

# Echtzeitverarbeitung 1 Georgios Markou

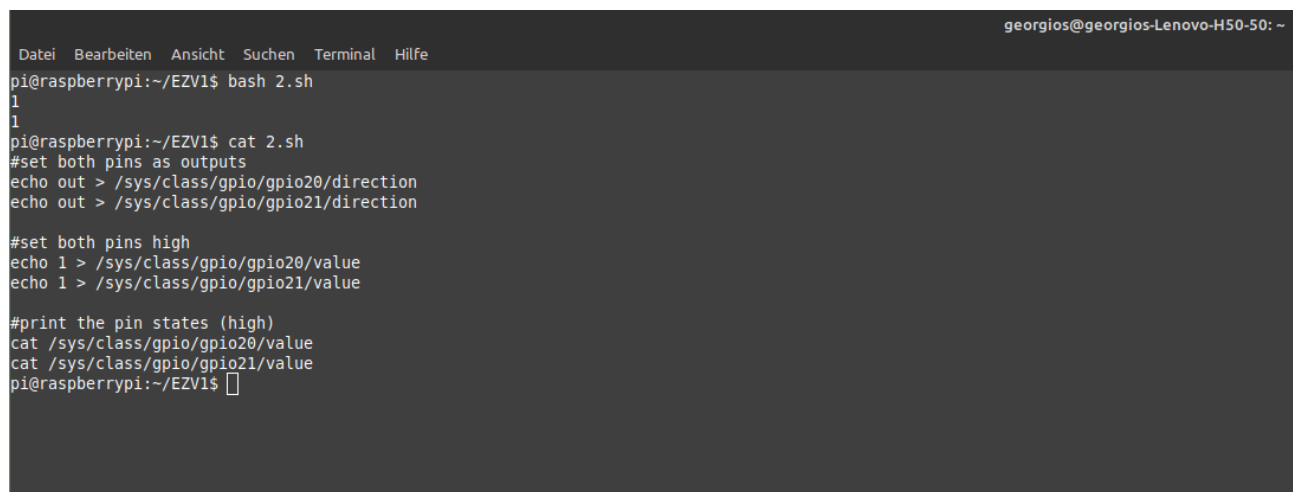
1)

Hinweis: Aufgaben mit dem Suffix „\_2“ sind die neuen Lösungen. Ich habe die alten Messungen trotzdem nicht entfernt um Unterschiede feststellen zu können.

2)

Ich verwende Linux Ubuntu 20 und gebe “sudo picocom /dev/ttyUSB0“ ein. Nach kurzer Zeit gebe ich den Benutzernamen und das Passwort ein.

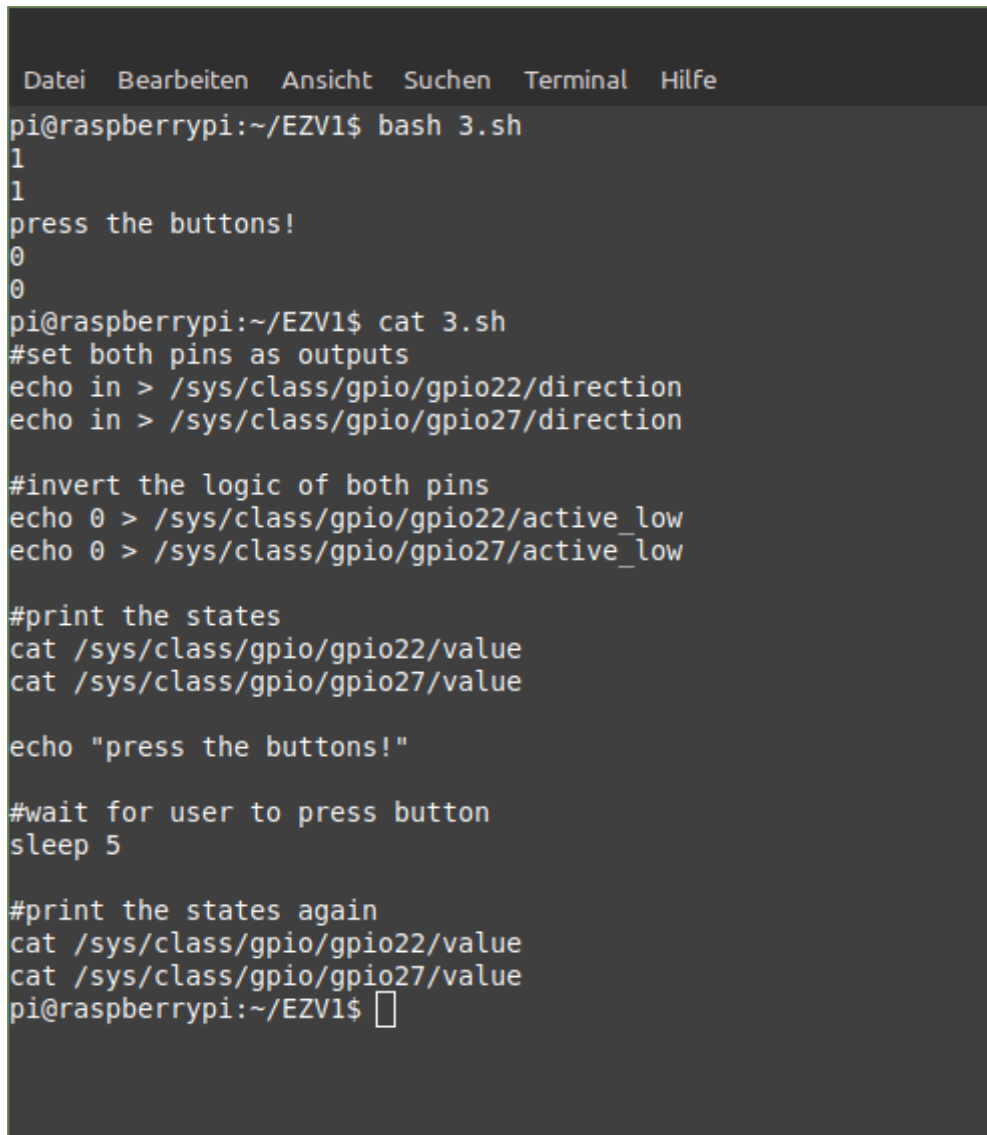
3)



```
georgios@georgios-Lenovo-H50-50: ~  
Datei Bearbeiten Ansicht Suchen Terminal Hilfe  
pi@raspberrypi:~/EZV1$ bash 2.sh  
1  
1  
pi@raspberrypi:~/EZV1$ cat 2.sh  
#set both pins as outputs  
echo out > /sys/class/gpio/gpio20/direction  
echo out > /sys/class/gpio/gpio21/direction  
  
#set both pins high  
echo 1 > /sys/class/gpio/gpio20/value  
echo 1 > /sys/class/gpio/gpio21/value  
  
#print the pin states (high)  
cat /sys/class/gpio/gpio20/value  
cat /sys/class/gpio/gpio21/value  
pi@raspberrypi:~/EZV1$
```

Wie man erkennt habe ich mich eingeloggt und das Skript “2.sh“ mit dem Kommando “bash 2.sh“ ausgeführt. Das Skript gibt zwei mal “1“ aus was bedeutet, dass beide Pin “HIGH“ sind (die LEDs leuchten nach den 2 Befehlen nach dem Kommentar “set both pins high“). Der nächste Befehl “cat 2.sh“ gibt mein Skript aus.

4)



```
Datei Bearbeiten Ansicht Suchen Terminal Hilfe
pi@raspberrypi:~/EZV1$ bash 3.sh
1
1
press the buttons!
0
0
pi@raspberrypi:~/EZV1$ cat 3.sh
#set both pins as outputs
echo in > /sys/class/gpio/gpio22/direction
echo in > /sys/class/gpio/gpio27/direction

#invert the logic of both pins
echo 0 > /sys/class/gpio/gpio22/active_low
echo 0 > /sys/class/gpio/gpio27/active_low

#print the states
cat /sys/class/gpio/gpio22/value
cat /sys/class/gpio/gpio27/value

echo "press the buttons!"

#wait for user to press button
sleep 5

#print the states again
cat /sys/class/gpio/gpio22/value
cat /sys/class/gpio/gpio27/value
pi@raspberrypi:~/EZV1$
```

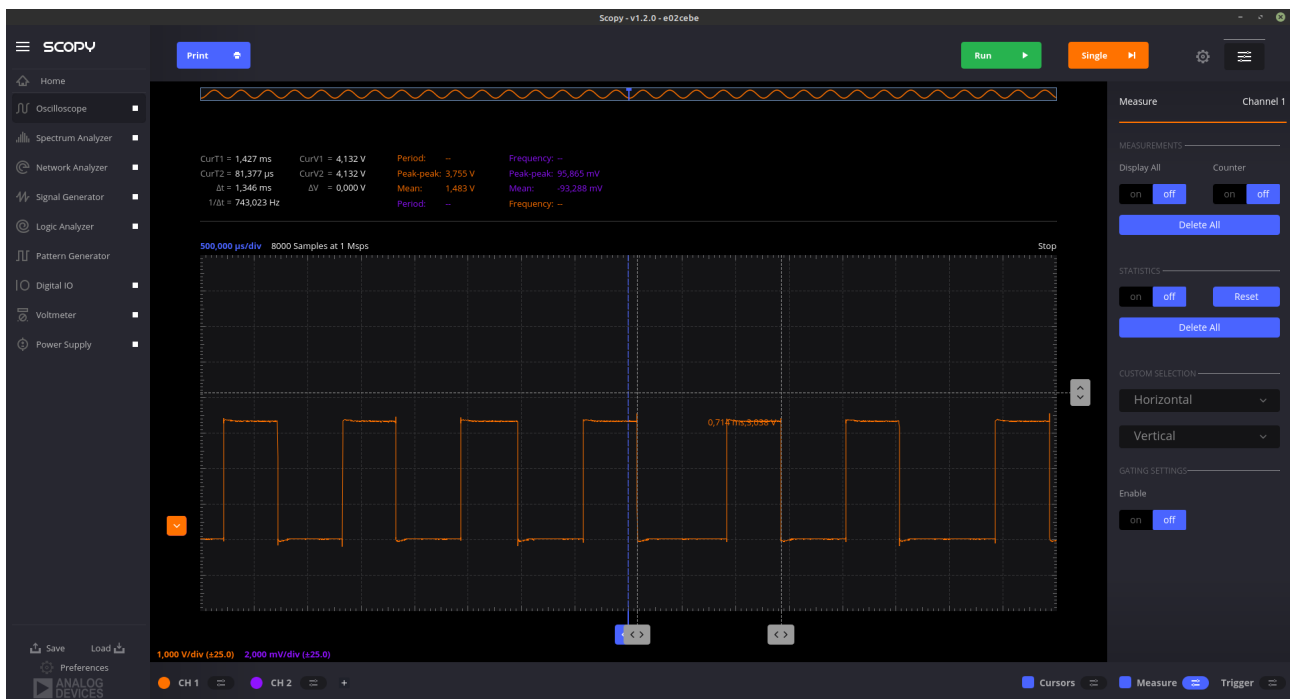
Wie man erkennt habe ich das Skript “3.sh“ ausgeführt. Es zeigt den Zustand der beiden Taster an (1 = nicht-gedrückt). Man wird dann aufgefordert die beiden Taster zu drücken und sieht wenn sie gedrückt werden ist die Ausgabe “low“ (0 = Taster gedrückt). Mit dem nächsten Befehl zeige ich den Inhalt meines Skripts an. Die Kommentare erklären die einzelnen Funktionen.

5)

```
Datei Bearbeiten Ansicht Suchen Terminal Hilfe
pi@raspberrypi:~/EZV1$ bash 4.sh
^C
pi@raspberrypi:~/EZV1$ cat 4.sh
#set pin as outputs
echo out > /sys/class/gpio/gpio21/direction

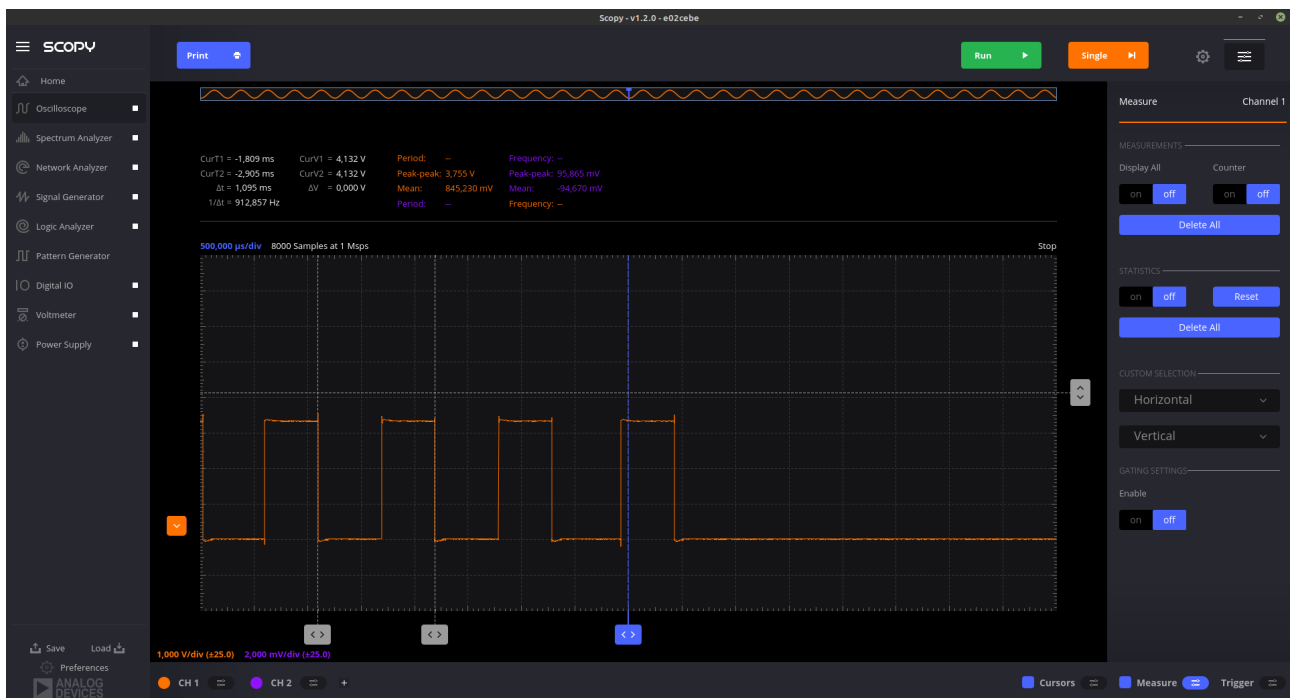
while true
do
    echo 1 > /sys/class/gpio/gpio21/value
    echo 0 > /sys/class/gpio/gpio21/value
done
pi@raspberrypi:~/EZV1$
```

Das Vorgehen ist gleich wie in 3) und 4).



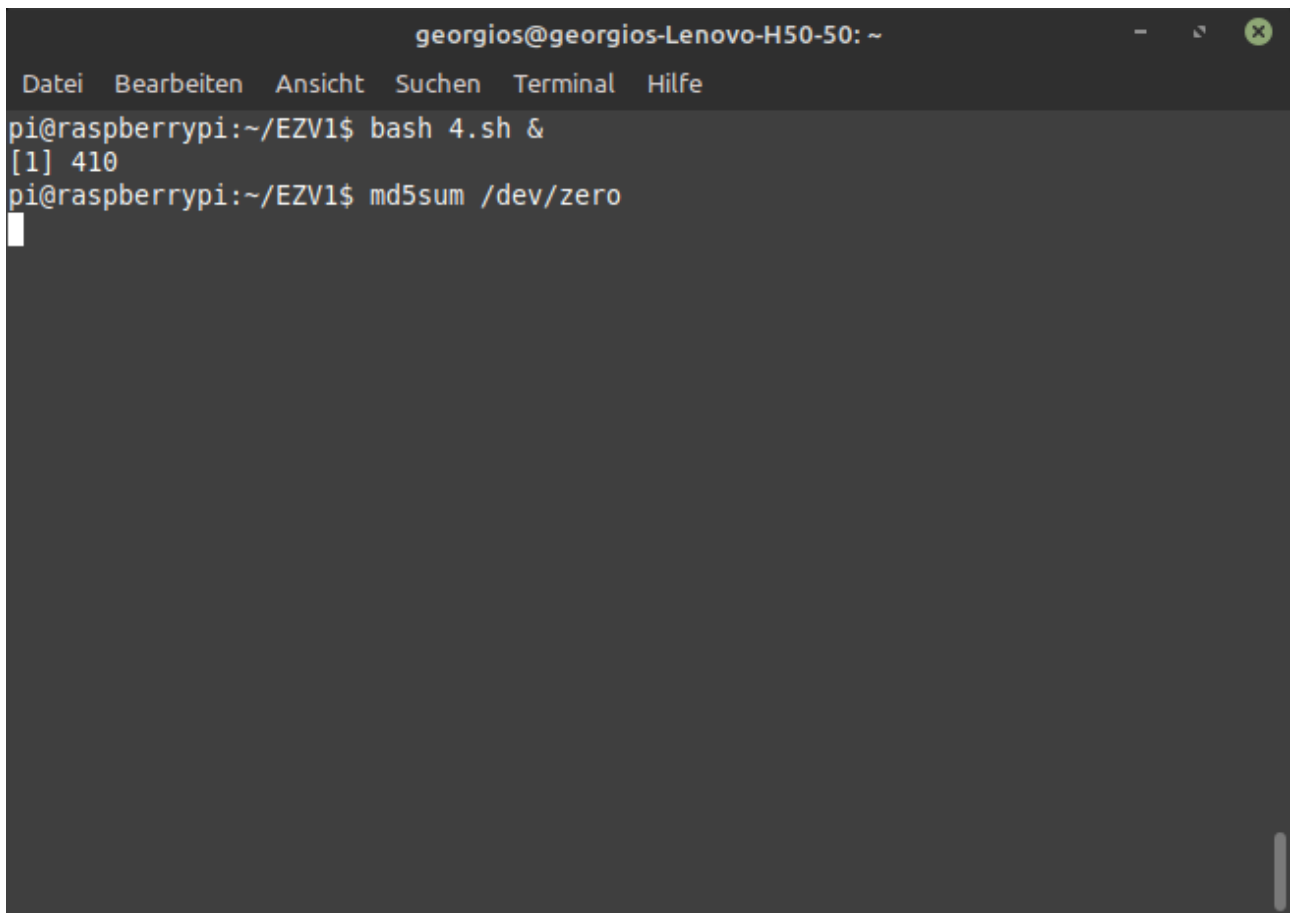


Diese 2 Schnapschüsse von Scopy zeigen, dass die Frequenz schwankt (1. Bild ca. 743 Hz; 2. Bild 934 Hz). Bei mir gab es selten Aussetzer (keine anderen Programme wurden ausgeführt). Die Aussetzer kommen vom Betriebssystem (Scheduler), weil mehrere Ressourcen gleichzeitig verwaltet werden müssen.



In diesem Schnapschuss sieht man, dass es Aussetzer gab. Die gemessene Frequenz war ca. 913 Hz jedoch sieht man auf der rechten Seite dass es keine steigenden Flanken gibt. Das System hat im Hintergrund das Bash-Skript ausgeführt, während es im Vordergrund ein anderes Programm

ausgeführt hat. Man sieht, dass das Betriebssystem (Scheduler) Zeit braucht und eine gewisse Latenz erzeugt.

A terminal window titled 'georgios@georgios-Lenovo-H50-50: ~' with a menu bar (Datei, Bearbeiten, Ansicht, Suchen, Terminal, Hilfe). The prompt is 'pi@raspberrypi:~/EZV1\$'. The first command is 'bash 4.sh &', which returns '[1] 410'. The second command is 'md5sum /dev/zero', followed by a blank line, indicating a delay in execution.

```
georgios@georgios-Lenovo-H50-50: ~
Datei Bearbeiten Ansicht Suchen Terminal Hilfe
pi@raspberrypi:~/EZV1$ bash 4.sh &
[1] 410
pi@raspberrypi:~/EZV1$ md5sum /dev/zero

```

Mit dem ersten Befehl wurde ein neuer Prozess gestartet der “4.sh“ ausführt. Gleichzeitig wurde auch “md5sum /dev/zero“ ausgeführt.

5\_2)

Gleiches Vorgehen wie bei 5). Anbei sind die aktuellen Messungen.

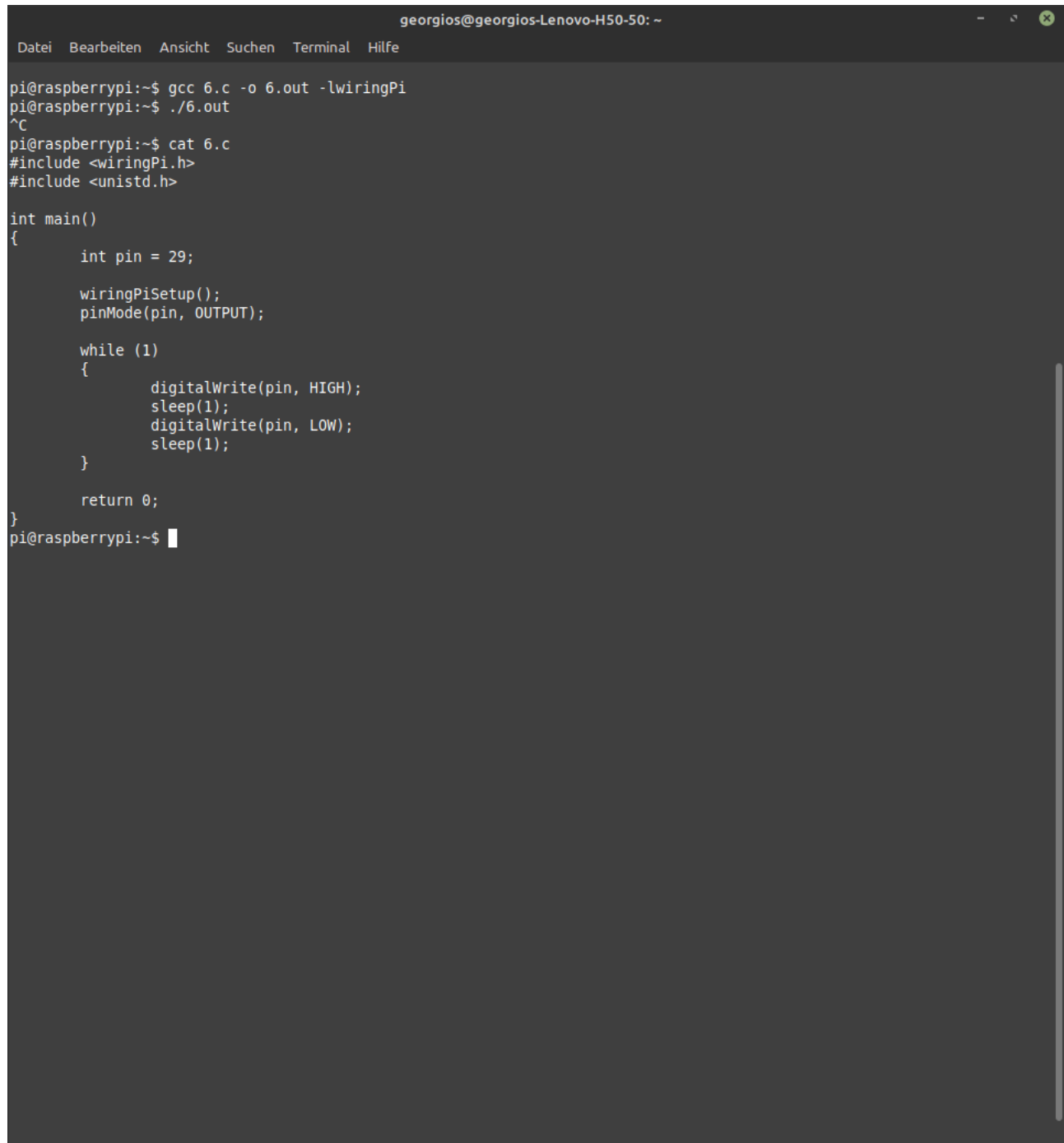
Die nächsten beiden Bilder zeigen die Messungen ohne Last. Wie man sieht sind die Unterschiede zur alten Abgabe minimal.



Die nächsten 2 Bilder zeigen die Messungen mit Last. Es werden höhere Frequenzen erreicht. Die Aussetzer sind trotzdem vorhanden.



6)

A terminal window titled 'georgios@georgios-Lenovo-H50-50: ~' with a menu bar containing 'Datei', 'Bearbeiten', 'Ansicht', 'Suchen', 'Terminal', and 'Hilfe'. The terminal shows the following commands and output:

```
pi@raspberrypi:~$ gcc 6.c -o 6.out -lwiringPi
pi@raspberrypi:~$ ./6.out
^C
pi@raspberrypi:~$ cat 6.c
#include <wiringPi.h>
#include <unistd.h>

int main()
{
    int pin = 29;

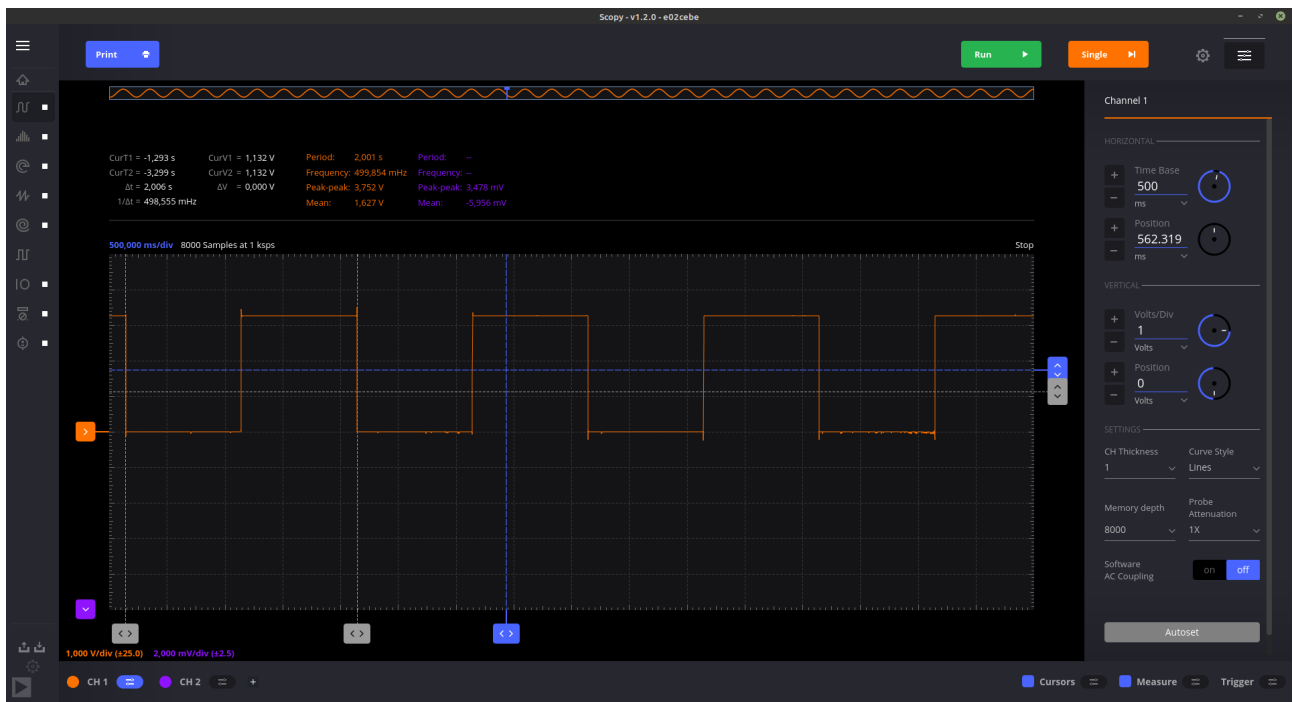
    wiringPiSetup();
    pinMode(pin, OUTPUT);

    while (1)
    {
        digitalWrite(pin, HIGH);
        sleep(1);
        digitalWrite(pin, LOW);
        sleep(1);
    }

    return 0;
}
pi@raspberrypi:~$
```

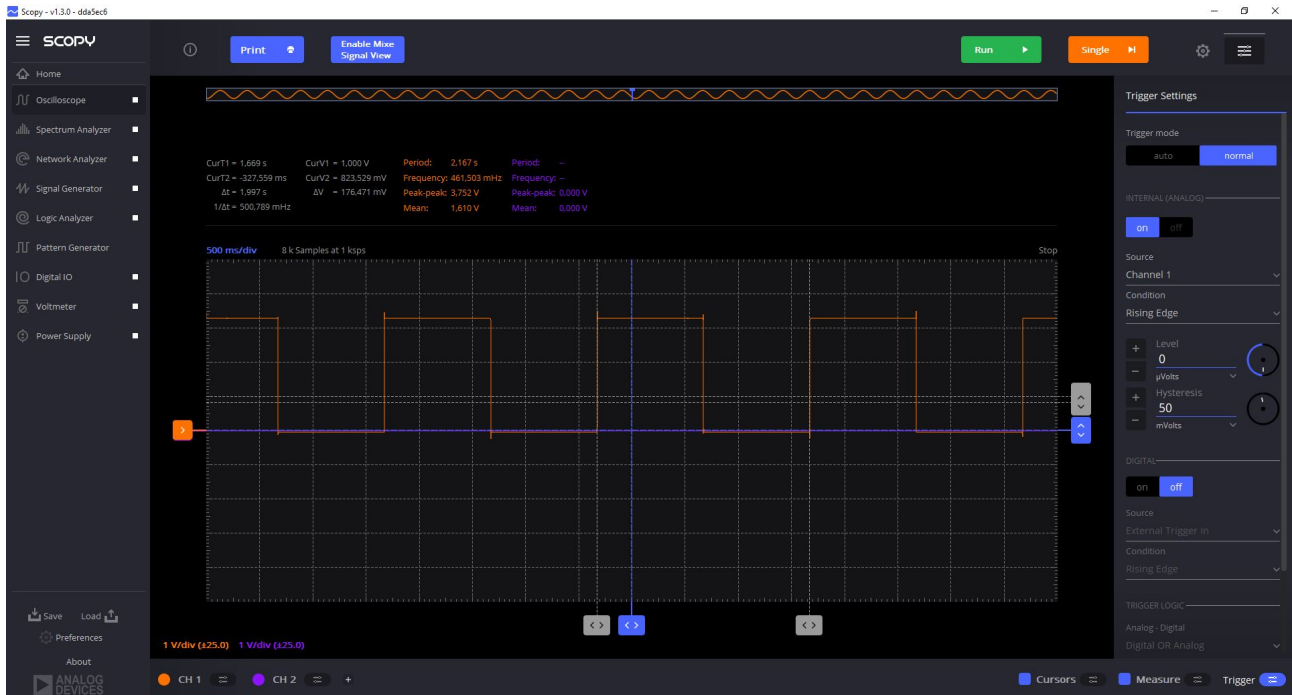
Mit dem ersten Befehl führe ich das Programm “6.out“ aus und mit dem nächsten Befehl gebe ich den C-Code, der ausgeführt wurde, aus.





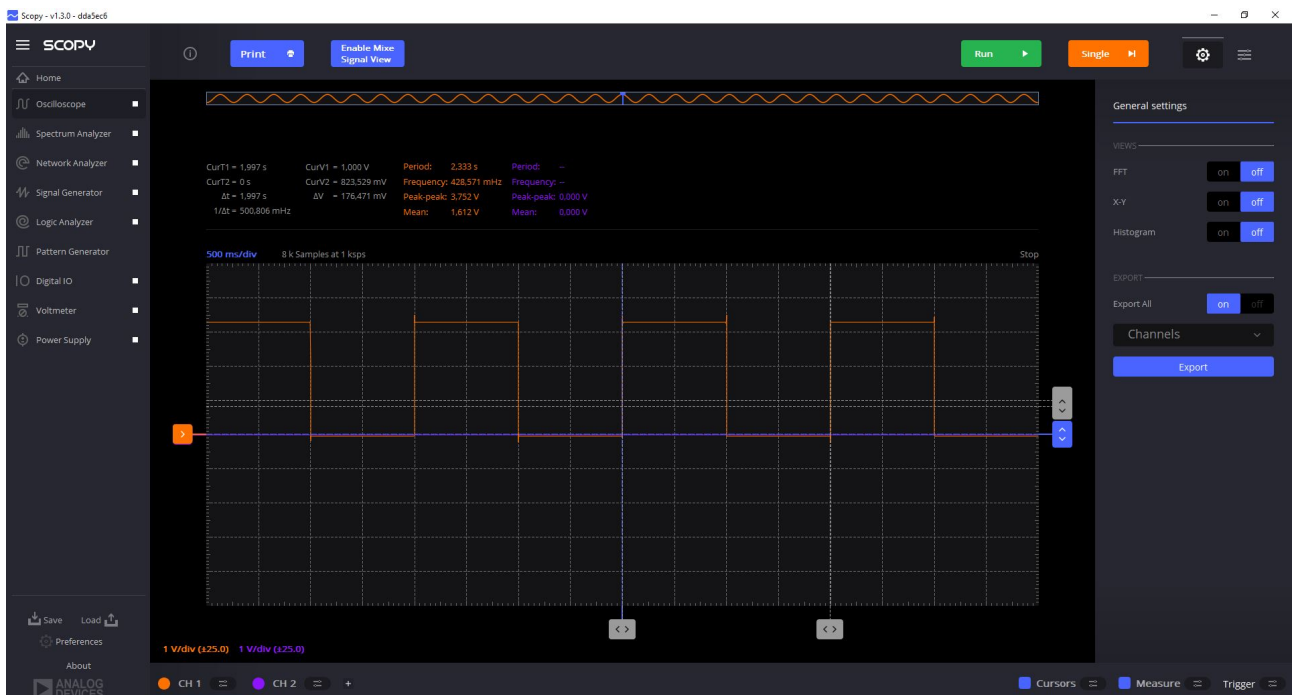
Wie man erkennt funktioniert das Programm zuverlässig und die Frequenz ist immer ca. 500 Hz

6\_2) Gleiches Verfahren wie bei 6). Das erzeugte Signal ist bei beiden Abgaben zuverlässig.



Das nächste Bild zeigt die gleiche Aufgabe unter Last (wie immer “md5sum /dev/zero“).

Die Frequenz ist trotzdem stabil.



7)

```
Datei Bearbeiten Ansicht Suchen Terminal Hilfe
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0^C
pi@raspberrypi:~$ cat 7.c
#include <wiringPi.h>
#include <stdio.h>

int main()
{
    int pin = 2;

    wiringPiSetup();
    pinMode(pin, INPUT);

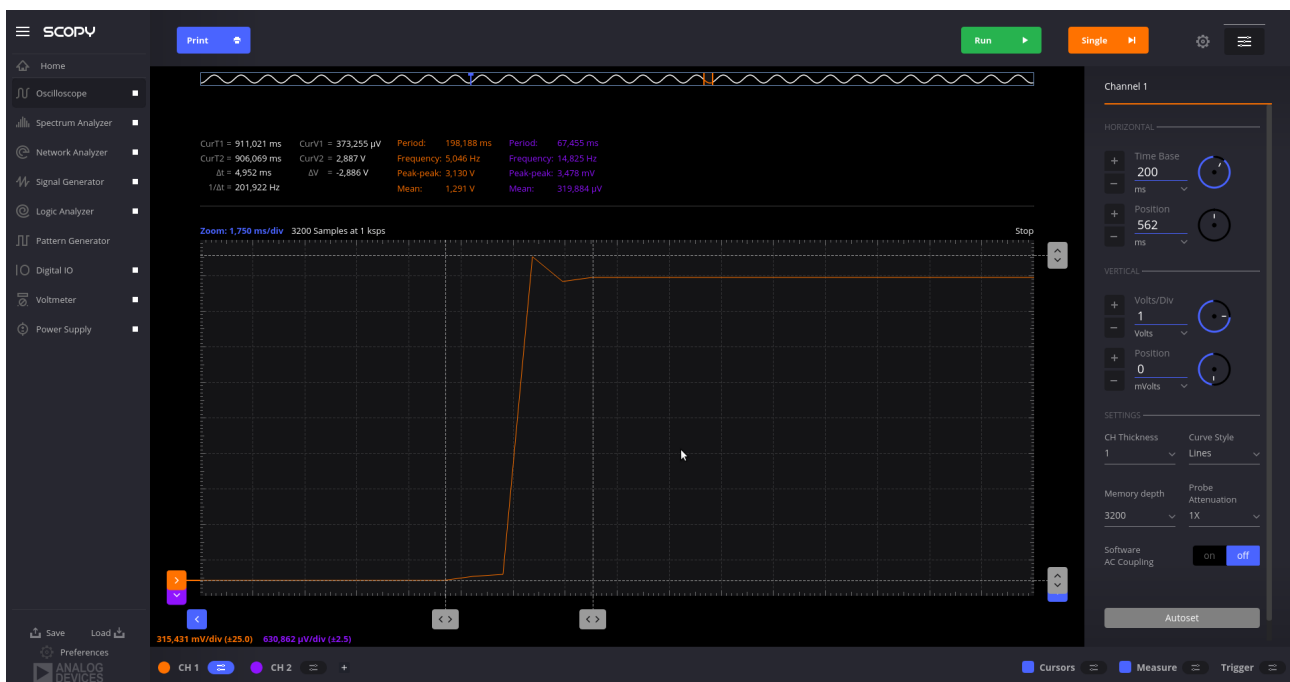
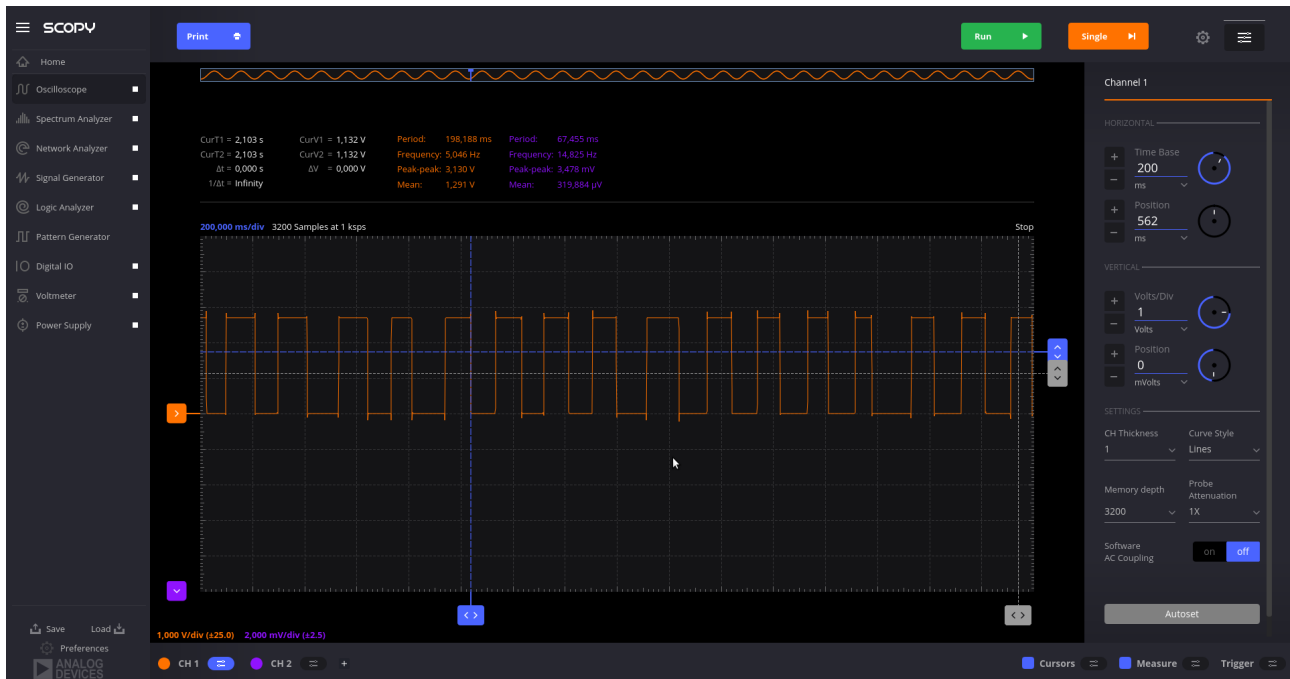
    while (1)
    {
        if (digitalRead(pin) == LOW)
        {
            printf("1\n");
        }

        else
        {
            printf("0\n");
        }

        delay(1);
    }

    return 0;
}
pi@raspberrypi:~$
```

Die ersten Zeilen "0" sind die Ausgaben vom Programm "7.out". Mit dem Befehl "cat 7.c" wird der C-Code des Programms angezeigt.



Im ersten Bild sieht man das Signal wenn ich den Taster sehr schnell betätige. Im zweiten Bild habe ich mir eine zufällige steigende Flanke ausgesucht. Entgegen meinen Erwartungen hat der Taster nicht geprellt.

7\_2)

Ich habe das Program geändert. Das neue Programm ist folgendes.

```

pi@raspberrypi:~/l$ cat 7.c
#include <wiringPi.h>
#include <stdio.h>

int main()
{
    int pin = 2;

    wiringPiSetup();
    pinMode(pin, INPUT);

    int oldStatus = digitalRead(pin);
    int newStatus = oldStatus;
    int counter = 0;

    while (1)
    {
        if (oldStatus != newStatus)
        {
            if (oldStatus == 0)
            {
                puts("1 -> 0");
            }

            else
            {
                puts("0 -> 1");
            }

            oldStatus = newStatus;
        }

        newStatus = digitalRead(pin);
    }

    return 0;
}
pi@raspberrypi:~/l$ ./7.out
0 -> 1
1 -> 0
0 -> 1
1 -> 0
0 -> 1
1 -> 0
0 -> 1
1 -> 0
0 -> 1
1 -> 0
0 -> 1
1 -> 0
0 -> 1
1 -> 0

```

Leider kann ich kein Prellen erkennen. Der Tastendruck wird zuverlässig erkannt, genauso wie bei der alten Abgabe.