***Hinweis:***
***Dies ist eine gekürzte Version des C-Projektes für unser ATMEL-Board ATM1***

***„Einführung in die Digitale Verarbeitung von Analogen Signalen (= DSP-Grundlagen) mit dem Microcontroller“.***

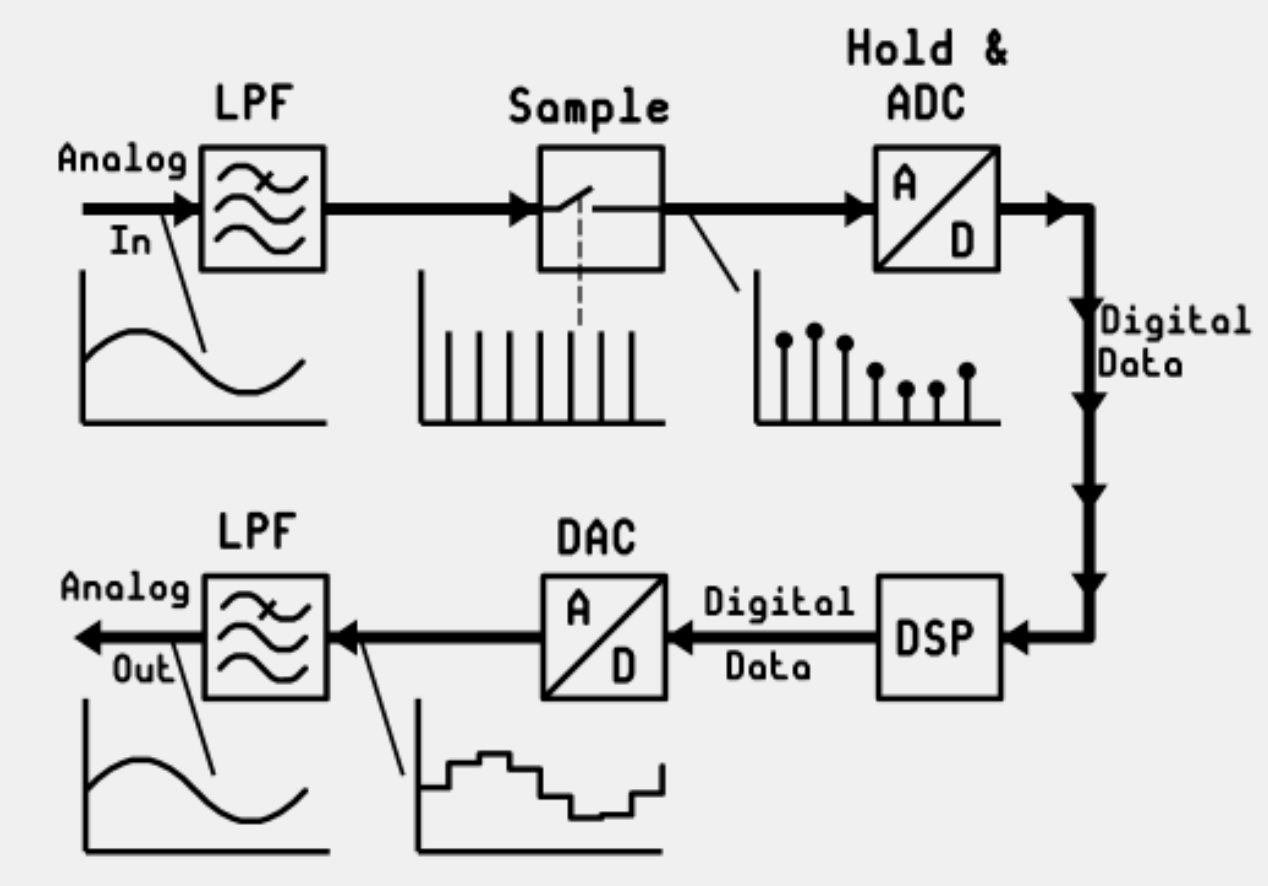
***Das vollständige Projekt incl. aller C-Programme findet sich als Kapitel 12 im Tutorial Band 2 „C-Programme für das ATMEL-Board“.***

## Einführung in die Digitale Verarbeitung von Analogen Signalen (= DSP)

Von Gunthard Kraus

### 1. Prinzip der Digitalen Signalverarbeitung

Das Analogsignal wird zuerst durch einen Tiefpass geschickt und auf diese Weise „bandbegrenzt“. Anschließend speist es eine „Sample - Schaltung“, in der sein Augenblickswert durch extrem kurze „Sample-Impulse“ laufend abgefragt wird. Diese „Abfrage-Ergebnis“ wird in einem Kondensator (= „Hold“) zwischengespeichert und einem Analog-Digital-Wandler zugeführt. So entsteht eine laufende Folge von „probes“ mit einer konstanten Frequenz (= sample rate), die in Form eines seriellen 8 Bit- oder 16 Bit-Datenstroms im darauf folgenden Rechner oder Digitalem Signalprozessor oder in einem Mikrocontroller verarbeitet wird.
Die Bearbeitung im Rechner liefert anschließend ein Ergebnis, als ob man z. B. das Analogsignal durch eine Filterschaltung geschickt hätte (….wobei man durch das Programm festlegt, ob das ein Tiefpass, ein Hochpass, ein Bandpass oder eine Bandsperre ist).
Übergibt man die so bearbeitete Wertefolge einem Digital-Analog-Wandler, so braucht man an dessen Ausgang nur noch einen passenden Tiefpassfilter zur Unterdrückung unerwünschter Signalanteile (= Abtastfrequenz und ihre Oberwellen). So erhält man wieder eine analoges Ausgangssignal.



## Vorteile der Digitalen Signalverarbeitung:

Man kann nun allein durch ausreichende Rechenleistung Schaltungen und Methoden realisieren, bei denen eine „Analoge Ausführung“ viel zu teuer oder nur mit gigantischem technischem Aufwand oder überhaupt nicht zu verwirklichen wäre. Außerdem lassen sich die Filterwerte (Grenzfrequenz, Filtertyp, Sperrdämpfung usw.) sehr leicht durch geänderte Faktoren im Rechenprogramm umschalten. Auch „selbstanpassende“ (= adaptive) Filter lassen sich programmieren, Verschlüsselungen oder Entschlüsselungen können vorgenommen werden usw.

## 2. Signalfrequenzen und Sample Rate

Folgende eiserne Grundregel muss eingehalten werden, wenn das alles richtig funktionieren soll. Sie ist unter dem Namen „**Shannon-Bedingung**“ oder „**Nyquist-Kriterium**“ allgemein bekannt:

Handelt es sich beim Eingangssignal um eine

**rein sinusförmige Spannung mit der Frequenz $f_{ein}$, dann muss die Abtastfrequenz (= Sample Rate) mindestens doppelt so hoch gewählt werden wie diese analoge Eingangsfrequenz.**

(….sicherer ist natürlich eine mehr als doppelt so hohe Sample Rate….).

**Nur dann lässt sich ganz am Ende nach der Digital-Analog-Wandlung und der darauf folgenden Filterung das Analogsignal wieder eindeutig und ohne zusätzliche Fehler rekonstruieren.**

**Aber Vorsicht:**
Sobald die Kurvenform vom Sinus abweicht, enthält ein solches Signal außer der Grundfrequenz ein ganzes Sortiment an zusätzlichen, sinusförmigen Teilspannungen. Das sind die „Oberwellen“ oder „Harmonische“ und sie stellen z. B. bei Audiosignalen den „Klirrfaktor“ dar. Sie sind stets ganzzahlige Vielfache der Grundfrequenz.

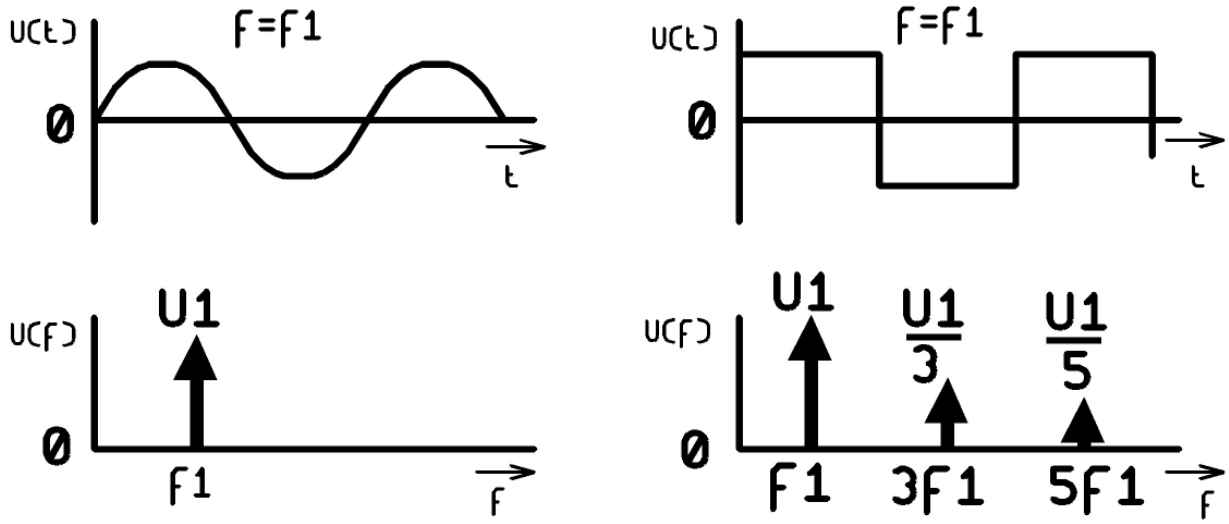
**Beispiel „Sinussignal“**
**Oben:**
**Rein sinusförmige Eingangsspannung auf dem Bildschirm eines Oszilloskops**

**Unten:**
**Zugehöriges Schirmbild beim Spectrum-Analyzer**

**Beispiel „Rechtecksignal“**
**Oben:**
**Rechteckspannung auf dem Bildschirm eines Oszilloskops**

**Unten:**
**Zugehöriges Schirmbild beim Spectrum-Analyzer**

**(Achtung: die negativen Frequenzanteile wurden nicht in die Spektren eingetragen, da sie ein Spektrumanalyzer nicht getrennt anzeigt!)**




Jetzt müssen wir die Shannon-Bedingung präziser formulieren:

**Die Sample Rate muss nun mindestens doppelt so hoch sein wie die höchste vorkommende Einzelfrequenz (oder Oberwelle) im Eingangssignal.**

Wer sich etwas mit Oberwellen auskennt, wird da gleich fragen:

„***Die Oberwellen eines nicht sinusförmigen Signals reichen theoretisch bis Unendlich, aber dafür werden ihre Amplituden immer kleiner. Welche Sample Rate soll ich denn nun wählen?“***

Da gibt es nur eine einzige rigorose Möglichkeit. Nämlich einen Kompromiss, bei dem man irgendwo brutal abschneidet und auf die letzten Feinheiten des Analogsignals verzichtet: **Am Eingang der Schaltung (also noch vor dem A-D-Wandler) wird deshalb ein Tiefpass („Anti-Aliasing-Lowpass-Filter“) mit einer bestimmten Grenzfrequenz und einem sehr, sehr steilen Anstieg der Dämpfung im Sperrbereich eingebaut. Dann wählt man eine Sample Rate, die mehr als doppelt so hoch ist wie die Grenzfrequenz dieses Tiefpasses.** (Beispiel: Bei einem CD-Player beträgt die **Audio-Grenzfrequenz exakt 20 kHz.** Anschießend wird aber mit **44,1 kHz** abgetastet).

Hier steckt man natürlich schon wieder in einer Entscheidungsklemme:
Wählt man die **Tiefpass-Grenzfrequenz sehr niedrig**, dann kommt man mit einer recht niedrigen Sample Rate aus. Das bedeutet weniger Rechenleistung, aber es besteht die Gefahr, dass nun Oberwellen des Eingangssignals fehlen, die noch wichtig für die korrekte Kurvenform des Analogsignals sind.

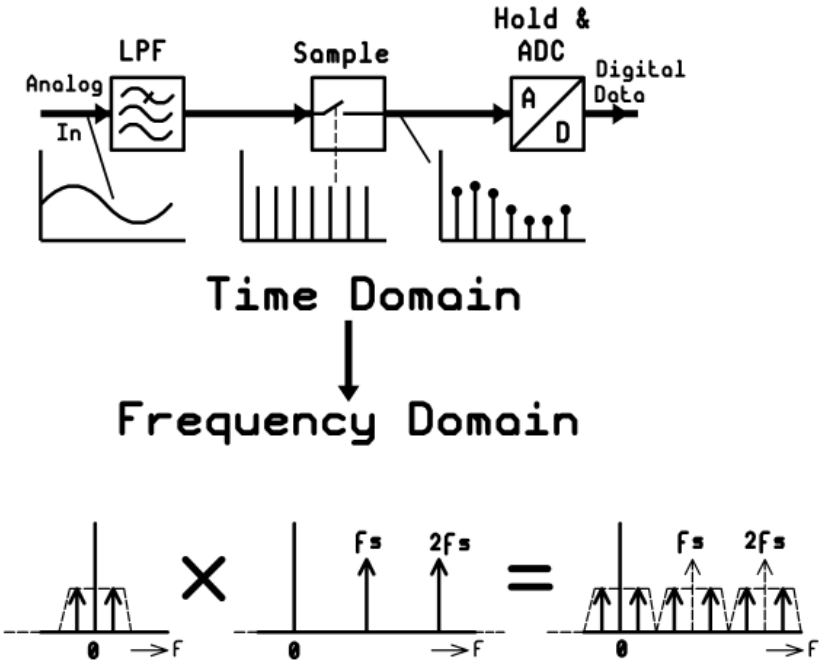
Wählt man dagegen die **Grenzfrequenz und die Sample Rate höher als erforderlich**, dann stimmt hinterher noch die Kurvenform des Audiosignals. Aber es ergeben sich folgende Kehrseiten:

a) Die höhere Bandbreite beim Eingangssignal erhöht den „**Eigenrauschpegel**“ der Anordnung. Das kann bei Nutzsignalen in der Mikrovolt-Größenordnung recht bitter sein.

b) Die höhere Sample Rate erfordert (bei Echtzeit-Betrieb) unter Umständen so **hohe Rechenleistungen**, das selbst modernste PCs und Prozessoren an ihre Grenzen stoßen. Da helfen nur echte „DSPs“ (= Digitale Signalprozessoren), die eine andere Hardware-Struktur aufweisen und z. T. mehr als eine Milliarde 32-Bit-Multiplikationen pro Sekunde ausführen können.

(Zur Information: da die Leistungsfähigkeit der Rechner und DSPs sozusagen stündlich steigt, ist die ausreichende Rechenleistung nicht mehr das Problem. Deshalb wählt man heute oft Lösung b) und tastet viel öfter ab als grundsätzlich notwendig wäre. Der Datenstrom wird dann nach der A-D-Wandlung über **ein „Digitales Decimation-Filter“** geschickt, das die Grenzfrequenz wieder künstlich reduziert und damit auch eine „neue“, niedrigere Sample-Rate zulässt, die nach Shannon nur noch mindestens doppelt so hoch sein muss wie die Informationsbandbreite).
Nun wollen wir zum Abschluss noch folgende Frage klären:

**Was geschieht, wenn die Vorschrift: „Abtastfrequenz immer höher als die doppelte maximale Eingangsfrequenz’“ nicht eingehalten wird?**



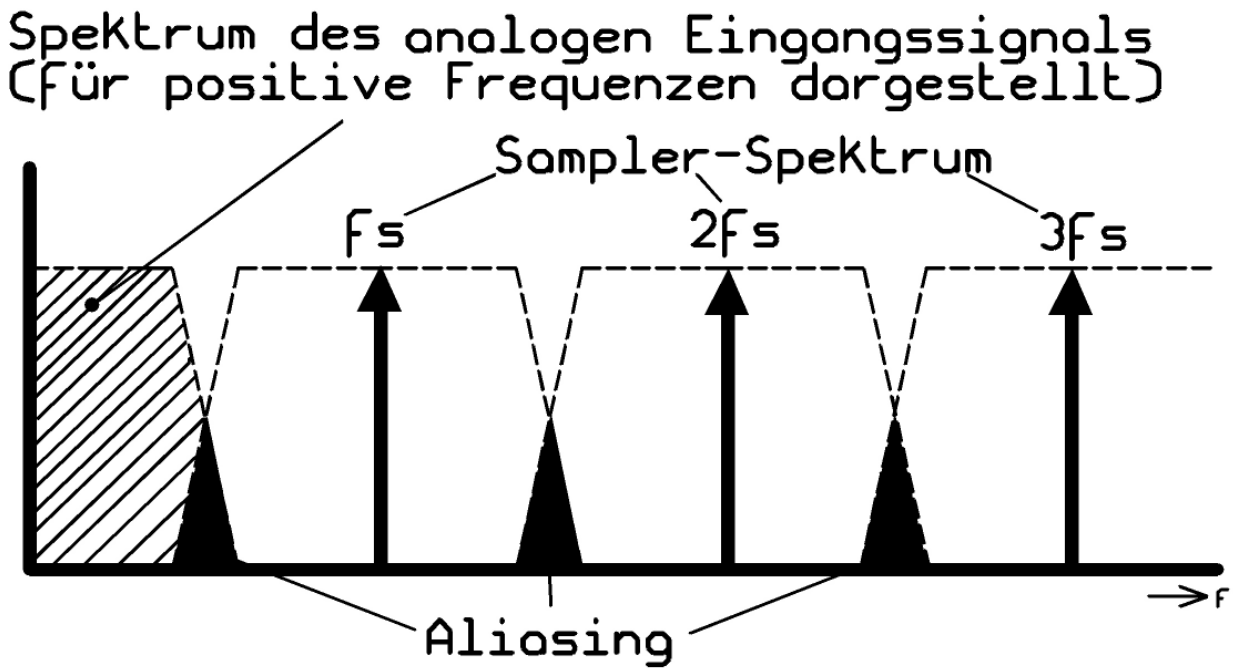

Dazu sehen wir uns alle beteiligten Signale sowohl in der Time Domain wie auch in der Frequency Domain genauer an:

Den Abtastvorgang kann man (im Zeitbereich, also in der **„Time Domain“**) als ***Multiplikation*** des Analog-Eingangssignals mit der Folge der Sample-Pulse betrachten. Folglich erhält man am Ausgang des Samplers wieder eine Impulsfolge, deren Amplituden dem Analogsignal am Eingang folgen.

Im Frequenzbereich (= **Frequency Domain**) ergibt diese **Multiplikation** jedoch die **Bildung von Summen- und Differenzfrequenz der beiden beteiligten Signale und ihrer Frequenzanteile.**

**Bitte beachten:**

Der ***Samplepuls*** besitzt ein Spektrum, bei dem sämtliche gerad- und ungeradzahligen Oberwellen mit derselben Amplitude drinstecken. Jede enthaltene Oberwelle wird also mit dem Analog-Eingangssignal **amplitudenmoduliert**. Folglich findet man als Endergebnis bei jeder Oberwelle nun zusätzlich die **Summen- und Differenzfrequenz aus dieser Oberwelle und der analogen Eingangsfrequenz**. Es entstehen (da das analoge Eingangssignal meist mehr als nur eine einzige Frequenz enthält) folglich so genannte „**untere und obere Seitenbänder" bei der Sample-Grundfrequenz und allen ihren Oberwellen!!**




Jetzt ist gut zu erkennen, was bei zu niedriger Abtastfrequenz geschieht:

Die AM-Spektren beginnen sich zu überschneiden und das führt zu äußerst unangenehmen Verzerrungen. Dieser Vorgang heißt

## ALIASING

und die Verzerrungen heißen logischerweise „**Aliasing-Verzerrungen**". Leider klingen sie nicht nur sehr unangenehm, sondern lassen sich -- wenn dieses Unglück passiert ist -- ***auf keine Weise mehr beseitigen***!
Diesen Effekt soll der Tiefpass vor dem A-D-Wandler verhindern. Deshalb trägt er den treffenden Namen „**ANTI-ALIASING-LOWPASS-FILTER"**.

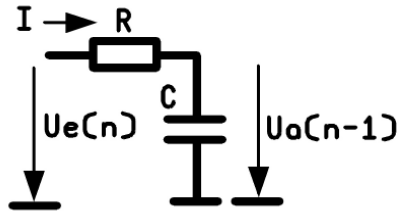
Also nochmals gut einprägen:

**Beim Eingangssignal muss ALLES oberhalb der halben Abtastfrequenz unterdrückt werden -- koste es, was es wolle! Sonst gibt es furchtbaren Ärger……**

## 3. Praxisprojekt: Ein einfacher Digitaler Filter

### 3.1. Einführung

In einem Digitalen Filter wird der vom Sampler ausgegebene Serielle Datenstrom einer mathematischen Behandlung unterzogen: es werden die verschiedenen Abtastwerte mit Faktoren multipliziert und nach einem bestimmten System zusammenaddiert. Das wirkt sich so aus, als wenn das Analogsignal über eine echte diskrete Filterschaltung gelaufen wäre. Um zu verstehen, wie dabei vorgegangen werden muss, sehen wir uns eine Schaltung an, die Jeder kennt: den simplen **analogen RC-Tiefpass**.




Zu einem bestimmten Zeitpunkt beobachtet man am Kondensator die **Ausgangsspannung $U_{a(n-1)}$**.

Jetzt ändert sich plötzlich die Eingangsspannung, weil die Sampleschaltung den nächsten Abtastwert anliefert. Damit liegt nun in diesem Moment am **Eingang** die **neue Spannung $U_{e(n)}$**, während die **Kondensatorspannung noch den alten Wert $U_{a(n-1)}$** hat.

Wir fragen jetzt nach der ***neuen*** **Ausgangsspannung $U_{a(n)}$**, die sich in der Zwischenzeit bis zum Eintreffen des nächsten Abtastwertes -- also nach dem Zeitunterschied

**$\Delta t = t_{sample}$**

am Kondensator bilden wird.

$$U_{a(n)} = U_{a(n-1)} + \Delta U_C = U_{a(n-1)} + \frac{\Delta Q}{C} = U_{a(n-1)} + \frac{I \bullet \Delta t}{C} = U_{a(n-1)} + I \bullet \left( \frac{t_{Sample}}{C} \right)$$

$$U_{a(n)} = U_{a(n-1)} + \left( \frac{U_{e(n)} - U_{a(n-1)}}{R} \right) \bullet \left( \frac{t_{Sample}}{C} \right) = U_{a(n-1)} + \left[ U_{e(n)} - U_{a(n-1)} \right] \bullet \left( \frac{t_{Sample}}{RC} \right)$$

Jetzt sollte man zuerst „$t_{sample}$“ durch die Samplefrequenz (**$f_{sample} = 1 / t_{sample}$**) und dann die Zeitkonstante RC durch die Grenzfrequenz des Tiefpasses ausdrücken. Dafür gilt in der Nachrichtentechnik:

$$R = \frac{1}{\omega_{grenz} \bullet C} \Rightarrow \omega_{grenz} = \frac{1}{RC} \Rightarrow 2\pi f_{grenz} = \frac{1}{RC} \Rightarrow \mathbf{f_{grenz} = \frac{1}{2\pi \bullet RC}}$$

Setzt man beide Erkenntnisse bei der vorigen Formel für $U_{a(n)}$ in die ganz rechte Klammer ein, dann erhält man:

$$U_{a(n)} = U_{a(n-1)} + \left[ U_{e(n)} - U_{a(n-1)} \right] \bullet \left( \frac{2\pi \bullet f_{grenz}}{f_{sample}} \right)$$

Um etwas Überblick zu schaffen, sollte man noch den ganz rechten Klammerausdruck abkürzen und ihn z. B. mit dem Buchstaben „k“ bezeichnen. Das ergibt nach dem Ausmultiplizieren:

$$U_{a(n)} = U_{a(n-1)} + k \bullet \left(U_{e(n)} - U_{a(n-1)}\right) = U_{a(n-1)} + k \bullet U_{e(n)} - k \bullet U_{a(n-1)}$$

$$\mathbf{U_{a(n)} = k \bullet U_{e(n)} + (1-k) \bullet U_{a(n-1)}}$$

Wir sehen, dass wir nun den ***neuen Eingangswert*** $\mathbf{U_{e(n)}}$ mit **„k", den *alten Ausgangswert*** $\mathbf{U_{a(n-1)}}$ **dagegen mit „(1-k)" multiplizieren** und beide Ergebnisse **addieren** müssen, um die Ausgangsreaktion auf den neuen Eingangswert zu erhalten.

Das ist eine gängige und dauernd nötige Prozedur bei der Digitalen Signalverarbeitung, sie trägt die Kurzbezeichnung „**MAC" (= multiply and accumulate).**

## 3.2. Übungsaufgabe: Untersuchung der RC-Schaltung „von Hand"

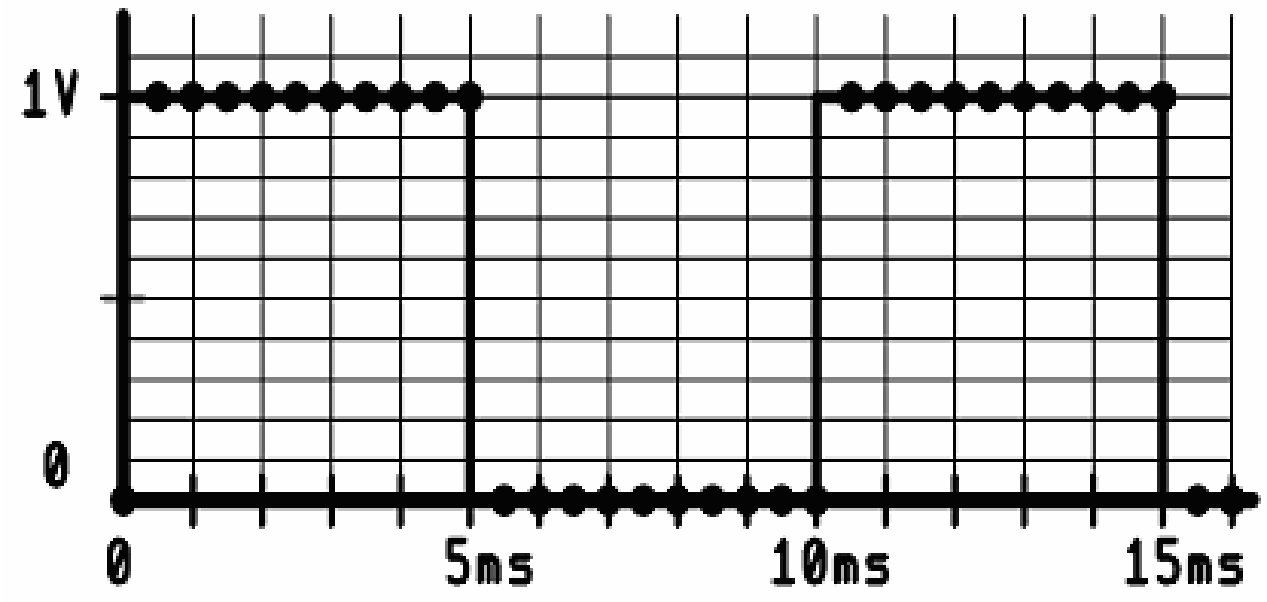
Wir wollen uns eine Tiefpass-Grenzfrequenz von 100 Hz sowie eine Samplefrequenz von 2 kHz vorgeben. Damit ist die Shannon-Bedingung (= Analoge Eingangsfrequenz kleiner als die halbe Samplefrequenz) bequem erfüllt.

**1. Schritt:**
Wir berechnen die beiden Faktoren „k" und „(1-k)":

$$k = \frac{2\pi \bullet 100Hz}{2000Hz} = 0{,}314$$

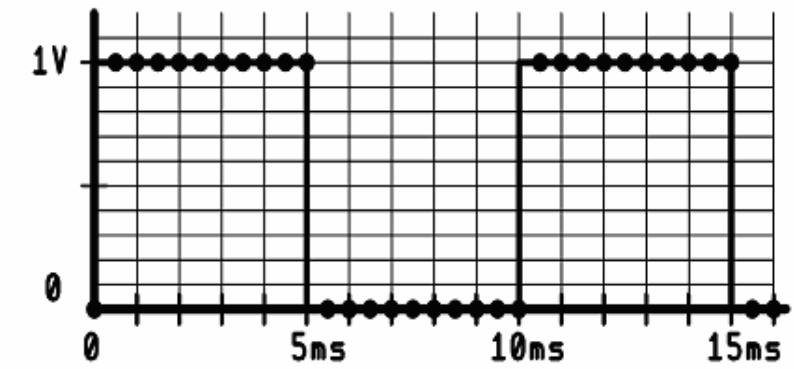
$$(1-k) = 1 - 0{,}314 = 0{,}686$$

**2. Schritt:**
Der Startwert der Ausgangsspannung bei t = Null sei Null Volt. Ab dem Zeitpunkt t = Null wird als Eingangssignal eine Rechteckspannung mit der Frequenz f = 100 Hz und den Amplitudenwerten 1V bzw. 0V zugeführt. Wir skizzieren den zeitlichen Verlauf des Eingangssignals für t = Null bis t = 15 Millisekunden und markieren darin die Abtastungs-Zeitpunkte:

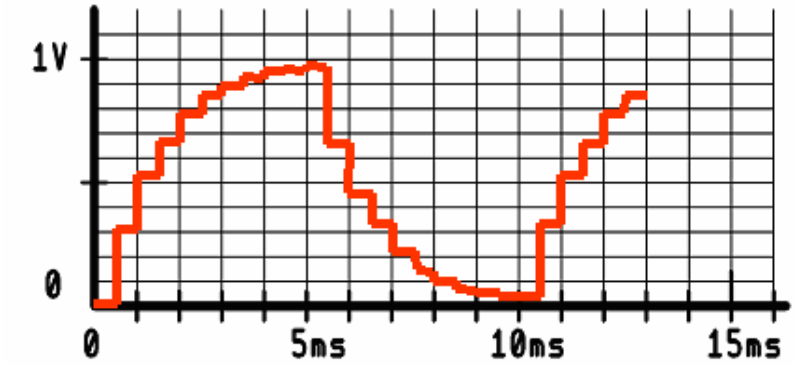


**3. Schritt**:
Jetzt legen wir eine **Tabelle** für alle beteiligten Größen an und berechnen (geht am schnellsten mit ***Excel***, aber natürlich auch mit dem Taschenrechner) bei jedem neuen Eingangssample das erzeugte Ausgangssignal am Kondensator.

| Sample-Nr.<br><br>**n** | Neue Eingangs-Spannung<br><br>**Ue(n)** | k * Ue(n)<br>=<br>**0,314 * Ue(n)** | Alte Ausgangs-Spannung<br><br>**Ua(n-1)** | (1 – k)*Ua(n-1)<br>=<br>**0,686*Ua(n-1)** | Neue Ausgangs-Spannung:<br>Ua(n)<br>=<br>**0.314*Ue(n)<br>+0,686*Ua(n-1)** |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0,314 | 0 | 0 | 0,314 |
| 2 | 1 | 0,314 | 0,314 | 0,215 | 0,529 |
| 3 | 1 | 0,314 | 0,529 | 0.363 | 0.677 |
| 4 | 1 | 0,314 | 0,677 | 0,464 | 0,778 |
| 5 | 1 | 0,314 | 0,778 | 0,533 | 0,848 |
| 6 | 1 | 0,314 | 0,848 | 0,581 | 0,895 |
| 7 | 1 | 0,314 | 0,895 | 0,614 | 0,928 |
| 8 | 1 | 0,314 | 0,928 | 0,637 | 0,95 |
| 9 | 1 | 0,314 | 0,95 | 0,652 | 0,966 |
| 10 | 1 | 0,314 | 0,966 | 0,662 | 0,977 |
| 11 | 0 | 0 | 0,977 | 0,670 | 0,670 |
| 12 | 0 | 0 | 0,67 | 0,459 | 0,459 |
| 13 | 0 | 0 | 0,459 | 0,315 | 0,315 |
| 14 | 0 | 0 | 0,315 | 0,216 | 0,216 |
| 15 | 0 | 0 | 0,216 | 0,148 | 0,148 |
| 16 | 0 | 0 | 0,148 | 0,1 | 0,1 |
| 17 | 0 | 0 | 0,1 | 0,069 | 0,069 |
| 18 | 0 | 0 | 0,069 | 0,0479 | 0,0479 |
| 19 | 0 | 0 | 0,0479 | 0,0328 | 0,0328 |
| 20 | 0 | 0 | 0,0328 | 0,0225 | 0,0225 |
| 21 | 1 | 0,314 | 0,0225 | 0,015 | 0,329 |
| 22 | 1 | 0,314 | 0,329 | 0,226 | 0,54 |




Das ist das analoge Eingangssignal mit den markierten Abtast-Zeitpunkten….




…und dieses Ausgangssignal erhält man nach der Digitalen Signalverarbeitung im Controller und der D-A-Wandlung laut obiger Tabelle.

Wer den analogen Tiefpass kennt, der kennt auch dieses Ausgangssignal. Es muss nur noch gefiltert und damit von den Resten der Samplefrequenz befreit werden!

## 3. FIR – Filter = Nichtrekursive Filter

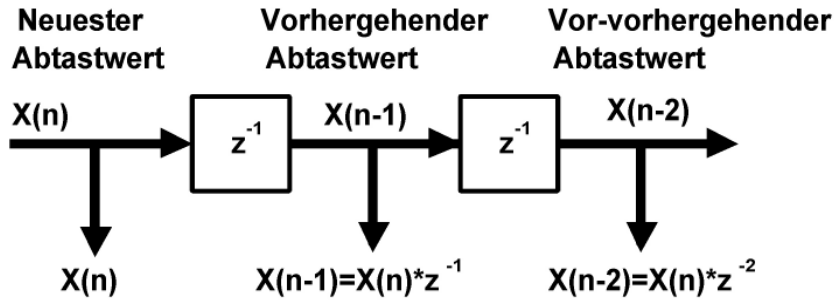
In der Praxis benötigt man wesentlich aufwendigere Filter als den eben behandelten einfachen Tiefpass. Ihr Entwurf erfordert Kenntnisse der höheren Mathematik (= Komplexe Rechnung und Integralrechnung), da hier die „Z-Transformation“ verwendet wird.
Deren Sinn und Aufgabe wird aber erst klar, wenn man (in etwas vereinfachter Form) folgende Tatsache kennt:

**Jedes Mal, wenn im Blockschaltbild ein Klötzchen mit der Bezeichnung**

$$Z^{-1}$$

**auftaucht, muss an seinem Ausgang der (bereits vergangene, aber hoffentlich irgendwo gespeicherte) vorige Samplewert benützt werden!**

So könnte dann beim Filterentwurf ein Teil des Schaltbildes aussehen:

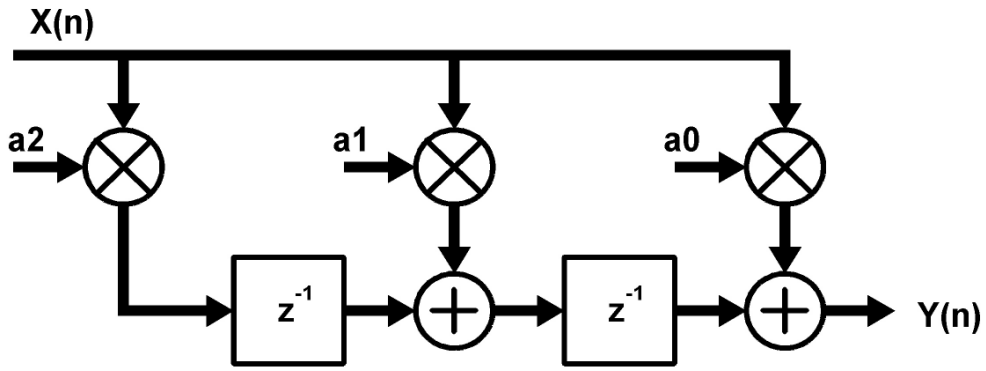



Mit diesem Wissen können wir schon den prinzipiellen Aufbau der „FIR“-Filter verstehen.

Erläuterung:
„**FIR**“ bedeutet „**finite impulse response**“ (also: „**endliche Impulsantwort**“).Solche Filter antworten auf ein Eingangssignal in Form eines Nadelimpulses immer mit einem Ausgangssignal, **das wieder bis auf Null abklingt. Diese Filter können also nie schwingen und sind grundsätzlich stabil.**

Man erkennt sie stets an der „Einbahnstraßenstruktur vom Eingang in Richtung Ausgang“ -- es gibt ***NIEMALS*** eine Rückführung des Ausgangssignals in Richtung Eingang. Eine weitere Bezeichnung dieser Filterart lautet: „**Nicht-Rekursive Filter**“.

Beispiel:




Für das Ausgangssignal gilt:

$$Y_{(n)} = a_0 \bullet X_{(n)} + a_1 \bullet X_{(n)} \bullet z^{-1} + a_2 \bullet X_{(n)} \bullet z^{-2}$$

und das bedeutet:

$$Y_{(n)} = a_0 \bullet X_{(n)} + a_1 \bullet X_{(n-1)} + a_2 \bullet X_{(n-2)}$$

Klammert man aus der rechten Seite der Gleichung den Faktor „X(n)“ aus, so lässt sich anschließend sehr leicht die **Übertragungsfunktion (= Transfer Function)** darstellen:

$$Y_{(n)} = X_{(n)} \bullet \left[a_0 + a_1 \bullet z^{-1} + a_2 \bullet z^{-2}\right]$$

$$\Rightarrow \frac{Y_{(n)}}{X_{(n)}} = H(z) = a_0 + a_1 \bullet z^{-1} + a_2 \bullet z^{-2}$$

(Ein kleiner Hinweis: ein Filter ist dann absolut stabil, wenn in dieser Übertragungsfunktion der Ausdruck „$z^{-1}$“ **nur im Zähler vorkommt**).

**Vorteile der FIR-Filter:**

a) Absolute Stabilität, also keinerlei Schwingneigung

b) Wird der Filter symmetrisch aufgebaut (also wenn a0 = a2), dann weist er einen linearen Phasengang und damit eine „konstante Gruppenlaufzeit“ auf.

c) Nicht so empfindlich gegen „Streuungen“ oder Lässigkeiten beim Entwurf, also z. B. Auf- oder Abrundungen oder Vereinfachungen im Programm

**Nachteile der FIR-Filter**: Nicht so steile Filterflanken bei gleichem Filtergrad wie IIR-Filter.

---

## Einige Hinweise zum Entwurf solcher FIR-Filter.

Neben umfangreichen Anleitungen (..die aber gute Mathematikkenntnisse voraussetzen) gibt es immer mehr Programme im Internet, die einem diese Arbeit abnehmen. Trotzdem sollte man einige grundsätzliche Dinge wissen:

1) Die Summe der Koeffizienten (a0 / a1 / a2…) ergibt die **Gleichspannungsverstärkung** des Filters.

2) Beim Entwurf darf **nicht die gewünschte Grenzfrequenz** verwendet werden, sondern nur die „**normierte Grenzfrequenz Fg**“. Darunter versteht man

**das Verhältnis der gewünschten echten Grenzfrequenz zur Samplefrequenz**

**und das muss immer deutlich unter 0,5 liegen (…da war doch was mit der „Shannon-Bedingung“…).**

3) Der **Filtergrad sollte immer ungerade sein**, denn dann hat man automatisch bei der **halben Samplefrequenz eine Nullstelle im Frequenzgang** (und das unterstützt das Einhalten der Shannonbedingung).

Beispiel:

Gegeben sei ein Filter für Fg = 0,25 und Filtergrad N = 5 mit den Filterkoeffizienten

**a0 = a5 = -0,00979 a1 = a4 = +0,00979 a2 = a3 = +0,5**

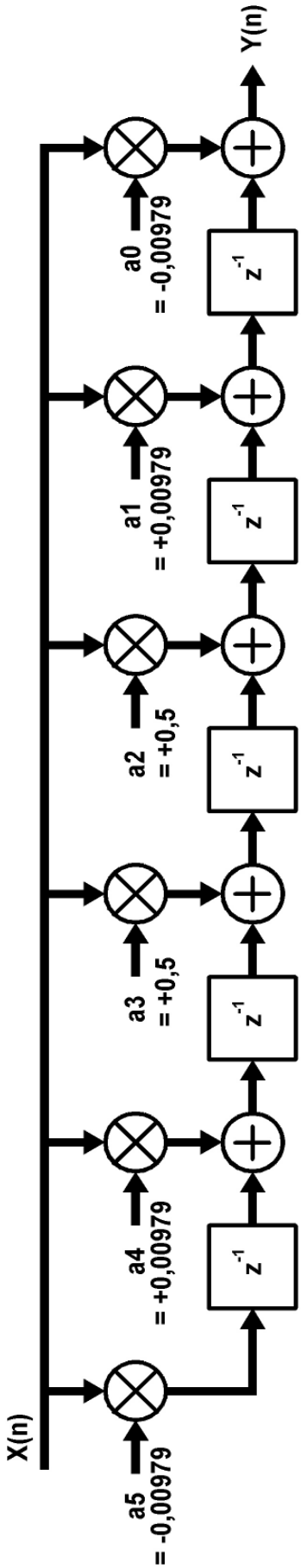
Berechnen Sie die „echte Grenzfrequenz, wenn mit 8 kHz abgetastet wird und skizzieren Sie die Schaltung.

Lösung:

$$f_{grenz} = F_g \bullet f_{Sample} = 0{,}25 \bullet 8\text{kHz} = 2\text{kHz}$$

Schaltung: Siehe nächstes Blatt!

Lösung:


X(n)
Y(n)
a5
= -0,00979
a4
= +0,00979
a3
= +0,5
a2
= +0,5
a1
= +0,00979
a0
= -0,00979
z-1
z-1
z-1
z-1
z-1

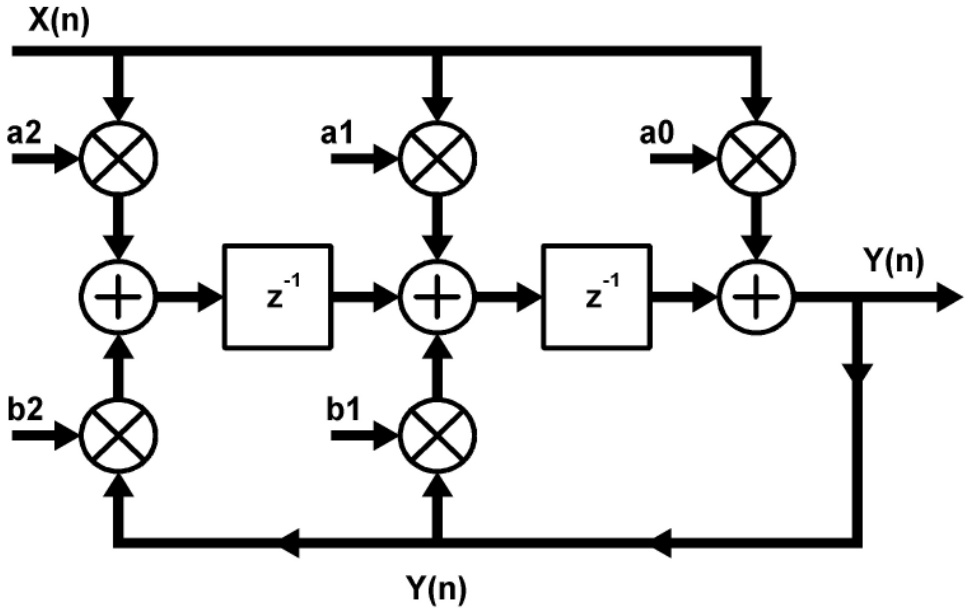
## 4. IIR-Filter = Rekursive Filter

Sie besitzen immer eine „Rückführung **vom Ausgang zum Eingang**“ und brauchen deshalb wesentlich mehr Sorgfalt beim Entwurf -- sie neigen bei der kleinsten Schlamperei zum Schwingen (..das kennt man in der Analogtechnik von rückgekoppelten Verstärkern…). Dazu reicht schon ein zu frühes Aufrunden oder Abrunden oder Abbrechen bei langen Zahlen!.

**Dafür erhält man bei gleichem Filtergrad wesentlich steilere Flanken und damit bessere Filterwirkungen gegenüber den vorhin besprochenen FIR-Filtern.**

Übrigens: „IIR“ heißt „Infinite Impulse Response“ = „unendlich lange Impulsantwort“. Die Antwort auf einen Nadelimpuls am Eingang klingt also niemals ganz bis auf Null ab, sondern wird (hoffentlich“) nur immer kleiner. Außerdem liefern diese komischen Vögel bereits eine Antwort am Ausgang, bevor noch der Eingangsimpuls angelegt wird…..also eine ganz verrückte Rasse, die man vorsichtig behandeln muss.

Beispiel für eine solche Filteranordnung:




$$Y_{(n)} = a_0 \bullet X_{(n)} + a_1 \bullet X_{(n)} \bullet z^{-1} + a_2 \bullet X_{(n)} \bullet z^{-2} + b_1 \bullet Y_{(n)} \bullet z^{-1} + b_2 \bullet Y_{(n)} \bullet z^{-2}$$

$$\Rightarrow Y_{(n)} \bullet \left(1 - b_1 \bullet z^{-1} - b_2 \bullet z^{-2}\right) = X_{(n)} \bullet \left(a_0 + a_1 \bullet z^{-1} + a_2 \bullet z^{-2}\right)$$

Daraus folgt die Übertragungsfunktion:

$$\frac{Y_{(n)}}{X_{(n)}} = \frac{\left(a_0 + a_1 \bullet z^{-1} + a_2 \bullet z^{-2}\right)}{\left(1 - b_1 \bullet z^{-1} - b_2 \bullet z^{-2}\right)}$$

Wie man sieht, kann der Nenner der Transfer Function möglicherweise Null werden -- speziell dann, wenn die Koeffizienten b1 und b2 positive Werte aufweisen. **Dann schwingt die Anordnung!**
Außerdem müssen wir zur Abwechslung nun auch die vorausgegangenen ***Ausgangswerte*** für die Berechnung speichern und heranziehen.

Man ahnt schon: …bei sehr aufwendigen Strukturen muss man ein ganzes Sortiment an Eingangs- UND Ausgangswerten dauernd zwischenspeichern…

Einen solchen Filter haben wir in einfacher Form schon kennen gelernt: es war die Realisierung des RC-Tiefpasses in Kapitel 3.2. Erinnern Sie sich noch? Dort galt:

$$U_{a(n)} = k \bullet U_{e(n)} + (1-k) \bullet U_{a(n-1)}$$

und die echte Grenzfrequenz war 100 Hz bei einer Samplefrequenz von 2 kHz.

Dafür erhielten wir die Koeffizienten

**k = 0,314** und **(1-k) = 0,686**

**Aufgabe:**

a) Leiten Sie die Übertragungsfunktion dieses Filters mit diesen beiden Koeffizienten ab.

b) Zeichnen Sie die Schaltung dieses Filters und tragen Sie die korrekten Werte ein.

Lösung a):

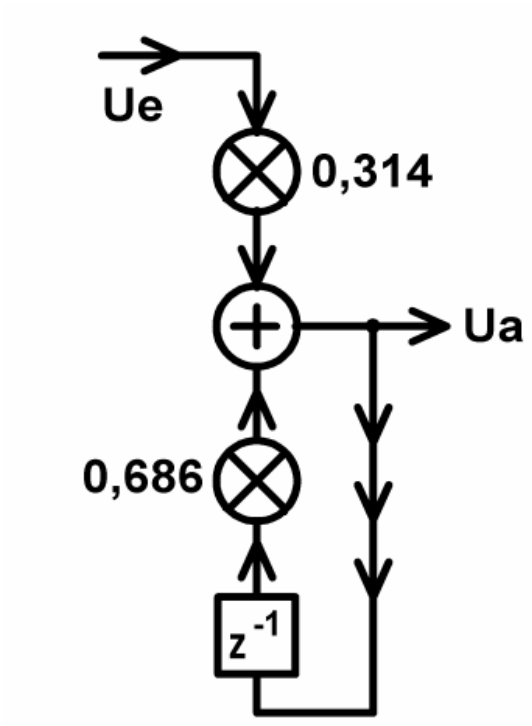
$$U_{a(n)} = k \bullet U_{e(n)} + (1-k) \bullet U_{a(n-1)}$$

$$U_{a(n)} = k \bullet U_{e(n)} + (1-k) \bullet U_{a(n)} \bullet (z^{-1})$$

$$U_{a(n)} - (1-k) \bullet U_{a(n)} \bullet (z^{-1}) = k \bullet U_{e(n)}$$

$$U_{a(n)} \bullet [1 - (1-k) \bullet z^{-1}] = k \bullet U_{e(n)}$$

$$\frac{U_{a(n)}}{U_{e(n)}} = H(z) = \frac{k}{1-(1-k) \bullet z^{-1}}$$ damit gilt hier: $$H(z) = \frac{0{,}314}{1-0{,}686 z^{-1}}$$




So sieht das Blockschaltbild des „RC-Tiefpasses in Digitaler Form“ aus!

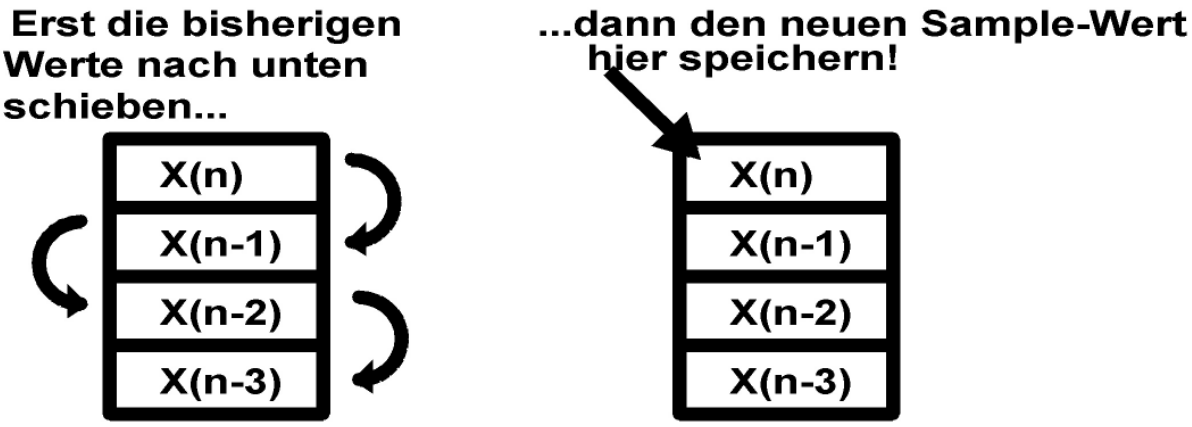
## 5. Organisation der Abtastwerte im Speicher

Es wird immer eine **konstante Anzahl von „Samples**“ im DSP verarbeitet (fast immer wird eine Zweierpotenz gewählt, also z. B. 512 oder 1024 oder 2048 Werte). Sobald aber der nächste Abtastwert aus der Sample-Hold-Schaltung kommt, gibt es Gedränge:

**Man muss nämlich nicht nur den aktuellen, neu einlaufenden Eingangswert „$X_n$“ abspeichern, sondern auch (da ja die Gesamtzahl der „Sample-Sammlung“ gleich bleibt!) eine *Aktualisierung* vornehmen! (= der älteste Wert wird verworfen, aber der neueste Wert wird an die Spitze der Liste gesetzt. Zusätzlich müssen dann alle übrigen Werte in der Liste „umnummeriert“ werden).**

a)
Bei einer kleinen Anzahl an gespeicherten Informationen geht es am schnellsten, wenn man nach der Berechnung des Ausgangssignals gleich alle Werte umspeichert und an der Spitze Platz für den nächsten Wert schafft.

Beispiel für die Verarbeitung von 4 Samples:




b)
Bei einer größeren Anzahl von Samples dauert das viel zu lange, deshalb wird folgender Weg gewählt:

Man arbeitet immer mit einer Sample-Anzahl, die einer ***Zweierpotenz*** entspricht ( = 2, 4, 8, 16, 32, 64, 128….1024, 2048, 4096…).
Dann organisiert man den Samplespeicher als ***Array in Ringform*** und speichert die Adresse des „Anfangs der Kette“ als „***POINTER***“ in einer Variablen. Dieser Pointer zeigt damit immer auf den neuesten Samplewert.

Läuft nun der nächste Samplewert ein, dann wird er auf den

**letzten Platz des Ringes gespeichert!**

Damit wird der älteste Wert überschrieben. Folglich muss man nun den Pointer auf diesen neuesten Samplewert (= neue Startadresse des Ring-Arrays) umstellen und kann dann mit der nächsten Berechnung anfangen.