

## Echtzeitverarbeitung 2 Georgios Markou

1)

Was passiert, wenn Sie die Zykluszeit sehr gering wählen?

Was passiert mit dem Muster der PWM? Warum?

2)

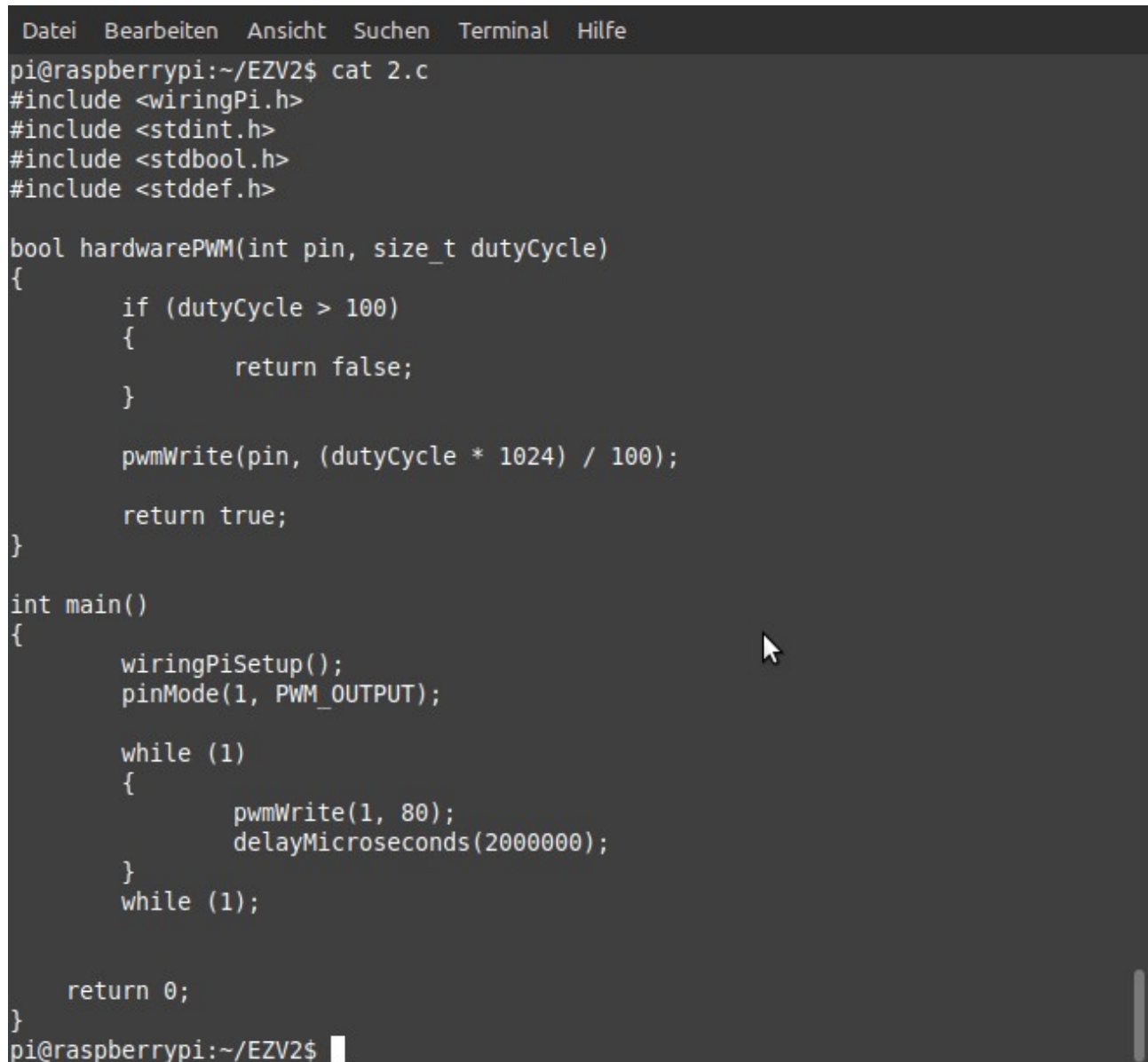
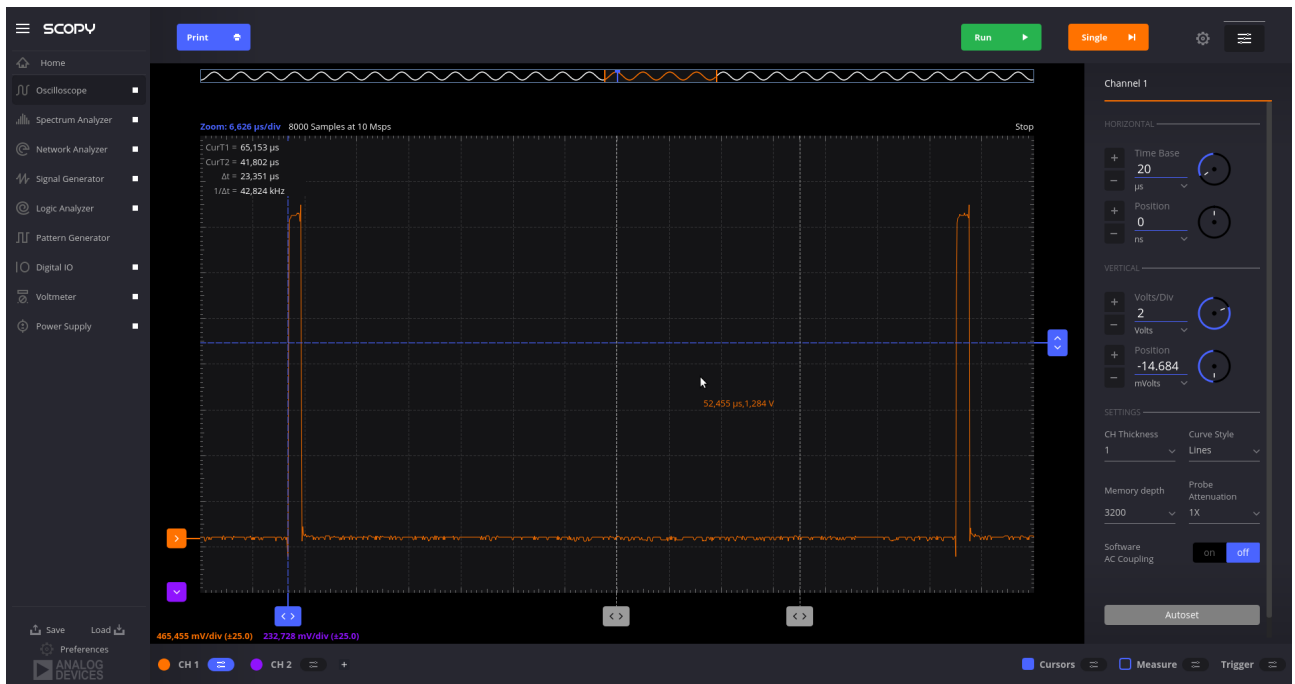
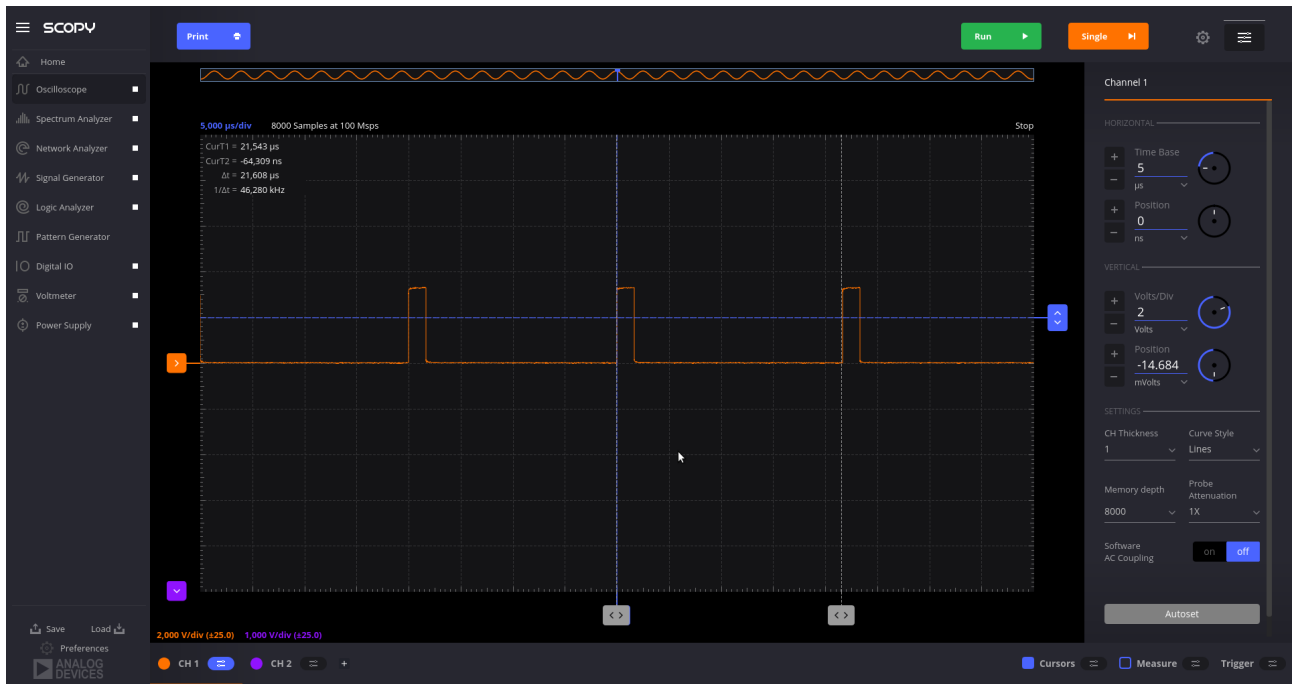
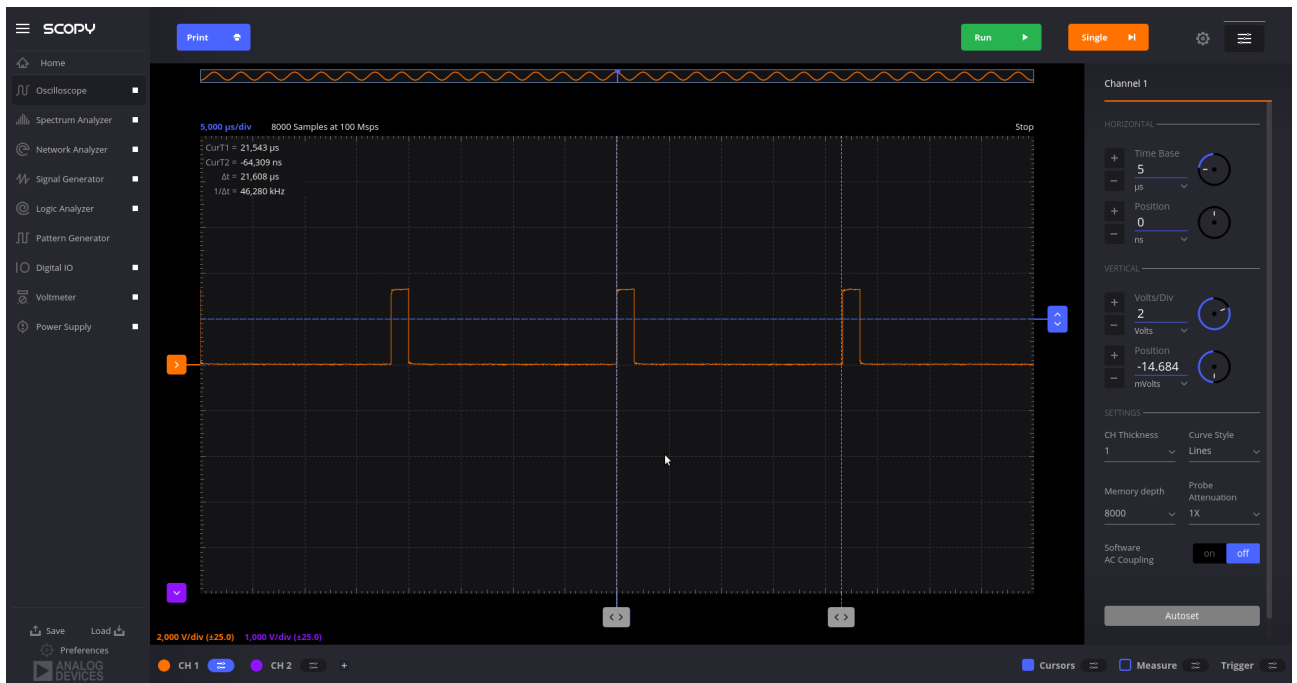
A screenshot of a terminal window on a Raspberry Pi. The window has a dark background and a menu bar at the top with options: Datei, Bearbeiten, Ansicht, Suchen, Terminal, and Hilfe. The terminal shows the command 'cat 2.c' being executed, displaying the contents of a C file named 2.c. The code defines a function 'hardwarePWM' that takes a pin and a duty cycle as arguments. It checks if the duty cycle is greater than 100, and if so, returns false. Otherwise, it calls 'pwmWrite' with the pin and a calculated value, and returns true. The 'main' function sets up the wiringPi library, configures pin 1 as a PWM output, and enters a loop where it writes a duty cycle of 80 to pin 1 and delays for 2,000,000 microseconds. The terminal prompt is 'pi@raspberrypi:~/EZV2\$'.

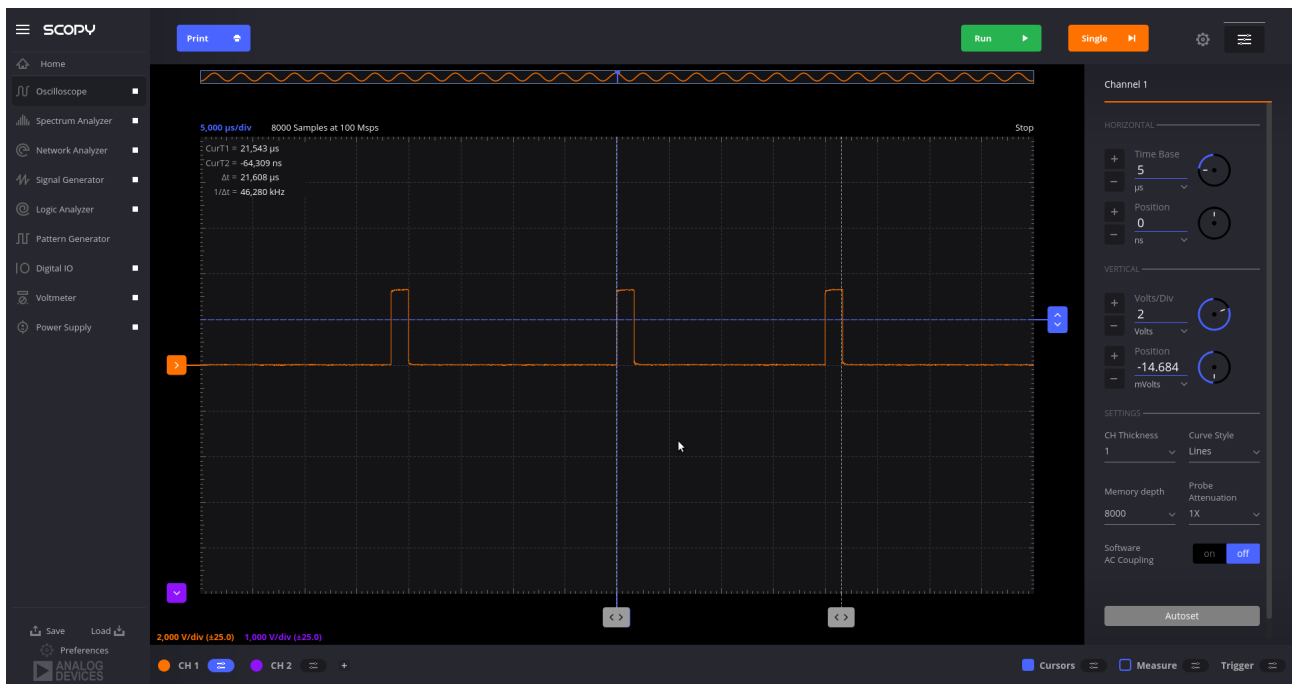
Bild des Codes (die PWM-Duty-Cycle und das Delay wurden zum Testen geändert).





Diese 3 Bilder wurden ohne Last gemacht. Die Frquenz hatte keine Schwankungen.



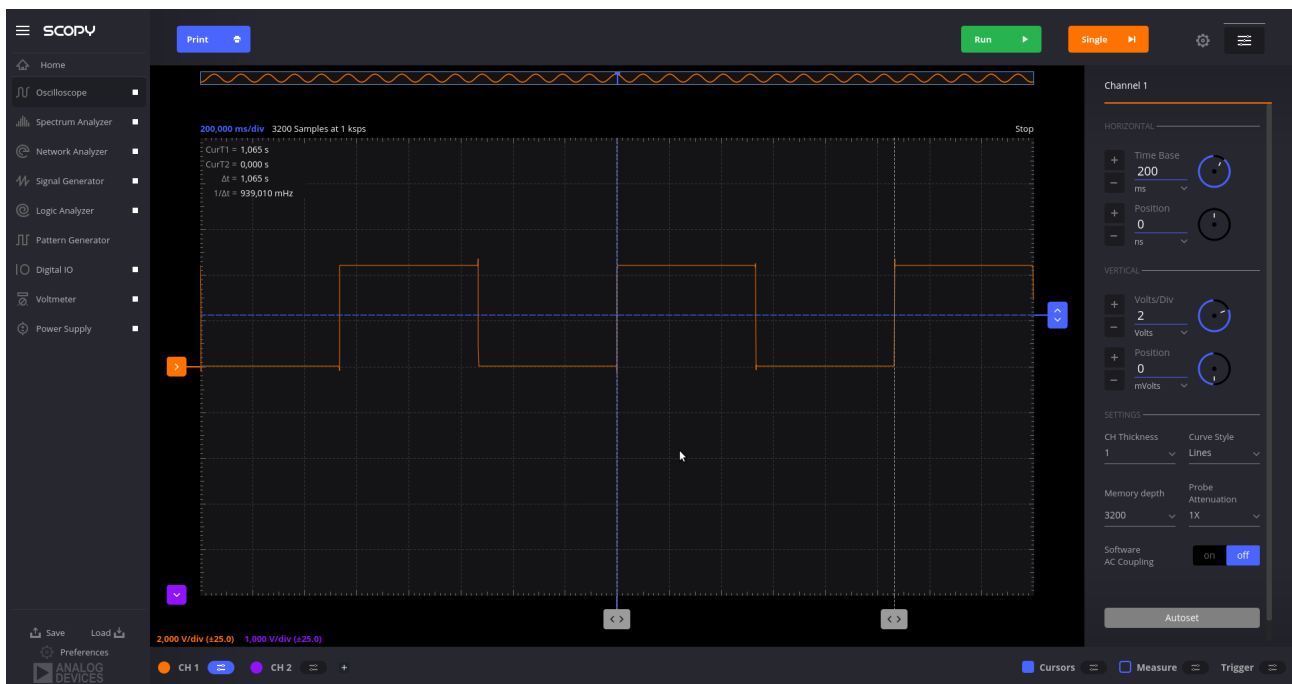


Die letzten wurden bei Last (“md5sum /dev/zero“) gemacht. Wie man sieht habe im zweiten Bild einen “Ausreißer“ erwischt.

Die Hardware-PWM hatte keine Schwankungen.

3)





Wie man sieht lag die höchste gemessene Frequenz bei 54.522 kHz und die niedrigste bei 939 mHz. Bei der hohen Frequenz gab es Schwankungen im Signal, aber keine Aussetzer

4)

```
Datei  Bearbeiten  Ansicht  Suchen  Terminal  Hilfe

pi@raspberrypi:~/EZV2$ cat 4.c
#include <wiringPi.h>
#include <stdint.h>
#include <stdlib.h> //strtoul

int main(int argc, char* argv[])
{
    int switchPin = 2; //SW2
    int gpioPin = 1; //GPIO 18

    wiringPiSetup();
    pinMode(switchPin, INPUT);
    pinMode(gpioPin, OUTPUT);
    digitalWrite(gpioPin, LOW);

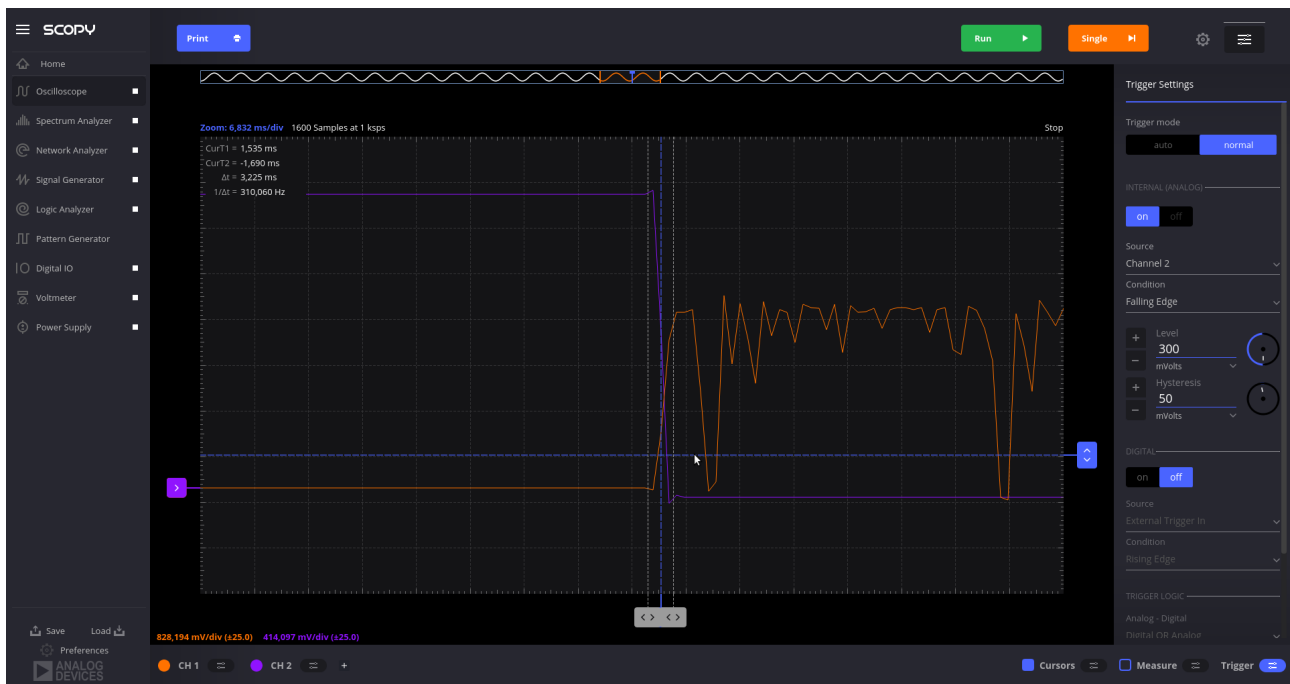
    while (1)
    {
        if (digitalRead(switchPin) == LOW)
        {
            digitalWrite(gpioPin, HIGH);
            delayMicroseconds(1);
            digitalWrite(gpioPin, LOW);
        }

        delayMicroseconds(1);
    }

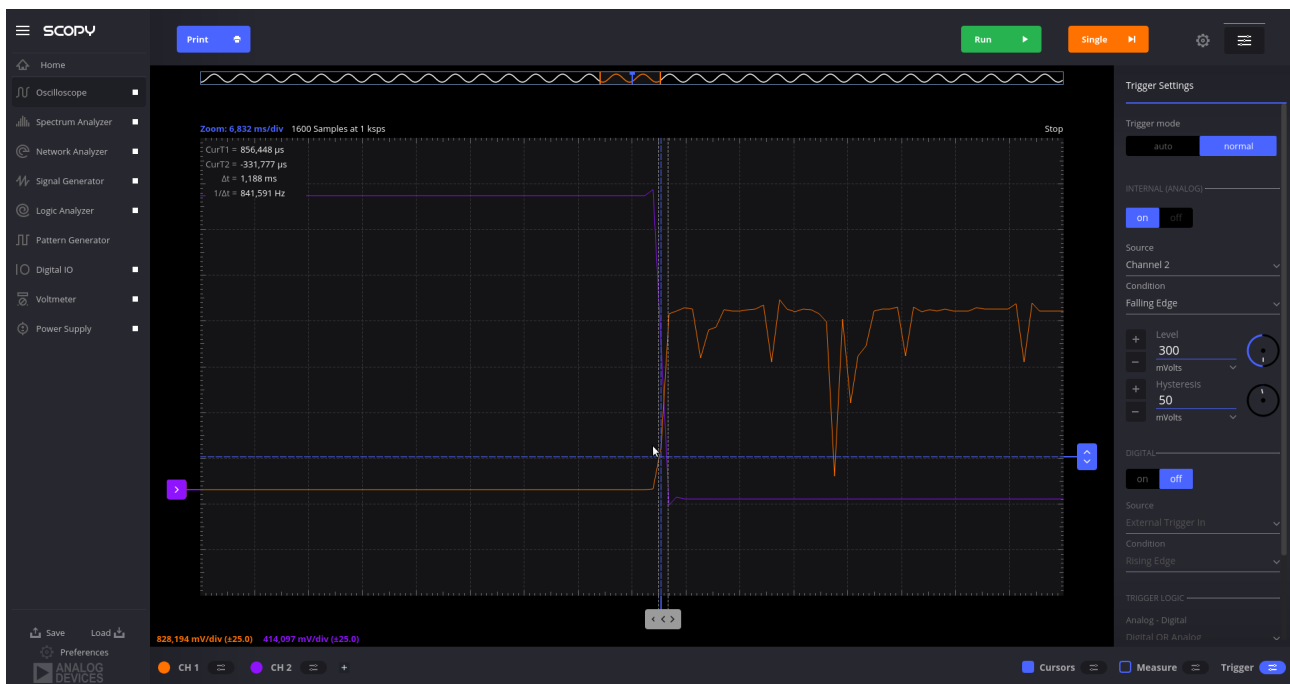
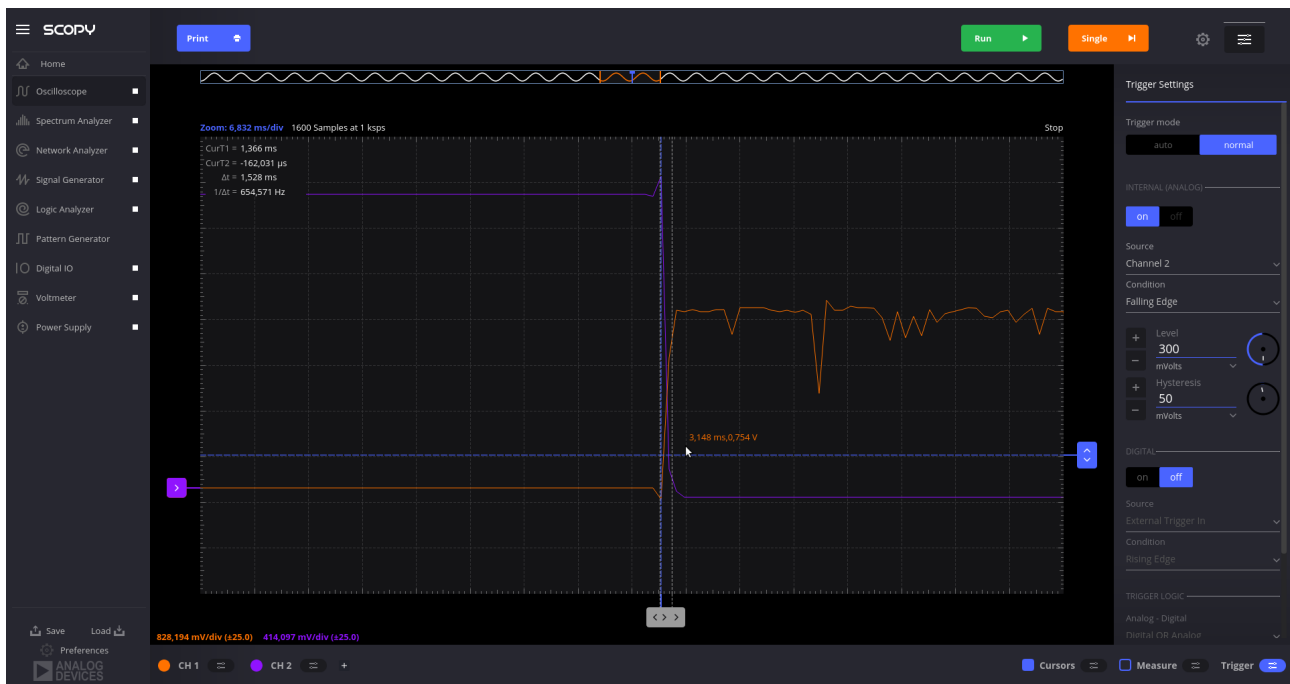
    return 0;
}

pi@raspberrypi:~/EZV2$
```

Der Code für die Aufgabe, da ich den Code in Visual Studio Code schreibe und dann in nano (Raspberry Pi) kopiere ist er nicht richtig eingerückt. Das letzte Delay wurde für die Messungen verändert.



Die zwei Bilder, zeigen (bei gleichen Delay-Einstellungen Delay = 1  $\mu\text{S}$ ) wie sich Last (“md5sum /dev/zero &“) auf die Reaktionszeit auswirkt.



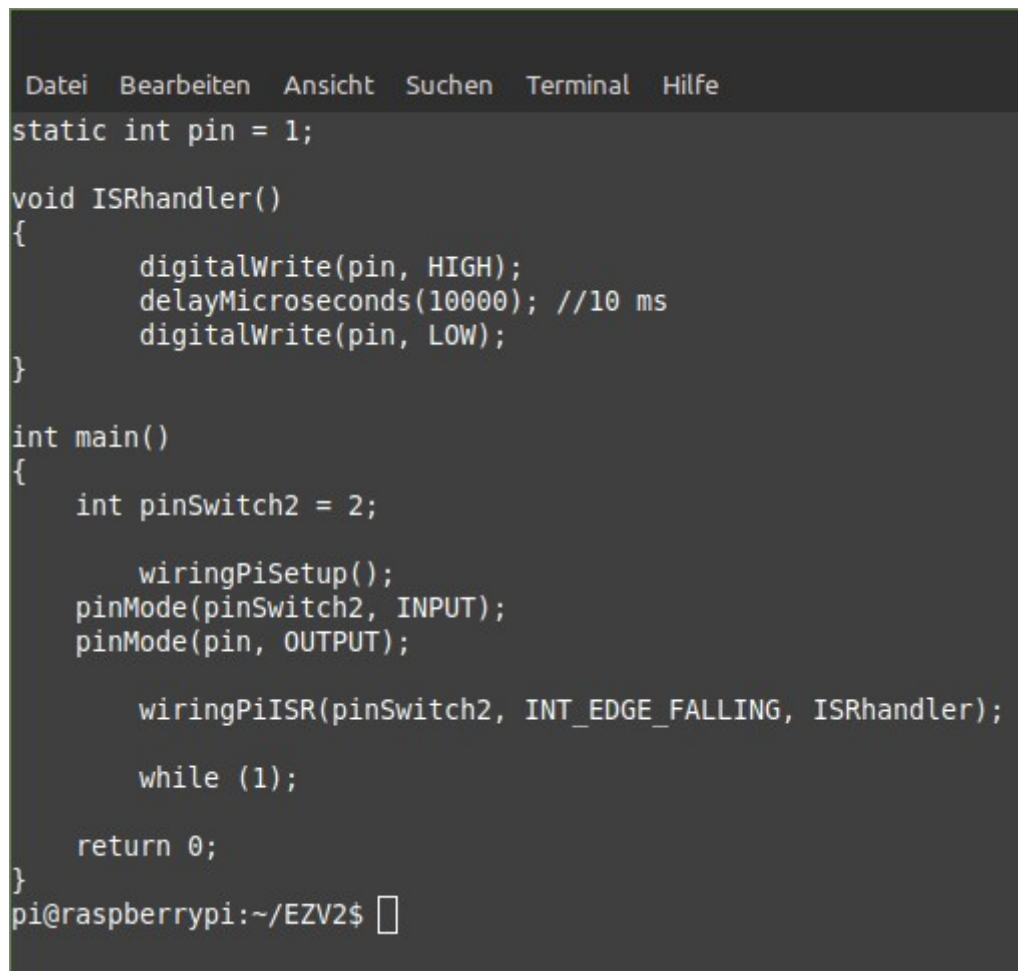
Die zwei Bilder, zeigen (bei gleichen Delay-Einstellungen Delay = 10  $\mu$ S) wie sich Last (“md5sum /dev/zero &“) auf die Reaktionszeit auswirkt.

Die Reaktions-/Antwortszeit hängt auch natürlich auch von der Delay-Zeit ab. Wir verwenden ja kein Interrupt und die Delay Dauer wird immer eingehalten. In dieser Zeit reagiert unser Programm nicht. Allerdings ist die Delay-Zeit hier sehr klein und somit vernachlässigbar.



Wie man hier erkennt, mit Last verschlechtert sich die Antwortzeit.

5)



```
Datei  Bearbeiten  Ansicht  Suchen  Terminal  Hilfe
static int pin = 1;

void ISRhandler()
{
    digitalWrite(pin, HIGH);
    delayMicroseconds(10000); //10 ms
    digitalWrite(pin, LOW);
}

int main()
{
    int pinSwitch2 = 2;

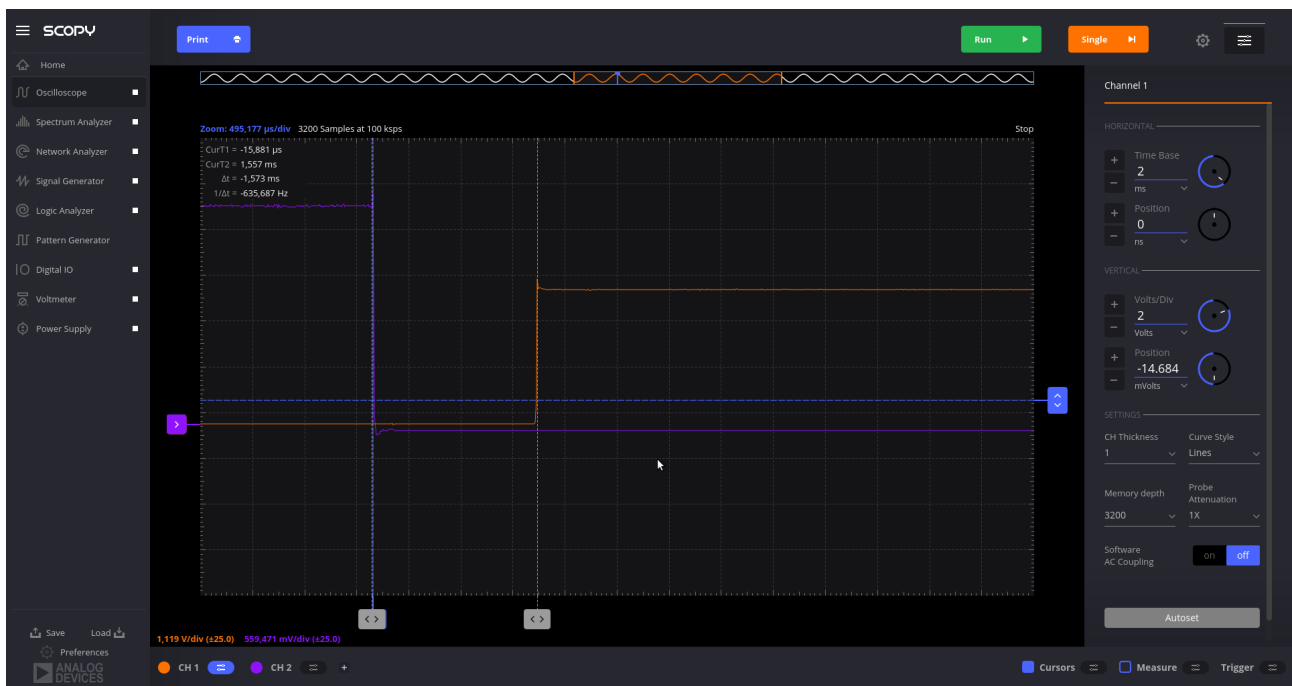
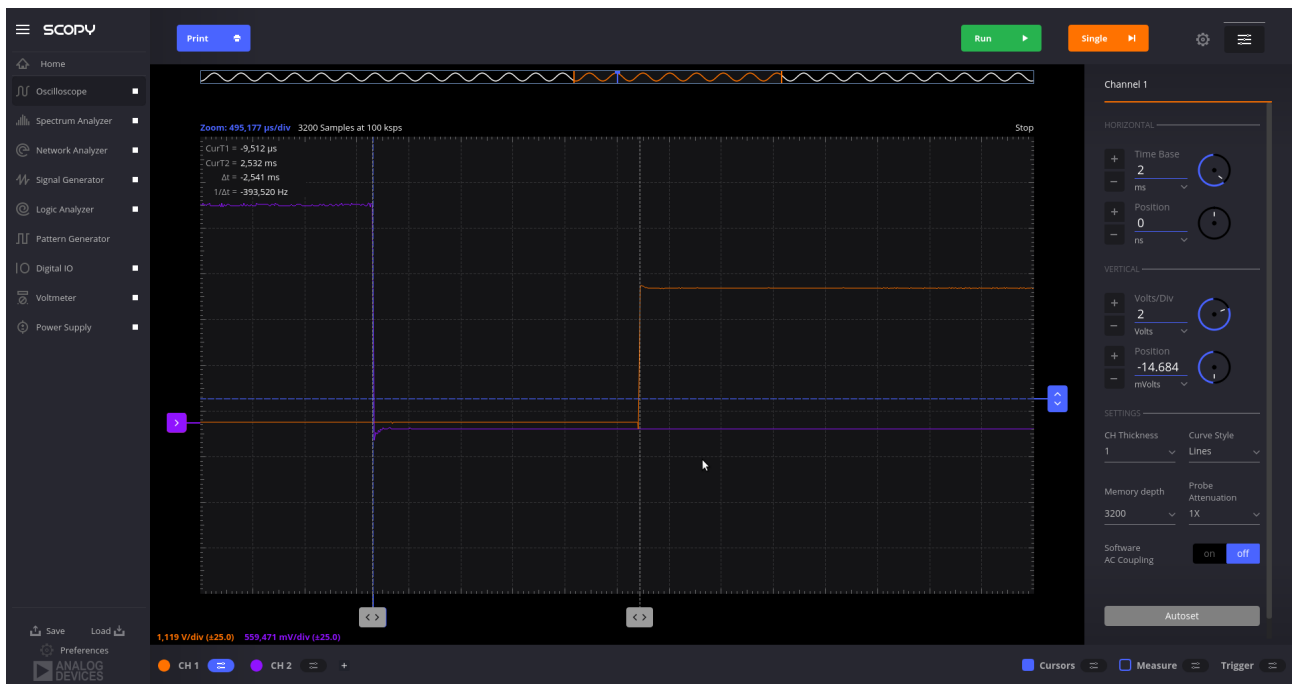
    wiringPiSetup();
    pinMode(pinSwitch2, INPUT);
    pinMode(pin, OUTPUT);

    wiringPiISR(pinSwitch2, INT_EDGE_FALLING, ISRhandler);

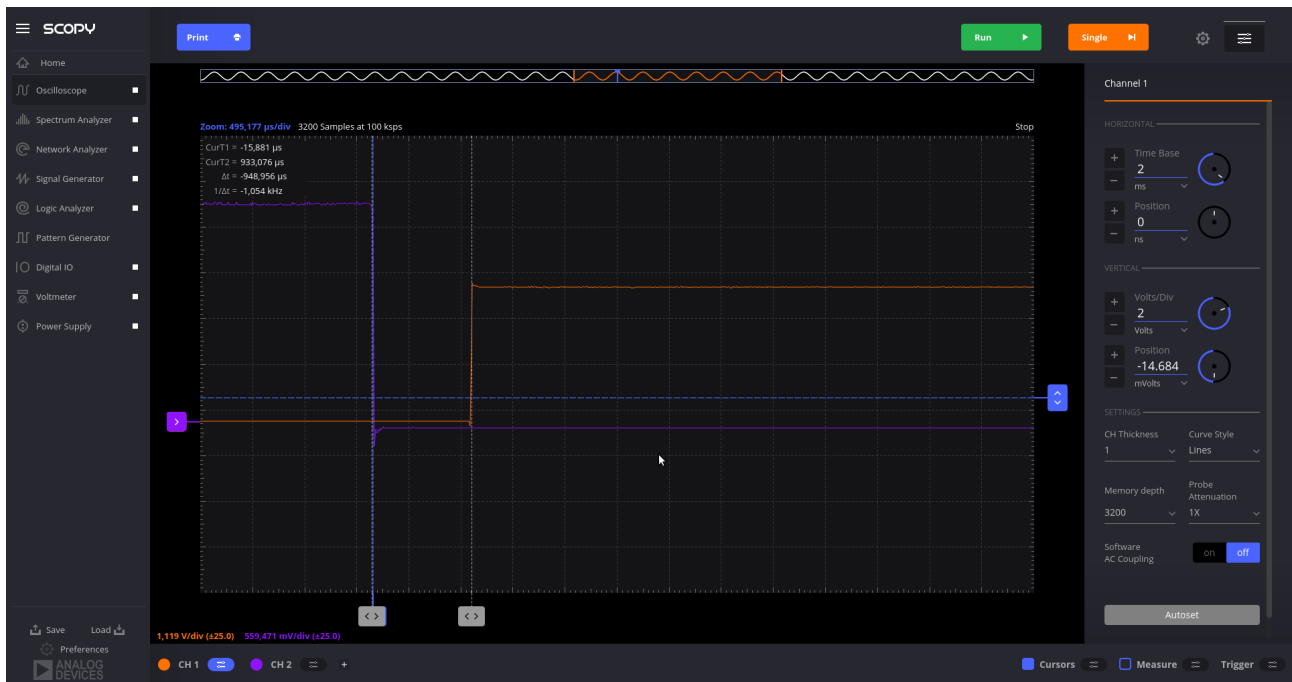
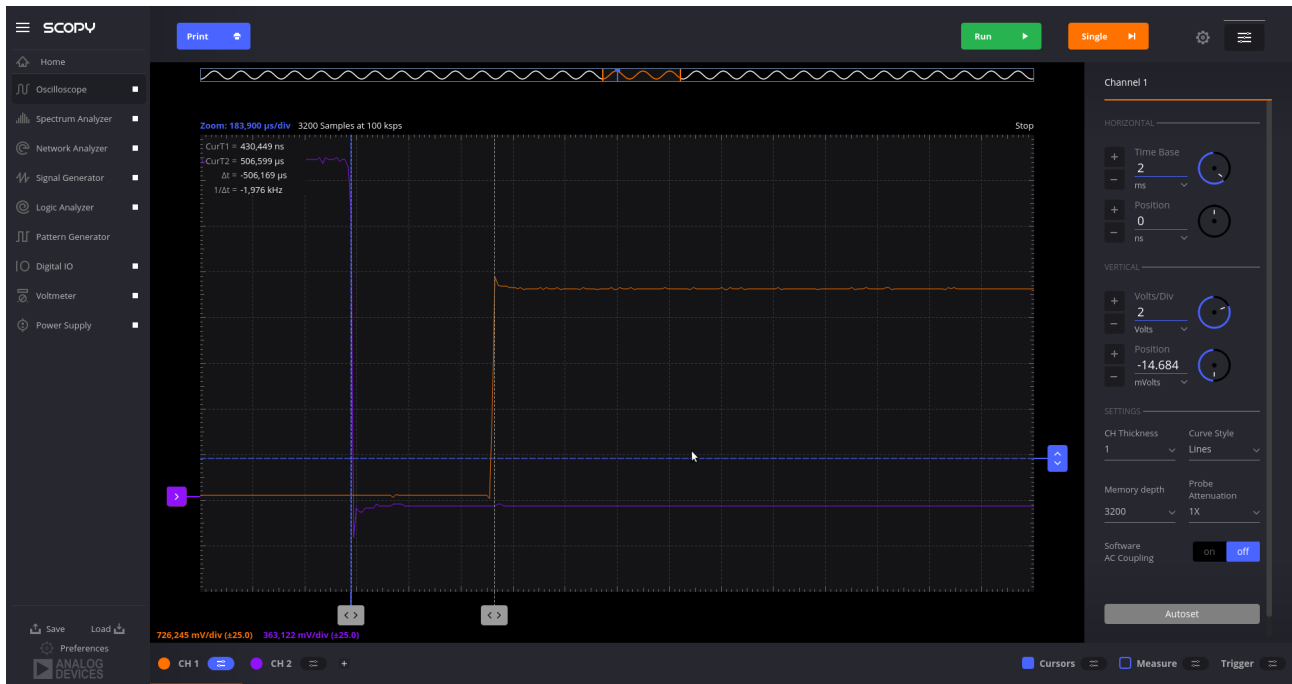
    while (1);

    return 0;
}
pi@raspberrypi:~/EZV2$
```

Code für Aufgabe 5.



Diese zwei Bild wurden bei Last gemacht (“md5sum /dev/zero &“). Es gibt einen kleinen (1 mS) Unterschied in der Reaktionszeit gezeigt in den beiden Bildern.



Diese zwei Bilder wurden ohne Last gemacht, die Reaktionszeiten sind erheblich besser.