

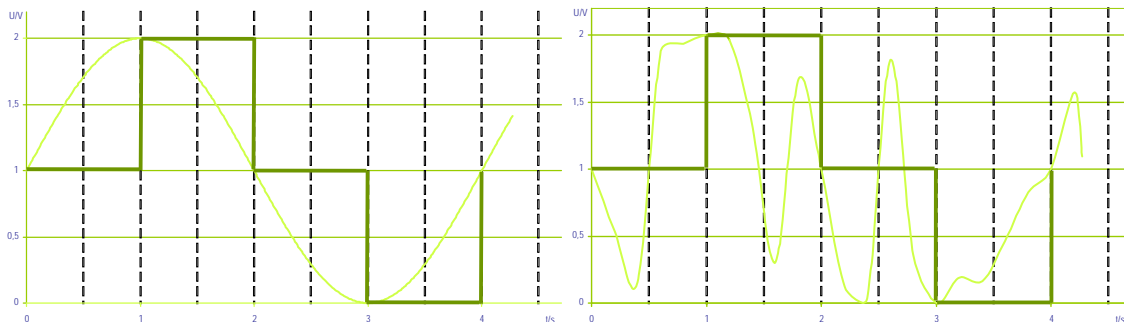
## 2.1 Musterlösung

### Signalverarbeitung

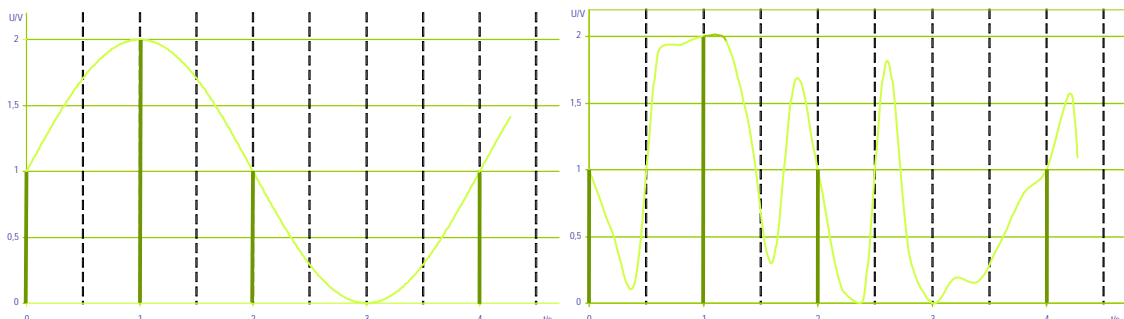
1. In dem innerhalb dieser Aufgabe verwendeten Mikrocontroller ATmega168 wird eine A/D-Wandler verwendet, der nach dem Prinzip der **sukzessiven Approximation** arbeitet („The ATmega48/88/168 features a 10-bit successive approximation ADC.“, Quelle: Datenblatt ATmega168, S. 243).
2. Es gibt einige weitere Verfahren, von denen beispielsweise folgende in Frage kommen:
  1. **Flash-** bzw. **Parallel-**Verfahren
  2. **Single-Slope-**Verfahren
  3. **Dual-Slope-**Verfahren
  4. **Delta-Sigma-**Verfahren
  5. **Spannung-Frequenzumsetzungs**verfahren
  6. **Nachlauf**verfahren

Als Antwort sollten lediglich drei Verfahren genannt werden.

3. Die Abtastung der vorgegebenen Signale führt zu folgenden Ergebnissen, wobei wir das originale Signal hell und die Lösung dunkel abgebildet haben:

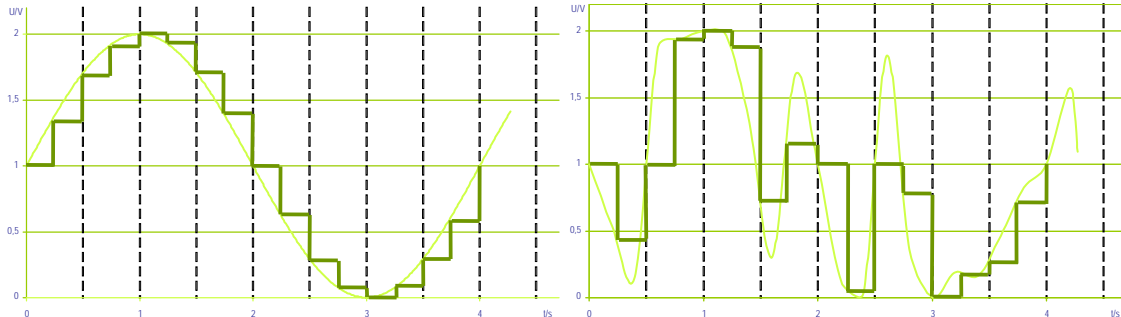


Es ist dabei auch eine andere Darstellung der Lösung denkbar:

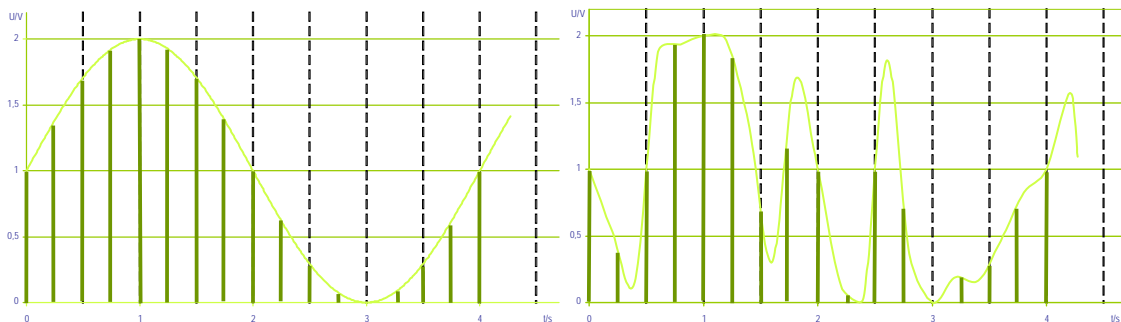


4. Die Abtastung mit einer Abtastfrequenz von 4 Hz liefert Signale, die auf der nächsten Seite dargestellt sind. Es wird deutlich, dass während in der Aufgabe 3 die Signale nicht unterscheidbar sind, an dieser Stelle eine Differenzierung der Signale problemlos möglich ist. Der Vorteil der höheren Abtastfrequenz ist somit die **Erhöhung der Genauigkeit** der Abtastung. In diesem Zusammenhang ist das Nyquist-Shannonsche Abtasttheorem zu berücksichtigen, das an dieser Stelle als Hinweis für die weitere Betrachtung der Thematik angesehen werden kann.

## 2.1 Musterlösung



Auch hier gibt es eine alternative Darstellungsmöglichkeit:



5. Aus dem Datenblatt geht hervor, dass der AD-Wandler eine Auflösung von 10 Bit besitzt („The ATmega48/88/168 features a 10-bit successive approximation ADC.“, Quelle: Datenblatt ATmega168, S. 243). Aus diesem Grund sind 1024 ( $=2^{10}$ ) darstellbar (0...1023). Bei einer Referenzspannung von 1,1 V ergibt sich somit eine Auflösung von **1,07 mV**:

$$U_{step} = \frac{U_{ref}}{n}$$

$$U_{step} = \frac{1,1 \text{ V}}{2^{10}}$$

$$U_{step} = 1,07 \text{ mV}$$

6. Die Anwendung der Spannungsteilerregel und des ohmschen Gesetzes führt zu einer Spannung  $U_{div} = 963 \text{ mV}$ . Da es sich hier um eine Reihenschaltung handelt sind die Ströme  $I_1$  und  $I_2$  identisch und betragen **96,3  $\mu\text{A}$** .

$$U_{div} = U_e \cdot \frac{R_2}{R_1 + R_2}$$

$$U_{div} = 3,85 \text{ V} \cdot \frac{10 \text{ k}\Omega}{30 \text{ k}\Omega + 10 \text{ k}\Omega}$$

$$U_{div} = 963 \text{ mV}$$

$$I_1 = \frac{U_e}{R_1 + R_2}$$

$$I_1 = \frac{3,85 \text{ V}}{30 \text{ k}\Omega + 10 \text{ k}\Omega}$$

## 2.1 Musterlösung

$$I_1 = 96,3 \mu A$$

$$I_2 = I_1$$

$$I_2 = 96,3 \mu A$$

7. Bei der Entwicklung der Formel muss beachtet werden, dass die gemessene Spannung zunächst die Eingangsspannung  $U_e$  des Spannungsteilers ist. Erst die Ausgangsspannung des Spannungsteilers  $U_{div}$  wird dem A/D-Wandler zugeführt und in die entsprechenden Werte umgewandelt:

$$A_{A/D} = \frac{U_{div}}{U_{step}}$$

$$A_{A/D} = \frac{U_e \cdot \frac{R_2}{R_1 + R_2}}{\frac{U_{ref}}{n}}$$

$$A_{A/D} = \frac{n \cdot R_2}{U_{ref} \cdot (R_1 + R_2)} \cdot U_e$$

$$U_e = \frac{U_{ref} \cdot (R_1 + R_2)}{n \cdot R_2} \cdot A_{A/D}$$

$$U_e = \frac{1,1 \text{ V} \cdot (30 \text{ k}\Omega + 10 \text{ k}\Omega)}{2^{10} \cdot 10 \text{ k}\Omega} \cdot A_{A/D}$$

$$U_e \approx 4,3 \text{ mV} \cdot A_{A/D}$$

8. Der EEPROM des Mikrocontrollers kann laut Datenblatt 512 Bytes speichern (vgl. S. 6 des Datenblattes), wobei 1 Byte 8 Bits umfasst. Überdies muss beachtet werden, dass ein A/D-Wandler-Wert aus 10 Bits besteht, sodass folgende zwei Möglichkeiten denkbar sind:

1. Jeder Wert des A/D-Wandlers wird als 2 Bytes im Datenspeicher abgelegt. Dies führt dazu, dass **256 Werte** gespeichert werden können:

$$n_{A/D-Value} = \frac{512 \text{ Byte}}{2 \frac{\text{Byte}}{\text{Value}}}$$

$$n_{A/D-Value} = 256 \text{ Value}$$

2. Eine aufwändigere und speicherplatzschonere Lösung ist das Abspeichern von 5 A/D-Wandler-Werten in 4 Bytes, sodass insgesamt **409 Werte** gespeichert werden können:

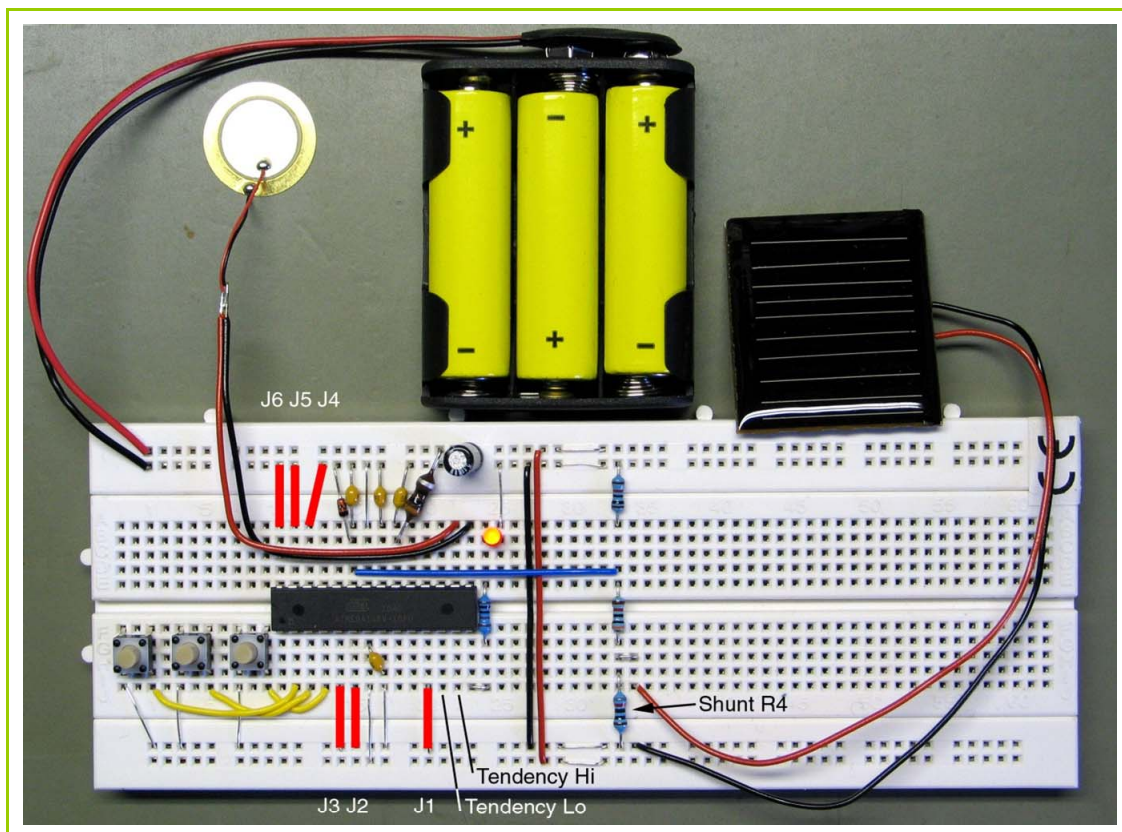
$$n_{A/D-Value} = \frac{512 \text{ Byte}}{5 \frac{\text{Byte}}{4 \text{ Value}}}$$

$$n_{A/D-Value} = 408 \text{ Value}_{comp} + 1 \text{ Value}_{uncomp}$$

## 2.2 Musterlösung

### Datenlogger

- Ein Messgerät/Multimeter setzt die gemessene Größe direkt in einen Anzeigenwert um und gibt diesen aus. Es erfolgt keine längerfristige Speicherung. Ein Datenlogger ist ein Gerät für die **Aufnahme von Messreihen über einen längeren Zeitraum** in zuvor festgelegten Intervallen. Die Messdaten werden **dauerhaft gespeichert** und meist erst nach Abschluss der Messreihe (an einem PC) **ausgewertet**.
- Die Einsatzgebiete eines Datenloggers sind sehr vielfältig, sodass an dieser Stelle nur einige Beispiele genannt werden können:
  - **Wetterstation**: Aufzeichnung von Klimadaten über den Tagesverlauf
  - **Smartmeter** (intelligenter Stromzähler): Aufzeichnung von Verbrauchsdaten zu Abrechnungszwecken.
  - Am Körper getragenes **Langzeit-EKG**
  - **GPS Tracker**: Gibt den Verlauf einer abgefahrenen/abgelaufenen Wegstrecke wieder.
  - ...
- Unserer Aufbau des Datenloggers sieht wie folgt aus:



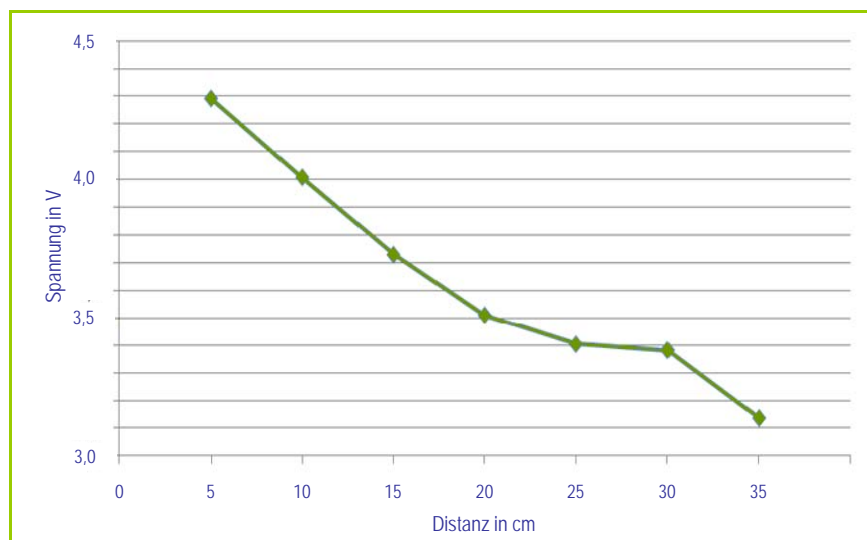
- Der Datenlogger gibt folgende Meldung nach dem Einschalten aus:  
 $^c_T$  WELCOME TO ILC2011, TASK 2  $A_R$

## 2.3 Musterlösung

### Solarzelle

- Die Abhängigkeit der Ausgangsspannung der Solarzelle von der Distanz kann anhand des folgenden Diagramms dargestellt werden, wobei wir statt fünf sieben Werte ermittelt haben, die auch in der Tabelle zusammengefasst sind.

Distanz	Morsecode	ADC Wert	Spannung
5cm	.....	999	4,29 V
10cm	.....	933	4,01 V
15cm	.....	868	3,73 V
20cm	.....	817	3,51 V
25cm	.....	792	3,40 V
30cm	.....	787	3,38 V
35cm	.....	730	3,13 V

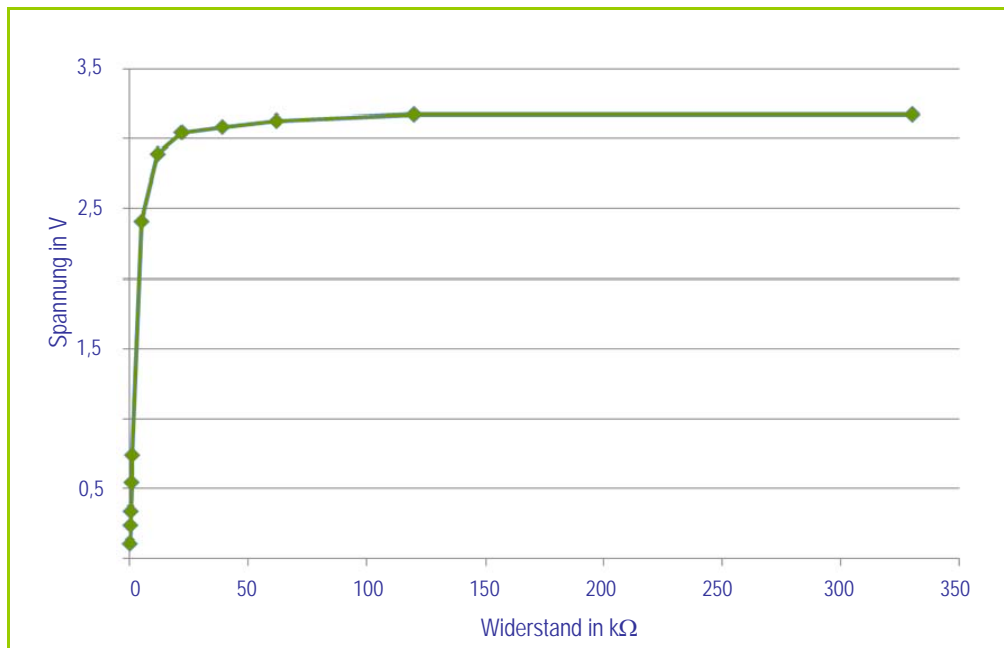


Es ist erkennbar, dass die Spannung mit steigender Entfernung abnimmt. Aus dem Diagramm könnte bereits die notwendige Entfernung für die nächste Aufgabe abgelesen werden. Bei unserem Aufbau erreichen wir die Ausgangsspannung von 3,2V bei einer Entfernung von 33 cm.

- Nach einer Kalibrierung des Aufbaus kann folgendes Diagramm aufgenommen werden, wobei wir auch hier eine Tabelle mit den Messwerten darstellen.

Widerstand in $\Omega$	Morsecode	ADC Wert	Spannung
Ohne	.....	751	3,23 V
120	.....	25	0,11 V
360	.....	55	0,23 V
510	.....	79	0,34 V
820	.....	127	0,55 V
1100	.....	172	0,74 V
5100	.....	561	2,41 V
12000	.....	673	2,89 V
22000	.....	709	3,05 V
39000	.....	718	3,09 V
62000	.....	728	3,13 V
120000	.....	739	3,18 V
330000	.....	739	3,18 V

## 2.3 Musterlösung



Auf dem Diagramm ist erkennbar, dass je größer der Widerstand, desto größer ist die Ausgangsspannung. Dies liegt daran, dass die Solarzellen, die einen Innenwiderstand  $R_i$  haben, immer weniger belastet werden.

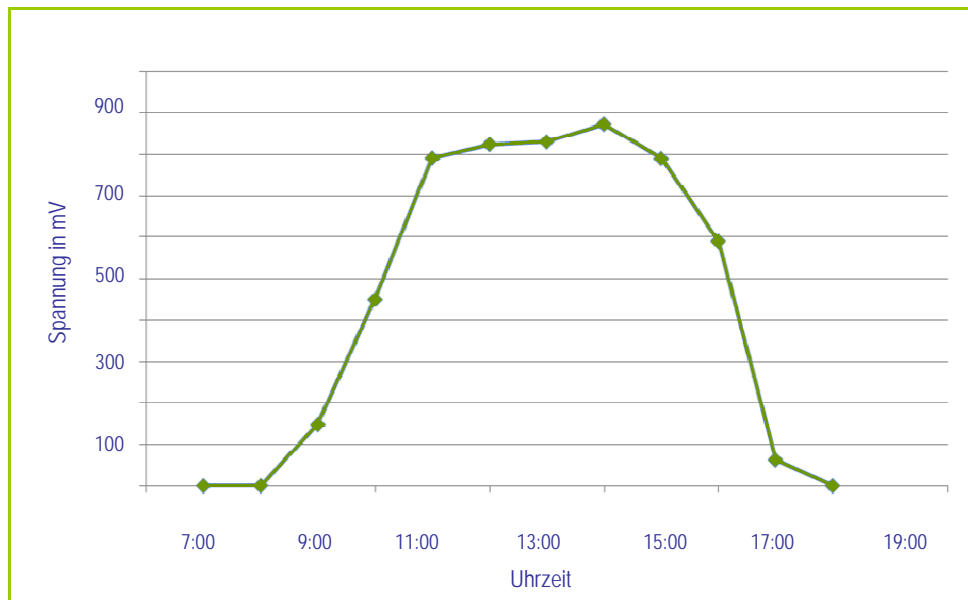
3. Auch in dieser Aufgabe muss der Aufbau zunächst kalibriert werden, wobei selbstverständlich die Kalibrierung aus der vorausgegangenen Aufgabe verwendet werden muss.

	Horizontales Verdecken			Vertikales Verdecken		
	Morsecode	Wert	Spannung	Morsecode	Wert	Spannung
ohne Pappe	.....	748	3,21 V	.....	748	3,21 V
1/4 verdeckt	.....	678	2,91 V	.....	686	2,95 V
1/2 verdeckt	.....	189	0,81 V	.....	611	2,63 V
3/4 verdeckt	.....	58	0,25 V	.....	237	1,02 V
vollst. verdeckt	.....	14	0,06 V	.....	14	0,06 V

Wird die Solarzelle parallel (horizontal) zu den Linien abgedeckt, sinkt die Spannung stark ab. Bei senkrechter Abdeckung (vertikal) nimmt die Spannung langsamer ab. Ursache ist die Tatsache, dass beispielsweise bei der horizontalen Abdeckung (1/2 verdeckt) eine komplette Einzelzelle verdeckt wird.

## 2.3 Musterlösung

4. Unsere Messung haben wir um 7:00 Uhr gestartet, sodass die Messung bis 19:00 Uhr durchgeführt wurde.



5. Ausgehend von den Ergebnissen der Aufgabe 4 wurde das Maximum der Kurve bei uns um 14:00 Uhr erreicht, da offensichtlich die Sonne in günstiger Position bzgl. unseres Fensters steht. Auf einem Feld würde das Maximum um 12:00 Uhr erreicht werden, wenn die Sonne am höchsten steht.

6. Die Kurve in der Aufgabe 4 weist eine hohe Ungenauigkeit auf, die durch folgende Maßnahmen verringert werden könnte:

- Die Messung muss öfter als ein Mal pro Stunde durchgeführt werden, sodass sich **mehr Messpunkte** ergeben.
- Es können „Zwischenmesspunkte“ mathematisch durch **Mittelwertbildung** bestimmt werden, um so den tatsächlichen Verlauf genauer nachzubilden.
- Die Messung kann an mehreren Orten (z. B. Fenster, Garten, Dach etc.) durchgeführt werden, damit durch **die Überlagerung** der Werte ein genauer Verlauf entsteht.

## 2.4 Musterlösung

### Morse Decoder Script

1. Bei der Morsecode-Tabelle fehlten einige Zeichen. Die vollständige Implementierung ist wie folgt:

```

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

```

## 2.4 Musterlösung

- Der erste Foreach-Block in der Funktion „`anauszulangkurz()`“ analysiert die in der Datei detektierten **An-Aus-Zeiten** und verwendet diese Information, um **automatisch den langsamen oder schnellen Morsemodus zu erkennen**. Die Ruhephase am Beginn der Aufnahme – die erste Aus-Zeit – wird ignoriert, da diese nicht Teil der Morseübertragung ist und eine beliebig lange Pause darstellen kann. Diese hängt von der Zeit ab, die zwischen dem Start der WAV-Aufzeichnung und dem Beginn der Datenwiedergabe vom Datenlogger vergangen ist.
- Die Funktion „`langkurzzumorsebuchstaben()`“ hat die Aufgabe die **Tonfolge in Morsebuchstaben umzuwandeln**. Das Array `@toene` (Eingabedaten) enthält in der WAV-Datei gefundenen **Morsezeichen in Form von Kurz, Lang oder Buchstabenpause**. Diese werden in den Strich-Punkt-Code der Morsebuchstaben und anschließend in die Morsebuchstaben selbst übersetzt. Mit der Buchstabenpause wird ein Buchstabe abgeschlossen. Das Ergebnis ist das Array `@morsebuchstaben`, bei dem jedes Element die decodierten **Morsebuchstaben** beinhaltet.
- Die Übertragung der Liste „`@morsebuchstaben`“ in die Zeichenfolge „`$morsetext`“ sowie die Ausgabe der Zeichenfolge kann wie folgt realisiert werden:

```
foreach my $morse (@morsebuchstaben)
{
    $morsetext .= $buchstabe{$morse};
}
print "Ergebnis als Text:\n$morsetext\n";
```

**Hinweis:** Der Code wandelt `\n` nicht in einen Zeilenumbruch um.

- Die Daten, die in „`@morsebuchstaben`“ enthalten sind, werden von der **Funktion „@langkurzzumorsebuchstaben“** erzeugt. Es handelt sich dabei um die **decodierten Morsebuchstaben**.
- Die von uns mitgelieferte WAV-Datei enthält folgende Meldung:  
DAS PFERD FRISST KEINEN GURKENSALAT.
- Unter der Annahme, dass Zahlen aus 4 ASCII-Ziffern bestehen und mit einem Leerzeichen getrennt werden, kann folgender Code verwendet werden:

```
my @daten = grep (/^[0-9]{4}$/ , split (' ', $morsetext));
print "Messwerte:\n";
print join ("\n", @daten);
```