

## 2. Von der Sonne zum PC

Solarzellen werden überall dort zur Energieversorgung von Geräten eingesetzt, wo kein Stromanschluss zur Verfügung steht oder nur mit hohen Kosten hergestellt werden kann. Jeder von uns hat bereits Parkscheinautomaten in den Städten gesehen, die mit einer Solarzelle ausgestattet sind und die Energie der Sonne in elektrische Energie wandeln. Hier ist es möglich, aufgrund der sparsamen Elektronik auf eine konventionelle Energieversorgung zu verzichten und Kosten für die Verbindung des Parkscheinautomaten mit dem Stromnetz zu sparen. Durch die Solarenergie ist es zudem möglich, stärker die Nutzung von nicht-regenerativen Energiequellen zu verzichten und damit unsere Umwelt und unsere Energiereserven zu schonen. Dies und natürlich auch der finanzielle Aspekt führt dazu, dass immer mehr Photovoltaik-Anlagen auf vielen deutschen Dächern elektrische Energie liefern und sie in das Stromnetz einspeisen.

Bei der Erzeugung der elektrischen Energie ist deren Messung unabdingbar, um zum einen die Qualität der Anlagen beurteilen zu können und zum anderen ggf. Abrechnungsinformationen über die ins Netz gespeiste Energie zu erhalten. Dabei ist es wichtig, die Daten nicht nur auf einem Display ablesen zu können, sondern sie für den Benutzer auf seinem PC oder Notebook verfügbar zu machen.



Abbildung 1 Komponenten einer Datenübertragungstrecke

Im Rahmen der ersten Unteraufgabe wird eine solche Messung zunächst theoretisch vorbereitet. Hier werden die Grundlagen der von uns entwickelten Datenübertragungstrecke (eine Auswahl der Komponenten aus Abbildung 1) betrachtet, wobei in diesem Teil der Aufgabe ein Mikrocontroller mit seinen Komponenten thematisiert wird. Mit diesem Wissen werdet ihr einen Datenlogger aufbauen und euch mit einer akustischen Datenübertragung befassen, sodass innerhalb der dritten Unteraufgabe die Eigenschaften „unserer“ Anlage genauer untersucht werden können. Für die akustische Datenübertragung werdet ihr neben dem Mikrocontroller, der für diese spezielle Aufgabe von uns programmiert wurde, ein Piezoelement und einen selbstgebauten Resonanzkörper verwenden, damit die Daten sicher euren PC erreichen. Erst dann, also in der vierten Unteraufgabe, wird es möglich sein, die übertragenen Daten zu decodieren (entziffern) und ggf. in einer Tabelle für die weitere Verarbeitung vorzubereiten.

## 2.1 Signalverarbeitung

Dieser Aufgabenteil soll einen Einstieg in die Technik des Mikrocontrollers bieten. Ein Mikrocontroller kann als ein PC im Kleinformat gesehen werden. Er verfügt über flüchtigen (RAM) und nichtflüchtigen (FLASH/EEPROM) Speicher, hat einen CPU<sup>1</sup>-Kern und kann mithilfe von Peripherie-Bausteinen (z.B. A/D-Wandler) mit der Außenwelt kommunizieren. Die sich in dem FLASH des Mikrocontrollers befindlichen Software steuert dabei die einzelnen Funktionsblöcke und verleiht ihm dadurch seine Funktionalität. Für die Intel® Leibniz Challenge wurde der Mikrocontroller mit einer, extra für diese Aufgabe geschriebenen, Software programmiert. Im Unterschied zum PC kann diese Software nicht ohne Weiteres geändert werden, da hierzu ein spezielles Programmiergerät notwendig ist. Durch diese Software übernimmt der Mikrocontroller in unserem Datenlogger die Aufgabe eines Messgerätes, das eine analoge Spannung ( $U_{div}$ ), die im Bereich von 0 bis 1,1 Volt liegt, in einen Zahlenwert ( $A_{10bit}$ ) übersetzt und diesen Wert nach der Überführung in Morse-Code akustisch ausgibt. Das Blockschaltbild (vgl. Abbildung 2) verdeutlicht diese Funktionalität, wobei hier noch ein Spannungsteiler dargestellt ist, auf den wir später zu sprechen kommen.

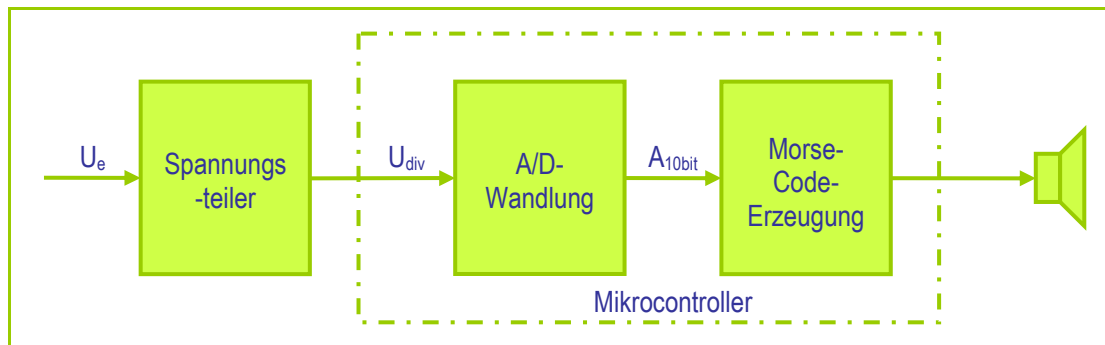


Abbildung 2 Blockschaltbild des Senders

Der integrierte A/D-Wandler (analog-to-digital converter bzw. ADC) innerhalb des Mikrocontrollers überführt, wie bereits angedeutet, die analoge Spannung in einen auswertbaren digitalen Wert. Bei dem analogen Signal handelt es sich um ein kontinuierliches Signal, das innerhalb eines Wertebereiches jeden beliebigen Wert annehmen kann. Im Gegensatz dazu ist ein digitales Signal sowohl zeit- als auch wertdiskret. Dies bedeutet, dass das Signal nur zu bestimmten Zeitpunkten definiert ist und nur bestimmte abgestufte und abzählbare Werte annehmen kann.

1. Welche Art der Analog-Digital-Wandlung kommt beim verwendeten Mikrocontroller zum Einsatz?

Hinweis: Bei dieser Frage ist es empfehlenswert auf die Datenblätter für den verwendeten Mikrocontroller zurückzugreifen: [http://www.atmel.com/dyn/resources/prod\\_documents/2545S.pdf](http://www.atmel.com/dyn/resources/prod_documents/2545S.pdf) und [http://www.atmel.com/dyn/resources/prod\\_documents/doc2545.pdf](http://www.atmel.com/dyn/resources/prod_documents/doc2545.pdf) (Kapitel 23)

2. Nennt drei weitere Arten der Analog-Digital-Wandlung!

Eine Analog-Digital-Wandlung erfolgt in der Regel in zwei Schritten. Zunächst wird das analoge kontinuierliche Signal in einzelne Messwerte zerlegt, die zu unterschiedlichen Zeitpunkten aufgenommen werden. Es entsteht ein zeitlich diskretes Signal, wobei hier von einer Abtastung des Signals gesprochen wird. Im Anschluss an diesen Schritt erfolgt die Wandlung der Werte in diskrete Zahlenwerte. Damit wird das analoge Signal in ein binäres Format mit einem endlichen Wertebereich überführt. Die Fachbezeichnung für diesen Schritt lautet Quantisierung.

<sup>1</sup> Central Processing Unit

## 2.1 Signalverarbeitung – Fortsetzung I

In der folgenden Aufgabe sollen zwei analoge Signale, die in Abbildung 3 dargestellt sind, abgetastet werden. Ihr übernehmt somit den ersten Schritt der A/D-Wandlung, wobei ihr mit unterschiedlichen Abtastfrequenzen arbeiten werdet. Die Abtastfrequenz (bzw. Abtasthäufigkeit) bestimmt, wie oft Werte aus dem analogen Signal übernommen werden.

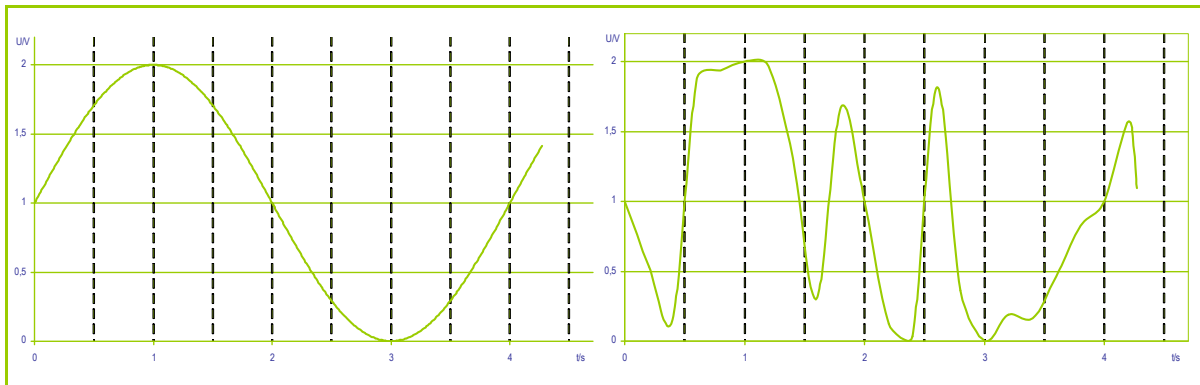


Abbildung 3 Analoge Signale

- 3) Überführt die analogen Signale aus Abbildung 3 in zeitdiskrete Signale mit einer Abtastfrequenz von 1 Hz. Beginnt mit der Abtastung zum Zeitpunkt  $t_0 = 0$  s und zeichnet die beiden zeitdiskreten Signale in jeweils ein Diagramm bis zum Zeitpunkt  $t_1 = 4$  s.

**Hinweis:**

- Abtastfrequenz von 1 Hz bedeutet, dass das Signal ein Mal pro Sekunde abgetastet wird.
- Verwendet für diese und die nächste Aufgabe die Vorlage aus Abbildung 4!

- 4) Wiederholt die Abtastung mit einer Abtastfrequenz von 4 Hz und zeichnet die beiden zeitdiskreten Signale in jeweils ein Diagramm. Beschreibt in einem Satz, welchen Vorteil die höhere Abtastrate für die Wandlung hat.

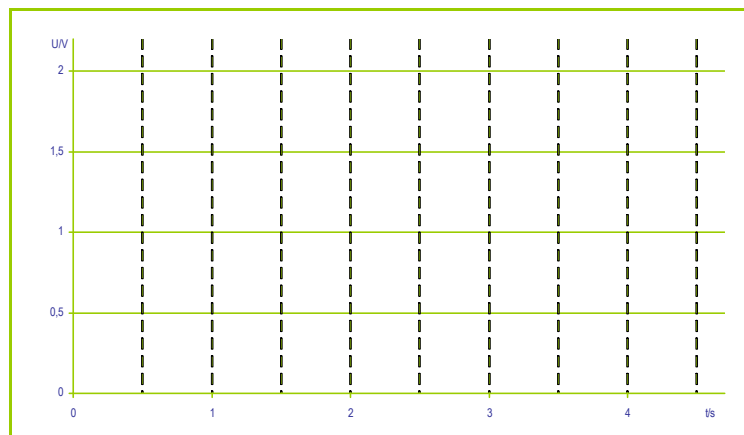


Abbildung 4 Vorlage für die Lösung der Aufgabe 3 und 4

## 2.1 Signalverarbeitung – Fortsetzung II

Für den zweiten Schritt der Wandlung wird eine Referenzspannung benötigt, die den kleinsten durch den A/D-Wandler erkennbaren Unterschied zwischen zwei Werten der analogen Spannung maßgeblich bestimmt.

- 5) Die Software des Mikrocontrollers verwendet eine interne Referenzspannung von 1,1 V als Bezug für die A/D-Wandlung. Wie groß ist der kleinste auflösbare Spannungsunterschied bei maximal möglicher Auflösung der Wandlung?

Bei der Verwendung der internen Referenzspannung des Mikrocontrollers muss zusätzlich ein Spannungsteiler verwendet werden, der aus einem einfachen Widerstandsnetzwerk aufgebaut sein kann.

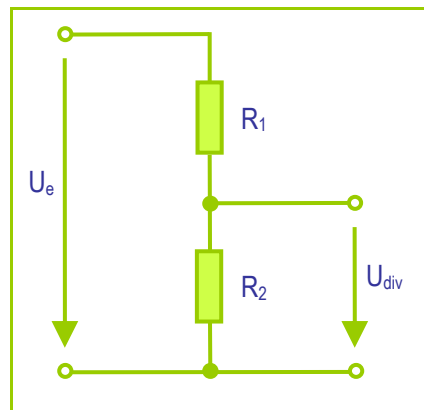


Abbildung 5 Spannungsteiler

- 6) Berechnet die Ströme  $I_{R1}$  und  $I_{R2}$  sowie die Spannung  $U_{div}$ . Betrachtet dabei den Spannungsteiler in Abbildung 5 mit den Werten für  $U_e = 3,85 \text{ V}$ ,  $R_1 = 30 \text{ k}\Omega$  und  $R_2 = 10 \text{ k}\Omega$ !
- 7) Entwickelt eine Formel, mit der ihr aus dem Zahlenwert der Analog-Digital-Wandlung des Mikrocontrollers direkt die an der Solarzelle anliegende Spannung berechnen könnt.  
Hinweis: Verwendet dabei die Erkenntnisse aus der Aufgabe 2.1.5 und 2.1.6!
- 8) Für Langzeitmessungen ist dieser Mikrocontroller praktischerweise mit einem EEPROM zur nicht flüchtigen Ablage von Messergebnissen ausgestattet. Wie viele reine Messwerte lassen sich maximal direkt im EEPROM des Mikrocontrollers ablegen?

### Form der Lösung für die Unteraufgabe 2.1

- Antworten auf die Fragen 1 – 2
- 2 Diagramme als Antwort auf die Frage 3
- 2 Diagramme und die Nennung des Vorteils als Antwort auf Frage 4
- Antworten auf Frage 5 – 6 inkl. Rechenweg
- Formel als Antwort auf Frage 7
- Antwort auf Frage 8 inkl. Rechenweg

## 2.2 Datenlogger

Für die ersten Messungen wird der Mikrocontroller als Multimeter-Ersatz mit akustischer Ausgabe statt einem Zeigerinstrument oder einer digitalen Anzeige benutzt. So kann sich der Benutzer auf die Schaltung konzentrieren und muss nicht die Anzeige im Blick behalten, da der gemessene Wert auf Tastendruck sofort akustisch wiedergegeben wird. Für die Langzeitmessungen reicht ein Multimeter bzw. unser Multimeter-Ersatz jedoch nicht aus, sodass ein Datenlogger benötigt wird. Der Mikrocontroller bzw. die Software übernimmt auch diese Aufgabe. Den Zeitraum für die Langzeitmessungen haben wir in der Software auf 12 Stunden festgelegt. Er umfasst zum Zeitpunkt der Aufgabenbearbeitung die komplette Sonnenscheindauer eines Tages. Ein normales Multimeter wäre mit dieser Aufgabe überfordert und ihr sollt auch nicht alle paar Minuten Messwerte aufnehmen müssen. Die Wiedergabe der Messwerte erfolgt ebenfalls akustisch per Morsezeichen, wobei wir uns nicht wie der Erfinder des Morsecodes Samuel Morse im Jahr 1833 nur auf die Zahlen beschränken (vgl. Tabelle 1). Es ist somit nicht nur möglich Messwerte sondern auch andere Botschaften zu senden. Die Verwendung des Morsecodes erlaubt euch zum einen die Werte selbst zu entziffern und zum anderen eine automatisierte Verarbeitung am PC.

Tabelle 1 Morsecode

Zeichen	Code	Zeichen	Code	Zeichen	Code	Zeichen	Code	Zeichen	Code
A	· —	L	· — · ·	W	· — — —	7	— — — · · ·	&	· — — · · ·
B	— · · ·	M	— — —	X	— · · — —	8	— — — — · ·	:	— — — — · · · ·
C	— · — ·	N	— ·	Y	— · — — —	9	— — — — — ·	;	— · — · — — ·
D	— · ·	O	— — — —	Z	— — — · ·	.	· — · — — —	=	— · · — — —
E	·	P	· — — — ·	0	— — — — — —	,	— — — · · — —	+	· — — — — ·
F	· · — ·	Q	— — — · —	1	· — — — — —	?	· · — — — · ·	-	— · · — — —
G	— — — ·	R	· — ·	2	· · — — — —	'	· — — — — — ·	_	· · — — — — ·
H	· · · ·	S	· · ·	3	· · · — — —	!	— · — · — — —	"	· — · — — · ·
I	· ·	T	—	4	· · · · — —	/	— · · — — ·	\$	· · · — — — —
J	· — — — —	U	· · —	5	· · · · ·	(	— · — — — ·	@	· — — — — ·
K	— · —	V	· · · —	6	— · · · ·	)	— · — — — —		

Während ihr wahrscheinlich den Anfang und das Ende einer Übertragung ohne Probleme erkennen würdet, benötigt ein PC je ein bestimmtes Zeichen, das diese beiden Vorgänge erkennbar macht. Diese besonderen Zeichen werden als Signalcodes (engl. Prosigns) bezeichnet (vgl. Tabelle 2), die aus zwei Buchstaben bestehen und in Gegensatz zur gewöhnlichen Zeichen ohne Pause übertragen werden. Um die Signalcodes bei der Beschreibung erkennbar zu machen, verwenden wir eine besondere Schreibweise: CT (<sup>C</sup><sub>T</sub>) signalisiert den Beginn einer Übertragung („Commence Transmission“) und AR (<sup>A</sup><sub>R</sub>) das Ende einer Übertragung („All Right“).

Tabelle 2 Kodierung der Sonderzeichen (nach ITU-R M.1677)

Zeichen	Code	Zeichen	Code	Zeichen	Code
CT	— · — · —	AR	· — · — ·	Fehler	· · · · · · ·

## 2.2 Datenlogger – Fortsetzung I

- 1) Was unterscheidet einen Datenlogger von einem einfachen Messgerät/Multimeter und welche Aufgaben muss ein Datenlogger erfüllen?
- 2) Nennt zwei mögliche Einsatzgebiete für einen Datenlogger.

Nun könnt ihr mit dem Aufbau des Datenloggers beginnen. Beachtet dabei die Beschreibung des Bausatzes und unsere Tipps für die Fehlersuche. Falls ihr noch keine Erfahrungen mit dem Aufbau elektronischer Schaltungen habt, so könnt ihr zuerst eine kleine Beispielschaltung aufbauen, um den Umgang mit dem Board und den Bauteilen ein wenig zu üben. Die Beschreibung einer solchen Beispielschaltung könnt ihr ebenfalls von unserer Internetseite herunterladen.

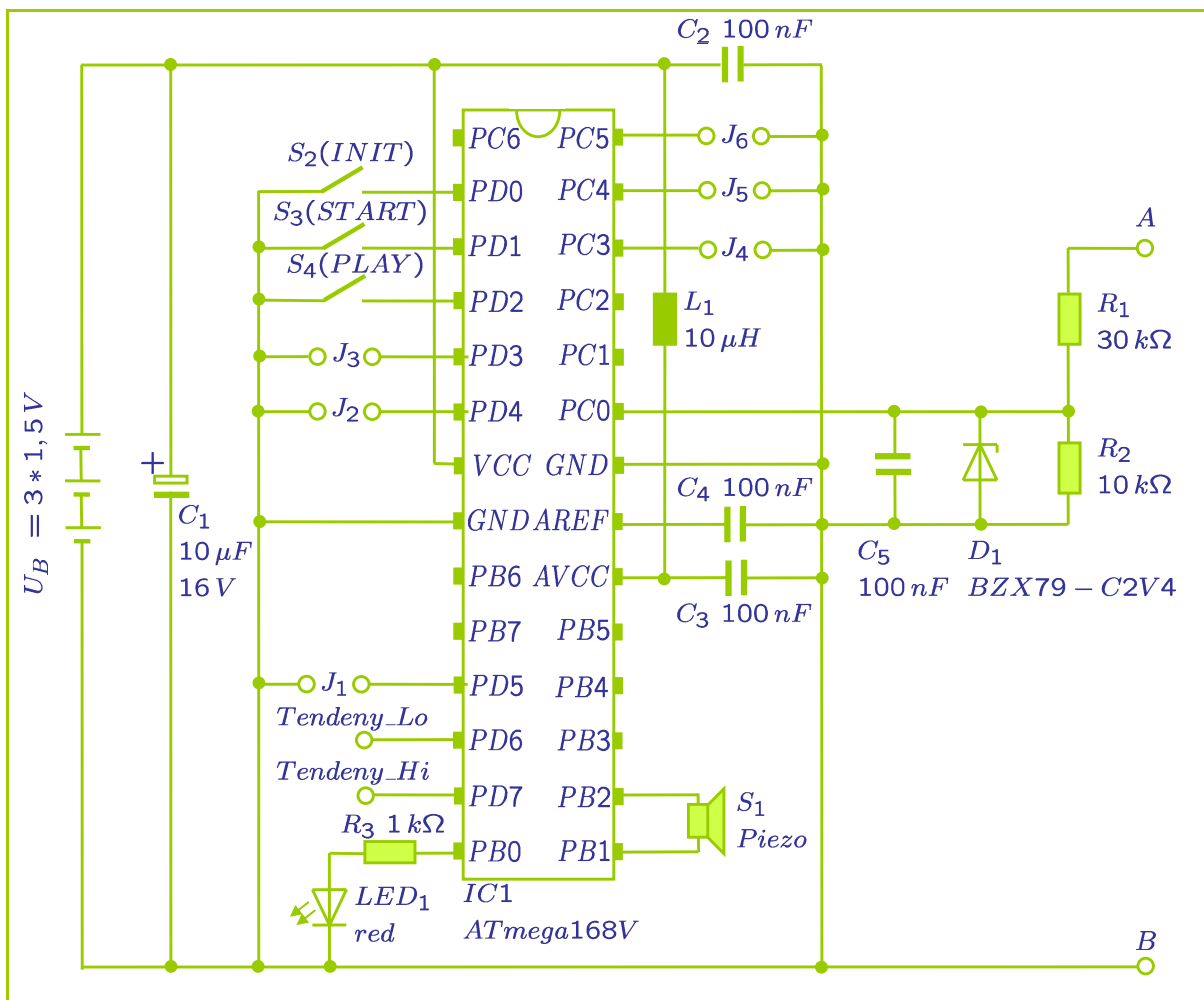


Abbildung 6 Grundsaltung des Datenloggers

## 2.2 Datenlogger – Fortsetzung II

- 3) Baut nun die Grundschialtung des Datenloggers nach dem Schaltplan (vgl. Abbildung 6) auf. Gebt als Lösung dieser Teilaufgabe ein Foto vom funktionierenden Aufbau ab.

Hinweis: Die ausführliche Anleitung zum Aufbau und Funktionsweise des Datenloggers befindet sich in der Datei „ILC2011\_Datenlogger.pdf“. Entnehmt diesem Dokument, welche Drahtbrücken (J1 bis J6) für die Aufgabe zwei und für die langsame Ausgabe der Morsezeichen gesteckt werden müssen.

- 4) Spielt die Einschaltmeldung des Datenloggers ab und wertet sie aus. Welcher Text wird übertragen?

### Form der Lösung für die Unteraufgabe 2.2

- Antworten auf die Fragen 1 – 2
- Foto des aufgebauten Datenloggers als Antwort auf die Frage 3
- Antwort auf die Frage 4

## 2.3 Solarzelle

Wie ihr bereits in der Einleitung erfahren habt, soll nun die Solarzelle näher untersucht werden. Als Messinstrument werdet ihr nun den Datenlogger verwenden, an den ihr die Solarzelle gemäß des Schaltplans in Abbildung 7 anschließt. Mit dieser Erweiterung werdet ihr die Ausgangsspannung der Solarzelle messen. Hiermit ist es möglich eine der wichtigsten Eigenschaften der Solarzelle zu untersuchen, wobei es sich hier um die abgegebene elektrische Leistung handelt. Die abgegebene elektrische Leistung ist proportional zur Ausgangsspannung, die im Folgenden bei unterschiedlichen „Betriebsarten“ der Solarzelle gemessen wird.

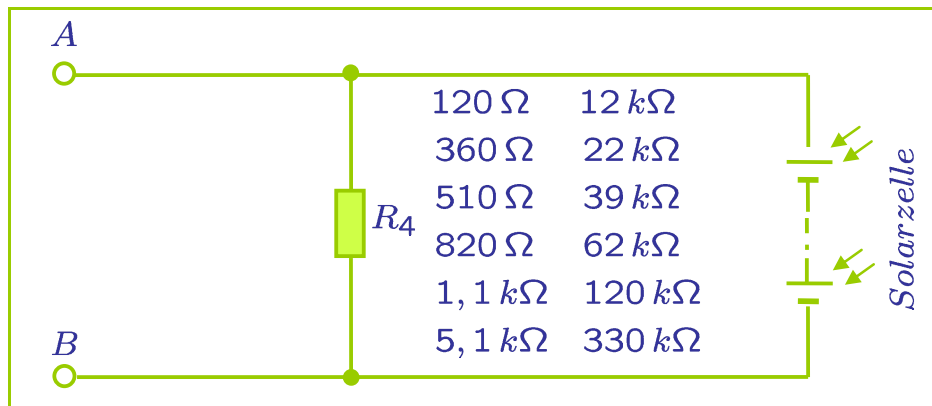


Abbildung 7 Erweiterung des Datenloggers

Der Widerstand  $R_4$ , der die Solarzelle belastet, wird zunächst nicht verwendet. Um die ersten Messungen vorzunehmen, muss der Datenlogger in die Betriebsart für Einzelmessungen gesetzt werden. In diesem Modus arbeitet der Datenlogger als ein gewöhnliches Messgerät. Dazu müsst ihr die Drahtbrücke  $J_2$  öffnen und die Drahtbrücke  $J_1$  schließen.

**WARNUNG:** Wenn ihr eine sehr helle Lichtquelle verwendet, achtet darauf, nicht mit den Augen in die Lichtquelle zu schauen!

- 1) Die von der Solarzelle erzeugte Ausgangsspannung ist abhängig von der Entfernung zu einer Lichtquelle. Nehmt eine beliebige Lichtquelle, die hell genug ist (z.B. eine 11 W Energiesparlampe). Erstellt ein Diagramm (Ausgangsspannung in Abhängigkeit von der Entfernung,  $U_e = f(d)$ ) mit fünf Messwerten. Nehmt die einzelnen Messwerte mit unterschiedlichen Abständen der Solarzelle zur Lichtquelle.
- 2) Stellt nun den Abstand der Lichtquelle so ein, dass die Solarzelle eine Ausgangsspannung von 3,2 V liefert. Haltet diesen Abstand nun konstant und messt die Spannungswerte für verschiedene Widerstände, die ihr als  $R_4$  in die Schaltung einsetzt. Tragt die Ausgangsspannung in Abhängigkeit von den Widerstandsgrößen in einem Diagramm auf ( $U_e = f(R)$ ).

Hinweis: Verwendet die in Abbildung 7 angegebenen Widerstandswerte.

- 3) Auch die vom Licht bestrahlte Fläche beeinflusst die Ausgangsspannung. Um zu sehen, was in realen Anwendungen ein Verdecken durch Wolken, Laub oder Schnee bedeutet, sollt ihr nun die Solarzelle mit Pappe abdecken. Entfernt den Widerstand  $R_4$  wieder aus der Schaltung und stellt den Abstand so ein, dass die Solarzelle eine Ausgangsspannung von 3,2 V liefert. Messt nun die Spannung ohne Pappe,  $\frac{1}{4}$  verdeckt,  $\frac{1}{2}$  verdeckt,  $\frac{3}{4}$  verdeckt und ganz verdeckt. Gebt die gemessenen Werte in einer Tabelle an und führt die Untersuchung für zwei „Verdeckungsrichtungen“ (parallel zu den Solarzellen; senkrecht zu den Solarzellen) durch. Beschreibt das beobachtete Verhalten in maximal 3 Sätzen.

Hinweis: Verwendet die Tabelle 3 als Vorlage.

## 2.3 Solarzelle – Fortsetzung

Tabelle 3 Vorlage für die Aufgabe 2.3.3

	Horizontales Verdecken	Vertikales Verdecken
	Ausgangsspannung in V	Ausgangsspannung in V
ohne Pappe		
1/4 verdeckt		
1/2 verdeckt		
3/4 verdeckt		
vollst. verdeckt		

Während für die vorausgegangenen Aufgaben Einzelmessungen durchgeführt wurden, werdet ihr nun den „Automatik-Modus“ verwenden, in dem eine Langzeitmessung möglich ist. Hierzu müsst ihr die Drahtbrücke  $J_2$  schließen. Nun wird der Datenlogger nach dem Start der Messung für 12 Stunden (länger als die Sonnenscheindauer eines Tages) stündlich einen Messwert aufzeichnen. Dabei bildet der Datenlogger automatisch jeweils einen Mittelwert aus mehreren Messungen, die er öfter als nur einmal in der Stunde durchführt.

Beachtet bitte, dass die Übertragung und das Anhören der 12 Messwerte einige Zeit in Anspruch nehmen wird. Nehmt am besten die „gemorsten“ Töne als Wave-Datei auf und hört sie dann nacheinander an. Zum Aufnehmen von Wave Dateien (WAV) bietet sich z.B. die kostenlose Software Audacity an. Diese könnt ihr unter <http://audacity.sourceforge.net/> herunterladen. Die mit Audacity aufgenommenen Dateien sollten als WAV-Datei exportiert werden. Für die spätere automatische Dekodierung empfiehlt es sich bereits jetzt, die Aufnahme möglichst ohne Hintergrundgeräusche anzufertigen.

- 4) Wie verhält sich die Leistung der Solarzelle im Verlauf eines Tages? Erstellt ein Diagramm über den Tagesverlauf (Ausgangsspannung in Abhängigkeit von der Zeit,  $U_e = f(t)$ ).

Hinweis: Um diese Frage zu beantworten legt ihr den Datenlogger auf ein Fensterbrett, richtet die Solarzelle nach außen und startet möglichst kurz vor Sonnenaufgang die Langzeitmessung. Der Tagesverlauf wird dadurch aufgezeichnet. Hört euch anschließend die Messwerte an und berechnet jeweils die Spannungswerte, die die Solarzelle geliefert hat.

- 5) Zu welchem Tageszeitpunkt wurde die maximale Leistung abgegeben? Erläutert warum das Maximum erreicht wurde.
- 6) Die Messreihe bildet den tatsächlichen Verlauf der Sonnenstrahlung nicht gut ab. Nennt mindestens eine Möglichkeit, wie die Messreihe verbessert werden kann?

### Form der Lösung für die Unteraufgabe 2.3

- Je ein Diagramm als Antwort auf die Frage 1 und 2
- Tabelle sowie kurze Beschreibung (max. 3 Sätze) als Antwort auf Frage 3
- Diagramm als Antwort auf die Frage 4
- Antworten auf die Frage 5 und 6

## 2.4 Morse Decoder Skript

Herzlichen Glückwunsch zum erfolgreichen Aufbau des Datenloggers. Wie ihr gemerkt habt, ist das Abhören der Aufzeichnung ein recht langwieriger und fehleranfälliger Prozess. Zum Glück gibt es Computer, die uns in allen Lebenslagen bei der Auswertung großer Datenberge unterstützen können. Wie beim Mikrocontroller wird auch hier nur die richtige Software benötigt. Für die automatische Auswertung des Morsecodes vom Datenlogger sollt ihr nun diese Software erstellen. Mit ihr könnt ihr den Datenlogger dann auch im „Schnellabspiel-Modus“ betreiben. Doch keine Panik, wir geben einige Programmteile vor, damit es nicht ganz so schwer wird.

Der Vorteil von den meisten Automatisierungs- und Datenverarbeitungsaufgaben ist die Tatsache, dass es nicht unbedingt einer komplizierten Anwendung in einer sog. Hochsprache wie C++ bedarf, um sie zu lösen. Tatsächlich ist oft eine Skript-Sprache die beste Wahl. Skript-Sprachen zeichnen sich in erster Linie dadurch aus, dass sie interpretiert werden. D.h. der Programmcode wird nicht durch einen Kompilervorgang in ein ausführbares Programm („EXE“) übersetzt, sondern quasi direkt ausgeführt. Der sogenannte Interpreter wandelt den geschriebenen Text zur Laufzeit in die abzuarbeitenden Maschinenbefehle um. Zudem sind Skript-Sprachen meist derart angelegt, dass sie bestimmte Programmierdetails vereinfachen, wie z.B. Variablen- oder Speicherverwaltung. Für neue Aufgaben ist daher die Einstiegsschwelle besonders niedrig, um Ergebnisse schnell zu erzielen. Für diese Aufgabe haben wir Perl als Skriptsprache gewählt, wobei wir eine kurze Anleitung für euch erstellt haben, mit der ihr Perl installiert und das erste kleine Skript erstellt.

Zur Datenanalyse dieser Aufgabe haben wir bereits ein Perl-Skript erstellt, das ihr ebenfalls von unserer Seite herunterladen könnt. Dieses Skript liest eine WAV-Datei und extrahiert aus ihr die „gemorsten“ Zeichen. Die WAV-Dateien selbst sind ein beliebtes und sehr einfaches Datenformat zur Darstellung von Tönen. Im einfachsten Fall setzt sich eine WAV-Datei wie folgt zusammen:

- Am Anfang der Datei befindet sich ein kurzer Vorspann, der als Header oder Präambel bezeichnet wird. Im Vorspann wird festgelegt, in welchem Format die nachfolgenden Audiodaten vorliegen.
- Hinter dem Vorspann befinden sich die tatsächlichen Audio-Daten, wobei die einzelnen „Töne“ als „Samples“ (Abtastwerte) verwaltet werden. Für jeden Abtastwert wird eine Zahl gespeichert, die der Größe des Schallausschlags entspricht. Diese Werte können als ein 8- oder 16-Bit-Wort gespeichert werden und für einen („Mono“) oder zwei („Stereo“) Kanäle vorliegen. Auch diese Information kann dem Vorspann entnommen werden.

Kommen wir nun zurück zum Decodieren der Morse-Zeichen. Um die Daten des Datenloggers verarbeiten zu können, müssen sie als eine WAV-Datei vom PC aufgenommen werden. Wie bereits in der Aufgabe 2.3 empfehlen wir auch hier die Verwendung<sup>2</sup> der Software Audacity, die noch einen Vorteil aufweist: Die Audiodaten werden visualisiert, sodass die Töne auch mit dem Auge betrachtet werden können.

Bei der Decodierung geht das Skript schrittweise vor:

- WAV-Datei einlesen und als abgetastete Schallschwingungen interpretieren.
- Anhand der Abtastwerte Zeiten der Tonaktivität und -inaktivität („an“ / „aus“) ermitteln.
- Aus der Zeitdauer der „An-/Aus“-Phasen lange und kurze Töne sowie Pausen erkennen.
- Aus der Folge von Tönen und Pausen die gesendeten Zeichen erkennen.

Die Kommentare im Skript verdeutlichen ggf. die Vorgehensweise zusätzlich, wobei ihr nicht unbedingt die gesamte Datei analysieren müsst. Das Skript berücksichtigt weiterhin, dass die aufgezeichneten Schallwellen sich nicht nur durch „an“ oder „aus“ im Sinne des Morsecodes darstellen, sondern das Piepen an sich auch eine Schwingung ist. D.h. der Verlauf der abgetasteten Schallwellen eines solchen Morse-Tons kann sich in etwa wie hier in Audacity darstellen, wobei Abbildung 8 ein Ausschnitt von ungefähr 27 ms darstellt.

---

<sup>2</sup> Verwendet bitte bei der Aufnahme der Audiodaten die Einstellung „CD-Qualität“ (Abtastrate 44,1 kHz und 16 Bits pro Abtastwert).

## 2.4 Morse Decoder Skript – Fortsetzung I

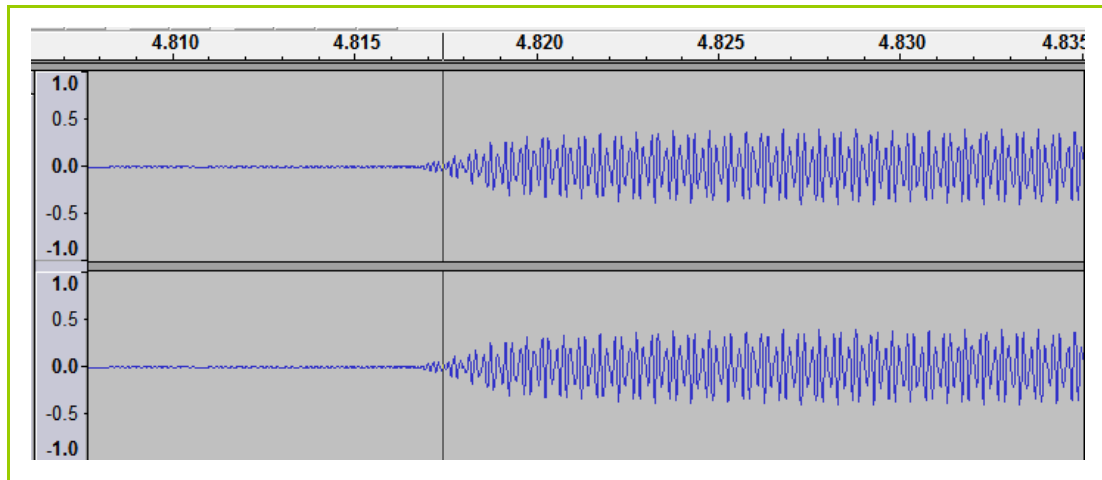


Abbildung 8 Ausschnitt eines Morse-Zeichens

Die An-Aus-Folge wird erst dann erkennbar, wenn ein größerer Abschnitt der WAV-Datei betrachtet wird. Abbildung 9 zeigt die Schwingungen in einem Zeitbereich von ca. 2 s, wobei hier eine sogenannte Hüllkurve deutlich wird. Sie stellt eine Art Umriss der Schwingungen dar und wird von dem Skript für die Erkennung der einzelnen Morse-Töne verwendet.

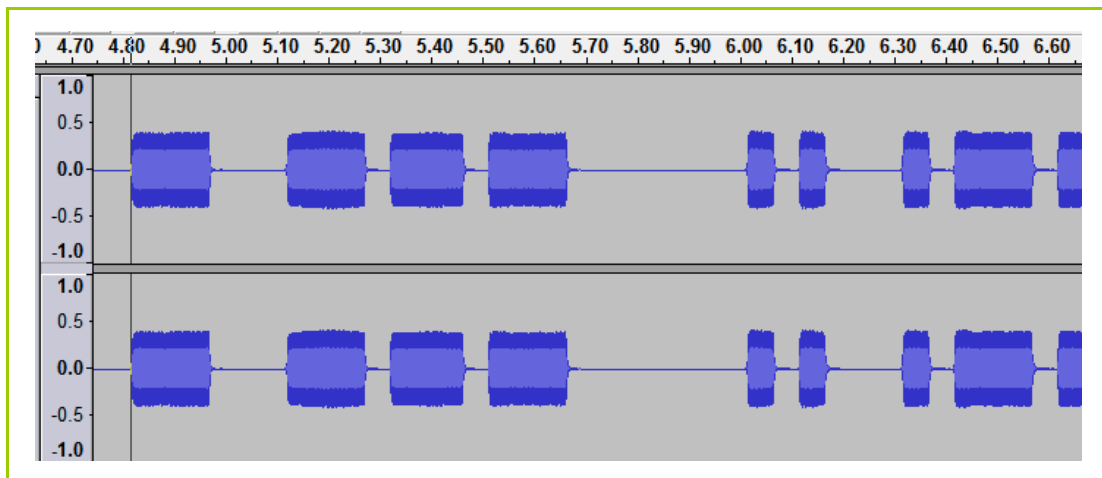


Abbildung 9 Ausschnitt einer Morse-Übertragung

Aus dem Quelltext des mitgelieferten Skriptes sind sowohl elementare Perl-Operationen zu ersehen als auch der oben ausgeführte Lösungsansatz. Im Hauptprogrammteil, das mit dem Wort „MAIN“ im Quelltext markiert wird, werden nacheinander „Unterprogramme“ (sog. Funktionen, wie z.B. „formatermittlung()“) aufgerufen, die – auch ihrem Namen nach – den angegebenen Schritten entsprechen. Zur Lösung der Aufgabe müssen die Programmteile nicht alle im Detail verstanden werden. Ihr sollt euch jedoch mit einzelnen Aspekten beschäftigen:

- 1) Die Morsecode-Tabelle, d.h. die Zuordnung der Lang-Kurz-Sequenzen zu Buchstaben, ist im vorgegebenen Skript als sog. „Hash“ (einer Art Zuordnungstabelle) implementiert. Dieses Hash namens „%buchstabe“ ist jedoch unvollständig, d.h. es fehlen im Quelltext einige Morse-Zeichen. Ergänzt dieses Hash um die fehlenden Zeichen gemäß Tabelle 1.

## 2.4 Morse Decoder Script – Fortsetzung II

- 2) Untersucht den ersten Abschnitt der Funktion „`anauzulangkurz()`“. Es handelt sich dabei um den ersten Block, der bei „`foreach my $anauwert (@anau)`“ beginnt. Erläutert welche Werte dort ermittelt werden und welche Schlussfolgerungen aus ihnen gezogen werden können?
- 3) Untersucht die Funktion „`langkurzzumorsebuchstaben()`“. Welches Ziel verfolgte der Programmierer bei der Erstellung dieser Funktion? Welche Bedeutung haben die benötigten (Eingabe-)Daten und welche (Ausgabe-)Daten werden hier generiert?
- 4) Die Funktion „`morsebuchstabenzutext()`“ ist im mitgelieferten Skript leer und somit noch ohne Funktionalität. Füllt sie mit eigenem Code, der aus der Liste (Array) „`@morsebuchstaben`“ eine Zeichenfolge „`$morsetext`“ erstellt und sie auf dem Bildschirm ausgibt. Verwendet für diese Aufgabe den Hash „`%buchstabe`“ aus dem Aufgabenteil 2.4.1.
- 5) Woher kommen die Daten, die in `@morsebuchstaben` enthalten sind, und welche Bedeutung haben sie?
- 6) Dekodiert mit Hilfe des von Euch modifizierten Skripts die mitgegebene WAV-Datei „`ILC2011_A2.4.wav`“.

Hinweis: Es steht Euch frei, auf anderen kreativen Wegen den gesendeten Text zu ermitteln, falls das Skript nicht funktioniert.

- 7) Der Funktionsblock „`textzuzahlenreihen()`“ ist ebenfalls leer. Eure Aufgabe ist es hier, die empfangenen Messreihen aus Aufgabe 2.3.4 mit dem Skript nicht nur zu Textfolgen umzuwandeln, sondern diesen Text als Folge von Zahlen zu interpretieren. Dafür muss der Programmcode in der Zeichenkette „`$morsetext`“ Leerzeichen finden und Zahlen sowie Buchstaben trennen. Speichert die gefundenen Zahlen in einer selbst gewählten Datenstruktur (z. B. Array, Hash, String), die eine einfache Weiterverarbeitung der Zahlen erlaubt. Gebt den Inhalt dieser Datenstruktur auf dem Bildschirm aus.

### Form der Lösung für die Unteraufgabe 2.4

- Vollständiger Code der Hash-Tabelle (Textdatei) als Antwort auf die Frage 1
- Antworten auf die Fragen 2 und 3
- Vollständige Funktion „`morsebuchstabenzutext()`“ (Textdatei) als Antwort auf die Frage 4
- Antwort auf die Frage 5
- Wortlaut des aufgezeichneten codierten Textes als Antwort auf die Frage 6
- Vollständige Funktion „`textzuzahlenreihen()`“ (Textdatei) als Antwort auf die Frage 7

## Wichtige Informationen

Falls ihr Fragen zu den Aufgaben habt oder eine Hilfestellung benötigt, so schaut doch einfach in unser Forum:

<http://www.intel-leibniz-challenge.de/forum/>

Abgabe der Lösungen:

**Wo:** [www.intel-leibniz-challenge.de/portal](http://www.intel-leibniz-challenge.de/portal)

**Wie:** Genauigkeit der Lösungen

Falls nicht anders gefordert, gebt bei den Lösungen maximal drei signifikante Stellen an (z. B. 1,52 mA, 42,1 kW, 123 V etc.)!

**Form der Abgabe und Dateibenennung:**

Für jede Unteraufgabe soll nur eine Datei abgegeben werden. Falls mehrere Dateien vorhanden sind, müssen sie in eine zip-Datei gepackt werden. Die Datei muss wie folgt benannt werden:

Gruppenname\_Aufgabe\_Unteraufgabe.zip

Für die Abgabe der Aufgabe 2.1 müsste die „Muster Gruppe“ folgende Datei hochladen:

MusterGruppe\_2\_1.zip

Verwendet bitte keine Leerzeichen und Sonderzeichen in den Dateinamen!

**Zulässige Dateiformate:**

Textformate: PDF mit eingebetteten Bildern, txt

Bildformate: jpg, bmp, png, wmf

Videoformate: flv, avi, mpg, ogg

Audioformate: mp3, wma, wav, ogg

**Dateigrößen und Dateiinhalt**

Die Dateien sollten nicht größer als 7,5 MB sein! Bitte gebt in der Datei (nicht im Dateinamen) auch euren Teamnamen, die Namen der Gruppenmitglieder sowie deren Schulen an. Erzeugt dafür eine zusätzliche Textdatei!

**Wann:** Bis zum 03.04.2011 um 23:59 Uhr

**Hinweis:** Um sicher zu gehen, dass eure Dateien wirklich fehlerfrei und für die Korrektoren zu öffnen sind, solltet ihr eure Zip-Dateien nochmals von eurem Account runterladen und öffnen. Dateien, die sich nicht öffnen lassen, können nicht bewertet werden!

Die AGB und weitere Informationen findet ihr unter: [www.intel-leibniz-challenge.de](http://www.intel-leibniz-challenge.de)  
Der Rechtsweg ist ausgeschlossen!