PWM

Tim & Dominic

Om de opdracht te beginnen moesten we een major nummer kiezen. Hieronder een lijst met major nummers die al in gebruik waren. Uiteindelijk hebben we zelf voor nummer 99 gekozen. Dit nummer was vrij en werd ook in de slides gebruikt.

```
/bin/sh: can't access tty; job control turned off
# ls -l /dev/
brw-r--r--    1 0        0          31,    3 Sep 10  2010 mtdblock3
crw-r--r--    1 0        0           1,    3 Sep  9  2010 null
crw-rw-rw-    1 0        0          99,    2 Jan  1 00:22 pwm1_duty
crw-rw-rw-    1 0        0          99,    0 Jan  1 00:21 pwm1_enabl
crw-rw-rw-    1 0        0          99,    1 Jan  1 00:22 pwm1_freq
crw-rw-rw-    1 0        0          99,    5 Jan  1 00:23 pwm2_duty
crw-rw-rw-    1 0        0          99,    3 Jan  1 00:22 pwm2_enabl
crw-rw-rw-    1 0        0          99,    4 Jan  1 00:22 pwm2_freq
brw-r--r--    1 0        0           8,    1 Sep  9  2010 sda1
brw-r--r--    1 0        0           8,    2 Sep  9  2010 sda2
drwxrwxrwx    2 0        0              1024 Apr 29  2021 shm
crw-r--r--    1 0        0           4,    2 Jan  1 02:39 tty2
crw-r--r--    1 0        0           4,    3 Jan  1 02:39 tty3
crw-r--r--    1 0        0           4,    4 Jan  1 02:39 tty4
```

bron : http://www.makelinux.net/ldd3/chp-3-sect-2

In de init registeren we dit nummer en in de exit maken we dit nummer weer vrij. Daarnaast zetten we de LCD uit en de pwm clock aan om daadwerkelijk de PWM poorten te kunnen gebruiken.

```
int __init sysfs_init(void)
{
    int rtnval = 0;
    int result = 0;

    rtnval = register_chrdev(99, DEVICE_NAME, &Fops);
    if(rtnval < 0)
    {
        printk("Error registering device");
        return rtnval;
    }

    //disable lcd
    iowrite32(LCD_DISABLE_BIT,io_p2v(LCD_BIT));
    //enable timer
    iowrite32(CLOCK_ENABLE_BIT,io_p2v(PWM_CLOCK_SIGN));

     return result;
}

void __exit sysfs_exit(void)
{
    unregister_chrdev(99, DEVICE_NAME);
}
```

Zoals te zien is in het eerste plaatje zijn er 6 devices: pwm1_enable, pwm1_freq, pwm1_duty, pwm2_enable, pwm2_freq, pwm2_duty. Deze hebben allemaal het major nummer 99, maar hebben allemaal een ander minor nummer. Dit minor nummer gebruiken we om bij de device open een pointer naar het stuk geheugen te zetten waaruit later gelezen of geschreven moet worden.

```c
/* Called when a process tries to open the device file, like */
static int device_open(struct inode *inode, struct file *fp)
{
    int minorNr;
    if (Device_Open) return -EBUSY;

    Device_Open++;
    minorNr = MINOR(inode->i_rdev);

    switch(minorNr)
    {
        case PWM1_ENABLE:
        msg_Ptr = pwm1Enable;
            break;
        case PWM1_FREQ:
        msg_Ptr = pwm1Freq;
            break;
        case PWM1_DUTY:
        msg_Ptr = pwm1Duty;
            break;
        case PWM2_ENABLE:
        msg_Ptr = pwm2Enable;
            break;
        case PWM2_FREQ:
        msg_Ptr = pwm2Freq;
            break;
        case PWM2_DUTY:
        msg_Ptr = pwm2Duty;
            break;
    }

    return 0;
```

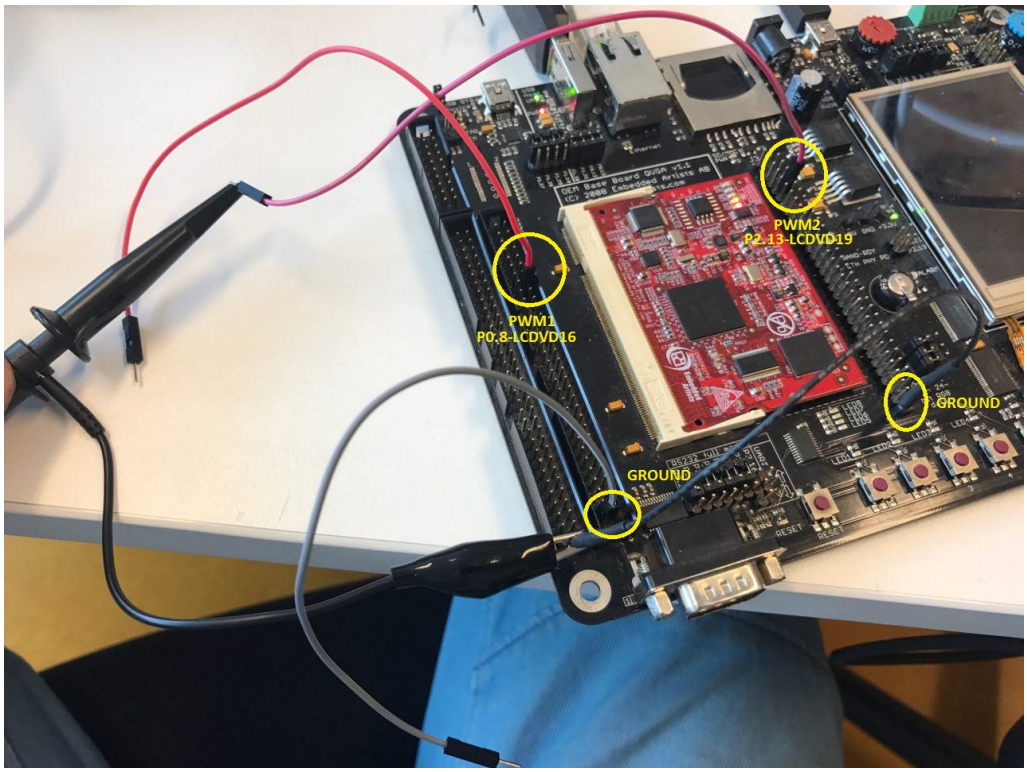
De device read gebruikt de pointer de gezet is in de device open om het stukje geheugen uit te lezen en weer te gegeven aan de gebruiker. Hiervoor gebruiken wij put_user. Dit commando zet data van kernel space naar user space. Device write gebruikt de pointer de gezet is in de device open om in dat stuk geheugen te schrijven. Hiervoor gebruiken wij get_user, deze functie zet data van user space naar kernel space. Een voorbeeld om deze lees en schrijf functies te gebruiken is te zien in onderstaand plaatje:

```
# echo 0 > /dev/pwm1_enable
# cat /dev/pwm1_enable
0
# echo 40 > /dev/pwm1_duty
# cat /dev/pwm1_duty
40
#
```

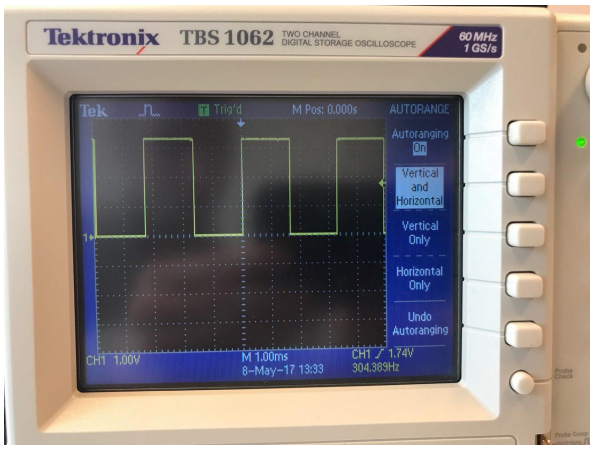
Elke keer als de gebruiker via echo een nieuwe waarde schrijft wordt het PWM signaal geupdate. De waardes worden uit de char[] gelezen en omgezet naar een uint32_t. Deze waarde wordt vervolgens in het PWM register geschreven. Dit gebeurt voor beide PWMs.

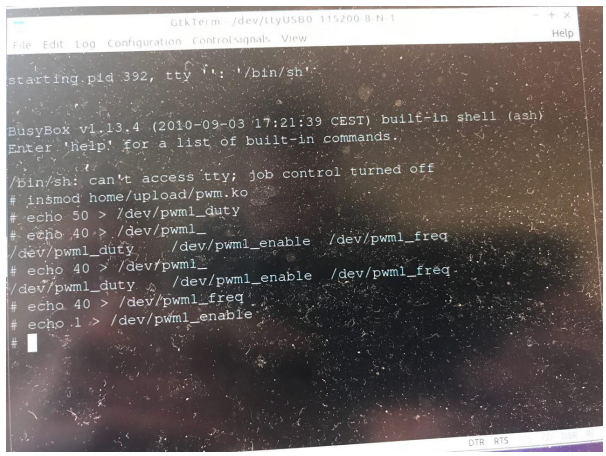
Demo

Hieronder is een overzicht hoe de oscilloscoop is aangesloten op de LPC3250.



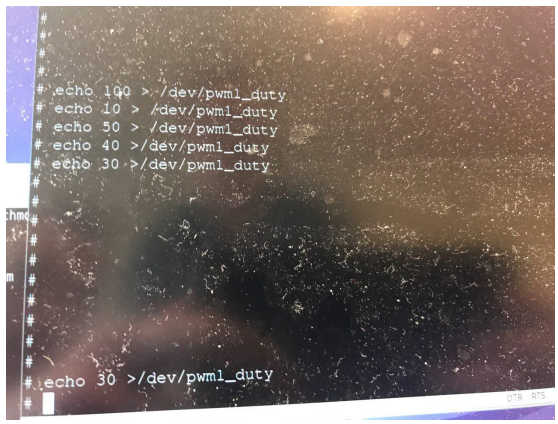
PWM 1





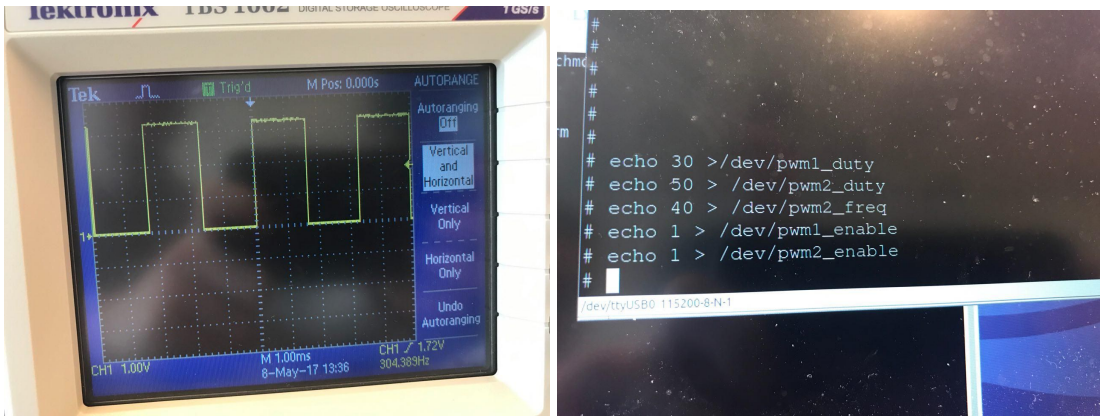
Op het bovenstaande plaatje is de output te zien die verkregen is over PWM pin 1 op de oscilloscoop. Bij deze output is er een

- Dutycycle van 50
- Frequency van 40
- Enable van 1



Bij dit resultaat is alleen de dutycycle veranderd naar 30. Hierdoor is er te zien dat de pieken van het pwm signaal korter worden.

PWM2



Op het bovenstaande plaatje is de output te zien die verkregen is over PWM pin 2 op de oscilloscoop. Bij deze output is er een

- Dutycycle van 50
- Frequency van 40
- Enable van 1

Bij dit resultaat is alleen de frequency veranderd naar 50. Hierdoor is er te zien dat er meerdere pieken in dezelfde tijd voorkomen.