Modul: Arduino-Mikrocontroller

Ein Smart-Meter mit Hilfe von Arduino-Microcontrollern simulieren

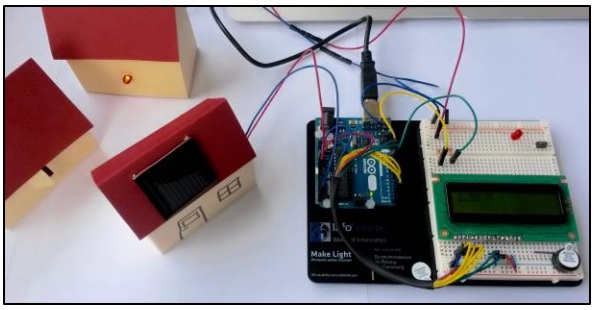
Fragestellung

Welche Rolle spielt die Informatik beim Messen und Steuern im Smart Home? Wie kann die Einbindung regenerativer Energien gesteuert werden?

Material

- Ca. 10 Computer
- Beamer
- ca. 10 ArduinoUno + Bauteile (siehe Liste im Anhang)
- Arbeitsbögen

Durchführung



Nach einer Einführung in die Bestandteile des Arduino-Microcontrollers und dem Erschließen eines Beispiels setzen die Schülerinnen und Schüler ein selbst gewähltes Feature mithilfe des Arduino-Microcontrollers um. Das Szenario stellt dabei einen Haushalt dar, der eine Photovoltaik-Anlage und diverse Verbraucher (in Form von LEDs) besitzt. Weiterhin ist das Haus über Smart Grid mit Verbrauchern in seiner Nähe verbunden. Ein typisches Feature das umgesetzt werden kann, ist das Versorgen von anderen Haushalten, wenn die eigene Anlage genug Energie erzeugt, um sie in das Netz einzuspeisen. Im Anschluss an die Erarbeitung stellen alle Gruppen ihre Ideen vor.

Auswertung

Da das gewählte Szenario einen hohen Abstraktionsgrad besitzt, ist es Aufgabe der Lehrkraft, gemeinsam mit den Schülerinnen und Schülern den Bezug zum Themengebiet Nachhaltigkeit herzustellen. Im Anschluss wird gesammelt, welche Daten bei der Nutzung des Smart-Meters entstehen und wie diese genutzt werden können (Anknüpfungsmöglichkeit: Modul „Stationenlernen“).

Wozu das Ganze?

Das Smart Meter ist ein wichtiges Bestandteil des zukünftigen Smart Grid. Während es als zentrales Steuerungselement im Haushalt erlaubt, effiziente Steuerungsprozesse zu regeln und eine Kommunikation zwischen Netz und Privathaushalt herzustellen, erlaubt es auch, Verbraucherdaten zu sammeln. Im Zusammenhang mit Datenschutz und Datensicherheit stellt das Smart-Meter einen kritischen Bestandteil des Smart Grid dar. Als zukünftige Verbraucher sollen die Schülerinnen und Schüler diese Technologie nicht als Blackbox auffassen, sondern dazu in der Lage sein, Vor- und Nachteile der Technologie zu beurteilen.

Dauer

Ca. 180 Minuten

Anhang

Artikel „Entwicklung und Dokumentation einer Unterrichtseinheit zum Thema Smart Grid“ von Christoph van Heteren-Frese, inklusive Arbeitsblättern, Quellcode, Verlaufsplanung und Folienätzen sowie Erläuterungen für die Lehrkraft.

Freie Universität Berlin

Entwicklung und Dokumentation einer Unterrichtseinheit zum Thema Smart Grid

Seminararbeit Modul *S/P: Vertiefung Fachdidaktik Informatik*
(LV-Nr.: 19314511/19314630)

Angefertigt an der
Freien Universität Berlin
Fachbereich: Mathematik und Informatik
Studiengang: Lehramtsmaster 120 LP

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

bspw.	beispielsweise
d.h.	das heisst
ebd.	ebenda
ggf.	gegebenenfalls
i.d.R.	in der Regel
Sek. I	Sekundarstufe I
SG	Smart Grid
SuS	Schülerinnen und Schüler
u.	und (Zitation)
u.a.	unter anderem
u. a.	und andere (Zitation)
vgl.	vergleiche
z.B.	zum Beispiel

1. Vorbemerkungen und Einleitung

Die vorliegende Arbeit ist im Rahmen des Didaktikseminars „Vertiefung Fachdidaktik Informatik (S+P)“ im Wintersemester 2014/2015 an der Freien Universität Berlin entstanden. Während dieser Veranstaltung ist von den Lehramtsstudierenden der Informatik und Physik eine fachübergreifende Unterrichtsreihe zum Thema Smart Grid konzeptioniert und in zwei Durchläufen unabhängig voneinander in den Schülerlaboren *PhysLab* und *MILab* erprobt worden.

Die Reihe besteht aus drei Teilen: Einer allgemeinen Einführung in das Smart Grid, einer vertiefenden Betrachtung der soziotechnischen Aspekte (Datenschutz, Technikfolgenbewertung und Technikfolgenabschätzung) und einem praktischen Teil, in dem mit Hilfe eines Arduino Mikrocontrollers ein Smart Meter simuliert und programmiert wird. Diese Arbeit befasst sich ausschließlich mit dem zuletzt genannten praktischen Teil dieser Unterrichtsreihe und stellt u.a die Dokumentation der entwickelten Schaltung und Bibliotheksfunktionen dar.

1.1. Inhaltliche Relevanz

Das Smart Grid (SG) wird schon seit einiger Zeit als ein vielversprechendes Konzept zur Lösung von Energie- und Umweltproblemen gehandelt[1]. Im Zentrum steht die Verknüpfung der unterschiedlichen Akteure des Energiemarktes (Erzeuger, Verbraucher etc.). Obwohl Struktur und Funktion bereits klar definiert sind, ist eine konkrete Realisierung in größerem Maßstab bis jetzt noch nicht in Sicht. Mitverantwortlich dafür sind u.a. die komplexen und daher nur unvollständig abzusehenden Folgen im Bereich des Datenschutzes. Das liegt nicht zuletzt daran, das verschiedene Interessengruppen an der Gestaltung beteiligt sind.

Im Mittelpunkt des hier vorgestellten praktischen Teils steht das Smart Meter. Vereinfacht ausgedrückt handelt es sich hierbei um einen intelligenten Stromzähler, der in den Haushalten der Verbraucher verbaut die Steuerzentrale diverser Smart Grid-fähiger Endgeräten (Waschmaschinen, Gefrierschränke) darstellt. Von hier aus werden alle wesentlichen Einstellungen getätigt, um die Geräte tageszeitabhängig zu steuern. Hierfür werde durch die Energielieferanten spezielle variable Stromtarife angeboten.

InformatikerInnen besitzen die Fachkompetenz, die zur Planung und Umsetzung eines solchen Systems benötigte wird. Die SuS erhalten im Zuge der Auseinandersetzung mit der Programmierung des Smart Meter einen Einblick in die gesellschaftliche

[1] Für grundlegende Informationen und den aktuellen Entwicklungsstand sei auf die Website *Smart Grids European Technology Platform — SmartGrids* (2016) verwiesen.

Relevanz und Verantwortung der Informatik. Das Smart Meter stellt somit als Teil des Smart Grids einen aktuellen und relevanten Unterrichtsgegenstand dar, anhand dessen sich zeigen lässt, das die Informatik als wissenschaftliche Disziplin eine wichtige Teilkomponente gesellschaftlicher Entwicklung und Innovation ist.

1.2. Standard- und Kompetenzbezug

Der hier vorgestellte Praxisteil ermöglicht den SuS besonders das selbstständige Planen und Lösen einer Programmieraufgabe und folgt somit dem Ansatz des problemorientierten Lernens. In Tabelle 1 sind die Kompetenzen zusammengestellt, die durch die Unterrichtseinheit in der Sekundarstufe I (Sek. I) besonders gefördert werden.

Tabelle 1: Übersicht über die im Fokus stehenden Kompetenzen.

Kompetenz-bereich	**Kompetenzbezug** Die SuS der Sek. I...	**Konkretisierter Standard** Die SuS...
Fachwissen	„...erstellen kleine Programme unter Benutzung vorgegebener Bausteine [...]." (vgl. Senatsverwaltung für Bildung 2006, S. 10)	...implementieren unter Verwendung der vorgegebenen Bibliotheksfunktionen kleine Funktionen eines Smart Meters.
Kommunikation	„...erkennen, analysieren und betreiben aktiv Kommunikationsvorgänge in verschiedenen Ebenen und mit unterschiedlichen Beteiligten: Präsentation und Diskussion von Arbeitsgruppenergebnissen." (vgl. Senatsverwaltung für Bildung 2006, S. 12)	...präsentieren ihre in Partnerarbeit geplanten und entwickelten Funktionen des simulierten Smart Meters.

Die Unterrichtseinheit lässt sich z.B. im Wahlpflichtfach Informatik der Sek. I im Wahlmodul WP4 „Automatische und technische Systeme" verorten. In der Inhaltsbeschreibung heißt es: „Das Modul beschäftigt sich mit Informatiksystemen, die nicht in Schreibtisch-/Bürorechnern, sondern in Geräten des alltäglichen Gebrauchs, des Verkehrs oder der industriellen Technik [...] implementiert sind." (vgl. Senatsverwaltung für Bildung 2006, S. 29). Das Smart Meter wird wie anfangs dargestellt nach der Einführung die zentrale Kommunikations- und Kommandozentrale sein, mit der verschiedene Geräte des Haushalts kommunizieren und gesteuert werden. Somit wird es tatsächlich zum Gegenstand des alltäglichen Gebrauchs. Gleichzeitig stellt es eine Schnittstelle zu industriellen Standards der Hausautomation dar.

1.3. Struktur der Unterrichtseinheit

Im folgenden wird die Struktur der Unterrichtseinheit überblicksartig beschrieben. Ein detaillierter Unterrichtsverlaufsplan ist dieser Arbeit angehängt (Anhang A).

Die Unterrichtseinheit beginnt mit der Einführung des Begriffs, der Merkmale und der Funktionsweise eines Mikrocontrollers (μC) in Form eines Foliengestützten Lehrervortrags (LV). Der verwendete Foliensatz ist in Anhang B abgebildet. Diese Phase dient der ersten Annäherung an den abstrakten Begriff (Abbildung 3 bis 6 in Anhang B). Es folgt die Präsentation eines konkreten d.h. physischen Beispiels, nämlich des *ArduinoUno*, sowie der zum Einsatz kommenden Komponenten und der Entwicklungsumgebung. Je nach zur Verfügung stehender Zeit bietet es sich an anschließend exemplarisch ein einfaches Feature live zu implementieren (cognitive apprenticeship).

Nach dieser lehrerzentrierten Demonstrationsphase folgt eine kurze Sicherungsphase, in der die SuS zunächst aufgefordert werden die ggf. noch unklar gebliebene Sachverhalte im Plenum zu besprechen. Anschließend wird das bisher Besprochene mit Hilfe eines kurzen Quiz verschriftlicht und ausgewertet.

In der zweiten Phase liegt der Fokus auf Partner- bzw. Kleingruppenarbeit. Zunächst wird durch die Lehrkraft das Szenario vorgestellt (Folie 7 bis 8): Die SuS sollen nun den Arduino stellvertretend für ein Smart Meter eigenverantwortlich programmieren. Dazu soll mit Hilfe des Arbeitsmaterials (siehe Anhang C) zunächst entweder eine eigene oder aber eine dort vorgeschlagene Funktionalität entworfen bzw. ausgewählt werden. Anschließend soll gemeinsam die Planung und Implementierung der gewählten Funktionalität in Angriff genommen werden.

Die Sicherung der Gruppenarbeit erfolgt durch die Präsentationen der Ergebnisse durch die SuS. Während dieser abschließenden Phase sollen die SuS sowohl ihren bisherigen Ergebnisse sowie ggf. aufgetreten Probleme vorstellen und diskutieren.

2. Benötigte Hard- und Software

Damit die Unterrichtseinheit wie oben beschrieben durchgeführt werden kann, werden neben den hier zur Verfügung gestellten Materialien einige weitere Komponenten benötigt. Nachfolgend werden diese kurz beschrieben. Dabei wird vor allem auf den *ArduinoUno* und die dazugehörige Entwicklungsumgebung eingegangen.

2.1. Arduino-Mikrocontroller

Der hier verwendete *ArduinoUno* ist ein Mikrocontroller, der aufgrund seiner Flexibilität und einfachen Handhabung schon seit einiger Zeit in diversen Gebieten Verbreitung findet[2]. Er ist äußerst robust und arbeitet nur mit ungefährlichen Spannungen. Zugleich bietet er durch seine offene Bauweise einen direkten Zugang zu einem Mikrocontroller, an dem alle wesentlichen Bauelemente zugänglich und erfahrbar sind. Zudem ist er Quelloffen im Sinne einer LGPL/GPL-Lizenz[3]. Dementsprechend lässt er sich auch für den Informatikunterricht der Sekundarstufe gewinnbringend einsetzten. Das Schülerlabor *InfoSphere* der RWTH Aachen hat beispielsweise Ende 2013 in Kooperation mit dem Bundesministerium für Bildung und Forschung (BMBF) eine ganze Unterrichtsreihe zum Thema „Informatik enlightened“ produziert (vgl. *Informatik enlightened - Was Blumen, Autos und Solarzellen verbindet — Schülerlabor Informatik - InfoSphere* 2016).

Auf dem kleinen Board kommt ein ATmega328-Mikrocontroller zum Einsatz, der entweder über USB (5 V) oder eine externe Spannungsquelle (7–12 V) versorgt wird. Die wesentliche Schnittstelle zur Außenwelt bildet eine Reihe von digitalen Ein- und Ausgängen, die zur Anbindung elektronischer Schaltungen dient. Zusätzlich verfügt das Board über einige Eingänge, die analoge Signale mit Hilfe eines 10-Bit A/D-Wandlers verarbeiten können. Ebenso können einige Pins ein analoges Ausgangssignal simulieren[4].

Die Ein- bzw. Ausgänge können zum Anschluss kompletter Erweiterungsplatinen, sog. *Shields* dienen, die eine erweiterte Funktionalität (z.B. WLAN, Bluetooth, RFID) zur Verfügung stellen und direkt auf den Controller aufgesteckt werden. Ebenso können eigene Schaltungen, die mit Hilfe von einzelnen elektronischen Bauelementen (Wiederstand, Spule, Kondensator, Transistor) auf einfachen Steckplatinen realisiert werden können, angesteuert und programmiert werden. Hier liegt die große Stärke des Arduino, da praktisch alle elektronischen Bauteile Verwendung finden können. Die Übertragung des entsprechenden Programms auf den Controllers geschieht über den eingebauten USB-Anschluss.

[2] Der Mikrocontroller wird z.B. auch „fachfremd“ für die Erstellung von Kunstinstallationen verwendet.

[3] Die GNU General Public License ist eine weit verbreitete Software-Lizenz. Sie gestattet das Ausführen, Studieren, Ändern und Verbreiten (d.h. Kopieren) der betreffenden Software.

[4] Dies geschieht durch eine 8-Bit breite Pulsweitenmodulation (PWM).

2.2. Sonstige Bauteile

Neben dem *ArduinoUno* sind noch einige andere Bauteile für den Aufbau der Smart Meter-Simulation notwendig. Um die unten beschrieben Schaltung aufzubauen, wird zunächst eine ausreichend große Steckplatine benötigt. Zusätzlich müssen die einzelnen Bauteile (LEDs, Widerstände, Piezo-Signalgeber, Steckverbinder usw.) beschafft werden. Die meisten dieser Elemente sind in Elektronikfachmärkte für einige Cents bzw. wenige Euro verfügbar. Da der *ArduinoUno* ohne USB-Kabel ausgeliefert wird, muss auch dieses extra besorgt werden. Tabelle 2 stellt eine vollständige Liste der benötigten Teile dar.

Tabelle 2: Liste aller benötigen Bauelemente. Die Positionen 1 bis 10 sind Bestandteile des „*InfoSphere*-Kit“. Postion 11 und 12 werden nur für das „Erzeuger-Szenario “ benötigt (siehe Abschnitt 3.3).

Pos.	Bezeichnung	Menge	Bezugsquelle
1.	Arduino UNO Platine	1	z.B. Conrad
2.	USV-Verbindungskabel	1	z.B. Conrad
3.	Steckplatine	1-2	z.B. Conrad
4.	Steckbrückenset und Kabel	1	z.B. Conrad
5.	LED	2-4	z.B. Conrad
6.	RGB-LED	1	z.B. Conrad
7.	Widerstand (220 Ohm)	4-6	z.B. Conrad
8.	Widerstand (100 kOhm)	1	z.B. Conrad
9.	Fotowiderstand	1	z.B. Conrad
10.	Piezo-Signalgeber	1	z.B. Conrad
11.	Punkt-Matrix-Anzeige	1	z.B. Conrad
12.	Solarzelle	1-2	z.B. Conrad

Wie oben bereits beschrieben hat das Schülerlabor *InfoSphere* der RWTH-Aachen eine vollständige Unterrichtsreihe mit dem Arduino produziert. In diesem Zusammenhang haben die Autoren mit Hilfe der Firma Watterott ein kostengünstiges Bauteilset auf den Markt gebracht, das neben den hier benötigten Teilen viele weitere Bauelemente enthält. Es kann online unter `http://www.watterott.com/de/Infosphere-Kit` bestellt werden.

2.3. Arduino-Entwicklungsumgebung

Die Arduino-Entwicklungsumgehung stellt die Programmierschnittstelle dar, mit der der Mikrocontroller direkt programmiert werden kann. Sie ist ebenfalls im Sinne einer LGPL/GPL-Lizenz quelloffen. Für die Programmierung des Controllers kommt eine vereinfachte Version der objektorientierten Programmiersprache *Processing* zum Einsatz, die selbst auf C/C++ basiert. Durch die Vorgabe fester Strukturen und die Kapselung vieler Details (z.B. Header-Dateien) ist diese besonders für Programmiereinsteiger geeignet.

Im Prinzip lässt sich das Programm durch das betätigen von nur zwei Schaltflächen bedienen: Während der linke runde Button unter der Menüzeile den geschriebenen Quelltext lediglich kompiliert (und somit auf syntaktische Korrektheit überprüft), wird mit dem rechten Button das Programm anschließend zusätzlich auf den Mikrocontroller übertragen (Siehe Folie 6 im Anhang B).

3. Materialien und Schaltungsaufbau

Zur Vorbereitung der Unterrichtseinheit sollten sämtliche Schaltungen aufgebaut und getestet werden. Diese Aufgabe kann alternativ auch im Vorfeld von den SuS übernommen werden. Dazu sollte am Besten eine zusätzliche Einzelstunde eingeplant und zudem eine detaillierte Anleitung zur Verfügung gestellt werden. Es ist denkbar, diese Arbeit auch fachübergreifend im Physikunterricht zu erledigen.

Die folgenden Abschnitte beschreiben zunächst den Schaltungsaufbau des Smart Meter-Szenarios und der dazugehörigen Smart-Grid Bibliothek. Anschließend wird ein weiteres Szenario vorgestellt, das im Nachgang des o.g. Seminars entwickelt wurde.

3.1. Smart Meter-Szenario

Das simulierte Smart Meter besteht im Prinzip aus vier Modulen: Es verfügt erstens über eine sog. Status-LED (Pin 13) in der rechten oberen Ecke sowie zweitens über eine RGB-LED (Pin 3,5 und 6) oben in der Mitte. Am unteren Rand ist auf der linken Seite drittens ein Fotowiederstand (Pin A5) zur Simulation einer Solarzelle zu finden. Unten rechts ist viertens ein Piezzo-Signalgeber (Pin 8) verbaut. Abbildung 1 zeigt detailliert den Aufbau der eben beschriebenen Schaltung.

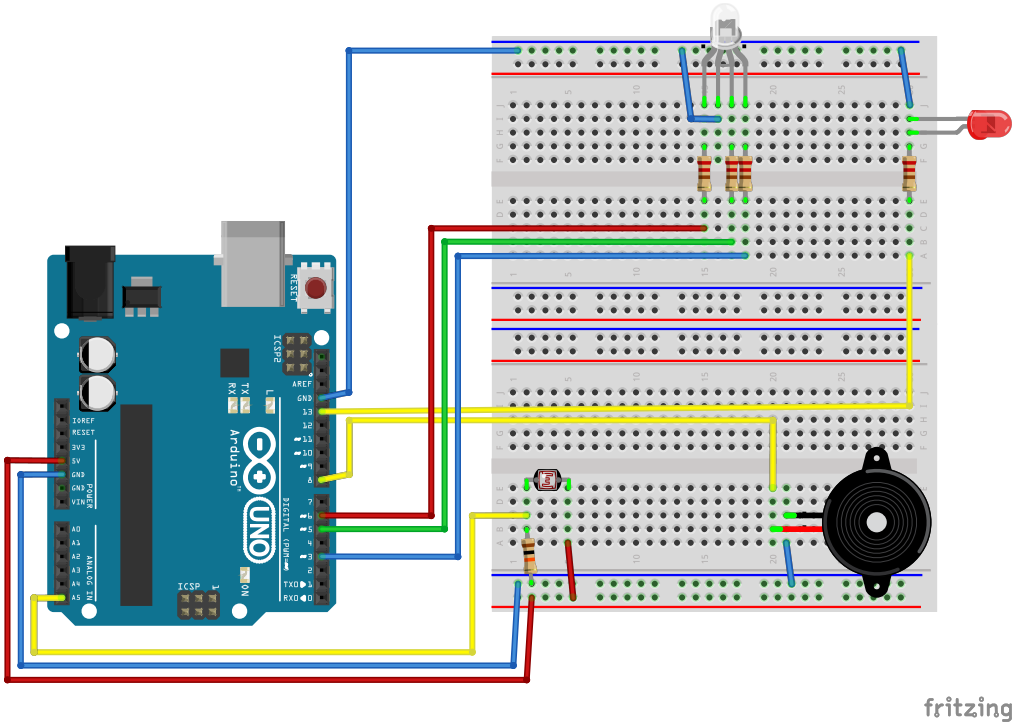


Abb. 1: Schaltungsplan der Smart Meter-Simulation. (Grafik erstellt mit *fritzing*: `http://fritzing.org/`)

Da die in Anschlüsse der eben beschriebenen Module in die Bibliotheksfunktionen fest einprogrammierter sind, ist beim Aufbau besonders auf den korrekten Anschluss an die entsprechenden Ein- bzw. Ausgängen des *ArduinoUno* zu achten.

3.2. Die Smart Grid-Bibliothek

Die Bibliothek implementiert ein Smart Meter, auf dem verschiedene Funktionen (genauer: Methoden) aufgerufen werden können. Bei der Entwicklung wurde vor allem Wert auf intuitive Funktionsbezeichner und die Kapselung und somit Verschleierung von englischsprachigen Funktionsbezeichnungen sowie einiger technischer Details gelegt. Des Weiteren verbergen einige Funktionen die Tatsache, dass die Anweisungen in der `loop()`-Funktion wiederholt ausgeführt werden: Sowohl `statusLEDBlink(float p)` als auch `pieptonAn(float p)` erzeugen anhand ihrer Parameters ein periodisches Signal. Die Logik des An- bzw. Ausschaltens der entsprechenden Komponente wird so vor den SuS verborgen. Die Funktion `RGB(float r, float, g, float, b)` steuert die RGB-LED an. Die Parameter entsprechen den Anteilen der Farbwerte rot, grün und

blau. Eine weitere wichtige Funktion ist `gibSpannung()`, die den Helligkeitswert des Fotowiederstandes ausliest und in eine Spannung umgerechnet zurückgibt. In Kombination mit der Funktion `schreibeAufKonsole()` lässt sich so der aktuelle Spannungswert auf der seriellen Konsole ausgeben. Die Signaturen der Bibliotheksfunktionen sind zusammen mit einer kurzen Beschreibung auch auf dem Arbeitsblatt (siehe Angang C) dokumentiert.

Um die Smart Grid-Bibliothek zu nutzen, muss sie über das `Menü Sketch -> Include Library` eingebunden werden. Für genauere Informationen zu Bibliotheken und deren Erstellung sei auf das *LibraryTutorial*[5] verwiesen. Sobald die Smart Grid-Bibliothek geladen wurde, stehen unter dem Menüpunkt `Datei -> Beispiele -> SmartMeter` u.a. die Beispieldatei `Beispiel_1` zur Verfügung, die als Ausgangspunkt für die Programmiertätigkeit der SuS genutzt werden kann. Sie ist in Anhang D dargestellt. Das Beispiel implementiert bereits einige Vorschläge, die auf dem Arbeitsbogen (siehe Angang C) vorgeschlagen werden.

3.3. Erzeuger-Szenario

Im Rahmen der kontinuierlichen Überarbeitung der Unterrichtsreihe ist im Nachgang des o.g. Seminars ein weiteres Szenario entstanden. Durch Hinzunahme einer Solarzelle und eines LCD-Moduls wurde ein weiteres Modell konstruiert, dass bspw. zur Simulation eines weiteren Aspekts des SG verwendet werden kann. Der Fokus liegt hier auf der Erzeugung und Einspeisung von Solarenergie durch private Photovoltaikanlagen. Mit der so erzeugten Energie kann das eigene Haus, bei ausreichend Überschuss ggf. auch das der Nachbarn gespeist werden.

Der Fotowiderstand wurde durch eine Solarzelle ersetzt, die nun die erzeugte Spannung direkt in den Arduino einspeist. Zwei (oder mehr) LEDs symbolisieren die aktive Versorgung eines Haushaltes bei ausreichend produzierter Energie. Ein zusätzliches LCD-Modul ermöglicht das direkte Ablesen der erzeugten Spannung ohne Verwendung der seriellen Konsole. Somit kann das Modell (sofern ein passendes Netzteil vorhanden ist) autark, d.h. ohne Computer betrieben werden. Zur anschaulicheren Darstellung kann die Solarzelle auf dem Dach eines kleinen Modellhauses befestigt werden. Dementsprechend lassen sich die LEDs innerhalb solcher Häuser anbringen. Die Schaltung ist in Abbildung 2 dargestellt.

[5] `https://www.arduino.cc/en/Hacking/LibraryTutorial`

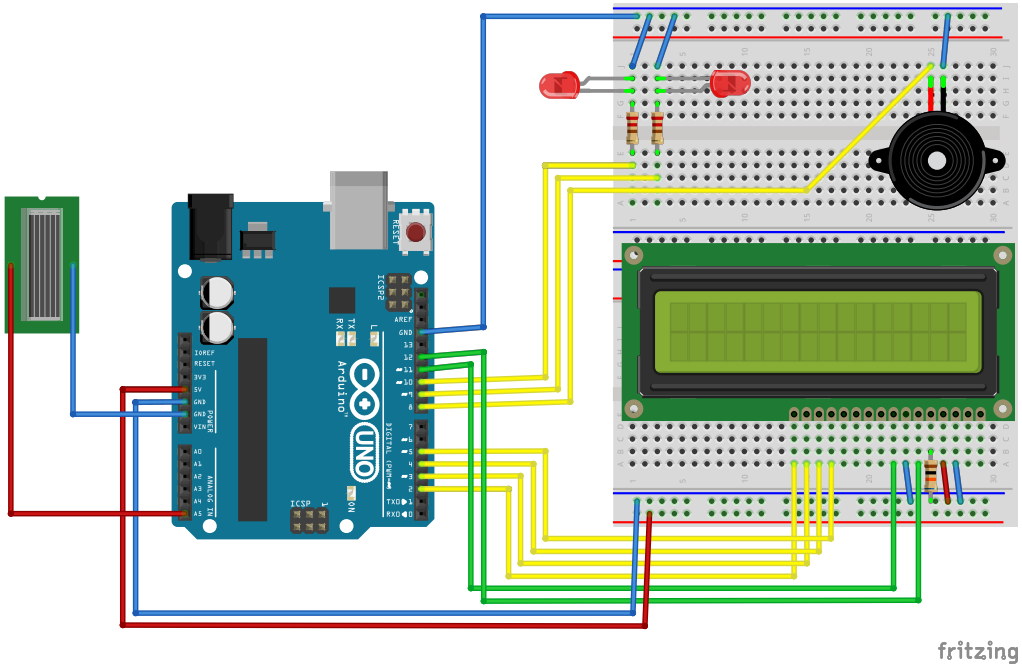


Abb. 2: Schaltungsplan des Erzeuger-Szenarios. Die hier dargestellte Pinbelegung der LCD-Anzeige (Reihenfolge von links nach rechts: $14, 13, 12, \ldots 2, 1, 15, 16$) ist für das Modul *ANAG Vision AV1624* typisch. Nähere Informationen zur technischen Spezifikation siehe Anhang. (Grafik erstellt mit *fritzing*: `http://fritzing.org/`)

4. Anmerkungen zur Durchführung

Die bisherigen Durchführungen der Unterrichtseinheit haben gezeigt, dass zu Beginn der zweiten Phase besonders auf das Modell des Smart Meters und das dadurch simulierte Szenario eingegangen werden sollte. Dies wurde während der ersten Durchführung aus zeitlichen Gründen vernachlässigt, so dass die Abhängigkeit der Funktionen von der verfügbaren Energie nicht deutlich wurde. Infolgedessen haben die programmierten Features kaum die (durch den Fotowiederstand simulierte) verfügbare Sonnenenergie einbezogen, sondern lediglich die Status-LED bzw. die RGB-LED gesteuert. Des weiteren sollte die zweite Phase zeitlich nicht zu knapp bemessen sein. Zwar schafften es die meisten Gruppe gemäß Arbeitsauftrag wenigstens ein Feature zu implementieren, jedoch handelte es sich dabei meistens lediglich um das Einschalten der Status-LED.

Diese Unterrichtseinheit wird zusammen mit den anderen Bestandteilen der Reihe ständig weiterentwickelt. Die aktuelle Version ist zusammen mit den dazugehörigen Materialien unter `https://github.com/cvhf/Unterrichtsreihe_Smart_Grid` vefügbar.

Literaturverzeichnis

Informatik enlightened - Was Blumen, Autos und Solarzellen verbindet — Schülerlabor Informatik - InfoSphere (2016). URL: http://schuelerlabor.informatik.rwth-aachen.de/modul/informatik-enlightened-was-blumen-autos-und-solarzellen-verbindet (besucht am 19. 01. 2016) (siehe S. 4).

Senatsverwaltung für Bildung, Jugend und Sport Berlin (2006). „Rahmenlehrplan für die Sekundarstufe I: ITG Informatik Wahlpflichtfach“. In: Hrsg. von Jugend und Sport Berlin Senatsverwaltung für Bildung. URL: http://www.berlin.de/imperia/md/content/sen-bildung/schulorganisation/lehrplaene/sek1%3Csub%3Ei%3C/sub%3Etg%3Csub%3Ei%3C/sub%3Enformatik.pdf (siehe S. 2).

Smart Grids European Technology Platform — SmartGrids (2016). URL: http://www.smartgrids.eu/ (besucht am 10. 01. 2016) (siehe S. 1).

A. Unterrichtsverlaufsplan

Tabelle 3: Unterrichtsverlaufplan der Unterrichtseinheit zum Smartgrid. Abk.: L: LehrerIn, AB: Arbeitsblatt, MC: Mikrocontroller

Zeit/Dauer/ Phase	**Inhalt und Lehrimpulse**	**Erwartetes Verhalten der SuS**	**Methode/Materialien/Sozialform**
5 Min. Einstieg	Vorstellung des Arduino – *Microcontroler (MC)? Was ist das?* – Bezug zum SG: *Wo gibt es im SG MC?* – Grundsätzliche Darstellung der Features und – Kurze Klärung der sichtbaren Komponenten	SuS äußern – Vorstellungen über MC und – Rolle von MC im SG – Verständnisfragen und – Fragen zur Funktionalität	Lehrervortrag, SmartBoard, *ArduinoUno*
10 Min. Erarbeitung	Präsentation der Zielvorstellung. L. zeigt bereits aufgebauten Arduino und erläutert – die aufgebaute Schaltung – die verbauten Komponenten – Grundsätzliche Darstellung der Features – die umzusetzenden Funktionen	SuS folgen der Präsentation und stellen Fragen	Lehrervortrag, SmartBoard, *ArduinoUno*, AB
10 Min. Erarbeitung	**Livecoding**: L. implementiert live ein Feature	SuS folgen der Präsentation	SmartBoard, *ArduinoUno*, Plenum (Cognitive Apprenticeship)

(Fortsetzung auf der nächsten Seite)

Tabelle 3: (Fortzsetzung)

Zeit/Dauer/ Phase	Inhalt und Lehrimpulse	Erwartetes Verhalten der SuS	Methode/Materialien/Sozialform
5 Min. Sicherung	**Quiz** (digital): Wiederholung der für die selbstständige Arbeit wesentlichen Punkte	SuS beantworten Fragen zum Arduino und den Komponenten sowie zu den bisherigen Inhalten	PC, Plenum
5 Min. Einstieg SW-Planung	L. teilt Gruppen ein und erklärt SuS den Arbeitsauftrag: – In Gruppen **je ein Feature aussuchen** und gemeinsam die Umsetzung planen – Anschließend in Partnerarbeit geplantes **Feature implementieren**	SuS hören zu und sammeln sich in ihren Gruppen	
30 Min. Erarbeitung SW-Planung	SuS setzen Arbeitsauftrag um. Studenten leisten Hilfe.	SuS diskutieren Funktionen und Möglichkeiten der Umsetzung	AB, *ArduinoUno*; Gruppenarbeit
25 Min. Auswerung/ Sicherung	Vorstellung der implementierten Features: – Phase 1: In den Gruppen – Phase 2: Im Plenum	SuS stellen ihre Ergebnisse (und Schwierigkeiten vor)	*ArduinoUno*; Gruppenarbeit, Plenum

B. Präsentationsfolien

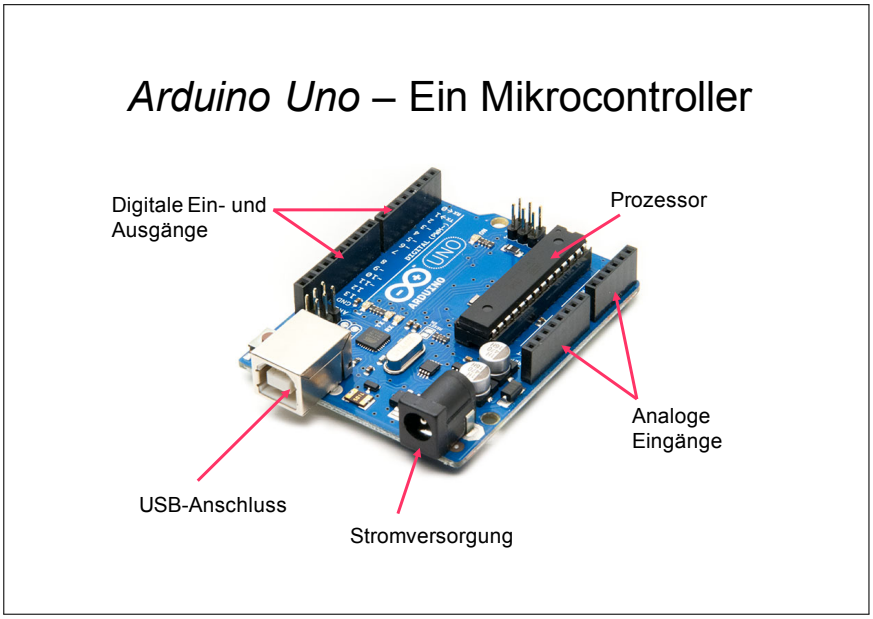


Abb. 3: *ArduinoUno* mit Benennung der wichtigsten Komponenten. Die Bezeichnungen können auch erst nachträglich eingeblendet werden.

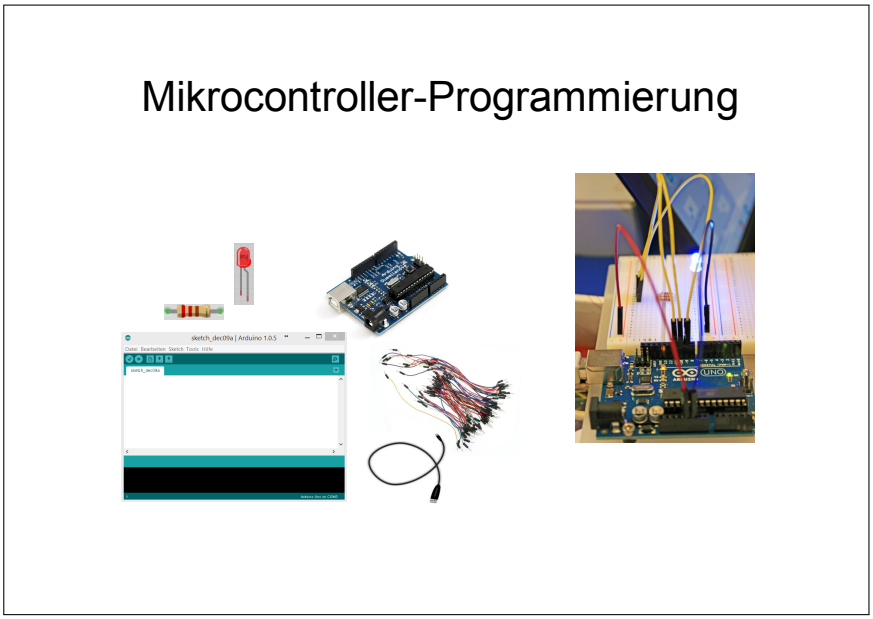


Abb. 4: Übersicht einiger wesentlicher Elemente zur Vorbereitung auf den praktischen Umgang.

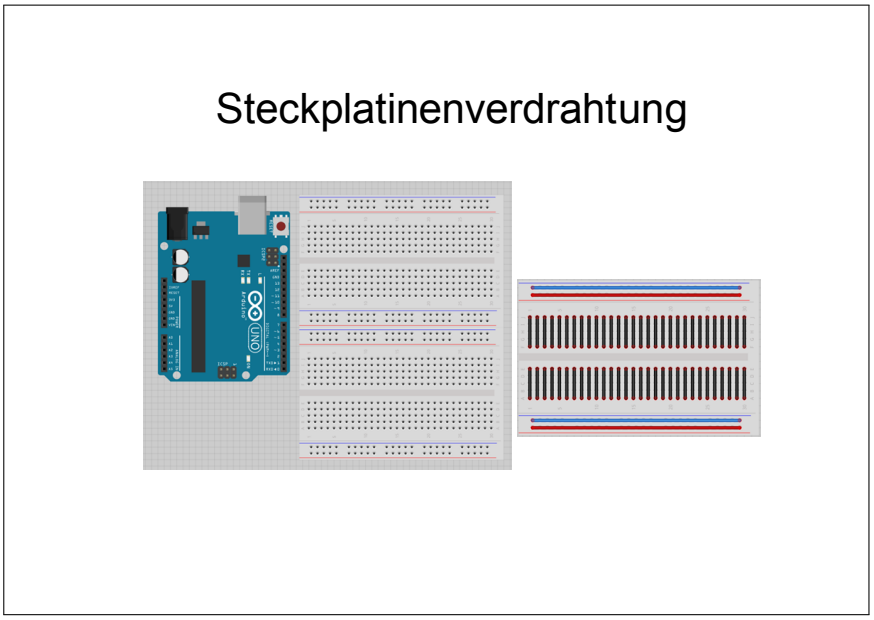


Abb. 5: Steckplatinenlayout. Mit Hilfe dieser Folie kann die Beschaltung der Seckplatine erläutert werden. Die blau bzw. rot markierten Reihen sind horizontal verdrahtet, die restlichen Steckplätze Spaltenweise, d.h vertikal.

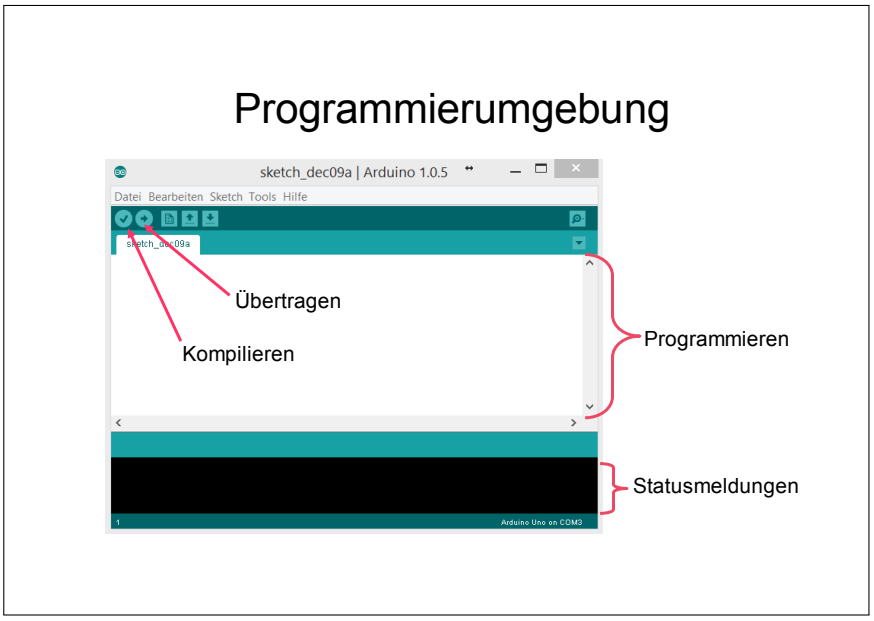


Abb. 6: Arduino Entwicklungsumgegung. Die Folie dient der Veranschaulichung der Arbeitsweise und der wesentlichen Funktionen. Mit den beiden runden Schaltflächen kann das Programm kompiliert (linker Button) bzw. auf den Mikrocontroller übertragen werden (rechte Schaltfläche).

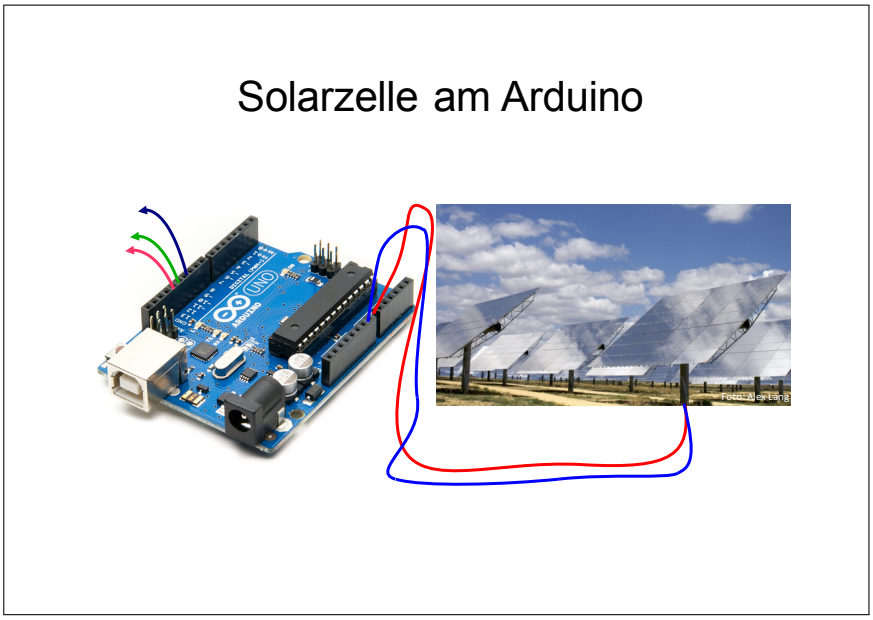


Abb. 7: Schaltungsschema zur Modellierung eines Smart Meters.

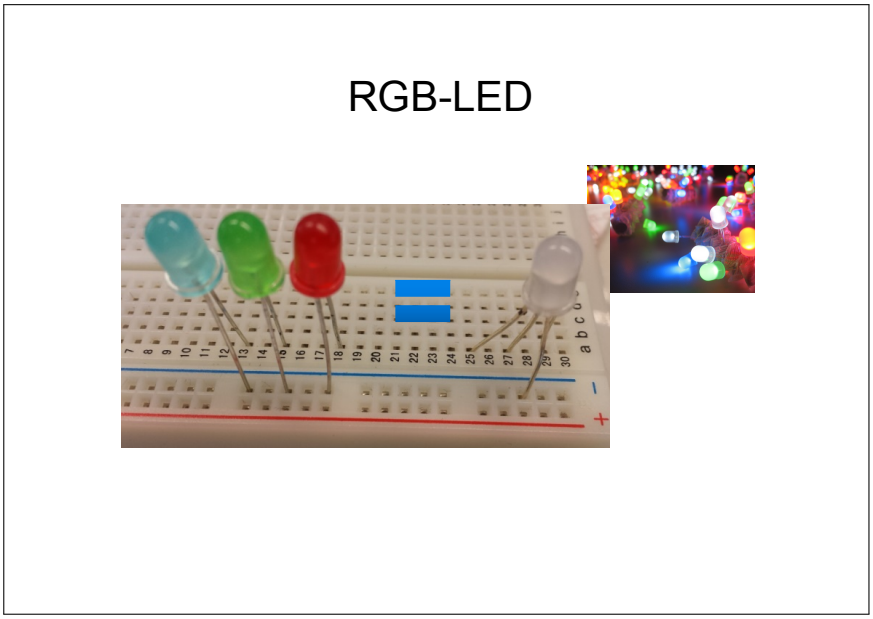


Abb. 8: Veranschaulichung der Funktionsweise einer RGB-LED.

Anhang

C. Arbeitsblatt zur Smart Meter-Programmierung

Datum:	***Smart Meter*-Programmierung**	Blatt-Nr.

Damit ein Smart Meter funktioniert, müssen verschiedenen Funktionen einprogrammiert werden:

Wichtige Funktionalitäten (Features)

1) Warnsignal

Wenn zu wenig Energie zur Verfügung steht, soll dies durch

- leuchten einer LED angezeigt werden.
- blinken einer LED angezeigt werden.
- ein akustisches Warnsignal bekannt gemacht werden.

2) Statusanzeige

Soll auf den ersten Blick zeigen, ob zu wenig, genug, oder mehr als genug Energie produziert wird. Eine RGB-LED zeigt das durch

- die Farben rot, blau, und grün an.
- ihre Helligkeit an: Je mehr Energie erzeugt wird, desto heller leuchtet sie.

3) DEINE IDEE

Aufgaben

1. Überlegen Sie sich im Team, wie Sie ein Warnsignal und eine Statusanzeige programmieren können. Benutzen Sie dazu die Methoden auf der Rückseite. Was benötigen Sie noch? (z.B. Variablen, `if`/`else`-Verzweigung etc.).
2. Setzen Sie ihre Ideen um. Programmieren Sie jeweils zu zweit oder zu dritt den *Arduino* Microkontroller. Starten Sie ihre Arbeit in der Beispieldatei `SM_basics` an der markierten Stelle.
3. Überlegen Sie zusammen mindestens eine weitere Funktion, die ein Smart Meter bereitstellen sollte und programmieren Sie auch diese.

Freie Universität Berlin PhysLab MI.Lab

Methoden und Programmbausteine

Achtung: Die *Methoden* können nur auf einem *Objekt* vom Typ `SmartMeter` aufgerufen werden, z.B. `SmartMeter.statusLEDAn()`, falls ein Objekt `SmartMeter` existiert. (Keine Panik, in den Beispieldateien ist das alles schon erledigt!)

```
// Schaltet die Status-LED an
void statusLEDAn()

// Schaltet die Status-LED aus
void statusLEDAus()

// Status-LED blinken lassen (p ist die Dauer der Pause)
void statusLEDBlink(float p)

// Stellt Farbe und Helligkeit der RGB-LED ein. Die Parameter
// können Werte zwischen 0 (aus) und 255 (sehr hell) annehmen
void RGB(float rot, float gruen, float blau)

// Erzeugt einen Piepton mit einer Unterbrechng der Länge p
void pieptonAn(float p)

// Schaltet den Piepton aus
void pieptonAus()

// gibt aktuellen Wert der Spannung an Solarzelle zurück
float gibSpannung()

// Schreibt Nachricht im Abstand von p Sekunden auf
// die serielle Konsole
void schreibeAufKonsole(int Nachricht, float p)
void schreibeAufKonsole(float Nachricht, float p)
void schreibeAufKonsole(String Nachricht, float p)
```

Tips & Tricks

Es ist hilfreich die aktuellen Spannungswerte der Solarzelle zu beobachten. Dafür muss eine Variable vom Typ `float` deklariert werden, in der dann mit der Funktion `gibSpannung` der aktuelle Wert gespeichert werden kann. Danach kann der Wert mit der Funktion `schreibeAufKonsole` ausgegeben und kontrolliert werden.

```
// Spannungswert der Solarzelle in der Variablen
// 'spannung' speichern
spannung = SmartMeter.gibSpannung();

// Spannungswert auf der seriellen Konsole ausgeben
SmartMeter.schreibeAufKonsole(spannung,1);
```

Freie Universität Berlin PhysLab MI.Lab

D. Quellcode der Smart Meter-Bibliothek

Diese folgenden Dateien stellen zusammen die Smart Meter-Bibliothek dar.

Listing 1: `SmartMeter.cpp`. Die lautsprachigen Funktionsbezeichner ermöglichen den SuS einen intuitiven Gebrauch der verschiedenen Funktionen. Syntax: C++.

```
/*  Smart Meter v.0.1
 *  Author: Christoph van Heteren-Frese
 *
 *  Bibliothek zur Simulation eines Smart Meter mit Hilfe
 *  des ArduinoUno
 */

#include "Arduino.h"
#include "SmartMeter.h"

// Konstruktor
SmartMeter::SmartMeter() {
  _zaehler = 0;
  _zaehlerKonsole = 0;
  pinMode(13, OUTPUT);
  pin1 = 13;
}

// Schaltet die Status-LED an.
void SmartMeter::statusLEDAn() {
    digitalWrite(pin1, HIGH);
}
// Schaltet die Status-LED aus.
void SmartMeter::statusLEDAus() {
    digitalWrite(pin1, LOW);
}
// Schaltet den Piepton aus.
void SmartMeter::pieptonAus() {
    digitalWrite(8, LOW);
}
// Erzeugt einen Piepton der Intervalllänge 'pause'.
void SmartMeter::erzeugePiepton(float pause) {
    if (_zaehlerPiepton == 0 && _piept == false) {
        _piept = true;
        _zaehlerPiepton = pause*1000;
        digitalWrite(8, HIGH);
    } else if (_zaehlerPiepton == 0 && _piept == true) {
        _piept = false;
        _zaehlerPiepton = pause*1000;
        digitalWrite(8, LOW);
    } else {
        _zaehlerPiepton--;
```

```
        delay(1);
    }
}
// Lässt die Status-LED im Interval 'pause' blinken.
void SmartMeter::statusLEDBlink(float pause) {
    if (_zaehler == 0 && _ledOn == false) {
        _ledOn = true;
        _zaehler = pause*1000;
        digitalWrite(pin1, HIGH);
    } else if (_zaehler == 0 && _ledOn == true) {
        _ledOn = false;
        _zaehler = pause*1000;
        digitalWrite(pin1, LOW);
    } else {
        _zaehler--;
        delay(1);
    }
}
// Gibt die gemessene Spannung der Solarzelle zurück.
float SmartMeter::gibSolarSpannung() {
    return (5.0 / 1024.0 * analogRead(1));
}
// Schreibt eine Integerzahl 'text' im Intervall 'pause' auf
   die serielle Konsole.
void SmartMeter::schreibeAufKonsole(int text, float pause) {
    if (_zaehlerKonsole == 0 ) {
        _zaehlerKonsole  = pause*1000;
        Serial.println(text);
    } else {
        _zaehlerKonsole--;
        delay(1);
    }
}
// Schreibt eine Gleitkommazahl 'text' im Intervall 'pause'
   auf die serielle Konsole.
void SmartMeter::schreibeAufKonsole(float text, float pause) {
    if (_zaehlerKonsole == 0 ) {
        _zaehlerKonsole  = pause*1000;
        Serial.println(text);
    } else {
        _zaehlerKonsole--;
        delay(1);
    }
}
// Schreibt einen String 'text' im Intervall 'pause' auf die
   serielle Konsole.
void SmartMeter::schreibeAufKonsole(String text, float pause)
   {
    if (_zaehlerKonsole == 0 ) {
```

```
        _zaehlerKonsole  = pause*1000;
        Serial.println(text);
    } else {
        _zaehlerKonsole--;
        delay(1);
    }
}
// Schaltet die RGB-LED an. 'rot', 'gruen' und 'blau' steuern
   den entsprechenden Farbanteil.
void SmartMeter::RGB(int rot, int gruen, int blau)  {
    analogWrite(6, rot);
    analogWrite(5, gruen);
    analogWrite(3, blau);
}
```

Listing 2: `SmartMeter.h`. Headerdatei mit Prototypen und Variablendeklarationen. Syntax: C++.

```
#ifndef SmartMeter_h
#define SmartMeter_h

#include "Arduino.h"

class SmartMeter
{
  public:
    SmartMeter();
    void statusLEDAn();
    void statusLEDAus();
    void statusLEDBlink(float);
    void erzeugePiepton(float);
    void pieptonAus();
    void aktualisiere();
    float gibSolarSpannung();
    float gibSpannung();
    void RGB(int,int,int);
    void schreibeAufKonsole(int,float);
    void schreibeAufKonsole(String,float);
    void schreibeAufKonsole(float,float);
    int pin1;
    int pause;
  private:
    int _zaehler;
    int _zaehlerKonsole;
    int _zaehlerPiepton;
    bool _ledOn;
    bool _piept;
    bool _konsolenAusgabe;
```

```
};
#endif
```

Listing 3: keywords.txt. In diese Datei werden alle Schlüsselwörter für das Syntax Highlighting gespeichert.

```
SmartMeter          KEYWORD1
blink               KEYWORD2
gibSpannung         KEYWORD2
aktualisiere        KEYWORD2
schreibeAufKonsole  KEYWORD2
RGB                 KEYWORD2
statusLEDBlink      KEYWORD2
statusLEDAn         KEYWORD2
statusLEDAus        KEYWORD2
pieptonAn           KEYWORD2
pieptonAus          KEYWORD2
```

Listing 4: `Beispiel_1.ino`. Beispielprogramm dass je nach Spannungswert die RGB-LEDunterschiedlich färbt und zusätzlich bei Unterversorgung die Status LED blinken sowie ein Signalton ertönen lässt.

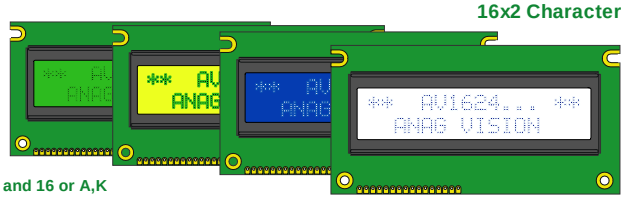
```
// SmartMeter Bibliothek einbinden
#include <SmartMeter.h>

// SmartMeter-Objekt erzeugen
SmartMeter SmartMeter;

// Variablen deklarieren
float spannung;

void setup() {
  // Serielle Konsole starten
  Serial.begin(9600);
}

void loop() {
  // Spannungswert holen
  spannung = SmartMeter.gibSpannung();

  // Spannungswert auf der Konsole ausgeben
  SmartMeter.schreibeAufKonsole(spannung,1);

  // Falls Spannung "mittelmäßig"
  if (spannung < 3 && spannung > 2 ) {
```

```
    // Status-LED aus, piepsen aus
    SmartMeter.statusLEDAus();
    SmartMeter.pieptonAus();

    // Färbe RGB-LED blau
    SmartMeter.RGB(0,0,128);
  }
  // Falls Spannung "niedrig"
  else if (spannung < 2 ) {
    // Status-LED blinkt, Singnalton ertönt
    SmartMeter.statusLEDBlink(0.1);
    SmartMeter.erzeugePiepton(0.2);

    // Färbe RGB-LED rot
    SmartMeter.RGB(128,0,0);
  }
  // Sonst ist Spannung "gut"
  else {
    // Status-LED aus, piepsen aus
    SmartMeter.statusLEDAus();
    SmartMeter.pieptonAus();

    // Färbe RGB-LED grün
    SmartMeter.RGB(0,128,0);
  }
}
```

E. Technische Dokumentation des LCD-Moduls

ANAG VISION AV1624

16x2 Character



- 5x7 dots with cursor
- 1/16 duty
- +5V single supply
- Built in Controller (KS0066 or Equivalent)
- B/L driven by pin1 and 2, 15 and 16 or A,K

Pin Assignment

No.	Symbol	Function
1	V_{SS}	Gnd, 0V
2	V_{dd}	+5V
3	V_0	LCD Drive
4	RS	Function Select
5	R/W	Read/Write
6	E	Enable Signal
7-14	DB0-DB7	Data Bus Line
15	A*	4.2V for LED
16	K	Power Supply for LED 0V

Absolute Maximum Rating

Item	Symbol	Standard Value			Unit
		min.	typ.	max.	
V-Module	Vdd-Vss	-0.3	- - -	7.0	V
V-Input	VI	-0.3	- - -	Vdd	V

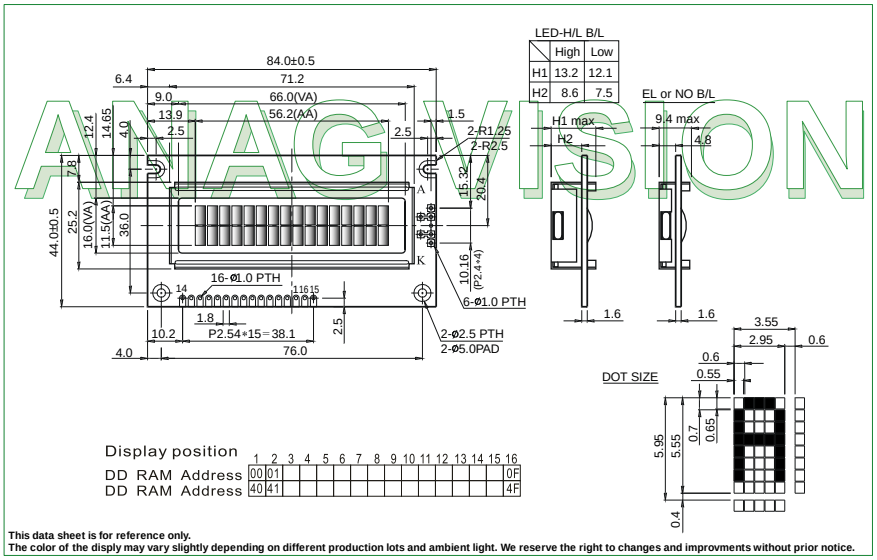
Vss=0V, Vdd=5.0V

Mechanical Data

Item	Standard Value	Unit
Module Size	84.0 x 44.0	mm
Viewing Area	66.0 x 16.0	mm
Dot Size	0.55 x 0.55	mm
Character Size	2.95 x 5.55	mm

Electronical Characteristics

Item	Symbol	Condit.	Standard Value			Unit
			min.	typ.	max.	
Input Voltage	Vdd	Vdd=+5V	4.7	5.0	5.3	V
Supply Curent	Idd	Vdd=+5V	- - -	1.2	1.5	mA
Recommended LC Driving Voltage for Standard Temp. Modules	Vdd-V0	0 °C	- - -	- - -	4.2	V
		25 °C	- - -	3.8	- - -	
		50 °C	3.5	- - -	- - -	
LED Forward Voltage	Vf	25 °C	- - -	4.2	4.6	V
LED Forward Current	If	25 °C	- - -	130	195	mA
LED weiß Voltage *	ILED	3.5 V	30	40	50	mA



This data sheet is for reference only.
The color of the disply may vary slightly depending on different production lots and ambient light. We reserve the right to changes and improvments without prior notice.

Verfügbar STN:	gelb-grün reflectiv positiv	gelb-grün LED positiv LED gelb	blau negativ LED weiß	grau positv LED weiß
CONRAD Best.-Nr:	183342	184594	181651	181664