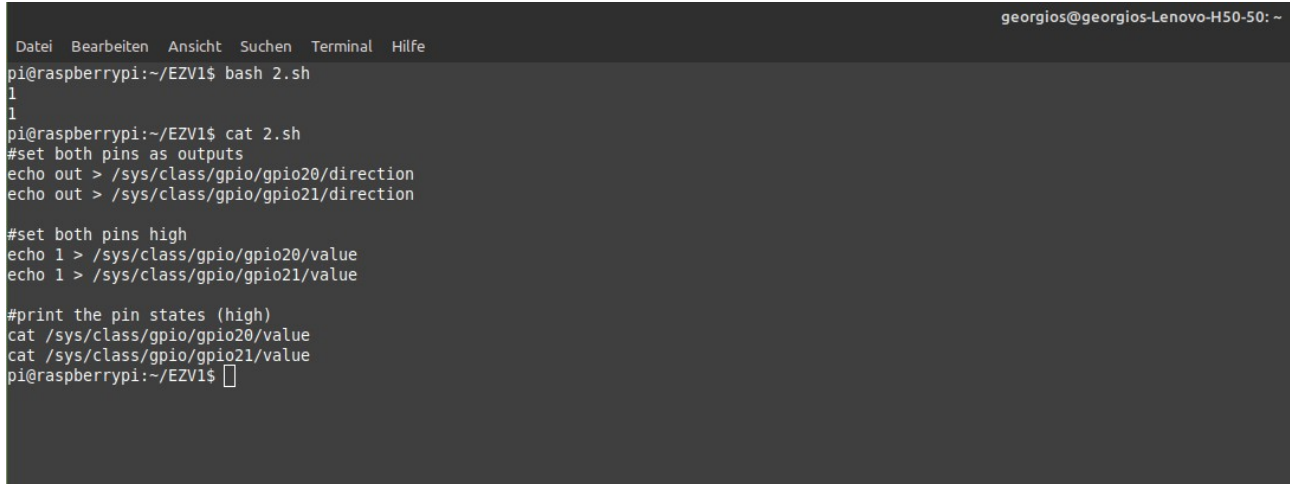


Echtzeitverarbeitung 1 Georgios Markou

1)

3)

A terminal window titled 'georgios@georgios-Lenovo-H50-50: ~' with a menu bar (Datei, Bearbeiten, Ansicht, Suchen, Terminal, Hilfe). The terminal shows the execution of a script '2.sh' on a Raspberry Pi. The script sets GPIO pins 20 and 21 as outputs, sets them high, and prints their states. The output shows two '1's, indicating the pins are high.

```
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“ zeigt mein Skript.

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$
```

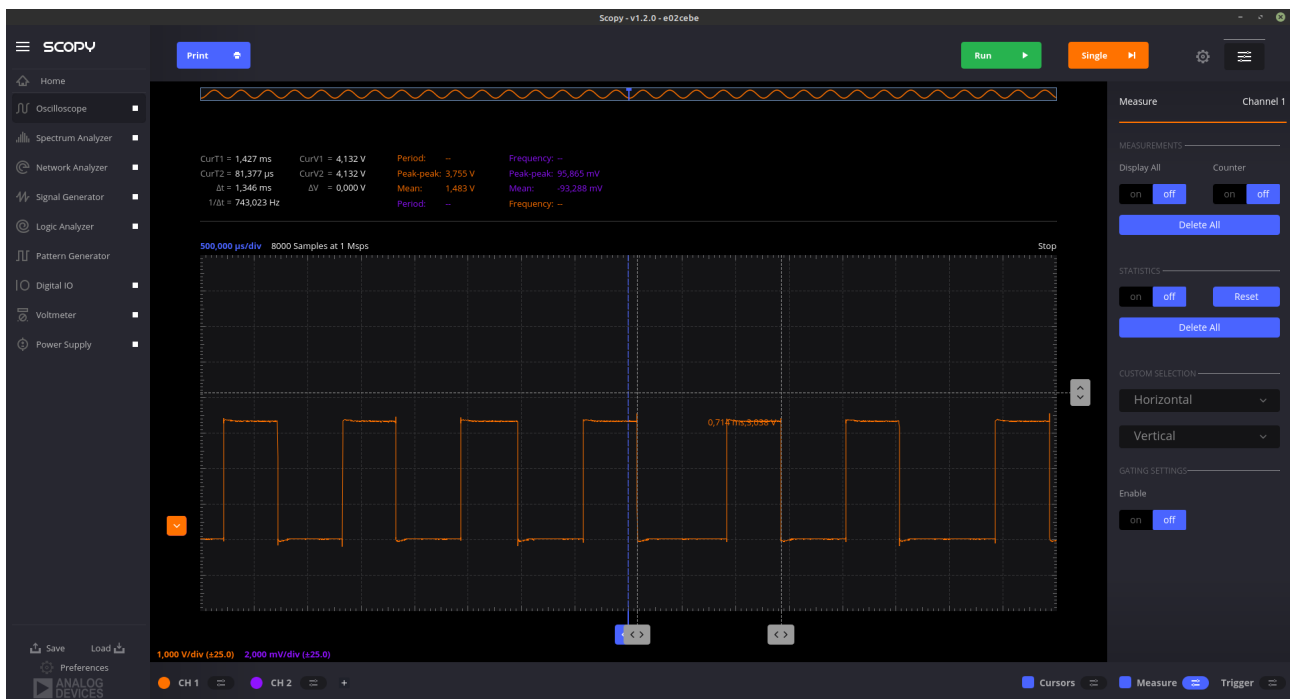
Wie man erkennt habe ich das Skript "3.sh" ausgeführt. Es zeigt den Zustand der beiden Taster an (1 = nicht-gedrückt). Man wird dann aufgefordert die beiden Taster zu drücken und sieht wenn sie gedrückt werden ist die Ausgabe "low" (0 = Taster gedrückt). Mit dem nächsten Befehl zeige ich den Inhalt meines Skripts. Die 2 Zeilen unter dem Kommentar sorgen dafür, dass die Pins "HIGH" sind wenn die Taster nicht gedrückt werden.

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

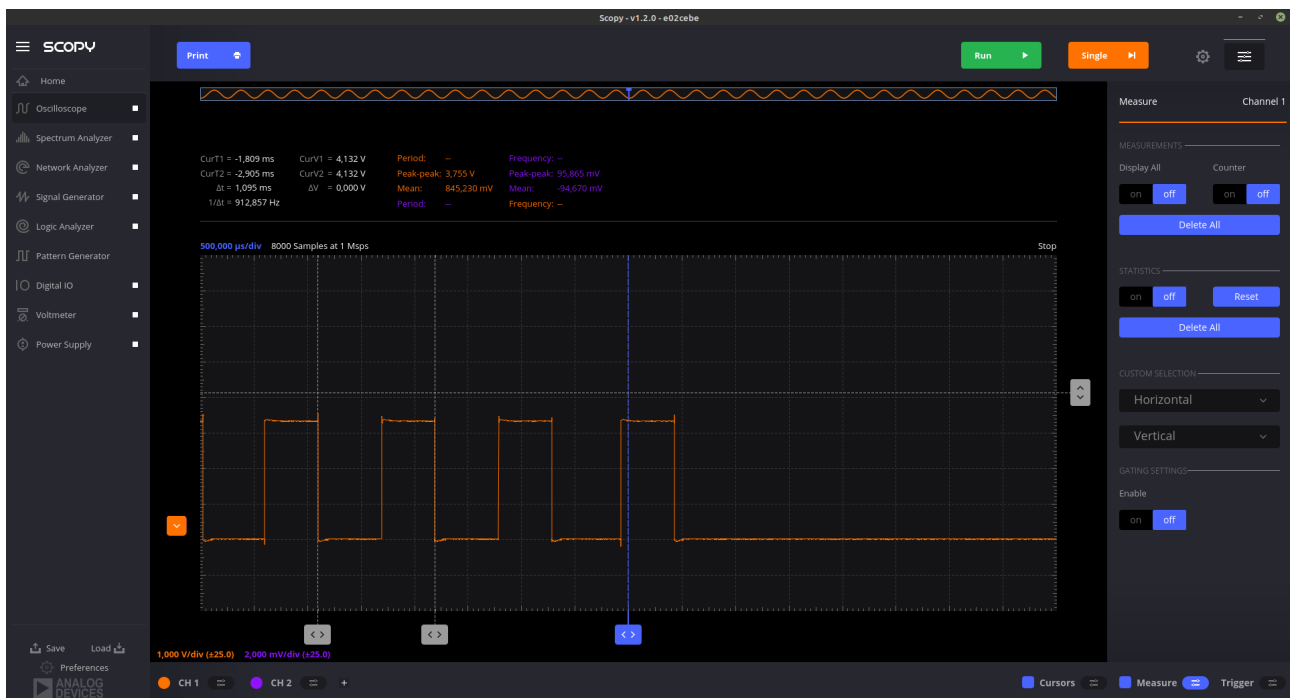
while true
do
    echo 1 > /sys/class/gpio/gpio21/value
    echo 0 > /sys/class/gpio/gpio21/value
done
pi@raspberrypi:~/EZV1$
```

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



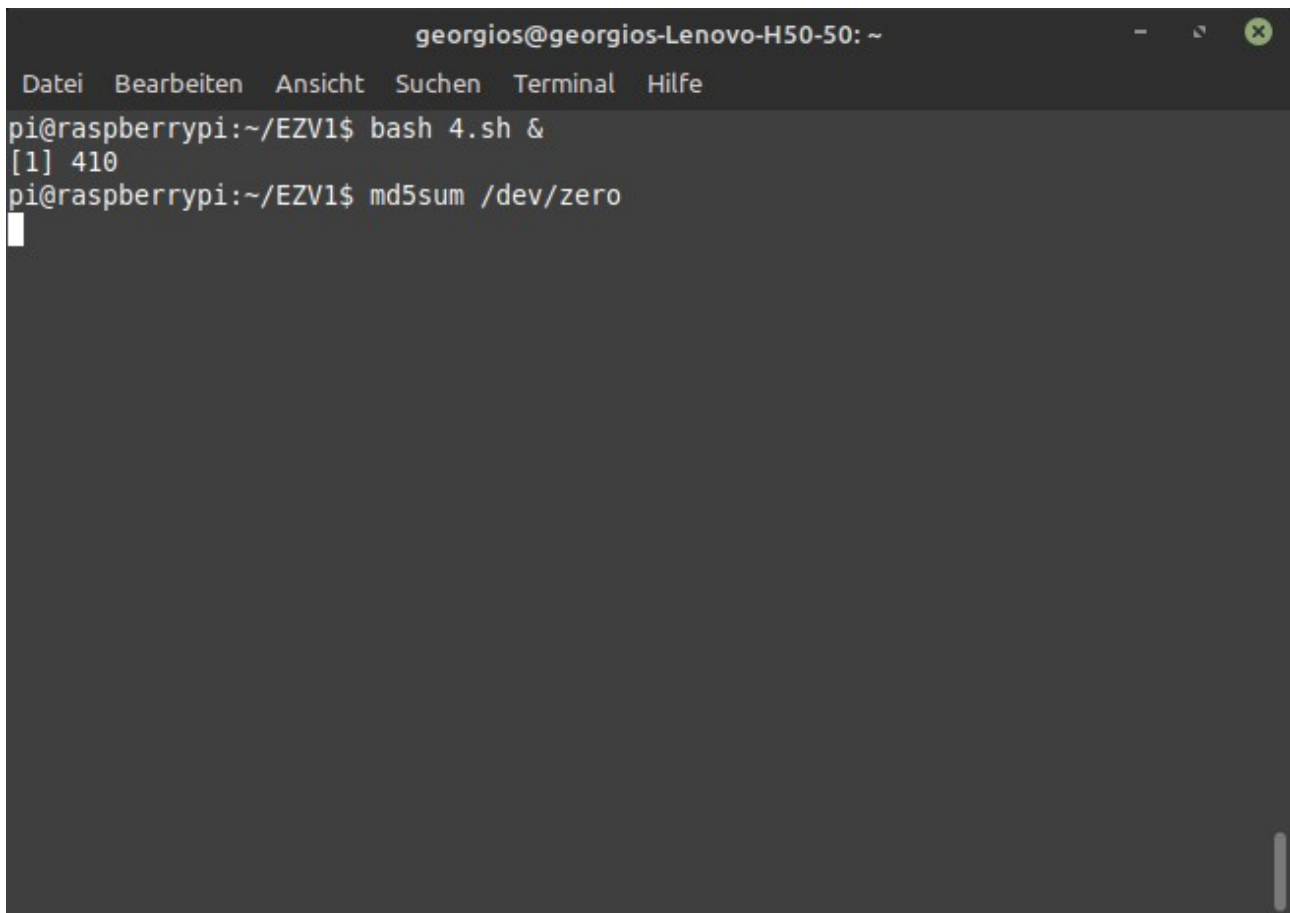


Diese 2 Schnapschüsse von Scopy zeigen, dass die Frequenz schwankt (1. Bild ca. 743 Hz; 2. Bild 934 Hz). Bei mir gab es selten Aussetzer (keine anderen Programme wurden ausgeführt). Die Aussetzer kommen vom Betriebssystem (Scheduler), weil mehrere Ressourcen gleichzeitig verwaltet werden müssen.



In diesem Schnapschuss sieht man, dass es Aussetzer gab. Die gemessene Frequenz war ca. 913 Hz jedoch sieht man auf der rechten Seite das 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.

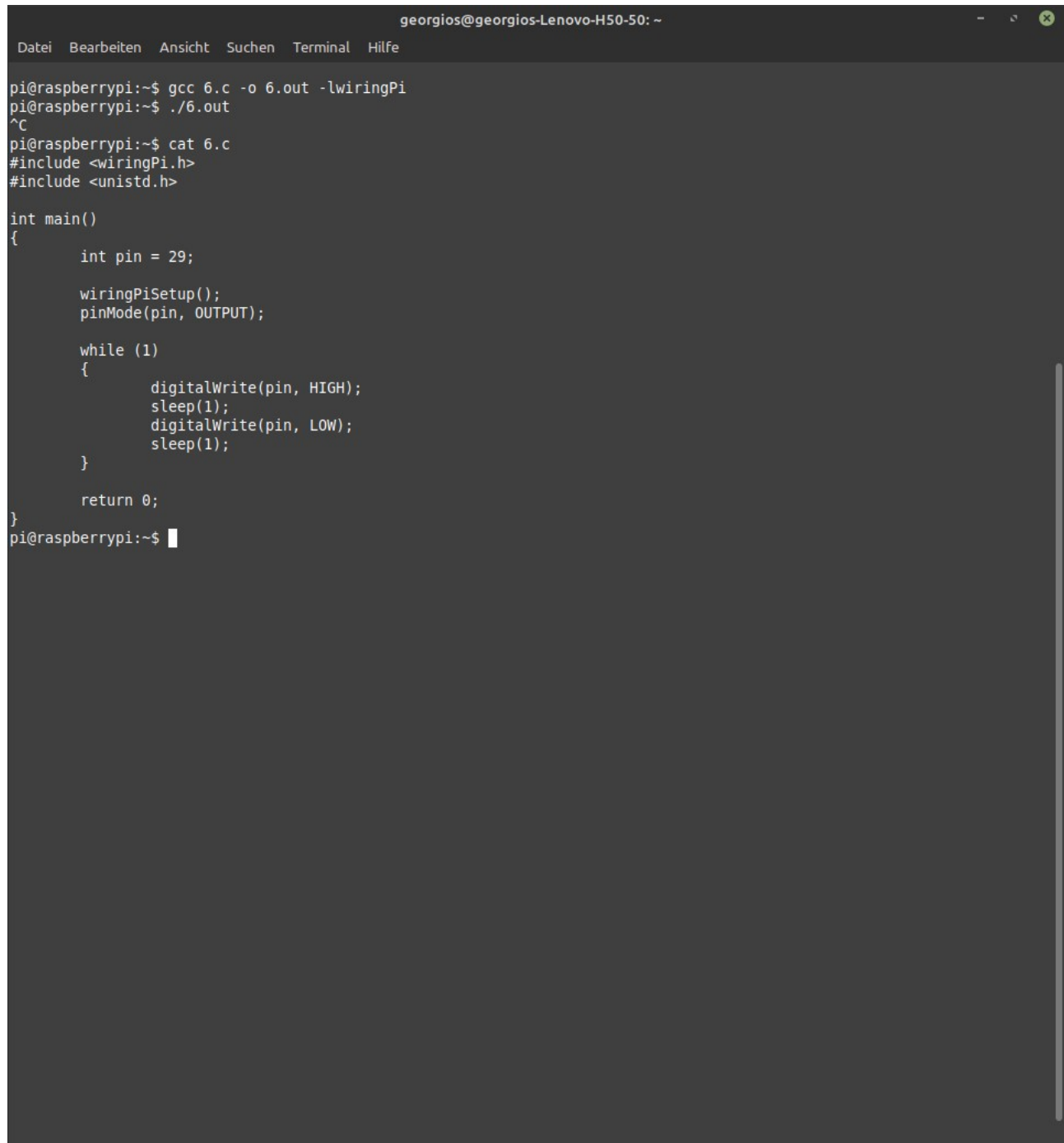
A terminal window titled 'georgios@georgios-Lenovo-H50-50: ~' with a menu bar containing 'Datei', 'Bearbeiten', 'Ansicht', 'Suchen', 'Terminal', and 'Hilfe'. The terminal shows the user 'pi' at 'raspberrypi' in the directory '~/EZV1'. The first command is 'bash 4.sh &', which returns '[1] 410'. The second command is 'md5sum /dev/zero', followed by a blank line.

```
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.

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>

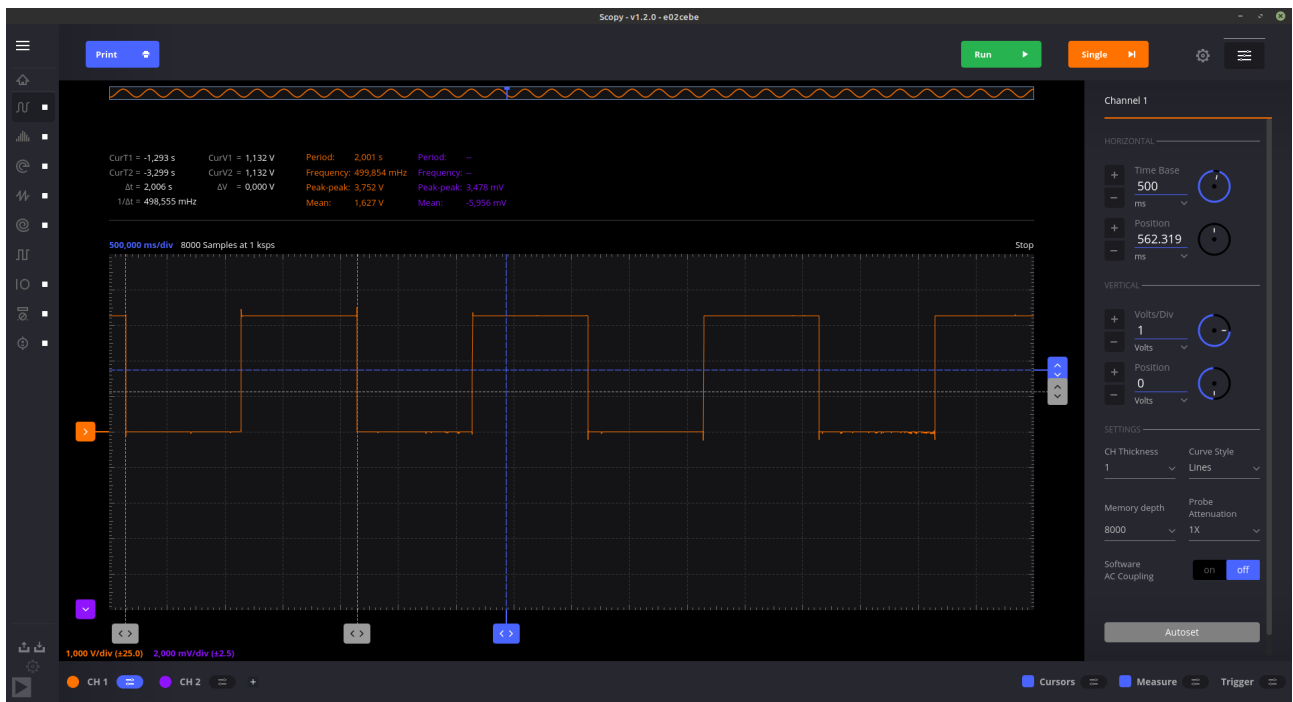
int main()
{
    int pin = 29;

    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

7)

[illegible]

pl

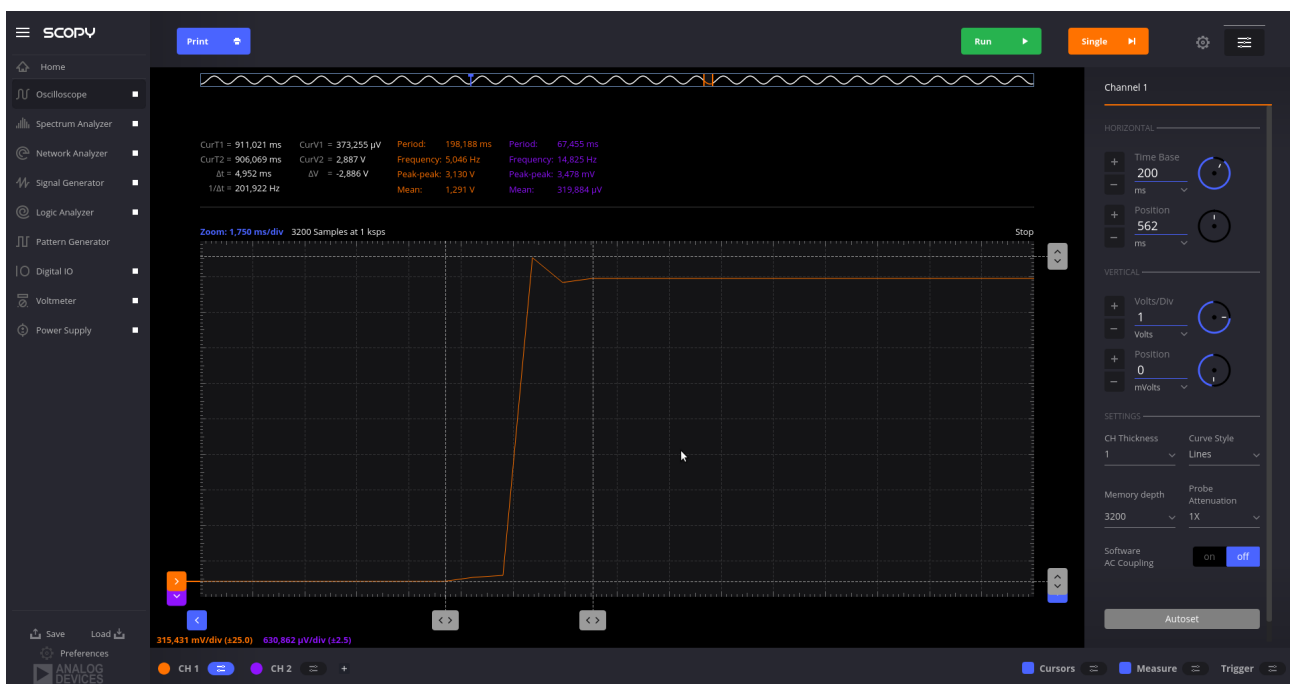
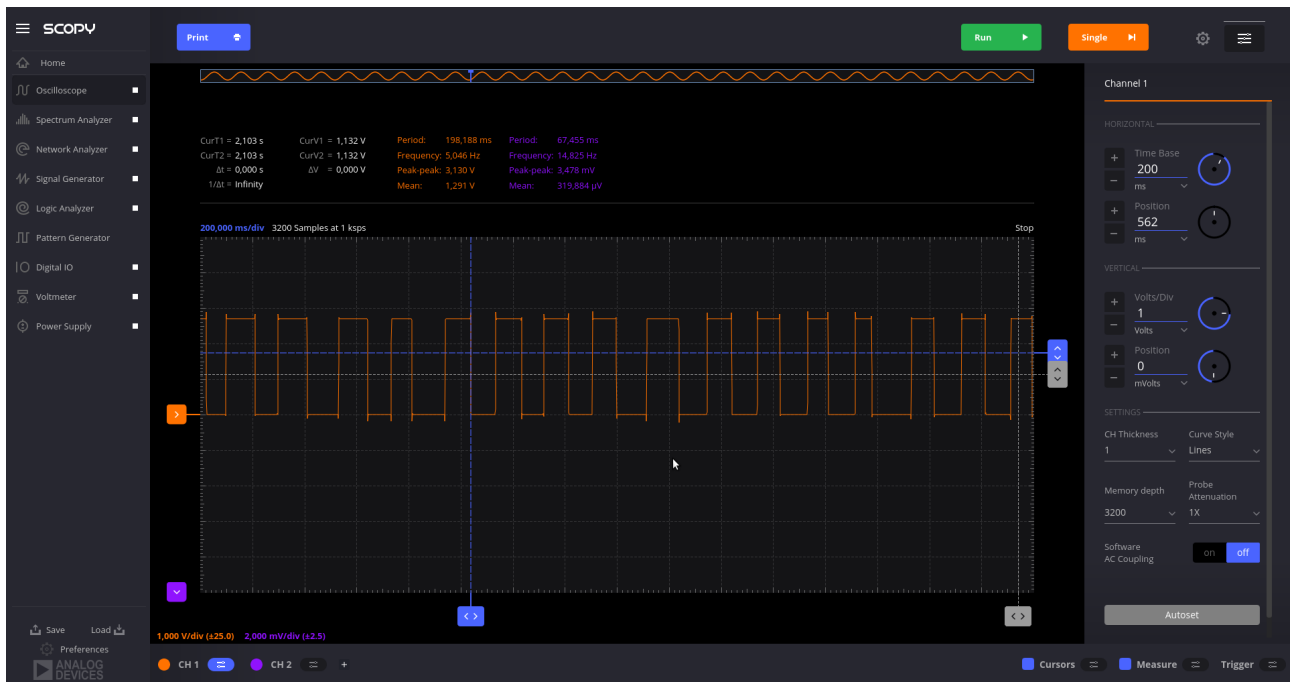
```
#include <stdio.h>
```

p1

}

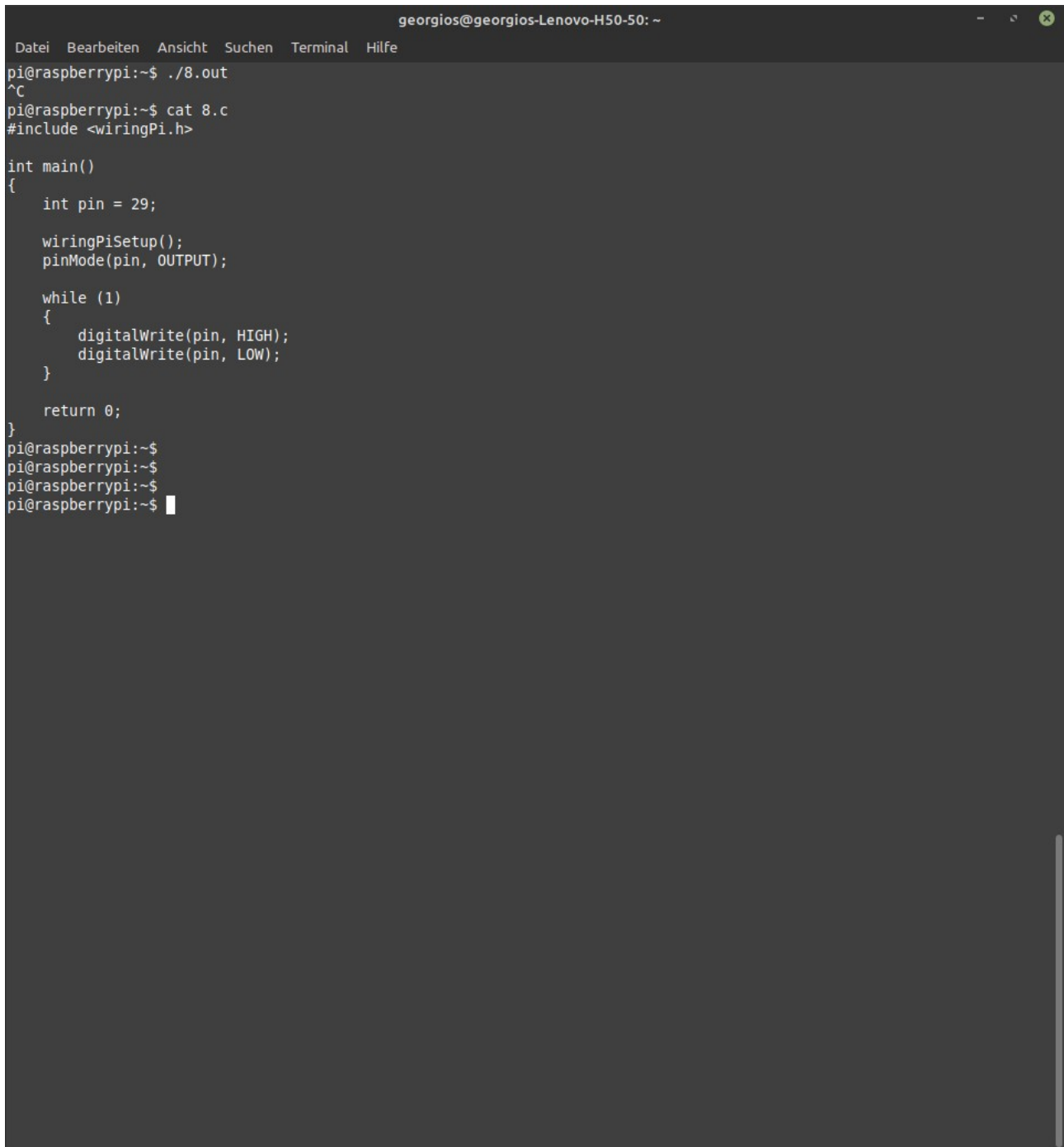
```
pi@raspberrypi:~$
```

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



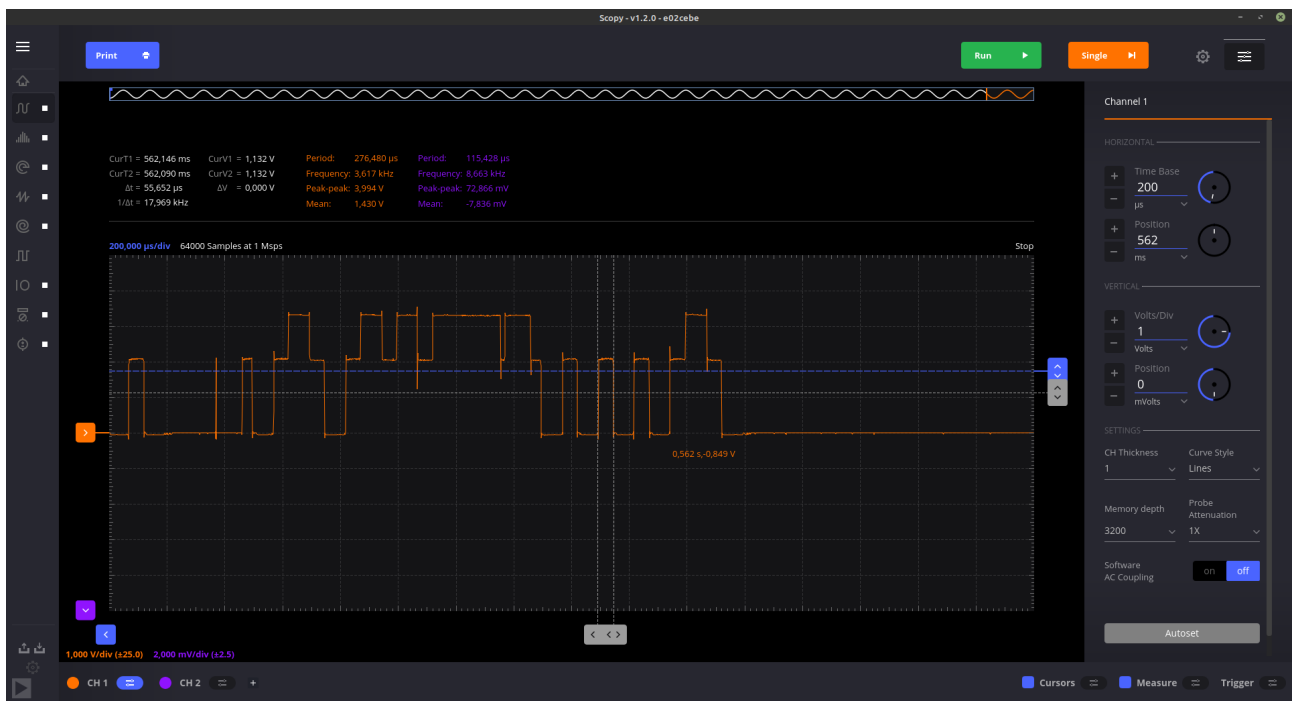
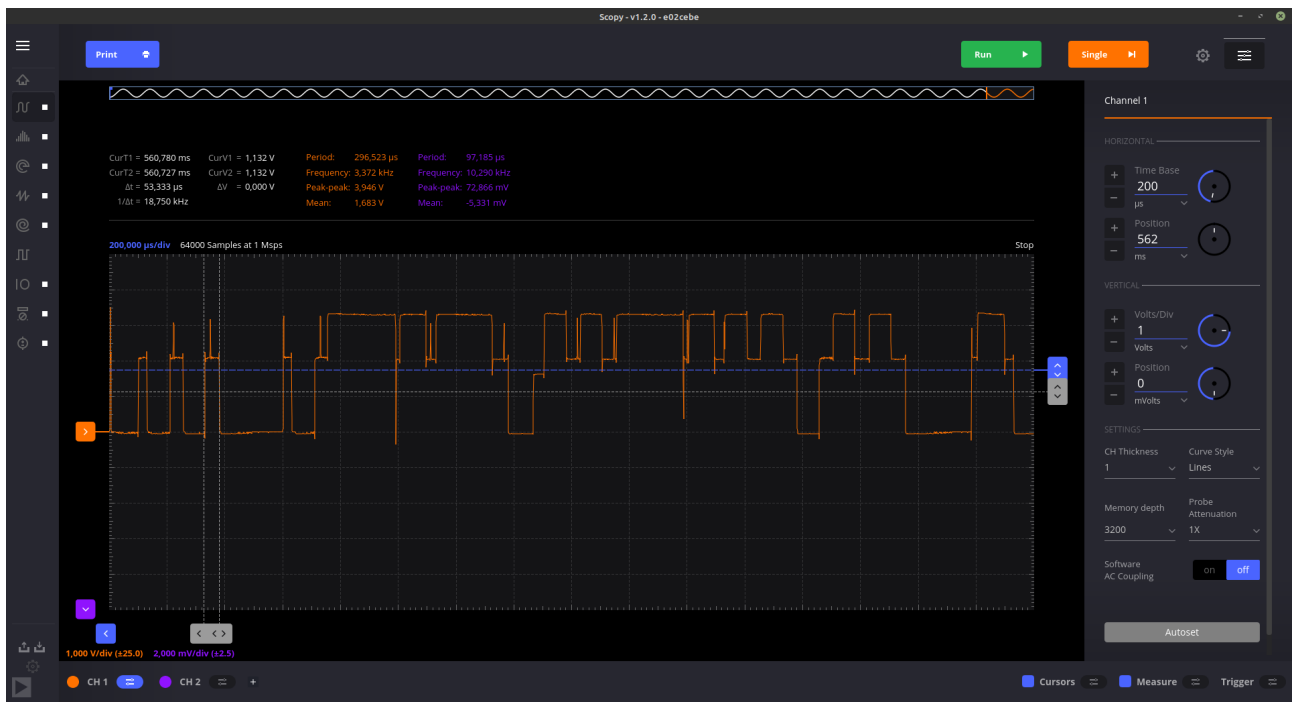
Im ersten Bild sieht man das Signal wenn ich den Taster sehr schnell betätige. Im zweiten Bild habe ich mir eine zufällige steigende Flanke ausgesucht. Entgegen meinen Erwartungen hat der Taster nicht geprellt.

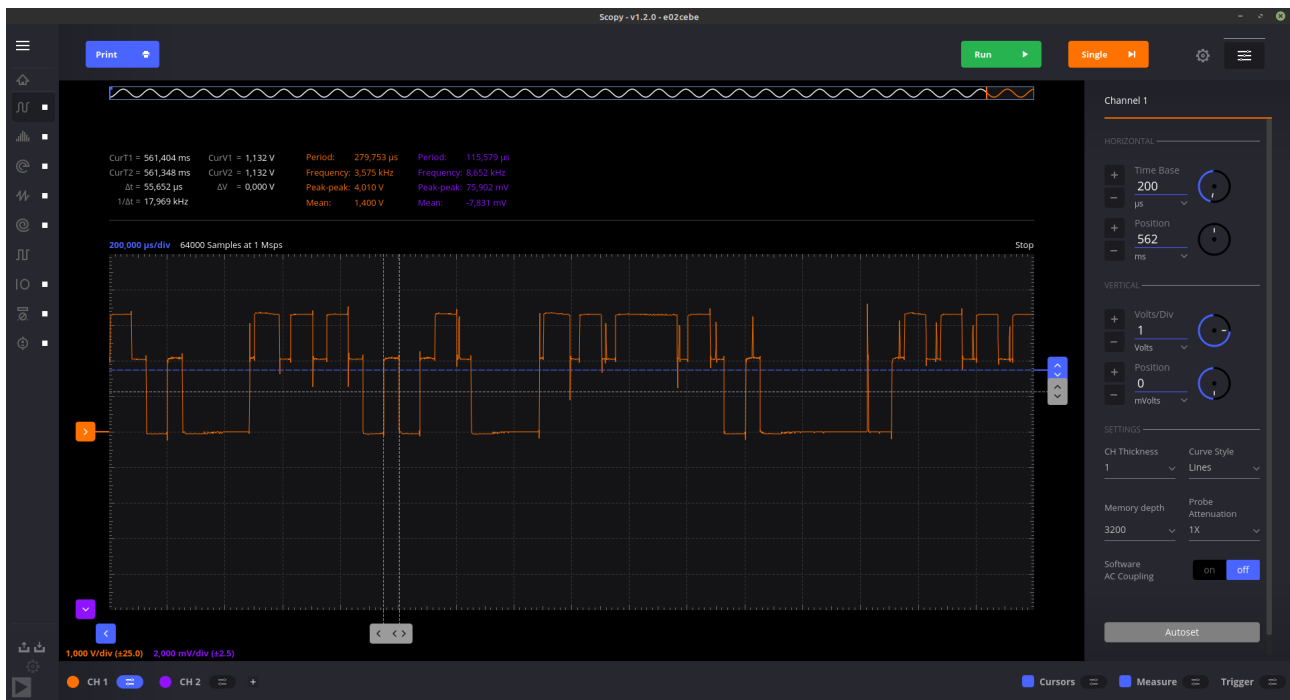
8)



```
georgios@georgios-Lenovo-H50-50: ~  
Datei Bearbeiten Ansicht Suchen Terminal Hilfe  
pi@raspberrypi:~$ ./8.out  
^C  
pi@raspberrypi:~$ cat 8.c  
#include <wiringPi.h>  
  
int main()  
{  
    int pin = 29;  
  
    wiringPiSetup();  
    pinMode(pin, OUTPUT);  
  
    while (1)  
    {  
        digitalWrite(pin, HIGH);  
        digitalWrite(pin, LOW);  
    }  
  
    return 0;  
}  
pi@raspberrypi:~$  
pi@raspberrypi:~$  
pi@raspberrypi:~$  
pi@raspberrypi:~$
```

Mit dem ersten Befehl führe ich das Programm “8.out“ aus und mit dem nächsten Befehl gebe ich den C-Code, der ausgeführt wurde, aus. Erstmal wird das Programm getestet ohne andere Programme im Hintergrund.

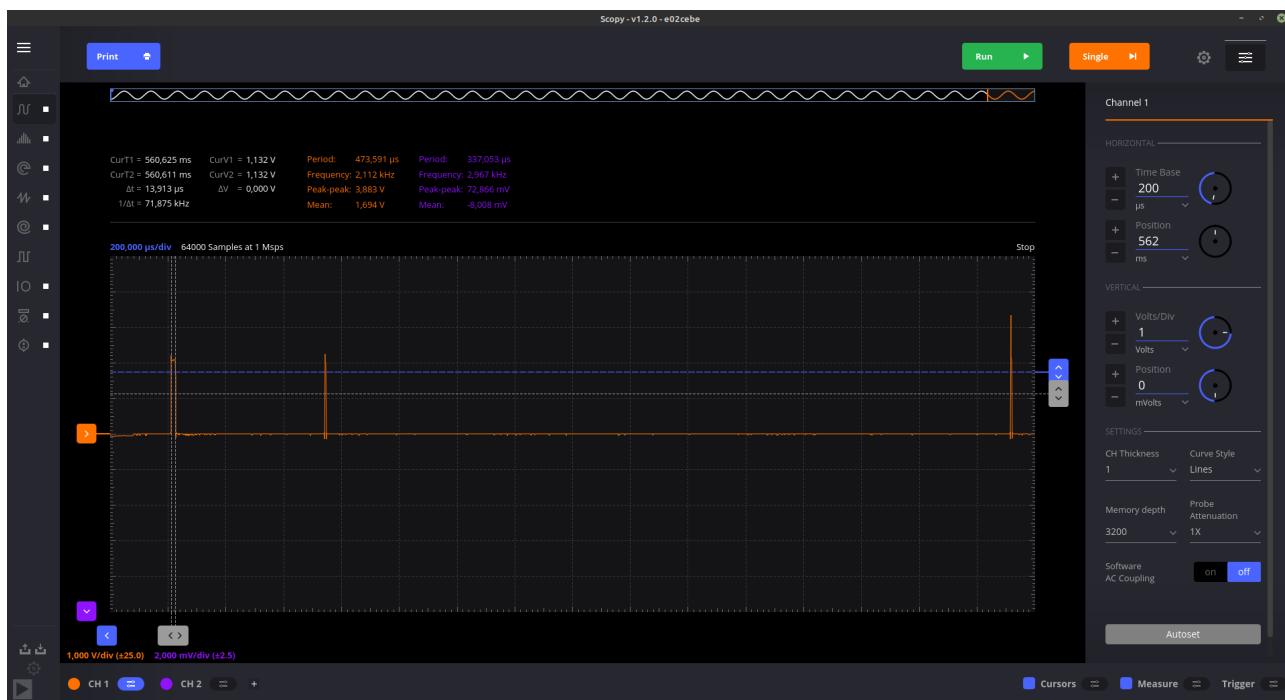




Wie man unschwer erkennt schwankt die Frequenz permanent und das Signal ist absolut unzuverlässig. Es gibt oft Pulse mit einer Periodendauer von ca. 55 μ S. Seitdem ich die neue SD-Karte verwende ist das gesamte System sehr langsam ("nano" braucht mehrere Sekunden um zu starten, genauso wie der "gcc" bis er ein Programm kompiliert hat).

```
Datei Bearbeiten Ansicht Suchen Terminal Hilfe
pi@raspberrypi:~$ ./8.out &
[1] 583
pi@raspberrypi:~$ md5sum /dev/zero
^C
pi@raspberrypi:~$
```

Für das nächste Experiment starte ich das Programm „8.out“ im Hintergrund und „md5sum“ im Vordergrund.





Das Signal hat oft kleine Peaks und ganze Perioden fallen aus. Jedoch erreichen die Peaks oft höhere Frequenzen von ca. 29 kHz.