

Schülerlabor-Modul zur Programmierung von Tinkerforge-Mikrocontrollern mit Java im Kontext Hausautomation

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

Fabian Dominik Blasius
Matrikelnummer 289549

Aachen, 19.02.2013

Gutachter: Prof. Dr. Ulrik Schroeder, RWTH Aachen

Lehrstuhl: Lehr- und Forschungsgebiet Informatik 9
Learning Technologies Research Group

Betreuer: Dipl.-Gyml. Nadine Bergner
Dipl.-Gyml. Jan Holz



Inhaltsverzeichnis

1. Motivation	6
1.1. Begründung des InfoSphere-Moduls	6
1.2. Didaktische Zielsetzung	7
1.3. Auswahl der Mikrocontroller	8
1.4. Einbettung in den Kontext Schule	9
1.4.1. Einordnung in den Lehrplan	10
1.4.2. Einordnung in die Abiturvorgaben	11
1.4.3. Bezugsetzung zum Schulunterricht	11
2. Technische Entwicklung	13
2.1. Gegenstand der Entwicklung	13
2.2. Ansprüche an die Projekte	14
2.3. Auswahl der Projekte	14
2.4. Gestaltung der einzelnen Hausautomationsprojekte . . .	16
3. Didaktik	19
3.1. Hypothetische Bedingungsanalyse	19
3.2. Sachanalyse	20
3.3. Handlungs- und produktionsorientierter Unterricht . . .	22
3.4. Didaktische Gestaltung	24
3.4.1. Gesamtverlauf der Exkursion	24
3.4.2. Analyse der Leitpräsentation	32
3.4.3. Teilnehmermaterial	38
3.4.4. Betreuermaterial	48
4. Reflexion	53
4.1. Testdurchführungen und Optimierungen	53
4.2. Kritische Reflexion des didaktischen Gesamtkonzeptes .	55
4.3. Fazit	57
Anhänge	58
A. Literaturverzeichnis	58
B. Exkursionspräsentation	60



C. Teilnehmermaterialien	94
C.1. Einstiegsphase	95
C.2. Projektarbeitsphase	123
C.3. Optionalphase	133
D. Betreuermaterialien	136
E. Digitalisat	152



Zusammenfassung

Die vorliegende Staatsexamensarbeit befasst sich mit der Entwicklung einer praktisch orientierten Exkursion für Schülerinnen und Schüler der Oberstufe zum Thema Hausautomation im Rahmen des InfoSphere - Schülerlabor Informatik der RWTH Aachen. Sie soll ihnen über den Schulunterricht hinaus Einblicke in Anwendungsfelder der Informatik geben und sie für das Fach interessieren. Die Teilnehmer können in der projektorientierten Veranstaltung selbstständig Hausautomations-Produkte herstellen und dabei auf entdeckende Weise neues Wissen und neue, fachübergreifende Kompetenzen erwerben. Für die Herstellung der Hausautomationsprodukte programmieren sie Mikrocontroller unter Java.

In dieser Arbeit wird nach einer didaktischen Zielsetzung und Einbettung in den Schulkontext die Entwicklung der Exkursion geschildert, bevor anschließend die didaktische Konzeption erläutert, analysiert und abschließend reflektiert wird.



Danksagung

An dieser Stelle möchte ich allen Personen bedanken, die mich bei der Erstellung dieser Staatsexamensarbeit direkt oder indirekt unterstützt haben. Zuerst möchte ich meinen Dank richten an Bastian Nordmeyer und Olaf Lüke von der Tinkerforge GmbH. Mit ihrem freundlichen Entgegenkommen und ihrer Hilfe haben sie die Realisierung des InfoSphere-Moduls stark gefördert.

Desweiteren gilt mein Dank dem Team vom LuFG Informatik 9 der RWTH Aachen um Prof. Dr. Schroeder, den HiWis und besonders Nadine Bergner und Jan Holz, die mich sehr geduldig beraten und unterstützt haben.

An meinen Bruder Adrian vielen Dank für die handwerkliche Beratung und Hilfe.

Zuletzt, aber dafür am meisten, möchte ich mich bedanken bei meiner Freundin Kadda. Du hast mich nicht nur beim Schreiben dieser Arbeit, sondern mein gesamtes Studium über trotz vieler Entbehrungen mit Ausdauer großartig unterstützt und mir oft den Rücken frei gehalten. Ich kann die Bedeutung kaum in Worte fassen. Danke.

Kapitel 1 Motivation

1.1. Begründung des InfoSphere-Moduls

Der Ursprung der vorliegenden Staatsexamensarbeit liegt in dem Bestreben, für Schülerinnen und Schüler¹ der Informatik in der Oberstufe einen Exkurs² zu entwickeln, der ihnen einen Einblick in weniger bekannte Themenfelder der Informatik bietet und sie für den Facettenreichtum und die Möglichkeiten der Informatik interessiert. Als mittelfristiges Ziel verbirgt sich dahinter, den SuS ein differenzierteres Bild der Informatik zu vermitteln und mehr Studienanfänger und damit Fachkräfte für die Informatik zu gewinnen.

Dieser Exkurs sollte inhaltlich über den Rahmen des gewöhnlichen Schulunterrichts hinausgehen, die SuS dabei aber nicht überfordern, sondern sie durch eine spannende Thematik und exklusive Hardware motivieren. Als Thema der Exkursion wurde die Hausautomation ausgewählt, da es für das Erreichen der Exkursionsziele besonders geeignet erscheint. Näheres hierzu wird im Kapitel 1.2 erläutert. Hausautomation meint hier die digitale Überwachung und Steuerung einer Wohneinrichtung unter der Zielsetzung der Komfort-, Sicherheits- und Energieeffizienzsteigerung. Zu diesem Thema sollen die SuS im Modellformat mehrere typische Elemente eines automatisierten Hauses mit Hilfe von Mikrocontrollern herstellen.

Das InfoSphere Schülerlabor Informatik [3] bot sich aufgrund seiner inhaltlichen Ausrichtung und Zielsetzung als Rahmen für die Gestaltung der Exkursion an, sodass der Exkurs als Schülerlabor-Modul umgesetzt wurde.

¹ Zum Zwecke einer verbesserten Lesbarkeit wird im Folgenden nur noch die Abkürzung SuS verwendet.

² An dieser Stelle und im Folgenden ist der Begriff Exkurs bzw. Exkursion im Sinne eines Workshops zu verstehen.



1.2. Didaktische Zielsetzung

Zunächst stellt sich die Frage nach der didaktischen Zielsetzung des InfoSphere-Moduls. Das Modul soll mehrere Ziele erfüllen. Dazu gehören neben der Motivation der SuS und dem Erwecken ihres Interesses an Informatik auch die Wissens- und Kompetenzvermittlung. Für das Erreichen dieser Ziele wird die Hausautomation als inhaltlicher Aufhänger verwendet. Dieser Themenbereich wird nicht vom obligatorischen Teil des Schulunterrichts (vgl. dazu auch 1.4) abgedeckt und bietet den SuS somit einen neuen Erfahrungsbereich. In den Kontext der für SuS nachvollziehbaren, alltagsnahen und anschaulichen Thematik Hausautomation, die zusätzlich motivierend wirken soll, wird die Mikrocontroller-Programmierung als Werkzeug eingebettet.

Lernziele

Die Wissensvermittlung beinhaltet das Kennenlernen des Themenbereichs Hausautomation, Grundbegrifflichkeiten rund um Mikrocontroller und Wissen über Fähigkeiten und Eigenschaften der Programmiersprache Java.

Die SuS sollen sowohl theoretische als auch praktische Grundlagen im Bereich der Mikrocontroller-Programmierung kennen lernen, wobei der Schwerpunkt dabei deutlich auf den praktischen Aspekten liegt. Die SuS sollen lernen, was Mikrocontroller sind und wie ihre grobe Funktionsweise ist. Gleichzeitig soll das InfoSphere-Modul mehrere wichtige, fachübergreifende Kompetenzen der SuS fördern und ihnen neue Fähigkeiten aneignen. Gefördert werden soll die Teamfähigkeit der SuS und ihre Fähigkeiten zum selbstständigen Arbeiten mit unbekanntem Inhalt. Weiterhin sollen die SuS lernen, mit Software-Dokumentationen umzugehen. Zusätzlich sollen ihre Fertigkeiten im Umgang mit Quelltexten verbessert werden.

Lehrstrategie

Die geschilderten Lernziele sind sehr gut mit der Thematik Hausautomation zu erarbeiten, da die Arbeitsweisen in der Entwicklung mit Mikrocontrollern die im Exkurs zu erlernenden Kompetenzen erfordern. Um einen anschaulichen Einblick in die Hausautomation zu erhalten, sollen die SuS selber kleine Produkte mit Mikrocontrollern entwickeln. Welche Mikrocontroller dafür verwendet werden und warum wird in 1.3 erläutert.

Die Lerninhalte sollen hauptsächlich Funktionsprinzipien und nicht



konkrete Details der Mikrocontroller-Programmierung beinhalten, damit das Gelernte möglichst leicht auf neue Lernsituationen der SuS übertragbar ist. Dementsprechend soll die Mikrocontroller-Programmierung mit der Hochsprache Java erfolgen, mit der die Mikrocontroller auf einem abstrahierten Niveau programmiert werden können. Weiterer Vorteil und gleichzeitig Notwendigkeit der Verwendung von Java ist, dass die SuS diese bereits kennen. Dadurch entfällt nicht nur das aufwändige Erlernen einer domänenspezifischen Programmiersprache, sondern wird gleichzeitig an den Schulunterricht angeknüpft.

Motivation und Interesse der SuS sollen außerdem dadurch gefördert werden, dass die Gestaltung des Exkurses von gewohnten Unterrichtsformen der Schule abweicht und den Inhalten auf diese Weise mit weniger Vorbehalten begegnet wird.

1.3. Auswahl der Mikrocontroller

Da die Mikrocontroller in einer Exkursion für SuS eingesetzt werden, sind an ihre Eignung mehrere Anforderungen gestellt. So müssen sie ohne großen Vorbereitungsaufwand und mit wenig Hintergrundwissen bedienbar sein. Zusätzlich müssen sie robust genug sein, um auch nicht sachgemäßen Umgang durch SuS zu überstehen. Zuletzt müssen sie für das InfoSphere finanzierbar sein.

Unter Berücksichtigung dieser drei Auswahlaspekte bieten sich zwei alternative Produkte für die Verwendung an: Zum einen die bekannte und verbreitete Arduino-Plattform [1], die unter anderem dazu konzipiert wurde, eine leicht bedienbare Lernplattform für den Einstieg in die Programmierung und Mikrocontroller-Programmierung zu sein. Zum anderen die erst seit 2011 erhältliche Mikrocontroller-Plattform Tinkerforge³ [10]. Das Tinkerforge-System ist als modulares Baukastensystem zum einfachen Bauen und Programmieren von Mikrocontroller-Systemen mit diversen Hochsprachen ohne Mikrocontroller-Hintergrundwissen ausgelegt.

Da beide Systeme eine ähnliche Zielsetzung haben, gilt es ihre Unterschiede herauszuarbeiten und auf dieser Basis eine Entscheidung für eines der beiden Systeme zu treffen. Im Vergleich zum Tinkerforge-System ist die Arduino-Plattform näher an der Hardware zu programmieren und erfordert Kenntnisse in einer Arduino-Programmiersprache, die der Programmiersprache C ähnlich ist [1]. Diese müssten die InfoSphere-Modul Teilnehmer⁴ erst erlernen, wohingegen das Tinkerforge-System nativ in Java programmierbar ist

³ Der Begriff Tinkerforge bezieht sich im Folgenden immer auf die Firma Tinkerforge GmbH und ihre Produkte

⁴ Zur Verbesserung der Lesbarkeit wird im Folgenden nur die maskuline Form verwendet.



und die Teilnehmer Java-Kenntnisse als Vorwissen mitbringen. Eine Verwendung von Arduino wäre ungleich zeitaufwendiger, da den SuS deutlich mehr Hintergrundwissen zu vermitteln wäre. Dies hätte zur Konsequenz, dass von den SuS nur kleinere Bauprojekte umgesetzt werden könnten. Außerdem würde damit das eigentliche Thema Hausautomation zugunsten eines Programmier-Schwerpunktes in den Hintergrund treten. Eine Verwendung des Tinkerforge-Systems hingegen unterstützt den Themenschwerpunkt Hausautomation durch eine sehr geringe Einarbeitungszeit. Da hiermit auch eine den Teilnehmern bekannte Programmiersprache und damit auch eine gewohnte Programmierumgebung verwendbar ist, kann sich hierbei deutlich mehr auf die Konzepte und Arbeitsweisen in der Hausautomatisierung und Mikrocontroller-Programmierung konzentriert werden. Nicht zuletzt passt das Tinkerforge-System durch ein objektorientiertes Programmierkonzept, in dem jeder Baustein einem Objekt entspricht, zu der Vorstellungswelt der SuS aus ihrem Informatikunterricht. Es fördert durch die real greifbaren Objekte sogar das Verständnis der SuS für Objektorientierung. Zusätzlich spricht das modulare Baukastensystem von Tinkerforge mit seinen vorgefertigten Steckverbindungen, die ein korrektes Anschließen der Komponenten stark vereinfachen, für eine Auswahl dessen [10].

Insgesamt kann das Tinkerforge-System durch seine besonders leichte Verwendbarkeit mit Java, geringe Einarbeitungszeit und das SuS-gerechte Design als Baukastensystem überzeugen und wird für die Verwendung im InfoSphere-Modul ausgewählt.

1.4. Einbettung in den Kontext Schule

Die Zielgruppe umfasst SuS der Oberstufe. Mit einer Einbettung in den Schulkontext kann dem unterrichtenden Lehrpersonal die Begründung für eine Teilnahme seiner SuS am Schülerlabor-Modul vor der Schulverwaltung erleichtert werden. Das ist auch deshalb wichtig, weil Lehrkräfte ein wichtiger Werbeträger für das InfoSphere sind und als solche gewonnen werden müssen.

Außerdem liefert die Einbettung in den Kontext Schule Erkenntnisse über den zu erwartenden Vorwissensstand der SuS. Diese Vorkenntnisse spielen eine wichtige Rolle in der didaktischen Ausgestaltung der Exkursion, da diese dort berücksichtigt werden müssen. Bedeutsam ist auch ein Beleg für den nachhaltigen Nutzen für den Schulunterricht.

Zuerst wird das InfoSphere-Modul in den Lehrplan eingeordnet, um eine Übersicht über die Passgenauigkeit zu den bindenden Vorgaben des Bildungsministeriums zu erhalten. Anschließend wird herausgearbeitet, wie die Teilnahme am InfoSphere-Modul in den Alltagsunterricht integriert werden könnte.



1.4.1. Einordnung in den Lehrplan

Da die Schulbildung Angelegenheit der Bundesländer ist und sich die Vorgaben und Lehrpläne der einzelnen Länder unterscheiden, kann an dieser Stelle nur eine exemplarische Einordnung in einen Lehrplan vorgenommen werden. Da das InfoSphere Schülerlabor Informatik von der RWTH Aachen in Nordrhein-Westfalen betrieben wird und somit hauptsächlich SuS aus Nordrhein-Westfalen, erfahrungsgemäß meist SuS eines Gymnasiums oder einer Gesamtschule, zur Zielgruppe gehören, wird der Exkurs in den Lehrplan Informatik für die Sekundarstufe II - Gymnasium/Gesamtschule in Nordrhein-Westfalen [7] eingeordnet. Dabei wird den drei Betrachtungswinkeln des Lehrplans - „Fachliche Inhalte“, „Methoden und Formen selbstständigen Arbeitens“ und „Lernen im Kontext der Anwendung“ - gefolgt. Zuerst zu den Fachlichen Inhalten:

Die Teilnehmer der Exkursion werden für die in der Hausautomation auftretenden Probleme Lösungsstrategien modellieren und konstruieren. Dabei werden sie Lösungen nach dem Programmierkonzept der objektorientierten Programmierung in Java realisieren, überprüfen und ihre Problemlösungen optimieren und weiterentwickeln. In dieser Hinsicht entspricht die Exkursion den Vorsehungen des Lehrplans (vgl. auch Abschnitt 1.4.2).

Hinsichtlich des Aspektes „Lernen im Kontext der Anwendung“ ist der Anwendungskontext Hausautomation in den Lehrplanabschnitt „Messen, Steuern, Regeln“ einzuordnen. Das Unterrichtsangebot der Schule wird also auch hier lehrplankonform ergänzt. „Lernen im Kontext der Anwendung“ beinhaltet auch, Systematiken zur Lösung spezifischer Anwendungssituationen kennenzulernen. Da Mikrocontroller systematisch zur Lösung von Hausautomationsproblemen eingesetzt werden, wird auch dieser Aspekt erreicht. Das Erzeugen eines Gesamtbildes der Informatik ausgehend von ihrer Anwendungsvielfalt wird, wie bereits in der Zielsetzung erläutert, ebenfalls erreicht. Auch damit wird das InfoSphere-Modul den Ansprüchen des Lehrplans gerecht.

Bezüglich der „Methoden und Formen selbstständigen Arbeitens“ ist an dieser Stelle nur zu erwähnen, dass diese den Vorgaben des Lehrplans folgend ausgewählt wurden. Welche das konkret sind und die weiteren Details dazu wird in Kapitel 3 näher erläutert. Die Exkursion wird unter dem zentralen Gestaltungsprinzip des selbstständigen Arbeitens und entdeckenden Lernens umgesetzt und folgt damit dem unter diesem Aspekt im Lehrplan festgehaltene Kernanliegen.

Die Exkursion ist produktionsorientiert, denn in ihrem Verlauf stellen die SuS verschiedene Komponenten eines automatisierten Hauses her. Genauer dazu wird im Kapitel 3 erörtert. Damit erfüllt die Exkursion alle Kriterien eines projektorientierten Unterrichts gemäß Lehrplan [7]. Dieses InfoSphere-Modul entspricht dem Lehrplan also in mehrfacher



Hinsicht und bietet sich somit auch als Informatikkurs-Exkursionsziel an.

Weiterhin sind dem Lehrplan die zu erwartenden Vorkenntnisse der teilnehmenden SuS zu entnehmen:

Die für eine erfolgreiche Teilnahme notwendigen Kenntnisse in Objektorientierung sind laut Lehrplan obligatorisch. Die verwendete Programmiersprache Java ist zwar nicht Pflicht für den Informatikunterricht, wird in der Unterrichtspraxis aber an vielen Schulen Nordrhein-Westfalens gelehrt.

Insgesamt ist für das InfoSphere-Modul die vollständige Kompatibilität zum Informatik-Schulunterricht sichergestellt. Somit sind die Modulinhalte als Ergänzungen zum Schulunterricht aufzufassen, die sich für den fachlichen und fachübergreifenden Unterricht über diese Exkursion hinaus verwerten lassen und damit einen nachhaltigen Nutzen bieten.

1.4.2. Einordnung in die Abiturvorgaben

Die Vorgaben für das Zentralabitur sind für Oberstufenlehrkräfte eine starke Richtlinie und dienen mit dementsprechender Wichtigkeit ihrer Orientierung. Daher ist es besonders erstrebenswert, dass das InfoSphere-Modul in Einklang mit den Abiturvorgaben steht. Erster Schwerpunkt dieser Vorgaben ist das objektorientierte Modellieren und Implementieren von kontextbezogenen Anwendungen [9]. In der Exkursion werden die Mikrocontroller mit Java unter dem Paradigma der Objektorientierung programmiert. Da die Mikrocontroller auf Programmierenebene als Objekte repräsentiert werden und gleichzeitig anfassbare Objekte der realen Welt sind, unterstützt die Exkursion das Verständnis von Objektorientierung durch Anschaulichkeit. So ist auch im Hinblick auf die Abiturvorgaben ein deutlicher Gewinn für die Teilnehmer ermöglicht.

1.4.3. Bezugsetzung zum Schulunterricht

In diesem Abschnitt werden Beispiele genannt, wie die Teilnahme am InfoSphere-Labor in den Alltagsunterricht eingebettet werden kann.

Zuerst eine genauere Eingrenzung der Zielgruppe: Unter Berücksichtigung der benötigten Vorkenntnisse, die nach dem Beginn des Informatikunterrichts erst vermittelt werden müssen, ist eine Teilnahme erst für SuS, die mindestens ein Schulhalbjahr Informatik absolviert haben, ratsam. Die folgenden Schilderungen betreffen also hauptsächlich SuS der Stufen 11 und 12 am Gymnasium mit achtjähriger Schulzeit und SuS der Stufen 12 und 13 an Gesamtschulen



und Schulen mit neunjähriger Schulzeit.

Da es schon allein aus finanziellen Gründen nicht üblich ist, dass SuS im Rahmen des Schul-Informatikunterrichts Mikrocontroller programmieren und Hausautomatisierung nicht zur Obligatorik des Informatikunterrichts gehört, ist die Teilnahme an der Exkursion nicht im Zuge einer detaillierten Unterrichtsreihe zu erwarten. Deshalb ist die Exkursion so gestaltet, dass die Teilnehmer ohne spezielle Vorkenntnisse, außer den genannten, erfolgreich an ihr teilnehmen können.

Es ist möglich, dass eine Lehrkraft eine Teilnahme mit einem kompletten Informatikkurs beabsichtigt. In diesem Fall ist eine Einbettung der Exkursion in eine kurze Unterrichtsreihe zum Thema Mikrocontroller denkbar, aber nicht notwendig.

Zusätzlich bietet es sich aus Sicht der Lehrpersonen an, unabhängig von einer Exkursionsteilnahme mit dem ganzen Kurs auf das Schülerlabor-Angebot im Allgemeinen und das Mikrocontroller-Programmierungs-Modul im Speziellen aufmerksam zu machen und einzelne interessierte SuS zu einer Teilnahme zu motivieren.



Kapitel 2 Technische Entwicklung

Die technische Entwicklung des InfoSphere-Moduls ist Voraussetzung für die weitergehende didaktische Gestaltung der Exkursion, da hier die technischen Möglichkeiten der Exkursion ausgelotet und die Schwierigkeiten in der programmiertechnischen Umsetzung gefunden werden. Dieser Arbeitsschritt ist somit mit der späteren Sachanalyse eng verknüpft. Gleichzeitig muss die Entwicklung unter Berücksichtigung der Anforderungen, die sich aus einer Verwendung durch SuS ergeben, geschehen. Die folgenden Unterkapitel beschreiben diesen Prozess näher.

2.1. Gegenstand der Entwicklung

Für die Verständlichkeit der folgenden Abschnitte ist ein Vorgriff auf das Kapitel 3 notwendig:

Das InfoSphere-Modul soll unter dem didaktisch-methodischen Konzept des handlungs- und produktionsorientierten Unterrichts gestaltet werden. Außerdem sollen die SuS im Modul möglichst selbstständig und nach eigenen Interessen arbeiten und gestalten können. Aus beiden Bedingungen ergibt sich, dass die SuS im Verlaufe des Moduls eigenständig und nach eigenen Vorstellungen Produkte zum Thema Hausautomation herstellen können. Das soll realisiert werden durch ein Angebot verschiedener einzelner Projekte, in denen die SuS jeweils einen Aspekt der Hausautomation umsetzen und so insgesamt ein Modell eines automatisierten Hauses bilden. Ein völlig freier Gestaltungsspielraum für die SuS ist dabei nicht zu ermöglichen, da die Realisierung einer Hausautomation Material- und Hardwarevorbereitungen erfordert, die die SuS im zeitlichen Rahmen der Exkursion nicht leisten können und die außerdem nicht zu den Lernzielen gehören. Zu diesen Vorbereitungen gehört zum Beispiel der Bau geeigneter Modelle von Hauselementen wie Türen und Fenstern, in deren Zusammenhang die



SuS das Haus automatisieren. Deshalb wird eine Vorauswahl an Projekten getroffen und die dafür notwendigen Material- und Hardwarevorbereitungen geleistet. Sie werden den SuS später als Grundbausteine bzw. Modelle für ihr jeweiliges Projekt zur Verfügung gestellt. Diese Grundbausteine sollen den SuS aber möglichst viel Gestaltungsfreiraum für ihre Lösung lassen. Das ist in der technischen Umsetzung zu beachten.

Entwickelt werden müssen für die Projekte also Modelle von Hauselementen, zeitaufwendig zu bauende und sicherheitskritische Hardware, die die SuS aus nicht selber herstellen sollen und beispielhafte Programmierlösungen, an denen die Schwierigkeiten und Anforderungen an die didaktische Umsetzung deutlich werden.

2.2. Ansprüche an die Projekte

Die erste und wichtigste Voraussetzung, die das InfoSphere-Modul bzw. die Projekte erfüllen müssen, ist die Gewährleistung der Sicherheit der Teilnehmer. Weitere Bedingungen für die technische Umsetzung sind die Robustheit der Modelle und ihre angemessene Finanzierbarkeit. Die Modelle sollen dabei nicht kindlich anmutend gestaltet sein, damit den Exkursionsteilnehmern eine altersgerechte Arbeitsumgebung zur Verfügung steht. Sie sollen außerdem eine geeignete Größe haben, die ein handliches Arbeiten ermöglicht aber dennoch gute Transportierbarkeit gewährleistet. Weiterhin sollen sie den SuS in der Umsetzung ihrer Projekte möglichst viel Gestaltungsspielraum geben und eine leichte programmiertechnische Umsetzung ermöglichen. Diese Kriterien fließen in die Auswahl der Hausautomationsprojekte ein.

2.3. Auswahl der Projekte

Die Auswahl der Hausautomationsprojekte soll möglichst typische Szenarien der Hausautomation abbilden und die in Kapitel 2.2 beschriebenen Voraussetzungen erfüllen. Unter dieser Berücksichtigung ergeben sich sieben Projekte. Ihre Eignung wurde durch praktische Umsetzung getestet, die Eignungskriterien werden neben einer knappen Beschreibung folgend genannt.

1. **Projekt Alarmanlage** Dieses Projekt ermöglicht den SuS die Umsetzung einer alarmanlagenähnlichen Funktionalität. Dafür stehen ihnen ein Türmodell und ein Fenstermodell zur Verfügung, die jeweils mit Drucksensoren ausgestattet sind, sodass sich ihr jeweiliger Zustand - geöffnet oder geschlossen - messen lässt. Zusätzlich steht ein LCD für die Ausgabe von Informationen zur



Verfügung.

Dieses Projekt eignet sich aufgrund der Vielfalt einfach realisierbarer Funktionalitäten, seiner Alltagsnähe zur Lebenswelt der SuS und guter Anschaulichkeit durch die Fenster- und Türmodelle.

2. **Projekt Feuchtigkeitsmessung** Das Projekt stellt den Teilnehmern einen Feuchtigkeitssensor, einen Heizofen, der über eine ebenfalls zur Verfügung gestellte Steckdose schaltbar ist, und ein LCD-Display zur Verfügung. Damit lassen sich verschiedene Funktionen zur Regulierung der Luftfeuchtigkeit realisieren. Dieser Funktionsbereich eignet sich einerseits durch seine Nähe zur Lebenswelt der SuS, andererseits durch seine motivierende Besonderheit, die er durch seine geringe Anwendungshäufigkeit in der Realität aufweist. Zusätzlich sind die Funktionen einfach umsetzbar.
3. **Projekt Jalousiesteuerung** Für die Jalousiesteuerung wird ein Fenstermodell samt steuerbarer Jalousie, zwei Helligkeitssensoren und LCD-Display zur Verfügung gestellt. Die realisierbare Vielfalt von einfach umsetzbaren Automatikfunktionen, die Nähe zur Lebenswelt der SuS und die Anschaulichkeit der Funktionalität sowie die gute Testbarkeit während der Entwicklungsphase qualifizieren dieses Projekt für die Aufnahme in die Exkursion.
4. **Projekt Klimasteuerung** Mit Temperatursensor, schaltbaren Steckdosen, Heizofen, Ventilator und LCD-Display lässt sich eine automatische Raumklimasteuerung bauen. Die Grundfunktionen sind schnell umsetzbar. Die Alltagsnähe macht dieses Projekt für SuS interessant.
5. **Projekt Lichtsteuerung** Die Teilnehmer erhalten für die Umsetzung einen Entfernungsmesser, einen Umgebungslichtsensor, eine schaltbare Steckdose und eine Lampe. Damit lassen sich verschiedene Varianten realisieren, beispielsweise gestengesteuertes Licht, zeitlich gesteuertes Licht oder Kombinationen davon. Die Gestensteuerung macht das Projekt für SuS interessant. Einfache Umsetzbarkeit, Anschaulichkeit und Alltagsrelevanz empfehlen die Lichtsteuerung als Projekt.
6. **Projekt Türöffner** Hierfür erhalten die Teilnehmer ebenfalls ein Türmodell. Die Tür lässt sich durch einen Motor öffnen und schließen. Dazu bekommen sie zwei Entfernungsmesser. Durch geeignete Kombination lässt sich damit ein automatischer Türöffner bauen, wie er den SuS aus dem Alltag bekannt ist. Die Arbeit mit einem Motor weckt das Interesse am Türöffner und



gleicht die geringere Anzahl der realisierbaren Funktionen aus. Der Türöffner kann in angemessener Zeit gebaut werden.

- 7. Projekt Monitoring** Dieses Projekt unterscheidet sich durch fehlende Hardware von den anderen Projekten. Es zielt auf die Kombination aller vorangegangenen Arbeiten zu einem komplett automatisierten Modellhaus ab und bedingt eine vorherige Umsetzung der anderen Projekte. Die Funktionalität sieht vor, dass auf einem Computermonitor der aktuelle Status aller anderen Projekte in Echtzeit angezeigt wird und alle Projekte nur von einem einzigen, zentralen Computer aus gesteuert werden. Auf diese Weise wird ein Zusammenhang zwischen den einzelnen Projekten geschaffen und gleichzeitig eine für die Hausautomatisierung typische zentrale Steuereinheit hergestellt. Es eignet sich nicht für eine parallele Bearbeitung zu den anderen Projekten, was in der didaktischen Umsetzung berücksichtigt werden muss.

2.4. Gestaltung der einzelnen Hausautomationsprojekte

In diesem Abschnitt werden besonders relevante Aspekte des Gestaltungsprozesses hervorgehoben.

So ist zu erwähnen, dass der Sicherheit der SuS oberste Priorität eingeräumt wurde. Die schaltbaren Steckdosen sind mit Personenschutzschaltern versehen, um bei einem eventuellen Elektrounfall Personenschäden zu verhindern (vgl. Abbildung 2.1).

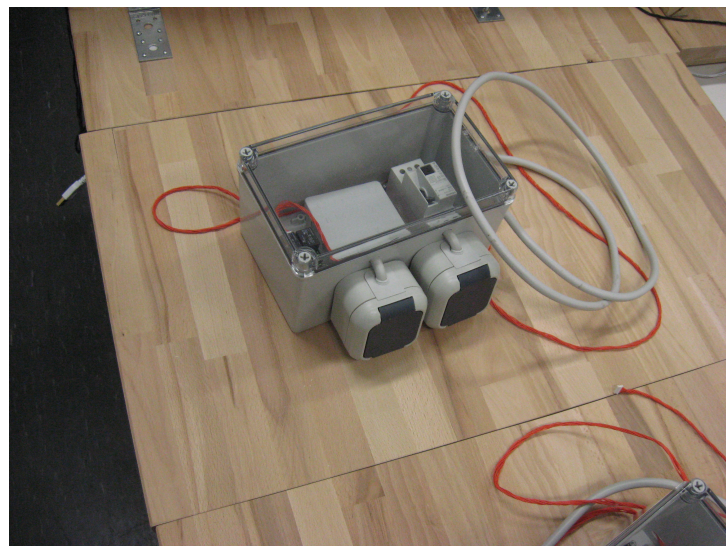


Abbildung 2.1.: Steuerbare Steckdosen



Auch sind alle Modelle durch geeignete Konstruktion gegen Umkippen gesichert, damit die teils schweren Holzmodelle nicht versehentlich auf SuS fallen können.

Darüber hinaus ist zur Gestaltung zu erwähnen, dass zu jedem Projekt eine Bodenplatte aus Holz gehört (vgl. Abbildung 2.2). Auf diesen sind entweder die Holzmodelle montiert (Beispiel automatischer Türöffner Abbildung 2.3) oder die SuS können ihre Hardware auf diesen Platten aufbauen. Diese Bodenplatten ergeben zusammengesetzt den Grundriss des Modellhauses (vgl. Abbildung 2.4). Dieses Detail ist in der didaktischen Gestaltung der Exkursion verwendbar.

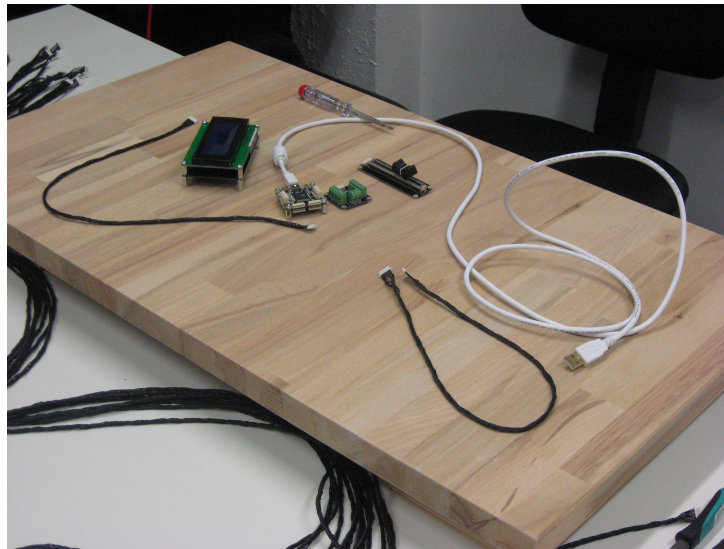


Abbildung 2.2.: Bodenplatte

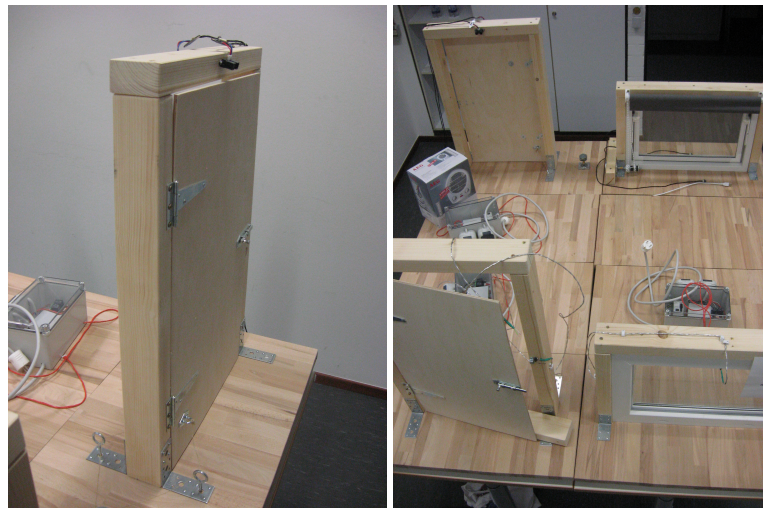


Abbildung 2.3.: Türöffner

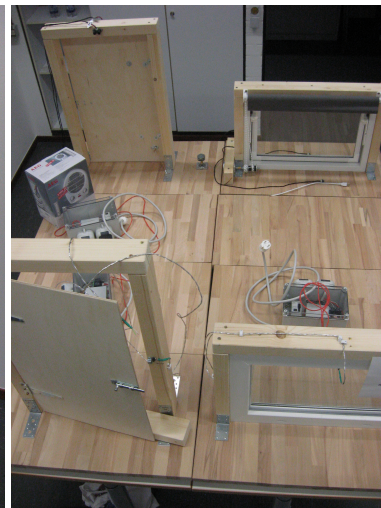


Abbildung 2.4.: Hausgrundriss

Die entwickelten Quellcode-Musterlösungen sind exemplarische Lösungen und stellen keinesfalls die einzige Umsetzungsmöglichkeit



dar. Da individuelle Lösungsansätze und Funktionen von den SuS gewünscht sind, erfüllt eine Musterlösung hier auch nicht ihren eigentlichen Sinn. Vielmehr sollen sie den Betreuern des InfoSphere-Moduls eine Idee davon vermitteln, wie eine Lösung beispielsweise aussehen könnte. Sie liegen in digitalisierter Form den Exkursionsunterlagen bei.



Kapitel 3 Didaktik

Dieses Kapitel bildet den Schwerpunkt der Arbeit. Es beschäftigt sich mit der didaktischen Gestaltung des InfoSphere-Moduls. Inhaltlich werden neben der Bedingungs- und Sachanalyse die Gesamtstrukturierung der Exkursion und die Detailgestaltungen thematisiert.

3.1. Hypothetische Bedingungsanalyse

Um die Gestaltung des InfoSphere-Moduls möglichst gut an die Bedürfnisse der Teilnehmer anzupassen, werden diese genauer betrachtet. Die Schwierigkeit ergibt sich dabei aus der in Kapitel 1.4 genannten Zielgruppe. Das InfoSphere-Modul kann nicht individuell auf einen konkreten Klassen-, Kurs oder losen Gruppenverbund angepasst werden, da die Teilnehmer wechseln und jeweils unterschiedliche Fähigkeiten- und Anforderungsprofile mitbringen. Es lassen sich somit keine spezifischen Teilnehmeranalysen beispielsweise hinsichtlich ihres Vorwissens und ihrer Programmierfähigkeiten machen.

Um diesem Zustand gerecht zu werden, gilt es, die Schnittmenge der Gemeinsamkeiten unter den Teilnehmern in der Modulgestaltung zu berücksichtigen. Weiterhin muss die Exkursion so angelegt sein, dass sie die über diese Schnittmenge hinausgehenden Teilnehmeranforderungen durch größtmögliche Individualisierbarkeit berücksichtigt. Es ist also Ziel, möglichst viele Lern- und Arbeitstypen zu bedienen und die SuS möglichst wenig einzuschränken.

Die erwähnte Schnittmenge der Gemeinsamkeiten unter den Teilnehmern lässt sich nur auf Basis des Lehrplans vorhersagen. Allerdings lässt selbst der Lehrplan keine hinreichend genaue Prognose zu, da sich aus den schulinternen Curricula und der Leistungsfähigkeit der SuS Unterschiede im schulischen Unterrichtsverlauf ergeben. Hinzu kommt, dass die Teilnehmerschaft sich regelmäßig nicht aus einem festen Klassenverbund, sondern aus einer losen Zusammenkunft von SuS verschiedener Schulen und Stufen zusammensetzt.



Um die Bedingungsanalyse dennoch auf Tatsachen und nicht auf reine Spekulationen zu stützen, müssen ebendiese Tatsachen bei der Exkursionsgestaltung geschaffen werden. Das geschieht durch Festsetzen von Mindeststandards für die Teilnehmer. Diese Mindeststandards beschreiben die minimal mitzubringenden Fähigkeiten für eine erfolgreiche Teilnahme. Sie werden der Exkursionbeschreibung beigelegt, so dass Interessenten selbstständig über ihre Eignung urteilen und gegebenenfalls Lücken selbstständig vor der Teilnahme schließen können. Als Teilnahmevoraussetzung werden - unter Berücksichtigung der sich aus der Sachanalyse in Kapitel 3.2 ergebenden Notwendigkeiten - folgende Punkte in der Modulbeschreibung angegeben:

- **Programmiererfahrung** ist schwierig messbar. Als ungefährer Richtwert wird die Programmiererfahrung eines halben Schuljahrs bei regelmäßigem Programmieren mit Java angegeben (siehe dazu auch Abschnitt 1.4.3).
- **Fähigkeiten im Binärrechnen** sollten für gewöhnlich vorausgesetzt werden können, wird sicherheitshalber aber aufgeführt.

Der Inhalt der Anforderungsliste wird vom Lehrplan NRW (vgl. Kapitel 1.4.1) abgedeckt, sodass diese Anforderungen ohne weitere Bemühungen von allen Interessenten aus der Zielgruppe erfüllt werden können sollten.

Auf diesem Wege ist sichergestellt, dass es einerseits - ohne hypothetische Annahmen - eine verlässliche Kompetenzbasis bei den Teilnehmern gibt, die in der Modulgestaltung vorausgesetzt werden kann, und andererseits keine Interessenten durch hohe Teilnahmebedingungen abgeschreckt werden.

3.2. Sachanalyse

Die Sachanalyse beschäftigt sich mit der Analyse der zu vermittelnden Inhalte unter der Zielsetzung, den inhaltlichen Umfang abzustecken und sich eventuell ergebende Verständnisschwierigkeiten für SuS aufzudecken, damit diese in der didaktischen Ausgestaltung angemessen berücksichtigt werden können.

Kontext des InfoSphere-Moduls

Zuerst zum Oberthema des InfoSphere-Moduls, der Hausautomation: Den SuS ist zu vermitteln, was Hausautomation meint und was sie



bezweckt. Dabei sind alltagsrelevante und verbreitete Beispiele wie automatische Klimasteuerungen, Jalousiesteuerungen oder Sicherheits- und Alarmanlagen anzuführen. Als Zweck der Hausautomation sind die Vergrößerung des Komforts oder Erhöhung von Sicherheit zu nennen.

Als nächstes folgt eine Einführung in den Bereich der Mikrocontroller, die den inhaltlichen Schwerpunkt der Exkursion bilden. Zunächst muss den SuS der Unterschied zwischen einem Mikrocontroller und einem aus dem Alltag bekannten Computer deutlich gemacht werden. In diesem Zusammenhang sollen auch die unterschiedlichen Anwendungsbereiche und Aufgaben von Mikrocontrollern vermittelt werden. Da in der Exkursion mit konkreten Mikrocontroller-Produkten gearbeitet wird, ist eine inhaltliche Konzentration auf diese speziellen Exemplare sinnvoll. Das schließt eine Erläuterung zu Mikrocontrollern im Allgemeinen nicht aus. Im konkreten Anwendungsszenario der Tinkerforge-Mikrocontroller übernimmt es der Computer, Daten zu verarbeiten und Aktionen von Aktoren im Gesamtüberblick zu steuern. Die Aufgaben der Tinkerforge-Mikrocontroller liegen in der technischen Steuerung der Aktoren und Sensoren sowie der Weitergabe von Messdaten an den Computer. Die Flussrichtungen von Mess- und Steuerdaten vom Sensor über den Mikrocontroller zum Computer und zurück über den Mikrocontroller an den Aktor ist für das Verständnis notwendig und somit zu vermittelndes Wissen. Die Begrifflichkeiten Sensor und Aktor sind ebenfalls unbekannt und müssen erläutert werden. Ihre Bedeutung lässt sich mit jeweiligen Beispielen - wie Temperatursensoren und Motoren als Aktoren - verdeutlichen. In dieses Themengebiet fallen viele weitere relevante Fachbegriffe, wie Chip, Firmware, System-on-a-Chip und Platine, die den SuS bekannt sein könnten und die von diesen thematisiert werden könnten. Das ist in der didaktischen Gestaltung zu berücksichtigen.

Mikrocontroller

Weiterhin muss den SuS der Umgang mit den Tinkerforge-Mikrocontrollern beigebracht werden. Ihnen muss beigebracht werden, wie die Mikrocontroller mit Sensoren, Aktoren und dem Computer zu verbinden sind und wie die Tinkerforge Verwaltungssoftware, der Brick-Viewer, bedient werden muss. Weiterhin müssen die SuS lernen, die Mikrocontroller mit eigenem Java-Quellcode zu steuern. Welche Fähigkeiten und welches Wissen dafür notwendig ist, ergibt sich hauptsächlich aus den Eigenschaften der Tinkerforge-Produkte und lässt sich der dazugehörigen Tinkerforge-Dokumentation [6] entnehmen. Die einzelnen Ergebnisse der technischen Ausarbeitung fließen direkt in die didaktische Gestaltung der Exkursionsmaterialien ein und



werden in diesem Zusammenhang näher erläutert.

Ähnliches gilt für die von den SuS zu erlernenden Java-Spracheigenschaften. An dieser Stelle sei erwähnt, dass die SuS neben weiteren kleineren Neuheiten mit try-catch-Blöcken, Exceptions und Listnern umgehen müssen. Es ist von ihnen keinerlei Vorwissen über diese Konzepte zu erwarten, sodass dieses in geeigneter Form vermittelt werden muss.

3.3. Handlungs- und produktionsorientierter Unterricht

Wie der didaktischen Zielsetzung (siehe Kapitel 1.2) angegeben, sollen praktische Elemente und selbstständiges Arbeiten der Teilnehmer im Vordergrund der didaktisch-methodischen Konzeption des InfoSphere-Moduls stehen. Dieser Zielsetzung entspricht das Prinzip des handlungsorientierten Unterrichts. Was mit diesem Prinzip gemeint ist und weshalb es sich für die Umsetzung in der Exkursion anbietet, wird in den folgenden Abschnitten geklärt.

Charakterisierung handlungsorientierten Lehrens und Lernens

Handlungsorientierter Unterricht fokussiert Schüleraktivität als zentrales Element des Unterrichts und verbindet Kopf- und Handarbeit zu einem ganzheitlichen Unterricht [4, vgl. S. 214 ff.]. Gudjons stellt dabei als wesentliches Merkmal für Handeln im inhaltlich-pädagogischen Sinne die Selbst- oder Mitbestimmung der SuS, ihre Teilhabe an der Planung und ihre Identifikation mit dem Sinn der Aufgabe heraus [2, vgl. S. 69].

Bei der Umsetzung werden verschiedene didaktische Ansätze, wie beispielsweise selbstgesteuertes Lernen oder entdeckendes Lernen, verfolgt. Als typisches Konzept handlungsorientierten Unterrichts stellt Gudjons den Projektunterricht vor [2].

Pädagogische Begründung handlungsorientierten Unterrichts

Herbert Gudjons fasst drei Begründungsebenen für handlungsorientierten Unterricht zusammen: eine sozialisationstheoretische, eine anthropologisch-lernpsychologische und eine didaktisch-methodische. Auf der ersten Ebene wird argumentiert, dass Schule von einem „Schwund kontinuierlicher Erfahrungen mit Menschen und Dingen“ [2, S.



67] betroffen sei und es diese fehlende Erfahrung der SuS mit handlungsorientiertem Unterricht auszugleichen gilt. Auf diesem Weg sollen handelnd Denkstrukturen aufgebaut und ein Zugang zur realen Welt durch vielfältige Sinneserfahrungen gegeben werden. [2, vgl. S. 68]

„Auf anthropologisch-lernpsychologischer Ebene wird handlungsorientiertes Lernen dadurch begründet, dass der Mensch in einer dialektischen Person-Umwelt-Beziehung gesehen wird. Person und Umwelt werden nicht als getrennte Gegebenheiten aufgefaßt; vielmehr verschränken sich im Begriff der Handlung Aufbau/ Veränderung individueller Strukturen und Herstellung/ Gestaltung von materieller und sozialer Umwelt.“ [2, S. 70]

Zuletzt zur didaktisch-methodischen Begründungsebene: Handlungsorientierter Unterricht ist hiernach als Zusammenfassung bereits vorliegender verwandter Konzepte unter einer Leitperspektive zu verstehen. Dabei stellt Gudjons fest, dass hierin keine Begründung für handlungsorientierten Unterricht, sondern eine Beschreibung zu sehen ist. [2, vgl. S. 70]

Für die Begründung, das InfoSphere-Modul handlungsorientiert zu gestalten, scheint besonders die erste Ebene relevant. Der schulische Informatikunterricht ist aus verschiedenen Gründen nur begrenzt in der Lage, die SuS lebensweltliche Dinge der Informatik angemessenen erfahren zu lassen. Um dies zu unterstützen ist eine zusätzliche Erfahrung mit Aspekten der Informatik, ermöglicht durch eine handlungsorientierte Exkursion, zu begrüßen.

Projektunterricht als Konzept handlungsorientierten Unterrichts

Wie bereits erwähnt ist Projektunterricht ein anerkanntes Konzept handlungsorientierten Unterrichts. Was das ist und warum er für die Anwendung im InfoSphere-Modul geeignet ist, klärt dieser Abschnitt. Projektunterricht lässt sich anhand verschiedener Merkmale beschreiben oder definieren:

Zuerst sollte ein Situationsbezug vorhanden sein. Dieser ist durch das Thema der Exkursion, der Hausautomatisierung, gegeben. Weiteres Merkmal für Projektunterricht ist seine Orientierung an den Interessen der Beteiligten. Dieses Kriterium ist in der didaktischen Gestaltung angemessen zu berücksichtigen. Wie genau die Neigungen der Teilnehmer einbezogen werden, wird im Kapitel 3.4 erläutert. Gesellschaftliche Praxisrelevanz wird ebenfalls als Eigenschaft von Projektunterricht angesehen. Diese ist in der Exkursion ebenfalls durch den Bezug auf das praxisrelevante und alltagsnahe Themenfeld Hausautomation gegeben. Weiterhin sind die Selbstorganisation der Teilnehmer, Produktorientierung und soziales Lernen als Eigenschaften aufzuführen. Wie



diese drei Aspekte konkret realisiert werden, wird ebenfalls in der didaktischen Gestaltung geklärt. Festzuhalten ist an dieser Stelle, dass sich der Exkursionsinhalt prinzipiell für eine Erfüllung dieser Kriterien eignet und somit einer Konzeptuierung des InfoSphere-Moduls als projektorientierte Veranstaltung nichts entgegensteht. Wie gut das gelingt und was die Grenzen der Projektorientierung in der Exkursion sind, wird in der Reflexion dieser Arbeit, in Teil 4, thematisiert.

3.4. Didaktische Gestaltung

In den folgenden Unterkapiteln wird die Ausgestaltung des InfoSphere-Moduls im Detail erläutert und begründet. Zuerst wird der Gesamtverlauf der Exkursion in unreflektierter Form dargestellt (3.4.1). Anschließend werden die Leitpräsentation (3.4.2), Teilnehmermaterialien (3.4.3) und Betreuermaterialien (3.4.4) im Gesamtkontext, in der Reihenfolge ihrer Verwendung, vorgestellt und diskutiert. Alternative Gestaltungsmöglichkeiten werden beschrieben und gegen die gewählte Variante abgewogen. Das Zeitmanagement der Exkursion wird im letzten Abschnitt von Kapitel 3.4.1, nachdem der Gesamtverlauf bekannt ist, erläutert.

3.4.1. Gesamtverlauf der Exkursion

Der Gesamtverlauf des InfoSphere-Moduls ist in Abbildung 3.1 grafisch dargestellt. Diese Grafik entstammt dem Betreuermaterial und ist ursprünglich als Orientierungshilfe für die Exkursionsbetreuer entworfen. Sie dient an dieser Stelle aber auch als optischer Leitfaden für die folgenden Ausführungen. Soweit nicht anders erwähnt sind alle Verweise in diesem Unterkapitel auf diese Abbildung bezogen.

Die Struktur der Grafik weist zwei Spalten auf: Die linke Spalte gibt die inhaltliche Phase des InfoSphere-Moduls wieder, die rechte Spalte stellt die jeweilige Arbeits- bzw. Sozialform zur entsprechenden Phase dar.

Das InfoSphere-Modul ist eingebettet in den allgemeinen organisatorischen Rahmen des Schülerlabors, der zu Beginn eine Begrüßungsrunde, organisatorische Anmerkungen und eine Fach-Vorwissenserhebung der Teilnehmer und zum Abschluss eine Besprechung und Evaluation sowie eine weitere Fachwissenserhebung umfasst. Sowohl Beginn als auch Ende sind durch das Schülerlabor als obligatorisch festgesetzt und gehören nicht zum eigentlichen InfoSphere-Modul. Sie sind in der Grafik jeweils grau dargestellt und spielen in den folgenden Erörterungen keine Rolle. Das gesamte InfoSphere-Modul wird durch eine Präsentation geleitet. Sie stellt den inhaltlichen roten Faden dar, leitet durch die einzelnen Abschnitte und



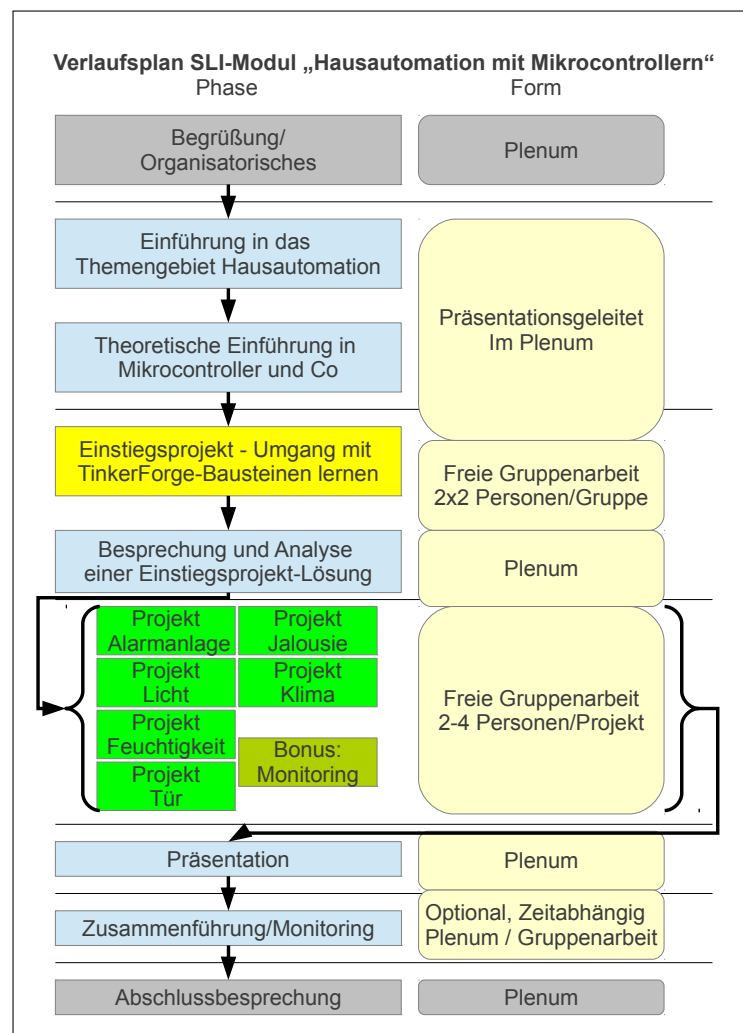


Abbildung 3.1.: Verlaufsplan InfoSphere-Modul

dient als Träger für Informationen und als Tafelersatz. Sie wird im Unterkapitel 3.4.2 genauer vorgestellt und begründet.

Betreuerteam

Für die Durchführung des InfoSphere-Moduls ist keine feste Anzahl an Betreuerinnen und Betreuern¹ notwendig, es ist aufgrund einer verbesserten individuellen Betreubarkeit der Teilnehmer aber wünschenswert, dass mindestens zwei, besser drei Betreuer anwesend sind. Diese Anzahl entspricht auch der Betreueranzahl, die das Schülerlabor gewöhnlicherweise durchführt. Da das Betreuerteam

¹ Zur Verbesserung der Lesbarkeit wird im Folgenden nur die maskuline Form verwendet.



wechseln kann, sind entsprechende Hilfsmaterialien zur Einarbeitung und als Orientierungshilfe für Betreuer notwendig. Näheres dazu wird in Kapitel 3.4.4 thematisiert.

Theoretische Einführung

Das Hausautomations-Modul beginnt mit einem einführenden theoretischen Inhaltsblock, bestehend aus zwei inhaltlichen Phasen: Einer Einführung in das Themengebiet Hausautomation und einer theoretischen Einführung in den Themenkomplex Mikrocontroller. Dieser Inhaltsblock hat die Aufgabe, einen inhaltlichen Einstieg zu schaffen und die SuS auf einen einheitlichen minimalen Wissensstand zu den Themen zu bringen. Weiterhin stellt dieser Block die Motivation für alle weiteren Inhalte dar.

Die Einführung in das Themengebiet Hausautomation klärt die Frage, was Hausautomation ist und stellt einige haushaltsübliche Beispiele für Hausautomationen vor. Nebeneffekt ist, dass hierbei einige Hausautomationstechniken vorgestellt werden, die die SuS im späteren Verlauf teilweise selber umsetzen können. Dadurch wird bereits an dieser Stelle implizit die Begründung und der Alltagsbezug für die sechs Projekt- bzw. Arbeitsbereiche geliefert.

Die theoretische Einführung in den Themenkomplex Mikrocontroller vermittelt den SuS das Grundwissen darüber, was Mikrocontroller eigentlich sind, was sie von herkömmlichen Computern unterscheidet und wie ihre grundlegende Funktionsweise ist. Dabei werden auch ausgewählte Fachbegriffe eingeführt, die für die weiteren Arbeitsschritte hilfreich sind. Aufgrund der begrenzten zur Verfügung stehenden Zeit (s.u.) und auch, um die SuS kognitiv nicht durch Überladung zu überlasten, kann in dieser Phase nicht auf Detailspekte des Themengebiets Mikrocontroller eingegangen werden. Viele Anwendungsbeispiele und Typen von Mikrocontrollern können nicht vorgestellt werden. Es wird sich deshalb bei konkreten Beispielen auf die Tinkerforge-Mikrocontroller konzentriert. Die Funktionsprinzipien und Grundlagen werden also exemplarisch an diesen Mikrocontrollern vermittelt. Das hat zur Folge, dass sie SuS gleichzeitig die abstrakte Funktionsweise der Produkte kennen lernen, die sie im weiteren Verlauf verwenden.

Die Inhalte beider Phasen werden präsentationsgeleitet im Plenum diskutiert und vorgestellt. Der Grund hierfür liegt im Zeitmanagement. Eine Erarbeitung in Einzel- oder Gruppenarbeit würde deutlich mehr Zeit in Anspruch nehmen und zwangsläufig eine Sicherungsphase nach sich ziehen, die die SuS nach ihren eigenen Recherchen auf einen



einheitlichen Wissensstand bringt. Dieser Aufwand entfällt, da alle Inhalte sofort der gesamten Teilnehmerschaft zugänglich gemacht werden. Da zusätzlich der didaktische Schwerpunkt auf Handlungs- und Produktionsorientierung liegt (vgl. Kapitel 3.3), ist ein großer zeitlicher Anteil an theoretischen Arbeitsphasen nicht wünschenswert. Deshalb ist diese theoretische Einstiegsphase zeitlich kurz und inhaltlich knapp und prägnant angelegt.

Praktische Einführung

Durch den Präsentationsleitfaden unterstützt geht der theoretische Inhaltsblock fließend über in den praktischen. Dieser anschließende Block beinhaltet die praktische Einführung in den Umgang mit den Tinkerforge-Mikrocontrollern und besteht wieder aus zwei Phasen: Der praktischen Einarbeitungsphase (Der gelbe Kasten im Schaubild) und der Ergebnisbesprechungsphase. Die Leitpräsentation führt wieder durch diese Phasen hindurch und bietet eine Orientierung im Gesamttablauf.

Die Teilnehmer arbeiten in diesem Inhaltsblock in Zweiergruppen zusammen. Zu Beginn der praktischen Einarbeitungsphase erhalten die SuS alle Materialien, die sie zur Bearbeitung des Einstiegsprojektes benötigen. Dazu gehören Laptops und Mikrocontroller samt Zubehör. Ein Betreuer stellt nun nacheinander die Schritte vor, die nötig sind, um die Mikrocontroller in Betrieb zu nehmen. Dazu gehört erstens das Anschließen der Mikrocontroller untereinander und an den Computer und zweitens eine Einführung in die Bedienung der Tinkerforge-Steuersoftware Brick-Viewer. Die einzelnen dafür notwendigen Arbeitsschritte werden vom Betreuer vor dem Plenum vorgeführt. Jeder Arbeitsschritt wird von allen Teilnehmerpaaren durch direktes Nachahmen nachvollzogen und begleitet. Das sichert zum einen die Aufmerksamkeit der SuS und zum anderen, dass jeder Teilnehmer den Arbeitsschritt verstanden hat und das Plenum einen gleichmäßigen Arbeitsfortschritt erzielt. Da die einzelnen Handlungen kurz sind, entsteht für leistungsstarke Teilnehmerpaare kaum Wartezeit. Gleichzeitig bleiben leistungsschwächere Teilnehmerpaare durch die enge Anleitung nicht zurück. Parallel dazu füllen die Paare vorgefertigte Merkzettel aus, auf denen sie die einzelnen notwendigen Arbeitsschritte für das Betreiben der Mikrocontroller notieren. Durch diese Methodik soll ein zügiges Einarbeiten in die grundlegenden Arbeitsschritte erfolgen. Das direkte praktische Nachvollziehen des Gesehenen festigt das Gelernte Wissen sofort und macht es besser wiederabrufbar, da die SuS in den Freiarbeitsphasen nicht mit unbekanntem Situationen konfrontiert werden. Eventuell dennoch entstandene Erinnerungslücken werden durch



die mit eigenen Worten ausgefüllten Merktzettel ausgeglichen. Verglichen mit einem fertig vorgegebenen Merktzettel können sich die SuS beim Lesen ihrer eigenen Worte besser an das in diesem Kontext Gesehene erinnern. Nebenbei wird so selbstständige Erstellen von Notizen trainiert. Diese Vorgehensweise erspart den Teilnehmern ein langwierigeres und dadurch ermüdenderes Durchlaufen eines schriftlichen aufgabengeleiteten Tutorials. Die durch die enge mündliche Anleitung erzielten schnellen kleinen Teilerfolge wirken zusätzlich motivierend. Insgesamt wird durch diesen Arbeitsschritt gesichert, dass alle Teilnehmer in den folgenden Arbeitsphasen selbstständig in der Lage sind, die Mikrocontroller korrekt anzuschließen und die Kontrollsoftware zu bedienen. Anschließend müssen die SuS nur noch lernen, wie man die Mikrocontroller mit eigenem Quellcode verwendet.

Dazu lösen sich die SuS wieder von ihren Arbeitsmaterialien und verfolgen im Plenum die Präsentation. Der moderierende Betreuer erarbeitet unter Einbeziehung der Teilnehmer das Grundgerüst des für den Betriebs der Mikrocontroller notwendigen Java-Quellcodes mitsamt der den SuS unbekanntem Eigenschaften von Java (vgl. Kapitel 3.2). Zur Sicherstellung der Aufmerksamkeit und um an vorhandenes Vorwissen anzuknüpfen, werden die nötigen Schritte in eng angeleiteter Weise gemeinsam mit den SuS erarbeitet. Im Unterschied zu einem reinen Vortrag wird den SuS so auch die Ähnlichkeit zu ihnen bekannten Quelltexten besser vermittelt, was die Hemmschwelle im Umgang mit dem neuen Code verringern soll. Wie genau der Quellcode erarbeitet wird, wird in Kapitel 3.4.2 bei direkter Betrachtung der anleitenden Präsentation erläutert.

An dieser Stelle könnte man die SuS den Quellcode, während er gemeinsam erarbeitet wird, mitschreiben lassen. Diese Methode kostet aber sehr viel Zeit, da das Abtippen von der Tafel sehr zeitintensiv und fehleranfällig ist. Deswegen wird darauf verzichtet und stattdessen anschließend der erarbeitete Quellcode in ausgedruckter Form ausgeteilt. An dieser Stelle endet das Arbeiten im Plenum, jeweils zwei Paare werden anschließend zu Vierergruppen vereint, womit die Arbeitsform übergeht zur freien Gruppenarbeit. Die Gruppen erhalten nun weitere Arbeitsmaterialien in Form einer Aufgabenstellung, einer ausgedruckten Version des zuvor gemeinsam erarbeiteten Beispielquelltextes und diesen in digitaler Form als Grundlage für die weitere Arbeit. Ziel der freien Gruppenarbeitsphase ist das weitere Lernen und Üben des Umgangs mit den Tinkerforge-Mikrocontrollern. Die damit verbundene Aufgabenstellung lässt die Teilnehmer kein Produkt für Hausautomation herstellen, sondern durchläuft alle für die Projektphase benötigten Inhalte. Durch diese Übungsphase sollen die SuS so sicher im Umgang mit den Mikrocontrollern werden, dass sie in der folgenden Gruppenarbeit selbstständig und unter dem inhaltlichen Schwerpunkt der Hausautomation arbeiten können.

Als Arbeits- und Sozialform wird die Gruppenarbeit gewählt, um der



genannten Zielsetzung der Steigerung der Sozial- und Teamarbeitskompetenz gerecht zu werden. Als weiteres Argument gilt, dass sich die SuS innerhalb der Vierergruppe bei Schwierigkeiten gegenseitig helfen können und leistungsstärkere Teilnehmer leistungsschwächere unterstützen und mitziehen. Durch die Gestaltung der Aufgabe soll der gegenteilige Effekt, dass nur Leistungsstärkere arbeiten und Leistungsschwächere nur zusehen, vermieden werden. Wie diese Ziele genau erreicht werden sollen, wird ebenfalls unter Betrachtung der Arbeitsmaterialien in Kapitel 3.4.2 gezeigt.

Als zweite Phase schließt sich die Sicherungsphase an. Hier wird, wieder im Plenum, eine mögliche Lösung einer Gruppe von dieser selbst vorgestellt und besprochen. So können verbliebene Fragen geklärt werden. Außerdem erhalten eventuelle Nachzüglergruppen nun wieder Anschluss.

Mit dem Ende der Sicherungsphase endet auch der zweite Inhaltsblock, der die Teilnehmer auf den Hauptarbeitsblock vorbereitet hat. Der Hauptarbeitsblock und damit die inhaltliche Kernphase (in der Abbildung grün dargestellt) der Exkursion beginnt nun.

Kernarbeitsphase: Hausautomation mit Mikrocontrollern

Im Hauptarbeitsblock können die SuS Mikrocontroller für verschiedene Hausautomatationen einsetzen. Es gibt sechs unterschiedliche Hausautomations-Projekte², die sich alle mit der Realisierung einer alltagstypischen Hausautomationstechnik beschäftigen (vgl. dazu auch Kapitel 2.3). Den Teilnehmern werden die Inhalte der einzelnen Projekte kurz vorgestellt. Anschließend können sich die SuS entsprechend ihrer Interessen auf die einzelnen Projekte aufteilen.

Dabei muss, bei geringer Teilnehmerzahl, nicht jedes Projekt besetzt werden. Sofern die Teilnehmerzahl es zulässt, ist die maximale Gruppengröße pro Projekt auf vier Teilnehmer beschränkt. Die optimale Besetzung pro Projekt liegt bei zwei bis drei Teilnehmern, Einzelarbeiten sind zu vermeiden. Diese Angaben zur Gruppengröße basieren einerseits auf der Analyse der einzelnen Projekte während der technischen Gestaltung. Diese hat gezeigt, dass der Arbeitsaufwand zu gering ist, um mehr als vier Personen zu beschäftigen. Bei größeren Gruppengrößen ist zu befürchten, dass sich einzelne Gruppenmitglieder langweilen bzw. die Gruppenmitglieder sich durch ihre Anzahl gegenseitig beim Arbeiten behindern. Andererseits basieren diese Werte auf Testläufen mit SuS (vgl. Kapitel 4.1), in denen sich die genannten Werte als praktisch durchführbar erwiesen haben. Gruppenarbeit wird als Sozialform aus der gleichen Motivation heraus gewählt, wie bereits die Gruppenarbeit im Einstiegsprojekt. Die Begründung gilt analog.

² Sie werden im Folgenden als Projekte bezeichnet.



Entsprechend der Zielsetzung (vgl. Kapitel 1.2) sind die einzelnen Projekte so gestaltet, dass sich verschiedene Funktionen im jeweiligen Projektkontext entwickeln lassen, sodass die SuS gemäß ihren eigenen Neigungen, Interessen und Fähigkeiten frei arbeiten können. Die Projekte sollen einen Kompromiss aus frei gestaltendem Arbeiten, inhaltlicher Impulsgebung und Finanzierbarkeit bilden.

Es ist zu erwarten, dass die einzelnen Projektgruppen nicht zeitgleich mit ihren Arbeiten fertig werden, da die Leistungsfähigkeit der Teilnehmer variiert. Fertige Gruppen sollen von den Betreuern zuerst angeregt werden, ihre Lösungen weiter auszubauen und zu optimieren, bevor für die Gruppe beschäftigungsfreie Zeit entsteht. Darüber hinaus gibt es ein Bonusprojekt für vorzeitig fertige Gruppen. Es beinhaltet das Programmieren einer Monitoring-Software für die Verwendung in der optionalen Zusammenführungsphase. Die Bonus-Projektbearbeitung ist allerdings an einige Bedingungen geknüpft. Aus diesem Grund werden die Einsatzbedingungen des Bonusprojekts erst im Abschnitt Zeitmanagement erläutert.

Sollte eine Arbeitsgruppe wider Erwarten große Schwierigkeiten mit der Lösung ihrer Aufgabe haben, liegen Hilfestellungen bereit. Diese Hilfestellungen sind im Moodle-Kurs für Teilnehmer unsichtbar hinterlegt und können von den Betreuern bei Bedarf freigeschaltet werden. Es gibt für jedes Projekt Hilfestellungen in zwei Stufen: Die erste Stufe bietet einige vorimplementierte Funktionen und mehr Hilfestellungen durch anleitende Kommentare an, sodass weniger Eigenleistung von den SuS erwartet wird. Die zweite Hilfestellung funktioniert nach dem gleichen Prinzip, geht aber noch weiter. Sie stellt eine fast fertige Lösung dar, zu der die SuS nur noch wenig Funktionalität hinzufügen müssen. Diese Hilfestellungen sind nur für Ausnahmefälle gedacht, falls Teilnehmer trotz intensiver Betreuung absolut keinen Zugang zur Lösung der Aufgabe bekommen. Ein Nachteil dieser Hilfestellungen ist, dass sie die Effekte der Freiarbeit zunichte machen, da die SuS sich nun am Lösungsweg und Funktionsumfang des Autors orientieren müssen und ihre Kreativität dabei nicht gefördert wird. Deshalb ist das Herausgeben der Hilfestellung nur als Notlösung zu verstehen, um den Teilnehmern noch zu einem minimalen Teilnahmeerfolg zu verhelfen.

Ergebnispräsentation

Im Anschluss an die Projektarbeitsphase sollen sich die Teilnehmer ihre Projektergebnisse gegenseitig vorstellen. Praktisch erfolgt das, indem die Teilnehmerschaft geschlossen nacheinander alle bearbeiteten Projekte besucht und die entsprechende Gruppe kurz ihre Arbeit erläutert und das Ergebnis demonstriert. Die Präsentationen sollen weder mit



Hilfsmitteln vorbereitet werden noch soll der Ergebnisquellcode besprochen werden, da beides zu viel Zeit für zu wenig Nutzen in Anspruch nimmt. Ziele der Präsentationsphase sind, dass die Teilnehmer einen Eindruck davon bekommen, was sonst noch mit den Tinkerforge-Mikrocontrollern möglich ist und welche hilfreichen Funktionalitäten den anderen Teilnehmern für Hausautomation eingefallen sind. Außerdem soll der Fokus der Teilnehmer weg von ihrem Projekt hin zum Gesamtüberblick des Kontextes „Wir automatisieren ein Haus“ gelenkt werden.

Je nach Zeitmanagement, das in den nächsten Abschnitten vorgestellt wird, bildet die Ergebnispräsentation den inhaltlichen Abschluss des InfoSphere-Moduls. Da die Präsentation - wie erläutert - als Ausblick oder Überblick über das Themengebiet verstanden werden kann, dient es gleichzeitig als Abrundung des Gelernten und als inhaltlicher Kreischluss an den Beginn des InfoSphere-Moduls, in dem Beispiele für Hausautomationstechniken gesammelt wurden.

Optionale Phase: Zusammenführung der Projektarbeiten

Als alternativer inhaltlicher Abschluss des InfoSphere-Moduls bietet sich, falls das Zeitmanagement es zulässt, eine projektübergreifende Arbeit an, die die einzelnen Projektergebnisse in das Gesamtziel eines vollständigen Hauses integriert. Arbeitsziel dieser optionalen Phase ist ein zentrales Monitoring-System für die einzelnen Projektarbeiten.

Die einzelnen Projektgruppen tragen ihre Hardware-Ergebnisse auf einem großen Tisch zusammen und bauen dadurch den Hausgrundriss auf. Die Ergebnis-Quellcodes werden, angeleitet durch ein weiteres Arbeitsblatt, um die für das Monitoring notwendige Funktionalität ergänzt. Da in diesem Zusammenhang die Steuerung aller gemeinsamen Projekte nur noch von einem Laptop aus zentral übernommen wird, müssen die Mikrocontroller aller Gruppen untereinander verbunden werden. Außerdem müssen sich die SuS an Absprachen mit anderen Teilnehmern bezüglich Software-Schnittstellen halten. Die Teilnehmer üben dadurch die teamübergreifende Zusammenarbeit und erfüllen damit einen weiteren Aspekt der didaktischen Zielsetzung.

Spätestens nach dieser Phase endet das Schülerlabor-Modul inhaltlich. Dieser Abschluss bietet die gleichen Vorteile wie ein Abschluss nach der Präsentationsphase, allerdings deutlich intensiver, da die SuS sich noch einmal praktisch mit dem Gesamtkontext auseinandersetzen. Durch die Bearbeitung der optionalen Phase kann der gesteigerte Erfahrungserwerb in Teamarbeit als zusätzlicher Vorteil gelten.



Zeitmanagement

Zuletzt ist zu beantworten, in welchem zeitlichen Rahmen das InfoSphere-Modul durchgeführt wird und welche Phase wie viel Zeit zugeteilt bekommt.

Die Gesamtdauer für den inhaltlichen Teil des Moduls ist auf fünf Zeitstunden festgelegt worden. Für eine ergiebige Bearbeitung des Exkursionsthemas wird viel Zeit benötigt, andererseits findet das Schülerlabor oft nachmittags an Schultagen statt, sodass eine übermäßige Belastung der Teilnehmer zu vermeiden ist.

Zunächst zur Zeiteinteilung für die einzelnen Modulphasen: Die Grobe Zeitverteilung sieht in etwa zwei Stunden für die Einführung, zwei Stunden für den Hauptteil in der Projektarbeitsphase und eine Stunde für die Optionalphase vor. Dabei wird für die komplette theoretische Einführung, also dem ersten Inhaltsblock, eine Viertelstunde eingeplant.

Da sich mit wechselnden Teilnehmergruppen auch die Leistungsfähigkeit der Teilnehmer ändert, ist eine exakte Zeitplanung nicht möglich. Dementsprechend sind die hier genannten Werte eher als grobe Richtlinie zu verstehen. Es ist denkbar, dass die Optionalphase aus Zeitmangel nicht mehr umgesetzt werden kann und die Exkursion damit nach der Präsentationsphase endet.

Bezüglich der erwähnten Bonusaufgabe in der Projektphase ist deshalb zu ergänzen, dass diese sinnvollerweise nur dann vergeben werden kann, wenn die Betreuer zu diesem Zeitpunkt die Durchführung der Optionalphase für wahrscheinlich halten. Sollte diese Bonusaufgabe nicht vergeben werden können, so sind die Teilnehmer zu weiteren Arbeiten an ihrem Projekt anzuregen.

Da von den Teilnehmern keine ununterbrochene konzentrierte Arbeit über den Gesamtzeitraum von fünf Stunden zu erwarten ist, sind Pausen anzusetzen. Das Schülerlabor sieht gewöhnlicherweise vor, dass sich die Teilnehmer frei bewegen und den individuellen Bedürfnissen entsprechend Ess- und Trinkpausen einlegen dürfen. Demzufolge ergibt sich möglicherweise nicht der Bedarf für gesammelte Pausen. Es obliegt den Betreuern, den Bedarf für Pausen zu erkennen und die Zeitsetzung vorzunehmen.

3.4.2. Analyse der Leitpräsentation

Bevor die Leitpräsentation im Detail analysiert wird, zuerst eine Begründung für die Medienwahl: Eine folienbasierte Präsentation bietet sich für das Schülerlabor an, weil auf diesem Wege Tafelbilder vorbereitet werden können, was während der Exkursion viel Zeit spart. Außerdem sind die optischen Gestaltungsmöglichkeiten um ein Vielfaches



größer, sodass den Teilnehmern die Inhalte ansprechender vorgestellt werden können. Da dem Schülerlabor ein Smartboard zur Verfügung steht, sind die Tafelbilder gegebenenfalls live erweiterbar. Nachdem der Gesamtverlauf und der Zweck der Präsentation im vergangenen Kapitel erläutert wurden, folgen an dieser Stelle die Begründungen für die Inhalte und Konzepte einzelner Präsentationsfolien. Da die Analyse nicht für jede Folie neue Ergebnisse liefert, wird nur ein Teil der gesamten Folien betrachtet. Die vollständige Präsentation befindet sich im Anhang unter dem Abschnitt B.

Theoretische Einführung

Zu Beginn ist eine Gesprächsanregung zum Themenbereich Hausautomation eingefügt. Sie dient als Einstieg in den Kontext. Das mit dieser Folie verfolgte Ziel ist, die Teilnehmer von Beginn der Präsentation an sofort mit einzubeziehen, um eine reine Passivität der SuS zu vermeiden und ihre Aufmerksamkeit auf das Geschehen zu lenken. Erst im Anschluss wird der Tagesablauf vorgestellt, der den Schülern vorab einen Überblick über den Tagesverlauf gibt und ihnen damit eine Orientierung im Geschehen ermöglicht.

Danach wird die Eingangs-Fragestellung wieder aufgegriffen und einige typische Hausautomations-Beispiele gezeigt. Spätestens danach sollte den Teilnehmern eine grundlegende Idee davon vermittelt worden sein, was Hausautomation ist und mit welchen Funktionalitäten sich typischerweise beschäftigt wird.

Dem Gesamtverlauf entsprechend werden die SuS anschließend in den Themenbereich Mikrocontroller eingeführt. Ohne zuerst auf Mikrocontroller im Allgemeinen einzugehen, wird direkt das Tinkerforge System als das verwendete vorgestellt. Diese Reihenfolge soll den Teilnehmern helfen, sich direkt auf ein konkretes Produkt zu konzentrieren, damit alles folgende allgemeine Mikrocontrollerwissen sofort auf ein konkretes Beispiel übertragen und dadurch leichter verstanden werden kann. Anschließend werden sukzessiv allgemeine Grundlagen und Grundbe-

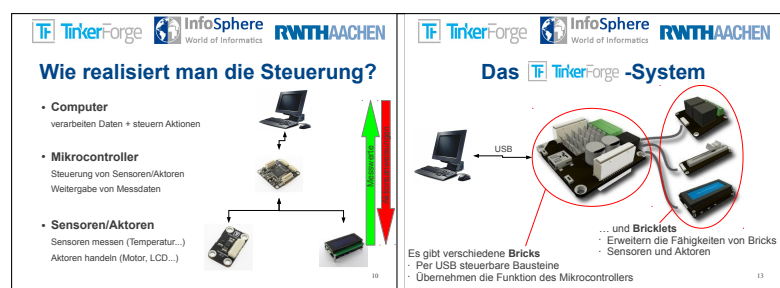


Abbildung 3.2.: Folie 10

Abbildung 3.3.: Folie 13



griffe rund um Mikrocontroller eingeführt (Abbildung 3.2). Dabei werden neben dem schematischen Verbindungsplan die Begrifflichkeiten Mikrocontroller, Sensor und Aktor erläutert. Außerdem wird der Datenfluss von Steuer- und Messdaten verdeutlicht. Auf den Folien sind damit alle für die weiteren Arbeiten wichtigen Grundlagen enthalten. Die Betreuer können, nach eigenem Ermessen die Vorkenntnisse und Einwände der Teilnehmer berücksichtigend, auf weitere Details und Fachbegriffe eingehen. Zusätzlich können sie beispielsweise gemeinsam mit den SuS den Unterschied zwischen Computer und Mikrocontroller klären. Inhaltlicher Leitfaden und Orientierung dafür ist das Betreuermerkblatt Mikrocontroller (vgl. Abbildung 3.16), das in Kapitel 3.4.4 vorgestellt wird. Prinzipiell ist dieses inhaltlich tiefgehende Gespräch erstrebenswert, da es den Wissenserwerb der Teilnehmer vergrößert. Es sollte deshalb ausdrücklich nur dann von den Betreuern übersprungen werden, wenn sie die Teilnehmer damit überfordert sehen. Die weitergehenden Inhalte vom Betreuermerkblatt Mikrocontroller werden nicht mit in die Präsentation aufgenommen. Sollten diese Inhalte übersprungen werden, soll den Teilnehmern durch das offensichtliche Überblättern der Folien kein Gefühl der Benachteiligung entstehen. Außerdem ermöglicht das Auslassen die individuelle Anpassung des Gesprächs an die Zielgruppe hinsichtlich Schwierigkeitsgrad, Geschwindigkeit und Umfang.

Bis Folie 13 (Abbildung 3.3) werden die Tinkerforge-eigenen Produktbezeichnungen Brick und Bricklet eingeführt. Diese firmenspezifische Namensgebung müssen die SuS kennen, da sie unter anderem die Original-Dokumentation der Firmenwebseite verwenden und sich entsprechend orientieren können müssen.

Praktische Einführung

Damit ist der erste Inhaltsblock der Exkursion (vgl. Kapitel 3.4.1) abgeschlossen und die praktische Einführung beginnt. Nach der Einführung in den Umgang mit den Mikrocontrollern werden die notwendigen Java-Kenntnisse für die Verwendung der Tinkerforge-Bausteine mit eigenem Quellcode vermittelt. Das Prinzip des folgenden Präsentationsabschnittes ist die schrittweise Einführung von Quellcode-Grundstruktur, Einbinden der Mikrocontroller-Bibliotheken, Aufbau der Verbindung zwischen Programm und Mikrocontroller, Listnern, Exceptions und try-catch-Blöcken.

Dabei ist dieser Abschnitt in zwei Unterabschnitte aufgeteilt. Der erste beinhaltet die Erarbeitung des Quellcode-Grundgerüsts für die Mikrocontroller-Verwendung, der zweite Abschnitt beschäftigt sich mit dem eigentlichen Programm und den unbekanntem Java-Konzepten. Der erste Abschnitt umfasst die Folien 17 bis 37 (Abbildung



3.4), der zweite die Folien 40 bis 54 (Abbildung 3.5). Dazwischen wird auf die Checkliste hingewiesen, auf der die Schüler die notwendigen Arbeitsschritte zum Einbinden eines Mikrocontrollers in den Quellcode mit eigenen Worten eintragen (vgl. dazu auch Kapitel 3.4.3).

Das Schema der Folien funktioniert wie folgt: Zuerst wird genannt,

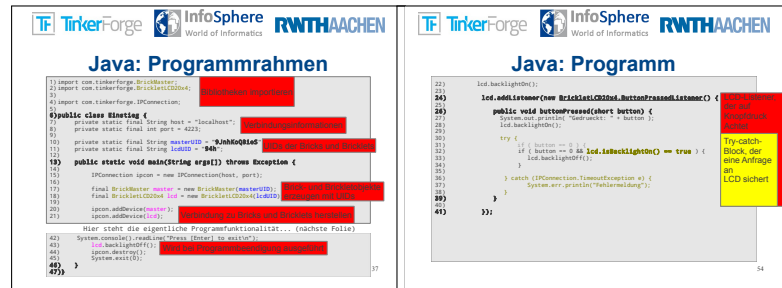


Abbildung 3.4.: Folie 37

Abbildung 3.5.: Folie 54

welche nächste Funktionalität implementiert wird. Diese wird dann im Quellcode umgesetzt und mit Kommentarblöcken versehen, die die Orientierung im Quelltext vereinfachen. Bedeutende Schlagwörter im Quelltext werden farbig markiert, damit der Zusammenhang in der Namensgebung und die Entwicklung einzelner Objekte deutlich wird. Die Folien 37 (Abbildung 3.4) und 54 (Abbildung 3.5) zeigen, wie der entwickelte Quellcode mit den erwähnten Kommentaren und Markierungen aussieht. Der Aufbau dieser Quellcodes geschieht nicht zeilenweise, da dies die logischen Zusammenhänge zwischen den Codefragmenten nicht herausstellen und darunter die Verständlichkeit deutlich leiden würden. Vielmehr werden die entsprechenden logisch zusammenhängenden Codebestandteile nacheinander eingeblendet.

Für die Durchführung der Quellcode-Präsentation ist es von entscheidender Bedeutung für das Verständnis der SuS, dass sie einbezogen werden und keine reine Ergebnis-Präsentation stattfindet. Den SuS müssen die Parallelen zu ihren aus der Schule bekannten Quelltexten und Programmierprojekten bewusst gemacht werden, damit die Hemmschwelle vor dem Umgang mit dem scheinbar neuen Code gesenkt wird. Außerdem muss sichergestellt sein, dass alle Teilnehmer die neu eingeführten Java-Eigenschaften verstehen und nachvollziehen können. Das geschieht, indem die SuS eigene Programmervorschläge machen und die neu kennengelernten Inhalte mit eigenen Worten wiedergeben und erläutern.

Zum Inhalt der vorgestellten Beispiele ist anzumerken, dass sie nach mehreren Kriterien ausgewählt wurden: Einerseits sollten alle vorkommenden, den SuS unbekanntem Java-Eigenschaften vorgestellt werden. Andererseits musste die Einfachheit der Beispiele zugunsten einer besseren Verständlichkeit berücksichtigt werden. Dementsprechend wurde ein Mikrocontroller (Master-Brick) und ein Aktor (LCD-Bricklet) in das Beispiel eingebaut. Beide Bausteine finden im weiteren



Verlauf eine hohe Anwendungsrate, was sie für die exemplarische Vorstellung geeignet macht. Außerdem bietet das LCD-Bricklet alle Fähigkeiten, um die Prinzipien aller benötigten Java-Eigenschaften an ihm zu demonstrieren.

Die implementierten Funktionsbeispiele sind so gewählt, dass sie einer typischen Anwendung in der Projektphase entsprechen, ihr Umfang aber gleichzeitig die Übersichtlichkeit der Beispiele gewährleistet.

Um die Notwendigkeit für den Einsatz eines try-catch-Blockes besser zu verdeutlichen, wird auf Folie 51 (auch zu sehen auf Folie 54, Abbildung 3.5) in den Zeilen 31 bis 34 explizit ein Beispiel mit einer versteckten Bricklet-Abfrage eingebaut. Das soll die SuS für versteckte Fallstricke sensibilisieren.

Im Anschluss an die Quellcode-Entwicklung werden verbleibende Fragen der SuS im Plenum geklärt. Das bietet den Vorteil, dass sofort alle anderen Teilnehmer ebenfalls die Antwort erhalten und nicht gestellte, gleiche Fragen mitbeantwortet werden. Außerdem stellen die wiederholten Erklärungen eine Vertiefung des Erlernen dar, was die Inhalte etwas festigen kann. Die SuS erhalten das am Smartboard entwickelten Quellcode-Grundgerüst, also abzüglich des Funktionalitätsbeispiels, auch in ausgedruckter und digitaler Form. Er stellt die Grundlage für alle weiteren Arbeiten dar, die somit direkt an das Gelernte anknüpfen. Danach wird in das praktische Einführungsprojekt übergeleitet. Nach der darauf folgenden Lösungsbesprechung endet der zweite Inhaltsblock, es wird in die Hauptarbeitsphase übergeleitet, in der die Projekte bearbeitet werden.

Projektarbeitsphase

Zuerst wird den Teilnehmern die Gestaltung der Projektphase erläutert, bevor die einzelnen zu bearbeitenden Projekte auf Folie 62 (Abbildung 3.6) vorgestellt und durch die Betreuer kurz beschrieben werden. Abschließend können sich die Teilnehmer entsprechend ihren Neigungen auf die sechs Arbeitsgruppen verteilen (Folie 63, Abbildung 3.7). Die Teilnehmernamen werden von den Betreuern in die sechs Kästchen eingetragen, sodass die Zuordnung für jeden ersichtlich ist. Den einzelnen Projekten sind Smileys in den Farben Grün und Gelb zugeordnet, die den Schwierigkeitsgrad der Projekte beschreiben. Auf diese Weise können die SuS ihre Selbsteinschätzung bezüglich ihrer Leistungsfähigkeit mit in ihre Projektwahl einfließen lassen (vgl. Folie 63, Abbildung 3.7). Eine genauere Beschreibung des Schwierigkeitsgrads ist nicht vorgenommen worden, da dieser sehr schwer objektiv messbar ist und feinere Unterscheidungen kaum zu rechtfertigen wären. Erst nach dem Ende der anschließenden Arbeits-



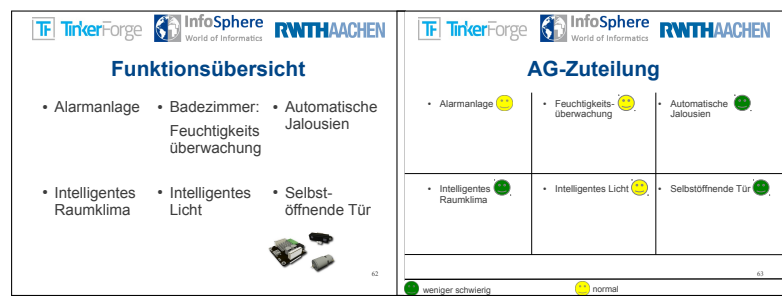


Abbildung 3.6.: Folie 62

Abbildung 3.7.: Folie 63

phase wird die Präsentation wieder als roter Faden aufgenommen, sie leitet dann die Präsentationsphase ein.

Optionale Phase

Daran schließt sich nur noch die optionale Phase an, deren Arbeitsablauf, ergänzend zu den Anleitungen der Betreuer, erklärt wird. Zuletzt schließt die Präsentation die optionale Phase und damit auch den inhaltlichen Teil der Exkursion und leitet über in das obligatorische Abschlussprozedere des Schülerlabors.

Foliengestaltung

Die allgemeine Gestaltung der Folien soll der didaktischen Zielsetzung (vgl. 1.2) in mehrfacher Weise gerecht werden.

Die Gestaltung der einzelnen Folien ist ein Kompromiss aus schlichter Sachlichkeit und motivierenden Elementen wie beispielsweise kleinen Bildern (vgl. Folie 62, Abbildung 3.6). Quelltext-Kommentare und Einblendeffekte bei der Quelltext-Entwicklung sollen optisch das Verständnis für den sukzessiven Aufbau des Codes erleichtern.

Die Folien, die keine fachlichen Inhalte vermitteln, sondern der Organisation dienen, sind besonders karg gestaltet. Dadurch soll ein Zurücktreten der Folien in den Hintergrund zugunsten einer Fokussierung der Teilnehmer auf die Erklärungen der Betreuer gefördert werden.



3.4.3. Teilnehmermaterial

Die Arbeitsmaterialien bekommen von den SuS viel Aufmerksamkeit. Demzufolge nimmt ihre Gestaltung hinsichtlich ihrer didaktischen Bedeutung einen hohen Stellenwert ein. Diese wird in den nächsten Abschnitten ausführlich erläutert.

Die Materialien werden erst bei entsprechendem Bedarf an die Teilnehmer ausgehändigt und liegen in ausgedruckter Form vor. Je nach erwarteter Nutzungshäufigkeit werden sie in mehrfacher Ausführung ausgeteilt, beispielsweise die Kurzdokumentation, damit ein paralleles Arbeiten innerhalb der Gruppe erleichtert wird.

Einführungsprojekt

Die Unterlagen zum praktischen Einführungsprojekt bestehen aus mehreren Teilen. Kernmaterial ist der „Leitfaden Einstiegsprojekt“. Er wird ergänzt durch eine Tinkerforge-Kurzdokumentation, einen Leitfaden als Nachschlagewerk für das Anschließen der Mikrocontroller, den in der Präsentation erarbeiteten Grundlagen-Quelltext und, in digitaler Form, durch eine Moodle [5] Webseite zur Bereitstellung des Quellcodes sowie der Tinkerforge-Online-Dokumentation [6].

Das erste Arbeitsmaterial, das die Teilnehmer erhalten, ist die Checkliste für das Anschließen der Bausteine (siehe Anhang C). Sie besteht aus zwei Abschnitten mit vorgedruckten Zeilen, sodass die SuS nur noch ihre Stichpunkte eintragen müssen. Die Anzahl der erwarteten Stichpunkte ist vorgegeben, was das korrekte Ausfüllen unterstützt. Beim Ausfüllen unterstützt werden die SuS während der Präsentation von den Betreuern, die die einzelnen Arbeitsschritte in geeigneter Form vorstellen.

Alle Arbeitsmaterialien werden beim Austeilen erläutert, um den SuS einen Überblick über das Material zu geben und Orientierungsprobleme zu vermeiden.


Die folgenden Unterlagen beziehen sich bereits auf die Einführungs-Arbeitsphase und werden mit Beginn dieser vollständig ausgehändigt.

Zuerst zum Leitfaden Einführungsprojekt (zu finden in Anhang C.1): Er wird, um Verwirrungen zu vermeiden und die Handhabbarkeit zu steigern, in gebundener Form ausgeteilt. Das Deckblatt (Abbildung 3.8) ist eine bebilderte Auflistung der Hardware, die die Teilnehmer erhalten müssen. Die Hardware ist bereits in der vorherigen Arbeitsphase ausgehändigt worden, damit die SuS die einzelnen Arbeitsschritte zum







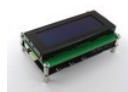




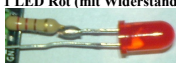

Anschließen der Mikrocontroller nachmachen konnten. Vor der weiteren Bearbeitung sind die Teilnehmer durch die Auflistung in der Lage, die Vollständigkeit ihrer Hardware zu überprüfen und gegebenenfalls zu reklamieren. Die Bebilderung hilft, die einzelnen Bausteine den entsprechenden Namensgebungen zuzuordnen. Insgesamt soll das Deckblatt unnötige Verzögerungen im späteren Arbeiten durch fehlende Hardware unterbinden. Die zweite Seite beinhaltet ausschließlich

Schülerlabor Informatik
Einstiegsprojekt
Leitfaden



Vorbereitung:

1) Sichtet euer Material! Ihr braucht:

<p>2 Laptops</p> 	<p>1 Master Brick</p> 	<p>1 USB Kabel</p> 
<p>4 Bricklet-Kabel</p> 	<p>1 LCD-Bricklet</p> 	<p>1 Linear-Poti Bricklet</p> 
<p>1 Schraubendreher</p> 	<p>1 Temperature-Bricklet</p> 	<p>1 IO4-Bricklet</p> 
<p>1 LED Rot (mit Widerstand)</p> 	<p>1 LED Grün (mit Widerstand)</p> 	

Falls etwas fehlt: Meldet euch bei den Betreuern!

Hinweis: Sollten Bausteine nicht korrekt funktionieren, bekommen sie meist zu wenig Strom. Versucht es mit einem anderem USB-Port des Laptops.

1

Abbildung 3.8.: Leitfaden Einstiegsprojekt Seite 1

die Anweisung zum Herunterladen des digitalen Quellcodes und bereitet damit das eigentliche Arbeiten vor. Zur Erinnerung: Die SuS arbeiten in dieser Phase in Vierergruppen, wobei sie sich jeweils zu zweit ein Laptop teilen und jedes der Laptop-Pärchen unterschiedliche Aufgabenstränge bearbeitet. Beide Paare laden den Quellcode zur Bearbeitung herunter. Die SuS sollen das Quellcode-Grundgerüst,



welches zuvor gemeinsam am Smartboard entwickelt wurde, verwenden, damit die Schreibearbeit für das Grundgerüst wegfällt und mehr Zeit für die Beschäftigung mit der eigentlichen Funktionalität übrig bleibt. Der Lernfortschritt beim Schreiben des Grundgerüsts ist gering genug, um ein Entfallen dieses Arbeitsschrittes zu rechtfertigen.

Anschließend trennen sich die Bearbeitungspfade der beiden Paare. Auf Basis des gleichen Programmgrundgerüsts implementieren sie zwei unterschiedliche Funktionalitäten. Ziel dieser Vorgehensweise ist, der Vierergruppe innerhalb kurzer Zeit viele Aspekte der Mikrocontrollerprogrammierung mit Java nahezubringen und dabei die bewusste Teamarbeit zu fördern. Teamarbeits-Kompetenz wird durch den Zwang zur Kommunikation mit dem jeweiligen anderen Paar gefördert.

Die Seiten Vier und Fünf leiten die beiden Paare durch ihre jeweilige Aufgabe. Zuerst wird das Arbeitsziel genannt, damit den SuS der Sinn der nächsten Arbeitsschritte bewusst ist. Danach werden einzelne Arbeitsschritte angegeben, die im Gesamtergebnis zur Lösung des genannten Aufgabenziels führen. Diese schrittweise Anleitung unterstützt die SuS in einer strukturierten Problemlösung und nimmt ihnen die Planung des Lösungsweges ab. So können sie sich auf die neuen Java-Inhalte konzentrieren. Ohne diese schrittweise Bearbeitung ist eine Überforderung der SuS zu befürchten. Positiver Nebeneffekt ist, dass die SuS mit jedem erfolgreich abgeschlossenen Teilschritt ein Erfolgserlebnis haben, das zur Weiterarbeit motiviert und den Spaßfaktor hoch hält.

Die letzten beiden Arbeitsanweisungen beinhalten das Zusammenfügen der beiden Paar-Ergebnisse zu einem Gesamtgruppenergebnis. Hier ist ebenfalls wieder eine enge Kooperation unter den Arbeitspaaren notwendig, was wieder die Teamfähigkeit stärkt. Außerdem lernen die Paare den Quellcode des jeweils anderen kennen. Die anschließende Besprechung einer Beispiellösung ergänzt dies, sodass im Anschluss alle Teilnehmer alle bearbeiteten Java-Eigenschaften kennen gelernt haben. Der Umfang der kennenzulernenden Java-Aspekte macht eine intensive Bearbeitung aller Aspekte durch alle Teilnehmer zeitlich unmöglich. Die Auswahl der von den Paaren zu bearbeitenden Aufgaben stellt die häufigsten in der anschließenden Projektphase auftretenden Implementierungsszenarios dar. Nachteiliger Aspekt dieser Aufgaben-Zusammenstellung ist die fehlende Einbettung in den Kontext Hausautomation. Beides zusammen ist jedoch nicht in angemessener Bearbeitungszeit möglich.

Die letzte Seite des Einstiegs-Leitfadens beinhaltet ein Java-Merkblatt, auf dem alle im Folgenden benötigten Java-Funktionen gesammelt sind und jeweils knapp erläutert werden (vgl. Abbildung 3.9). Dieses Merkblatt ist für die Teilnehmer die erste Anlaufstelle für Fragen bezüglich der neuen Java-Konzepte. Die Erklärungen sind fachlich unvollständig




und haben den Charakter von Faustregeln. Das soll eine möglichst gute Verständlichkeit bei sehr knappen Umfang ermöglichen. Inhaltlich geht es über das hinaus, was für das Einstiegsprojekt benötigt wird. Es dient also auch für die folgende Projektphase als Merkblatt.

Schülerlabor Informatik

Einstiegsprojekt

Leitfaden



Merkblatt Java

Listener:
 Ein Listener überprüft in vorgegebenem Rhythmus einen Sensorwert oder, ob ein bestimmtes Ereignis (Bsp.: Knopf wurde gedrückt; Festgelegter Temperaturwert wurde überschritten) eintritt. Tritt das Ereignis ein, so ruft er eine Methode auf, der er den Wert des Ereignisses übergibt.

```

// Messrhythmus vorgeben mit 1000 ms
temp.setTemperatureCallbackPeriod(1000);

// Listener, der jede Sekunde Temperatur ausgibt
temp.addListener(new BrickletTemperature.TemperatureListener() {

    public void temperature(short temperature) {
        System.out.println("Temperature: " + temperature/100.0 + " °C");
    }
});

```

Try-Catch-Block:
 Stellt man eine Anfrage an einen Baustein (Bsp.: Wie ist die Temperatur? Ist das LCD-Licht an?), so muss diese innerhalb eines try-catch-Blockes stehen:

```

try {

    if ( lcd.isBacklightOn() == true ) {
        // Anfrage an Baustein
        lcd.backlightOff();
    }

} catch (IPConnection.TimeoutException e) {

    System.err.println("Fehlermeldung");

}

```

Variablen konvertieren:
 Der Datentyp short ist zu verwenden wie int, es passen aber nur kleinere Zahlen hinein.

int y zu short: short x = (short) y;
String y zu short: short x = Short.valueOf(y);
Short y zu String: String x = y.toString();
 Für die Folgenden müsst ihr java.lang.Integer importieren: import java.lang.Integer;
int y zu String: String x = toString(y);
String y zu int: Integer.parseInt(y);

Programm warten lassen:
 Diese Zeile lässt das Programm 5 Sekunden (angegeben in Millisekunden) warten:
 Thread.sleep(5000);

6

Abbildung 3.9.: Leitfaden Einstiegsprojekt Seite 6

Zusätzlich zum Einstiegsleitfaden erhalten die Teilnehmer ein Nachschlagewerk für den Einstieg in Tinkerforge. Ein vollständiges Exemplar befindet sich im Anhang unter Abschnitt C. Dieser ist in keine Aufgabe eingebunden und wird nur gebraucht, falls die SuS im Nachhinein Fragen oder Erinnerungslücken zur Vorführung der Betreuer haben. Sie soll verhindern, dass Teilnehmer zurückbleiben und ermöglicht eine Wiederholung der einzelnen Schritte in individuellem Tempo. Die zuvor mündlich vorgestellte Einführung ist in schriftlicher Form mit Bebilderung inhaltlich identisch.



In diesem Nachschlagewerk wird besonders intensiv auf die Verwendung des IO4-Bricklets eingegangen, da die Konfiguration dieses Bausteins im Verhältnis besonders aufwändig ist und damit eine potentielle Schwierigkeit darstellt. Das korrekte Anschließen der LEDs wird mit einem entsprechenden Bild angeleitet.

Die Tinkerforge-Kurzdokumentation ist eine Auszug aus der offiziellen Online-Dokumentation [6]. In ihr wurden die häufigsten gebrauchten Abschnitte der Dokumentation zusammengetragen. Die Sammlung liegt den SuS in ausgedruckter Form vor, damit ein schnelles Nachschlagen möglich ist. Ausführlichere Passagen der Dokumentation sind ausgelassen, stattdessen wird auf die Online-Dokumentation verwiesen. In Testdurchläufen hat sich die Kurzdokumentation auch deshalb bewährt, weil sie zeitgleichen Einblick in die Dokumentation und den Quellcode der Teilnehmer ermöglicht. Die Kurzdokumentation befindet sich im Ahnhang unter dem Abschnitt C.1.

Die Gründe dafür, dass den Teilnehmern Dokumentationen zur Verfügung gestellt werden und nicht jede gebrauchte Tinkerforge- bzw. Java-Funktionalität didaktisch aufbereitet vermittelt wird, liegen in der Zielsetzung der Exkursion: Eine detaillierte didaktische Aufbereitung widerspräche dem Ziel, die Lernatmosphäre des InfoSphere-Moduls unterschiedlich zu der Lernatmosphäre von Schulunterricht zu gestalten. Weiterhin wäre die Zeit für eine detaillierte Anleitung der einzelnen Arbeitsschritte zu knapp, auf diesem Weg wird dafür die Fähigkeit zum selbstständigen Arbeiten gefördert. Zusätzlich gehört es zur Zielsetzung, den Teilnehmern einen Eindruck von professionellem Arbeiten im Kontext zu ermöglichen. Da die Arbeit mit unbekanntem Dokumentationen im Berufsalltag eines Informatikers und im Umgang mit Mikrocontrollern zum Standard gehört, drängt sich das Einführen dieser Arbeitsweise auf. Es ist zu berücksichtigen, dass die Teilnehmer in dieser Methodik unerfahren sind und entsprechende Unterstützung benötigen. Daher gehört es zu den Aufgaben der Exkursionsbetreuer, den SuS gegebenenfalls Hilfestellung und Anleitung zu geben. Wie genau das umzusetzen ist lässt sich aufgrund unterschiedlicher Teilnehmergruppen nicht pauschal beantworten und muss in den Entscheidungsspielraum der Betreuer verlegt werden.

Als Verteilerzentrum für digitale Inhalte fügt sich die Moodle-Webseite des InfoSphere-Moduls in die Reihe der Teilnehmermaterialien ein. Die Seite erfüllt zwei Funktionen: Zum einen bietet sie den Grundlagencode zum Download an, zum anderen stellt sie den Teilnehmern die Funktionalität zum Quellcodeaustausch zur Verfügung. Darüber hinaus soll das Moodle nicht in den Fokus der SuS rücken, um sie nicht abzulenken. Demzufolge ist die Webseite karg und übersichtlich gestaltet (vgl. Abbildung 3.10).



The screenshot shows a Moodle course page with the following content:

- Startseite » Meine Kurse » Hausautomatisierung**
- Themen dieses Kurses**
- Hier findet ihr die Dateien für die Projekte.
- 1 Einstieg**
Hier findet ihr alle Dateien, die ihr für die Einstiegsrunde braucht.
Einstieg.zip
- 2 Einstieg Dateiaustausch**
Hierüber könnt ihr eure Dateien austauschen.
Einstieg Dateiaustausch
- 3 Projekt Alarmanlage**
Diese Dateien werden nur von der Gruppe benötigt, die sich mit der Alarmanlage beschäftigt. Die anderen laden diese Dateien bitte nicht runter.
Alarmanlage.zip
- 4 Projekt Feuchtigkeitsmesser**
Diese Dateien werden nur von der Gruppe benötigt, die sich mit der Feuchtigkeitsmessung beschäftigt. Die anderen laden diese Dateien bitte nicht runter.
Feuchtigkeitsmesser.zip
- 5 Projekt Jalousiesteuerung**
Diese Dateien werden nur von der Gruppe benötigt, die sich mit der Jalousiesteuerung beschäftigt. Die anderen laden diese Dateien bitte nicht runter.
Jalousiesteuerung.zip
- 6 Projekt Klimasteuerung**
Diese Dateien werden nur von der Gruppe benötigt, die sich mit der Klimasteuerung beschäftigt. Die anderen laden diese Dateien bitte nicht runter.
Klimasteuerung.zip
- 7 Projekt Lichtsteuerung**
Diese Dateien werden nur von der Gruppe benötigt, die sich mit der Lichtsteuerung beschäftigt. Die anderen laden diese Dateien bitte nicht runter.
Lichtsteuerung.zip
- 8 Projekt Türöffner**
Diese Dateien werden nur von der Gruppe benötigt, die sich mit der automatisch öffnenden Tür beschäftigt. Die anderen laden diese Dateien bitte nicht runter.
Tueroeffner.zip
- 9 Bonus: Monitoring**
Diese Dateien werden nur von der Gruppe benötigt, die sich mit dem Monitoring / der Zentralsteuerung beschäftigt. Die anderen laden diese Dateien bitte nicht runter.
Monitoring.zip

Abbildung 3.10.: Moodle-Kurs

Projektarbeitsphase

In der Projektarbeitsphase erhalten die einzelnen Gruppen weitere Hardware und ein Aufgabenblatt. Da die Blätter für alle Projekte schematisch gleich und inhaltlich nur an das jeweilige Projekt angepasst sind, ist die Analyse eines einzelnen Beispielblattes stellvertretend für alle weiteren ausreichend. Die gesamte Sammlung ist im Anhang unter Abschnitt C.2 zu finden.


Das Schema der Projektarbeitsblätter ergibt sich aus einer bebilderten Liste der zusätzlich zu erhaltenden Hardware, Arbeitsanweisungen, einen Konzeptvorschlag bzw. eine Anregung für die Implementierung und, je nach Projekt, gegebenenfalls einer Anleitung bzw. Erläuterung zur Verwendung der verteilten Hardware. Vorgestellt wird das am Beispiel des Arbeitsblattes zum Projekt „Alarmanlage“ (Abbildungen 3.11 und 3.12), alle anderen Arbeitsblätter sind im Aufbau gleich. Im oberen Teil der Blätter befindet sich die Hardwareliste. Sie dient



Schülerlabor Informatik

Projekt Alarmanlage

Leitfaden und Arbeitsanweisungen



InfoSphere
World of Informatics

Vorbereitung:

1) Sichtet euer Material! Zusätzlich zu den Bausteinen aus dem Einstiegsprojekt erhaltet ihr:

Türmodell	Fenstermodell
------------------	----------------------

Beide Modelle bekommt ihr mit einer Einweisung von den Betreuern.

2) Lasst euch von einem Betreuer in die Handhabung der Modelle unterweisen. Geht sorgsam mit den Schaltern und Kabeln um, sie sind sehr empfindlich.

3) Ladet euch von der schon bekannten Moodle-Seite aus dem Abschnitt „Projekt Alarmanlage“ alle Dateien herunter. Entpackt sie in einem dafür neu erstellten Ordner. Die darin enthaltene Datei „Alarmanlage.java“ könnt ihr für euer Projekt verwenden.

4) Nun könnt ihr mit der Arbeit beginnen. Die Arbeitsschritte kennt ihr aus dem Einstiegsprojekt.

Ihr könnt euch ein völlig eigenes Konzept für eine Alarmanlage überlegen. Denkt euch aus, was ihr für praktisch und nützlich haltet und was mit den gegebenen Materialien (Bausteinen und Modellen) erreichbar ist. Als Anregung findet ihr unten einen Konzeptvorschlag für eine „inverse Alarmanlage“. Es ist aber nur eine Idee.

Inverse Alarmanlage

Überwacht, ob beim Verlassen des Hauses alle Fenster/Türen richtig geschlossen sind.

Konzept:

Mit Hilfe von Schaltern soll überprüft werden, ob alle Fenster/Türen korrekt verschlossen sind. Ist ein Schalter nicht gedrückt, dann ist das Fenster offen. Wird bei offenem Fenster die Haustüre abgeschlossen (Schalter im Schließmechanismus), dann soll die Alarm-LED leuchten und auf einem Display eine Warnung ausgegeben werden.

Solltet ihr Schwierigkeiten bei der Umsetzung haben, wendet euch an die Betreuer.

1

Abbildung 3.11.: Projektarbeitsblatt Alarmanlage Seite 1

dem gleichen Zweck wie das Deckblatt des Einstiegsprojekts. Von allen Hardware-Bausteinen sind Fotos verwendet, von den selbst gebauten Modellen allerdings nicht. Stattdessen ist ein Kommentar eingefügt, dass die SuS zuerst eine Einweisung in den Umgang mit den Modellen bekommen. Der Grund dafür liegt in der Praxis: Alle kleineren Hardware-Gegenstände bedürfen keiner Erläuterung und können leicht verteilt werden. Sie werden deshalb zuerst ausgegeben. Die großen und unhandlichen Modelle lassen sich nicht einfach umstellen, besonders nicht in einer tendenziell hektischen Situation wie einer Materialvergabe. Deshalb werden die Modelle zum Schluss ausgegeben. In dem Zusammenhang wird den Gruppenmitgliedern der Umgang mit ihnen erklärt. Die Erklärungen der Modelle beinhalten eine Sicherheitseinweisung, die vor allem bei den steuerbaren Steckdosen wichtig ist, und eine Betriebsanleitung.




Der Hardwareliste schließt sich die Arbeitsanweisung zum Runter-

Schülerlabor Informatik

Projekt Alarmanlage

Leitfaden und Arbeitsanweisungen



InfoSphere

World of Informatics

Anleitung für die Verwendung von Schaltern

Das IO4 Bricklet lässt sich auch verwenden um festzustellen, ob ein Schalter gedrückt wird, oder nicht.

Ein Beispiel, wie man das Bricklet für einen Schalter verwenden kann:
 Ein Schalter verbindet zwei Kabel, wenn er geschlossen (gedrückt) ist, dann haben beide Kabel Kontakt (Strom kann fließen), sonst nicht. Eines der beiden Kabel muss am entsprechend anderen Ende mit einem nummerierten Pin auf dem IO-Bricklet verbunden sein, das andere mit einem GND-Pin.
 Die Konfiguration für den entsprechenden (nummerierten) Pin muss sein:

„Direction = Input“
 „Value = Default“

Drückt man nun den Schalter dann wechselt der gemessene Value-Wert zu „Low“. Das bedeutet,

Setup | Master Brick 1.0 | Linear Poti Bricklet 1.0 | IO-4 Bricklet 1.0 | Temperature Bricklet 1.0 | LCD 20x4 Bricklet 1.0

Debounce Period: 100 Save

Pin: 2 Save

Direction: Input

Value: Default Save

Monoflop Time [ms]: 500 Go (-> 1.1.0 needed)

Pin:	0	1	2	3
Value:	High	High	Low	High
Direction:	Output	Output	Input	Input
Config:	High	High	Default	Pull Up
Monoflop Time [ms]:	0	0	0	0

dass der Stromkreis geschlossen bzw. der Schalter betätigt ist. Ein Beispielbild:
 In diesem Beispiel ist Pin 2 für einen Schalter konfiguriert, der Schalter ist im Moment der Aufnahme gedrückt. Pin 3 zeigt die Standard-Konfiguration der Pins.

Button auslesen in Java:

In der ausgedruckten Kurzdokumentation (Seite: IO4) findet ihr als Hilfestellung ein Beispiel, wie man in Java auslesen kann, welcher der ans IO4 angeschlossenen Buttons gedrückt ist.

2

Abbildung 3.12.: Projektarbeitsblatt Alarmanlage Seite 2

laden der Quelltext-Grundlage an. Auch hier soll wieder die Zeit für das Schreiben des Quelltext-Grundgerüsts zu Gunsten einer längeren Beschäftigung mit der Hausautomations-Funktionalität gespart werden.

Im nächsten Schritt wird den SuS bewusst gemacht, dass sie in der inhaltlichen Gestaltung ihrer Arbeit frei sind und der folgende Projektvorschlag nur als Anregung zu verstehen ist. Damit soll die Projektarbeitsphase der didaktischen Zielsetzung, den Teilnehmern möglichst viel gestalterischen Freiraum zu lassen und Kreativität zu fördern (vgl. Kapitel 1.2), gerecht werden.

Der Projektvorschlag skizziert jeweils grob eine mit der gegebenen Hardware realisierbare Funktionalität. Es findet keine detaillierte Beschreibung des Konzeptes statt, damit ausreichend Freiraum für eigene



Ideen der Teilnehmer bleibt. Der Vorschlag wird überhaupt nur deshalb gemacht, damit Arbeitsgruppen mit weniger Phantasie einen Anhaltspunkt für eigene Gedankengänge haben und im Zweifelsfall nicht zu viel Arbeitszeit mit ergebnislosen Ideensammlungen und Planspielen verloren geht.

Zuletzt ist, je nach verteilten Tinkerforge-Produkten, eine Anleitung für den Umgang mit diesen beigelegt. Das ist nur dann notwendig, wenn die Handhabung dieser Bausteine schwierig ist und sie nicht schon bereits erläutert wurde. Am Beispiel der Alarmanlage wird das IO-4 Bricklet für die Verwendung als Taster-Anschluss erläutert. Dieses Szenario wird nicht allen Teilnehmern beigebracht, weil es nur von der Alarmanlagen-Projektgruppe verwendet werden kann. Für die Projekte, die ein motorisiertes Modell beinhalten, ist die Reihenfolge vertauscht. In diesen Fällen ist die Erläuterung direkt nach der Hardwareliste eingefügt. Hintergrund ist, dass die Teilnehmer die dazugehörigen Warnhinweise sofort sehen und beim testweisen Anschließen an den Brick-Viewer beachten können.

Optionalphase


Das Arbeitsmaterial für die Optionalphase unterteilt sich in ein Arbeitsblatt für diejenigen Teilnehmer, die die Bonusaufgabe bearbeitet haben und in ein allgemeines Aufgabenblatt, das alle Teilnehmer erhalten. Das allgemeine Aufgabenblatt beinhaltet die Aufgabenstellung, das Quellcode-Ergebnis der Projektarbeit um die für das Monitoring notwendigen Dateiausgabe-Funktionen zu ergänzen. Zusätzlich bietet das Arbeitsblatt eine Anleitung samt erläuterten Beispiel, wie man in Java Informationen in eine Textdatei schreiben kann (vgl. Abbildung 3.13). Es ist davon auszugehen, dass die Teilnehmer noch nie mit dieser Technik gearbeitet haben. Die Erklärung am Beispiel bietet sich besonders an, da ein geeignetes Beispiel viel Aussagekraft hat. Diese wird verstärkt, indem die entscheidenden Stellen des Beispiels farbig markiert und erläutert sind. Auf diese Weise sollen die SuS schnell in die Lage versetzt werden, das Beispiel für ihre eigenen Zwecke anzupassen.

Das nicht allgemeine Aufgabenblatt ist nur für die SuS, die in der Projektphase die Bonusaufgabe gelöst haben. Es ergänzt das Bonusaufgabenblatt um weitere Arbeitsschritte, um das Ergebnis der Bonusaufgabe kompatibel zur Gesamtsituation zu machen. So wird die für die Bonusaufgabe verwendete Mühe gewürdigt und die Bonusaufgabe sinnvoll in den Gesamtkontext eingebettet, sodass den entsprechenden SuS nicht das Gefühl einer Beschäftigungstherapie vermittelt wird. Sollten keine SuS die Bonusaufgabe bearbeitet haben oder ihr Ergebnis für eine



Schülerlabor Informatik

Zusammenführung (Alg.)
Leitfaden und Arbeitsanweisungen



InfoSphere
World of Informatics

Daten in eine Datei schreiben

Wenn ihr Daten in eine Datei, beispielsweise eine Textdatei (.txt) schreiben wollt, so müsst ihr im entsprechenden Java-Programm zuerst die Bibliothek importieren, die diese Funktionen bereitstellt:

```
import java.io.*;
```

Anschließend könnt ihr folgendes Grundgerüst verwenden.
Angepasst werden muss noch der Dateiname („eureDatei.txt“) in der grünen Zeile. Wenn ihr die Datei an einen bestimmten Ort speichern wollt, könnt ihr vor dem Dateinamen den Pfad zur Datei angeben.
Außerdem angepasst werden muss der Text bzw. die Daten, die ihr in die Datei speichern wollt (blaue Zeile). Der Zeilenumbruch wird durch die im Beispiel rot markierte Zeile realisiert.
Den Rest könnt ihr übernehmen. Wollt ihr mehr als eine Zeile schreiben, könnt ihr einfach mehrere der blauen (und roten) Zeilen nacheinander schreiben.

```
try {
    Writer fw = null;
    fw = new FileWriter( "eureDatei.txt" );
    fw.flush();
    fw.write( "euer Text " + variablenname );
    fw.append( System.getProperty("line.separator") ); // e.g. "\n"
    if ( fw != null ) {
        try {
            fw.close();
        } catch ( IOException e ) {
            e.printStackTrace();
        }
    }
} catch ( IOException e ) {
    System.err.println( "Konnte Datei nicht erstellen" );
}
```

2

Abbildung 3.13.: Zusammenführung Arbeitsblatt Allgemein Seite 2

Weiterverwendung nicht ausreichend fortgeschritten sein, kann stattdessen die Musterlösung verwendet werden.

Mit dem ergänzenden Aufgabenblatt für die Bonusübung wird ein weiteres didaktisches Ziel erfüllt: Den leistungsstärkeren Teilnehmern wird mehr Verantwortung auch gegenüber allen anderen Teilnehmern übertragen, da sie maßgeblich für den Gesamtaufbau des Hausmodells mitverantwortlich sind. Implizit wird damit die Botschaft vermittelt, dass mehr Leistungsfähigkeit auch zu mehr Verantwortung für die Gemeinschaft führt. Somit enthält die Optionalphase ein zusätzliches, nichtfachliches Lernziel.



3.4.4. Betreuermaterial

Den Betreuern des InfoSphere-Moduls wird ebenfalls Material zur Verfügung gestellt, das sie bei der Durchführung der Exkursion unterstützt. Da die Betreuer wechseln und oft keine Zeit für eine intensive Einarbeitung vorhanden ist, müssen die Betreuermaterialien übersichtlich gestaltet sein und eine Vorbereitung auf die Moduldurchführung mit Minimalaufwand ermöglichen. Es sei jedoch gleich zu Beginn angemerkt, dass das Betreuermaterial den Betreuern die Einarbeitung nicht vollständig abnehmen kann. Es bleibt unabdingbar, sich mit dem Umgang mit den Tinkerforge-Bausteinen, dem Quellcode-Beispiel und den Modellen vertraut zu machen.

Den Betreuern stehen sowohl Materialien in gedruckter Form, gebunden, zur Verfügung, als auch digitale Hilfsmittel.

Analoge Materialien

Alle analogen Betreuermaterialien sind im Anhang D zu finden. Zuerst ist der bereits erwähnte und erläuterte Verlaufsplan (Abbildung 3.1) zu nennen. Ihm schließt sich, auf der gleichen Grafik basierend, ein Verlaufsplan für die Materialausgabe an (Abbildung 3.14). Er bietet eine Übersicht, wenn welche Materialien an die Teilnehmer auszugeben sind. Durch die zum Verlaufsplan identische grafische Gestaltung wird die Orientierung vereinfacht.

Weiteres wichtiges Hilfsmittel zur Moduldurchführung ist der Betreuerleitfaden. Er gibt zum einen eine Reihenfolge der einzelnen Arbeitsschritte an (vgl. Betreuerleitfaden Seite 1, Abbildung 3.15). Dabei orientiert er sich an der Präsentation und gibt zu den einzelnen Arbeitsschritten die jeweilige Folie an, zu der diese ausgeführt werden müssen. Diese genaue, relative Zeitangabe erleichtert die zeitliche Orientierung. Die Auflistung der Arbeitsschritte ist nicht detailliert, sie setzt die Kenntnis der Präsentation voraus. Eine feingliedrige und detaillierte Beschreibung ist für eine schnelle Orientierung während der Moduldurchführung zu unhandlich und deshalb nicht anwendbar. Außerdem erklärt sich ein Großteil der Arbeitsschritte und Inhalte, die in der Präsentation vorkommen, von selbst, daher ist eine detailliertere Anleitung gar nicht notwendig.

Die zweite Seite des Betreuerleitfadens beinhaltet eine Aufzählung aller Arbeitsmaterialien sowohl für die Teilnehmer als auch für die Betreuer, damit die Vollständigkeit aller Materialien überprüft werden kann. Außerdem ist jeweils angegeben, wie viele Exemplare an die Teilnehmer ausgegeben werden sollen.



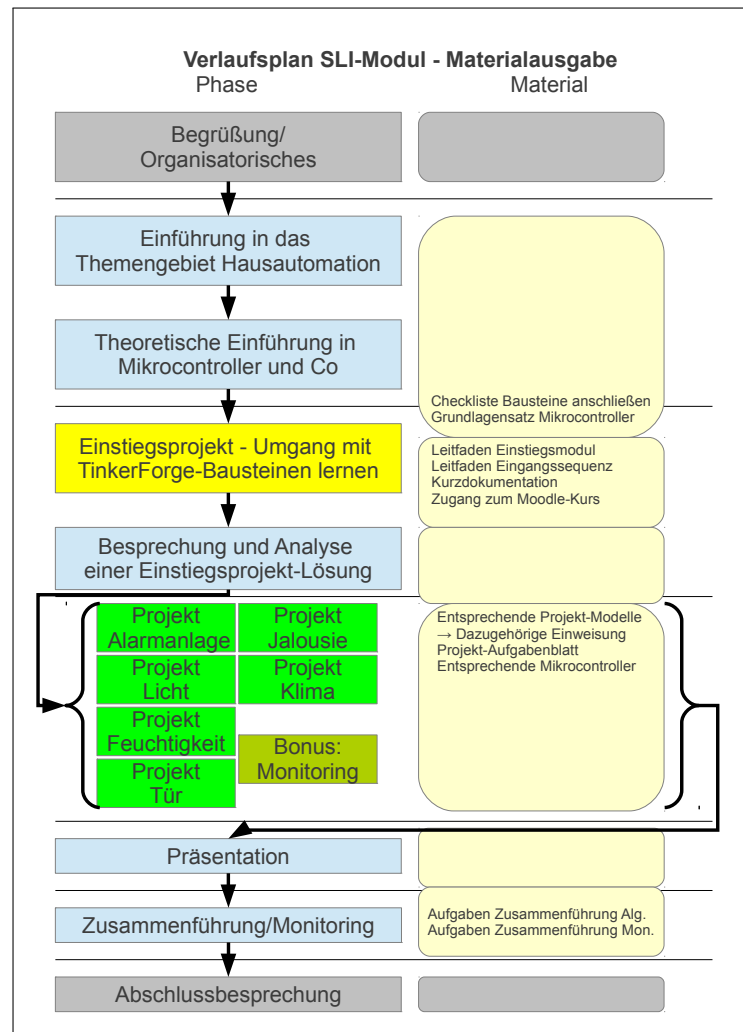



Abbildung 3.14.: Verlaufsplan Materialausgabe

Das Betreuermerkblatt Mikrocontroller (Abbildung 3.16) ist bereits im Kapitel 3.4.1 aufgegriffen und seine Verwendung erläutert worden. Auf ihm sind die sieben häufigsten und grundlegendsten Begriffe zum Themenbereich Mikrocontroller gesammelt und erläutert. Fachbegriffe innerhalb eines Erläuterungstextes sind wiederum selber als Stichwort aufgeführt, sodass das Merkblatt eine inhaltlich abgerundete und abgeschlossene Einheit darstellt. Platz für eigene Ergänzungen ist gegeben. Die Erklärungen sind einfach gehalten, um auch weniger sachkundigen Betreuern einen Einstieg in die Thematik zu ermöglichen.



Schülerlabor Informatik

Betreuerleitfaden
Anleitung für die Durchführung



InfoSphere
World of Informatics

Reihenfolge der Arbeitsschritte

1. Präsentation
2. Folie 15: Checkliste „Baustein anschließen“ verteilen. Material austeilen. Leitfaden austeilen. Zeigen, wie man Bricklets anschließt (An Merkblatt orientieren, SuS müssen das aufschreiben)
3. Folie 16: BrickViewer zeigen, Bausteine anschließen und verbinden, UID-Seite zeigen, alle Schritte in SuS-Wiederholungs-Leitfaden durchlaufen
4. Ab Folie 17: Programmbeispiel durchsprechen
6. Folie 38: SuS Checkliste zur Verwendung eines Bricklets ausfüllen lassen
7. Folie 56: Struktur des Einstiegsleitfadens erklären
8. SuS können Einstiegsprojekt starten
9. Nachdem alle Gruppen im Wesentlichen fertig sind: Eine SuS-Lösung am Beamer besprechen
10. Präsentation weiterführen ab Folie 60
11. Folie 63: AG-Zuteilung
12. Arbeitsphase Projekte, Material verteilen
Sollten Gruppen deutlich früher fertig sein, als der Rest:
 - Wenn die verbleibende Zeit das Bearbeiten der Optionalphase wahrscheinlich macht: Bonusaufgabe Monitoring verteilen.
 - Wenn die Optionalphase vermutlich nicht mehr bearbeitet werden kann: Die fertigen SuS zur Optimierung ihrer Projekte anregen.
13. Nach der Arbeitsphase (Folie 65): Gruppen stellen sich ihre Ergebnisse vor. Jede Gruppe führt ihr Ergebnis kurz vor. Alle anderen Begeben sich zum jeweiligen, gerade vorgestellten Projekt.
14. Abschluss: Zentralsteuerung. Aufgabenblatt „Zusammenführung Allgemein“ an alle austeilen. Falls keine Gruppe die Bonusaufgabe bearbeitet hat: Musterlösung verwenden und vorbereiten. Falls die Bonusaufgabe bearbeitet wurde: Für die entsprechenden Teilnehmer das Arbeitsblatt „Zusammenführung Monitoring“
15. SuS bauen von Betreuern angeleitet das gesamte Haus auf. Musterlösung Monitoring wird verwendet, falls keine geeignete SuS-Lösung vorliegt.
16. Abschlussbesprechung

1

Abbildung 3.15.: Betreuer Leitfaden Einstiegsprojekt Seite 1

Den Modulbetreuern steht weiterhin eine Musterlösung der Checkliste zum Anschließen der Bausteine zur Verfügung. Da die Teilnehmer diese ihrerseits mit eigenen Worten ausfüllen sollen, stellt die Musterlösung nicht den genauen zu verwendenden Wortlaut dar, vielmehr bietet sie eine Übersicht über die einzelnen Schritte und hilft den Betreuern bei der korrekten Vorführung. Erst sie ermöglicht eine unkomplizierte Abstimmung des Betreuervortrages bzw. der Vorführung auf die Checkliste der Teilnehmer.


Dem Betreuer team steht, genauso wie den Teilnehmern, eine Druckversion des in der Präsentation erarbeiteten Beispielprogramms zur Verfügung, um sie diesbezüglich von den Präsentationsfolien unabhängig zu machen und den einzelnen Betreuern ein individuelles Nachschlagen zu ermöglichen. Zusätzlich erhalten die Betreuer die



Schülerlabor Informatik

Mikrocontroller und Co

Merkblatt für Betreuer



InfoSphere
World of Informatics

Prozessor: Eine Maschine oder eine elektronische Schaltung, welche gemäß übergebener Befehle andere Maschinen oder elektrische Schaltungen steuert und dabei einen Prozess oder Algorithmus vorantreibt, was meist Datenverarbeitung beinhaltet. Am populärsten sind Prozessoren als zentrale Recheneinheiten von Computern, in denen sie Befehle (Software) ausführen.

Mikroprozessor: Moderne Form des Prozessors, der alle Bausteine des Prozessors auf einem Chip vereinigt.

Chip: Plättchen aus einem Halbleitermaterial, auf dem sich ein vollständiges, funktionfähiges elektronik-Bauteil befindet.

Platine: Eine Leiterplatte (oder gedruckte Schaltung) ist ein Träger für elektronische Bauteile. Sie dient der mechanischen Befestigung und elektrischen Verbindung. Nahezu jedes elektronische Gerät enthält eine oder mehrere Leiterplatten.

Mikrocontroller: Als Mikrocontroller werden Halbleiterchips bezeichnet, die mit dem Prozessor auch Peripheriefunktionen auf einem Chip vereinen. In vielen Fällen befindet sich der Arbeits- und Programmspeicher ebenfalls teilweise oder komplett auf dem selben Chip. Ein Mikrocontroller ist praktisch ein Ein-Chip-Computersystem.

System-on-a-Chip (SoC): Unter einem SoC versteht man die Integration aller oder eines großen Teils der Funktionen eines Systems auf einem Chip. Als *System* wird dabei eine Kombination unterschiedlicher Elemente (logischen Schaltungen, Taktgebung, selbständiges Anlaufen, usw.) aufgefasst, die zusammen eine bestimmte Funktionalität bereitstellen, beispielsweise ein Temperatursensor samt Auswertungselektronik.

Firmware: Unter Firmware (von engl. „firm“ = fest) versteht man Software, die fest in elektronische Geräte eingebettet (integriert) ist. Der Begriff leitet sich davon ab, dass Firmware funktional fest mit der Hardware verbunden ist, was bedeutet, dass das eine ohne das andere nicht nutzbar ist. Sie nimmt eine Zwischenstellung zwischen Hardware (also den physikalischen Anteilen eines Gerätes) und der Anwendungssoftware (den ggf. austauschbaren Programmen eines Gerätes) ein. Wird auch als „Betriebssoftware“ bezeichnet.

Platz für eigene Ergänzungen:

Basierend auf Wikipedia.de

Abbildung 3.16.: Betreuermerkblatt Mikrocontroller

Druckversion einer Beispiellösung für das Einstiegsprojekt. Damit können sie sich gegebenenfalls einen Einblick in die Erwartungen an die Teilnehmerlösungen verschaffen, wobei die einzelnen SuS-Lösungen unterschiedlich ausfallen dürfen. Außerdem können sie sich damit auf Fragen der Teilnehmer bezüglich dieser Aufgabe besser vorbereiten und eigene fehlende Kenntnisse über geeignete Lösungswege ausgleichen, ohne die Aufgabe selber gelöst zu haben. Das trägt wiederum zu einer Verkürzung der notwendigen Einarbeitungszeit bei.

Um dies inhaltlich zu ergänzen, steht den Betreuern ein „Betreuermerkblatt Java“ zur Verfügung, welches ihnen die zu verwendenden Java-Features wie den try-catch-Block oder Listener detaillierter erklärt. Dieses Merkblatt vermittelt auch ungeübten Programmierern



das nötige Hintergrundwissen, um den SuS diese Features erklären zu können.

Digitale Materialien

Neben den analogen, sprich in Druckversion vorliegenden, Materialien gibt es in digitaler Fassung Musterlösungen und ein Wiki.

Für alle vorkommenden Aufgabenstellungen ist für die Betreuer im Moodle-Kurs, nur für Betreuer einsehbar, eine Musterlösung hinterlegt. Wie bereits erwähnt dient keine dieser Musterlösungen als Referenzlösung, die von den Teilnehmern erreicht werden muss. Alle Musterlösungen sollen die Betreuer gegenüber den Teilnehmern in die Lage versetzen, einen denkbaren Lösungsweg für die jeweilige Aufgabe zu kennen und sie somit mit ausreichend Hintergrundwissen ausstatten, die SuS gegebenenfalls kompetent beraten zu können. Im Falle der Optionalphase dient die Musterlösung für die Bonusaufgabe aber auch als Ersatz, falls keine geeignete Teilnehmerlösung vorliegt.

Neben den rein inhaltlichen Problemen, die während der Durchführung des InfoSphere-Moduls auftreten können, sind eine Reihe weiterer Schwierigkeiten denkbar. Beispielsweise könnten technische Probleme mit den Modellen oder der Tinkerforge-Hardware auftreten. Um die nicht an der Modulentwicklung beteiligten Betreuer zu befähigen, diese Probleme schnell zu lösen und um ihnen eine geeignete Anlaufstelle für ihre Fragen zu geben, wird den Betreuern ein Wiki zur Verfügung gestellt. Dabei handelt es sich um ein Tiddly-Wiki [8], welches nur aus einer einzigen Datei besteht und somit leicht zu handhaben und zugleich portabel und Webserver-unabhängig ist. Die häufigsten zu erwartenden Schwierigkeiten sind dort bereits mit Lösungsansatz eingetragen. Da aber nicht alle eventuell auftretenden Probleme vorher absehbar sind, ist das Wiki naturgemäß unvollständig. Es kann aber von allen Betreuern mit einem Webbrowser editiert und ergänzt werden, damit neue Problemlösungen und Hinweise für Betreuer an nachfolgende Betreuergenerationen weitergegeben werden können. Dadurch wird eine kontinuierliche Fortentwicklung und Optimierung der Betreuermaterialien ermöglicht. Das Wiki befindet sich im Digitalisat (Anhang Abschnitt E).



Kapitel 4 Reflexion

Dieses Kapitel ist ein Rückblick auf das InfoSphere-Modul und seine Entwicklung und gibt Einblick in die Hintergründe diverser Gestaltungsentscheidungen. Weiterhin werden Stärken und Schwächen des Gesamtkonzeptes diskutiert, bevor abschließend einige allgemeine Erkenntnisse herausgefiltert werden.

4.1. Testdurchführungen und Optimierungen

Im Verlauf des Entwicklungsprozesses zum vorliegenden Ergebnis haben sich, ausgehend vom Erstentwurf des InfoSphere-Moduls, mehrere Veränderungen und Anpassungen ergeben. Damit wurde auf Probleme und Schwächen reagiert, die bei Testdurchläufen des InfoSphere-Modul-Prototypen festgestellt wurden. Eine Erläuterung dieser Optimierungen unterstreicht die einzelnen Argumentationen für die letztliche Modulgestaltung und verbessert das Verständnis für einzelne Designentscheidungen.

Optimierung der praktischen Einführungsphase

Die erste Testdurchführung sollte grobe Schwächen in der didaktischen Gesamtkonzeption und in den Prototypen der Arbeitsmaterialien aufdecken. Als Testteilnehmer wurden die Betreuer des Schülerlabors ausgewählt. Sie eigneten sich für eine Beurteilung besonders deshalb, weil sie die Zielgruppe bzw. Teilnehmerschaft der Schülerlabor-Module aus eigener Erfahrung gut kennen.

Neben vielen weiteren Detailergebnissen hat sich die Gestaltung der praktischen Einführungsphase, in der die SuS den Umgang mit den Tinkerforge-Mikrocontrollern lernen, als Schwachstelle herausgestellt.



Sie war ursprünglich so angelegt, dass sich die SuS alle nötigen Kenntnisse und Arbeitsschritte für den Mikrocontroller-Betrieb mit Hilfe eines verschriftlichten Tutorials aneignen und durcharbeiten würden. Problem dieses Ansatzes war, dass dieses Tutorial aufgrund des Umfangs der zu vermittelnden Inhalte sehr umfangreich und langatmig war. Ein solches Textwerk hätte eine demotivierende und abschreckende Wirkung auf die Teilnehmer gehabt. Zusätzlich wäre es durch seine Textaufgaben-Charakteristik dem Schulunterricht sehr ähnlich gewesen, was der eingangs formulierten Zielsetzung einer explizit außerschulischen Atmosphäre widersprach.

Die Verbesserungen, die aus der gewonnenen Erkenntnis resultierten, führten zur Vorläuferversion der in Kapitel 3.4.1 beschriebenen Gestaltung der Einführungsphase, die im Gegensatz zur ersten interaktiver angelegt ist und wesentlich weniger umfangreiches Arbeitsmaterial beinhaltet. Der letzte Verbesserungsschritt bis zur endgültigen Fassung gelang unter Berücksichtigung der Testergebnisse aus dem zweiten Testdurchlauf.

Optimierung des Präsentationsleitfadens

Der zweite Testlauf hatte mehrere freiwillige SuS, also Personen aus der realen Zielgruppe, als Testteilnehmer. Da die größten Unstimmigkeiten und Probleme des Modul-Prototypen nach dem ersten Testdurchlauf behoben waren, war eine Durchführung unter realistischen Bedingungen notwendig, um auch Schwachstellen in Gestaltungsdetails erkennen zu können. Besonders spannender Testaspekt war der Anforderungsgrad, der an die SuS gestellt wurde. Da die Modulbetreuer aus dem ersten Testdurchlauf wesentlich mehr fachliches Wissen mitbrachten, war eine realistische Abschätzung des Schwierigkeitsgrades und der Lösbarkeit der Aufgaben für SuS nur schwer möglich.

Erfreulicherweise kann als Testergebnis festgehalten werden, dass das vorausgesetzte Vorwissen der SuS in der Planung realistisch eingeschätzt wurde und alle getesteten Aufgaben von den Teilnehmern erfolgreich gelöst werden konnten. Dementsprechend war keine Überforderung der Teilnehmer zu erkennen. Eine Unterforderung war ebenfalls nicht erkennbar, die Optionalphase wurde innerhalb der angesetzten vier Teststunden nicht erreicht.

Daraus leitet sich das nächste Testergebnis ab: Die ursprünglich angesetzte Exkursionsdauer von vier Stunden war zu kurz. Um eine ausreichend lange Arbeitszeit in der Projektphase zu erhalten, wurde die Exkursionsdauer auf fünf Stunden angehoben.

Als größte Schwachstelle stellte sich aber die Gestaltung des Präsentationsleitfadens heraus: Einerseits enthielt er zu viele technische Details in der Vorstellung der allgemeinen Funktionsweise von



Mikrocontrollern im Allgemeinen und den Tinkerforge-Bausteinen im Speziellen. Diese waren für die Bearbeitung der einzelnen Aufgaben nicht relevant und stellten daher unnötig hohe Ansprüche an die Aufmerksamkeit und Konzentrationsfähigkeit der Teilnehmer. Zusätzlich nahmen sie zu viel Zeit in Anspruch, die später in der Projektphase fehlte.

Andererseits war die Vermittlung des Wissens für den Umgang mit den Mikrocontrollern durch die Präsentation immer noch unvorteilhaft. Der Ablauf sah vor, dass das Wissen und die Arbeitsschritte vortragsartig vorgestellt und erst anschließend von den SuS nachgemacht werden sollten. In der Praxis führte dies zu einer langen, passiven Aufnahmezeit für die Teilnehmer. Sie mussten erst lange zuhören, bevor sie das Gesehene anwenden konnten, und hatten bis dahin einen Teil wieder vergessen. Als Verbesserung resultierte daraus der oben erläuterte Modulablauf, der die SuS die vorgeführten Arbeitsschritte sofort nachmachen und aufschreiben lässt und diese Phase dadurch wesentlich interaktiver und aus Teilnehmersicht vor allem aktiver umsetzt. Die Wartezeiten entfallen und zusätzlich erleben die SuS in regelmäßigen, kurzen Abständen kleine Erfolgserlebnisse, die sie motivierend durch die Lernphase hindurchtragen.

Die letzten Testergebnisse beziehen sich vor allem auf kleinere Informationen, die die SuS nicht aus ihrem Vorwissen heraus besitzen, aber für das Lösen der Aufgaben benötigen. Dazu gehört beispielsweise das Typecasten von short-Variablen in Java. Derartige Ergebnisse sind in kleinen Anpassungen der Arbeitsmaterialien berücksichtigt worden.

4.2. Kritische Reflexion des didaktischen Gesamtkonzeptes

Wie gut die mit dem InfoSphere-Modul verfolgten Ziele erreicht werden, lässt sich erst nach einer längeren Testdauer im Realbetrieb fundiert beurteilen. Dennoch gibt es Eigenschaften und Aspekte, die vorab thematisiert werden können: So bringt die didaktische Gestaltung neben ihren Vorteilen naturgemäß auch Nachteile mit sich.

Zu den Schwächen des Konzeptes gehört der knappe zeitliche Rahmen. Es ist davon auszugehen, dass nur leistungstärkere Teilnehmergruppen schnell genug arbeiten und dadurch in der Lage sein werden, die Optionalphase zu bearbeiten. Eine Auslegung auf eine längere Exkursionsdauer wäre wünschenswert, lässt sich aber im Rahmen einer Eintagesexkursion schwerlich realisieren. Dieser Nachteil wiegt aber dank eines geeigneten alternativen Exkursionsabschlusses mit der Präsentationsphase nicht schwer. Die Bearbeitung der Optionalphase ist zwar aufgrund der gesteigerten Kompetenzvermittlung erstrebenswert, jedoch dennoch als Bonus zu verstehen.

Als weitere Schwachstelle des Konzepts kann die große Freiheit, die die



Teilnehmer bei der Bearbeitung der Projektphase haben, ausgelegt werden. Möglicherweise haben weniger zielstrebige Teilnehmer Schwierigkeiten, trotz fehlender Anleitung zeitnah ein Ergebnis herzustellen und sich nicht in der Vielfalt der sich anbietenden Optionen zu verlieren. Dies wird durch die unverbindlichen Vorschläge auf den Aufgabenblättern auszugleichen versucht. In letzter Instanz ist diesbezüglich auf die Exkursionsbetreuer zu vertrauen. Ihre Situationsanalyse und ihr Einsatz sollten das Gelingen der Exkursion durch geeignete Einflussnahme und Unterstützung der SuS sichern können. Das wiederum ist aber nicht als Schwachpunkt des Konzeptes zu verstehen. Vielmehr unterstreicht es die Menschlichkeit einer Lehr-Lern-Situation und zeigt die Unverzichtbarkeit kompetenter Lehrkräfte. Eine Lernsituation ist rein durch Lehrmaterialien, seien sie noch so gut, selten erfolgreich zu meistern.

Bedauernd ist, dass die Zielsetzung der außerschulischen Atmosphäre nur teilweise erfüllt werden konnte. Zu dieser Atmosphäre tragen die Örtlichkeit, Exkursionsinhalt, Organisation, Arbeitsform und die Form der Betreuung bei. Trotzdem bleiben viele aus dem Schulalltag bekannte Situationen bestehen: Es gibt weiterhin das Prinzip von Aufgabenstellung und Bearbeitungszeit, Lehrer und Lerner, und auch die Möglichkeiten der Mitbestimmung über den Lernverlauf sind begrenzt. Dieser Nachteil ist in Kauf zu nehmen. Angesichts begrenzter zeitlicher und finanzieller Ressourcen scheint ein völliges Loslösen von diesen Prinzipien nicht möglich, ein Rückgriff auf Schulmethoden ist daher stellenweise erforderlich.

Ein Problem, das in didaktischen Gestaltung gar nicht berücksichtigt wurde, ist ein Hardwareausfall. Angesichts der zur Verfügung stehenden Mittel ist ein Alternativ- oder Ersatzverfahren nicht realisierbar. Allerdings ist dieser Aspekt ohnehin nicht vollständig abzusichern. Er ist eher dem Bereich der höheren Gewalt zuzuordnen. Sollte also im Verlauf der Exkursion Hardware ausfallen, ist einmal mehr das situationsgerechte Handeln der Betreuer gefragt.

Gudjons führt als nachteiligen Aspekt von Projektunterricht auf, dass die SuS in der Lernphase vor der eigentlichen Projektarbeit meist wenig motiviert seien. [2, vgl. S. 91] Im InfoSphere-Modul wird das durch die Einbeziehung der Teilnehmer und eine kurz gehaltene theoretische Einführung ausgeglichen, sodass dieser Kritikpunkt an Bedeutung verliert.

Trotz vieler kleiner Erfolgserlebnisse, die die Teilnehmer im Verlaufe der Exkursion erleben sollten, sind auch immer wieder kleine Rückschläge zu erwarten. Die Programmierung von Mikrocontrollern erfordert, wie schon Programmierung im Allgemeinen, streckenweise Geduld und Frustrationstoleranz. Das ist in dieser Exkursion vor allem deshalb zu erwarten, da die SuS größtenteils noch nicht mit Software-Dokumentationen gearbeitet haben und sie darin ungeübt sind. Zum



unverfälschten Kennenlernen von Informatik gehört aber auch das Er-
fahren von frustrierenden Arbeitssituationen. Wichtig für die Exkursi-
on ist, dass die Erfolgserlebnisse die schwergängigeren Arbeitsphasen
überwiegen. Damit das gewährleistet werden kann, sind die Betreuer
gefordert, gegebenenfalls Hilfestellungen und Ratschläge zu geben, da-
mit den Teilnehmern einzelne Hürden nicht unüberwindbar werden.

4.3. Fazit

Zusammenfassend stellt sich die Frage, was das InfoSphere-Modul
leisten kann und was nicht. Es ist deutlich geworden, dass die Ex-
kursion keine tiefgehenden fachlichen Einblicke in Mikrocontroller-
Programmierung bieten kann, weil hierfür der Umfang der Veranstal-
tung schlicht nicht ausreicht. Da eine Wiederholung und Sicherung des
Gelernten zumindest im Rahmen der Exkursion nur rudimentär durch
die Präsentationsphase gegeben ist und eine gleichmäßige Beteiligung
aller SuS in der Gruppenarbeit nicht gesichert ist, ist von einem homo-
genen Erkenntnisgewinn und Lernzuwachs für die einzelnen SuS aus-
zugehen. Wenn auch der Erkenntnisgewinn nicht für alle Teilnehmer
gleich ist, so ist dennoch das Hauptziel erfüllt worden: Jeder Teilneh-
mer konnte individuelle Erfahrungen sammeln, die ihm einen besseren
Überblick über die Informatik verschaffen. Das InfoSphere-Modul ver-
schafft den SuS einen tiefen Einblick in das Themengebiet Hausauto-
mation und streift dabei viele seiner Facetten. Durch die intensive prak-
tische Auseinandersetzung mit typischen Inhalten, Problemen und Ar-
beitsweisen hinterlässt es bei den Teilnehmern einen nachhaltigen Ein-
druck von Aspekten der Informatik, den der gewöhnliche Schulunter-
richt in dieser Form nicht erzielen kann.



Anhang A Literaturverzeichnis

Printed References

- [2] Herbert Gudjons. *Handlungsorientiert lehren und lernen. Schüleraktivierung. Selbsttätigkeit. Projektarbeit*. Julius Klinkhardt Verlag, 2008. ISBN: 9783781516250.
- [4] Hilbert Meyer. „Handlungsorientierter, handelnder und schülerorientierter Unterricht“. In: *Hilbert Meyer: Unterrichtsmethoden. Band 1: Theorieband*. Cornelsen Scriptor, 1987. ISBN: 3589208503.
- [7] *Richtlinien und Lehrpläne für die Sekundarstufe II - Gymnasium/Gesamtschule in Nordrhein-Westfalen. Informatik*. 1. Auflage. Ministerium für Schule und Weiterbildung, Wissenschaft und Forschung des Landes Nordrhein-Westfalen. Völklinger Straße 49, 40221 Düsseldorf. Frechen: Ritterbach Verlag, 1999.
- [9] *Vorgaben zu den unterrichtlichen Voraussetzungen für die schriftlichen Prüfungen im Abitur in der gymnasialen Oberstufe im Jahr 2013. Vorgaben für das Fach Informatik*. Ministerium für Schule und Weiterbildung, Wissenschaft und Forschung des Landes Nordrhein-Westfalen. Völklinger Straße 49, 40221 Düsseldorf. 2010.

Online References

- [1] *Arduino offizielle Webseite*. URL: <http://arduino.cc/> (besucht am 02. 02. 2013).
- [3] *Infosphere Schülerlabor Informatik Webseite*. URL: <http://schuelerlabor.informatik.rwth-aachen.de/> (besucht am 02. 02. 2013).
- [5] *Moodle offizielle Webseite*. URL: <http://moodle.de/> (besucht am 02. 02. 2013).



- [6] *Online-Dokumentation der Tinkerforge GmbH.* URL: <http://www.tinkerforge.com/doc/index.html> (besucht am 02.02.2013).
- [8] *Tiddlywiki offizielle Webseite.* URL: <http://tiddlywiki.com/> (besucht am 02.02.2013).
- [10] *Webseite der Tinkerforge GmbH.* URL: <http://www.tinkerforge.com/> (besucht am 02.02.2013).



Anhang B Exkursionspräsentation



**Herzlich Willkommen im InfoSphere
Schülerlabor Informatik**

Hausautomation mit Mikrocontrollern



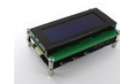
Was ist Hausautomation?

Welche Beispiele fallen euch ein?

2

Tagesablauf

- Kurze Einführung ins Thema
- Übungsrunde mit Mikrocontrollern
- Projekte zur Hausautomation umsetzen
- Präsentation eures Projektes
- (Zusammenführung der Projekte
→ Zentralsteuerung)
- Abschlussbesprechung



3







Intelligentes Haus

Automatik

Sicherheit

Komfort





Quelle: home.arcor.de/dirk.milewski/seite75.html

Quelle: Gira

Quelle: www.dimm.tv/heizung.html

4





Hausautomation realisieren wir mit dem TinkerForge -System

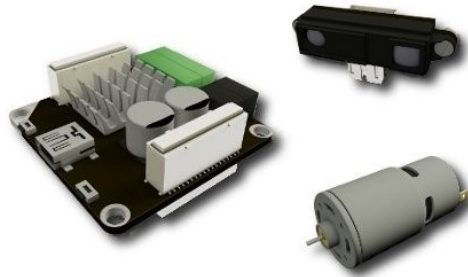


5



Das TinkerForge -System

... ist ein elektronischer, modularer Baukasten



6

Wie realisiert man die Steuerung?

- **Computer**
verarbeiten Daten + steuern Aktionen



7



Wie realisiert man die Steuerung?

- **Computer**

verarbeiten Daten + steuern Aktionen



- **Mikrocontroller**

Steuerung von Sensoren/Aktoren

Weitergabe von Messdaten



8

Bildquellen: IBM, Tinkerforge

Wie realisiert man die Steuerung?

- **Computer**

verarbeiten Daten + steuern Aktionen



- **Mikrocontroller**

Steuerung von Sensoren/Aktoren

Weitergabe von Messdaten



- **Sensoren/Aktoren**

Sensoren messen (Temperatur...)

Aktoren handeln (Motor, LCD...)



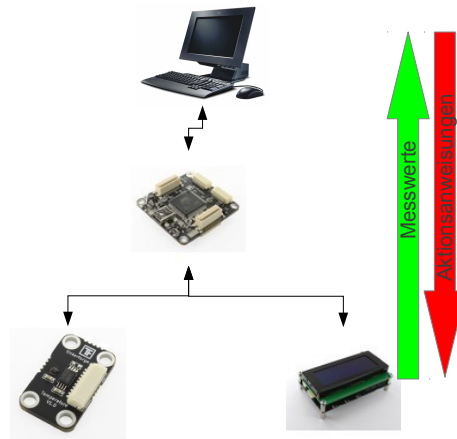
9

Bildquellen: IBM, Tinkerforge



Wie realisiert man die Steuerung?

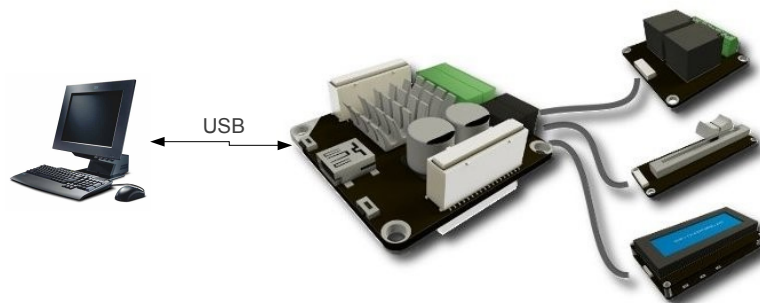
- **Computer**
verarbeiten Daten + steuern Aktionen
- **Mikrocontroller**
Steuerung von Sensoren/Aktoren
Weitergabe von Messdaten
- **Sensoren/Aktoren**
Sensoren messen (Temperatur...)
Aktoren handeln (Motor, LCD...)



10

Bildquellen: IBM, Tinkerforge

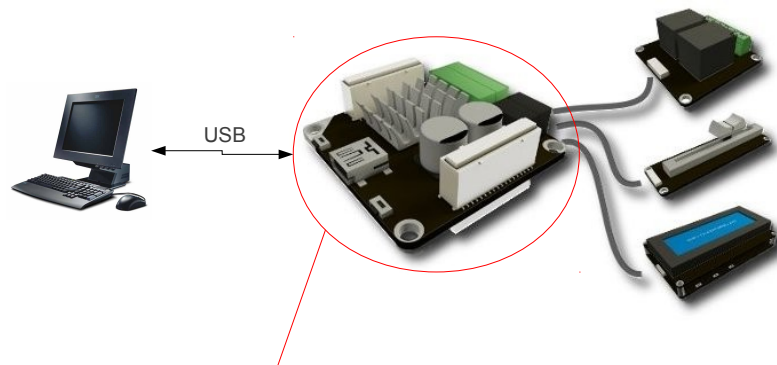
Das TinkerForge -System



11



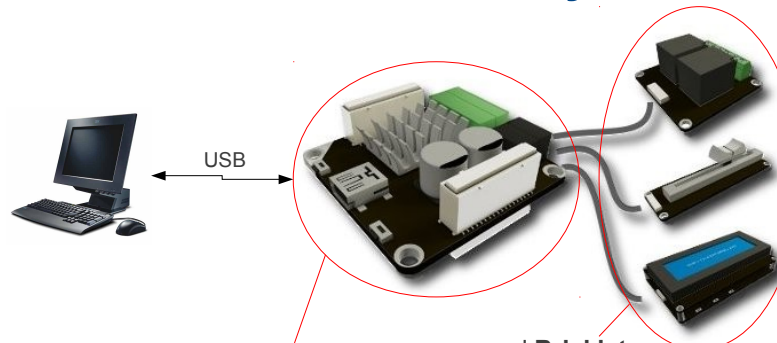
Das TinkerForge -System



- Es gibt verschiedene **Bricks**
- Per USB steuerbare Bausteine
 - Übernehmen die Funktion des Mikrocontrollers

12

Das TinkerForge -System



- Es gibt verschiedene **Bricks**
- Per USB steuerbare Bausteine
 - Übernehmen die Funktion des Mikrocontrollers
- ... und **Bricklets**
- Erweitern die Fähigkeiten von Bricks
 - Sensoren und Aktoren

13



Einführungsrunde

- Bevor ihr ein ganzes Haus automatisiert, müssen wir uns mit den Bricks und Bricklets vertraut machen.
- **Vorsichtig mit allen Bauteilen umgehen. Sehr empfindlich.**
- Die Betreuer teilen euch in 4er Gruppen ein.

14

Wie schließe ich Bausteine an?

- Betreuer zeigt ein Beispiel...
- Füllt die Checkliste „Bausteine anschließen“ aus!
- Baut eure Bausteine entsprechend zusammen!

15



Wie verwende ich den BrickViewer?

- Betreuer zeigt ein Beispiel...
- Macht die Schritte an euren Laptops nach!
 - Je 2 Personen pro Laptop

16

Java: Programmrahmen

Wie verwende ich die Bausteine mit Java?

17



Java: Programmrahmen

```
1)
2)
3)
4)
5)
6) public class Einstieg {
7)
8)
9)
10)
11)
12)
13) public static void main(String args[]) throws Exception {
14)
15)
16)
17)
18)
19)
20)
21)
```

Hier steht die eigentliche Programmfunktionalität... (nächste Folie)

```
42)
43)
44)
45)
46) }
47)}
```

18

Verbindung zu den Bausteinen -
Vorbereitung

19



Java: Programmrahmen

```

1)
2)
3)
4)
5)
6) public class Einstieg {
7)
8)
9)
10)
11)
12)
13) public static void main(String args[]) throws Exception {
14)
15)
16)
17)
18)
19)
20)
21)

```

Hier steht die eigentliche Programmfunktionalität... (nächste Folie)

```

42)
43)
44)
45)
46) }
47) }

```

20

Java: Programmrahmen

```

1)
2)
3)
4) import com.tinkerforge.IPConnection;
5)
6) public class Einstieg {
7)     private static final String host = "localhost";
8)     private static final int port = 4223;
9)
10)
11)
12)
13) public static void main(String args[]) throws Exception {
14)
15)     IPConnection ipcon = new IPConnection(host, port);
16)
17)
18)
19)
20)
21)

```

Bibliotheken importieren

Verbindungsinformationen

Hier steht die eigentliche Programmfunktionalität... (nächste Folie)

```

42)
43)
44)
45)
46) }
47) }

```

21



MasterBrick verwenden

22

Java: Programmrahmen

```
1)
2)
3)
4) import com.tinkerforge.IPConnection;
5)
6) public class Einstieg {
7)     private static final String host = "localhost";
8)     private static final int port = 4223;
9)
10)
11)
12)
13)     public static void main(String args[]) throws Exception {
14)
15)         IPConnection ipcon = new IPConnection(host, port);
16)
17)
18)
19)
20)
21)
```

Bibliotheken importieren

Verbindungsinformationen

Hier steht die eigentliche Programmfunktionalität... (nächste Folie)

```
42)
43)
44)
45)
46)     }
47) }
```

23



Java: Programmrahmen

```

1) import com.tinkerforge.BrickMaster;
2)
3)
4) import com.tinkerforge.IPConnection;
5)
6) public class Einstieg {
7)     private static final String host = "localhost";
8)     private static final int port = 4223;
9)
10)
11)
12)
13)     public static void main(String args[]) throws Exception {
14)
15)         IPConnection ipcon = new IPConnection(host, port);
16)
17)
18)
19)
20)
21)

```

Bibliotheken Importieren

Verbindungsinformationen

Hier steht die eigentliche Programmfunktionalität... (nächste Folie)

```

42)
43)
44)
45)
46) }
47) }

```

24

Java: Programmrahmen

```

1) import com.tinkerforge.BrickMaster;
2)
3)
4) import com.tinkerforge.IPConnection;
5)
6) public class Einstieg {
7)     private static final String host = "localhost";
8)     private static final int port = 4223;
9)
10)     private static final String masterUID = "9JnhKoQ8ieS";
11)
12)
13)     public static void main(String args[]) throws Exception {
14)
15)         IPConnection ipcon = new IPConnection(host, port);
16)
17)
18)
19)
20)
21)

```

Bibliotheken Importieren

Verbindungsinformationen

UIDs der Bricks und Bricklets

Hier steht die eigentliche Programmfunktionalität... (nächste Folie)

```

42)
43)
44)
45)
46) }
47) }

```

25



Java: Programmrahmen

```

1) import com.tinkerforge.BrickMaster;
2)
3)
4) import com.tinkerforge.IPConnection;
5)
6) public class Einstieg {
7)     private static final String host = "localhost";
8)     private static final int port = 4223;
9)
10)    private static final String masterUID = "9JnhKoQ8ieS";
11)
12)
13)    public static void main(String args[]) throws Exception {
14)
15)        IPConnection ipcon = new IPConnection(host, port);
16)
17)        final BrickMaster master = new BrickMaster(masterUID);
18)
19)
20)
21)

```

Bibliotheken importieren

Verbindungsinformationen

UIDs der Bricks und Bricklets

Brick- und Brickletobjekte erzeugen mit UIDs

Hier steht die eigentliche Programmfunktionalität... (nächste Folie)

```

42)
43)
44)
45)
46) }
47) }

```

26

Java: Programmrahmen

```

1) import com.tinkerforge.BrickMaster;
2)
3)
4) import com.tinkerforge.IPConnection;
5)
6) public class Einstieg {
7)     private static final String host = "localhost";
8)     private static final int port = 4223;
9)
10)    private static final String masterUID = "9JnhKoQ8ieS";
11)
12)
13)    public static void main(String args[]) throws Exception {
14)
15)        IPConnection ipcon = new IPConnection(host, port);
16)
17)        final BrickMaster master = new BrickMaster(masterUID);
18)
19)
20)        ipcon.addDevice(master);
21)

```

Bibliotheken importieren

Verbindungsinformationen

UIDs der Bricks und Bricklets

Brick- und Brickletobjekte erzeugen mit UIDs

Verbindung zu Bricks und Bricklets herstellen

Hier steht die eigentliche Programmfunktionalität... (nächste Folie)

```

42)
43)
44)
45)
46) }
47) }

```

27



Programm beenden

28

Java: Programmrahmen

```

1) import com.tinkerforge.BrickMaster;
2)
3)
4) import com.tinkerforge.IPConnection;
5)
6) public class Einstieg {
7)     private static final String host = "localhost";
8)     private static final int port = 4223;
9)
10)    private static final String masterUID = "9JnhKoQ8ieS";
11)
12)
13)    public static void main(String args[]) throws Exception {
14)
15)        IPConnection ipcon = new IPConnection(host, port);
16)
17)        final BrickMaster master = new BrickMaster(masterUID);
18)
19)        ipcon.addDevice(master);
20)
21)
22)
23)
24)
25)
26)
27)
28)
29)
30)
31)
32)
33)
34)
35)
36)
37)
38)
39)
40)
41)
42)    System.console().readLine("Press [Enter] to exit\n");
43)
44)
45)
46)    }
47) }

```

Bibliotheken importieren

Verbindungsinformationen

UIDs der Bricks und Bricklets

Brick- und Brickletobjekte erzeugen mit UIDs

Verbindung zu Bricks und Bricklets herstellen

Hier steht die eigentliche Programmfunktionalität... (nächste Folie)

29



Java: Programmrahmen

```

1) import com.tinkerforge.BrickMaster;
2)
3)
4) import com.tinkerforge.IPConnection;
5)
6) public class Einstieg {
7)     private static final String host = "localhost";
8)     private static final int port = 4223;
9)
10)    private static final String masterUID = "9JnhKoQ8ieS";
11)
12)
13)    public static void main(String args[]) throws Exception {
14)
15)        IPConnection ipcon = new IPConnection(host, port);
16)
17)        final BrickMaster master = new BrickMaster(masterUID);
18)
19)        ipcon.addDevice(master);
20)
21)
22)
23)
24)
25)
26)
27)
28)
29)
30)
31)
32)
33)
34)
35)
36)
37)
38)
39)
40)
41)
42)    System.console().readLine("Press [Enter] to exit\n");
43)
44)        ipcon.destroy();
45)        System.exit(0);
46)    }
47) }

```

Bibliotheken importieren

Verbindungsinformationen

UIDs der Bricks und Bricklets

Brick- und Brickletobjekte erzeugen mit UIDs

Verbindung zu Bricks und Bricklets herstellen

Hier steht die eigentliche Programmfunktionalität... (nächste Folie)

30

Java: Programmrahmen

```

1) import com.tinkerforge.BrickMaster;
2)
3)
4) import com.tinkerforge.IPConnection;
5)
6) public class Einstieg {
7)     private static final String host = "localhost";
8)     private static final int port = 4223;
9)
10)    private static final String masterUID = "9JnhKoQ8ieS";
11)
12)
13)    public static void main(String args[]) throws Exception {
14)
15)        IPConnection ipcon = new IPConnection(host, port);
16)
17)        final BrickMaster master = new BrickMaster(masterUID);
18)
19)        ipcon.addDevice(master);
20)
21)
22)
23)
24)
25)
26)
27)
28)
29)
30)
31)
32)
33)
34)
35)
36)
37)
38)
39)
40)
41)
42)    System.console().readLine("Press [Enter] to exit\n");
43)
44)        ipcon.destroy();
45)        System.exit(0);
46)    }
47) }

```

Bibliotheken importieren

Verbindungsinformationen

UIDs der Bricks und Bricklets

Brick- und Brickletobjekte erzeugen mit UIDs

Verbindung zu Bricks und Bricklets herstellen

Hier steht die eigentliche Programmfunktionalität... (nächste Folie)

Wird bei Programmbeendigung ausgeführt

31



Ihr seid dran:

- LCD-Display hinzufügen!

32

Java: Programmrahmen

```

1) import com.tinkerforge.BrickMaster;
2) import com.tinkerforge.BrickletLCD20x4;
3)
4) import com.tinkerforge.IPConnection;
5)
6) public class Einstieg {
7)     private static final String host = "localhost";
8)     private static final int port = 4223;
9)
10)    private static final String masterUID = "9JnhKoQ8ieS";
11)
12)
13)    public static void main(String args[]) throws Exception {
14)
15)        IPConnection ipcon = new IPConnection(host, port);
16)
17)        final BrickMaster master = new BrickMaster(masterUID);
18)
19)        ipcon.addDevice(master);
20)
21)
22)
23)
24)
25)
26)
27)
28)
29)
30)
31)
32)
33)
34)
35)
36)
37)
38)
39)
40)
41)
42)    System.console().readLine("Press [Enter] to exit\n");
43)
44)        ipcon.destroy();
45)        System.exit(0);
46)    }
47) }

```

Bibliotheken importieren

Verbindungsinformationen

UIDs der Bricks und Bricklets

Brick- und Brickletobjekte erzeugen mit UIDs

Verbindung zu Bricks und Bricklets herstellen

Hier steht die eigentliche Programmfunktionalität... (nächste Folie)

Wird bei Programmbeendigung ausgeführt

33



Java: Programmrahmen

```

1) import com.tinkerforge.BrickMaster;
2) import com.tinkerforge.BrickletLCD20x4;
3)
4) import com.tinkerforge.IPConnection;
5)
6) public class Einstieg {
7)     private static final String host = "localhost";
8)     private static final int port = 4223;
9)
10)    private static final String masterUID = "9JnhKoQ8ieS";
11)    private static final String lcdUID = "94h";
12)
13)    public static void main(String args[]) throws Exception {
14)
15)        IPConnection ipcon = new IPConnection(host, port);
16)
17)        final BrickMaster master = new BrickMaster(masterUID);
18)        final BrickletLCD20x4 lcd = new BrickletLCD20x4(lcdUID);
19)
20)        ipcon.addDevice(master);
21)
22)        Hier steht die eigentliche Programmfunktionalität... (nächste Folie)
23)
24)        System.console().readLine("Press [Enter] to exit\n");
25)
26)        ipcon.destroy();
27)        System.exit(0);
28)    }
29) }
30) }

```

Bibliotheken importieren

Verbindungsinformationen

UIDs der Bricks und Bricklets

Brick- und Brickletobjekte erzeugen mit UIDs

Verbindung zu Bricks und Bricklets herstellen

Wird bei Programmbeendigung ausgeführt

34

Java: Programmrahmen

```

1) import com.tinkerforge.BrickMaster;
2) import com.tinkerforge.BrickletLCD20x4;
3)
4) import com.tinkerforge.IPConnection;
5)
6) public class Einstieg {
7)     private static final String host = "localhost";
8)     private static final int port = 4223;
9)
10)    private static final String masterUID = "9JnhKoQ8ieS";
11)    private static final String lcdUID = "94h";
12)
13)    public static void main(String args[]) throws Exception {
14)
15)        IPConnection ipcon = new IPConnection(host, port);
16)
17)        final BrickMaster master = new BrickMaster(masterUID);
18)        final BrickletLCD20x4 lcd = new BrickletLCD20x4(lcdUID);
19)
20)        ipcon.addDevice(master);
21)
22)        Hier steht die eigentliche Programmfunktionalität... (nächste Folie)
23)
24)        System.console().readLine("Press [Enter] to exit\n");
25)
26)        ipcon.destroy();
27)        System.exit(0);
28)    }
29) }
30) }

```

Bibliotheken importieren

Verbindungsinformationen

UIDs der Bricks und Bricklets

Brick- und Brickletobjekte erzeugen mit UIDs

Verbindung zu Bricks und Bricklets herstellen

Wird bei Programmbeendigung ausgeführt

35



Java: Programmrahmen

```

1) import com.tinkerforge.BrickMaster;
2) import com.tinkerforge.BrickletLCD20x4;
3)
4) import com.tinkerforge.IPConnection;
5)
6) public class Einstieg {
7)     private static final String host = "localhost";
8)     private static final int port = 4223;
9)
10)    private static final String masterUID = "9JnhKoQ8ieS";
11)    private static final String lcdUID = "94h";
12)
13)    public static void main(String args[]) throws Exception {
14)
15)        IPConnection ipcon = new IPConnection(host, port);
16)
17)        final BrickMaster master = new BrickMaster(masterUID);
18)        final BrickletLCD20x4 lcd = new BrickletLCD20x4(lcdUID);
19)
20)        ipcon.addDevice(master);
21)        ipcon.addDevice(lcd);
22)
23)        Hier steht die eigentliche Programmfunktionalität... (nächste Folie)
24)
25)        System.console().readLine("Press [Enter] to exit\n");
26)
27)        ipcon.destroy();
28)        System.exit(0);
29)    }
30) }
31) }

```

Bibliotheken importieren

Verbindungsinformationen

UIDs der Bricks und Bricklets

Brick- und Brickletobjekte erzeugen mit UIDs

Verbindung zu Bricks und Bricklets herstellen

Wird bei Programmbeendigung ausgeführt

36

Java: Programmrahmen

```

1) import com.tinkerforge.BrickMaster;
2) import com.tinkerforge.BrickletLCD20x4;
3)
4) import com.tinkerforge.IPConnection;
5)
6) public class Einstieg {
7)     private static final String host = "localhost";
8)     private static final int port = 4223;
9)
10)    private static final String masterUID = "9JnhKoQ8ieS";
11)    private static final String lcdUID = "94h";
12)
13)    public static void main(String args[]) throws Exception {
14)
15)        IPConnection ipcon = new IPConnection(host, port);
16)
17)        final BrickMaster master = new BrickMaster(masterUID);
18)        final BrickletLCD20x4 lcd = new BrickletLCD20x4(lcdUID);
19)
20)        ipcon.addDevice(master);
21)        ipcon.addDevice(lcd);
22)
23)        Hier steht die eigentliche Programmfunktionalität... (nächste Folie)
24)
25)        System.console().readLine("Press [Enter] to exit\n");
26)
27)        lcd.backlightOff();
28)        ipcon.destroy();
29)        System.exit(0);
30)    }
31) }
32) }

```

Bibliotheken importieren

Verbindungsinformationen

UIDs der Bricks und Bricklets

Brick- und Brickletobjekte erzeugen mit UIDs

Verbindung zu Bricks und Bricklets herstellen

Wird bei Programmbeendigung ausgeführt

37



Checkliste

Füllt die zweite Checkliste aus!

38

Das eigentliche Programm

39



Listener, der auf Knopfdruck achtet,
einbauen

40

Java: Programm

```
22)
23)
24)
25)
26)
27)
28)
29)
30)
31)
32)
33)
34)
35)
36)
37)
38)
39)
40)
41)
```

41



Java: Programm

```
22)
23)
24) lcd.addListener(new BrickletLCD20x4.ButtonPressedListener() {
25)
26)     public void buttonPressed(short button) {
27)
28)
29)
30)
31)
32)
33)
34)
35)
36)
37)
38)
39)     }
40)
41) };
```

LCD-Listener,
der auf
Knopfdruck
Achtet

42

Licht einschalten

Welcher Button wurde gedrückt?

Bei Knopfdruck Licht einschalten

43



Java: Programm

```
22)
23)
24) lcd.addListener(new BrickletLCD20x4.ButtonPressedListener() {
25)
26)     public void buttonPressed(short button) {
27)
28)
29)
30)
31)
32)
33)
34)
35)
36)
37)
38)
39)     }
40)
41) };
```

LCD-Listener,
der auf
Knopfdruck
Achtet

44

Java: Programm

```
22) lcd.backlightOn();
23)
24) lcd.addListener(new BrickletLCD20x4.ButtonPressedListener() {
25)
26)     public void buttonPressed(short button) {
27)         System.out.println( "Gedrueckt: " + button );
28)         lcd.backlightOn();
29)
30)
31)
32)
33)
34)
35)
36)
37)
38)
39)     }
40)
41) };
```

LCD-Listener,
der auf
Knopfdruck
Achtet

45



Licht-Aus Schalter einbauen

46

Java: Programm

```
22) lcd.backlightOn();
23)
24) lcd.addListener(new BrickletLCD20x4.ButtonPressedListener() {
25)
26)     public void buttonPressed(short button) {
27)         System.out.println( "Gedruickt: " + button );
28)         lcd.backlightOn();
29)
30)
31)
32)
33)
34)
35)
36)
37)
38)
39)     }
40)
41) };
```

LCD-Listener,
der auf
Knopfdruck
Achtet

47



Java: Programm

```
22) lcd.backlightOn();
23)
24) lcd.addListener(new BrickletLCD20x4.ButtonPressedListener() {
25)
26)     public void buttonPressed(short button) {
27)         System.out.println( "Gedruickt: " + button );
28)         lcd.backlightOn();
29)
30)         if ( button == 0 ) {
31)             lcd.backlightOff();
32)         }
33)
34)     }
35)
36)
37)
38) }
39)
40)
41) });
```

LCD-Listener,
der auf
Knopfdruck
Achtet

48

Richtigen Lichtschalter einbauen

49



Java: Programm

```

22) lcd.backlightOn();
23)
24) lcd.addListener(new BrickletLCD20x4.ButtonPressedListener() {
25)
26)     public void buttonPressed(short button) {
27)         System.out.println( "Gedrueckt: " + button );
28)         lcd.backlightOn();
29)
30)
31)         if ( button == 0 ) {
32)
33)             lcd.backlightOff();
34)         }
35)
36)
37)
38)     }
39) }
40)
41) });

```

LCD-Listener,
der auf
Knopfdruck
Achtet

50

Java: Programm

```

22) lcd.backlightOn();
23)
24) lcd.addListener(new BrickletLCD20x4.ButtonPressedListener() {
25)
26)     public void buttonPressed(short button) {
27)         System.out.println( "Gedrueckt: " + button );
28)         lcd.backlightOn();
29)
30)
31)         if ( button == 0 ) {
32)             if ( button == 0 && lcd.isBacklightOn() == true ) {
33)                 lcd.backlightOff();
34)             }
35)
36)
37)
38)     }
39) }
40)
41) });

```

LCD-Listener,
der auf
Knopfdruck
Achtet

51



Problem: Anfrage an LCD (Licht an??)

Jede Anfrage an einen Baustein muss mit einem try-catch-Block gesichert werden.

52

Java: Programm

```

22) lcd.backlightOn();
23)
24) lcd.addListener(new BrickletLCD20x4.ButtonPressedListener() {
25)
26)     public void buttonPressed(short button) {
27)         System.out.println( "Gedrueckt: " + button );
28)         lcd.backlightOn();
29)
30)
31)         if ( button == 0 ) {
32)             if ( button == 0 && lcd.isBacklightOn() == true ) {
33)                 lcd.backlightOff();
34)             }
35)
36)
37)
38)
39)     }
40)
41) }

```

LCD-Listener,
der auf
Knopfdruck
Achtet

53



Java: Programm

```

22) lcd.backlightOn();
23)
24) lcd.addListener(new BrickletLCD20x4.ButtonPressedListener() {
25)
26)     public void buttonPressed(short button) {
27)         System.out.println( "Gedruickt: " + button );
28)         lcd.backlightOn();
29)
30)         try {
31)             if ( button == 0 ) {
32)                 if ( button == 0 && lcd.isBacklightOn() == true ) {
33)                     lcd.backlightOff();
34)                 }
35)             }
36)         } catch (IPConnection.TimeoutException e) {
37)             System.err.println("Fehlermeldung");
38)         }
39)     }
40)
41) };

```

LCD-Listener,
der auf
Knopfdruck
Achtet

Try-catch-
Block, der
eine Anfrage
an
LCD sichert

54

Fragen?

55



Einführungsrunde

- Aufgaben werden erläutert.
- LOS GEHTS!!! Viel Erfolg beim Üben.

56

Einführungsrunde

- Aufgabenstellung verstanden?

57



Wie funktioniert der TinkerForge-Baukasten?

Ihr habt's geschafft.

58

Wie funktioniert der TinkerForge-Baukasten?

Vorstellung + Besprechung einer
Lösung

59



Haus der Zukunft

Jetzt bauen wir das **Haus der Zukunft...**

60

Funktionen „Haus der Zukunft“

- 6 verschiedene Projekte
- Arbeit in Arbeitsgruppen
- Jede AG setzt ein Projekt um
- Selbstständig, kreativ, individuell

61



Funktionsübersicht

- Alarmanlage
- Badezimmer: Feuchtigkeitsüberwachung
- Automatische Jalousien
- Intelligentes Raumklima
- Intelligentes Licht
- Selbst-öffnende Tür



62

AG-Zuteilung

- | | | |
|-----------------------------|-------------------------------|----------------------------|
| • Alarmanlage 😊 | • Feuchtigkeits-überwachung 😊 | • Automatische Jalousien 🟢 |
| • Intelligentes Raumklima 🟢 | • Intelligentes Licht 😊 | • Selbstöffnende Tür 🟢 |

63

🟢 weniger schwierig

😊 normal



Arbeitsablauf

- Zusammenfinden in Gruppen
- Unterlagen und Bauteile entgegennehmen
- Unterlagen durchlesen, Bauteile sichten
- Projekt lösen, Unterlagen befolgen
- Auf kurze Vorstellung eurer Lösung vorbereiten

64

Projektvorstellungen

- Jede Gruppe stellt ihr Projekt kurz vor
- Vorstellung am Arbeitsplatz
- Die anderen Gruppen verteilen sich um den jeweiligen Arbeitsplatz
- Funktionalität demonstrieren
- Fragen beantworten

65



Arbeitsablauf

- Alle Projekte sollen nun zentral gesteuert werden
- Alle Ergebnisse werden zu einem Haus zusammengesetzt
- Neue Arbeitsanweisungen werden ausgegeben

66

Abschlussbesprechung

Vielen Dank

67




Anhang C Teilnehmermaterialien



C.1. Einstiegsphase

Checkliste

Schülerlabor Informatik

**InfoSphere**
World of Informatics

Checkliste
Zum Anschließen eines Bausteins

Zum Anschließen eines Bricklets an den PC müssen folgende Schritte durchgeführt werden:

1. _____
2. _____
3. _____
4. _____

Zum Verwenden eines Bausteins im Programm müssen folgende Schritte durchgeführt werden:

1. _____
2. _____
3. _____
4. _____
5. _____

1




Einstiegsprojekt Leitfaden

Schülerlabor Informatik

Einstiegsprojekt

Leitfaden





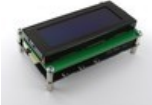




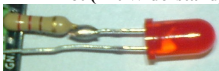
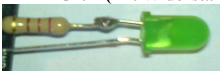


InfoSphere

World of Informatics

Vorbereitung:

1) Sichtet euer Material! Ihr braucht:

<p>2 Laptops</p> 	<p>1 Master Brick</p> 	<p>1 USB Kabel</p> 
<p>4 Bricklet-Kabel</p> 	<p>1 LCD-Bricklet</p> 	<p>1 Linear-Poti Bricklet</p> 
<p>1 Schraubendreher</p> 	<p>1 Temperature-Bricklet</p> 	<p>1 IO4-Bricklet</p> 
<p>1 LED Rot (mit Widerstand)</p> 	<p>1 LED Grün (mit Widerstand)</p> 	

Falls etwas fehlt: Meldet euch bei den Betreuern!

Hinweis: Sollten Bausteine nicht korrekt funktionieren, bekommen sie meist zu wenig Strom. Versucht es mit einem anderem USB-Port des Laptops.

1



Schülerlabor Informatik

Einstiegsprojekt
Leitfaden



InfoSphere
World of Informatics

2) **Auf beiden Laptops gleichzeitig:** Erstellt auf dem Desktop einen Ordner „Einstieg“, ladet die Zip-Datei „Einstieg.zip“ von der Moodle-Seite herunter, entpackt sie und öffnet die Datei Einstieg.java mit dem Java-Editor.

Darin findet ihr den Quelltext, wie er in der Präsentation besprochen wurde.

3) **Los Geht's...auf der nächsten Seite.**



**AUFGABENSTELLUNG: Einstiegsprojekt**

Die Einstiegsaufgabe besteht aus zwei Teilen. Eure Gruppe arbeitet in zwei Zweierteams. Am Ende müssen beide Teilaufgaben zusammengefügt werden, sodass ihr als Gruppe ein gemeinsames Ergebnis habt. Jede Teilaufgabe wird an einem Laptop gelöst. Das Gesamtergebnis muss auf einem beliebigen Laptop laufen. Brick und Bricklets müsst ihr euch in eurer Gruppe teilen. Legt sie zwischen euch, damit ihr nur das USB-Kabel umstecken müsst, wenn ihr euer Programm testen wollt.

Achtet bei der Umsetzung bitte besonders darauf, euren Quelltext gut zu kommentieren. Auch eine gute Einrückungsstruktur hilft beim Lesen des Quelltextes enorm.

Führt die folgenden Schritte an beiden Laptops gleichzeitig aus und sprecht euch ab, nach diesem Arbeitsschritt sollen eure Dokumente identisch sein.

Fügt entsprechend der Checkliste alle 4 Bricklets zu eurem Programm hinzu. Als Beispiel wurde bereits der MasterBrick eingefügt. Hier müsst ihr nur noch die UID anpassen. Alle zu bearbeitenden Stellen sind mit „ToDo“ markiert. In der gedruckten Dokumentation findet ihr die korrekten Namen der Bibliotheken. **Achtet darauf, dass beide Quellcodes danach noch gleich sein müssen.**

Nun arbeiten die Zweierteams an ihrer Aufgabenstellung, ab jetzt schreibt ihr individuelle Teile in euren Quellcode.

DOKUMENTATION: Zusätzlich zur Gedruckten: Öffnet im Webbrowser die Webseite <http://www.tinkerforge.com/doc/index.html>

Hier findet ihr alle notwendigen Informationen zum Umgang mit euren Bauteilen. Öffnet dazu in der Zeile des jeweiligen Bauteils den Link für Java:

Bricklets

- | | |
|---------------------------------|--|
| • Ambient Light | Modbus, TCP/IP, C/C++, C#, Delphi, Java, PHP, Python, Ruby |
| • Analog In | Modbus, TCP/IP, C/C++, C#, Delphi, Java, PHP, Python, Ruby |
| • Analog Out | Modbus, TCP/IP, C/C++, C#, Delphi, Java, PHP, Python, Ruby |

(Bildauszug. Ihr findet dort alle Bricks und Bricklets)



**Aufgabenstellung Teil A:**

Ziel:

- Per Knopfdruck eines LCD-Display-Buttons soll das Licht des LCD-Displays ein/aus geschaltet werden können.
- Durch Drücken eines weiteren Knopfes soll das LCD-Display die aktuell gemessene Temperatur anzeigen.

Löst die Aufgabe in diesen Teilschritten und testet das Programm nach jedem Zwischenschritt:

1. Das LCD-Licht soll eingeschaltet werden, wenn das Programm startet.
2. Das LCD-Licht soll ausgeschaltet werden, wenn das Programm beendet wird.
3. Beim Drücken des ersten Buttons soll das Licht umgeschaltet werden (an → aus, aus → an).

(Die Button findet ihr an der Seite der unteren Etage des LCDs als kleine Taster. Der linke Taster ist der erste.)

4. Wenn das Programm gestartet wird, soll das LCD geleert werden. Keine Schrift soll mehr angezeigt werden.
5. Wenn das Programm beendet wird, soll das LCD geleert werden.
6. Beim Drücken des zweiten Buttons soll das LCD die aktuelle Temperatur ausgeben.
7. Das LCD soll für den Benutzer Hinweise zu den Knöpfen anzeigen. („Button 1 = Licht, Button 2 = Temperatur“)
8. Fügt euer funktionierendes Programm mit dem eurer Gruppenpartner zusammen zu einem funktionierendem Programm. Dateitransfer per Moodle: Auf der Moodle-Webseite findet ihr unter Abschnitt 2 „Einstieg-Dateiaustausch“ einen Link. Klickt darauf, danach auf den Abschnitt „Eintrag hinzufügen“. Ladet dort eine eure Java-Dateien mit einem wiedererkennbaren Namen hoch. Das andere Zweierteam lädt diese Datei dann runter und fügt den Code bei sich ein.
9. Erklärt der anderen Zweiergruppe euren Quellcode.
10. Zeigt einem Betreuer euer Ergebnis.



**Aufgabenstellung Teil B:**

Ziel:

- Die LEDs sollen anzeigen, in welchem der folgenden 4 Bereiche sich das Potentiometer befindet:
- Am oberen oder unteren Ende (LEDs aus), grob in der Mitte (Beide LEDs an), in der oberen Hälfte (Rote LED an), in der unteren Hälfte (Grüne LED an)

Löst die Aufgabe in diesen Teilschritten und testet das Programm nach jedem Zwischenschritt:

1. Beim Programmstart soll auf der Java-Konsole die aktuelle Position des Potentiometers ausgegeben werden.
2. Beim Programmstart sollen alle LEDs kurz aufleuchten. (Für das Schalten der LEDs ist in der ausgedruckten Kurzdokumentation eine Hilfestellung gegeben)
3. Beim Beenden des Programms sollen alle LEDs ausgeschaltet werden.
4. Das Abfrageintervall für den Listener des Potentiometers soll auf 50 ms gesetzt werden.
5. Immer, wenn der Listener die aktuelle Position des Schiebers liefert, soll überprüft werden, in welchem Bereich er sich befindet. Die dementsprechenden LEDs müssen dann leuchten:
 1. Am oberen (Position 90-100) oder unteren (Position 0-10) Ende (LEDs aus), grob in der Mitte (Position 40-60) (Beide LEDs an), in der oberen Hälfte (Rote LED an), in der unteren Hälfte (Grüne LED an)
6. Fügt euer funktionierendes Programm mit dem eurer Gruppenpartner zusammen zu einem funktionierendem Programm. Dateitransfer per Moodle: Auf der Moodle-Webseite findet ihr unter Abschnitt 2 „Einstieg-Dateiaustausch“ einen Link. Klickt darauf, danach auf den Abschnitt „Eintrag hinzufügen“. Ladet dort eine eure Java-Dateien mit einem wiedererkennbaren Namen hoch. Das andere Zweierteam lädt diese Datei dann runter und fügt den Code bei sich ein.
7. Erklärt der anderen Zweiergruppe euren Quellcode.
8. Zeigt einem Betreuer euer Ergebnis.



**Merkblatt Java****Listener:**

Ein Listener überprüft in vorgegebenem Rhythmus einen Sensorwert oder, ob ein bestimmtes Ereignis (Bsp.: Knopf wurde gedrückt; Festgelegter Temperaturwert wurde überschritten) eintrat. Tritt das Ereignis ein, so ruft er eine Methode auf, der er den Wert des Ereignisses übergibt.

```
// Messrhythmus vorgeben mit 1000 ms
temp.setTemperatureCallbackPeriod(1000);

// Listener, der jede Sekunde Temperatur ausgibt
temp.addListener(new BrickletTemperature.TemperatureListener() {

    public void temperature(short temperature) {
        System.out.println("Temperature: " + temperature/100.0 + " °C");
    }

});
```

Try-Catch-Block:

Stellt man eine Anfrage an einen Baustein (Bsp.: Wie ist die Temperatur? Ist das LCD-Licht an?), so muss diese innerhalb eines try-catch-Blockes stehen:

```
try {

    if ( lcd.isBacklightOn() == true ) {
        // Anfrage an Baustein
        lcd.backlightOff();
    }

} catch (IPConnection.TimeoutException e) {

    System.err.println("Fehlermeldung");
}
```

Variablen konvertieren:

Der Datentyp short ist zu verwenden wie int, es passen aber nur kleinere Zahlen hinein.

int y zu short: short x = (short) y;

String y zu short: short x = Short.valueOf(y);

Short y zu String: String x = y.toString();

Für die Folgenden müsst ihr java.lang.Integer importieren: import java.lang.Integer;

int y zu String: String x = toString(y);

String y zu int: Integer.parseInt(y);

Programm warten lassen:

Diese Zeile lässt das Programm 5 Sekunden (angegeben in Millisekunden) warten:

```
Thread.sleep(5000);
```



Kurzdokumentation

Schülerlabor Informatik

Kurzdokumentation
Funktionen Bricks und BrickletsInfoSphere
World of Informatics**Master Brick:**

```
import com.tinkerforge.BrickMaster;
```

```
public class BrickMaster(String uid)
```

Erzeugt ein Objekt mit der eindeutigen Geräte ID *uid*:

```
BrickMaster master = new BrickMaster("YOUR_DEVICE_UID");
```

Dieses Objekt kann danach der IP Connection hinzugefügt werden

Linear Poti Bricklet:

```
import com.tinkerforge.BrickletLinearPoti;
```

```
public class BrickletLinearPoti(String uid)
```

Erzeugt ein Objekt mit der eindeutigen Geräte ID *uid*:

```
BrickletLinearPoti linearPoti = new BrickletLinearPoti("YOUR_DEVICE_UID");
```

Dieses Objekt kann danach der IP Connection hinzugefügt werden (siehe Beispiele [oben](#)).

```
public int getPosition()
```

Gibt die Position des Linearpotentionmeters zurück. Der Wertebereich ist von 0 (Schieberegler unten) und 100 (Schieberegler oben).

Wenn die Position periodisch abgefragt werden soll, wird empfohlen den Listener [PositionListener](#) zu nutzen und die Periode mit [setPositionCallbackPeriod\(\)](#) vorzugeben.

Listener siehe Online-Dokumentation



**LCD20x4 Bricklet:**

```
import com.tinkerforge.BrickletLCD20x4;
```

```
public class BrickletLCD20x4(String uid)
```

Erzeugt ein Objekt mit der eindeutigen Geräte ID *uid*:

```
    BrickletLCD20x4 lcd20x4 = new BrickletLCD20x4("YOUR_DEVICE_UID");
```

Dieses Objekt kann danach der IP Connection hinzugefügt werden (siehe Beispiele [oben](#)).

```
public void writeLine(short line, short position, String text)
```

Schreibt einen Text in die angegebene Zeile (0 bis 3) mit einer vorgegebenen Position (0 bis 19). Der Text kann maximal 20 Zeichen lang sein.

Beispiel: (0, 7, "Hallo") schreibt *Hallo* in die Mitte der ersten Zeile des Display.

```
public void clearDisplay()
```

Löscht alle Zeichen auf dem Display.

```
public void backlightOn()
```

Aktiviert die Hintergrundbeleuchtung.

```
public void backlightOff()
```

Deaktiviert die Hintergrundbeleuchtung.

```
public boolean isBacklightOn()
```

Gibt *true* zurück wenn die Hintergrundbeleuchtung aktiv ist, sonst *false*.

Listener siehe Online-Dokumentation



**Temperature Bricklet:**

```
import com.tinkerforge.BrickletTemperature;
```

```
public class BrickletTemperature(String uid)  
    Erzeugt ein Objekt mit der eindeutigen Geräte ID uid:
```

```
    BrickletTemperature temperature = new  
    BrickletTemperature("YOUR_DEVICE_UID");
```

Dieses Objekt kann danach der IP Connection hinzugefügt werden (siehe Beispiele [oben](#)).

```
public short getTemperature()
```

Gibt die Temperatur des Sensors zurück. Der Wertebereich ist von -2500 bis 8500 und wird in °C/100 angegeben, z.B. bedeutet ein Wert von 4223 eine gemessene Temperatur von 42,23 °C.

Wenn die Temperatur periodisch abgefragt werden soll, wird empfohlen den Listener [TemperatureListener](#) zu nutzen und die Periode mit [setTemperatureCallbackPeriod\(\)](#) vorzugeben.

Listener siehe Online-Dokumentation



**IO4-Bricklet:**

```
import com.tinkerforge.BrickletIO4;
```

```
public class BrickletIO4(String uid)
```

Erzeugt ein Objekt mit der eindeutigen Geräte ID *uid*:

```
BrickletIO4 io4 = new BrickletIO4("YOUR_DEVICE_UID");
```

Dieses Objekt kann danach der IP Connection hinzugefügt werden (siehe Beispiele [oben](#)).

```
public void setValue(short valueMask)
```

Setzt den Ausgangszustand (logisch 1 oder logisch 0) mittels einer Bitmaske. Die Bitmaske hat eine Länge von 4 Bit, *true* bedeutet logisch 1 und *false* logisch 0.

Beispiel: Der Wert 0b0011 setzt die Pins 0-1 auf logisch 1 und die Pins 2-3 auf logisch 0.

```
public short getValue()
```

Gibt eine Bitmaske der aktuell gemessenen Zustände zurück. Diese Funktion gibt die Zustände aller Pins zurück, unabhängig ob diese als Ein- oder Ausgang konfiguriert sind.

```
public void setConfiguration(short pinMask, char direction, boolean value)
```

Konfiguriert den Zustand und die Richtung eines angegebenen Pins. Mögliche Richtungen sind "i" und "o" für Ein- und Ausgang.

Wenn die Richtung als Ausgang konfiguriert ist, ist der Zustand entweder logisch 1 oder logisch 0 (gesetzt als *true* oder *false*).

Wenn die Richtung als Eingang konfiguriert ist, ist der Zustand entweder Pull-Up oder Standard (gesetzt als *true* oder *false*).

Beispiele:

- (15, 'i', true) setzt alle Pins als Eingang mit Pull-Up.
- (8, 'i', false) setzt Pin 3 als Standard Eingang (potentialfrei wenn nicht verbunden).
- (3, 'o', false) setzt die Pins 0 und 1 als Ausgang im Zustand logisch 0.
- (4, 'o', true) setzt Pin 2 als Ausgang im Zustand logisch 1.





```
public BrickletIO4.Configuration getConfiguration()
```

Gibt eine Bitmaske für die Richtung und eine Bitmaske für den Zustand der Pins zurück.

Beispiel: Ein Rückgabewert von 0b0011 und 0b0101 für Richtung und Zustand bedeutet:

- Pin 0 ist als Eingang mit Pull-Up konfiguriert,
- Pin 1 ist als Standard Eingang konfiguriert,
- Pin 2 ist als Ausgang im Zustand logisch 1 konfiguriert
- und Pin 3 ist als Ausgang im Zustand logisch 0 konfiguriert.

Das zurückgegebene Objekt enthält die Public Member Variablen `short directionMask` and `short valueMask`.

Listener siehe Online-Dokumentation

Hilfestellung für das Schalten von LEDs mit dem IO4:

Eine einfache Variante, die LEDs mit Java zu schalten, sieht so aus:

Die erste Zeile schaltet die LED an Pin X ein, die zweite aus:

```
io4.setConfiguration((short)(1 << X), 'o', true);  
io4.setConfiguration((short)(1 << X), 'o', false);
```

Dabei steht das X für den betroffenen Pin,
das 'o' für „output“ (Strom soll fließen),
der letzte Wert für LED-an (true) oder LED-aus (false).

Ein Beispiel, um die LED an Pin 3 einzuschalten:

```
io4.setConfiguration((short)(1 << 3), 'o', true);
```



**Hilfestellung für Alarmanlage (Auslesen, welcher der angeschlossenen Schalter geschlossen ist):**

```
// Es sollen Änderungen der Schalterzustände auf den ersten 3 Ports registriert
// werden (Ports 0-3 in Binärdarstellung 0111 hat Dezimalwert 7)
io4.setInterrupt( (short) 7 );

io4.addListener(new BrickletIO4.InterruptListener() {

    public void interrupt(short interruptMask, short valueMask) {

        // Belegung der Pins:

        // Pin 0 = Tür
        // Pin 1 = Türschloss
        // Pin 2 = Fenster
        // Pin 3 = LED

        System.out.println("Dezimalwert der Portzustände: " + valueMask);

        // ValueMask muss in Binärdarstellung umgerechnet werden (4-stellige
        // Binärzahl). Die Ziffer rechts-außen steht für den Pin 0. Jede Ziffer
        // steht für einen Pin.

    }

});
```



Schülerlabor Informatik

Kurzdokumentation
Funktionen Bricks und Bricklets



InfoSphere
World of Informatics

Alle weiteren verwendeten Bausteine in unsortierter Reihenfolge



**Stepper Brick:**

```
import com.tinkerforge.BrickStepper;
```

```
public class BrickStepper(String uid)  
    Erzeugt ein Objekt mit der eindeutigen Geräte ID uid:
```

```
    BrickStepper stepper = new BrickStepper("YOUR_DEVICE_UID");
```

Dieses Objekt kann danach der IP Connection hinzugefügt werden (siehe Beispiele [oben](#)).

```
public void setMaxVelocity(int velocity)
```

Setzt die maximale Geschwindigkeit des Schrittmotors in Schritten je Sekunde. Diese Funktion startet *nicht* den Motor, sondern setzt nur die maximale Geschwindigkeit auf welche der Schrittmotor beschleunigt wird. Um den Motor zu fahren können [setTargetPosition\(\)](#), [setSteps\(\)](#), [driveForward\(\)](#) oder [driveBackward\(\)](#) verwendet werden.

```
public int getMaxVelocity()
```

Gibt die Geschwindigkeit zurück, wie von [setMaxVelocity\(\)](#) gesetzt.

```
public int getCurrentVelocity()
```

Gibt die *aktuelle* Geschwindigkeit des Schrittmotors in Schritten je Sekunde zurück.

```
public void setSpeedRamping(int acceleration, int deceleration)
```

Setzt die Beschleunigung und die Verzögerung des Schrittmotors. Die Werte müssen in *Schritten/s²* angegeben werden. Eine Beschleunigung von 1000 bedeutet, dass jede Sekunde die Geschwindigkeit um 1000 *Schritte/s* erhöht wird.

```
public BrickStepper.SpeedRamping getSpeedRamping()
```

Gibt die Beschleunigung und Verzögerung zurück, wie von [setSpeedRamping\(\)](#) gesetzt.

Das zurückgegebene Objekt enthält die Public Member Variablen `int acceleration` and `int deceleration`.

```
public void fullBrake()
```

Führt eine aktive Vollbremsung aus.

Warnung

Diese Funktion ist für Notsituationen bestimmt, in denen ein unverzüglicher Halt





notwendig ist.

Ein Aufruf von `stop()` stoppt den Motor.

`public void setSteps(int steps)`

Setzt die Anzahl der Schritte die der Schrittmotor fahren soll. Positive Werte fahren den Motor vorwärts und negative rückwärts. Dabei wird die Geschwindigkeit, Beschleunigung und Verzögerung, wie mit `setMaxVelocity()` und `setSpeedRamping()` gesetzt, verwendet.

`public int getSteps()`

Gibt die letzten Schritte zurück, wie von `setSteps()` gesetzt.

`public int getRemainingSteps()`

Gibt die verbleibenden Schritte des letzten Aufrufs von `setSteps()` zurück. Beispiel: Wenn `setSteps()` mit 2000 aufgerufen wird und `getRemainingSteps()` aufgerufen wird wenn der Motor 500 Schritte fahren hat, wird 1500 zurückgegeben.

`public void driveForward()`

Fährt den Schrittmotor vorwärts bis `driveBackward()` oder `stop()` aufgerufen wird. Dabei wird die Geschwindigkeit, Beschleunigung und Verzögerung, wie mit `setMaxVelocity()` und `setSpeedRamping()` gesetzt, verwendet.

`public void driveBackward()`

Fährt den Schrittmotor rückwärts bis `driveForward()` oder `stop()` aufgerufen wird. Dabei wird die Geschwindigkeit, Beschleunigung und Verzögerung, wie mit `setMaxVelocity()` und `setSpeedRamping()` gesetzt, verwendet.

`public void stop()`

Stoppt den Schrittmotor mit der Verzögerung, wie von `setSpeedRamping()` gesetzt.

`public void setMotorCurrent(int current)`

Setzt den Strom in mA mit welchem der Motor angetrieben wird. Der minimale Wert ist 100mA, der maximale Wert ist 2291mA und der Standardwert ist 800mA.

`public int getMotorCurrent()`

Gibt den Strom zurück, wie von `setMotorCurrent()` gesetzt.

`public void enable()`





Aktiviert die Treiberstufe. Die Treiberparameter können vor der Aktivierung konfiguriert werden (maximale Geschwindigkeit, Beschleunigung, etc.).

```
public void disable()
```

Deaktiviert die Treiberstufe. Die Konfiguration (Geschwindigkeit, Beschleunigung, etc.) bleibt erhalten aber der Motor wird nicht angesteuert bis eine erneute Aktivierung erfolgt.

```
public boolean isEnabled()
```

Gibt *true* zurück wenn die Treiberstufe aktiv ist, sonst *false*.

```
public void setCurrentPosition(int position)
```

Setzt den aktuellen Schrittwert des internen Schrittzählers. Dies kann benutzt werden um die aktuelle Position auf 0 zu setzen wenn ein definierter Startpunkt erreicht wurde (z.B. wenn eine CNC Maschine eine Ecke erreicht).

```
public int getCurrentPosition()
```

Gibt die aktuelle Position des Schrittmotors in Schritten zurück. Nach dem Hochfahren ist die Position 0. Die Schritte werden bei Verwendung aller möglichen Fahrfunktionen gezählt ([setTargetPosition\(\)](#), [setSteps\(\)](#), [driveForward\(\)](#) oder [driveBackward\(\)](#)). Es ist auch möglich den Schrittzähler auf 0 oder jeden anderen gewünschten Wert zu setzen mit [setCurrentPosition\(\)](#).

```
public void setTargetPosition(int position)
```

Setzt die Zielposition des Schrittmotors in Schritten. Beispiel: Wenn die aktuelle Position des Motors 500 ist und [setTargetPosition\(\)](#) mit 1000 aufgerufen wird, dann verfährt der Schrittmotor 500 Schritte vorwärts. Dabei wird die Geschwindigkeit, Beschleunigung und Verzögerung, wie mit [setMaxVelocity\(\)](#) und [setSpeedRamping\(\)](#) gesetzt, verwendet.

```
public int getTargetPosition()
```

Gibt die letzte Zielposition zurück, wie von [setTargetPosition\(\)](#) gesetzt.

```
public void setStepMode(short mode)
```

Setzt den Schrittmodus des Schrittmotors. Mögliche Werte sind:

- Vollschrirt = 1
- Halbschrirt = 2
- Viertelschrirt = 4
- Achterschrirt = 8



Schülerlabor Informatik

Kurzdokumentation
Funktionen Bricks und Bricklets



InfoSphere
World of Informatics

```
public short getStepMode()
```

Gibt den Schrittmodus zurück, wie von `setStepMode()` gesetzt.



**Ambient Light Bricklet:**

```
import com.tinkerforge.BrickletAmbientLight;
```

```
public class BrickletAmbientLight(String uid)  
    Erzeugt ein Objekt mit der eindeutigen Geräte ID uid:
```

```
    BrickletAmbientLight ambientLight = new  
    BrickletAmbientLight("YOUR_DEVICE_UID");
```

Dieses Objekt kann danach der IP Connection hinzugefügt werden (siehe Beispiele [oben](#)).

```
public int getIlluminance()
```

Gibt die Beleuchtungsstärke des Umgebungslichtsensors zurück. Der Wertebereich ist von 0 bis 9000 und ist in Lux/10 angegeben, d.h. bei einem Wert von 4500 wurde eine Beleuchtungsstärke von 450 Lux gemessen.

Wenn die Beleuchtungsstärke periodisch abgefragt werden soll, wird empfohlen den Listener [IlluminanceListener](#) zu nutzen und die Periode mit [setIlluminanceCallbackPeriod\(\)](#) vorzugeben.

Listener siehe Online-Dokumentation



**Distance Infrarot Bricklet:**

```
import com.tinkerforge.BrickletDistanceIR;
```

```
public class BrickletDistanceIR(String uid)
```

Erzeugt ein Objekt mit der eindeutigen Geräte ID *uid*:

```
    BrickletDistanceIR distanceIR = new BrickletDistanceIR("YOUR_DEVICE_UID");
```

Dieses Objekt kann danach der IP Connection hinzugefügt werden (siehe Beispiele [oben](#)).

```
public int getDistance()
```

Gibt die gemessene Entfernung des Sensors zurück. Der Wert ist in mm und die möglichen Entfernungsbereiche sind 40 bis 300, 100 bis 800 und 200 bis 1500, in Abhängigkeit vom gewählten IR Sensor.

Wenn die Entfernung periodisch abgefragt werden soll, wird empfohlen den Listener [DistanceListener](#) zu nutzen und die Periode mit [setDistanceCallbackPeriod\(\)](#) vorzugeben.

Listener siehe Online-Dokumentation



**Dual Relay Bricklet:**

```
import com.tinkerforge.BrickletDualRelay;
```

```
public class BrickletDualRelay(String uid)  
    Erzeugt ein Objekt mit der eindeutigen Geräte ID uid:
```

```
    BrickletDualRelay dualRelay = new BrickletDualRelay("YOUR_DEVICE_UID");
```

Dieses Objekt kann danach der IP Connection hinzugefügt werden (siehe Beispiele [oben](#)).

```
public void setState(boolean relay1, boolean relay2)
```

Setzt den Zustand der Relais, *true* bedeutet ein und *false* aus. Beispiel: (true, false) schaltet Relais 1 ein und Relais 2 aus.

Wenn nur eines der Relais gesetzt werden soll und der aktuelle Zustand des anderen Relais nicht bekannt ist, dann kann der Zustand mit [getState\(\)](#) ausgelesen werden.

Laufende Monoflop Timer werden überschrieben wenn diese Funktion aufgerufen wird.

Der Standardwert ist (*false, false*).

```
public BrickletDualRelay.State getState()
```

Gibt den Zustand der Relais zurück, *true* bedeutet ein und *false* aus.

Das zurückgegebene Objekt enthält die Public Member Variablen `boolean relay1` and `boolean relay2`.



**Humidity Bricklet:**

```
import com.tinkerforge.BrickletHumidity;
```

```
public class BrickletHumidity(String uid)
```

Erzeugt ein Objekt mit der eindeutigen Geräte ID *uid*:

```
    BrickletHumidity humidity = new BrickletHumidity("YOUR_DEVICE_UID");
```

Dieses Objekt kann danach der IP Connection hinzugefügt werden (siehe Beispiele [oben](#)).

```
public int getHumidity()
```

Gibt die gemessene Luftfeuchtigkeit des Sensors zurück. Der Wertebereich ist von 0 bis 1000 und wird in %RH/10 angegeben (relative Luftfeuchtigkeit), z.B. bedeutet ein Wert von 421 eine gemessene Luftfeuchtigkeit von 42,1 %RH.

Wenn die Luftfeuchtigkeit periodisch abgefragt werden soll, wird empfohlen den Listener [HumidityListener](#) zu nutzen und die Periode mit [setHumidityCallbackPeriod\(\)](#) vorzugeben.

Listener siehe Online-Dokumentation



Kommentiertes Einstiegsbeispiel

```

1 import com.tinkerforge.BrickMaster;
2 import com.tinkerforge.BrickletLCD20x4;
3
4 import com.tinkerforge.IPConnection;
5
6 public class Einstieg {
7
8     private static final String host = "localhost";
9     private static final int port = 4223;
10
11     private static final String masterUID = "9JnhKoQ8ieS";
12     private static final String lcdUID = "94h";
13
14
15     public static void main(String args[]) throws Exception {
16
17         IPConnection ipcon = new IPConnection(host, port);
18
19         final BrickMaster master = new BrickMaster(masterUID);
20         final BrickletLCD20x4 lcd = new BrickletLCD20x4(lcdUID);
21
22         ipcon.addDevice(master);
23         ipcon.addDevice(lcd);
24
25         // Ab hier laeuft das eigentliche Programm *****
26         lcd.backlightOn();
27
28         lcd.addListener(new BrickletLCD20x4.ButtonPressedListener() {
29
30             public void buttonPressed(short button) {
31                 System.out.println( "Gedrueckt: " + button );
32                 lcd.backlightOn();
33
34                 try {
35                     if ( button == 0 ) {
36                         if ( button == 0 && lcd.isBacklightOn() == true ) {
37                             lcd.backlightOff();
38                         }
39                     }
40                 } catch (IPConnection.TimeoutException e) {
41                     System.err.println("Fehlermeldung");
42                 }
43             }
44         });
45         // Bis hier laeuft das eigentliche Programm *****
46
47         System.console().readLine("Press [Enter] to exit\n");
48         lcd.backlightOff();
49         ipcon.destroy();
50         System.exit(0);
51     }
52 }

```

Bibliotheken importieren für Bricks und Bricklets

Verbindungsinformationen

UIDs der Bricks und Bricklets

Brick- und Brickletobjekte erzeugen mit UIDs

Verbindung zu den Bricks und Bricklets herstellen

LCD-Listener, der auf Knopfdruck achtet


try-catch-Block, der eine Anfrage an LCD sichert

Wird bei Programmbeendung ausgeführt



Leitfaden Einstieg Wiederholung

Schülerlabor Informatik
Einstieg Wiederholung
Leitfaden zum Nachlesen



Einstieg in Tinkerforge

Leitfaden zum Nachschlagen der Eingangssequenz

Umgang mit dem Brickviewer
Vorbereitung und Anleitung für die nächsten Arbeitsschritte

1



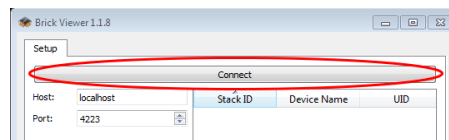


- 1) Öffnet das BrickViewer-Programm: Start → Alle Programme → „Tinkerforge“ → „Brickv 1.1.8“
- 2) Lest euch den Infokasten „BrickViewer“ durch.

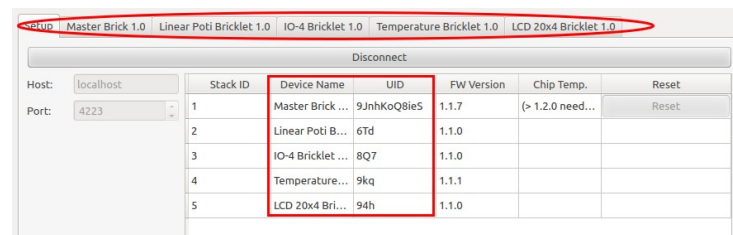
INFO: BrickViewer

Das BrickViewer-Programm dient dazu, die Funktionen leicht ausprobieren zu können und die UID der Bausteine auszulesen. Verwenden kann man die Bausteine auch mit einem eigenen Programm, solange in diesem eigenen Programm die UIDs der Bausteine korrekt eingetragen sind.

- 3) Verbindet den Brick und die Bricklets wie auf der Checkliste angegeben.
- 4) Wartet ein paar Sekunden, der Master-Brick initialisiert sich und blaue LEDs leuchten auf. Danach könnt ihr im BrickViewer auf „Connect“ klicken:



- 5) Nun sollte das Fenster so ähnlich aussehen:



Im Oval: Die Reiter für jeden angeschlossenen Baustein. Im Kasten: Name und UID der Bausteine





6) Schaut euch die neu hinzugekommenen Reiter (rot umkreist) gemeinsam an. Probiert die einzelnen Bausteine kurz aus, um euch mit ihren Funktionen vertraut zu machen:

- Betrachtet den ersten Reiter. Er zeigt für jeden angeschlossenen und erkannten Baustein eine eindeutige UID an. Diese wird verwendet, um das Bauteil eindeutig anzusprechen, wenn man ihm einen Befehl sendet.
- 1. Betrachtet den Verlauf des Poti-Bricklets beim Verschieben des Reglers
- 2. Betrachtet den Temperaturverlauf des Temperature-Bricklets, wenn ihr es anfasst und anpustet
- 3. Schreibt einen Text auf das LCD und schaltet das Licht ein und aus
- Der Reiter für das IO4-Bricklet ist besonders zu beachten:
 - Für die Konfiguration der einzelnen Pins (grüne Lüsterklemmen) des Bricklets sind die im Bild markierten Elemente notwendig. So kann angegeben werden, welcher Pin von der Einstellung betroffen sein soll. Die Debounce-Period könnt ihr vorerst ignorieren, sie gibt das Abfrageintervall an, falls man das Bricklet mit Schaltern verwendet.
 - Die Konfiguration für die Pins mit LEDs muss sein:
 „Direction = Output“
 „Value = High“ → LED leuchtet, „Value = Low“ → LED aus
 Um die eingestellten Werte zu übernehmen, muss rechts auf „Save“ geklickt werden.

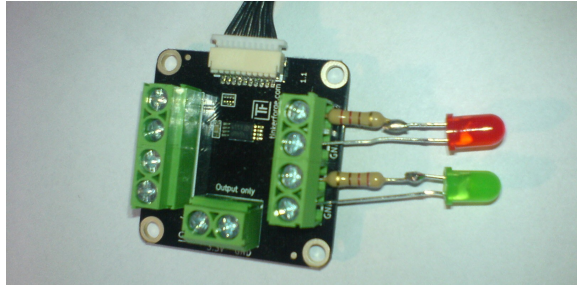
Pin:	0	1	2	3
Value:	High	High	High	High
Direction:	Output	Output	Input	Input
Config:	High	High	Pull Up	Pull Up
Monoflop Time [ms]:	0	0	0	0

Markiert: Die Einstellungen für die Pins. In diesem Beispiel leuchten beide LEDs, die an Pin 1 und Pin 0 angeschlossen sind.





- Ein Konfigurationsbeispiel, um LEDs zum Leuchten zu bringen: Das folgende Bild zeigt, wie die LEDs angeschraubt werden müssen. Der Kontakt mit dem Widerstand muss in einen der nummerierten Ausgangspins gesteckt werden, der andere Kontakt in einen mit „GND“ markierten Pin. Der Widerstand ist nötig, damit die LED nicht durchbrennt:



Quellcode Grundgerüst (Digital ausgegeben)

```
1 import com.tinkerforge.BrickMaster;
2 //TODO
3
4 import com.tinkerforge.IPConnection;
5
6 public class Einstieg {
7     // Verbindungsparameter definieren
8     private static final String host = "localhost";
9     private static final int port = 4223;
10
11     // UIDs der Bausteine speichern
12     private static final String masterUID = "9JnhKoQ8ieS"; // Anpassen
13     //TODO
14
15
16     public static void main(String args[]) throws Exception {
17
18         // Verbindung zur Steuerung aufbauen
19         IPConnection ipcon = new IPConnection(host, port);
20
21         // Brick- und Bricklet-Objekte erschaffen
22         final BrickMaster master = new BrickMaster(masterUID);
23         //TODO
24
25         // Verbindung zu Baustein aufbauen
26         ipcon.addDevice(master);
27         //TODO
28         // Bausteine nicht vorher verwenden
29
30         // Ab hier laeuft das eigentliche Programm *****
31
32
33         //TODO
34
35
36         // Bis hier laeuft das eigentliche Programm *****
37
38         System.console().readLine("Press [Enter] to exit\n");
39         //TODO
40         ipcon.destroy();
41         System.exit(0);
42     }
43 }
```



C.2. Projektarbeitsphase

Projekt Alarmanlage

Schülerlabor Informatik

Projekt Alarmanlage
Leitfaden und Arbeitsanweisungen



InfoSphere
World of Informatics

Vorbereitung:

1) Sichtet euer Material! Zusätzlich zu den Bausteinen aus dem Einstiegsprojekt erhaltet ihr:

Türmodell

Fenstermodell

Beide Modelle bekommt ihr mit einer Einweisung von den Betreuern.

2) Lasst euch von einem Betreuer in die Handhabung der Modelle unterweisen. Geht sorgsam mit den Schaltern und Kabeln um, sie sind sehr empfindlich.

3) Ladet euch von der schon bekannten Moodle-Seite aus dem Abschnitt „Projekt Alarmanlage“ alle Dateien herunter. Entpackt sie in einem dafür neu erstellten Ordner. Die darin enthaltene Datei „Alarmanlage.java“ könnt ihr für euer Projekt verwenden.

4) Nun könnt ihr mit der Arbeit beginnen. Die Arbeitsschritte kennt ihr aus dem Einstiegsprojekt.

Ihr könnt euch ein völlig eigenes Konzept für eine Alarmanlage überlegen. Denkt euch aus, was ihr für praktisch und nützlich haltet und was mit den gegebenen Materialien (Bausteinen und Modellen) erreichbar ist. Als Anregung findet ihr unten einen Konzeptvorschlag für eine „inverse Alarmanlage“. Es ist aber nur eine Idee.

Inverse Alarmanlage

Überwacht, ob beim Verlassen des Hauses alle Fenster/Türen richtig geschlossen sind.

Konzept:

Mit Hilfe von Schaltern soll überprüft werden, ob alle Fenster/Türen korrekt verschlossen sind. Ist ein Schalter nicht gedrückt, dann ist das Fenster offen. Wird bei offenem Fenster die Haustüre abgeschlossen (Schalter im Schließmechanismus), dann soll die Alarm-LED leuchten und auf einem Display eine Warnung ausgegeben werden.

Solltet ihr Schwierigkeiten bei der Umsetzung haben, wendet euch an die Betreuer.





Anleitung für die Verwendung von Schaltern

Das IO4 Bricklet lässt sich auch verwenden um festzustellen, ob ein Schalter gedrückt wird, oder nicht.

Ein Beispiel, wie man das Bricklet für einen Schalter verwenden kann:

Ein Schalter verbindet zwei Kabel, wenn er geschlossen (gedrückt) ist, dann haben beide Kabel Kontakt (Strom kann fließen), sonst nicht. Eines der beiden Kabel muss am entsprechend anderen Ende mit einem nummerierten Pin auf dem IO-Bricklet verbunden sein, das andere mit einem GND-Pin.

Die Konfiguration für den entsprechenden (nummerierten) Pin muss sein:

„Direction = Input“
„Value = Default“

Drückt man nun den Schalter dann wechselt der gemessene Value-Wert zu „Low“. Das bedeutet,

Pin:	0	1	2	3
Value:	High	High	Low	High
Direction:	Output	Output	Input	Input
Config:	High	High	Default	Pull Up
Monoflop Time [ms]:	0	0	0	0

dass der Stromkreis geschlossen bzw. der Schalter betätigt ist. Ein Beispielbild:

In diesem Beispiel ist Pin 2 für einen Schalter konfiguriert, der Schalter ist im Moment der Aufnahme gedrückt. Pin 3 zeigt die Standard-Konfiguration der Pins.

Button auslesen in Java:

In der ausgedruckten Kurzdokumentation (Seite: IO4) findet ihr als Hilfestellung ein Beispiel, wie man in Java auslesen kann, welcher der ans IO4 angeschlossenen Buttons gedrückt ist.



Projekt Feuchtigkeitsmesser

Schülerlabor Informatik

Pr. Feuchtigkeitsmesser
Leitfaden und ArbeitsanweisungenInfoSphere
World of Informatics**Vorbereitung:**

1) Sichtet euer Material! Zusätzlich zu den Bausteinen aus dem Einstiegsprojekt erhaltet ihr:

Humidity-Bricklet**Schaltbare Steckdosen**

Die Steckdosen erhaltet ihr zusammen mit einer Einweisung von den Betreuern.

Heizofen

2) Lasst euch von einem Betreuer in die Handhabung der Steckdosen unterweisen. Geht sorgsam mit den Steckdosen um, sie sind empfindlich. Öffnet das Gehäuse nicht. Achtung im Umgang mit Strom. Bei Problemen oder Beschädigungen der Steckdose wendet euch an die Betreuer.

3) Ladet euch von der schon bekannten Moodle-Seite aus dem Abschnitt „Projekt Feuchtigkeitsmesser“ alle Dateien herunter. Entpackt sie in einem dafür neu erstellten Ordner. Die darin enthaltene Datei „Feuchtigkeitsmesser.java“ könnt ihr für euer Projekt verwenden.

4) Nun könnt ihr mit der Arbeit beginnen. Die Arbeitsschritte kennt ihr aus dem Einstiegsprojekt.

Ihr könnt euch ein völlig eigenes Konzept für die Feuchtigkeitsmessung überlegen. Denkt euch aus, was ihr für praktisch und nützlich haltet und was mit den gegebenen Materialien (Bausteinen und Modellen) erreichbar ist. Als Anregung findet ihr unten einen Konzeptvorschlag. Es ist aber nur eine Idee.

Feuchtigkeitsmesser

Konzept: Feuchtigkeitssensoren sollen in Feuchträumen (Badezimmer) die Luftfeuchtigkeit messen. Steigt die Feuchtigkeit zu stark an und ist dadurch die Schimmelgefahr groß, wird eine Warnung auf einem LCD-Monitor ausgegeben, das Wasser (repräsentiert durch Leuchtdioden) wird ausgeschaltet und, je nach Bedarf, wird die Heizung eingeschaltet.

Solltet ihr Schwierigkeiten bei der Umsetzung haben, wendet euch an die Betreuer.



Projekt Klimasteuerung

Schülerlabor Informatik

Projekt Klimasteuerung
Leitfaden und ArbeitsanweisungenInfoSphere
World of Informatics**Vorbereitung:**

1) Sichtet euer Material! Zusätzlich zu den Bausteinen aus dem Einstiegsprojekt erhaltet ihr:

Temperature-Bricklet**Schaltbare Steckdosen**

Die Steckdosen erhaltet ihr mit einer Einführung von den Betreuern.

Heizofen**Ventilator**

2) Lasst euch von einem Betreuer in die Handhabung der Steckdosen unterweisen. Geht sorgsam mit den Steckdosen um, sie sind empfindlich. Öffnet das Gehäuse nicht. Achtung im Umgang mit Strom. Bei Problemen oder Beschädigungen der Steckdose wendet euch an die Betreuer.

3) Ladet euch von der schon bekannten Moodle-Seite aus dem Abschnitt „Projekt Klimasteuerung“ alle Dateien herunter. Entpackt sie in einem dafür neu erstellten Ordner. Die darin enthaltene Datei „Klimasteuerung.java“ könnt ihr für euer Projekt verwenden.

4) Nun könnt ihr mit der Arbeit beginnen. Die Arbeitsschritte kennt ihr aus dem Einstiegsprojekt.

Ihr könnt euch ein völlig eigenes Konzept für die Klimasteuerung überlegen. Denkt euch aus, was ihr für praktisch und nützlich haltet und was mit den gegebenen Materialien (Bausteinen und Modellen) erreichbar ist. Als Anregung findet ihr unten einen Konzeptvorschlag. Es ist aber nur eine Idee.

Klimasteuerung

Konzept: Mit Hilfe eines Heizofens und eines Ventilators soll die Raumtemperatur

in einem Raum geregelt werden. Zusatz-Option: Manuelle Steuerbarkeit.

- Das Humidity-Bricklet muss nicht verwendet werden, das ist optional.
- Wenn die Temperatur unter 21°C sinkt, dann muss der Lüfter abgeschaltet sein.
- Ab unter 20 °C muss die Heizung eingeschaltet werden.
- Ist die Temperatur über 21°C gestiegen, dann muss die Heizung ausgeschaltet werden.
- Steigt die Temperatur über 24°C, dann muss der Lüfter eingeschaltet werden.

Solltet ihr Schwierigkeiten bei der Umsetzung haben, wendet euch an die Betreuer.



Projekt Lichtsteuerung

Schülerlabor Informatik

Projekt Lichtsteuerung
Leitfaden und ArbeitsanweisungenInfoSphere
World of Informatics**Vorbereitung:**

1) Sichtet euer Material! Zusätzlich zu den Bausteinen aus dem Einstiegsprojekt erhaltet ihr:

AmbientLight-Bricklet**DistanceIR-Bricklet****Schaltbare Steckdosen****Lampen**

Die Steckdosen und Lampen erhaltet ihr mit einer Einführung von den Betreuern.

2) Lasst euch von einem Betreuer in die Handhabung der Steckdosen unterweisen. Geht sorgsam mit den Steckdosen um, sie sind empfindlich. Öffnet das Gehäuse nicht. Achtung im Umgang mit Strom. Bei Problemen oder Beschädigungen der Steckdose wendet euch an die Betreuer.

3) Ladet euch von der schon bekannten Moodle-Seite aus dem Abschnitt „Projekt Lichtsteuerung“ alle Dateien herunter. Entpackt sie in einem dafür neu erstellten Ordner.

4) Die darin enthaltene Datei „Lichtsteuerung.java“ könnt ihr für euer Projekt verwenden. Einige Hilfestellungen und Hinweise sind in den Kommentaren gegeben. Ihr müsst euch aber nicht daran halten.

5) Nun könnt ihr mit der Arbeit beginnen. Die Arbeitsschritte kennt ihr aus dem Einstiegsprojekt.

Ihr könnt euch ein völlig eigenes Konzept für die Lichtsteuerung überlegen. Denkt euch aus, was ihr für praktisch und nützlich haltet und was mit den gegebenen Materialien (Bausteinen und Modellen) erreichbar ist. Als Anregung findet ihr unten einen Konzeptvorschlag. Es ist aber nur eine Idee.

Lichtsteuerung

Konzept: In Abhängigkeit vom Umgebungslicht bzw. der Helligkeit im Raum soll Licht ein- bzw. ausgeschaltet werden. Außerdem kann man das Licht durch Gesten steuern (Winken).

Solltet ihr Schwierigkeiten bei der Umsetzung haben, wendet euch an die Betreuer.



Projekt Jalousiesteuerung

Schülerlabor Informatik

Projekt Jalousiesteuerung
Leitfaden und ArbeitsanweisungenInfoSphere
World of Informatics**Vorbereitung:**

1) Sichtet euer Material! Zusätzlich zu den Bausteinen aus dem Einstiegsprojekt erhaltet ihr:

Stepper-Brick**Fenstermodell**

Das Modell und die Lampen bekommt ihr mit einer Einführung von den Betreuern.

Lampen**2 AmbientLight-Bricklet**

Anmerkung: Solltet ihr Stepper-Brick UND Master-Brick gleichzeitig verwenden, so müsst ihr den Stepper AUF den Master stecken (die weißen Pfosten anknoppeln) und das USB-Kabel weiterhin an den Master anschließen. Es kann auch nur der Stepper-Brick verwendet werden.

2) Lasst euch von einem Betreuer in die Handhabung des Fenstermodells unterweisen. Geht sorgsam damit um, es ist empfindlich. Betreibt den Motor nur mit den vorgegebenen Konfigurationseinstellungen, da er sonst zerstört wird:

- **MotorCurrent** (Stromversorgung) auf 600 (0,6A)
- **MaxVelocity** (Maximalgeschwindigkeit) auf höchstens 100 Schritte/Sekunde
- **SpeedRamping** (Beschleunigung) für Anfahren und Bremsen je auf höchstens 100.

- Der Motor macht nach einer Bewegung Geräusche. Diese verschwinden, sobald er ausgeschaltet (disabled) wird. Dies muss auch nach jeder Bewegungsphase geschehen. Berücksichtigt dies in eurem Programm!

- Die Kupplungsmuffe an der Motorstange (Kupferfarbene Manschette mit 2 Schrauben) könnte sich von Zeit zu Zeit lösen. Ihr könnt die Schrauben der Muffe dann wieder anziehen, achtet darauf, dass die Winde dabei in einer geeigneten Position steht. Holt dafür unbedingt einen Betreuer hinzu!

- Bevor ihr die Motorsteuerung programmiert, solltet ihr im BrickViewer die oben angegebenen Konfigurationseinstellungen übernehmen und die gewünschten Bewegungen erst ausprobieren. Ein „Schritt“ des Schrittmotors ist eine Teilumdrehung der Motorstange. Probiert aus, wie viele Schritte welcher Größe ihr benötigt, um die Ziel- und Endposition des Motors zu erreichen. Diese Werte könnt ihr dann im Java-Programm übernehmen. Probiert zuerst kleine Werte aus. (**Achtung:** Im BrickViewer ist der Begriff „SpeedRamping“ durch die Begriffe „Acceleration“ und „Deceleration“ ersetzt. Tragt für beide Werte ebenfalls 100 ein. Die restlichen Werte sind identisch benannt. „Minimum Motor Voltage“ könnt ihr auf 24 setzen. Alle weiteren hier nicht erwähnten Werte können auf dem Standard-Wert bleiben.)





3) Ladet euch von der schon bekannten Moodle-Seite aus dem Abschnitt „Projekt Jalousiesteuerung“ alle Dateien herunter. Entpackt sie in einem dafür neu erstellten Ordner. Die darin enthaltene Datei „Jalousiesteuerung.java“ könnt ihr für euer Projekt verwenden.

4) Nun könnt ihr mit der Arbeit beginnen. Die Arbeitsschritte kennt ihr aus dem Einstiegsprojekt.

Ihr könnt euch ein völlig eigenes Konzept für die Jalousiesteuerung überlegen. Denkt euch aus, was ihr für praktisch und nützlich haltet und was mit den gegebenen Materialien (Bausteinen und Modellen) erreichbar ist. Als Anregung findet ihr unten einen Konzeptvorschlag. Es ist aber nur eine Idee.

Jalousiesteuerung

Konzept: Eine Jalousie soll automatisch der Uhrzeit oder dem Licht entsprechend gesteuert werden. Lichtsensoren drinnen und draußen liefern die Werte, auf denen die Steuerung basiert. Zusatz-Option: Manuelle Steuerbarkeit.

Solltet ihr Schwierigkeiten bei der Umsetzung haben, wendet euch an die Betreuer.



Projekt Türsteuerung

Vorbereitung:

1) Sichtet euer Material! Zusätzlich zu den Bausteinen aus dem Einstiegsprojekt erhaltet ihr:

Stepper-Brick**Türmodell**

Das Modell bekommt ihr mit einer Einführung von den Betreuern.

2 Distance-IR-Bricklets

Anmerkung: Solltet ihr Stepper-Brick UND Master-Brick gleichzeitig verwenden, so müsst ihr den Stepper AUF den Master stecken (die weißen Pfosten ankoppeln) und das USB-Kabel weiterhin an den Master anschließen. Es kann auch nur der Stepper-Brick verwendet werden.

2) Lasst euch von einem Betreuer in die Handhabung des Türmodells unterweisen. Geht sorgsam damit um, es ist empfindlich. Betreibt den Motor nur mit den vorgegebenen Konfigurationseinstellungen, da er sonst zerstört wird:

- **MotorCurrent** (Stromversorgung) auf 600 (0,6A)
- **MaxVelocity** (Maximalgeschwindigkeit) auf höchstens 100 Schritte/Sekunde
- **SpeedRamping** (Beschleunigung) für Anfahren und Bremsen je auf höchstens 100.

- Der Motor macht nach einer Bewegung Geräusche. Diese verschwinden, sobald er ausgeschaltet (disabled) wird. Dies muss auch nach jeder Bewegungsphase geschehen. Berücksichtigt dies in eurem Programm!

- Die Kupplungsmuffe an der Motorstange (Kupferfarbene Manschette mit 2 Schrauben) könnte sich von Zeit zu Zeit lösen. Ihr könnt die Schrauben der Muffe dann wieder anziehen, achtet darauf, dass die Winde dabei in einer geeigneten Position steht. Holt dafür unbedingt einen Betreuer hinzu!

- Bevor ihr die Motorsteuerung programmiert, solltet ihr im BrickViewer die oben angegebenen Konfigurationseinstellungen übernehmen und die gewünschten Bewegungen erst ausprobieren. Ein „Schritt“ des Schrittmotors ist eine Teilumdrehung der Motorstange. Probiert aus, wie viele Schritte welcher Größe ihr benötigt, um die Ziel- und Endposition des Motors zu erreichen. Diese Werte könnt ihr dann im Java-Programm übernehmen. Probiert zuerst kleine Werte aus. (**Achtung:** Im BrickViewer ist der Begriff „SpeedRamping“ durch die Begriffe „Acceleration“ und „Deceleration“ ersetzt. Tragt für beide Werte ebenfalls 100 ein. Die restlichen Werte sind identisch benannt. „Minimum Motor Voltage“ könnt ihr auf 24 setzen. Alle weiteren hier nicht erwähnten Werte können auf dem Standard-Wert bleiben.)





3) Ladet euch von der schon bekannten Moodle-Seite aus dem Abschnitt „Projekt Türsteuerung“ alle Dateien herunter. Entpackt sie in einem dafür neu erstellten Ordner. Die darin enthaltene Datei „Türsteuerung.java“ könnt ihr für euer Projekt verwenden.

4) Nun könnt ihr mit der Arbeit beginnen. Die Arbeitsschritte kennt ihr aus dem Einstiegsprojekt.

Ihr könnt euch ein völlig eigenes Konzept für die Türsteuerung überlegen. Denkt euch aus, was ihr für praktisch und nützlich haltet und was mit den gegebenen Materialien (Bausteinen und Modellen) erreichbar ist. Als Anregung findet ihr unten einen Konzeptvorschlag. Es ist aber nur eine Idee.

Türsteuerung

Konzept: Mit Hilfe von Abstandssensoren soll gemessen werden, ob sich etwas der Tür nähert. Die Tür soll entsprechend geöffnet und wieder geschlossen werden:

- Ist etwas im Erfassungsbereich → Tür öffnen
- Ist für 5 Sekunden nichts mehr im Erfassungsbereich → Tür schließen

Zusatz-Option: Manuelle Steuerbarkeit.

Solltet ihr Schwierigkeiten bei der Umsetzung haben, wendet euch an die Betreuer.



Bonusprojekt Monitoring

Schülerlabor Informatik

Bonus: Monitoring

Leitfaden und Arbeitsanweisungen



InfoSphere

World of Informatics

Vorbereitung:

1) Sichtet euer Material! Zusätzlich zu den Bausteinen aus dem Einstiegsprojekt erhaltet ihr:

Step-Down-Power-Supply

Anmerkung: Das Step-Down-Power-Supply dient der Stromversorgung von Baustein-Stapeln. Es wird dazu mit dem passenden Netzteil verbunden (schwarze Buchse) und unter den untersten verwendeten Brick gesteckt. Weitere relevante Funktionen bietet es für euch nicht. Ihr werdet es in diesem Arbeitsabschnitt noch nicht unbedingt brauchen. Ihr könnt es dennoch testweise schonmal unter euer Master-Brick stecken.

2) Ladet euch von der schon bekannten Moodle-Seite aus dem Abschnitt „Projekt Monitoring“ alle Dateien herunter. Entpackt sie in einem dafür neu erstellten Ordner. Die darin enthaltene Datei „Monitoring.java“ könnt ihr für euer Projekt verwenden.

3) Nun könnt ihr mit der Arbeit beginnen. Die Arbeitsschritte kennt ihr aus dem Einstiegsprojekt.

4) Ihr könnt euch ein völlig eigenes Konzept für das Monitoring überlegen. Denkt euch aus, was ihr für praktisch und nützlich haltet und was mit den gegebenen Materialien (Bausteinen und Modellen) erreichbar ist.

Allerdings ist es notwendig, dass ihr ein paar Bedingungen einhaltet. Diese findet ihr unter der folgenden Aufgabenstellung.

Monitoring

Ziel: Euer Programm soll dem Hausbewohner auf einen Blick eine Übersicht über alle automatisierten Vorgänge in seinem Haus geben. Es soll eine sich ständig aktualisierende Statusübersicht über alle wichtigen Funktionen und Sensorwerte der einzelnen Projekte erstellt werden, die auf der Java-Konsole ausgegeben wird.

Grundlage dafür sind Dateien, die die anderen Projekte im gleichen Ordner ablegen, in der auch eure Monitoring.java liegt. In diesen Dateien stehen jeweils höchstens vier Zeilen mit dem Status des entsprechenden Projektes. Diese Dateien werden andauernd aktualisiert.

Eine wichtige Funktion eures Programmes ist es außerdem, die anderen Projekt-Programme zuerst zu kompilieren und zu starten. Diese Programme erzeugen anschließend erst die Statusdateien, die ihr auslesen sollt. Wie diese heißen, ist unten aufgelistet.

Weitere Funktionen sind optional. Hier könnt ihr euch was einfallen lassen.



C.3. Optionalphase

Zusammenführung Allgemein

Schülerlabor Informatik

Zusammenführung (Alg.)
Leitfaden und Arbeitsanweisungen



InfoSphere
World of Informatics

Ziel:

Ziel ist nun, dass alle Projekte nun von einem einzigen Laptop aus gesteuert werden. Außerdem soll euer Programm seinen aktuellen Status in eine Datei speichern. Aus dieser Datei liest ein anderes Programm später diesen Status aus und zeigt dann den Zustand aller Projekte auf einer Seite an. Auf diese Weise kann der Hausbesitzer dann auf einen Blick sehen, was in seinem Haus passiert. Ihr könnt selber entscheiden, was zum Status eures Projektes gehört. Beispiele wären:

Tür offen
Temperatur 22°C
Licht aus

Aufgabe:

1) Ergänzt euer bisheriges Programm so, dass es den aktuellen Status eurer Anlage in immer aktueller Form in eine Textdatei speichert.
Eine Anleitung, wie man Informationen in eine Datei schreibt, findet ihr auf der nächsten Seite.

Die Datei soll vom Typ „.txt“ sein und muss den Namen eures Projektes tragen. Haltet euch an die vorgegebene Namensgebung.
Sie soll also beispielsweise:
„Alarmanlage.txt“ oder „Tueroeffner.txt“ heißen. Nur der erste Buchstabe wird groß, alle anderen klein geschrieben.
Sie soll in dem Ordner angelegt werden, in dem auch euer Java-Programm liegt.

Beachtet bitte: So, wie der Text in der Datei aussieht, bekommt ihn später auch der Hausbesitzer zu sehen. Außerdem: Der Status-Text darf nicht länger als 4 Zeilen sein, alle 4 Zeilen dürfen höchstens 20 Zeichen lang sein.

Testet dann euer Programm und überprüft, ob der Status wie gewünscht in der Textdatei steht.

2) Wenn alles funktioniert, ändert **zuletzt** die UID eures Masterbricks. Holt euch die neue UID für das MasterBrick von der Monitoring-Gruppe oder alternativ von einem Betreuer. Tragt die UID ein und speichert die Java-Datei.

3) Anschließend stellt ihr eure Java-Datei (ohne weitere Dateien, also auch ohne die angelegte .txt-Statusdatei) über Moodle der Allgemeinheit zur Verfügung.

4) Bringt euer gesamtes Projekt ohne das Laptop zu dem Tisch, den euch die Betreuer genannt haben. Dort baut ihr dann gemeinsam mit den anderen Gruppen das komplette Haus zusammen, indem ihr die Bodenplatten aneinander stellt.
Sammelt alle eure MasterBricks neben dem Jalousieprojekt.
Bei Bedarf könnt ihr andere Bricklet-Kabel verwenden. Die Betreuer werden euch helfen.





Daten in eine Datei schreiben

Wenn ihr Daten in eine Datei, beispielsweise eine Textdatei (.txt) schreiben wollt, so müsst ihr im entsprechenden Java-Programm zuerst die Bibliothek importieren, die diese Funktionen bereitstellt:

```
import java.io.*;
```

Anschließend könnt ihr folgendes Grundgerüst verwenden.

Angepasst werden muss noch der Dateiname („eureDatei.txt“) in der grünen Zeile. Wenn ihr die Datei an einen bestimmten Ort speichern wollt, könnt ihr vor dem Dateinamen den Pfad zur Datei angeben.

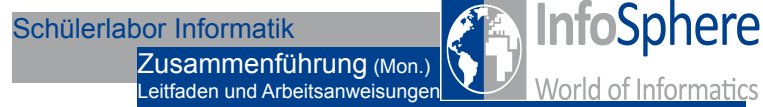
Außerdem angepasst werden muss der Text bzw. die Daten, die ihr in die Datei speichern wollt (blaue Zeile). Der Zeilenumbruch wird durch die im Beispiel rot markierte Zeile realisiert.

Den Rest könnt ihr übernehmen. Wollt ihr mehr als eine Zeile schreiben, könnt ihr einfach mehrere der blauen (und roten) Zeilen nacheinander schreiben.

```
try {  
    Writer fw = null;  
    fw = new FileWriter( "eureDatei.txt" );  
    fw.flush();  
    fw.write( "euer Text " + variablenname );  
    fw.append( System.getProperty("line.separator") ); // e.g. "\n"  
    if ( fw != null ) {  
        try {  
            fw.close();  
        } catch ( IOException e ) {  
            e.printStackTrace();  
        }  
    }  
}  
  
} catch ( IOException e ) {  
    System.err.println( "Konnte Datei nicht erstellen" );  
}
```



Zusammenführung Monitoringgruppe

**Ziel:**

Ziel ist nun, dass alle Projekte nun von einem einzigen Laptop aus gesteuert werden. Wie bereits in der letzten Aufgabenstellung angekündigt, bekommt ihr nun bald von den anderen Gruppen deren Programme. Diese brauchen jedoch zuerst die UID eures Masters. Sie werden euch danach fragen. Gebt sie ihnen.

Aufgabe:

- 1) Bringt euer gesamtes Projekt MIT dem Laptop zu dem Tisch, den euch die Betreuer genannt haben. Dort baut ihr dann gemeinsam mit den anderen Gruppen das komplette Haus zusammen, indem ihr die Bodenplatten aneinander stellt.
Sammelt alle eure MasterBricks neben dem Jalousieprojekt.
Bei Bedarf könnt ihr andere Bricklet-Kabel verwenden. Die Betreuer werden euch helfen.
- 2) Euer Master-Brick muss das unterste sein. Alle anderen Gruppen müssen ihre Bricks in beliebiger Reihenfolge auf euren Master aufstecken. Helft ihnen dabei.
- 3) Als unterster Baustein des Brick-Stapels kommt nun die Step-Down-Power-Supply. Schließt sie an das Netzteil an, sodass der Stapel mit Strom versorgt wird.
- 4) Sobald alle Gruppen ihre fertige Software im Moodle hochgeladen haben, könnt ihr die Java-Dateien herunterladen. Bringt sie in dem Ordner unter, in dem auch eure Monitoring.Java liegt.
- 5) Sobald das gesamte Haus zusammengebaut wurde und alle Bausteine angeschlossen sind, könnt ihr den USB-Stecker eures Master-Bricks an euer Laptop anschließen und dann euer Programm starten. Jetzt werden alle Projekte von eurem Rechner zentral gesteuert.



Anhang D Betreuermaterialien



Betreuerleitfaden

Schülerlabor Informatik

Betreuerleitfaden
Anleitung für die Durchführung



InfoSphere
World of Informatics

Reihenfolge der Arbeitsschritte

1. Präsentation
2. Folie 15: Checkliste „Baustein anschließen“ verteilen. Material austeilen. Leitfaden austeilen. Zeigen, wie man Bricklets anschließt (An Merkblatt orientieren, SuS müssen das aufschreiben)
3. Folie 16: BrickViewer zeigen, Bausteine anschließen und verbinden, UID-Seite zeigen, alle Schritte in SuS-Wiederholungs-Leitfaden durchlaufen
4. Ab Folie 17: Programmbeispiel durchsprechen
6. Folie 38: SuS Checkliste zur Verwendung eines Bricklets ausfüllen lassen
7. Folie 56: Struktur des Einstiegsleitfadens erklären
8. SuS können Einstiegsprojekt starten
9. Nachdem alle Gruppen im Wesentlichen fertig sind: Eine SuS-Lösung am Beamer besprechen
10. Präsentation weiterführen ab Folie 60
11. Folie 63: AG-Zuteilung
12. Arbeitsphase Projekte, Material verteilen
Sollten Gruppen deutlich früher fertig sein, als der Rest:
 - Wenn die verbleibende Zeit das Bearbeiten der Optionalphase wahrscheinlich macht: Bonusaufgabe Monitoring verteilen.
 - Wenn die Optionalphase vermutlich nicht mehr bearbeitet werden kann: Die fertigen SuS zur Optimierung ihrer Projekte anregen.
13. Nach der Arbeitsphase (Folie 65): Gruppen stellen sich ihre Ergebnisse vor. Jede Gruppe führt ihr Ergebnis kurz vor. Alle anderen Begeben sich zum jeweiligen, gerade vorgestellten Projekt.
14. Abschluss: Zentralsteuerung. Aufgabenblatt „Zusammenführung Allgemein“ an alle austeilen. Falls keine Gruppe die Bonusaufgabe bearbeitet hat: Musterlösung verwenden und vorbereiten. Falls die Bonusaufgabe bearbeitet wurde: Für die entsprechenden Teilnehmer das Arbeitsblatt „Zusammenführung Monitoring“
15. SuS bauen von Betreuern angeleitet das gesamte Haus auf. Musterlösung Monitoring wird verwendet, falls keine geeignete SuS-Lösung vorliegt.
16. Abschlussbesprechung





Materialübersicht

Betreuermaterialien:

1. Präsentation
2. Betreuerleitfaden (Das vorliegende Dokument)
3. Grafischer Verlaufsplan SLI-Modul
4. Grafischer Verlaufsplan Materialausgabe
5. Betreuermerkblatt „Mikrocontroller und Co“
6. Betreuermerkblatt Java
7. Musterlösung Checklisten
8. Gedruckte kommentierte Übersicht „Einstieg – Java-Programm“
9. Gedruckte Musterlösung „Einstieg_Musterlösung“
10. Hardware Packliste

Digital:

1. Betreuer-Wiki
2. Musterlösungen Quellcodes

SuS-Materialien für die Einstiegsphase:

- | | |
|--|---------------|
| 1. Checkliste „Verwendung der Bausteine“ | 1x/Teilnehmer |
| 2. Einstieg Leitfaden | 2x/4er-Gruppe |
| 3. Gedruckte Kurzdokumentation Tinkerforge | 2x/4er-Gruppe |
| 4. Leitfaden Einstieg-Wiederholung | 1x/4er-Gruppe |
| 5. Gedruckte kommentierte Übersicht „Einstieg – Java-Programm“ | 2x/4er-Gruppe |

Digital:

1. Quelltext (Im Moodle)
2. Tinkerforge Online-Dokumentation

SuS-Materialien für die Projektphase:

- Zu jedem Projekt gehört ein entsprechendes Aufgabenblatt. Dort ist die zugehörige zu verteilende Hardware vermerkt. 1x/Projektgruppe
- Arbeitsblatt „Bonusaufgabe Monitoring“ 1x/fertige Gruppe

SuS-Materialien für die Optionalphase:

- Für alle Teilnehmer: Arbeitsblatt „Zusammenführung Allgemein“ 1x/Projektgruppe
- Für Teilnehmer, die die Bonusaufgabe bearbeitet haben, zusätzlich: „Zusammenführung Monitoring“



Kommentiertes Einstiegsbeispiel

```

1 import com.tinkerforge.BrickMaster;
2 import com.tinkerforge.BrickletLCD20x4;
3
4 import com.tinkerforge.IPConnection;
5
6 public class Einstieg {
7
8     private static final String host = "localhost";
9     private static final int port = 4223;
10
11     private static final String masterUID = "9JnhKoQ8ieS";
12     private static final String lcdUID = "94h";
13
14
15     public static void main(String args[]) throws Exception {
16
17         IPConnection ipcon = new IPConnection(host, port);
18
19         final BrickMaster master = new BrickMaster(masterUID);
20         final BrickletLCD20x4 lcd = new BrickletLCD20x4(lcdUID);
21
22         ipcon.addDevice(master);
23         ipcon.addDevice(lcd);
24
25         // Ab hier laeuft das eigentliche Programm *****
26         lcd.backlightOn();
27
28         lcd.addListener(new BrickletLCD20x4.ButtonPressedListener() {
29
30             public void buttonPressed(short button) {
31                 System.out.println( "Gedrueckt: " + button );
32                 lcd.backlightOn();
33
34                 try {
35                     if ( button == 0 ) {
36                         if ( button == 0 && lcd.isBacklightOn() == true ) {
37                             lcd.backlightOff();
38                         }
39                     }
40                 } catch (IPConnection.TimeoutException e) {
41                     System.err.println("Fehlermeldung");
42                 }
43             }
44         });
45         // Bis hier laeuft das eigentliche Programm *****
46
47         System.console().readLine("Press [Enter] to exit\n");
48         lcd.backlightOff();
49         ipcon.destroy();
50         System.exit(0);
51     }
52 }

```

Bibliotheken importieren für Bricks und Bricklets

Verbindungsinformationen

UIDs der Bricks und Bricklets

Brick- und Brickletobjekte erzeugen mit UIDs

Verbindung zu den Bricks und Bricklets herstellen

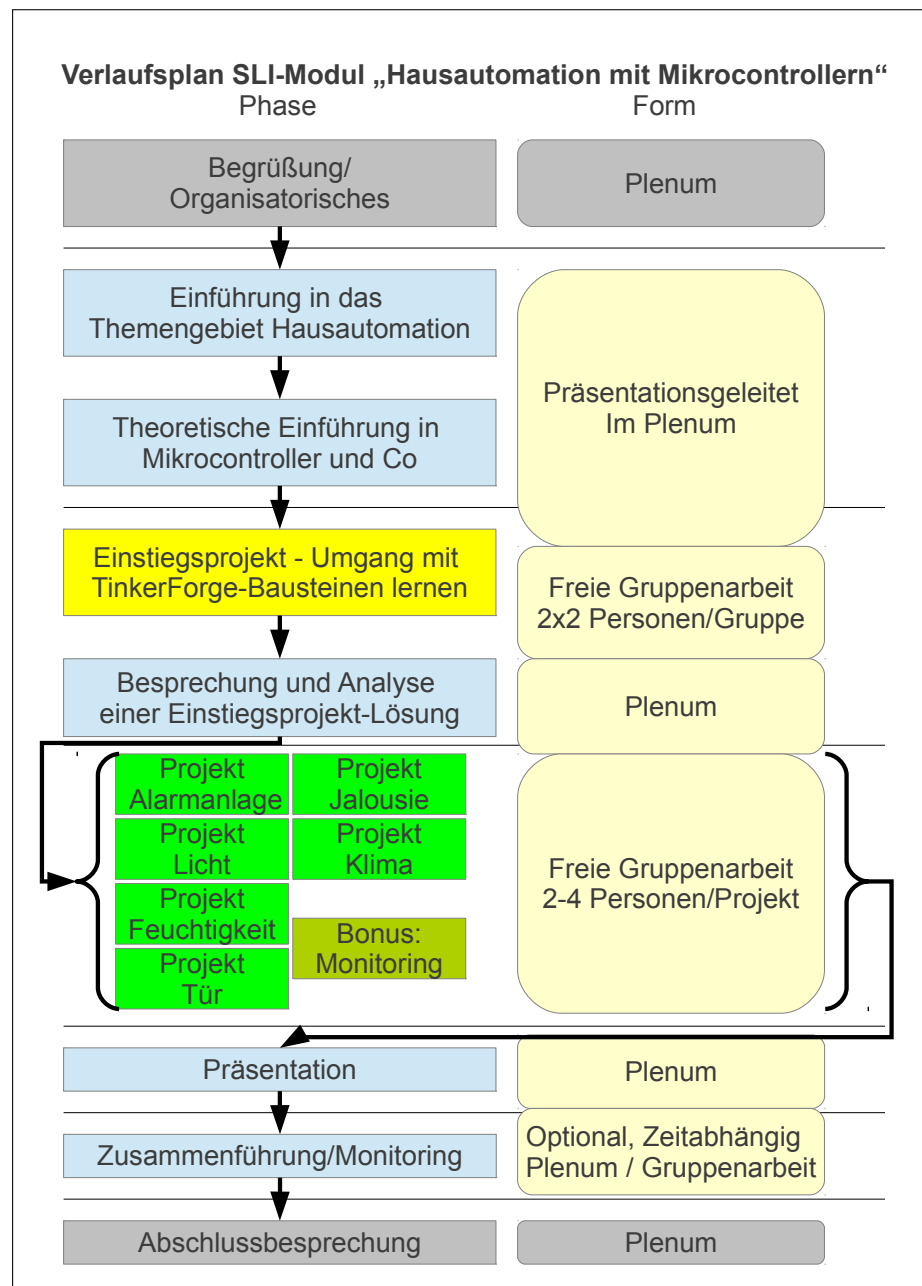
LCD-Listener, der auf Knopfdruck achtet

try-catch-Block, der eine Anfrage an LCD sichert

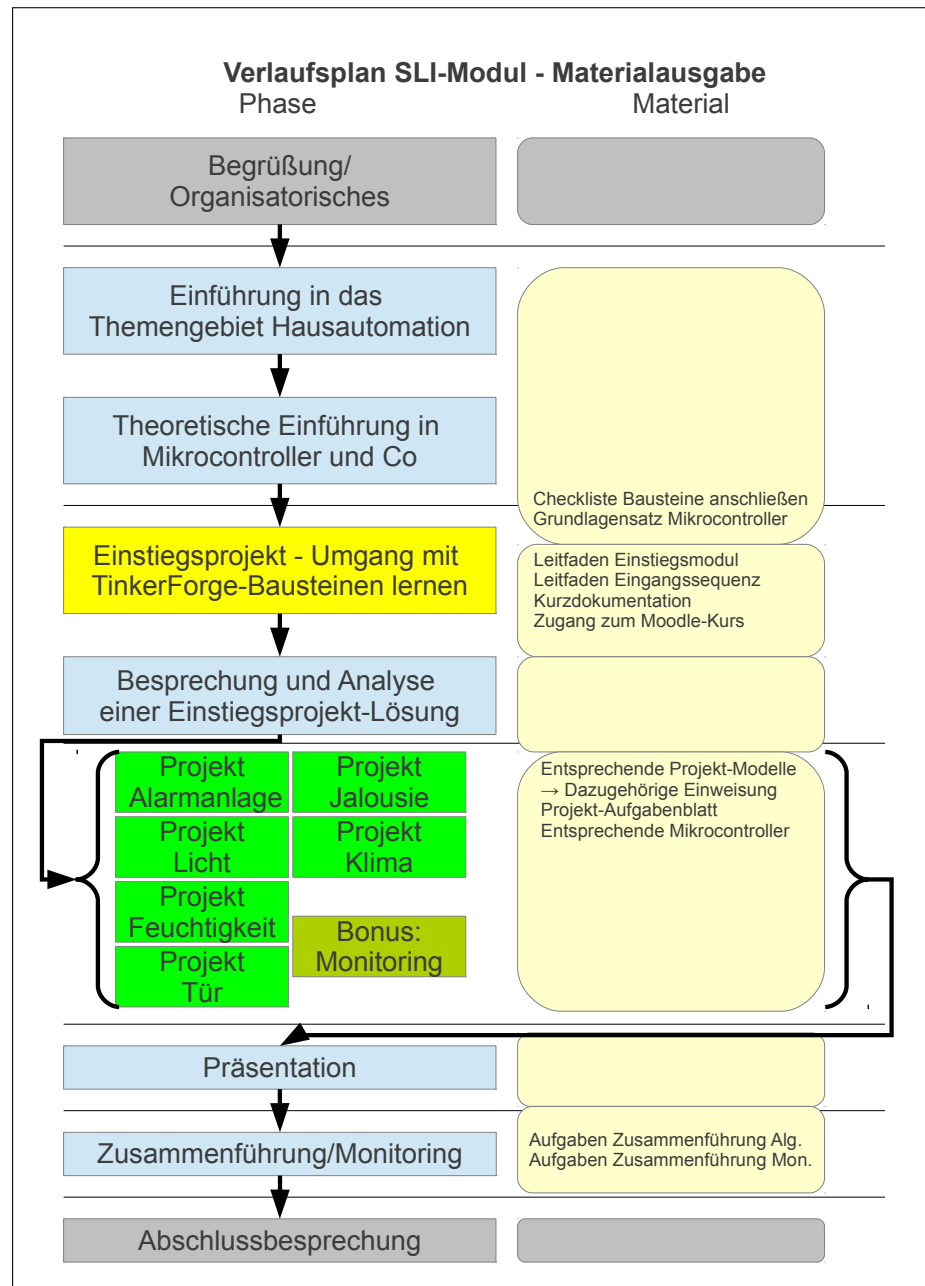
Wird bei Programmbeendung ausgeführt



Verlaufsplan SLI-Modul



Verlaufsplan Materialausgabe



Merkblatt Mikrocontroller

Schülerlabor Informatik

Mikrocontroller und Co
Merkblatt für Betreuer



InfoSphere
World of Informatics

Prozessor: Eine Maschine oder eine elektronische Schaltung, welche gemäß übergebener Befehle andere Maschinen oder elektrische Schaltungen steuert und dabei einen Prozess oder Algorithmus vorantreibt, was meist Datenverarbeitung beinhaltet. Am populärsten sind Prozessoren als zentrale Recheneinheiten von Computern, in denen sie Befehle (Software) ausführen.

Mikroprozessor: Moderne Form des Prozessors, der alle Bausteine des Prozessors auf einem Chip vereinigt.

Chip: Plättchen aus einem Halbleitermaterial, auf dem sich ein vollständiges, funktionsfähiges elektronik-Bauteil befindet.

Platine: Eine Leiterplatte (oder gedruckte Schaltung) ist ein Träger für elektronische Bauteile. Sie dient der mechanischen Befestigung und elektrischen Verbindung. Nahezu jedes elektronische Gerät enthält eine oder mehrere Leiterplatten.

Mikrocontroller: Als Mikrocontroller werden Halbleiterchips bezeichnet, die mit dem Prozessor auch Peripheriefunktionen auf einem Chip vereinen. In vielen Fällen befindet sich der Arbeits- und Programmspeicher ebenfalls teilweise oder komplett auf dem selben Chip. Ein Mikrocontroller ist praktisch ein Ein-Chip-Computersystem.

System-on-a-Chip (SoC): Unter einem SoC versteht man die Integration aller oder eines großen Teils der Funktionen eines Systems auf einem Chip. Als *System* wird dabei eine Kombination unterschiedlicher Elemente (logischen Schaltungen, Taktgebung, selbständiges Anlaufen, usw.) aufgefasst, die zusammen eine bestimmte Funktionalität bereitstellen, beispielsweise ein Temperatursensor samt Auswertungs elektronik.

Firmware: Unter Firmware (von engl. „firm“ = fest) versteht man Software, die fest in elektronische Geräte eingebettet (integriert) ist. Der Begriff leitet sich davon ab, dass Firmware funktional fest mit der Hardware verbunden ist, was bedeutet, dass das eine ohne das andere nicht nutzbar ist. Sie nimmt eine Zwischenstellung zwischen Hardware (also den physikalischen Anteilen eines Gerätes) und der Anwendungssoftware (den ggf. austauschbaren Programmen eines Gerätes) ein. Wird auch als „Betriebssoftware“ bezeichnet.

Platz für eigene Ergänzungen:

Basierend auf Wikipedia.de



Checkliste Musterlösung

Schülerlabor Informatik

Checkliste

Zum Anschließen eines Bausteins



InfoSphere

World of Informatics

Zum Anschließen eines Bricklets an den PC müssen folgende Schritte durchgeführt werden:

1. Falls verbunden: USB-Verbindung trennen
2. Bricklet und Brick mit Bricklet-Kabel verbinden
3. USB-Kabel anschließen
4. Brick-Viewer neu verbinden

Zum Verwenden eines Bausteins im Programm müssen folgende Schritte durchgeführt werden:

1. Bricklet-UID im Brick-Viewer auslesen und notieren
2. Bibliothek importieren
3. String-Variable für UID definieren und mit UID des Bricklets füllen
4. Neues Bricklet-Objekt mit Bricklet-UID-String als Übergabeparameter erstellen
5. Mit ipcon Bricklet zur Verbindung hinzufügen



Merkblatt Java

Schülerlabor Informatik

Betreuermerkblatt Java
Nützlich für die Programmierung



InfoSphere
World of Informatics

Listener

Manche Bausteine stellen so genannte Listener zur Verfügung. Sie werden benutzt durch ein Stück Quelltext, wie es beispielhaft zweimal in der Datei Einstieg.java zu finden ist. Ihr findet den entsprechenden Quelltext auch immer in der Dokumentation des Bausteins und könnt ihn von dort kopieren und anpassen.

Was sind Listener und was tun sie: Es gibt zwei verschiedene Sorten von Listnern.

Sorte 1: Immer, wenn ein bestimmtes Ereignis eintritt (z.B. wenn ein Knopf gedrückt wurde), aktiviert er besagten Quellcode-Abschnitt und gibt den entsprechenden Ereigniszustand in einer Variable aus. Diese kann innerhalb des Listener-Quellcode-Abschnittes verwendet werden.

Beispiel:

```
// Add and implement listener for pressed and released events
lcd.addListener(new BrickletLCD20x4.ButtonPressedListener() {
    public void buttonPressed(short button) {
        System.out.println("Pressed: " + button);
    }
});
```

Dies ist ein Beispiel für einen Listener eines LCD-Bricklets mit Namen „lcd“, die besagte Variable heißt hier „button“. Dieser Listener realisiert also, dass immer, wenn ein Button des LCD-Knopfes gedrückt wurde, die Nummer des Buttons in der Java-Konsole ausgegeben wird.

Sorte 2: Eine weitere Art von Listnern funktioniert ganz ähnlich: Sie überprüft nach einem festgelegten Intervall einen bestimmten Sensorwert (oder ähnliches) des Bausteins. Das Überprüfungsintervall kann selber festgelegt werden.

Auch dieser Listener gibt nach jeder Wertüberprüfung in einer Variable den gefundenen Wert an.

Beispiel:

```
// Set Period for position callback to 0.05s (50ms)
// Note: The position callback is only called every second if the
// position has changed since the last call!
poti.setPositionCallbackPeriod(50);

// Add and implement position listener (called if position changes)
poti.addListener(new BrickletLinearPoti.PositionListener() {
    public void position(int position) {
        System.out.println("Position: " + position);
    }
});
```

Dies ist ein Beispiel für einen Listener eines Potentiometer-Bricklets mit Namen „poti“, die Variable heißt „position“. Das Abfrageintervall wurde vorher auf 50 ms festgelegt. Dieser Listener realisiert, dass alle 50 ms die aktuelle Position des Schiebereglers auf der Java-Konsole ausgegeben wird.

Tipps:

- In Listnern müssen oft try-catch-Blöcke verwendet werden. Dazu auch der nächste Abschnitt.
- Variablen, die in Listnern erzeugt werden, können nicht außerhalb des Listeners verwendet werden.





try-catch:

Try-catch-Blöcke dienen dazu, diverse Fehler abzufangen. Ihr Grundprinzip ist wie folgt: Java versucht, den try-Block (grün) auszuführen. Gelingt dies ohne Fehler, wird das Programm wie gewöhnlich nach dem try-catch-Block fortgesetzt. Dann wird der catch-Block nicht beachtet. Verursacht jedoch ein Methodenaufruf im try-Block einen Fehler (z.B. die BrickletXY.Abfrage() im blauen Abschnitt), dann wird nur noch das ausgeführt, was im catch-Block (rot) steht und das Java Programm beendet.

Was genau ist ein Fehler? In Java gibt es das Konzept der so genannten Exceptions. Eine Methode kann eine solche Exception „werfen“. Es gibt Exceptions verschiedener Typen, um verschiedene Fehlertypen auseinanderhalten zu können. Der try-catch-Block dient dazu, diese geworfenen Exceptions „aufzufangen“. Er fängt immer die Art von Exceptions, die hinter dem „catch“ in der Klammerung steht. In diesem Beispiel also eine „IPConnection.TimeoutException“ (weiß). Sie erhält den Namen e.

Ihr müsst try-catch-Blöcke nicht komplett verstanden haben, um sie hier anwenden zu können. Ihr müsst nur folgendes Wissen: Manche Java-Methoden der Bausteine werfen Exceptions. Wenn dies der Fall ist, dann müsst ihr diese Methode innerhalb eines solchen try-catch-Blockes ausführen. Welche Methoden und Aufrufe das betrifft, steht jeweils in der Dokumentation der Tinkerforge-Webseite. Dort steht auch, welchen Typ von Exception die jeweilige Methode wirft. Diesen Typ müsst ihr nur noch in das weiße Feld des Beispiels kopieren. Den Rest des catch-Blockes könnt ihr so übernehmen.

Tipps:

- Im try-Block dürfen auch Anweisungen stehen, die keine Exceptions werfen.
- Im try-Block können auch mehrere Anweisungen/Methodenaufrufe stehen, die den gleiche Exception-Typ werfen. Diese Aufrufe brauchen also nicht alle einen eigenen try-catch-Block.
- Manche Methoden, die Exceptions werfen, müssen nicht in einem try-catch-Block stehen. Sie müssen es aber immer, wenn diese Methoden innerhalb eines Listeners verwendet werden.

Beispiel:

```
try {
    /*
     * Beliebige Anfragen an Bausteine innerhalb eines Listeners
     * müssen in einem solchen try-catch-Block stehen
     */
    if ( BrickletXY.Abfrage() == true ) {
        // Tue etwas
    }
}
catch (IPConnection.TimeoutException e) {
    System.err.println("Falsche UID eines Bricklets? Verbunden?");
}
```





Daten in eine Datei schreiben

Wenn ihr Daten in eine Datei, beispielsweise eine Textdatei (.txt) schreiben wollt, so müsst ihr im entsprechenden Java-Programm zuerst die Bibliothek importieren, die diese Funktionen bereitstellt:

```
import java.io.*;
```

Anschließend könnt ihr folgendes Grundgerüst verwenden. Try-catch-Blöcke habt ihr ja schon kennen gelernt.

Angepasst werden muss noch der Dateiname („eureDatei.txt“) in der grünen Zeile. Wenn ihr die Datei an einen bestimmten Ort speichern wollt, könnt ihr vor dem Dateinamen den Pfad zur Datei angeben.

Außerdem angepasst werden muss der Text bzw. die Daten, die ihr in die Datei speichern wollt (blaue Zeile). Der Zeilenumbruch wird durch die im Beispiel rot markierte Zeile realisiert.

Den Rest könnt ihr übernehmen. Wollt ihr mehr als eine Zeile schreiben, könnt ihr einfach mehrere der blauen (und roten) Zeilen nacheinander schreiben.

```
try {
    Writer fw = null;
    fw = new FileWriter( "eureDatei.txt" );
    fw.flush();
    fw.write( "euer Text " + variablenname );
    fw.append( System.getProperty("line.separator") ); // e.g. "\n"
    if ( fw != null ) {
        try {
            fw.close();
        } catch ( IOException e ) {
            e.printStackTrace();
        }
    }
} catch ( IOException e ) {
    System.err.println( "Konnte Datei nicht erstellen" );
}
```





Daten aus einer Datei lesen

Auch, um Daten aus einer Datei auszulesen, braucht man die Funktionen der Bibliothek `java.io.*`;

```
import java.io.*;
```

Das Prinzip funktioniert wie beim Schreiben in eine Datei: Ihr gebt den Dateinamen an (grüne Zeile). Anschließend verwertet ihr den eingelesenen String auf beliebige Art und Weise.

Beispielsweise könnt ihr ihn Ausgeben (blaue Zeile).

Den Rest des Grundgerüsts könnt ihr wieder so übernehmen.

```
try {
    System.out.println("/******");

    String testprogramm = getTextFileContent("eureDatei.txt");
    System.out.println("Ausgabe: " + testprogramm);

    System.out.println("/******");
} catch (IOException e) {
    System.err.println("Eine Datei konnte nicht eingelesen werden");
}
```

Warteschleife:

In manchen Situationen möchte man die Ausführung eines Programmes kurz anhalten, beispielsweise um auf etwas zu warten oder ähnliches.

Das folgende Beispiel realisiert diese Funktion:

```
try {
    Thread.sleep(1000);
} catch (InterruptedException e) {}
```

Die Zahl in der grünen Zeile gibt die Sekunden in tausendstel an. Im Beispiel wird also eine Sekunde gewartet. Dieses Gerüst könnt ihr so kopieren. Der `catch`-Block kann leer bleiben.



Beispiellösung Einstiegsprojekt

```

1 import com.tinkerforge.BrickMaster;
2 import com.tinkerforge.BrickletLCD20x4;
3 import com.tinkerforge.BrickletTemperature;
4 import com.tinkerforge.BrickletLinearPoti;
5 import com.tinkerforge.BrickletIO4;
6
7 import com.tinkerforge.IPConnection;
8
9
10 public class Einstieg_Musterloesung {
11     //Define connection parameters
12     private static final String host = "localhost";
13     private static final int port = 4223;
14
15     //Define UIDs of the devices
16     private static final String masterUID = "9JnhKoQ8ieS"; // Change to your UID
17     private static final String lcdUID = "94h";
18     private static final String tempUID = "9kq";
19     private static final String potiUID = "6Td";
20     private static final String io4UID = "8Q7";
21
22
23
24
25
26     // Note: To make the example code cleaner we do not handle exceptions. Exceptions you
27     // might normally want to catch are described in the comments below
28     public static void main(String args[]) throws Exception {
29
30         // Create connection to brickd
31         IPConnection ipcon = new IPConnection(host, port); // Can throw IOException
32
33
34         // Create device objects
35         final BrickMaster master = new BrickMaster(masterUID); // Create device object
36         final BrickletLCD20x4 lcd = new BrickletLCD20x4(lcdUID);
37         final BrickletTemperature temp = new BrickletTemperature(tempUID);
38         final BrickletLinearPoti poti = new BrickletLinearPoti(potiUID);
39         final BrickletIO4 io4 = new BrickletIO4(io4UID);
40
41         // Add device to IP connection
42         ipcon.addDevice(master); // Can throw IPConnection.TimeoutException
43         ipcon.addDevice(lcd);
44         ipcon.addDevice(temp);
45         ipcon.addDevice(poti);
46         ipcon.addDevice(io4);
47         // Don't use device before it is added to a connection
48
49
50
51         /*
52          * Ab hier laeuft das eigentliche Programm
53          */
54
55
56
57         /*
58          * Gruppe A: Bei Knopfdruck aktuelle Temperatur auf LCD ausgeben und Beleuchtung ueber
59          * Button steuern:
60          */
61
62         //LCD Licht an, Display leeren und Menue ausgeben
63         lcd.backlightOn();
64         lcd.clearDisplay();
65         lcd.writeLine((short)0, (short)0, "Licht schalten:");
66         lcd.writeLine((short)1, (short)10, "Button 1");
67         lcd.writeLine((short)2, (short)0, "Temperatur messen:");
68         lcd.writeLine((short)3, (short)10, "Button 2");
69
70         // Add and implement listener for pressed and released events
71         lcd.addListener(new BrickletLCD20x4.ButtonPressedListener() {
72             public void buttonPressed(short button) {
73                 //System.out.println("Pressed: " + button); //Debugging

```



```

74
75 // Button 0 (erster Button) wird gedruickt -> Licht an/aus schalten
76 try {
77 // Falls das Licht an ist:
78 if ( button == 0 && lcd.isBacklightOn() == true ) {
79     lcd.backlightOff();
80 // Falls das Licht aus ist:
81 } else if ( button == 0 && lcd.isBacklightOn() == false ) {
82     lcd.backlightOn();
83 }
84 } catch (IPConnection.TimeoutException e) {
85     System.err.println("BrickletLCD20x4 IPConnection.TimeoutException.");
86     System.err.println("Falsche UID des BrickletLCD20x4? Verbunden?");
87 }
88
89 // Button 1 (zweiter Button) wird gedruickt -> Aktuelle Temperatur auf LCD
ausgeben
90 if ( button == 1 ) {
91     lcd.clearDisplay();
92     try {
93         short temperature = temp.getTemperature(); // Can throw
94         IPConnection.TimeoutException
95         lcd.writeLine((short)0, (short)0, "Temperatur aktuell:");
96         lcd.writeLine((short)1, (short)0, (int)(temperature/100.0) + " °C");
97     } catch (IPConnection.TimeoutException e) {
98         System.err.println("TemperatureBricklet IPConnection.TimeoutException.");
99         System.err.println("Falsche UID des TemperatureBricklets? Verbunden?");
100     }
101 //Zurueck zum Menu:
102 lcd.writeLine((short)2, (short)0, "Zurueck zum Menu:");
103 lcd.writeLine((short)3, (short)10, "Button 3");
104 }
105 // Button 2 (dritter Button) wird gedruickt -> Menue ausgeben
106 if ( button == 2 ) {
107     lcd.clearDisplay();
108     lcd.writeLine((short)0, (short)0, "Licht schalten:");
109     lcd.writeLine((short)1, (short)10, "Button 1");
110     lcd.writeLine((short)2, (short)0, "Temperatur messen:");
111     lcd.writeLine((short)3, (short)10, "Button 2");
112 }
113 }
114 });
115
116
117
118 /*
119 * Gruppe B: LEDs durch Potentionmeter steuern:
120 */
121
122 //Potentiometer konfigurieren
123 // Set Period for position callback to 0.05s (50ms)
124 // Note: The position callback is only called every second if the
125 // position has changed since the last call!
126 poti.setPositionCallbackPeriod(50);
127 io4.setConfiguration((short)15, 'o', false); //alle aus
128
129 // Add and implement position listener (called if position changes)
130 poti.addListener(new BrickletLinearPoti.PositionListener() {
131     public void position(int position) {
132
133         // Poti ist am oberen oder unteren Ende -> Licht aus
134         if ( position == 0 || position == 100 ) {
135             io4.setConfiguration((short)15, 'o', false); //Rot = 0, Rot aus
136         }
137
138         // Poti ist in der Mitte -> Gruen an, Rot an
139         if ( position <= 55 && position >= 45 ) {
140             io4.setConfiguration((short)3, 'o', true); // Rot und Gruen an
141         }
142
143         // Poti ist in der unteren Halfte -> Gruen an, Rot aus
144         if ( position < 45 && position > 0 ) {
145             io4.setConfiguration((short)1, 'o', true); //Gruen = 1, Gruen an

```



```
146         io4.setConfiguration((short)2, 'o', false); //Rot = 0, Rot aus
147     }
148
149     // Poti ist in der oberen Hälfte -> Rot an, Gruen aus
150     if ( position < 100 && position > 55 ) {
151         io4.setConfiguration((short)1, 'o', false); //Gruen = 1, Gruen aus
152         io4.setConfiguration((short)2, 'o', true); //Rot = 0, Rot an
153     }
154 }
155 });
156
157 /*
158  * Bis hier laeuft das eigentliche Programm
159  */
160
161 System.console().readLine("Press [Enter] and then [Strg+c] to exit\n");
162 io4.setConfiguration((short)15, 'o', false);
163 lcd.clearDisplay();
164 lcd.backlightOff();
165 ipcon.destroy();
166 System.exit(0);
167 }
168 }
169 }
```



Hardware Packliste

Schülerlabor Informatik

Einstiegsprojekt
PacklisteInfoSphere
World of Informatics

Diese Hardware muss den SuS im Einstiegsprojekt zur Verfügung stehen:

2 Laptops



1 Master Brick



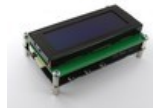
1 USB Kabel



4 Bricklet-Kabel



1 LCD-Bricklet



1 Linear-Poti Bricklet



1 Schraubendreher



1 Temperature-Bricklet



1 IO4-Bricklet



1 LED Rot (mit Widerstand)



1 LED Grün (mit Widerstand)



Anhang E Digitalisat

Auf dem Datenträger befinden sich neben den digitalen Versionen aller Materialien auch die Dateien, die nicht in gedruckter Form beiliegen, beispielsweise das Wiki und die Beispiellösungen für die Projektphase.



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.

Aachen, 19. Februar 2013

Fabian Blasius

