ELMC - NP - New

1) Spannungsteiler - ADC

Es soll ein Spannungsteiler entsprechend der Abbildung 1 aufgebaut werden. Von den beiden Widerständen ist ein Widerstand der aus der Übung bekannte 10kOhm Widerstand (ein Ring in oranger Farbe), der zweite Widerstand hat einen **im Moment unbekannten Wert!**

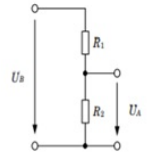
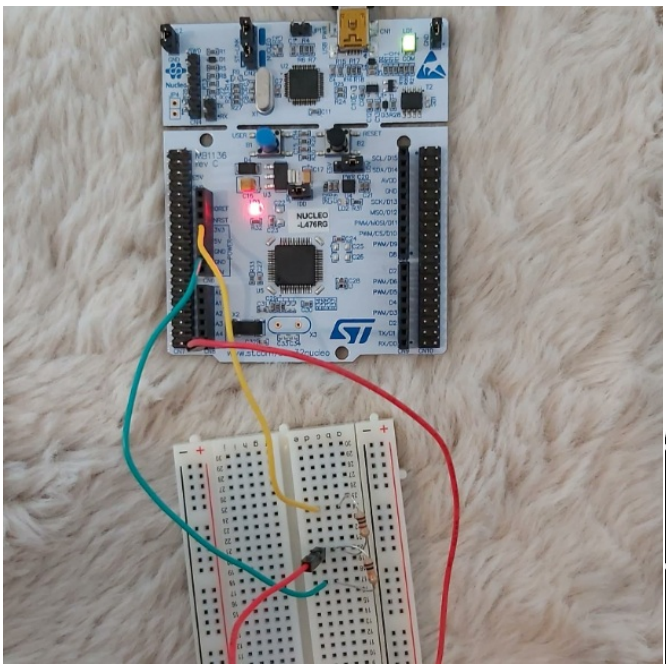


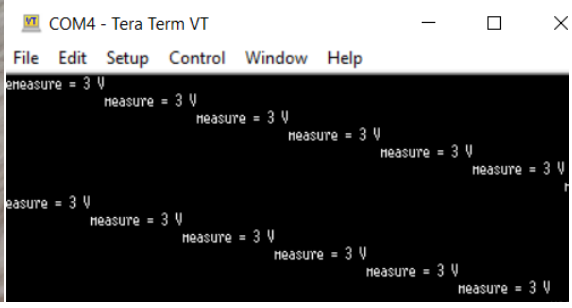
Abbildung1: Unbelasteter Spannungsteiler

Berechnen mit ADC, display auf TeraTerm

Bauen Sie die Schaltung auf, verwenden Sie R1= unbekannt und R2=10kOhm. Verwenden Sie +3,3V vom STM32L476 für die Eingangsspannung UB und Messen Sie Spannung UA mit dem ADC des STM32L476. Rechnen Sie den vom ADC gelieferten Zahlenwert in eine Spannung im Bereich 0-3,3V um und geben Sie diesen Wert (die Spannung) mit Hilfe der UART und dem virtuellen COMPort im Terminalprogramm aus.

```
#include "mbed.h"


//Bauen Sie die Schaltung auf, verwenden Sie R1= unbekannt und R2=10kOhm.
//Verwenden Sie +3,3V vom STM32L476 für die Eingangsspannung UB und Messen Sie Spannung UA mit dem ADC des STM32L476.
//Rechnen Sie den vom ADC gelieferten Zahlenwert in eine Spannung im Bereich 0-3,3V um
//und geben Sie diesen Wert (die Spannung) mit Hilfe der UART und dem virtuellen COMPort im Terminalprogramm aus.

//Unbekannter R - zu 3,3V
//Zwischen den Widerständen - A5
//Bekannter R - zu Ground


AnalogIn analog_value(A5);

int main() {
    float meas;

while(1) {
    meas = analog_value.read(); // Converts and read the analog input value (value from 0.0 to 1.0)
    meas = meas * 3.3; // Change the value to be in the 0 to 3300 range
    printf("measure = %.0f V\n", meas);
    wait_ms(200);
  }
}
```





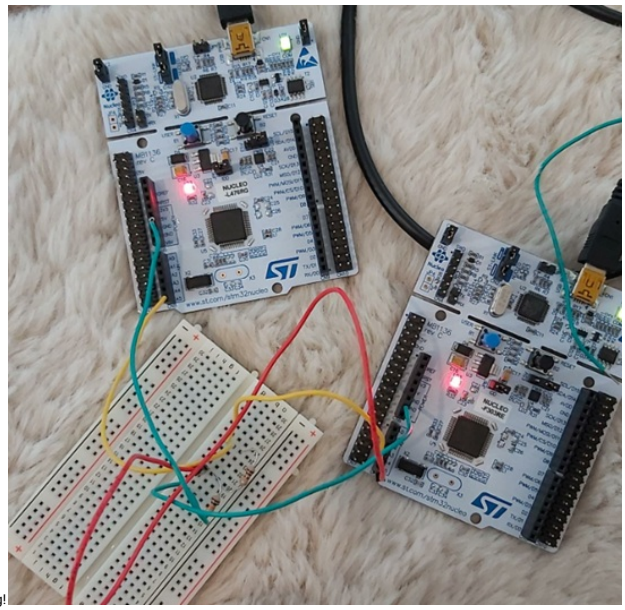
https://www.digikey.com/en/resources/conversion-calculators/conversion-calculator-resistor-color-code-4-band

- R1 *unbekannt*: 1 kOhm
- R2: 10 kOhm
- I = UA/10kOhm = 3/10000 = 0.0003
- Rges = UB/I = 3.3/0.0003 = 11000 Ohm = 11 Ohm
- R1 = Rges-R2 = 11kOhm - 10kOhm = 1 kOhm

Berechnen mit LEO

Bauen Sie die Schaltung auf und verwenden Sie als Eingangsspannung ein Rechtecksignal mit einer Frequenz von f= kHz. Erzeugen Sie dieses Rechtecksignal mit dem STM32L476.

Messen Sie die Ausgangsspannungen UA mit Hilfe des Little Embedded Oszilloskop – LEO am STM32F303. Verwenden Sie zuerst den 10kOhm Widerstand als R2 und den nicht bekannten Widerstand als R1. Berechnen Sie aus der



gemessenen Spannung den Strom und den Gesamtwiderstand der Schaltung!

```
#include "mbed.h"

//Bauen Sie die Schaltung auf und verwenden Sie als Eingangsspannung ein Rechtecksignal mit einer Frequenz von f= kHz.
//Erzeugen Sie dieses Rechtecksignal mit dem STM32L476.
//Messen Sie die Ausgangsspannungen UA mit Hilfe des Little Embedded Oszilloskop – LEO am STM32F303.
//Verwenden Sie zuerst den 10kOhm Widerstand als R2 und den nicht bekannten Widerstand als R1.
//Berechnen Sie aus der gemessenen Spannung den Strom und den Gesamtwiderstand der Schaltung!

/*
2 Widerstaende unten oben
L476 A5 zum linken Widerstand ganz unten; Ground zum 2 Widerstand ganz rechts
F303 Ground zum 2. Widerstand ganz rechts oberhalb des ersten
   A4 ganz unten in der mitte, wo sich beide Widerstaende treffen
   A5 zum ersten Widerstand ganz links oberhalb von L476 Kabel
*/


// 1kHz = 1000 Hz

DigitalOut dout(PC_0);

int main() {
    float frequency = 8000;
    float hoch = (1/frequency)/2;

while(1) {
    dout = !dout;
    wait(hoch);
  }
}
```

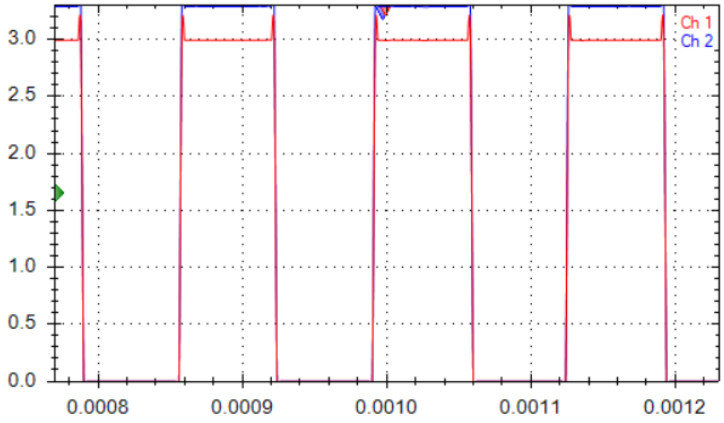
Der rote Strich bei 3,0 zeigt den Spannungsabfall. Den Wert einfach in die Gleichung einsetzen. Hier:

```
Channel 1 blau = Rechteckssignal mit 0-3,3V
Channel 3 rot = Spannungsabfal
```

Wenn mans umgekehrt ansteckt, ändern sich nur die Farben

```
R1 unbekannt: 1 kOhm
R2: 10 kOhm
```

I = UA/10kOhm = 3/10000 = 0.0003 Rges = UB/I = 3.3/0.0003 = 11000 Ohm = 11 Ohm R1 = Rges-R2 = 11kOhm - 10kOhm = 1 kOhm



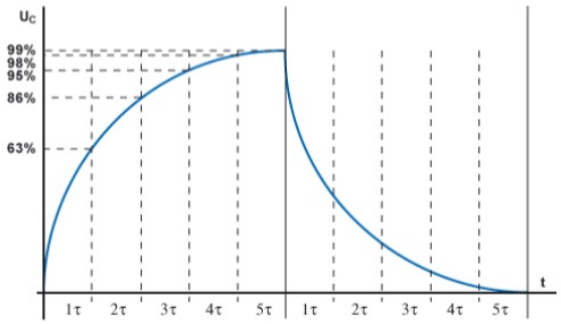
2) RC-Tiefpass

Bauen Sie nun einen RC-Tiefpass auf. Verwenden Sie dazu jenen Widerstand dessen Wert Sie soeben in Aufgabe 1 ermittelt haben!

- Kondensator: C= 100 mF
- Zeitkonstante: tau = C * R=100nF * 10kOhm = 1000 microsec = 1 milisec
  - Damit der Kondensator sich vollständig aufladen kann, braucht er 5tau, dh.:
    - 5 * 1ms = 5ms
- R = 1 kOhm

Erzeugen Sie ein Rechtecksignal mit dem STM32L476. Berechnen Sie die Frequenz für dieses Rechtecksignal damit der Kondensator sich jeweils voll auf- und entladen kann! (T/2 = 5 tau)

- T = 10,000 microsec = 10 milisec = 0,01 s

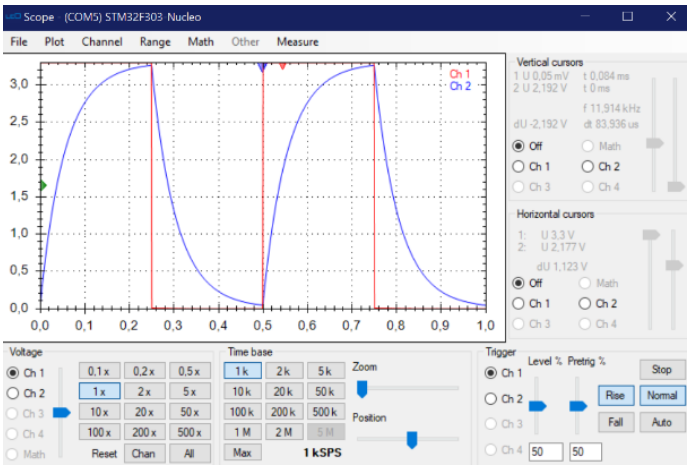


- f = 1/0,01 = 100 Hz

Fügen Sie hier einen Screenshot des LEO ein in dem das Rechtecksignal am Eingang U1 und die Spannung am Kondensator C – U2 sichtbar sind. Das Bild sollte 1-2 Perioden des Rechtecksignals zeigen! **Im folgenden Beispiel:**

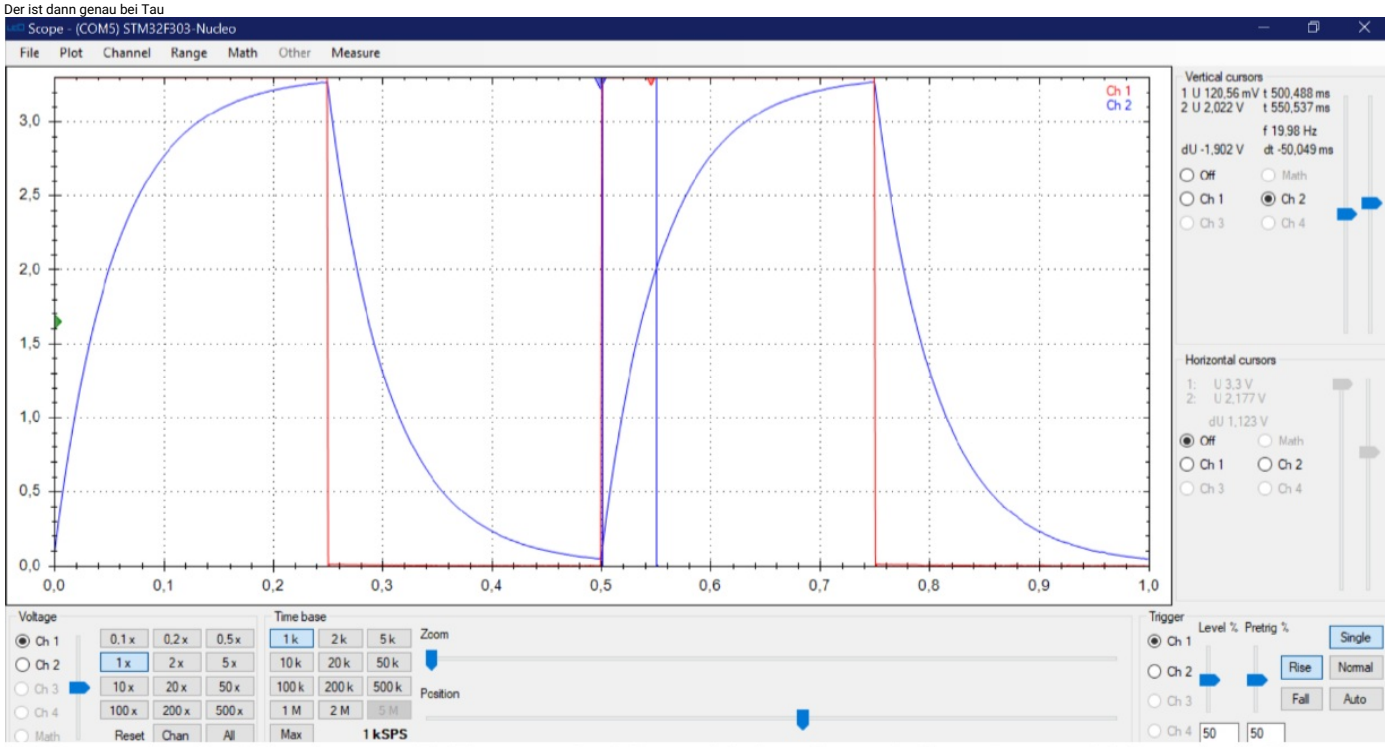
- Kondensator: 10 mikrofarad
- R: 5kOhm
- Zeitkonstante: T = R * C = 5k * 10 mikrofarad = 0,05 s = 50 ms dauert 1 tau
  - Wir wollen aber 5 tau haben , also 5 * 50 = 250 ms
- T(Periode) = (250 * 2) 500 ms
- f = 1/500ms = 2 Hz

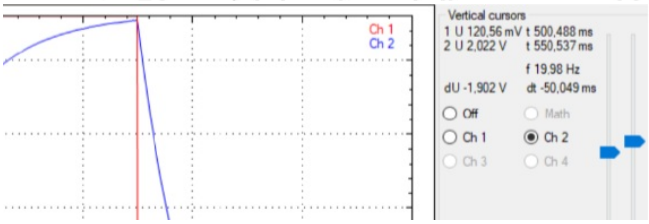
```
#include "mbed.h"
//Erzeugen Sie ein Rechtecksignal mit dem STM32L476.
//Berechnen Sie die Frequenz für dieses Rechtecksignal damit der Kondensator
//sich jeweils voll auf- und entladen kann! (T/2 = 5 tau )

/*
2 Widerstaende unten oben
L476 A5 zum linken Widerstand ganz unten; Ground zum 2 Widerstand ganz rechts
F303 Ground zum 2. Widerstand ganz rechts oberhalb des ersten
   A4 ganz unten in der mitte, wo sich beide Widerstaende treffen
   A5 zum ersten Widerstand ganz links oberhalb von L476 Kabel
*/

//tau = 10 kOhm * 10 nanofarad = 100 microsec = 0,1 milisec = 0,001 s
//tau = 5 kOhm * 10 mikrofarad = 50 milisec
//Wir wollen aber 5 tau haben , also 5*50 = 250 ms

//T = 5*tau*2 = 5*50*2 = 500 milisec
//f = 1/s = Hz
//T = 1/frequenz = s

//f = 1/0,001s = 1000 Hz

DigitalOut rectangle(PC_0);

int main()

{
        float frequency = 1000;
        float hoch = (1/frequency)/2;

while(1) {
    rectangle = 1;
    wait(hoch);
    rectangle = 0;
    wait(hoch);
  }
}
```



Fügen Sie hier einen Screenshot des LEO ein in dem das Rechtecksignal am Eingang U1 und die Spannung am Kondensator C – U2 sichtbar sind. Fügen Sie nun die Cursor hinzu und messen Sie die Zeitkonstante. Das Bild sollte keinesfalls mehr als 1 Periode des Rechtecksignals zeigen! Der Wert der Zeitkonstante soll als Differenz der beiden Cursor im Screenshot ablesbar sein.

- Den ersten Vertikalen bei 0 setzen
- Den Horizontalen auf 63%
- und DANN erst den zweiten Vertikalen auf den Schnittpunkt des horizontalen mit der Ladekurve
- Der ist dann genau bei Tau





- Gemessene Zeitkonsante 50ms (siehe dt und die roten und blauen pfeile oben)

3) PWM

Erzeugen Sie drei PWM-Signale mit unterschiedlichen Pulsweiten. Realisierung mbed 9 Punkte oder Realisierung Cube&Keil 12 Punkte! Frequenz: 7khz_____ a) PWM1 Pulsweite=35%___ b) PWM2 Pulsweite= 50%_____ c) PWM3 Pulsweite= 65%_____

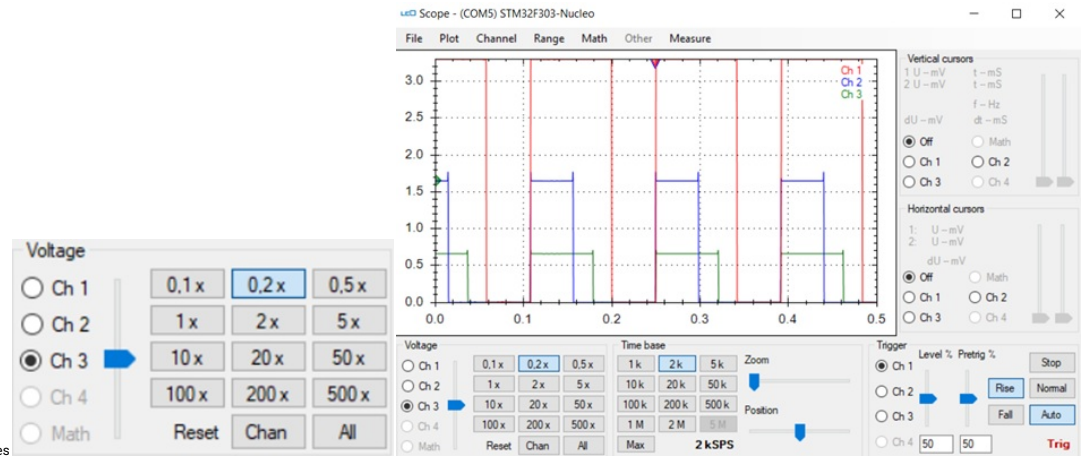
Fügen Sie hier einen Screenshot des LEO ein in dem die 3 PWM Signale gleichzeitig sichtbar sind. Das Bild sollte 2-3 Perioden zeigen!

Zuerst muss die Periode festgestellt werden:

- 1/7 = 0,142 sec = 142 milisec

PWM1 Pulsweite = 35% — > 0,35 * 142 = 49,7 PWM2 Pulsweite = 50% — > 0,50 * 142 = 71 PWM3 Pulsweite = 65% — > 0,65 * 142 = 92,3

```
#include "mbed.h"

// Von L476 D11 zum LEO A5, D10 zum A4, D9 zum A3

PwmOut mypwm(D10);
PwmOut mypwm2(D11);
PwmOut mypwm3(D9);

int main() {

mypwm.period_ms(142);       // = 1/7khZ = 0,142 sec = 142 milisec
mypwm.pulsewidth_ms(49.7);    //35% of the full signal (142 msec)
mypwm2.period_ms(142);       //
mypwm2.pulsewidth_ms(71);    //50% von period_ms
mypwm3.period_ms(142);       //
mypwm3.pulsewidth_ms(92.3);    //75% von period_ms

//printf("pwm set to %.2f %%\n", mypwm.read() * 100);
}

//beim Generator mit diesen Spezifikationen
//Square, 10Hz mit 50% Duty - 10 Hz sollten 100ms
```

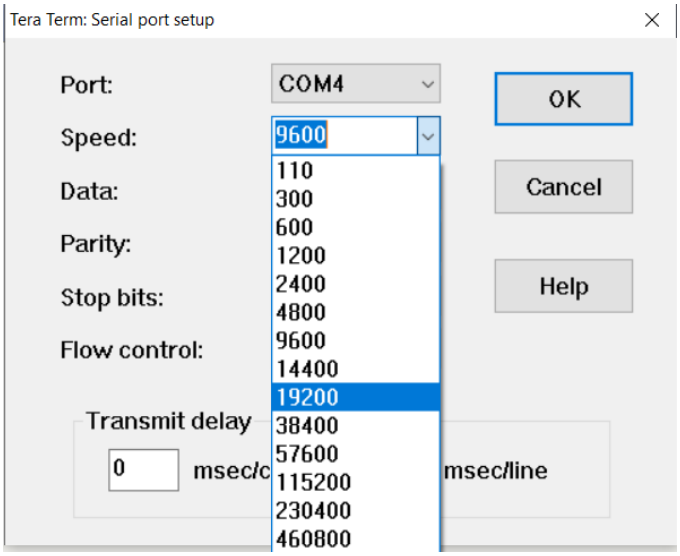
On Ch 1, Ch 2 and Ch 3 change these Voltages 

1) UART – Beispiel 1

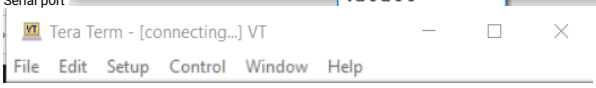
Erzeugen Sie mit einem Terminalprogramm (Teraterm) ein Signal am virtuellen COM-Port des ST-Link. Senden Sie das Zeichen n Mit einer Datenrate: 19200

WICHTIG: Datenrate

- Im Code einfach 9600 auf 19200 ändern
- Auf TeraTerm nach Starten in Einstellungen auf 19200 ändern
  - Setup

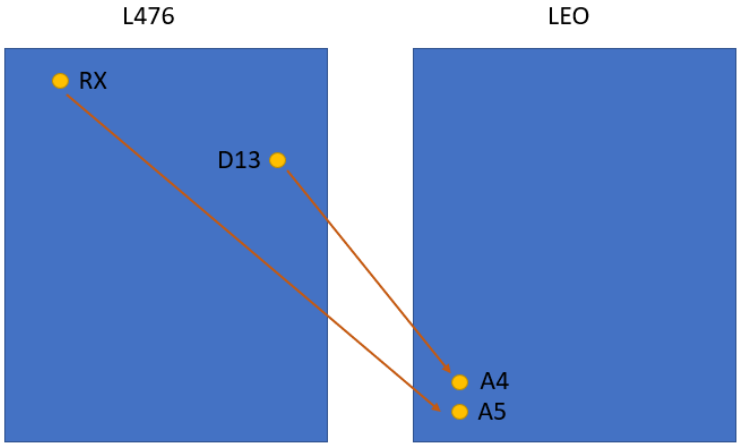


- Serial port



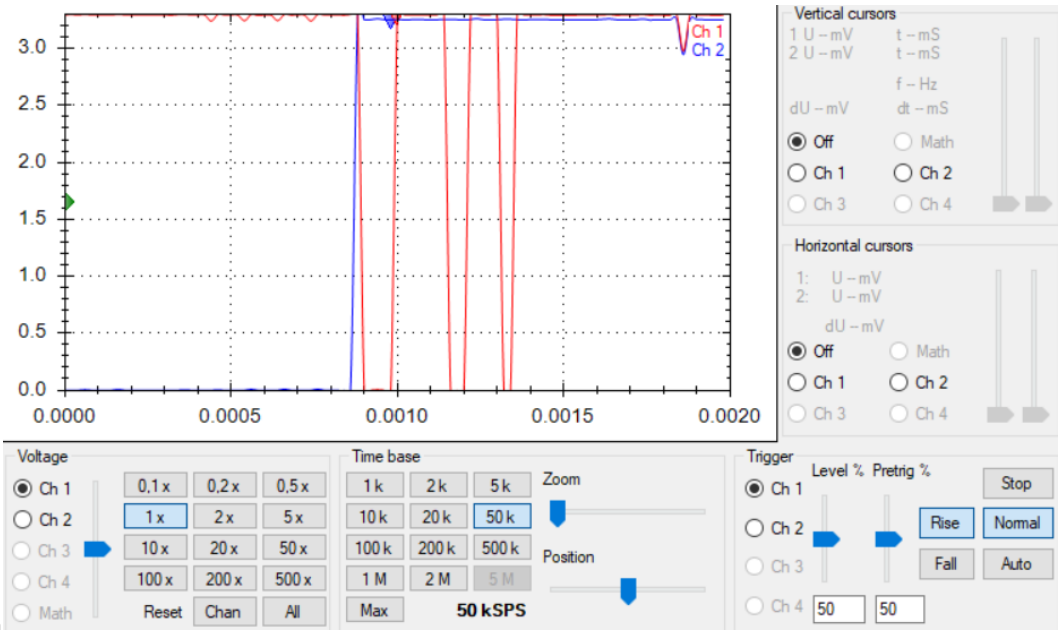
Jedes mal wenn dieses Zeichen empfangen wird soll die grüne USER-Led des STM32L476 ihren Zustand ändern.

Fügen Sie hier einen Screenshot des LEO ein in dem das Bitmuster für das empfangene Zeichen am Tx-Anschluss des ST-Link am Kanal Ch1 sichtbar ist. Am Kanal Ch2 soll sichtbar sein wie sich der Zustand des Pins PA_5 (User Led)

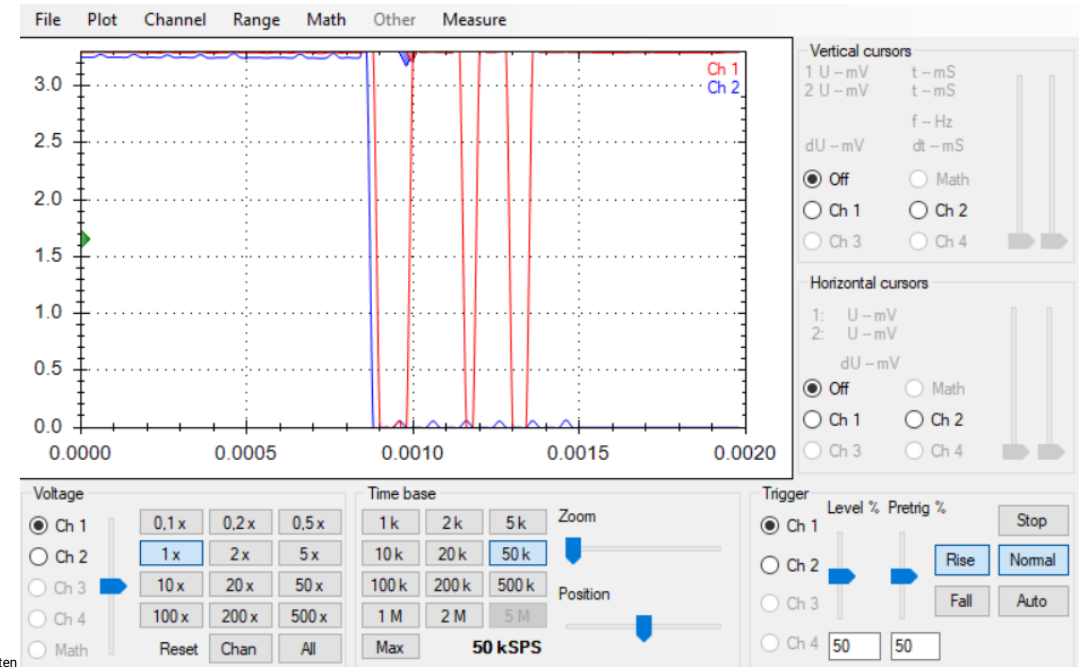


ändert unmittelbar nachdem das Zeichen empfangen wurde.

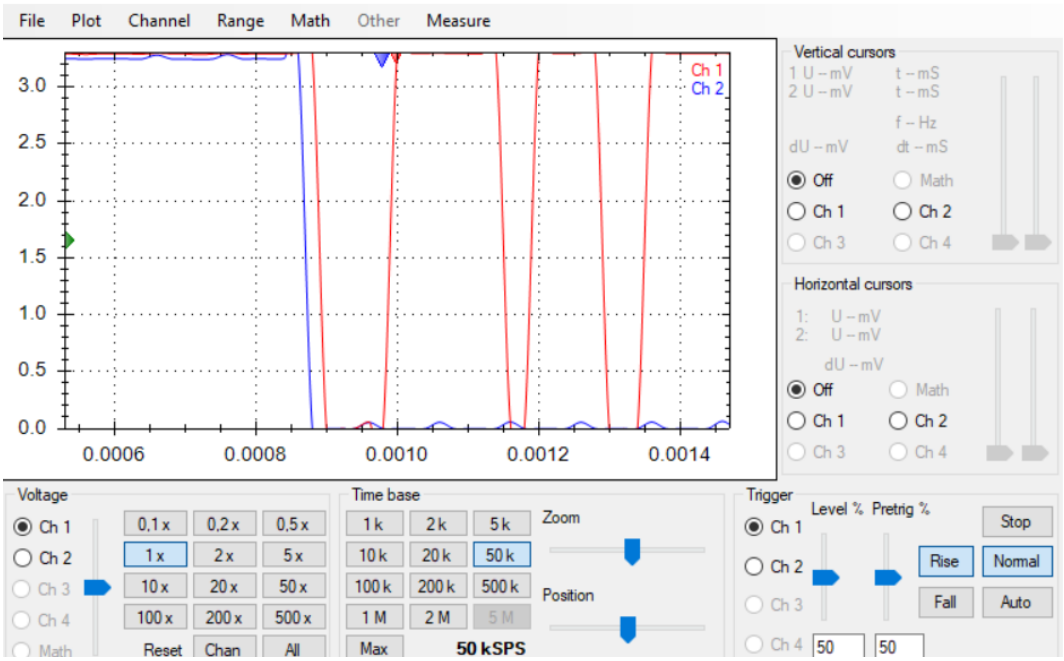
```
#include "mbed.h"

//Von L476 RX-pin zum A5 von LEO
//Von L476 D13 zum A4 von LEO

Serial pc(SERIAL_TX, SERIAL_RX, 19200);
DigitalOut myled(PA_5);//D13

int main()
{
    char userINPUT;

while(1) {
    //10 bit mit start und stopbit -> 1ms
    userINPUT = pc.getc(); //tx-micro
    if (userINPUT == 'n') myled=!myled;
    pc.printf("%c",userINPUT); //rx-micro

  }
}
```



LED on nach ‘n’ drücken: * blau (LED) unten -> oben



LED off nach ‘n’ erneut drücken * blau (LED) oben -> unten



Das empfangene Bitmuster in hexadezimaler Schreibweise lautet:

0110 1110

0x6E

received = read from right to left transmitted = vice versa

2) UART – Beispiel 2

Erzeugen Sie mit einem Terminalprogramm (TeraTerm) ein Signal am virtuellen COM-Port des ST-Link. Jedes mal wenn Sie im Terminalprogramm dieses Zeichen senden wird vom ST-Link das Bitmuster am Tx-Pin gesendet!

Senden Sie das Zeichen p

Mit einer Datenrate: 19200

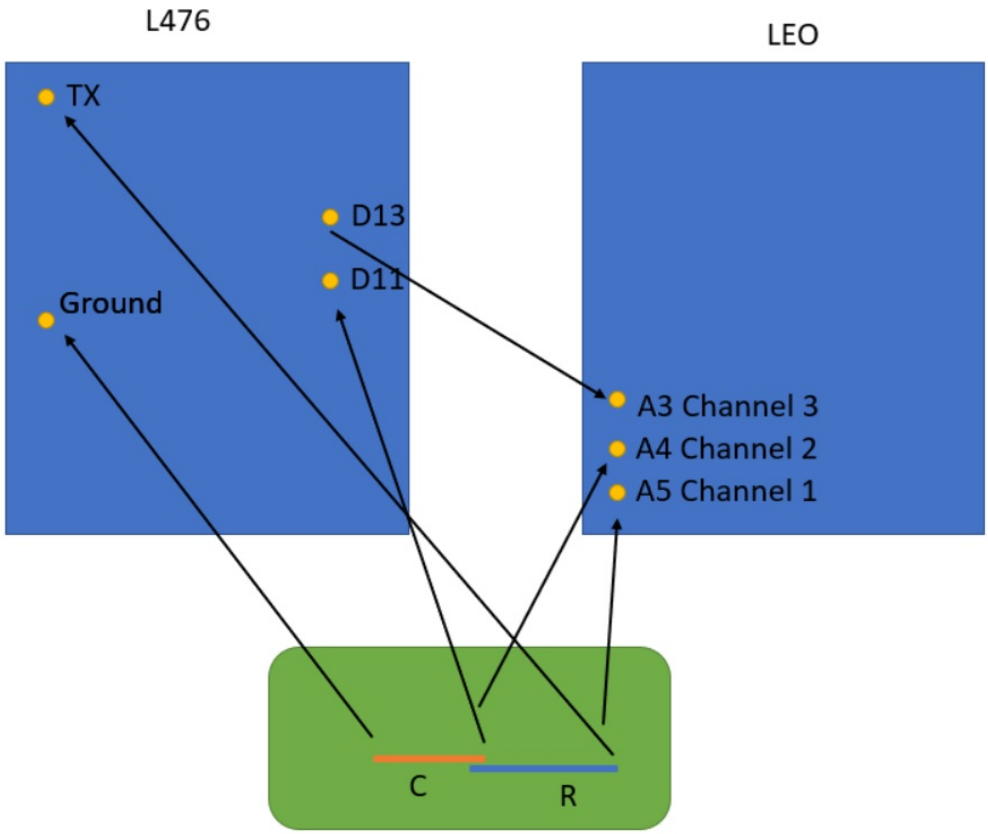
- Channel 1
  - Signal des Tx-Pins vom ST-Link am Eingang des RC-Tiefpasses
  - Senden 'n'
  - Bitmuster für das gesendete Zeichen am Eingang U1 des RC-Tiefpasses (=Tx-Pin des ST-Link)
- Channel 2
  - Digital-In - D7
  - 1 or 0
  - Bitmuster am Ausgang U2 (am Kondensator C) des RC-Tiefpasses
- Channel 3
  - Digital-Out - D13
  - Wenn din=1, dann dout=1
  - Wenn din=0, dann dout=0

* Signal am digitalen Output Pin

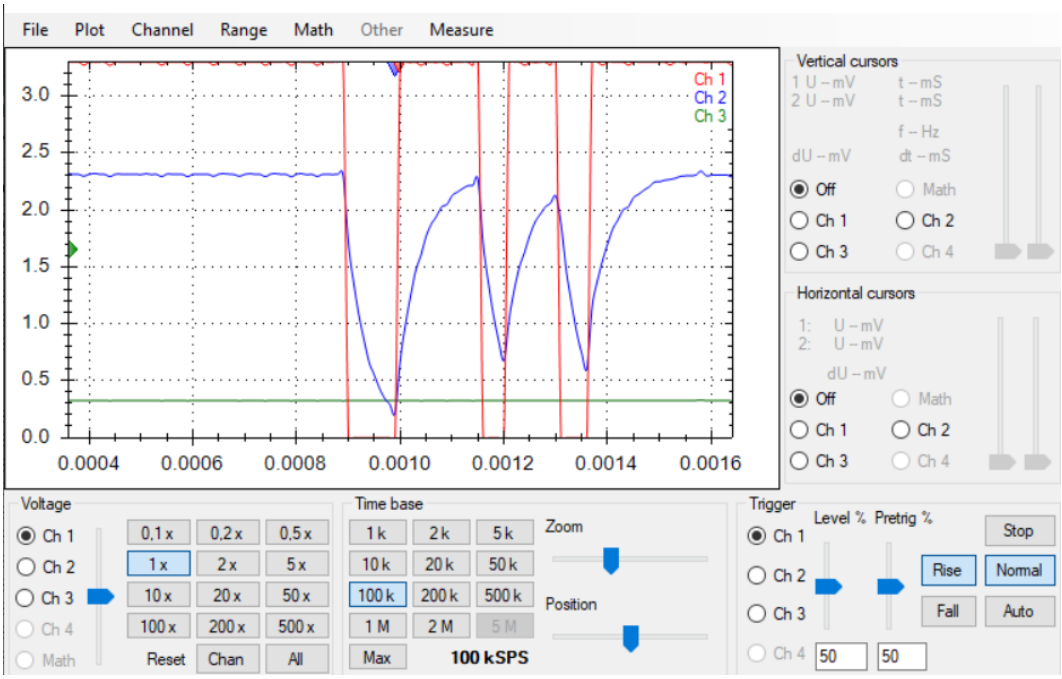
- Das Signal des Tx-Pins vom ST-Link schließen Sie nun am Eingang des RCTiefpasses an.
- Konfigurieren Sie nun einen digitalen Input Pin und schließen Sie diesen am Ausgang des RC-Tiefpasses an.
  - Wird am digitalen Input Pin eine „1" erkannt so soll ein digitaler Output Pin des STM32L476 ebenfalls auf 1 gesetzt werden.
  - Wird am digitalen Input Pin eine „0" erkannt so soll ein digitaler Output Pin des STM32L476 ebenfalls auf 0 gesetzt werden.
  - Der digitale Output Pin soll also den logischen Zustand ausgeben der am Input Pin erfasst/erkannt wird!

Fügen Sie hier einen Screenshot des LEO ein in dem: das Bitmuster für das gesendete Zeichen am Eingang U1 des RC-Tiefpasses (=Tx-Pin des ST-Link) und am Ausgang U2 (am Kondensator C) des RC-Tiefpasses zeigt.

Weiters soll der dritten Kanal das Signal am digitalen Output Pin zeigen. Realisierung mbed 7 Punkte oder Realisierung Cube&Keil 12 Punkte!


L476
LEO
TX
D13
D11
Ground
A3 Channel 3
A4 Channel 2
A5 Channel 1
C
R

```
#include "mbed.h"

//Von L476 RX-pin zum Widerstand und dort ein Kabel zu A5 von LEO
//Von L476 D10 zwischen C und R anstecken und dann ein Kabel zum A4 von LEO
//Von L476 D13 direkt zu A3 von LEO
//Von L476 Ground mit Kabel zu C

//A5 = Ch1 = DigitalOut D13
//A4 = Ch2 = DigitalIn D10
//A3 = Ch3 = D13

Serial pc(SERIAL_TX, SERIAL_RX, 19200);
DigitalOut dout(D13);
DigitalIn din(D10);

int main()
{
    //char userINPUT;

while(1) {
    pc.printf("n");
    dout = din;
    wait_ms(250);

    //userINPUT = pc.getc();
    //if (userINPUT == 'n') dout=din;
    //pc.printf("%c",userINPUT);
   }
}

/*

ASCII;Bin;Hex;Dec;;ASCII;Bin;Hex;Dec
@;[0]1000000;40;64;;';[0]1100000;60;96
A;[0]1000001;41;65;;a;[0]1100001;61;97
B;[0]1000010;42;66;;b;[0]1100010;62;98
C;[0]1000011;43;67;;c;[0]1100011;63;99
D;[0]1000100;44;68;;d;[0]1100100;64;100
E;[0]1000101;45;69;;e;[0]1100101;65;101
F;[0]1000110;46;70;;f;[0]1100110;66;102
G;[0]1000111;47;71;;g;[0]1100111;67;103
H;[0]1001000;48;72;;h;[0]1101000;68;104
I;[0]1001001;49;73;;i;[0]1101001;69;105
J;[0]1001010;4A;74;;j;[0]1101010;6A;106
K;[0]1001011;4B;75;;k;[0]1101011;6B;107
L;[0]1001100;4C;76;;l;[0]1101100;6C;108
M;[0]1001101;4D;77;;m;[0]1101101;6D;109
N;[0]1001110;4E;78;;n;[0]1101110;6E;110
O;[0]1001111;4F;79;;o;[0]1101111;6F;111
P;[0]1010000;50;80;;p;[0]1110000;70;112
Q;[0]1010001;51;81;;q;[0]1110001;71;113
R;[0]1010010;52;82;;r;[0]1110010;72;114
S;[0]1010011;53;83;;s;[0]1110011;73;115
T;[0]1010100;54;84;;t;[0]1110100;74;116
U;[0]1010101;55;85;;u;[0]1110101;75;117
V;[0]1010110;56;86;;v;[0]1110110;76;118
W;[0]1010111;57;87;;w;[0]1110111;77;119
X;[0]1011000;58;88;;x;[0]1111000;78;120
Y;[0]1011001;59;89;;y;[0]1111001;79;121
Z;[0]1011010;5A;90;;z;[0]1111010;7A;122
ASCII;Bin;Hex;Dec;;ASCII;Bin;Hex;Dec
[;[0]1011011;5B;91;;{;[0]1111011;7B;123
\;[0]1011100;5C;92;;|;[0]1111100;7C;124
];[0]1011101;5D;93;;};[0]1111101;7D;125
^;[0]1011110;5E;94;;~;[0]1111110;7E;126
_;[0]1011111;5F;95;;DEL;[0]1111111;7F;127
ASCII;Bin;Hex;Dec;;ASCII;Bin;Hex;Dec
0;[0]0110000;30;48;;SP;[0]0100000;20;32
1;[0]0110001;31;49;;!;[0]0100001;21;33
2;[0]0110010;32;50;;"""";[0]0100010;22;34
3;[0]0110011;33;51;;#;[0]0100011;23;35
4;[0]0110100;34;52;;$;[0]0100100;24;36
5;[0]0110101;35;53;;%;[0]0100101;25;37
6;[0]0110110;36;54;;&;[0]0100110;26;38
7;[0]0110111;37;55;;';[0]0100111;27;39
8;[0]0111000;38;56;;(;[0]0101000;28;40
9;[0]0111001;39;57;;);[0]0101001;29;41
:;[0]0111010;3A;58;;*;[0]0101010;2A;42
";";[0]0111011;3B;59;;+;[0]0101011;2B;43
<;[0]0111100;3C;60;;,;[0]0101100;2C;44
=;[0]0111101;3D;61;;-;[0]0101101;2D;45
>;[0]0111110;3E;62;;.;[0]0101110;2E;46
?;[0]0111111;3F;63;;/;[0]0101111;2F;47

*/
```



Das am digitalen Output Pin sichtbare Bitmuster in hexadezimaler Schreibweise lautet: 0x0

Der dritte Channel zeigt konstant 0V an (je nach C und R auch konstant 3,3V), da ich keine gescheiten R bzw C habe, die proportionell so gut zusammenpassen würden, dass man einen schönen Output kriegt. Das grüne Signal sollte auch ein Rechteckssignal sein.