindstorms NXT Bluetooth Developer Kit. 2006 ⟨URL: `http://mindstorms.lego.com/en-us/support/files/default.aspx`⟩ – Zugriff am 17.03.2011

Deinmann, Markus: Motivationale Bedingungen beim Lernen mit Neuen Medien. Medienunterstütztes Lernen – Beträge von der

WissPro-Wintertagung 2002

Destatis: Studienanfänger: Deutschland, Semester, Nationalität, Geschlecht, Studienfach. 2011 ⟨URL: https://www-genesis.destatis.de/genesis/online;jsessionid=72EBF94A713E8AE3C66434DC8F1D4031.tomcat_GO_2_1?levelindex=3\&levelid=1299594215242\&downloadname=21311-0012\&operation=ergebnistabelleDiagramm\&option=diagramm⟩ – Zugriff am 08.03.2011

Humbert, Ludger/Panske, Janin: Informatik und Gender – nehmt die Forschungsergebnisse ernst! Frauenarbeit und Informatik 34 2010

Kirchgraber, U. et al.: Manual zur Entwicklung von Leitprogrammen. 1999, ETH Zürich

Krapp, Andreas/Weidenmann, Bernd (Hrsg.): Pädagogische Psychologie. 5. Auflage. Weinheim, Basel: Beltz Verlag, 2006

Kuckart, Udo et al.: Qualitative Evaluation. Der Einstieg in die Praxis. 2. Auflage. Wiesbaden: VS Verlag für Sozialwissenschaften, 2008

Petersen, Ulrike et al.: Roberta Abschlussbericht. 2007

SLI: Schülerlabor Informatik der RWTH Aachen. 2011 ⟨URL: http://schuelerlabor.informatik.rwth-aachen.de/⟩ – Zugriff am 24.03.2011

Statista: Anzahl der abgesetzten Smartphone-Betriebssysteme weltweit von 2008 bis 2010. 2011 ⟨URL: http://de.

statista.com/statistik/daten/studie/12856/umfrage/absatz-von-smartphones-weltweit-von-2007-bis-2009/⟩ – Zugriff am 07.03.2011

Statista: Umsatz mit Smartphones in Deutschland in den Jahren 2008 bis 2011. 2011 ⟨URL: http://de.statista.com/statistik/daten/studie/77634/umfrage/umsatz-durch-verkaufte-smartphones-in-deutschland-\seit-2008/⟩ – Zugriff am 07.03.2011

Witzel, Andreas: Das problemzentrierte Interview. 2000 ⟨URL: http://www.qualitative-research.net/index.php/fqs/article/viewArticle/1132/2519⟩ – Zugriff am 16.03.2011

A. Anhang

A.1. Lerneinheiten

A.1.1. Vorbereitungen zum Einsatz

Diese Stichpunktliste dient als Checkliste vor dem Einsatz des Moduls. Alle aufgeführten Punkte müssen im Vorfeld erledigt werden, damit die SuS das Modul problemlos bearbeiten können. Zur Sicherheit werden bei der Software jeweils die Versionen angegeben, mit denen die Durchführung bereits problemlos ablief.

Vorbereitungen:

- Eclipse Classic 3.6.1 installieren
- Eclipse Icon auf dem Desktop platzieren
- Eclipse Workspace festlegen
- JDK 6u23 installieren
- Android SDK r08 installieren und benötigte Plattformen herunterladen (getestet mit 2.2)
- Eclipse Plugin von folgender Quelle installieren: https://dl-ssl.google.com/android/eclipse/
- Android SDK Pfad in den Eclipse Optionen setzen
- Vollständiges Projekt (NXT Remote oder Arduino Remote) in den Workspace importieren
- Testen ob das Projekt korrekt kompiliert! Wahrscheinlich kommt es zu Fehlern und das Projekt muss repariert werden: Rechtsklick auf das Projekt → Android Tools → Fix Project Properties. An-

schließend Eclipse neu starten

- Einstellungen am Android Smartphone: Einstellungen → Anwendungen → Unbekannte Quellen zulassen
- HTC Sync 3.0 installieren und Laptop anschließend neu starten
- Installieren der App testen: Smartphone per USB anschließen und App aus Eclipse heraus installieren
- App testen
- Java-Quelltext in C:\Programme\ArduinoRemote\Java bzw. C:\Programme\NXTremote\Java ablegen (bei NXTremote.java ohne vervollständigte armButton-Funktion)
- Vollständige App aus Eclipse entfernen und Skeleton App in den Workspace importieren und auf dem Smartphone installieren und Icon auf dem Homescreen ablegen
- Pairing von Smartphone und NXT bzw. Arduino durchführen
- Bei der Arduino Remote: Arduino IDE installieren und Icon auf dem Desktop platzieren

A.1.2. Materialien

Die Lerneinheiten werden aufgrund der interaktiven Elemente und ihres Umfangs an dieser Stelle nicht abgedruckt. Die vollständigen Lerneinheiten können auf der beiliegenden CD-ROM und online unter http://elearn.rwth-aachen.de/Holz gefunden werden.

A.2. Quelltexte der Android Apps

Die Eclipse-Projekte und die kompilierten Versionen der beiden Android Apps können auf der beiliegenden CD-ROM gefunden werden.

A.2.1. NXT Remote

In diesem Abschnitt ist der vollständige Java-Quelltext der Android App NXT Remote zu finden.

NXTremote.java:

```
1  package de.rwth.elearn.NXTremote;
2
3  import java.io.IOException;
4  import java.util.Set;
5
6  import android.app.Activity;
7  import android.bluetooth.BluetoothAdapter;
8  import android.bluetooth.BluetoothDevice;
9  import android.content.Intent;
10 import android.hardware.SensorManager;
11 import android.os.Bundle;
12 import android.view.View;
13 import android.widget.ArrayAdapter;
14 import android.widget.Spinner;
15 import android.widget.Toast;
16 import android.widget.ToggleButton;
17
```

```
18 public class NXTremote extends Activity {
19    public static ToggleButton startButton;
20    public static ToggleButton armButton;
21    public static String selectedName;
22    public static boolean connectionEstablished;
23    public static ConnectBluetooth connection;
24    public static SensorManager mySensorManager;
25
26    private static BluetoothAdapter
         myBluetoothAdapter;
27    private static Set<BluetoothDevice>
         pairedDevices;
28    private static BluetoothDevice
         selectedDevice;
29    private static CheckSensor orientationSensor
         ;
30
31    /* executed when NXT Remote is started */
32     @Override
33     public void onCreate(Bundle
          savedInstanceState) {
34         super.onCreate(savedInstanceState);
35
36         /* get default bluetooth adapter */
37         myBluetoothAdapter = BluetoothAdapter.
            getDefaultAdapter();
38         if (myBluetoothAdapter == null) {
```

```
39          Toast.makeText(getApplicationContext()
               , "NXT␣Remote␣funktioniert␣ohne␣
               Bluetooth␣leider␣nicht.", Toast.
               LENGTH_LONG).show();
40        }
41
42        /* ask to enable bluetooth */
43        if (!myBluetoothAdapter.isEnabled()) {
44          Intent enableBtIntent = new Intent(
               BluetoothAdapter.
               ACTION_REQUEST_ENABLE);
45          int requestBt = 0;
46          startActivityForResult(enableBtIntent,
                requestBt);
47      }
48
49        /* access the sensors of our smartphone
               */
50        mySensorManager = (SensorManager)
             getSystemService(SENSOR_SERVICE);
51
52        /* build layout */
53        setContentView(R.layout.main);
54    }
55
56    @Override
57    public void onResume() {
```

```
58        super.onResume();
59        /* identify our buttons */
60        startButton = (ToggleButton) findViewById
             (R.id.startButton);
61        armButton = (ToggleButton) findViewById(R
             .id.armButton);
62
63        /* query paired devices (no discovery to
             prevent hijacking of devices) */
64          pairedDevices = myBluetoothAdapter.
               getBondedDevices();
65          ArrayAdapter<String> pairedDevicesArray
                = new ArrayAdapter<String>(this,
               android.R.layout.simple_spinner_item)
               ;
66          pairedDevicesArray.
               setDropDownViewResource(android.R.
               layout.simple_spinner_dropdown_item);
67          if (pairedDevices.size() > 0) {
68           for (BluetoothDevice device :
                pairedDevices) {
69             /* check whether bluetooth is now
                  really enabled */
70             /* NXT is set to device class toy
                  robot (2052), so only add these
                  to the array */
71             if ((myBluetoothAdapter.isEnabled()
```

```
                ) && (device.getBluetoothClass().
                getDeviceClass() == 2052))
72              pairedDevicesArray.add(device.
                    getName());
73          }
74        }
75
76        /* fill spinner with items and start
              listening for a selection */
77        Spinner pairedDevicesSpinner = (Spinner
              ) findViewById(R.id.nxtSpinner);
78        pairedDevicesSpinner.setAdapter(
              pairedDevicesArray);
79        pairedDevicesSpinner.
              setOnItemSelectedListener(new
              SelectDevice());
80
81        /* uncheck our buttons. there cannot be
               any connection at this point */
82        startButton.setChecked(false);
83      armButton.setChecked(false);
84    }
85
86    /* stop NXT and cancel bluetooth connection
          before exiting */
87    @Override
88    public void onStop() {
```

```
89        if (connectionEstablished) {
90            connection.steerNXT((byte) 0xFF, (byte
                ) 0);
91            connection.cancel();
92            orientationSensor.cancel();
93        }
94        super.onStop();
95    }
96
97    /* start and stop the remote control */
98    public void startButtonClick(View v) {
99        if (startButton.isChecked()) {
100           /* get selected device by name */
101           for (BluetoothDevice device :
                pairedDevices) {
102               if (device.getName().equals(
                    selectedName)) selectedDevice =
                    device;
103           }
104
105           /* establish connection */
106           if (selectedDevice != null) {
107               connectionEstablished = false;
108               connection = new ConnectBluetooth(
                    selectedDevice);
109                   connection.start();
110                   /* check every 80ms whether
```

```
                    connection is now established.
                     abort after 8s */
111                 int i = 0;
112                 while(!connectionEstablished) {
113                     try{
114                         Thread.sleep(80);
115                     } catch(InterruptedException
                            sleepException){
116                         Toast.makeText(
                                getApplicationContext(),
                                 "Exception␣(line␣114):␣
                                startButtonClick,␣Thread
                                .sleep(80)", Toast.
                                LENGTH_LONG);
117                     }
118                     i++;
119                     if (i==100) break;
120                 }
121                 /* notify user about connection
                    status */
122                 if (connectionEstablished) {
123                     Toast.makeText(
                            getApplicationContext(), "
                            Verbindung␣hergestellt!",
                            Toast.LENGTH_LONG).show();
124                     orientationSensor = new
                            CheckSensor();
```

```
125                     orientationSensor.start();
126                 } else {
127                     connection.cancel();
128                     Toast.makeText(
                            getApplicationContext(), "
                            Verbindung␣nicht␣
                            hergestellt!", Toast.
                            LENGTH_LONG).show();
129                 }
130             }
131         } else {
132             /* turn all motors off, cancel
                   connection and uncheck armButton */
133             if (connectionEstablished) {
134                 connection.steerNXT((byte) 0xFF, (
                       byte) 0);
135                 connection.cancel();
136                 orientationSensor.cancel();
137                 armButton.setChecked(false);
138             }
139         }
140     }
141
142     /* turn robot arm on and off (port A) */
143     public void armButtonClick(View v) throws
           IOException {
144         if (connectionEstablished) {
```

```
145             if (armButton.isChecked()) {
146                 connection.steerNXT((byte) 0, (byte
                        ) 80);
147             } else {
148                 connection.steerNXT((byte) 0, (byte
                        ) 0);
149             }
150         } else {
151             Toast.makeText(getApplicationContext()
                    , "Zuerst Fernbedienung starten!",
                    Toast.LENGTH_LONG).show();
152             armButton.setChecked(false);
153         }
154     }
155 }
```

CheckSensor.java:

```
1 package de.rwth.elearn.NXTremote;
2
3 import android.hardware.Sensor;
4 import android.hardware.SensorEvent;
5 import android.hardware.SensorEventListener;
6 import android.hardware.SensorManager;
7
8 public class CheckSensor extends Thread
    implements SensorEventListener {
9     private float [] sensorValues;
```

```
10      private float [] powerPort = new float [2];
11      private int min = 60;
12      private int max = 90;
13      private int basis = 50;
14      private int yFactor = 5;
15
16
17      /* start listening to the orientation sensor
            */
18      public void run() {
19          NXTremote.mySensorManager.
                registerListener(this, NXTremote.
                mySensorManager.getDefaultSensor(
                Sensor.TYPE_ORIENTATION),
                SensorManager.SENSOR_DELAY_GAME);
20      }
21
22      /* stop listening to the orientation sensor
            */
23      public void cancel() {
24          NXTremote.mySensorManager.
                unregisterListener(this);
25      }
26
27      /* check the current state of the
            orientation sensor and use it to calculate
             to motor power */
```

```
28      public void onSensorChanged(SensorEvent
           event) {
29         /* calculate the new motor power */
30         if (NXTremote.connectionEstablished) {
31            /* get orientation values from the
                 sensor */
32            sensorValues = event.values;
33
34            /**
35             * powerPort[0] is for port B on the
                 NXT
36             * powerPort[1] is for port C
37             *
38             * sensorValues[1] is the rotation
                 around the X axis
39             * (which goes left to right across
                 the display) with values from -180
                 to 180
40             * sensorValues[2] is the rotation
                 around the Y axis
41             * (which goes bottom to top across
                 the display) with values from -90
                 to 90
42             * with this the control for driving
                 turns would be very sensitive
43             * so sensorValues[2] is divided by
                 yFactor
```

```
44          *
45          * furthermore it is distinguished
              between driving forwards and
              driving backwards
46          * and basis is added or subtracted as
               additional power to gain some
              speed
47          */
48         for (int i=0; i<=1; i++) {
49            /* moving forward */
50            if (sensorValues[1] >= 0) {
51               /* calculate basic speed */
52               powerPort[i] = sensorValues[1] +
                   basis;
53               /* deal with speed values above
                  max */
54               if (powerPort[i] > max)
                  powerPort[i] = max;
55               /* add/subtract value for
                  driving left or right */
56               if (i==0) powerPort[i] +=
                  sensorValues[2]/yFactor;
57               if (i==1) powerPort[i] -=
                  sensorValues[2]/yFactor;
58            /* moving backwards ... similar to
                moving forward */
59            } else {
```

```
60                  powerPort[i] = sensorValues[1] -
                        basis;
61                  if (powerPort[i] < -max)
                       powerPort[i] = -max;
62                  if (i==0) powerPort[i] -=
                       sensorValues[2]/yFactor;
63                  if (i==1) powerPort[i] +=
                       sensorValues[2]/yFactor;
64              }
65
66              /* motors are giving a buzzing
                   sound between 0 and min so just
                   stay 0 below min */
67              if ((powerPort[i] < min) && (
                   powerPort[i] > -min)) powerPort[i
                   ] = 0;
68          }
69
70          /* send instructions to the NXT */
71          NXTremote.connection.steerNXT((byte)
               1, (byte) powerPort[0]);
72          NXTremote.connection.steerNXT((byte)
               2, (byte) powerPort[1]);
73      }
74  }
75
76   public void onAccuracyChanged(Sensor sensor
```

```
        , int accuracy) {
77      /* abstract method that must be
           implemented. but we don't care for
           accuracy */
78    }
79 }
```

ConnectBluetooth.java:

```
1  package de.rwth.elearn.NXTremote;
2
3  import java.io.DataOutput;
4  import java.io.DataOutputStream;
5  import java.io.IOException;
6  import java.nio.ByteBuffer;
7  import java.nio.ByteOrder;
8  import java.util.UUID;
9
10 import android.bluetooth.BluetoothDevice;
11 import android.bluetooth.BluetoothSocket;
12
13 public class ConnectBluetooth extends Thread {
14    private BluetoothSocket mySocket;
15    private ByteBuffer commandBuffer;
16    private UUID myUUID;
17    private DataOutput myOutputStream;
18
19    /* create bluetooth socket with given UUID
```

```
     */
20   public ConnectBluetooth(BluetoothDevice
        device) {
21       myUUID = UUID.fromString("
            00001101-0000-1000-8000-00805F9B34FB"
            );
22       try {
23           mySocket = device.
                createRfcommSocketToServiceRecord
                (myUUID);
24       } catch (IOException socketException) {
             }
25   }
26
27   /* try to establish a connection through
        the socket and get OutputStream */
28   public void run() {
29       try {
30        mySocket.connect();
31        NXTremote.connectionEstablished = true
             ;
32        createCommandBuffer();
33       } catch (IOException connectException)
            { }
34
35       try {
36        myOutputStream = new DataOutputStream(
```

```
            mySocket.getOutputStream());
        } catch (IOException streamException) {
            }
    }

    /* close the socket and end the connection */
    public void cancel() {
        try {
            mySocket.close();
            NXTremote.connectionEstablished = false;
            NXTremote.startButton.setChecked(false);
         NXTremote.armButton.setChecked(false);
        } catch (IOException closeException) {
           }
    }

    /* create the default byte sequence to send to the NXT */
   private void createCommandBuffer() {
      // maximum 64 bytes
        commandBuffer = ByteBuffer.allocate(14);
      // has to be little endian by definition
        commandBuffer.order(ByteOrder.
```

```
            LITTLE_ENDIAN);
        // announce that the message is 12 bytes long
        commandBuffer.putShort((short) 12);
        // no respond wanted
        commandBuffer.put((byte) 0x80);
        // set NXT to direct bluetooth communication
        commandBuffer.put((byte) 0x04);
        // port: 0, 1, 2, 0xFF (all)
        commandBuffer.put((byte) 0);
        // power: between -100 and 100
        commandBuffer.put((byte) 0);
        // mode: turn specified motor on
        commandBuffer.put((byte) 0x01);
        // regulation: enable motor synchronization
        commandBuffer.put((byte) 0x02);
        // turn ratio: not needed
        commandBuffer.put((byte) 0);
        // run state: running
        commandBuffer.put((byte) 0x20);
        // duration: 0 means forever
        commandBuffer.putInt((int) 0);
    }

    /* send instructions to the NXT */
```

```
79      public void steerNXT(byte port, byte power)
           {
80         commandBuffer.put(4, (byte) port);
81         commandBuffer.put(5, (byte) power);
82         try {
83            myOutputStream.write(commandBuffer.
                 array());
84         } catch (IOException writeException) {
85            cancel();
86         }
87      }
88   }
```

SelectDevice.java:

```
1  package de.rwth.elearn.NXTremote;
2
3  import android.view.View;
4  import android.widget.AdapterView;
5  import android.widget.AdapterView.
      OnItemSelectedListener;
6
7  public class SelectDevice implements
      OnItemSelectedListener {
8
9     /* return the name of the selected bluetooth
          device */
10    public void onItemSelected(AdapterView<?>
```

```
        parent, View view, int pos, long id) {
11      NXTremote.selectedName = parent.
            getItemAtPosition(pos).toString();
12    }
13
14    public void onNothingSelected(AdapterView<?>
          parent) {
15      /* abstract method, that must be
            implemented. but this cannot happen
            with the spinner */
16    }
17  }
```

A.2.2. Arduino Remote

In diesem Abschnitt ist der vollständige Java-Quelltext der Android App Arduino Remote zu finden.

ArduinoRemote.java:

```
1  package de.rwth.elearn.ArduinoRemote;
2
3  import java.util.Set;
4
5  import android.app.Activity;
6  import android.bluetooth.BluetoothAdapter;
7  import android.bluetooth.BluetoothDevice;
8  import android.content.Intent;
```

```
import android.hardware.SensorManager;
import android.os.Bundle;
import android.view.View;
import android.widget.ArrayAdapter;
import android.widget.Spinner;
import android.widget.Toast;
import android.widget.ToggleButton;

public class ArduinoRemote extends Activity {
   public static ToggleButton startButton;
   public static String selectedName;
   public static boolean connectionEstablished;
   public static ConnectBluetooth connection;
   public static SensorManager mySensorManager;

   private static BluetoothAdapter
      myBluetoothAdapter;
   private static Set<BluetoothDevice>
      pairedDevices;
   private static BluetoothDevice
      selectedDevice;
   private static CheckSensor orientationSensor
      ;

   /* executed when Arduino Remote is started
      */
    @Override
```

```
31      public void onCreate(Bundle
           savedInstanceState) {
32          super.onCreate(savedInstanceState);
33
34          /* get default bluetooth adapter */
35          myBluetoothAdapter = BluetoothAdapter.
               getDefaultAdapter();
36          if (myBluetoothAdapter == null) {
37           Toast.makeText(getApplicationContext()
                , "Arduino␣Remote␣funktioniert␣ohne␣
                Bluetooth␣leider␣nicht.", Toast.
                LENGTH_LONG).show();
38          }
39
40          /* ask to enable bluetooth */
41          if (!myBluetoothAdapter.isEnabled()) {
42           Intent enableBtIntent = new Intent(
                BluetoothAdapter.
                ACTION_REQUEST_ENABLE);
43           int requestBt = 0;
44           startActivityForResult(enableBtIntent,
                 requestBt);
45        }
46
47          /* access the sensors of our smartphone
                */
48          mySensorManager = (SensorManager)
```

```
        getSystemService(SENSOR_SERVICE);

        /* build layout */
        setContentView(R.layout.main);
    }

    @Override
    public void onResume() {
      super.onResume();
      /* identify our buttons */
      startButton = (ToggleButton) findViewById
         (R.id.startButton);

      /* query paired devices (no discovery to
         prevent hijacking of devices) */
        pairedDevices = myBluetoothAdapter.
           getBondedDevices();
        ArrayAdapter<String> pairedDevicesArray
            = new ArrayAdapter<String>(this,
           android.R.layout.simple_spinner_item)
           ;
      pairedDevicesArray.
         setDropDownViewResource(android.R.
         layout.simple_spinner_dropdown_item);
      if (pairedDevices.size() > 0) {
         for (BluetoothDevice device :
            pairedDevices) {
```

```
66                /* check whether bluetooth is now
                     really enabled */
67                /* Arduino is set to device class
                     uncategorized (7936), so only add
                      these to the array */
68                if ((myBluetoothAdapter.isEnabled()
                     ) && (device.getBluetoothClass().
                     getDeviceClass() == 7936))
69                     pairedDevicesArray.add(device.
                          getName());
70            }
71        }
72
73        /* fill spinner with items and start
             listening for a selection */
74        Spinner pairedDevicesSpinner = (Spinner
             ) findViewById(R.id.btSpinner);
75        pairedDevicesSpinner.setAdapter(
             pairedDevicesArray);
76        pairedDevicesSpinner.
             setOnItemSelectedListener(new
             SelectDevice());
77
78        /* uncheck our buttons. there cannot be
              any connection at this point */
79        startButton.setChecked(false);
80    }
```

```
81
82      /* stop Arduino and cancel bluetooth
           connection before exiting */
83      @Override
84      public void onStop() {
85        if (connectionEstablished) {
86          connection.steerArduino((byte) 0);
87          connection.cancel();
88          orientationSensor.cancel();
89        }
90        super.onStop();
91      }
92
93      /* start and stop the remote control */
94      public void startButtonClick(View v) {
95        if (startButton.isChecked()) {
96          /* get selected device by name */
97          for (BluetoothDevice device :
               pairedDevices) {
98            if (device.getName().equals(
                 selectedName)) selectedDevice =
                 device;
99          }
100
101         /* establish connection */
102         if (selectedDevice != null) {
103           connectionEstablished = false;
```

```
104                 connection = new ConnectBluetooth(
                       selectedDevice);
105                    connection.start();
106                    /* check every 80ms whether
                          connection is now established.
                           abort after 8s */
107                    int i = 0;
108                    while(!connectionEstablished) {
109                       try{
110                          Thread.sleep(80);
111                       } catch(InterruptedException
                             sleepException){
112                          Toast.makeText(
                                getApplicationContext(),
                                 "Exception␣(line␣114):␣
                                startButtonClick,␣Thread
                                .sleep(80)", Toast.
                                LENGTH_LONG);
113                       }
114                       i++;
115                       if (i==200) break;
116                    }
117                    /* notify user about connection
                          status */
118                    if (connectionEstablished) {
119                       Toast.makeText(
                             getApplicationContext(), "
```

```
                    Verbindung hergestellt!",
                    Toast.LENGTH_LONG).show();
120                 orientationSensor = new
                    CheckSensor();
121                 orientationSensor.start();
122             } else {
123                 connection.cancel();
124                 Toast.makeText(
                    getApplicationContext(), "
                    Verbindung nicht
                    hergestellt!", Toast.
                    LENGTH_LONG).show();
125             }
126         }
127     } else {
128         /* turn all motors off and cancel
            connection */
129         if (connectionEstablished) {
130             connection.steerArduino((byte) 0);
131             connection.cancel();
132             orientationSensor.cancel();
133         }
134     }
135 }
136 }
```

CheckSensor.java:

```
package de.rwth.elearn.ArduinoRemote;

import android.hardware.Sensor;
import android.hardware.SensorEvent;
import android.hardware.SensorEventListener;
import android.hardware.SensorManager;

public class CheckSensor extends Thread
   implements SensorEventListener {
    private float [] sensorValues;
    private float powerA;
    private float powerB;
    private int basis = 40;
    private int max = 80;
    private int yFactor = 2;
    private boolean forwards = true;

    /* start listening to the orientation sensor
        */
    public void run() {
         ArduinoRemote.mySensorManager.
            registerListener(this, ArduinoRemote.
            mySensorManager.getDefaultSensor(
            Sensor.TYPE_ORIENTATION),
            SensorManager.SENSOR_DELAY_GAME);
    }
```

```
21
22      /* stop listening to the orientation sensor
           */
23      public void cancel() {
24          ArduinoRemote.mySensorManager.
               unregisterListener(this);
25      }
26
27      /* check the current state of the
           orientation sensor and use it to calculate
            to motor power */
28      public void onSensorChanged(SensorEvent
           event) {
29          /* get orientation values from the sensor
               */
30          sensorValues = event.values;
31
32          /* calculate the new motor power */
33          if (ArduinoRemote.connectionEstablished)
               {
34              /**
35               * sensorValues[1] is the rotation
                   around the X axis
36               * (which goes left to right across
                   the display) with values from -180
                   to 180
37               * sensorValues[2] is the rotation
```

```
around the Y axis
38          * (which goes bottom to top across
              the display) with values from -90
              to 90
39          * with this the control for driving
              turns would be very sensitive
40          * so sensorValues[2] is divided by
              yFactor
41          *
42          * furthermore it is distinguished
              between driving forwards and
              driving backwards
43          * and basis is added or subtracted as
               additional power to gain some
              speed
44          */
45         /* check whether the driving direction
              was changed and inform Arduino if
             necessary */
46         if (sensorValues[1] >= 0 && !forwards)
              {
47            ArduinoRemote.connection.
                steerArduino((byte) 251);
48            forwards = true;
49         }
50         if (sensorValues[1] < 0 && forwards) {
51            ArduinoRemote.connection.
```

```
                steerArduino((byte) 252);
            forwards = false;
        }

        /* calculate speed */
        if (sensorValues[1] < 0) sensorValues[1] *= -1;
        powerA = sensorValues[1] + basis;
        powerB = sensorValues[1] + basis;
        if (powerA > max) powerA = max;
        if (powerB > max) powerB = max;
        powerA = powerA - (sensorValues[2]/yFactor);
        powerB = powerB + (sensorValues[2]/yFactor) + 125;

        /* deal with speed values above max */
        if (powerA > 125) powerA = 125;
        if (powerB > 250) powerB = 250;

        /* send instructions to the NXT */
        ArduinoRemote.connection.steerArduino((byte) powerA);
        ArduinoRemote.connection.steerArduino((byte) powerB);
    }
}
```

```
73
74      public void onAccuracyChanged(Sensor sensor
            , int accuracy) {
75        /* abstract method that must be
              implemented. but we don't care for
              accuracy */
76      }
77  }
```

ConnectBluetooth.java:

```
1   package de.rwth.elearn.ArduinoRemote;
2
3   import java.io.IOException;
4   import java.io.OutputStream;
5   import java.util.UUID;
6
7   import android.bluetooth.BluetoothDevice;
8   import android.bluetooth.BluetoothSocket;
9
10  public class ConnectBluetooth extends Thread {
11     private BluetoothSocket mySocket;
12     private OutputStream myOutputStream;
13     private UUID myUUID;
14
15     /* create bluetooth socket with given UUID
          */
16      public ConnectBluetooth(BluetoothDevice
```

```
        device) {
17          myUUID = UUID.fromString("
               00001101-0000-1000-8000-00805F9B34FB"
               );
18          try {
19              mySocket = device.
                   createRfcommSocketToServiceRecord
                   (myUUID);
20          } catch (IOException socketException) {
               }
21      }
22
23      /* try to establish a connection through
           the socket and get OutputStream */
24      public void run() {
25          try {
26           mySocket.connect();
27           ArduinoRemote.connectionEstablished =
               true;
28          } catch (IOException connectException)
               { }
29
30          try {
31           myOutputStream = mySocket.
               getOutputStream();
32          } catch (IOException streamException) {
               }
```

```
33      }
34
35      /* close the socket and end the connection
           */
36      public void cancel() {
37          try {
38              mySocket.close();
39              ArduinoRemote.connectionEstablished
                    = false;
40              ArduinoRemote.startButton.
                   setChecked(false);
41          } catch (IOException closeException) {
               }
42      }
43
44     /* send instructions to the Arduino */
45     public void steerArduino(byte power){
46        try {
47           myOutputStream.write(power);
48        } catch (IOException writeException) {
49           cancel();
50        }
51     }
52  }
```

SelectDevice.java:

```
package de.rwth.elearn.ArduinoRemote;

import android.view.View;
import android.widget.AdapterView;
import android.widget.AdapterView.
   OnItemSelectedListener;

public class SelectDevice implements
   OnItemSelectedListener {

   /* return the name of the selected bluetooth
       device */
   public void onItemSelected(AdapterView<?>
      parent, View view, int pos, long id) {
      ArduinoRemote.selectedName = parent.
         getItemAtPosition(pos).toString();
   }

   public void onNothingSelected(AdapterView<?>
       parent) {
      /* abstract method, that must be
         implemented. but this cannot happen
         with the spinner */
   }
}
```

A.3. Der fahrbare Arduino Mikrocontroller

A.3.1. Materialliste

An dieser Stelle findet sich eine Liste der Materialien, die zum Bau des fahrbaren Arduinos verwendet wurden. Werkzeuge sind nicht aufgeführt.

Materialien:

- Arduino Duemilanove
- Ardumoto – Motor Driver Shield für DC Motoren
- 2x GM18 30:1 Mini Getriebemotor
- 2x Getriebehalterung
- 2x Pololu 42x19mm Rad
- BlueSMiRF Gold – Bluetooth Modem
- LED
- 470Ω Widerstand
- Bastelglas 16x9cm
- Frei drehbares Hinterrad 24mm
- 4x Schraubklemme (lötbare Leiterplatten-Anschlussklemmen)
- Batteriehalterung für 9V-Block mit Adapter für die Stromversorgung des Arduinos
- Litze

- Schrumpfschlauch
- Passende Schrauben
- 9V-Block

A.3.2. Bau

Der Bau des fahrbaren Arduino Mikrocontrollers wird an dieser Stelle zusammenfassend geschildert. Einzelne Arbeitsschritte werden nicht im Detail erläutert.

An die Grundplatte aus Bastelglas werden zunächst das Hinterrad und die Batteriehalterung geschraubt. Anschließend wird auch der Arduino Mikrocontroller an der Grundplatte befestigt. Die Steckerleisten und mitgelieferten Buchsenleisten werden an das Ardumoto Shield gelötet. Die Steckerleisten dienen später dem Anschluss der Motoren, während über die Buchsenleisten die Verbindung zum Arduino hergestellt wird. Wenn alle Teile angebracht sind, wird das Ardumoto Shield auf den Arduino aufgesteckt. An die Buchsenleisten des Ardumoto Shields kann nun der Bluetooth-Adapter BlueSMiRF angeschlossen werden.

An die Anschlüsse der Motoren wird Litze gelötet, die Lötstelle mit Schrumpfschlauch geschützt und die Motoren anschließend über die Schraubklemmen an das Ardumoto Shield angeschlossen. Mit den Getriebehalterungen werden die Motoren an der Grundplatte festgeschraubt. Anschließend können die Räder aufgesteckt werden.

Der Widerstand wird vor die LED gelötet und die LED gemäß den

Angaben im Sketch angeschlossen. Abschließend wird der Sketch per USB auf den Arduino übertragen. Der fahrbare Arduino ist nun fertig und die Stromversorgung kann angeschlossen werden.

A.3.3. Quelltext

Der Quelltext des Arduino Sketches:

```
byte btValue;
// LED connected to pin 8
int ledpin = 8;
// PWM control for motor outputs 1 and 2 is on
   digital pin 10
int pwm_a = 10;
// PWM control for motor outputs 3 and 4 is on
   digital pin 11
int pwm_b = 11;
// direction control for motor outputs 1 and 2
   is on digital pin 12
int dir_a = 12;
// direction control for motor outputs 3 and 4
   is on digital pin 13
int dir_b = 13;

void setup() {
  // baud rate for Bluetooth
  Serial.begin(57600);

```

```
17    pinMode(ledpin, OUTPUT);
18    pinMode(pwm_a, OUTPUT);
19    pinMode(pwm_b, OUTPUT);
20    pinMode(dir_a, OUTPUT);
21    pinMode(dir_b, OUTPUT);
22
23    // set both motors runnig forwards
24    digitalWrite(dir_a,LOW);
25    digitalWrite(dir_b,HIGH);
26  }
27
28
29  void loop() {
30    if (Serial.available()) {
31      // value goes from 0 to 255
32      btValue = Serial.read();
33      // turn on the LED if Bluetooth data is
          available
34     digitalWrite(ledpin, HIGH);
35    } else {
36      digitalWrite(ledpin, LOW);
37    }
38
39    // get signal for driving forwards
40    if (btValue == 251) {
41      digitalWrite(dir_a,LOW);
42      digitalWrite(dir_b,HIGH);
```

```
43    }
44
45    // get signal for driving backwards
46    if (btValue == 252) {
47      digitalWrite(dir_a,HIGH);
48      digitalWrite(dir_b,LOW);
49    }
50
51    // power motor A (0..125)
52    if (btValue >= 50 && btValue <= 125) {
53      analogWrite(pwm_a, btValue);
54    }
55
56    // power motor B (126..250)
57    if (btValue >= 175 && btValue <= 250) {
58      analogWrite(pwm_b, btValue-125);
59    }
60
61    // otherwise stop the motors
62    if (btValue < 50) {
63      analogWrite(pwm_a, 0);
64    }
65
66    if (btValue > 125 && btValue < 175) {
67      analogWrite(pwm_b, 0);
68    }
69  }
```

A.4. Interviewleitfaden

A.4.1. Vor der Lerneinheit

- Projekt kurz vorstellen
- Warum habt ihr Informatik gewählt? Was fasziniert euch daran?
- Was glaubt ihr, warum so wenig Schülerinnen Informatik wählen?
- Wie schätzt ihr euch selber im Umgang mit Technik ein? (große/kleine Technikdistanz)
- Gender-Artikel kurz vorstellen und Meinungen einholen
- Findet ihr es interessanter mit Handys/Smartphones zu arbeiten als nur mit dem PC?
- Interessiert ihr euch spontan eher für den NXT oder für den Arduino? Warum?

A.4.2. Nach der Lerneinheit

- Zuerst frei Feedback geben lassen!
- Welches Gerät hat euch am meisten interessiert? Warum? Welches am wenigsten? (Smartphone, NXT, Arduino, Laptop?)
- Wie wäre es mit einem Tablet statt einem Smartphone?
- Wie schätzt ihr euch selbst im Umgang mit Technik ein? (Unterschied zu zuvor?)
- Hat euch die Teamarbeit gefallen? Hättet ihr lieber größere

Teams oder Einzelarbeit?

- Hätte es euch mehr Spass gemacht wenn die Lerneinheit anders gestaltet gewesen wäre? (z.B.: Bauphase am Anfang, Gruppenpuzzle am Ende, länger/kürzer?)
- Hättet ihr lieber etwas anderes anstatt ferngesteuerte Autos gehabt? (z.B.: Käfer/Krabbeltiere)
- Fehlt eurer Meinung nach im Interview noch etwas?
- Wie waren die Interviews für euch?
- Anbieten die Ergebnisse zu mailen

B. Erklärung

Ich versichere, dass ich die schriftliche Hausarbeit – einschließlich beigefügter Zeichnungen, Kartenskizzen und Darstellungen – selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Alle Stellen der Arbeit, die dem Wortlaut oder dem Sinne nach anderen Werken entnommen sind, habe ich in jedem Fall unter Angabe der Quelle deutlich als Entlehnung kenntlich gemacht.

Ort, Datum	Jan Tobias Holz