FH CAMPUS WIEN
UNIVERSITY OF APPLIED SCIENCES

DIE HOCHSCHULE,
DIE MEHR KANN.

Electronic and Microcontroller

FH CAMPUS WIEN
UNIVERSITY OF APPLIED SCIENCES

DIE HOCHSCHULE,
DIE MEHR KANN.

FH CAMPUS WIEN
UNIVERSITY OF APPLIED SCIENCES

Analog Voltage

>Why Analog Voltage?
>Analog to Digital conversion
>STM32 – ADC

https://en.wikipedia.org/wiki/Universal_asynchronous_receiver-transmitter
https://upload.wikimedia.org/wikipedia/commons/1/1f/Serial_Programming.pdf

Analog Voltage

> Analog output is typical of most transducers and sensors.

> Some characteristics of analog signals.

physikalische Größen -> elektrische

Temperatur -> ohmscher Widerstand

Beschleunigung

Druck -> kapazität

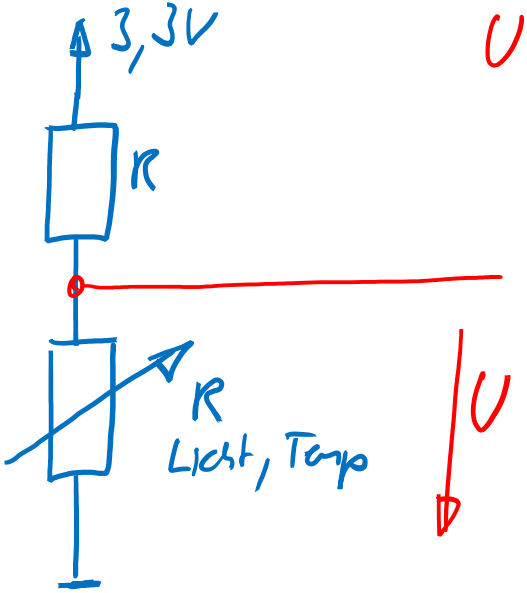
Licht

MEMS

Analog Voltage

>Analog output is typical of most transducers and sensors.

>Some characteristics of analog signals.



U entspricht der phys. Größe

$R_{20°C} = 10\,k\Omega$

$R_{durch} = 1\,k\Omega$

PTC
NTC

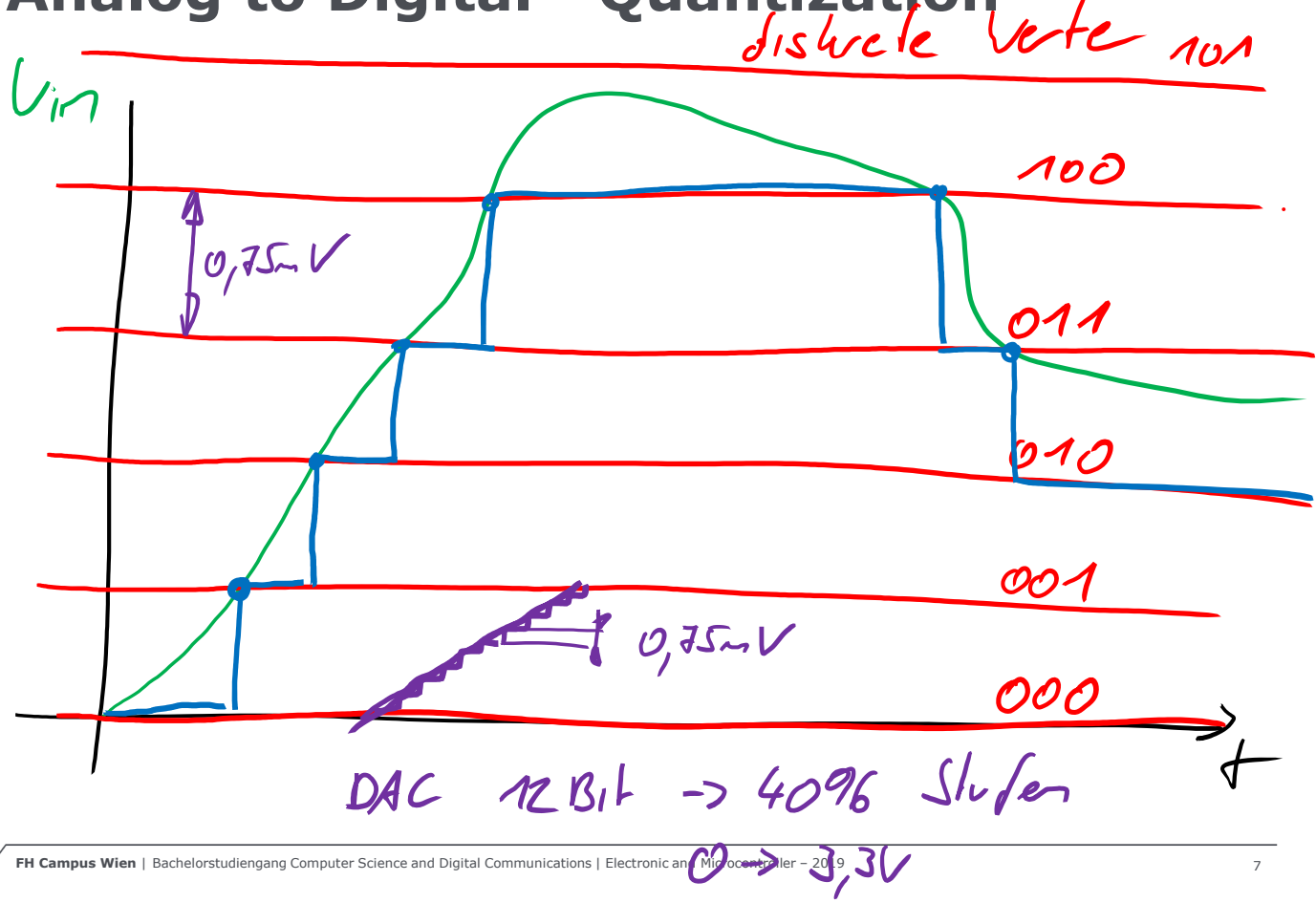
Analog Voltage

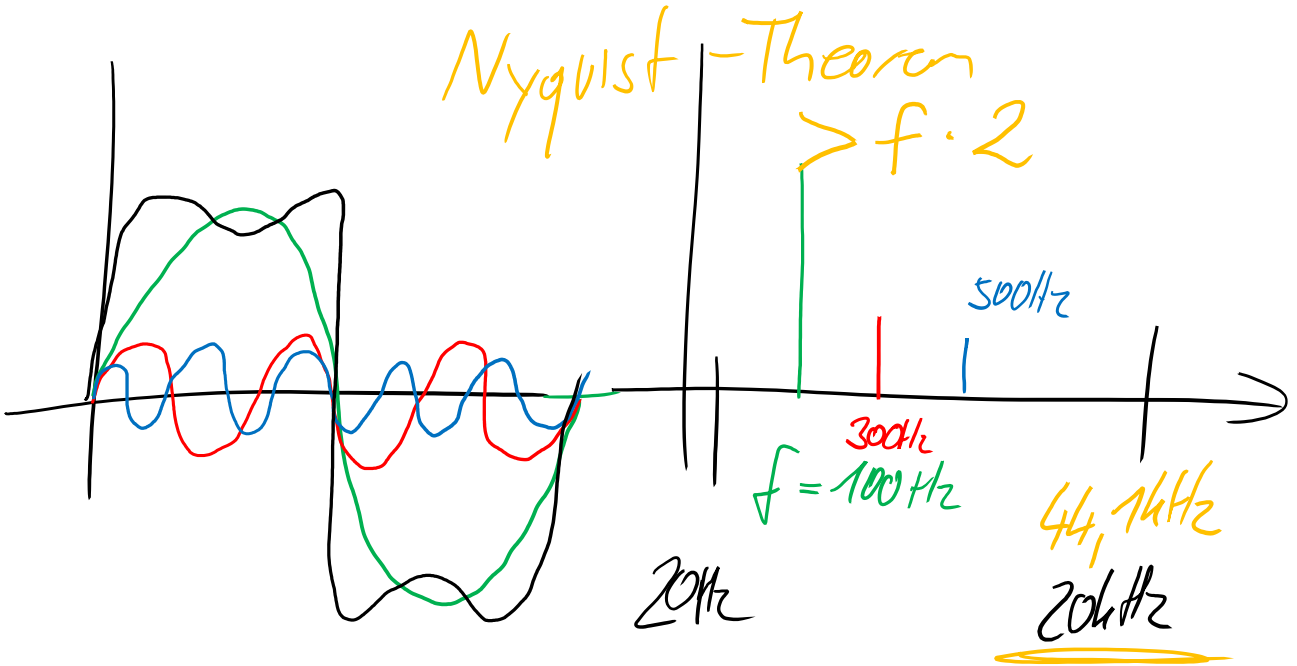
- Analog output is typical of most transducers and sensors.
- Need to convert these analog signals into a digital representation so the microcontroller can use it.

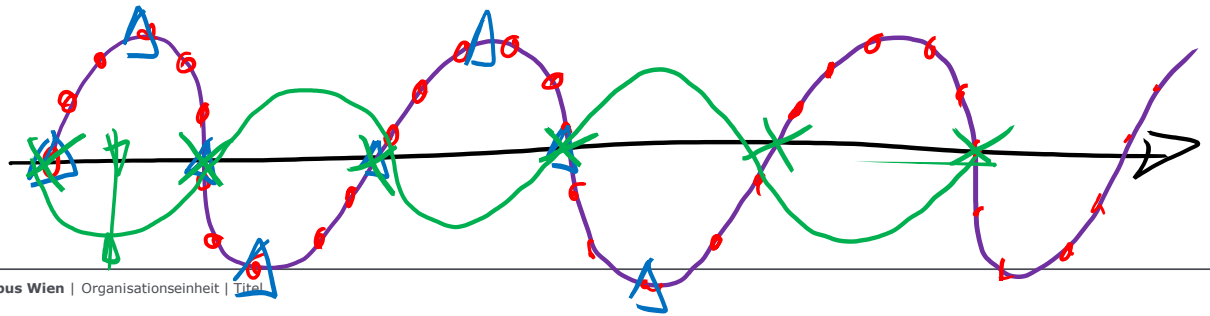
- Some characteristics of analog signals.
  - Maximum and minimum voltages
  - Precise continuous signals
  - Rate of voltage change
  - Frequency if not a steady state signal

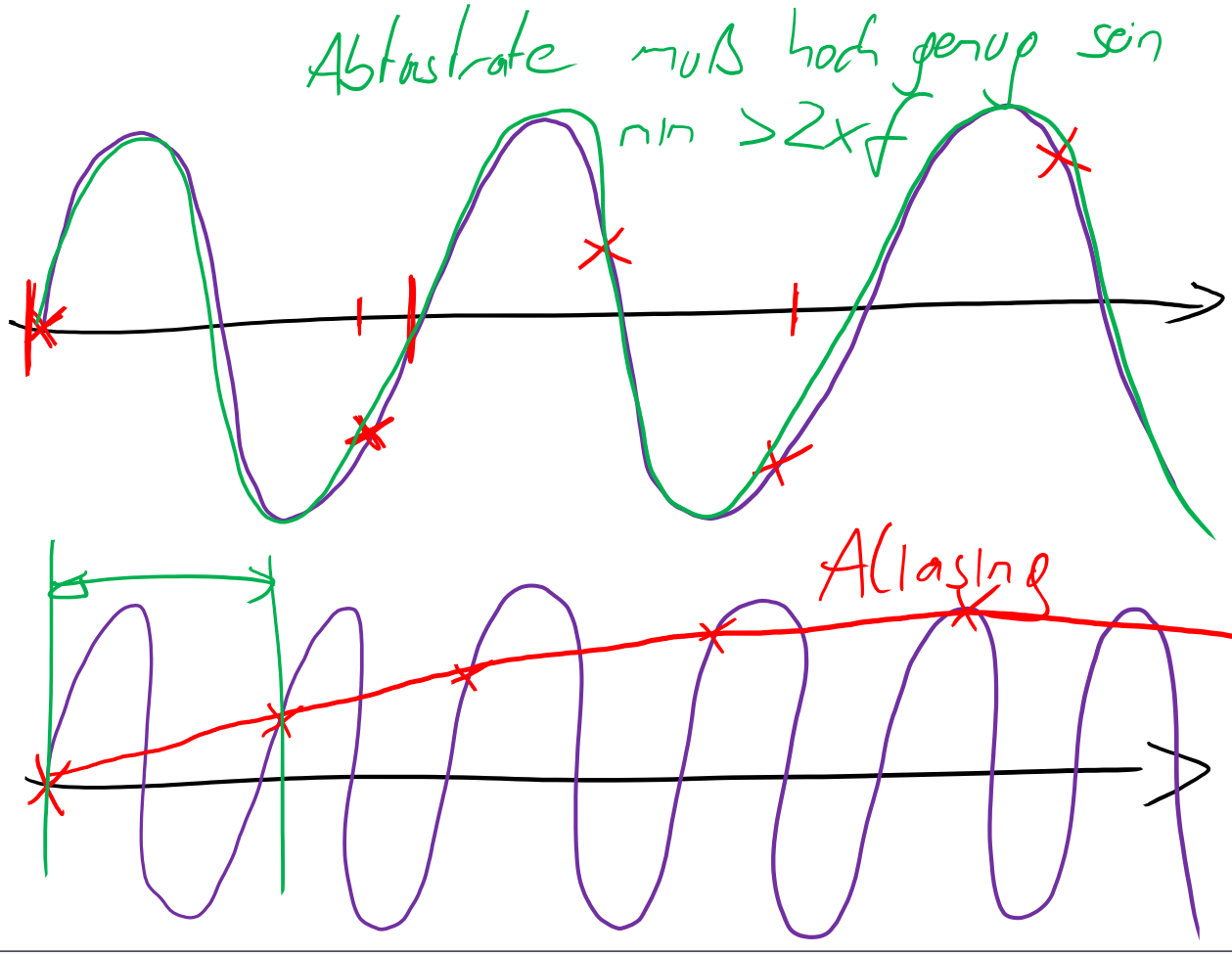
Absolut Wert

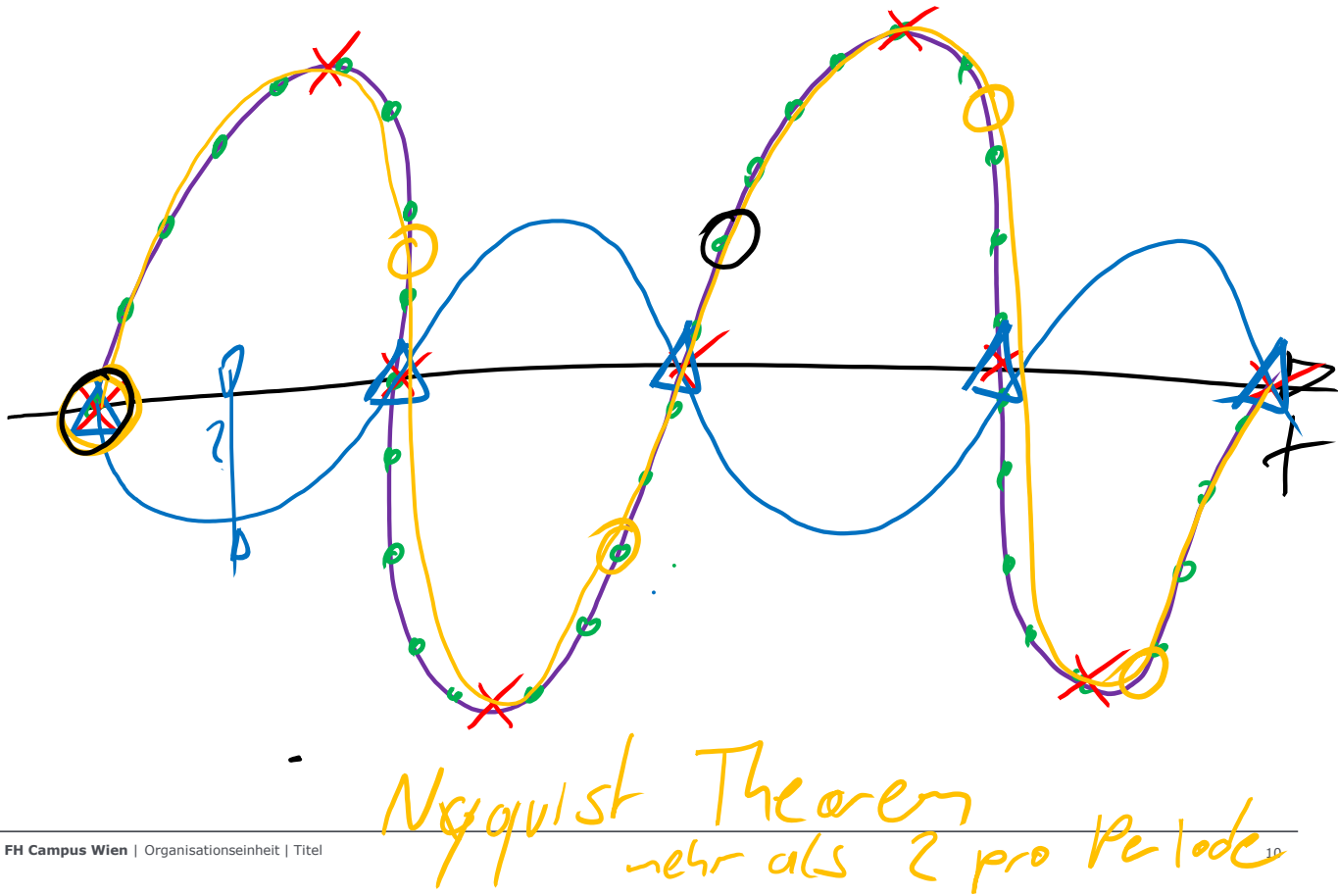
Analog to Digital - Quantization

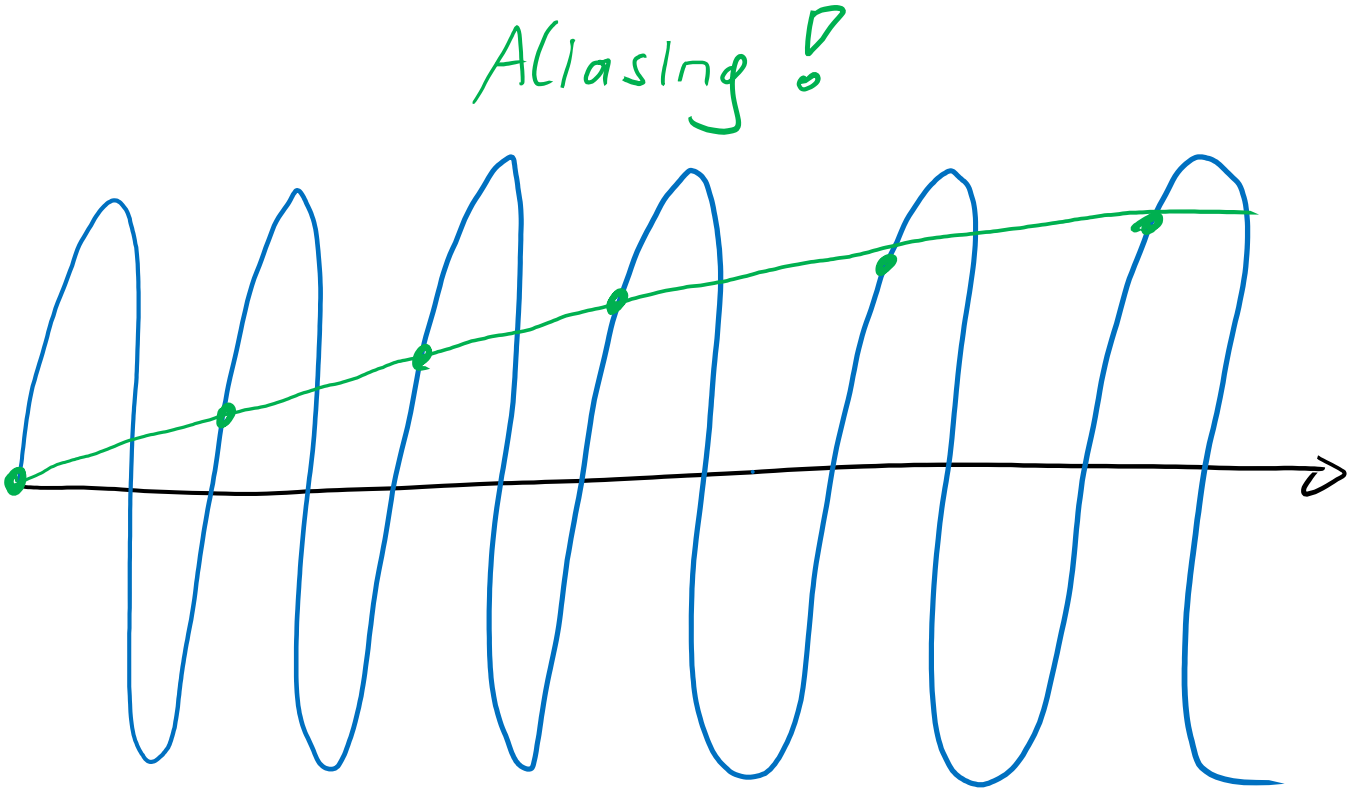



Nyquist-Theorem
> f · 2
500Hz
300Hz
f = 100 Hz
20Hz
44,1kHz
20kHz

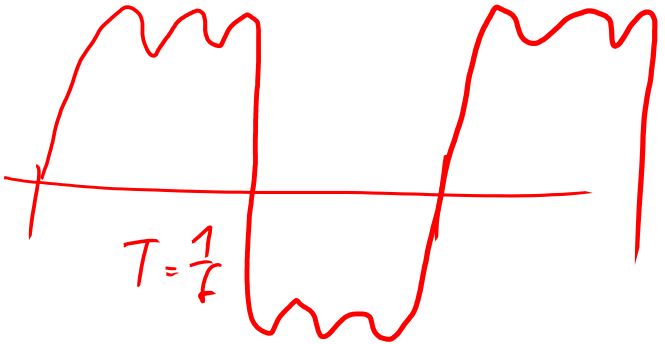


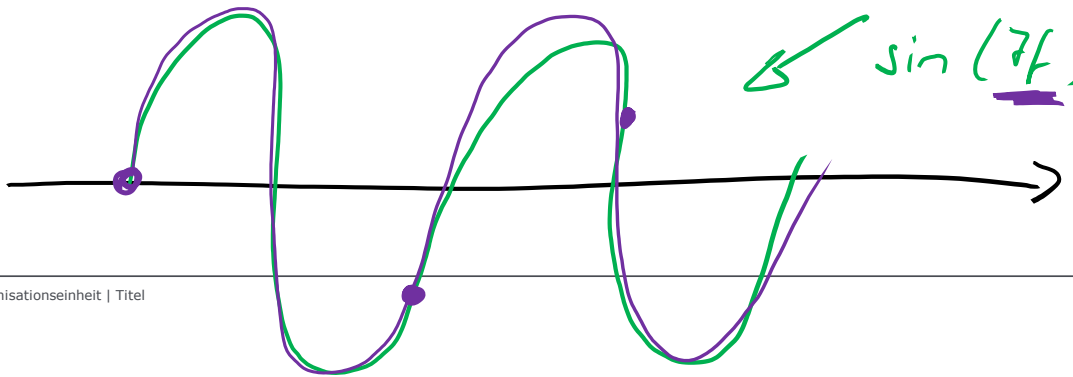

Abtastrate muß hoch genug sein
min >2xf
Aliasing


20 Messpunkte / T
Nyquist Theorem
mehr als 2 pro Periode

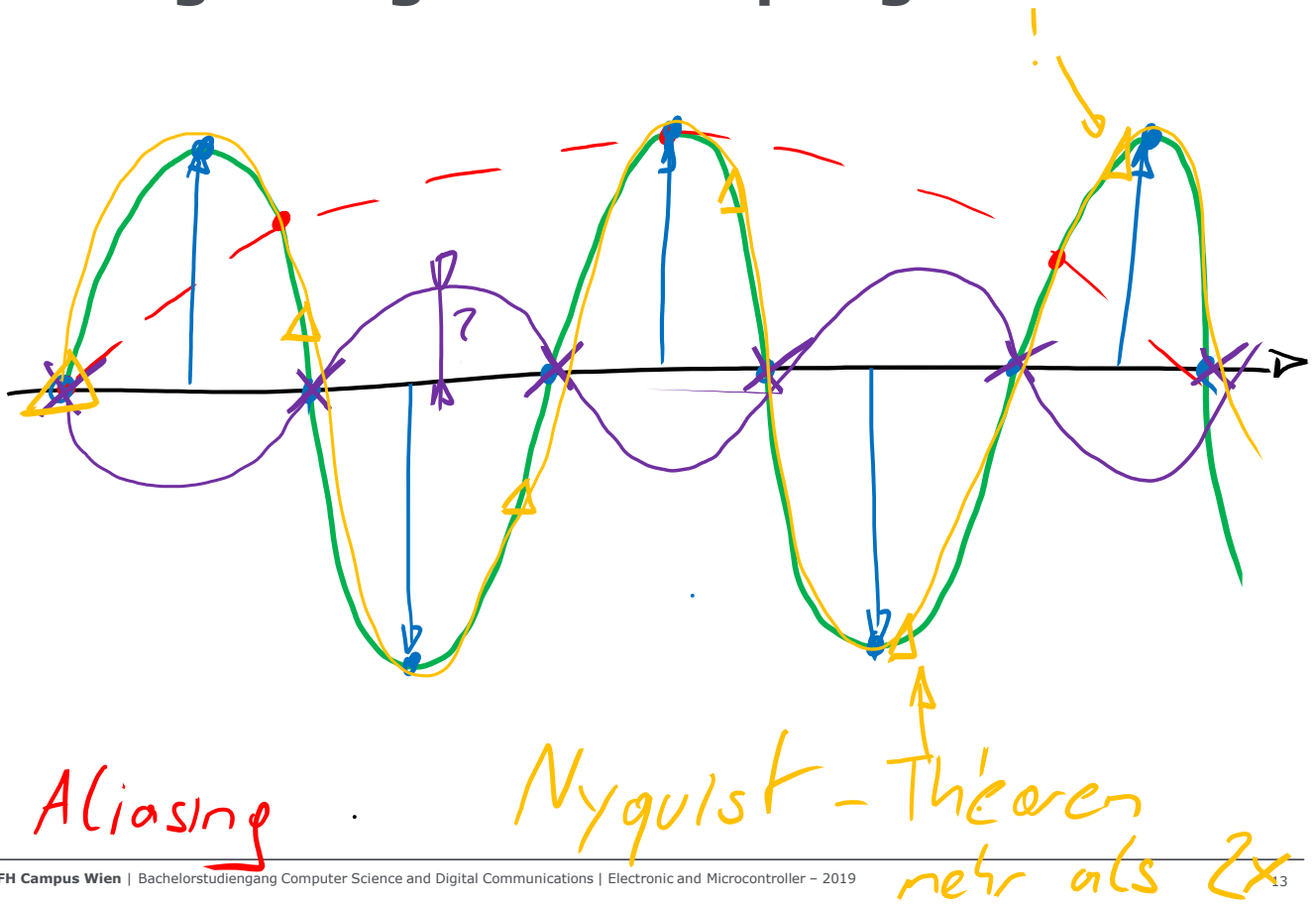

Aliasing !

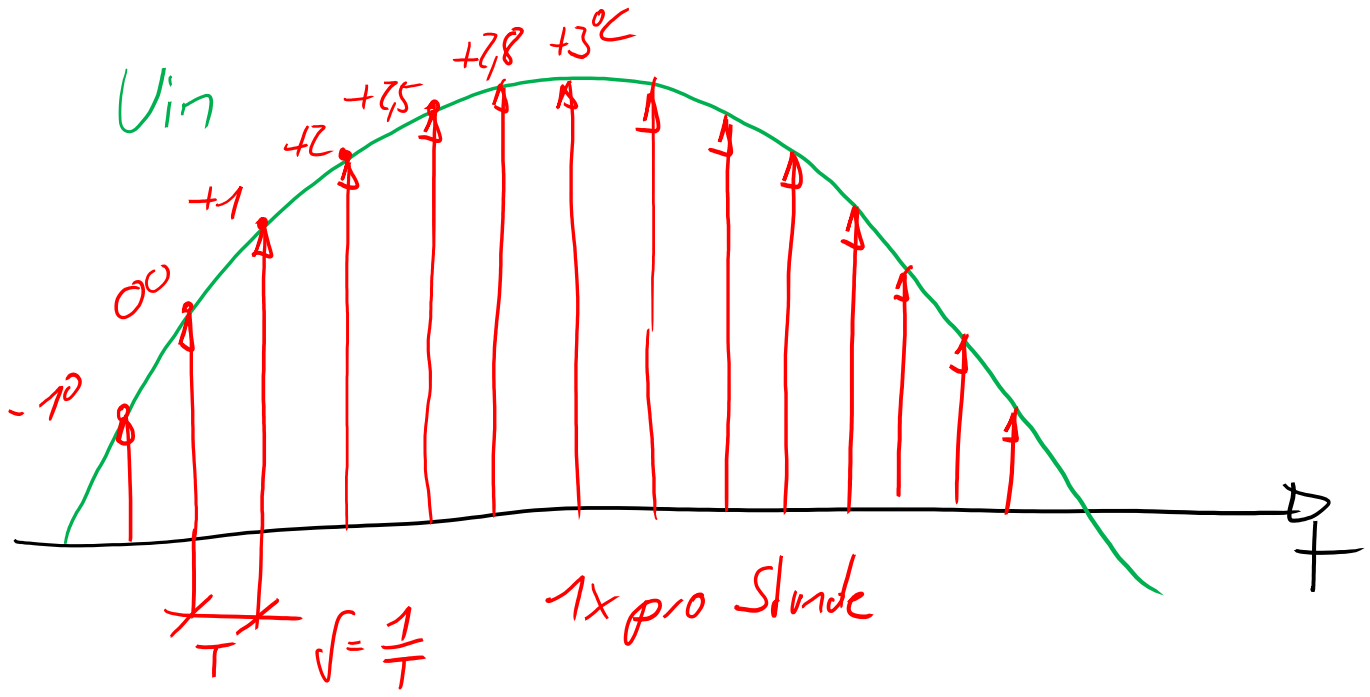
Höchste im Signal vorkommende Frequenz!





Analog to Digital – Sampling rate

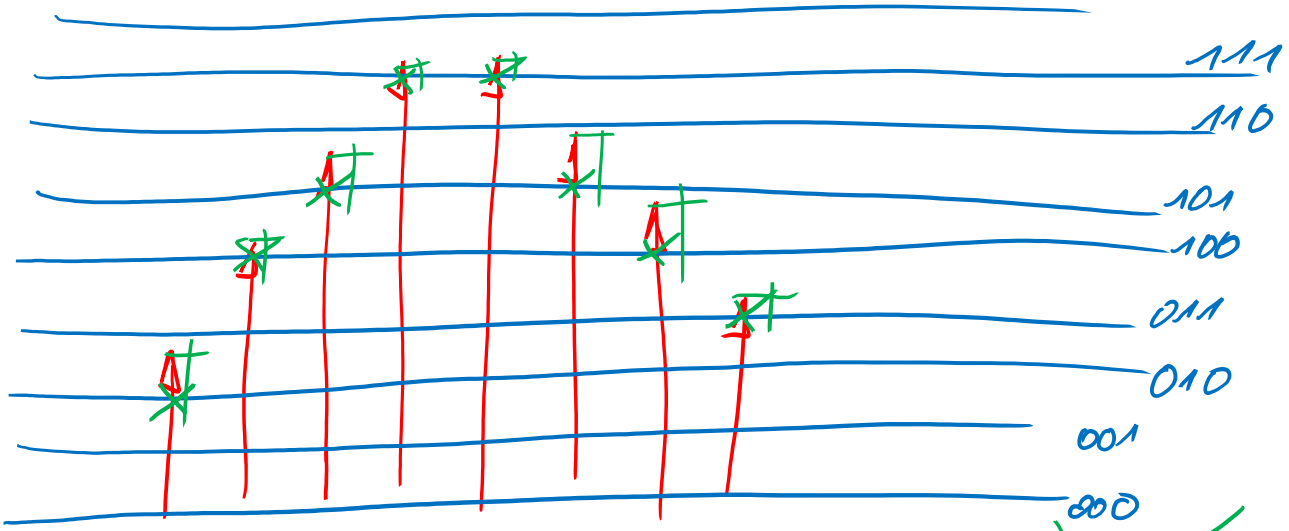




analog = wert, zeit kontinuierlich

digital = wert, zeit diskret

Δt


111
110
101
100
011
010
001
000
Auflösung
40% Stufen
12 Bit
abgetastetes Signal
jetzt folgt die Quantisierung
→ wird diskret

zum Zeitpunkt der Abtastung als Zahl verwendet wird, also

$$u[n] = u(t = nT_s) \tag{8}$$

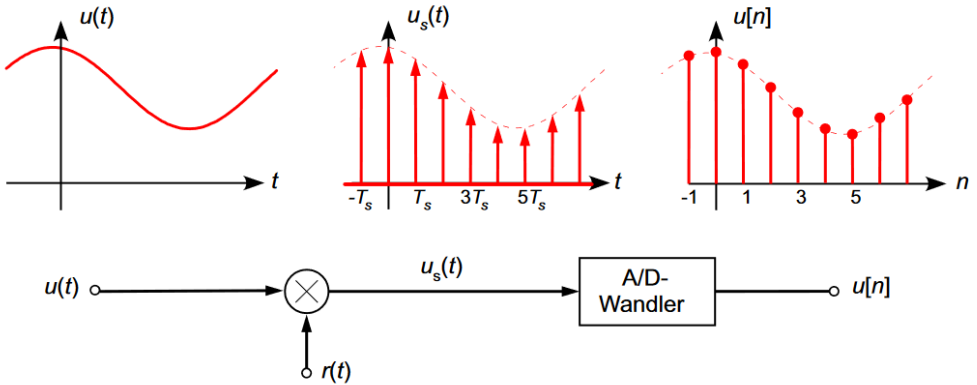
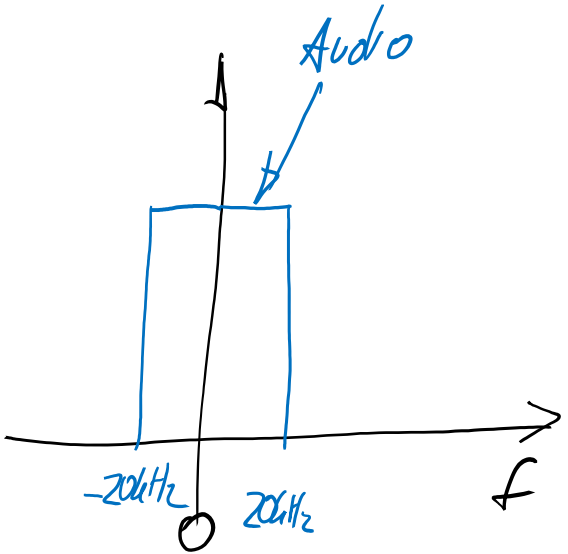
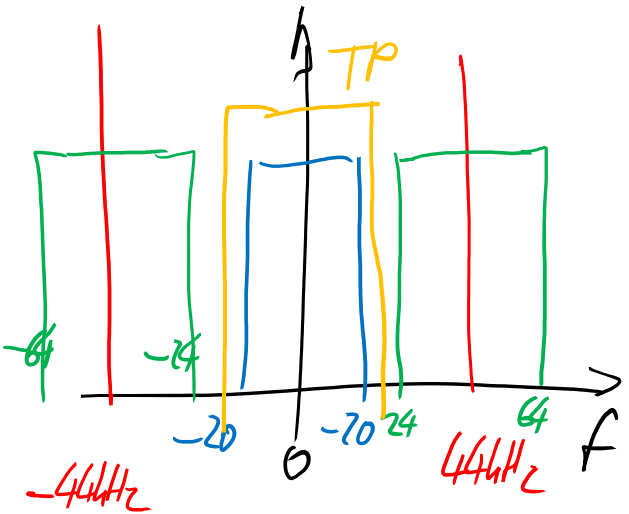


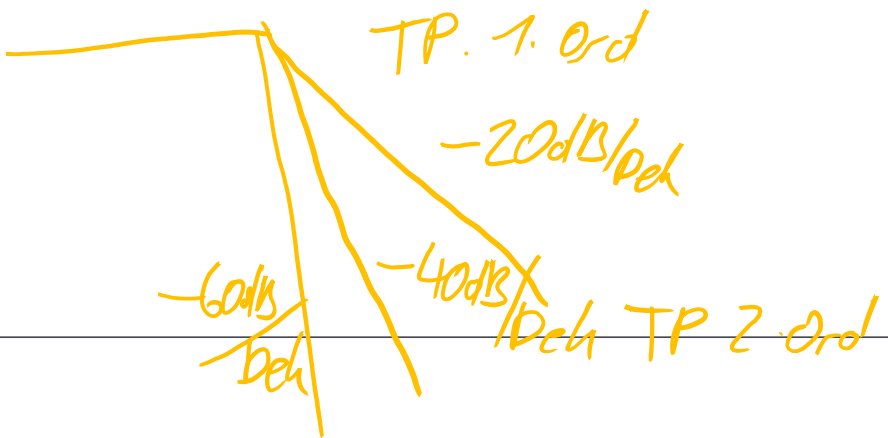
Fig. 8 Das vollständige Schema des idealen Abtasters und A/D-Wandlers

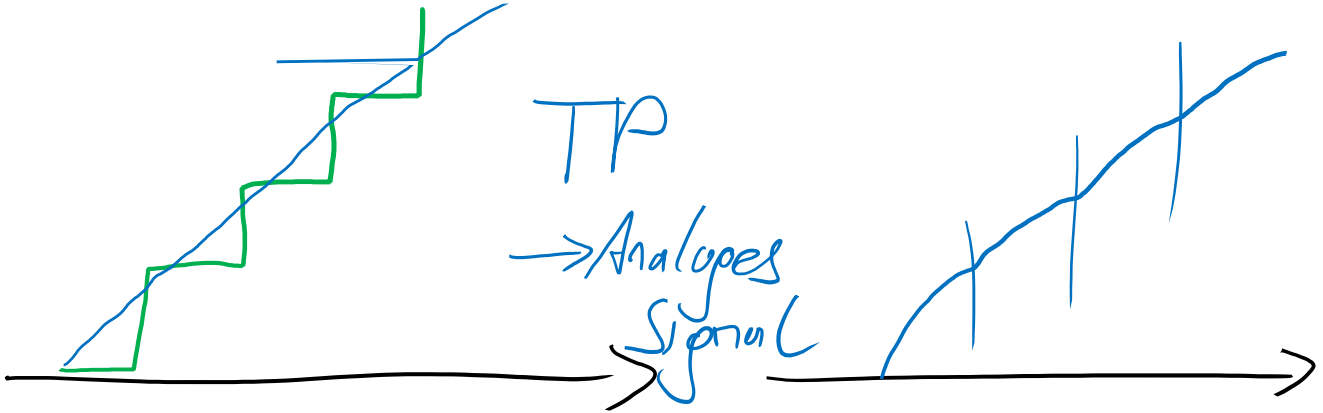




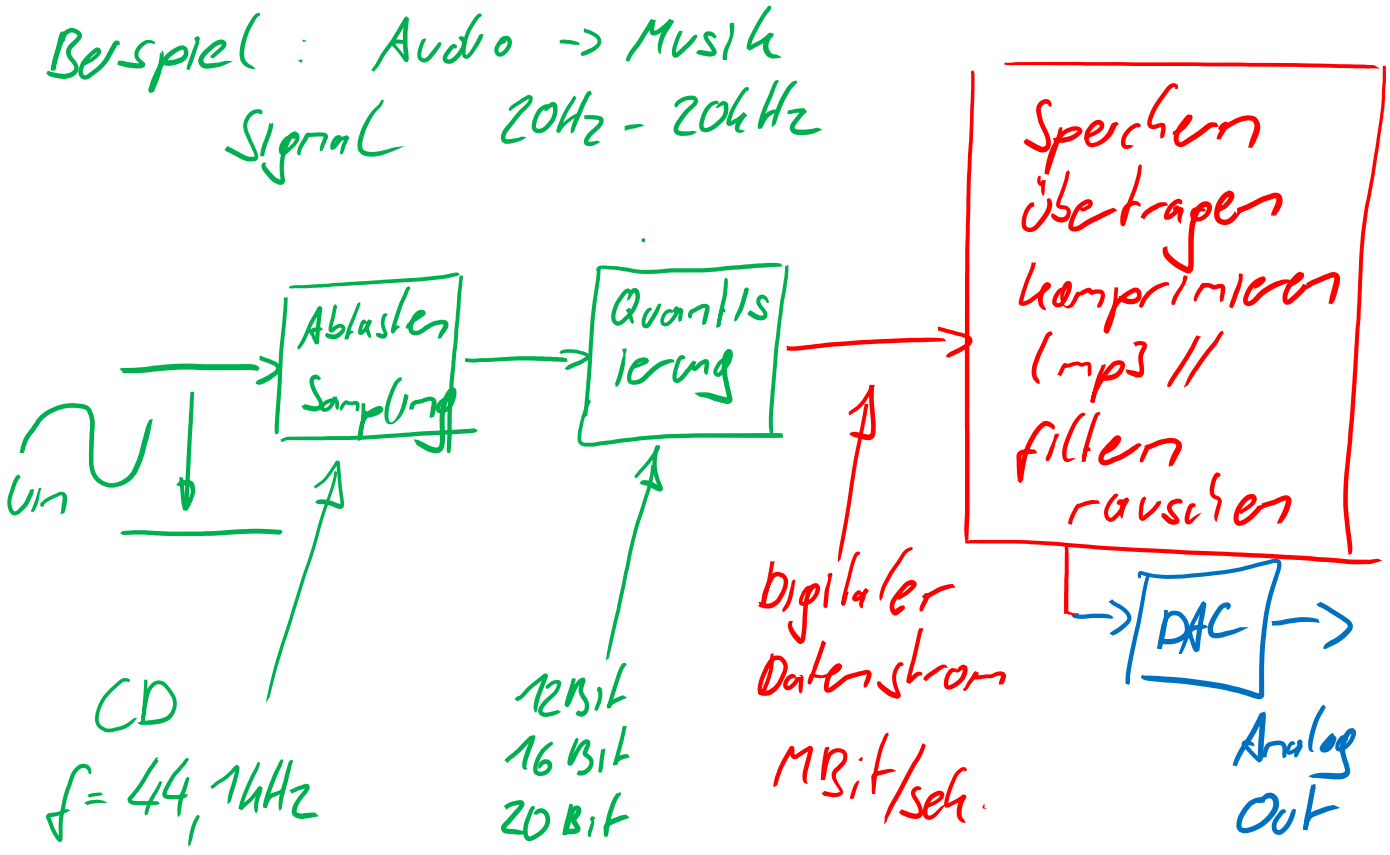
$f = 10f$

$A = \frac{A}{1000}$

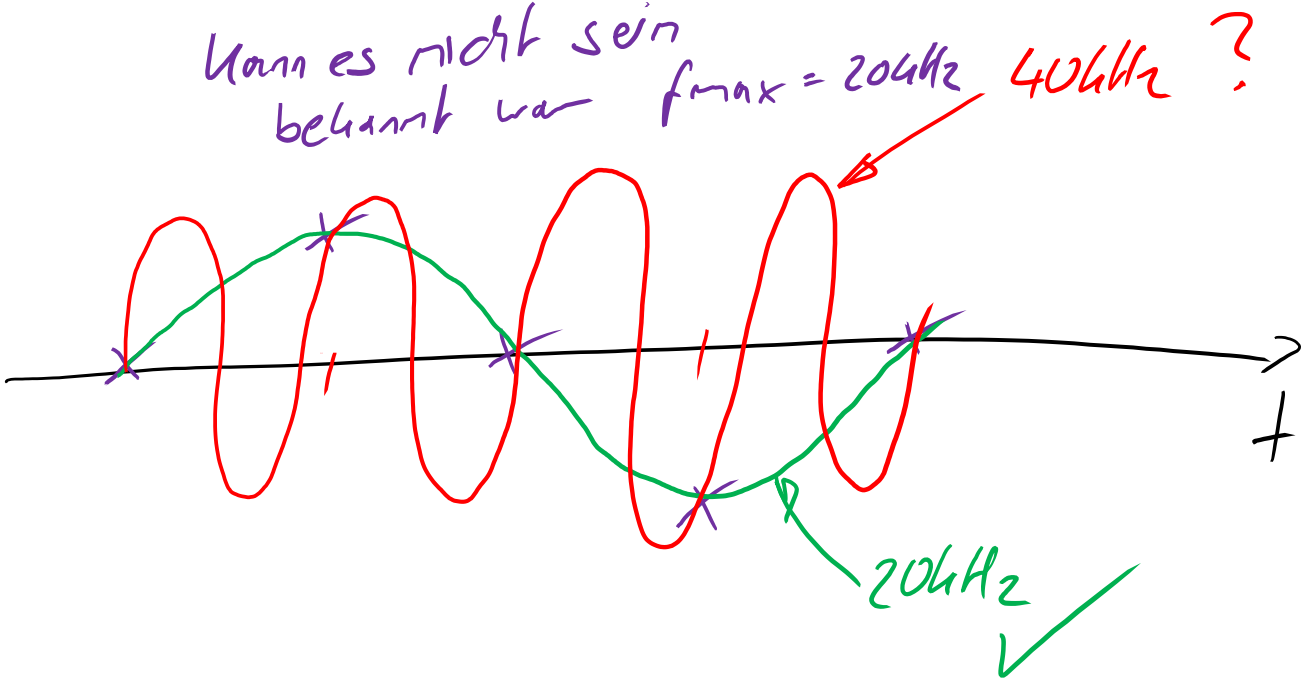



TP
→Analoges
Signal

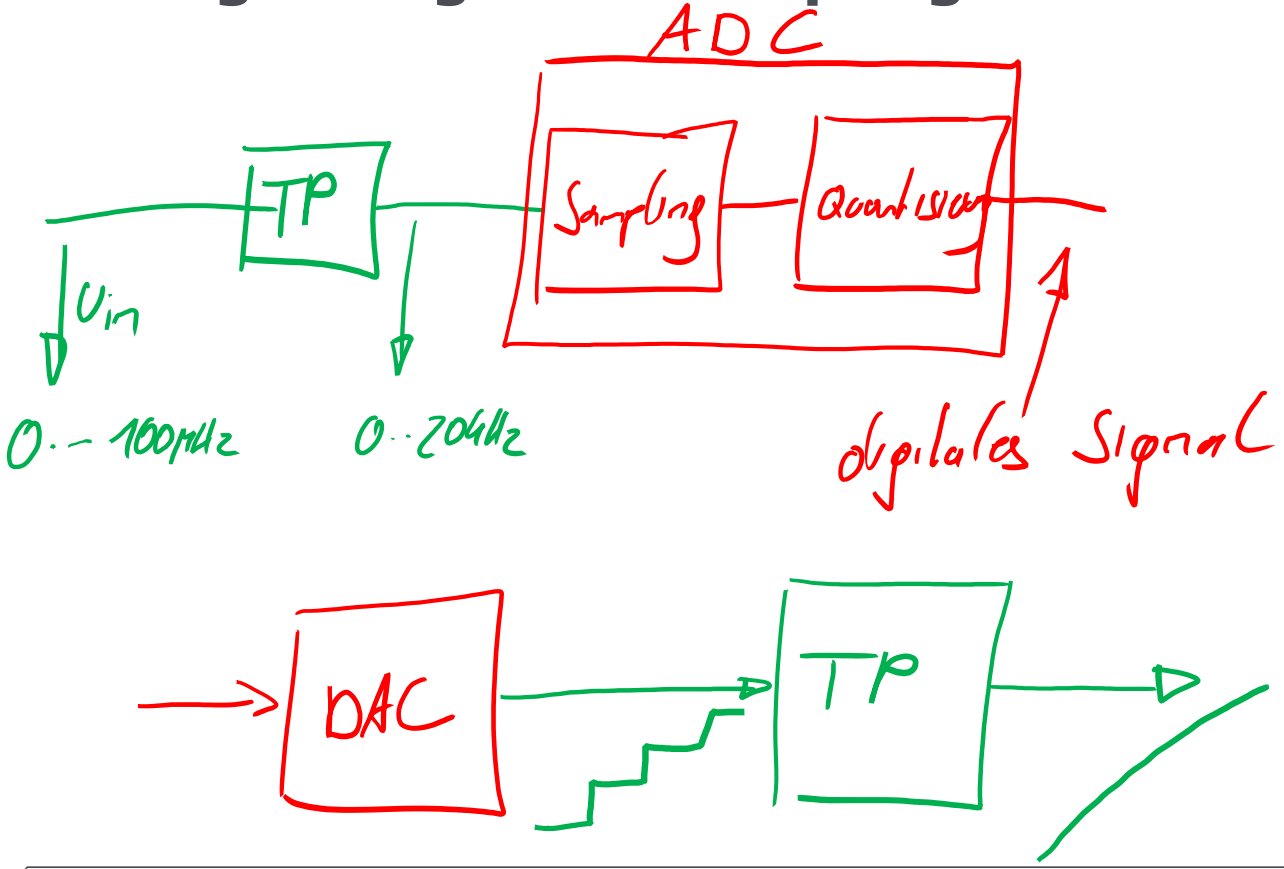
Analog to Digital – Sampling rate

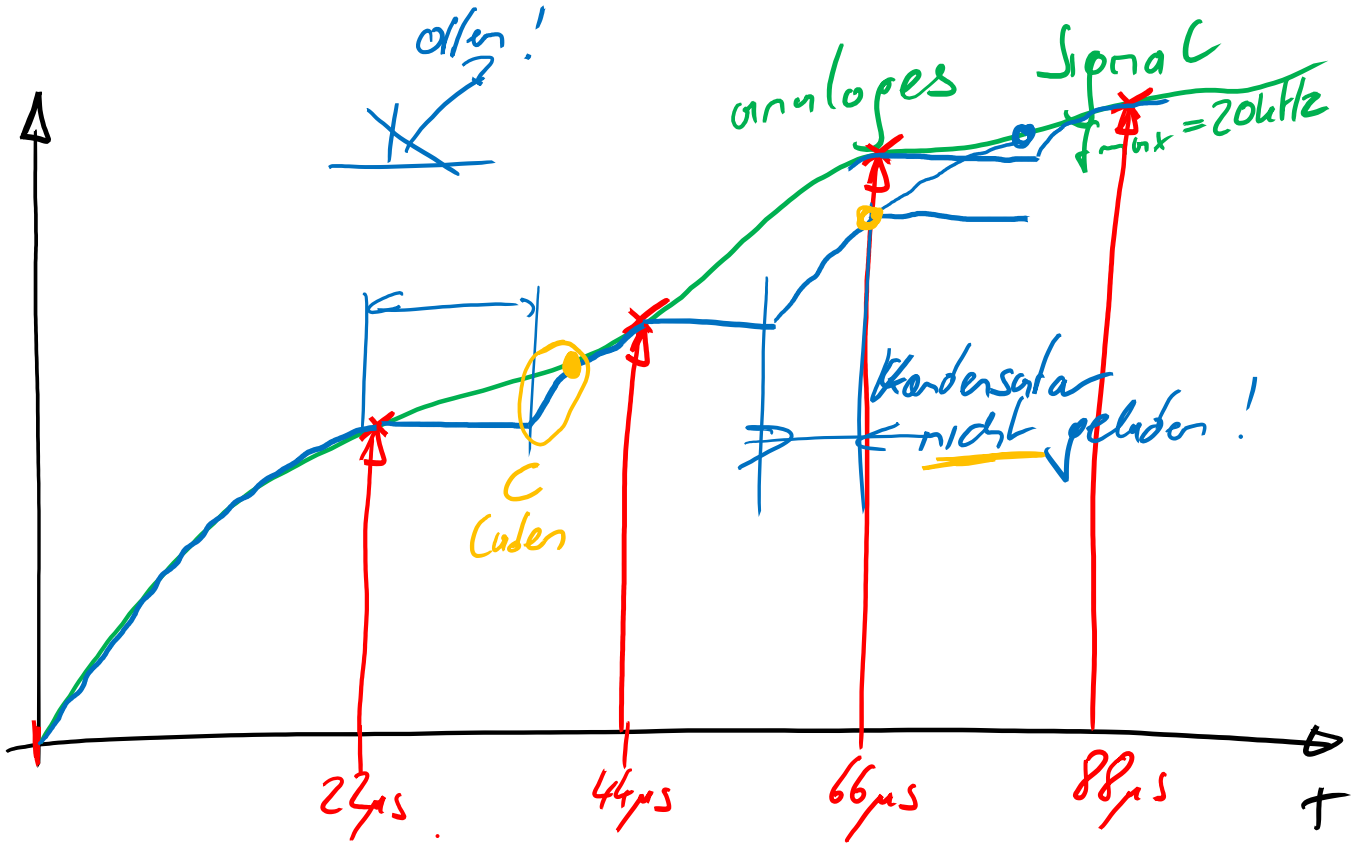


Analog to Digital – Sampling rate

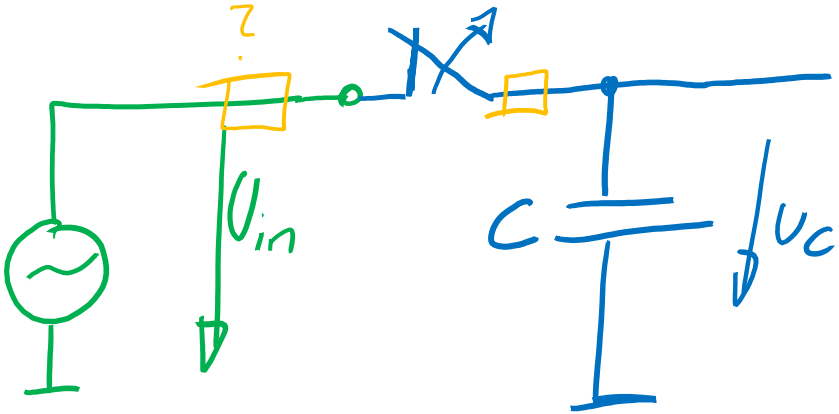


Analog to Digital – Sampling rate





Abtastrate $f = 44{,}1kHz \rightarrow T = 22\mu s$



$U_{in} = U_C$

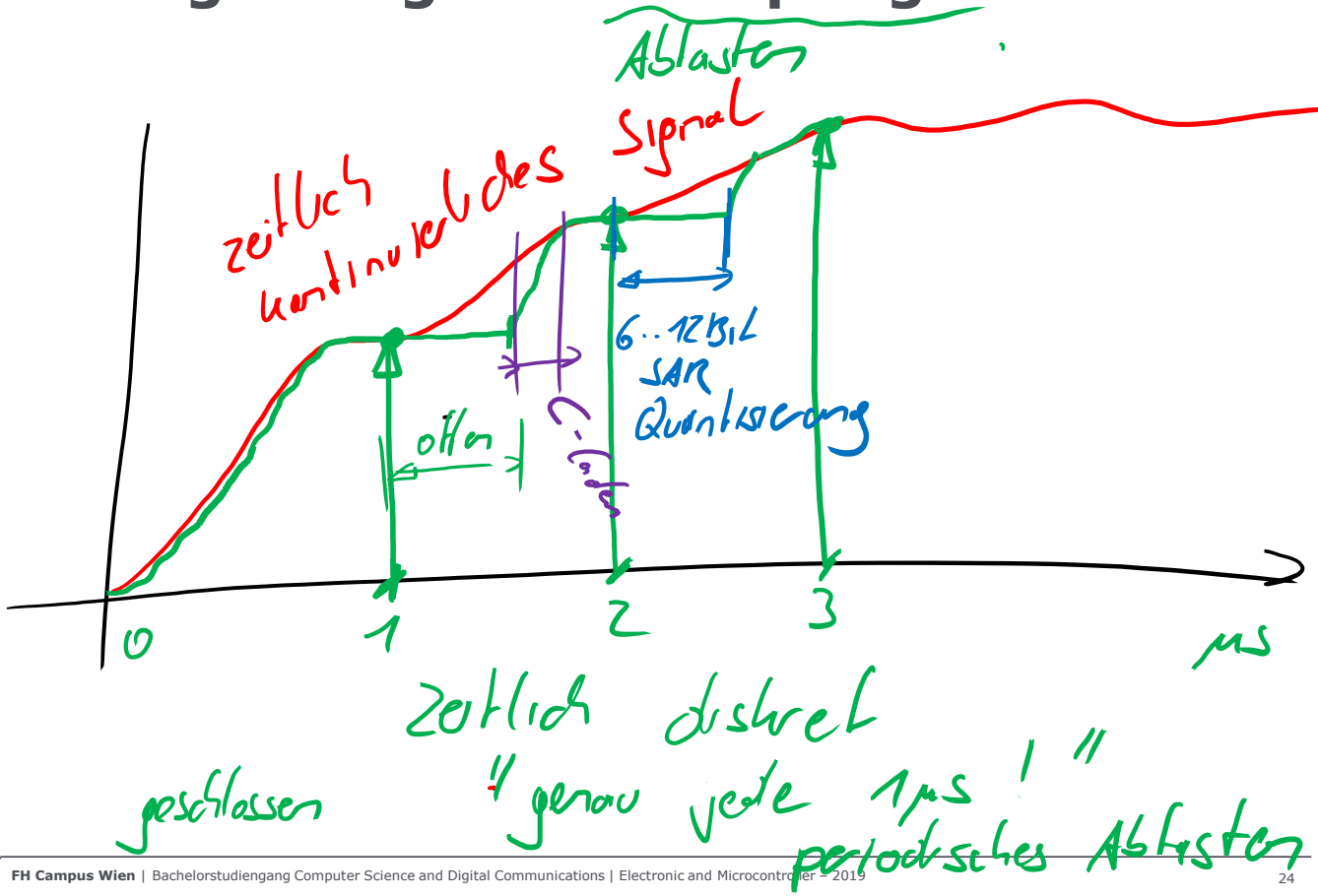
Sample & Hold

Schalter

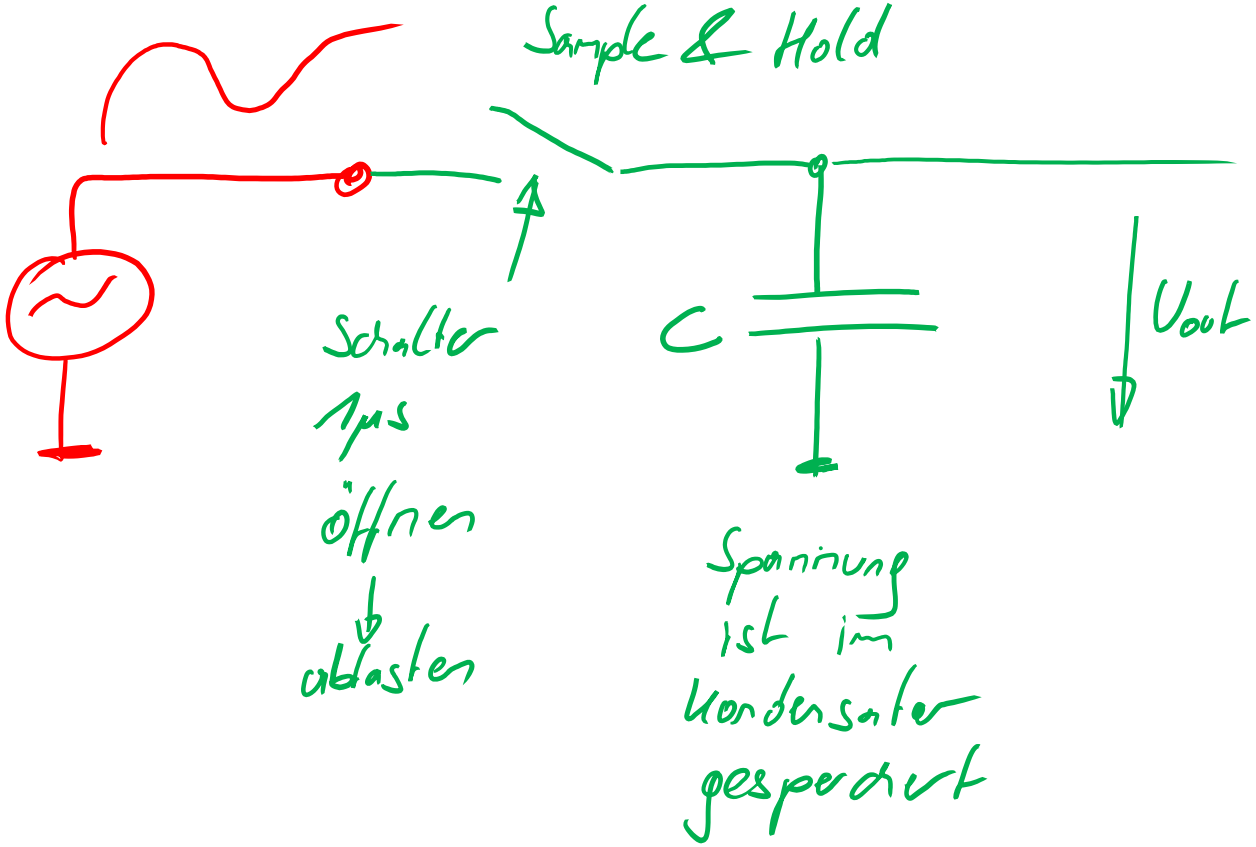
"öffnen bestimmt den Zeitpunkt!

gespeichert im Kondensator

Analog to Digital – Sampling rate



Analog to Digital – Sampling rate

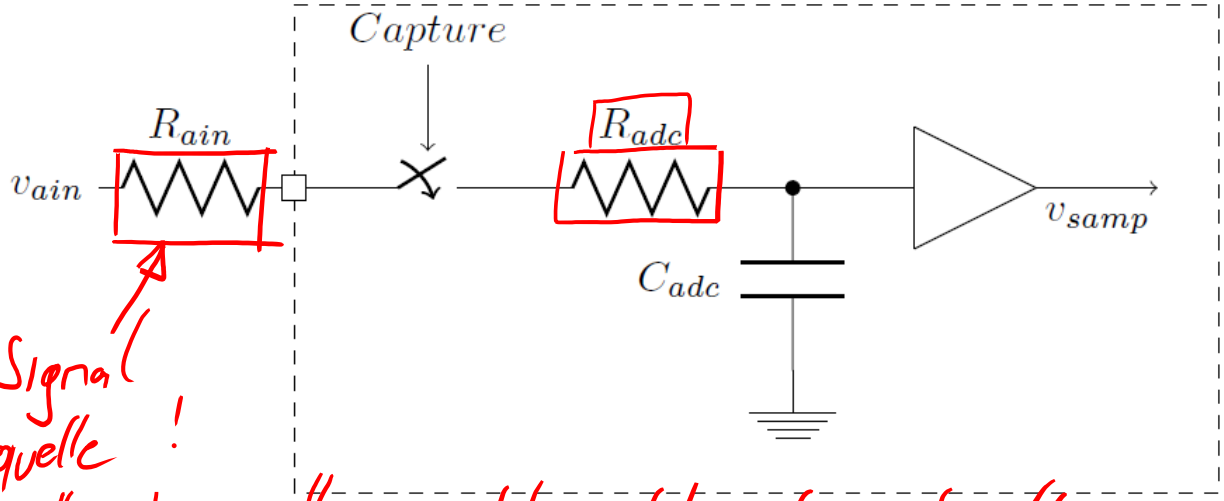


ADC – STM32 – Sampling time

ADC – STM32 – Sampling time

Channel-wise programmable sampling time

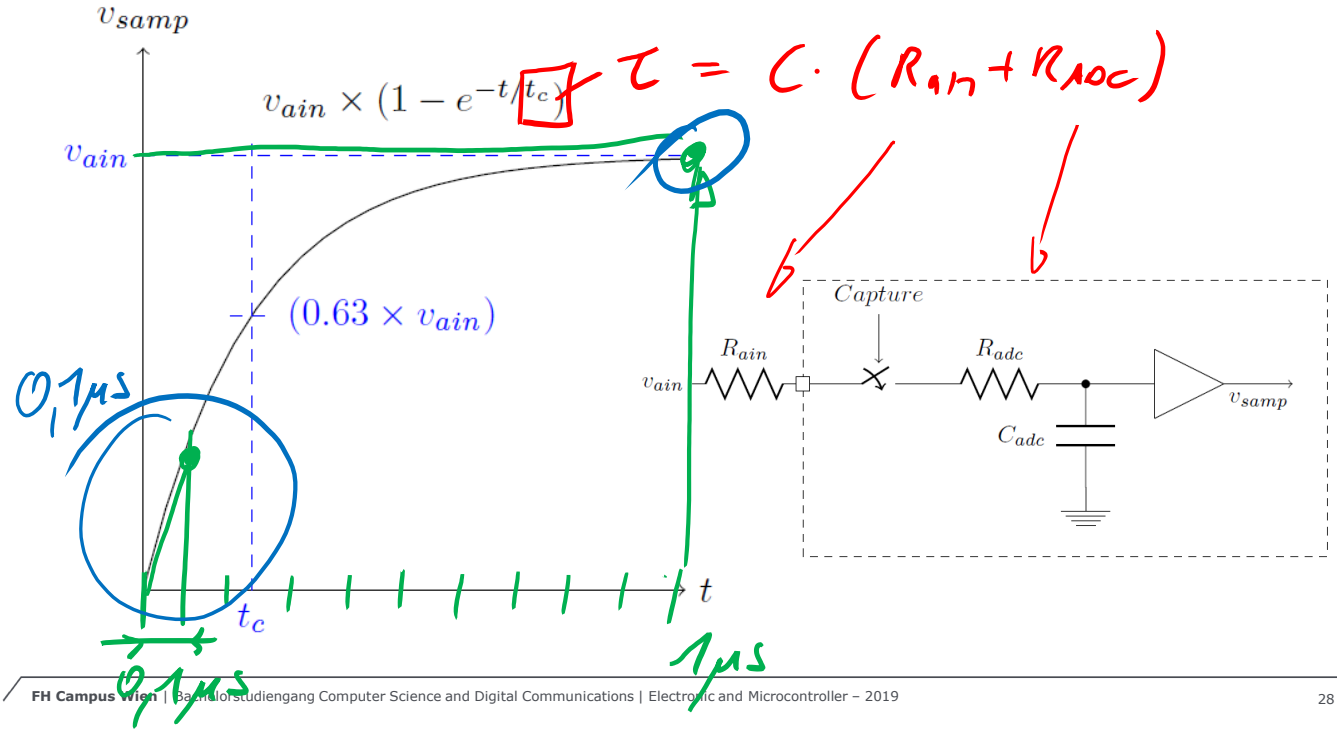
The ADC samples the input voltage for a number of ADCCLK cycles that can be modified using the SMP[2:0] bits in the ADC_SMPR1 and ADC_SMPR2 registers. Each channel can be sampled with a different sampling time.



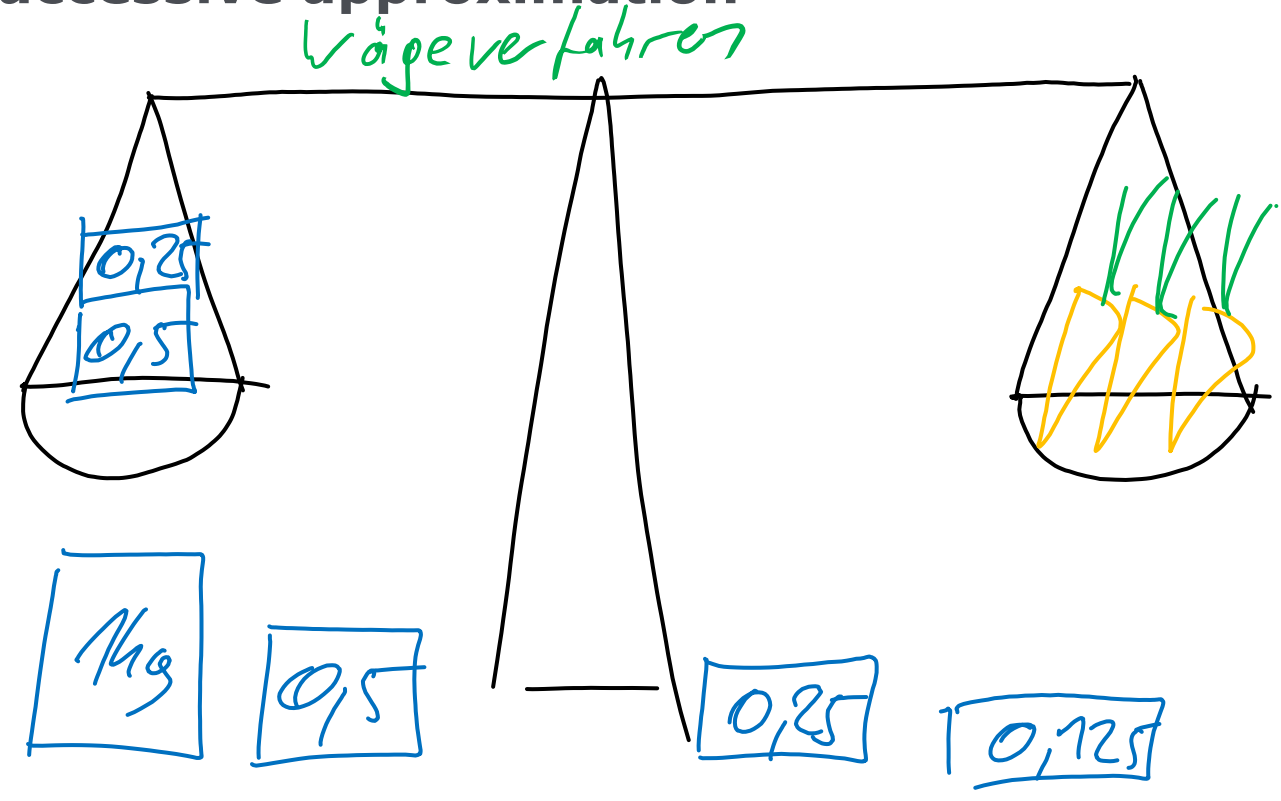
ADC – STM32 – Sampling time

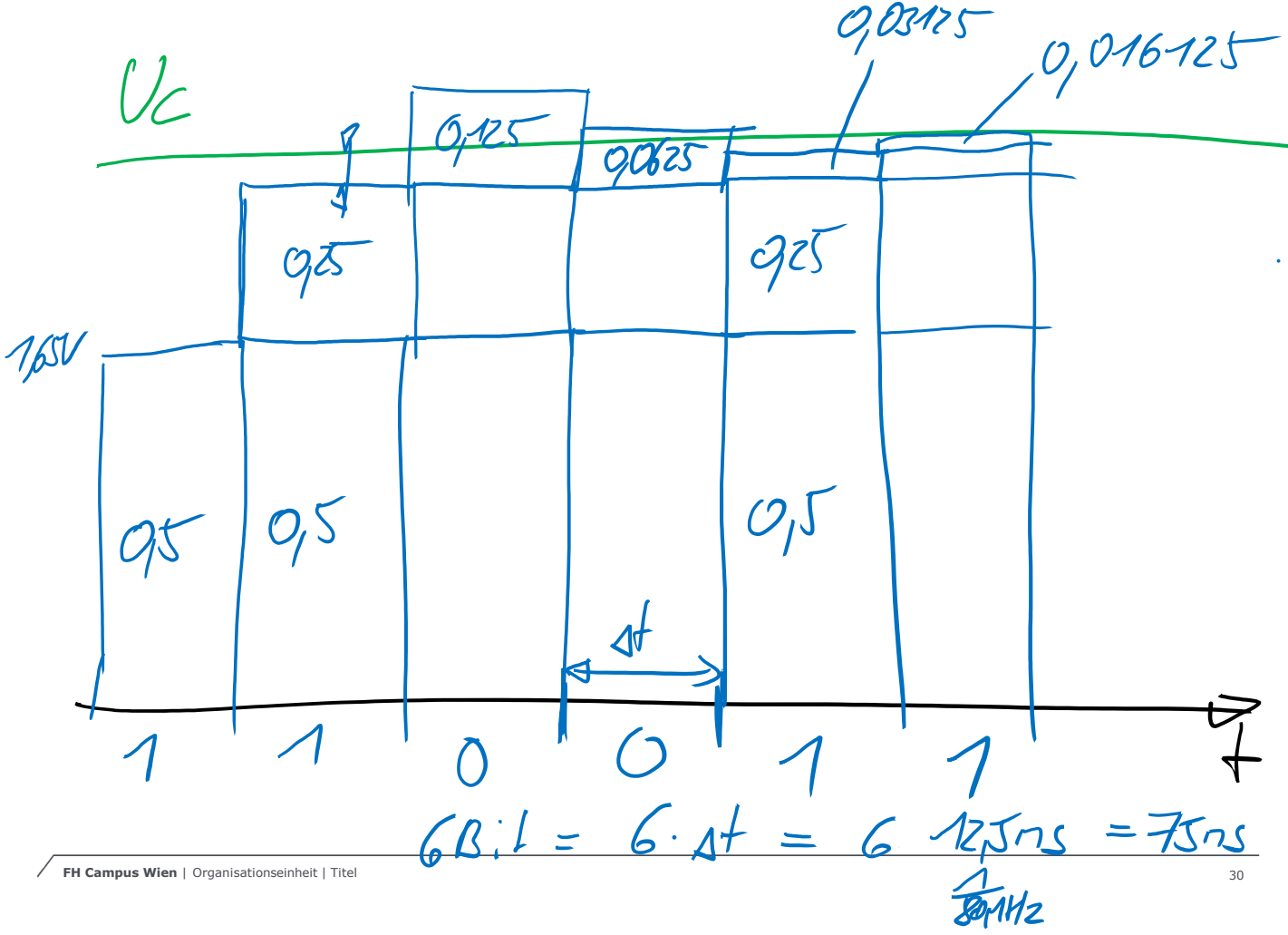
Signal Quelle

t_S (2)	Sampling time	f_{ADC} = 30 MHz	0.100	1µs	16	µs
			3	-	480	$1/f_{ADC}$



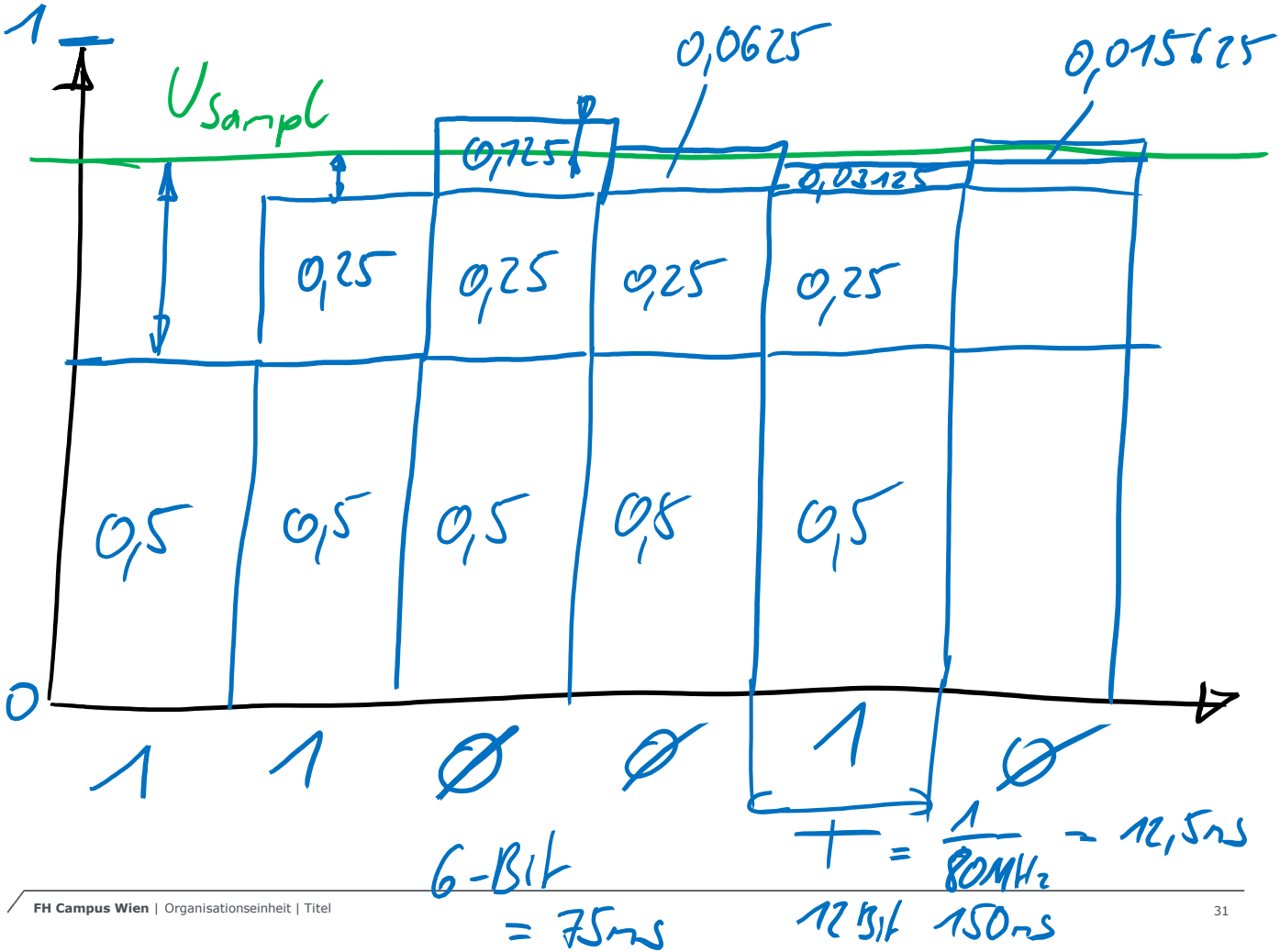
Successive approximation



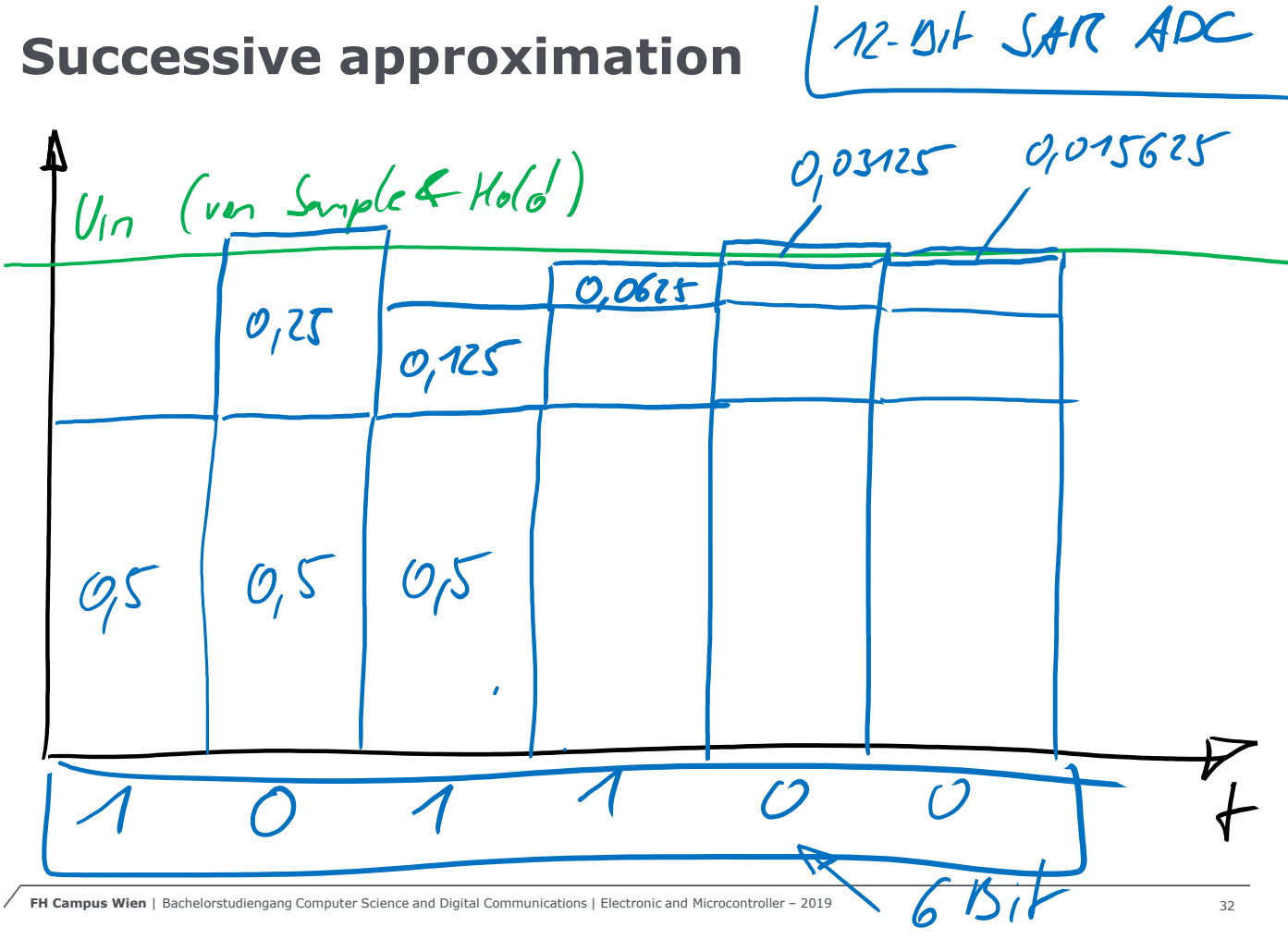


$6\,Bit = 6 \cdot \Delta t = 6 \cdot 12{,}5\,ns = 75\,ns$

$\frac{1}{80\,MHz}$

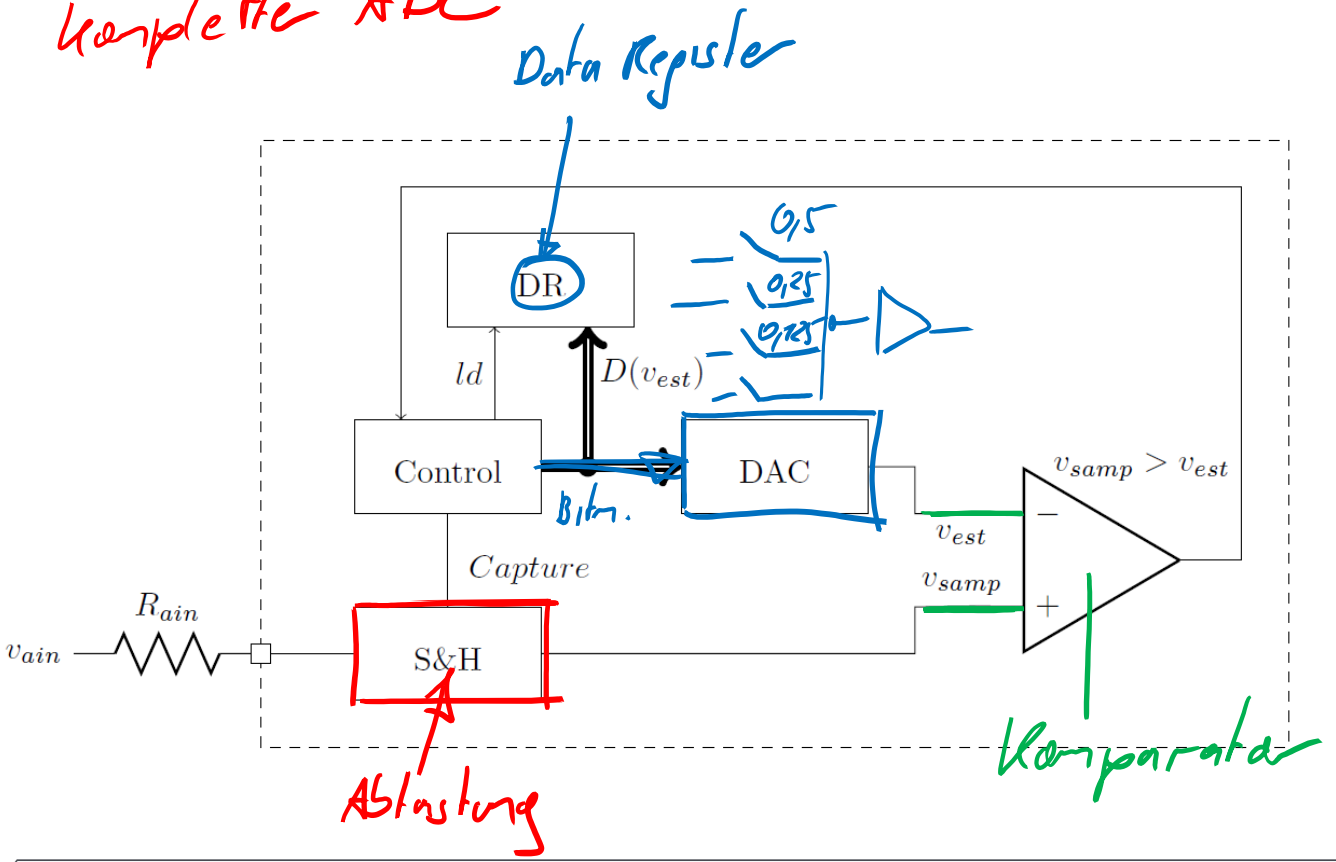

1
U_{Sampl}
0,0625
0,0156125
0,125
0,03125
0,25
0,25
0,25
0,25
0,5
0,5
0,5
0,5
0,5
0
1
1
0
0
1
0
6-Bit
= 75ns
T = 1/80MHz = 12,5ns
12 Bit 150ns

Successive approximation

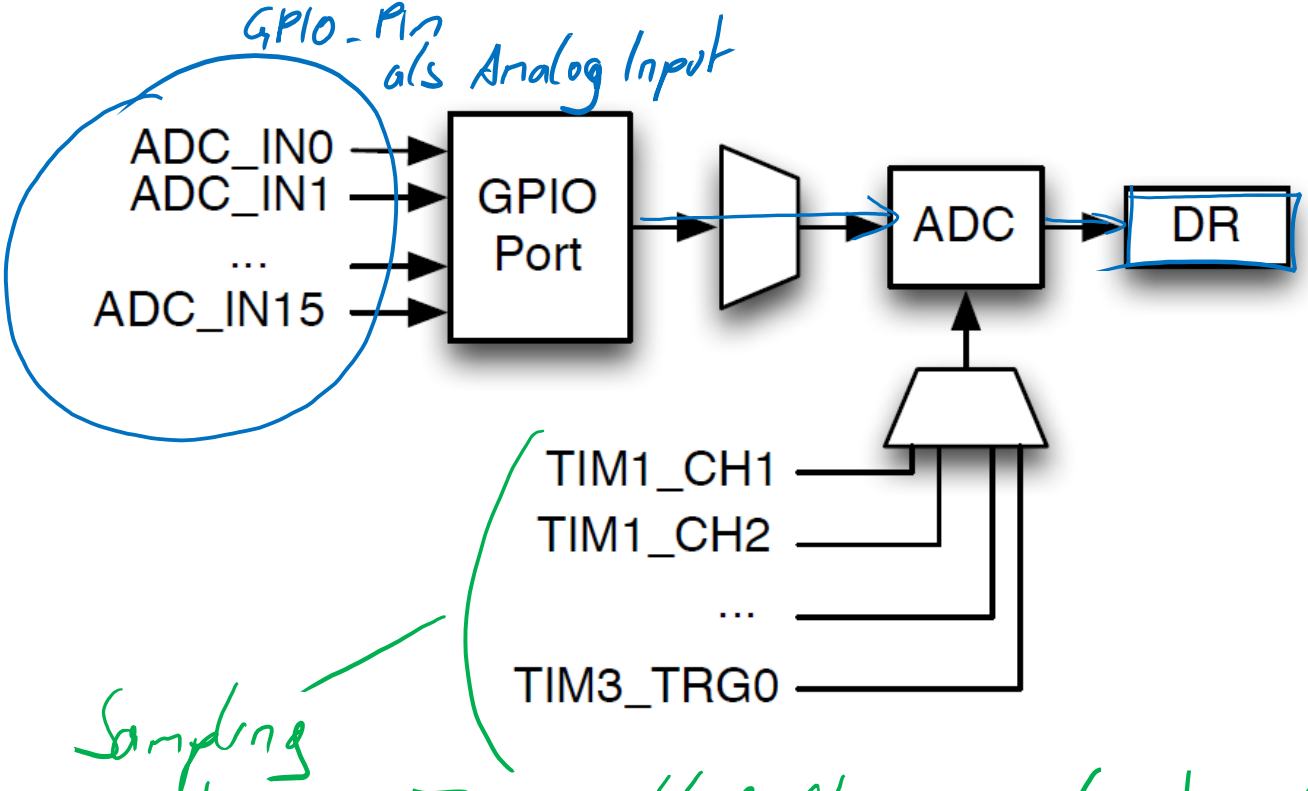


Successive approximation

Successive approximation



ADC – STM32 – Blockdiagram



ADC – STM32 – Conversion time

ADC timing

The elapsed time between the start of a conversion and the end of conversion is the sum of the configured sampling time plus the successive approximation time depending on data resolution:

$$T_{CONV} = T_{SMPL} + T_{SAR} = [\,2.5\ _{|min} + 12.5\ _{|12bit}\,] \times T_{ADC\_CLK}$$

$$T_{CONV} = T_{SMPL} + T_{SAR} = 31.25\ ns\ _{|min} + 156.25\ ns\ _{|12bit} = 187.5\ ns\ (\text{for } F_{ADC\_CLK} = 80\ MHz)$$

$\frac{1}{5M} = 0{,}2\,\mu s = 200\,ns$

6 Bit ~ 78ns

Audio bis 20kHz -> 44,1kHz

$T_{Conv} = \frac{1}{44{,}1kHz} \approx 22\,\mu s$

ADC – STM32 – L476

The device embeds 3 successive approximation analog-to-digital converters with the following features:

- 12-bit native resolution, with built-in calibration
- 5.33 Msps maximum conversion rate with full resolution
  - Down to 18.75 ns sampling time
  - Increased conversion rate for lower resolution (up to 8.88 Msps for 6-bit resolution)
- Up to 24 external channels, some of them shared between ADC1 and ADC2, or ADC1, ADC2 and ADC3.

Temperature sensor

The temperature sensor (TS) generates a voltage V_{TS} that varies linearly with temperature.

The temperature sensor is internally connected to the ADC1_IN17 and ADC3_IN17 input channels which is used to convert the sensor output voltage into a digital value.

V_{BAT} battery voltage monitoring

This embedded hardware feature allows the application to measure the V_{BAT} battery voltage using the internal ADC channel ADC1_IN18 or ADC3_IN18. As the V_{BAT} voltage may be higher than V_{DDA}, and thus outside the ADC input range, the VBAT pin is internally connected to a bridge divider by 3. As a consequence, the converted digital value is one third the V_{BAT} voltage.

ADC – STM32 – Watchdog

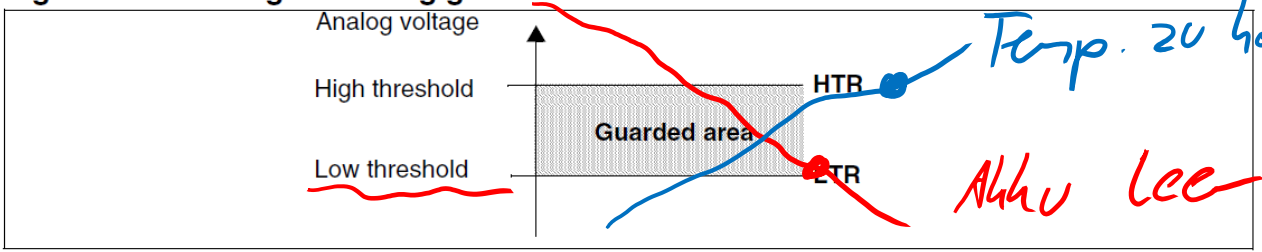
Analog watchdog

The AWD analog watchdog status bit is set if the analog voltage converted by the ADC is below a low threshold or above a high threshold. These thresholds are programmed in the 12 least significant bits of the ADC_HTR and ADC_LTR 16-bit registers. An interrupt can be enabled by using the AWDIE bit in the ADC_CR1 register.

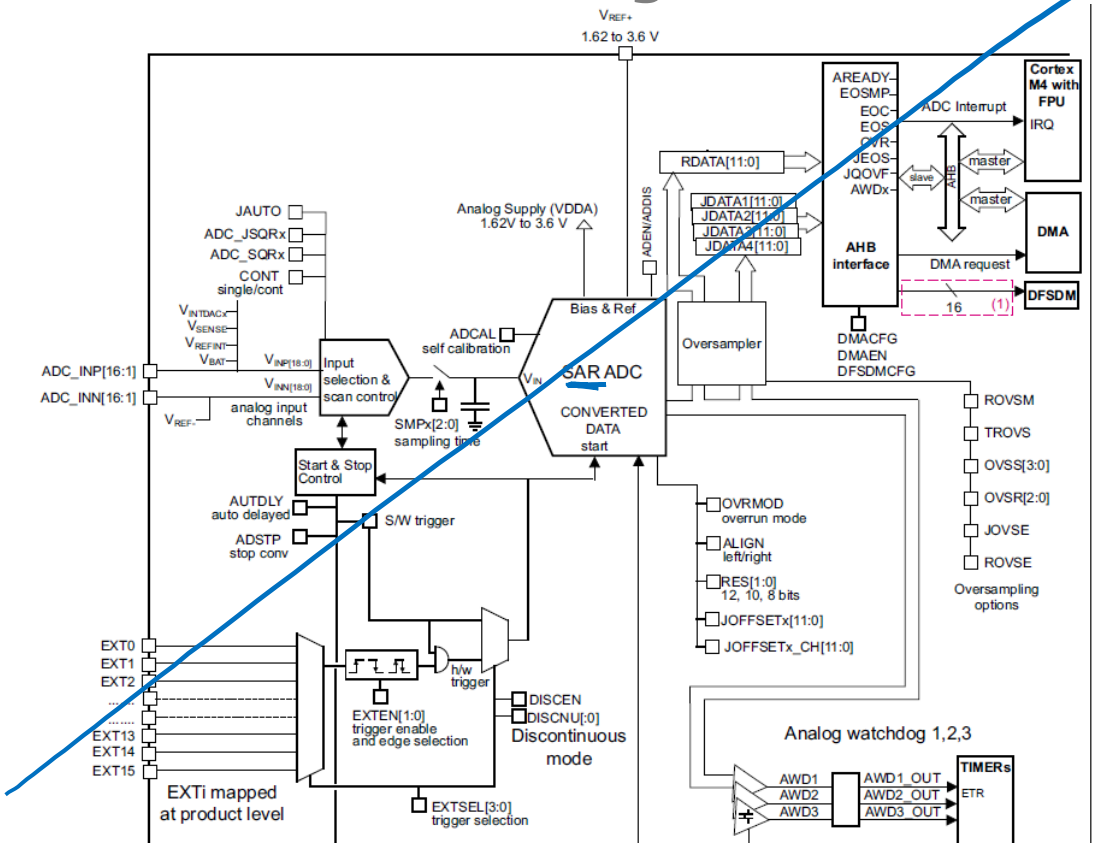
The threshold value is independent of the alignment selected by the ALIGN bit in the ADC_CR2 register. The comparison is done before the alignment (see *Section 10.5*).

The analog watchdog can be enabled on one or more channels by configuring the ADC_CR1 register as shown in *Table 58.*

Figure 26. Analog watchdog guarded area



ADC – STM32 – Blockdiagram



ADC – STM32 – Channels

There are 16 multiplexed channels. It is possible to organize the conversions in two groups: regular and injected. A group consists of a sequence of conversions which can be done on any channel and in any order. For instance, it is possible to do the conversion in the following order: Ch3, Ch8, Ch2, Ch2, Ch0, Ch2, Ch2, Ch15.

- The **regular group** is composed of up to 16 conversions. The regular channels and their order in the conversion sequence must be selected in the ADC_SQRx registers. The total number of conversions in the regular group must be written in the L[3:0] bits in the ADC_SQR1 register.
- The **injected group** is composed of up to 4 conversions. The injected channels and their order in the conversion sequence must be selected in the ADC_JSQR register. The total number of conversions in the injected group must be written in the L[1:0] bits in the ADC_JSQR register.

If the ADC_SQRx or ADC_JSQR registers are modified during a conversion, the current conversion is reset and a new start pulse is sent to the ADC to convert the new chosen group.

ADC – STM32 – Examples mbed

>Create voltage divider with potentiometer, read voltage value, convert value, output via USART.

>Set 4 threshold values (0.5 - 1.0 - 1.5 - 2.0V) on exceeding / falling turn LED on / off.

Shield D10, D11, D12, D13

"Thermometer"

Analog In

"Die aktuelle Spannung beträgt

ADC – STM32 – Examples mbed

Create voltage divider with potentiometer, read voltage value, convert value, output via USART

```
#include "mbed.h"

AnalogIn analog_value(A0);

int main() {
    float meas;

    printf("\nAnalogIn example\n");

    while(1) {
        meas = analog_value.read();
 // Converts and read the analog input value (value from 0.0 to 1.0)

        meas = meas * 3300;
// Change the value to be in the 0 to 3300 range

  printf("measure = %.0f mV\n", meas);
        wait(0.2); // 200 ms
    }
}
```

ADC – STM32 – Examples mbed

>Set 4 threshold values (0.5 - 1.0 - 1.5 - 2.0V) on exceeding / falling turn LED on / off.

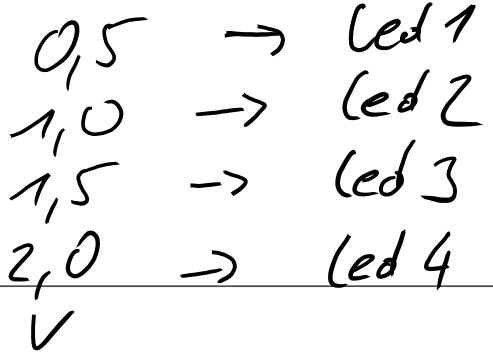
```
#include "mbed.h"

AnalogIn ain(A0);
DigitalOut led1(LED1);
DigitalOut led2(LED2);
DigitalOut led3(LED3);
DigitalOut led4(LED4);

int main() {
    while (1){
        led1 = (ain > 0.5f) ? 1 : 0;
        led2 = (ain > 1.0f) ? 1 : 0;
        led3 = (ain > 1.5f) ? 1 : 0;
        led4 = (ain > 2.0f) ? 1 : 0;
    }
}
```

ADC – STM32 – Examples

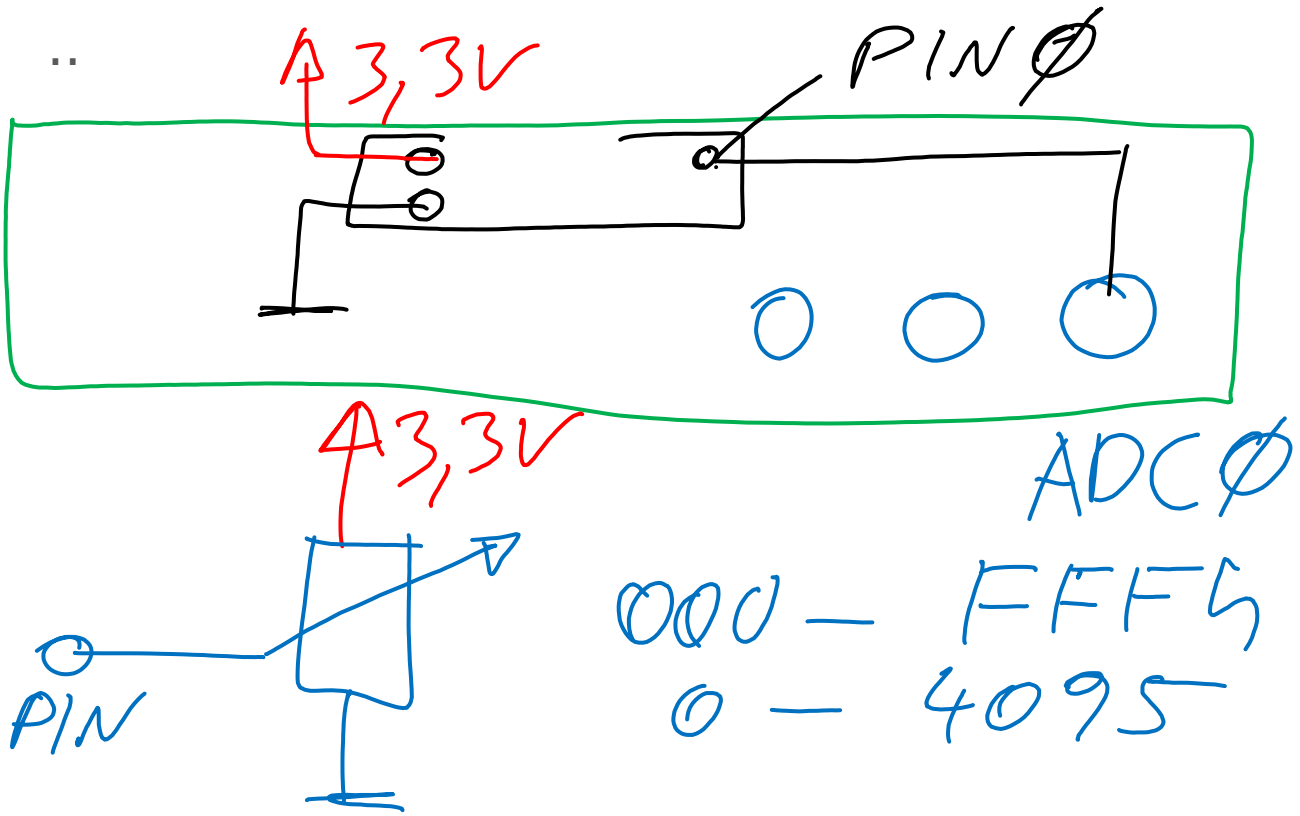
Calculate average, minimum and maximum while the program is running – send by USART

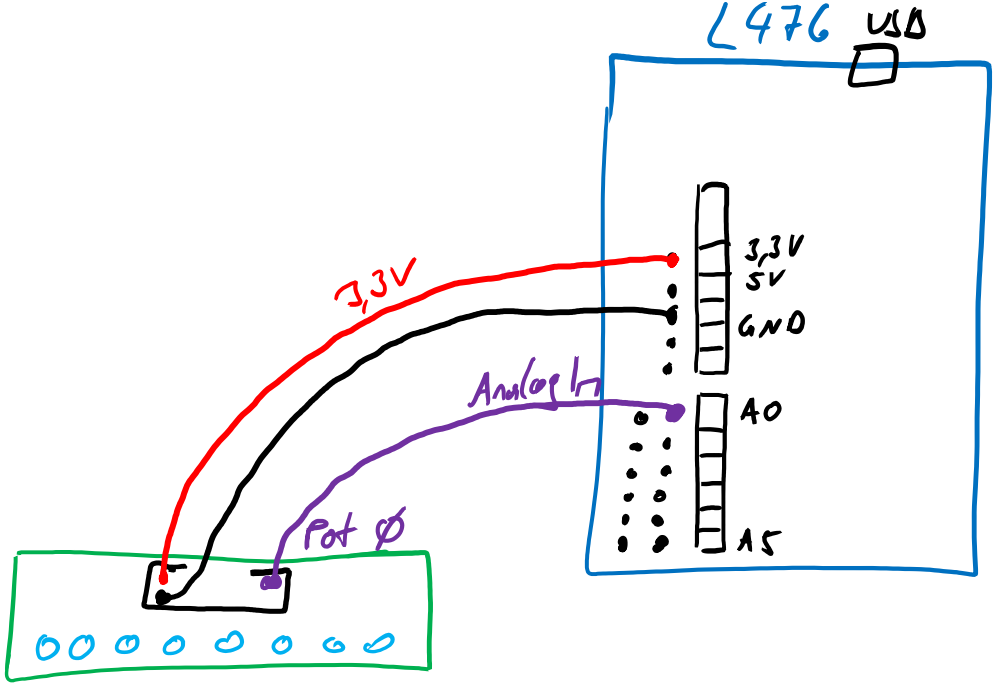
1) Ein Potentiometer an einen ADC-Pin anschließen und den Wert über die USART am Terminalprogramm ausgeben lassen.

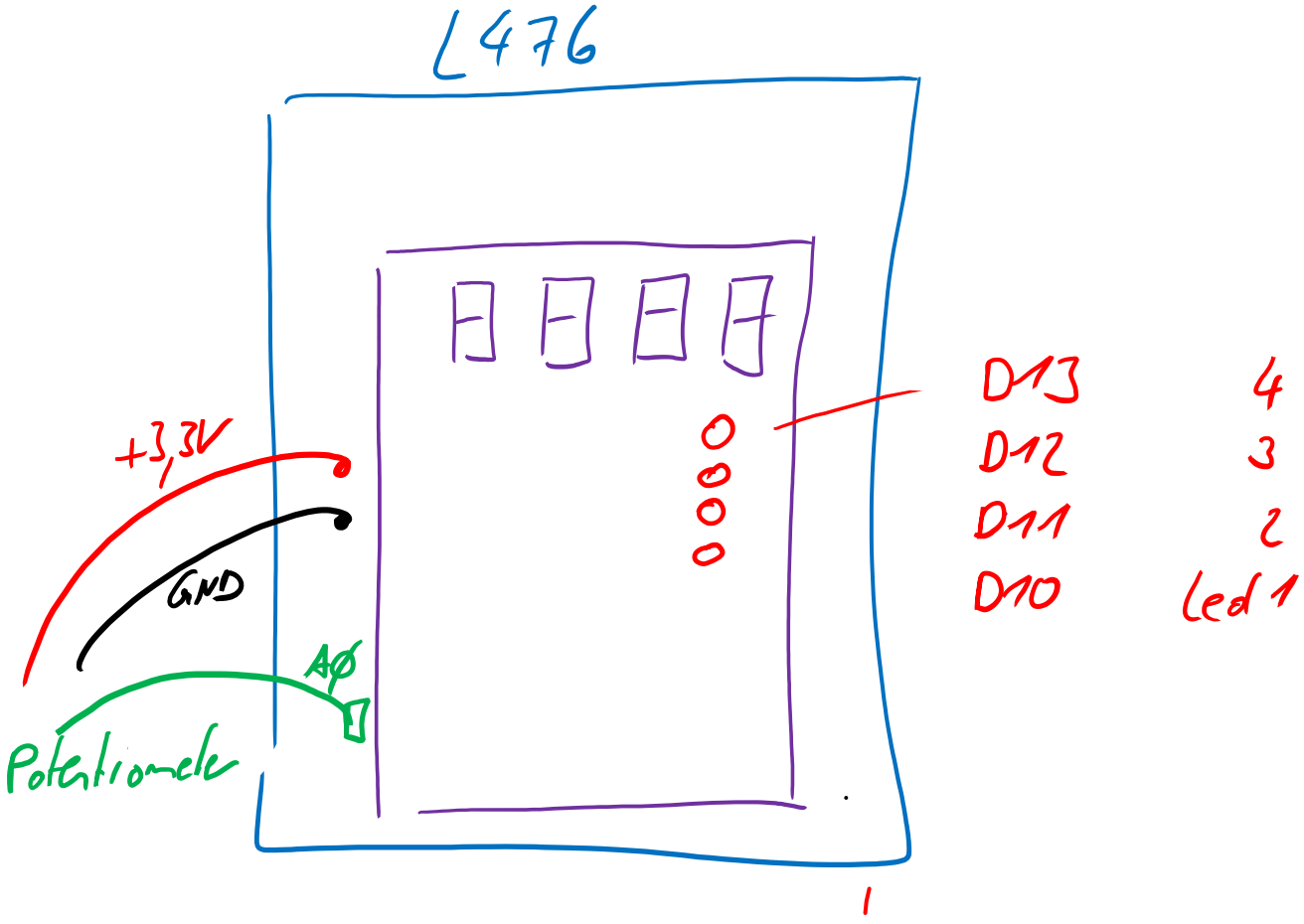
2) Als zweites Beispiel bietet sich an den gemessenen Wert mit 4 Schwellwerten zu vergleichen, und bei überschreiten jeder Schwelle eine der 4 LEDs am Multifunction-Shiel einzuschalten



Potentiometer




L476
USB
3,3V
5V
GND
A0
A5
3,3V
Analog In
Pot


L476
+3,3V
GND
A0
Potentiometer
D13
D12
D11
D10
4
3
2
Led 1
1

3a) ohne Array direkt ausgeben

aout = an;

3) Jetzt könnte man mit dem LEO-Funktionsgenerator eine Sinusspannung mit einer Frequenz von 1kHz erzeugen und diese Spannung mit 20kHz mit dem ADC einlesen (also 20 Werte für eine Periode) und in einem Array speichern.

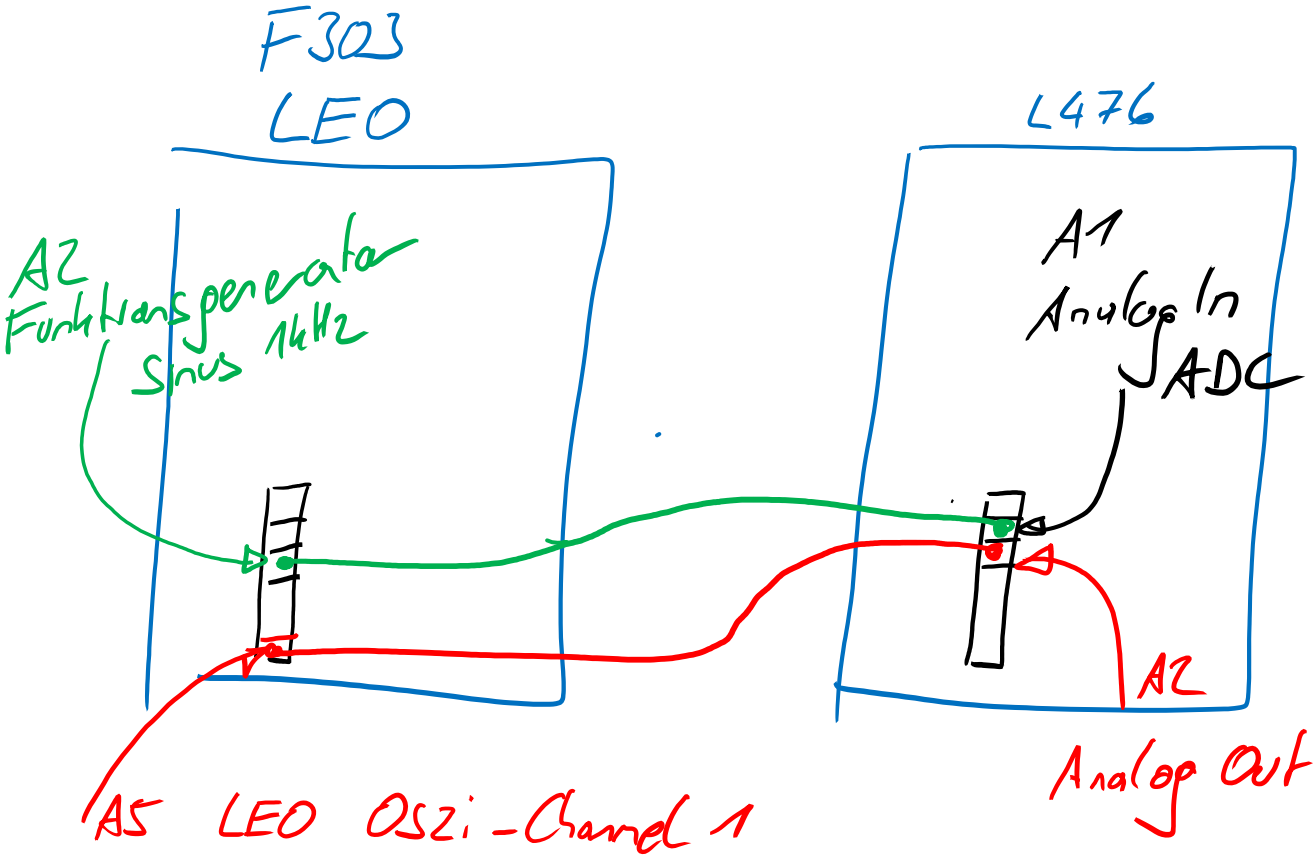
Anschließend diese Werte verwenden und mit dem DAC ausgeben.

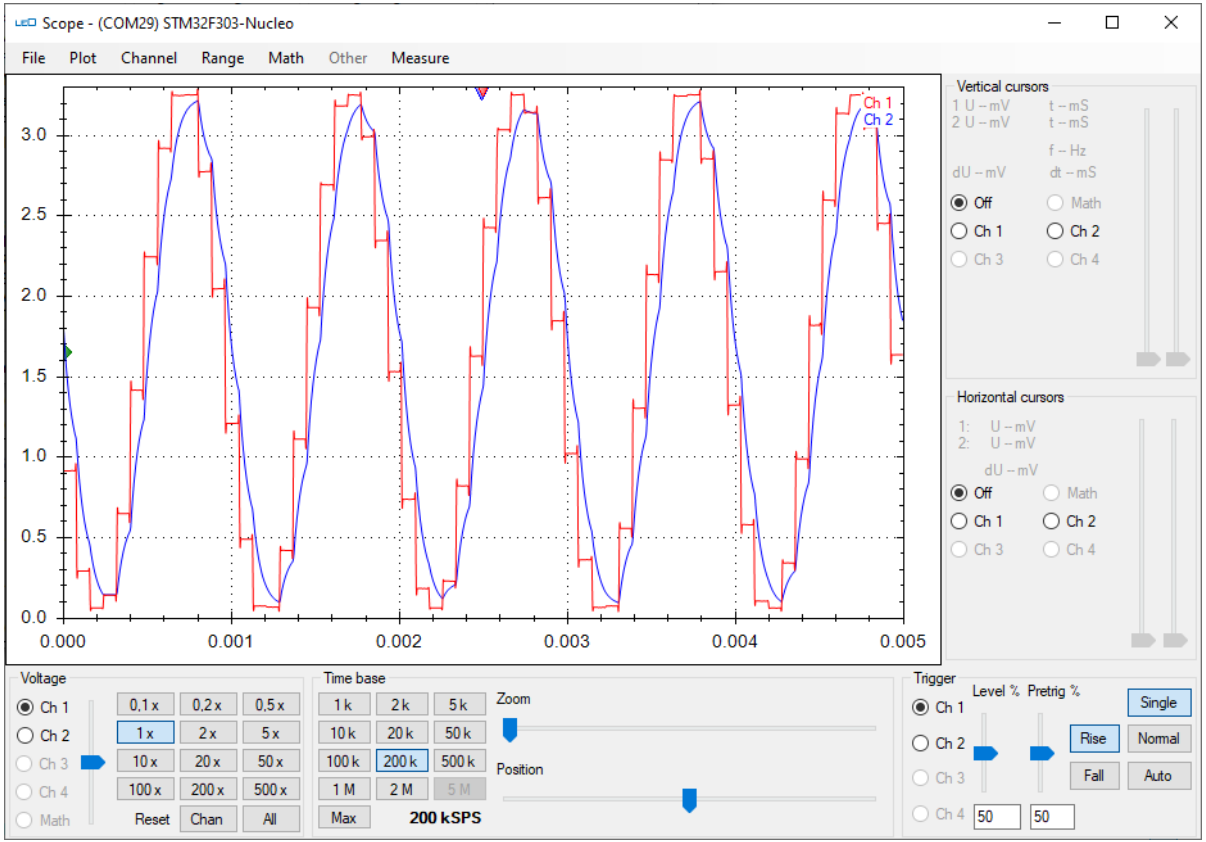
Je nachdem wie lange das Delay zwischen den einzelnen ausgegebenen Werte ist kann nun ein Sinus mit unterschiedlicher Frequenz erzeugt werden.

f = 20kHz T = 50µs

4) Ein weiteres mögliches Beispiel ist ein Lichtsensor, wird Dunkelheit erkannt dann wird das Licht (die LED) eingeschaltet.

wait_us(50);


F303
LEO
L476
A2
Funktionsgenerator
Sinus 1kHz
A1
Analog In
ADC
A2
A5 LEO OSZI - Channel 1
Analog Out

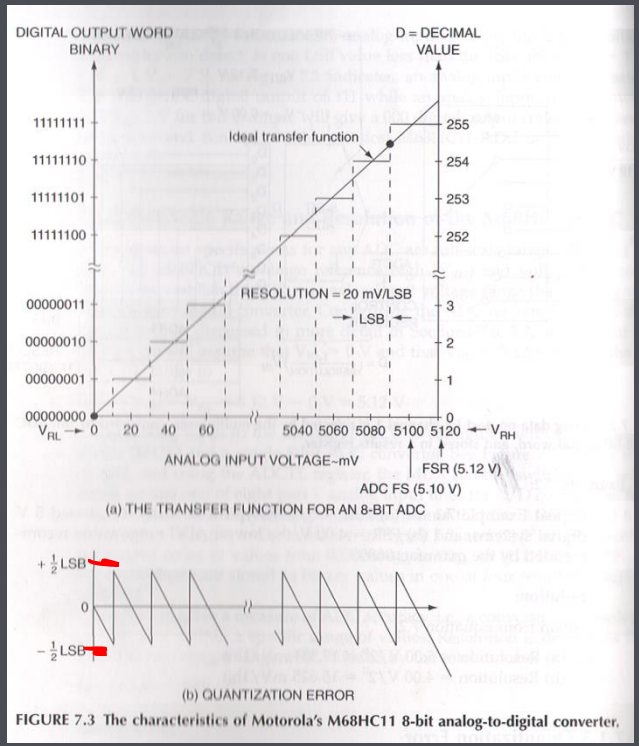

Scope - (COM29) STM32F303-Nucleo
File
Plot
Channel
Range
Math
Other
Measure
Ch 1
Ch 2
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0.000
0.001
0.002
0.003
0.004
0.005
Vertical cursors
1 U -- mV
2 U -- mV
t -- mS
t -- mS
f -- Hz
dU -- mV
dt -- mS
Off
Math
Ch 1
Ch 2
Ch 3
Ch 4
Horizontal cursors
1: U -- mV
2: U -- mV
dU -- mV
Off
Math
Ch 1
Ch 2
Ch 3
Ch 4
Voltage
Ch 1
Ch 2
Ch 3
Ch 4
Math
0,1 x
0,2 x
0,5 x
1 x
2 x
5 x
10 x
20 x
50 x
100 x
200 x
500 x
Reset
Chan
All
Time base
1 k
2 k
5 k
10 k
20 k
50 k
100 k
200 k
500 k
1 M
2 M
5 M
Max
200 kSPS
Zoom
Position
Trigger
Ch 1
Ch 2
Ch 3
Ch 4
Level %
Pretrig %
50
50
Single
Rise
Normal
Fall
Auto

```
#include "mbed.h"

AnalogOut aout(A2);
AnalogIn ain(A1);
int array[20];

int main()
{
    //uint16_t out_value = 0;
    //uint16_t in_value = 0;

     for(int i=0; i<20; i++){
            array[i] =  ain.read_u16();
            wait_us(25);
            }

    while(1) {
        //in_value = ain.read_u16();
        //out_value = in_value;
        //aout.write_u16(out_value);
        //wait_us(50);

        for(int j=0; j<20; j++){
            aout.write_u16(array[j]);
            wait_us(100);
            }
        }
}
```

Quantization Error

>Graphical view

>Note how discrete
> values represent
> the analog signal



Analog Voltage