Michael Enzenhofer
Musik & Medien
Musikschule Gramastetten
michael.enzenhofer@eduhi.at Juli 2013

Vers.: 0.2
Neueste Ausgaben auf:
http://michael-enzenhofer.jimdo.com/werkzeuge-tools/electronics/pcduino/

W o r k a r o u n d

pcDuino <-> Pure Data (PD) <-> pcDuino <-> Pure Data

Ein erster Versuch
GPIO-Pins vom pcDuino mit Pure Data ein- und auszulesen.

(mit Einsteigerniveau)

Links:

- http://www.pcduino.com
- https://learn.sparkfun.com/tutorials/programming-the-pcduino

Begriffserleuterungen:

http://de.wikipedia.org

LUBUNTU
pcDuino wird ausgeliefert mit der LUBUNTU-Distribution von LINUX.

.... ist ein offizielles Derivat der Linux-Distribution *Ubuntu*, das LXDE als Desktop-Umgebung nutzt. Das Betriebssystem ist auf alte und schwache Hardware ausgelegt

Der Name *Lubuntu* setzt sich aus *LXDE* und *Ubuntu* zusammen.
Hierbei steht *LXDE* für das „Lightweight X11 Desktop Environment“.

GPIO
Allzweckeingabe/-ausgabe (engl. *GPIO - General Purpose Input/Output*) ist ein allgemeiner Kontaktstift an einem integrierten Schaltkreis, dessen Verhalten, unabhängig, ob als Eingabe- oder Ausgabekontakt, durch logische Programmierung frei bestimmbar ist. GPIO-Kontakten ist kein Zweck vorgegeben, sie sind daher standardmäßig unbelegt.

ADC
Ein **Analog-Digital-Umsetzer** (**ADU**, engl. **ADC** für ***A**nalog-to-**D**igital-**C**onverter*), auch **Analog-Digital-Wandler** oder **A/D-Wandler**, ist ein elektronisches Gerät oder Bauteil zur Umsetzung analoger Eingangssignale in digitale Daten.

KOMMANDOZEILE
Die **Kommandozeile**, **Befehlszeile** oder aus dem Englischen **command-line interface**, kurz **CLI**, oft auch als Konsole oder Terminal bezeichnet, ist ein Eingabebereich (*interface*) für die Steuerung einer Software, der typischerweise (aber nicht zwingend) im Textmodus abläuft. Je nach Betriebssystem wird die Kommandozeile von einer Shell oder einem *Kommandozeileninterpreter* (im Englischen ebenfalls mit CLI für *command-line interpreter* abgekürzt - die Abk. hat also zwei Bedeutungen) ausgewertet und die entsprechende Funktion ausgeführt.

LXTerminal
Der Terminal-Emulator auf Basis des Xfce-Terminals in LUBUNTU

Vorbereitende Arbeitsgänge:

• Installieren von PD mit dem Packet-Manager innerhalb LUBUNTU

Search -> puredata

• Installieren von PD-Libraries wie Ggee

Search -> pd-ggee
Search -> pd-cyclone

• Verschiedene weitere Libraries sind zu empfehlen:

Jedenfalls: pd-comport, pd-zexy, pd-osc, pd-maxlib u.a.

Alle Packages "Mark for Installation" und anschließend "Apply" drücken.

PD starten

• entweder über LXLouncher -> Sound & Video -> PureData

• oder über File Manager (PCManFM) -> Applications -> Sound & Video -> PureData

• oder über LXTerminal -> pd -> ENTER eingegeben werden

• In manchen Fällen kann es nützlich sein das Programm als "SuperUser" zu öffnen, dann muss ins LXTerminal -> sudo pd -> ENTER eingegeben werden.
Die folgenden Ausführungen sind weitgehen im normalen Betriebsmodus ausgeführt.

PD-File abspeichern:

Falls, wie in meinem Fall, der PD-File auf dem Desktop liegt,
sollte der File so eingestellt sein, dass
er beim Doppelklick mit PD geöffnet wird.

- File mit rechter Mausttaste drücken,
bei Properties->Open with:
->Customize -> Sound & Video -> PureData einstellen.

Den File ohne Leerzeichen benennen!
(mit Leerzeichen will er nicht starten)

Des Weiteren muss der Suchpfad in den Preferences von PD auf die Externals verweisen:

Das ist immer auch dann notwendig, wenn im PD-Patch ein Object nur mit einer strichlierten Umrandung angedeutet wird.
Es bedeutet, dass das Object nicht gefunden wurde und entweder überhaupt erst durch zusätzliche Libraries installiert werden muss und/oder in den Preferences auf den Pfad der jeweiligen Library hingewiesen werden muss.

Der Speicherplatz eines Patches ist in diesem Zusammenhang nicht unerheblich. Wenn man auf Nummer sicher gehen möchte, sodass beim Start des Files alle Objects erkannt werden, könnte man die notwendigen Files der Libraries in den selben Ordner, wie den Patch legen.

In unserem Fall z.B. den shell.pd_linux -File aus der Ggee-Library.

Suchpfad einstellen:

• PureData starten -> Im Menü Media -> Preferences -> Path... -> New nach

/usr/lib/pd/extra/ggee

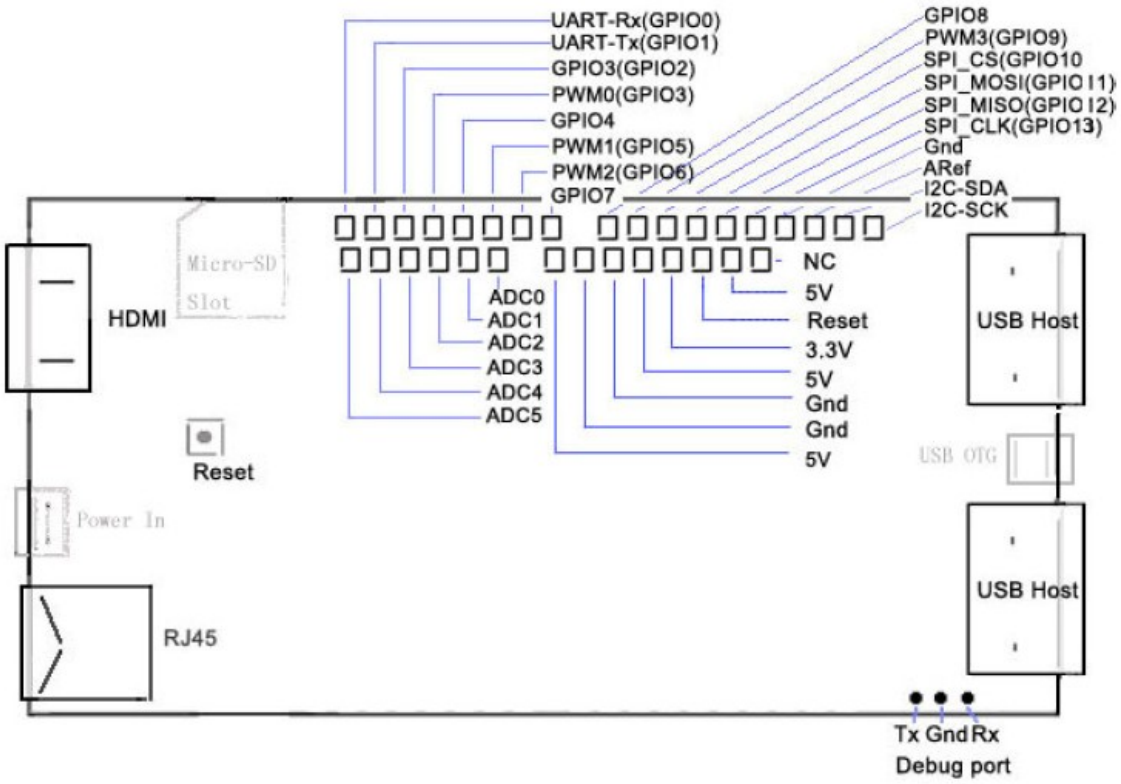
navigieren und "Apply" und "OK" drücken.

Das gleiche mit

/usr/lib/pd/extra/cyclone

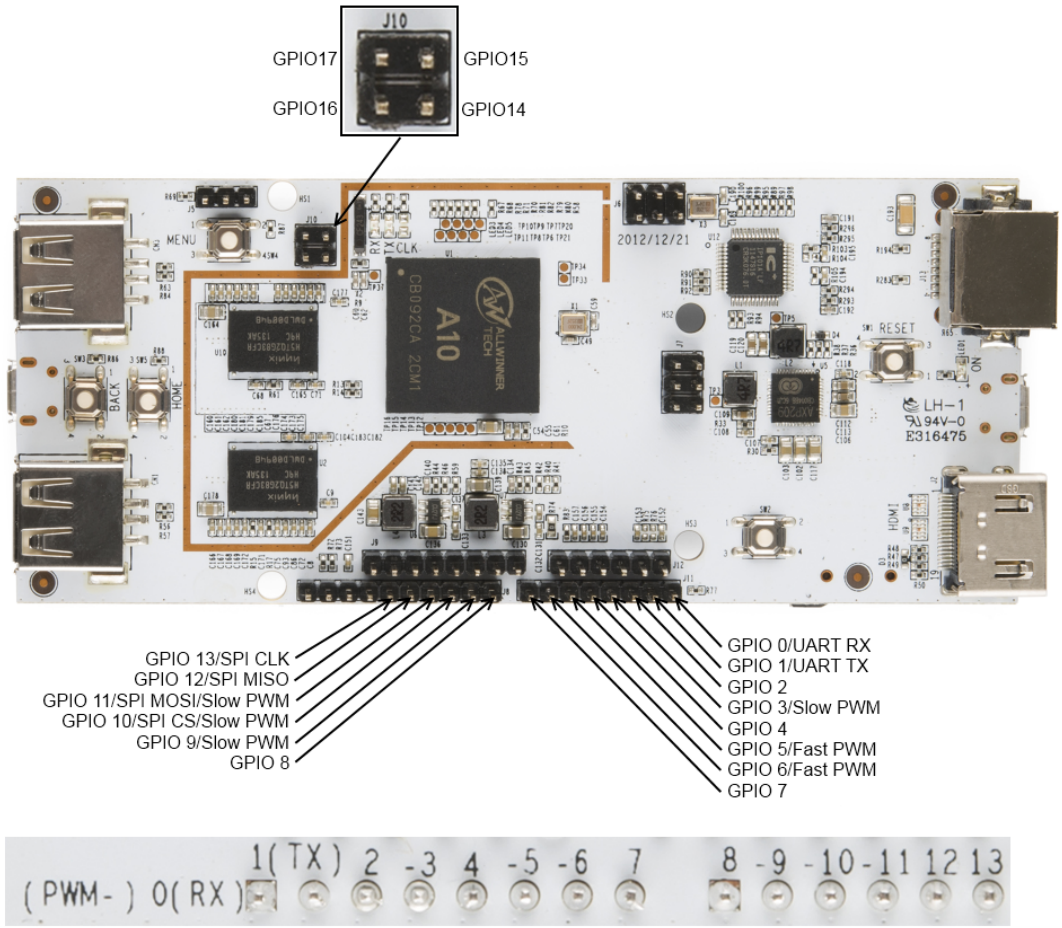
(ich hätte gehofft, dass
/usr/lib/pd/extra
für sämtliche Libraries gelten könnte?)

GPIOs und ADCs auf dem pcDuino



pcDuino Draufsicht
https://s3.amazonaws.com/pcduino/User+Guide/pcDuino+arduino+environment.pdf

oder zur allgemeinen Verwirrung eine ander Ansicht:



https://learn.sparkfun.com/tutorials/programmithe-pcduino/all

- Auf dem pcDuino befinden sich 14 GPIOs,
  die als digitale Ein- oder Ausgänge benutzt werden können
  plus 4 Pins zusätzlich auf Stecker (GPIO14-GPIO17)
  ACHTUNG: Die Stromstärke an den GPIOs darf 4mA nicht überschreiten!

- GPIO0 (GPIO Null) und GPIO1 können als UART-Rx und UART-Tx verwendet werden

- GPIO3, GPIO5, GPIO6 sowie GPIO9, GPIO10,GPIO11 können als PWM-Ausgänge benutzt werden.

"Pins 5 and 6 are true 520Hz 8-bit PWM;
the others are limited to a range of 0-20 at 5Hz."

das heißt für mich:
GPIO5 und GPIO6 steuern mit 520Hz 8-bit PWM;
GPIO3, 9, 10, 11 steuern mit einstellbarer 0-20bit-Genauigkeit mit 5Hz Abtastfrequenz.

"The high-level output for all the pins is 3.3V."

- Sechs Pins sind als Analogeingänge ADC0 bis ADC5 ausgeführt
  ADC0 und ADC1 lösen mit 6 Bit-Genauigkeit auf.

  dh.: 2 hoch 6 = 64 Stufen
  " returning a value from 0-63 over a range from 0-2V"

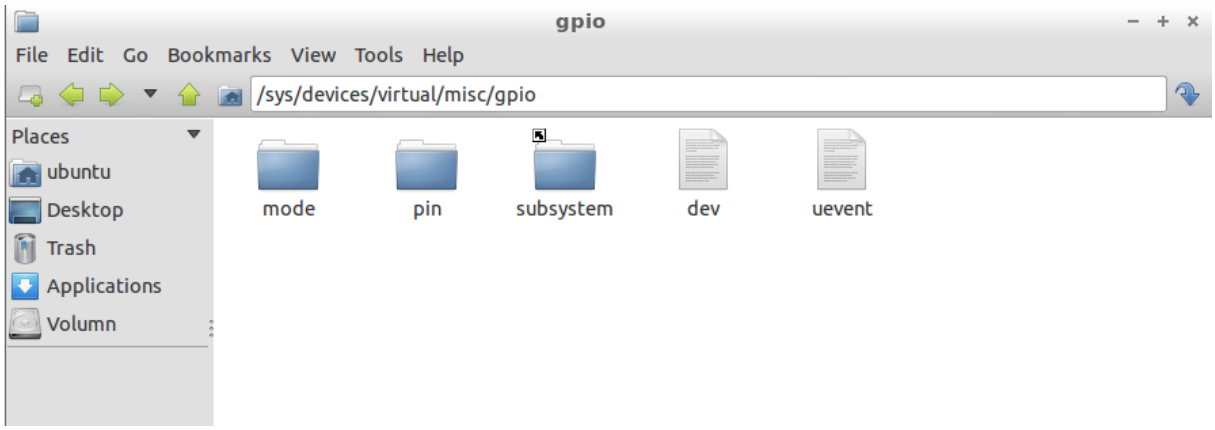
  ADC2 , ADC3, ADC4 und ADC5 lösen mit 12 Bit auf.

  dh.: 2 hoch 12 = 4096 Stufen
  "operating across the full 3.3V range"

- Des Weiteren sind 5V, 3,3V und GND ausgeführt

Zum GPIO-Verständnis

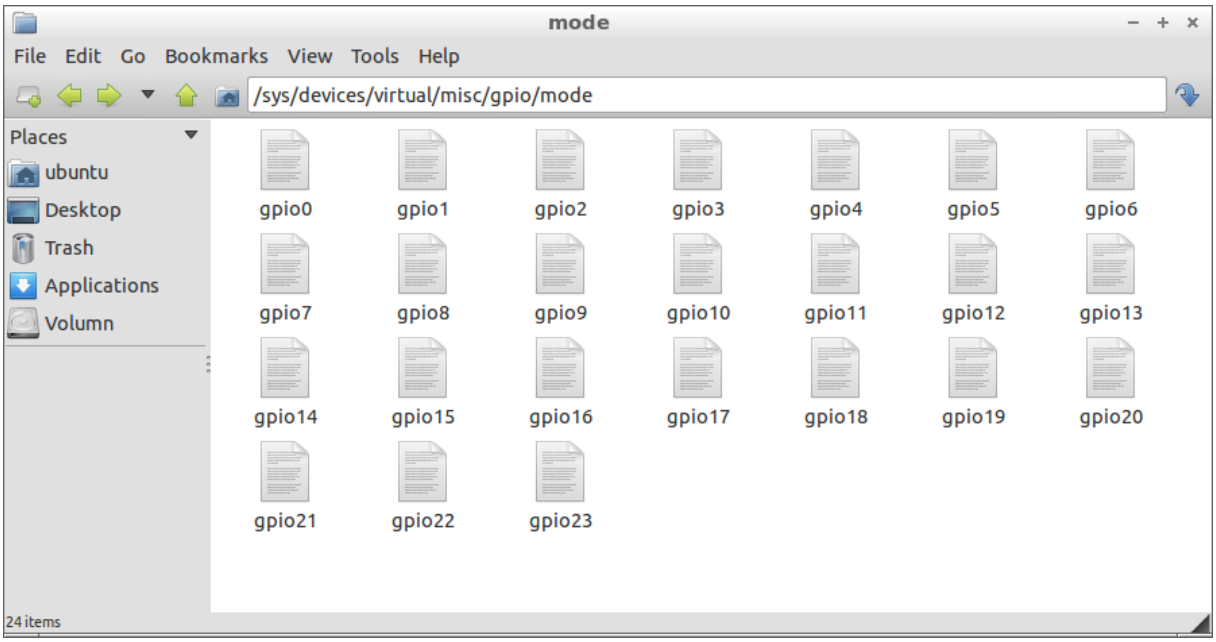
- Files nichts als Files
  In Linux ist alles File, sogar die Werte der Pins sind Files
  und liegen als Textfiles in Ordnern.
  (... diese Vorstellung hilft vielleicht...)



- GPIO-Mode
  Im GPIO-Mode wird festgelegt welche Funktion der jeweilige Pin hat.

... da jeder Pin frei als Aus- oder Eingang definiert werden muss gibt es dafür ein File im Verzeichnis:

```
/sys/devices/virtual/misc/gpio/mode/
```

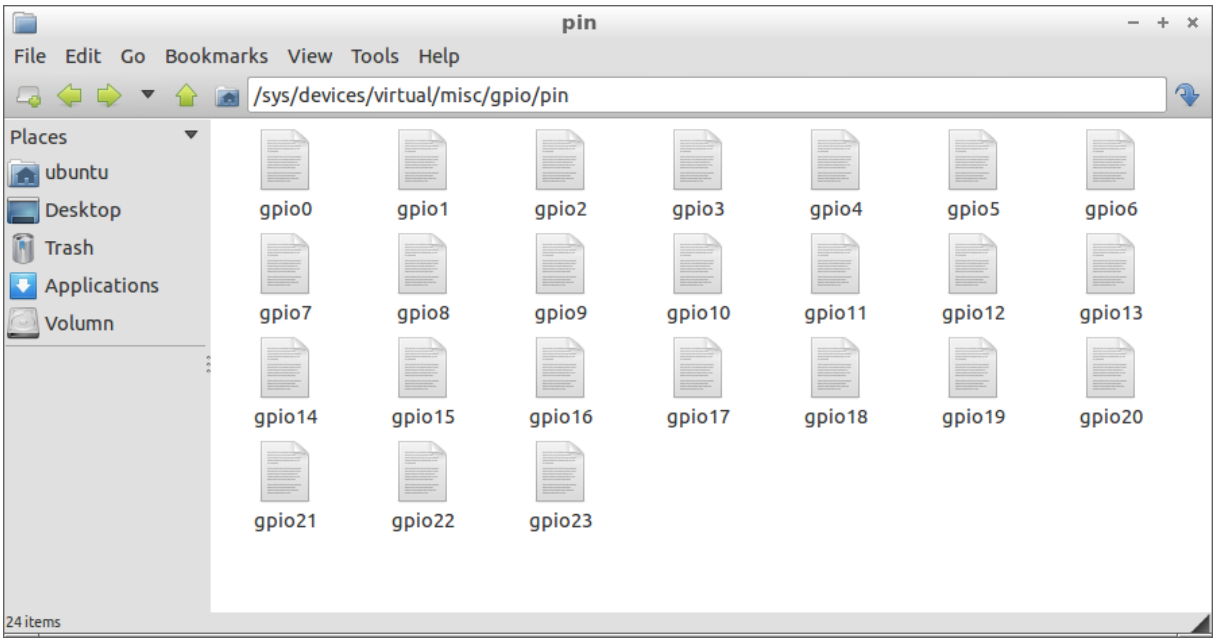


hier liegen die Files: GPIO0, GPIO1....GPIO23
und darin sind Werte von 0 bis angeblich möglichen 8? geschrieben.
(Als Text!!)

Wir brauchen:
0 für: dieser Pin ist EINGANG
1 für: dieser Pin ist AUSGANG
2 für: dieser Pin ist EINGANG mit Pullup-Widerstand
3 für: dieser Pin ist eine Serielle Schnittstelle (Tx oder Rx)
... die restlichen Modes habe ich bis jetzt noch nirgends gefunden....

• GPIO-Pin
hier steht geschrieben, ob dieser Pin ein oder ausgeschaltet ist.

`/sys/devices/virtual/misc/gpio/pin/`



... es kann geschrieben oder gelesen werden...

hier liegen ebenfalls die Files: GPIO0, GPIO1....GPIO23
und speichern Werte von 0 oder 1 (wieder in Textform)

Die einfachste aber mühsamste Art, die Pins anzusteuern oder abzufragen ist, die Textfiles zu ändern oder zu lesen.
D.h.: Text-File im Verzeichnis aufsuchen, auslesen oder schreiben und speichern.

Aufgepasst:
Voreingestellt sind die GPIOs als Eingang.
Das heißt in den Mode-Files sind 0 (Nullen) abgespeichert.

Wenn nun ein Pin ein- und ausgeschaltet werden soll
(also als Ausgang funktionieren soll)
muss man vorher z.B den GPIO0 - File (nehmen wir den ersten GPIO)
in

```
/sys/devices/virtual/misc/gpio/mode/
```

öffnen und den Wert auf 1 ändern und abspeichern.
Anschließend kann man den GPIO0-File
in

```
/sys/devices/virtual/misc/gpio/pin/
```

öffnen und 0 für AUS oder 1 für EIN eingeben und abspeichern.

Eine LED am GPIO0-Pin müsste damit geschaltet werden.

Diese Prozedur erscheint mir für das erste Verständnis nützlich.

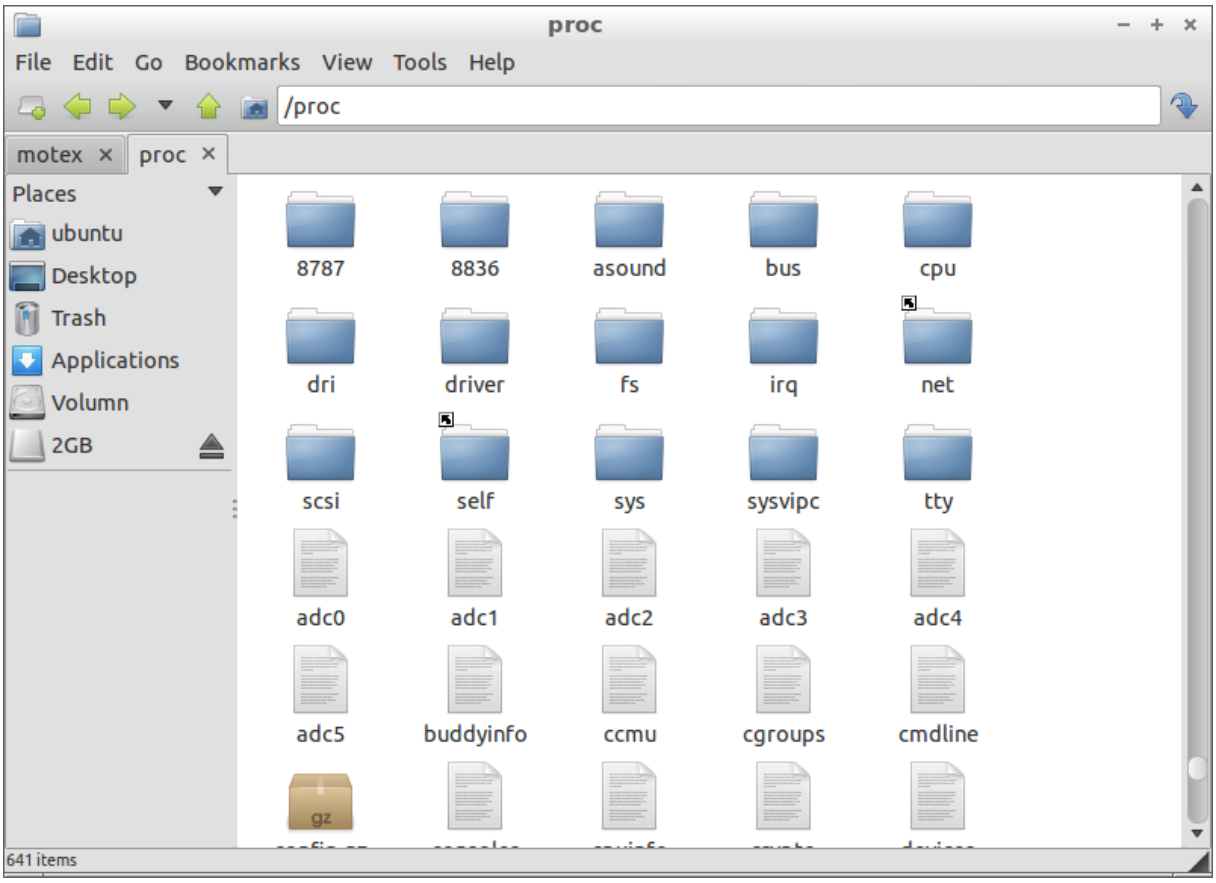
ACHTUNG:
Die Spannungswerte der Pins dürfen 3,3 Volt nicht überschreiten.
Bzw.: High-Pegel sind 3,3V hoch.
Bei Verbindung mit 5V-Geräten muss hier ein Adapter gebaut werden.

ADC-Files
(die Werte der Analogeingänge)

- Werte lesen
  Die Werte der ADC0 bis ADC5-Eingänge können im Verzeichnis

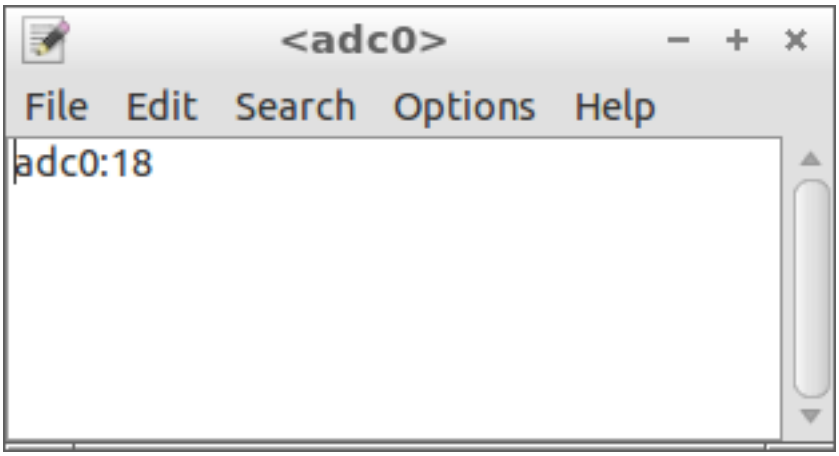
  ```
  /proc
  ```

  in den entsprechenden Textfiles gelesen werden



hier besteht die Herausforderung, dass der Text so aussieht:

adc0:18

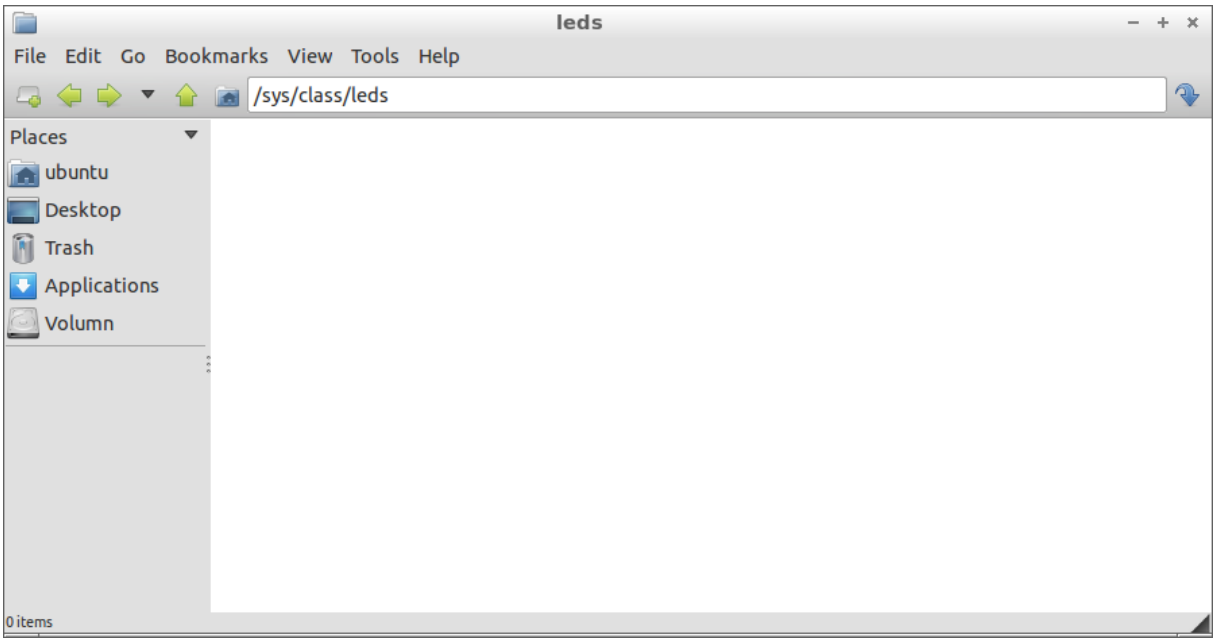


der Wert nach dem Doppelpunkt ist für uns interessant
Lösungsvorschlag weiter unten...

• Werte von PWM0 bis PWM6 können angeblich nach

```
/sys/class/leds/pwm0/brightness
```

geschrieben werden



(dieser Punkt konnte noch nicht nachvollzogen werden, weil der Ordner leer ist ?)

```
## Let's dim some LEDs! The method for controlling a PWM pin on the pcDuino is
##   to send to the command interpreter (via os.system() this command:
##   echo <value> > /sys/class/leds/pwmX/brightness
##   where <value> varies from 0 to the maximum value found in the
##   max_brightness file, and X can be from 0-5.
```

https://learn.sparkfun.com/tutorials/programming-the-pcduino/all#analog-input-and-output

Ansteuern der GPIOs und ADCs mit Kommandos

• Steuern mit der Kommandozeile

Mit dem LXTerminal können die Files mit einem Befehl angesprochen und dadurch verändert oder ausgelesen werden:

(Achtung Leerzeichen beachten!)

• Ändern des Pin-Modes (von GPIO0 bis GPI023) bzw. setzen als EINGANG

```
echo 1 > /sys/devices/virtual/misc/gpio/mode/gpio0
echo 1 > /sys/devices/virtual/misc/gpio/mode/gpio1
.
.
.
echo 1 > /sys/devices/virtual/misc/gpio/mode/gpio23
```

• Ändern des Pin-Wertes von GPIO0 EIN

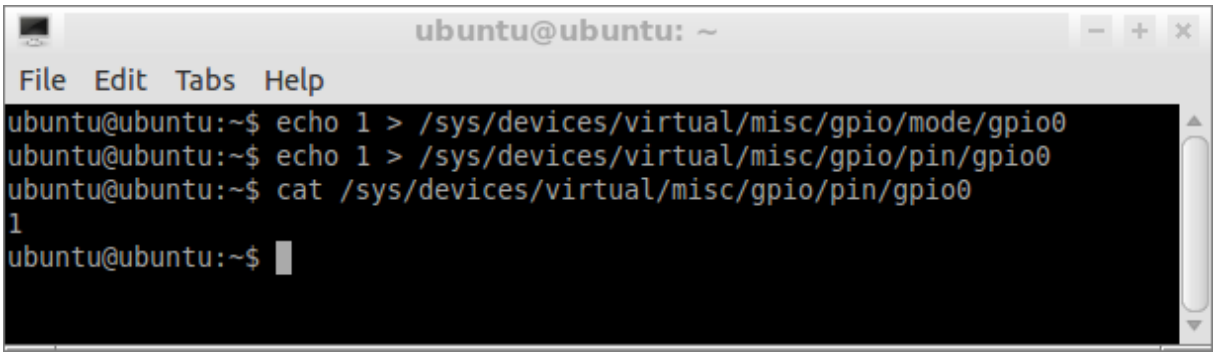
echo 1 > /sys/devices/virtual/misc/gpio/pin/gpio0

• Ändern des Pin-Wertes von GPIO0 AUS

echo 0 > /sys/devices/virtual/misc/gpio/pin/gpio0

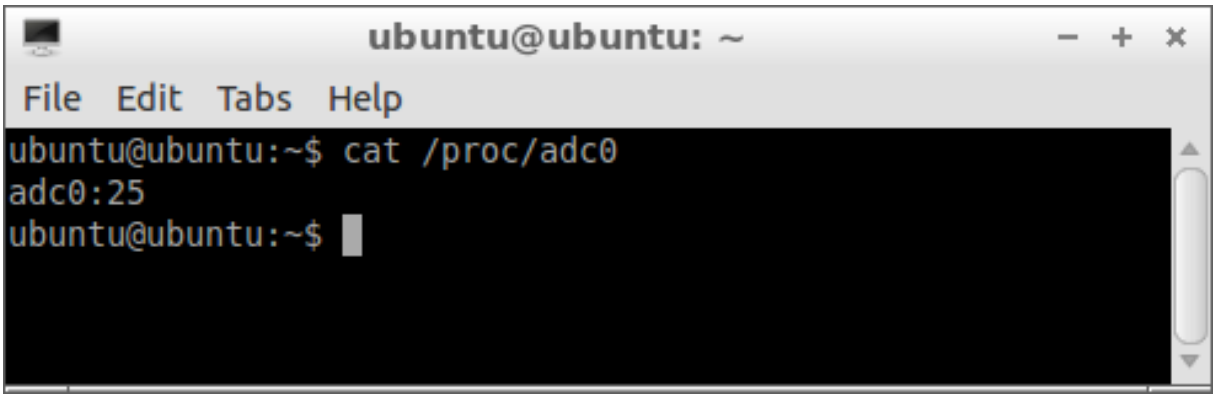
• Auslesen des Pin-Wertes von GPIO0

cat /sys/devices/virtual/misc/gpio/pin/gpio0



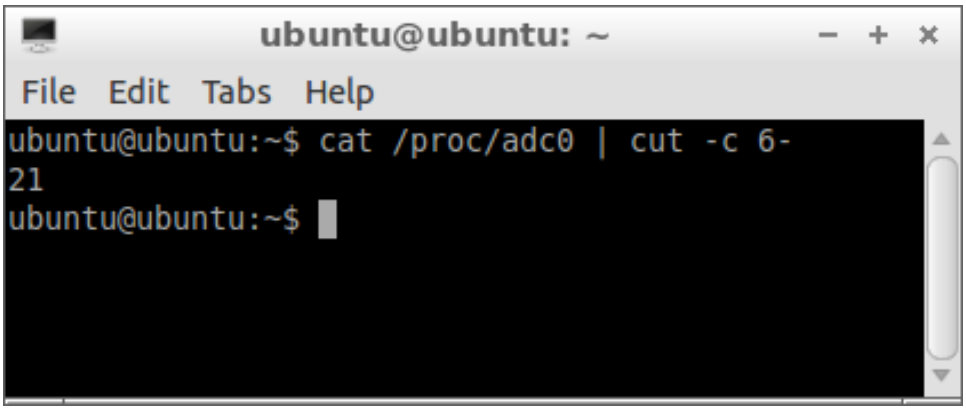
• Auslesen des ADC0-Pin-Wertes

cat /proc/adc0



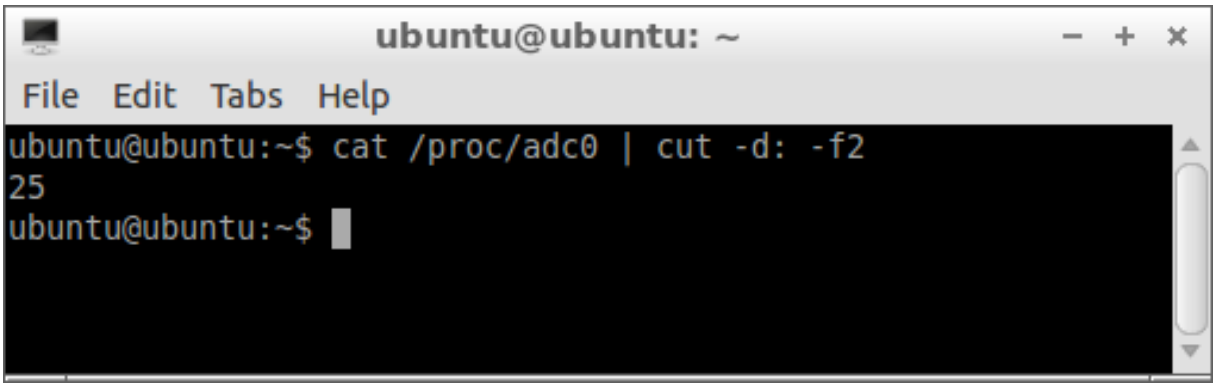
hier ist die vollständige analoge Information:
ADC-Pin-Nummer + "Doppelpunkt" + Wert.

Diese Information kann aufgeteilt werden:
Hier bieten sich verschiedene Varianten an:
Variante I:



Alles ab dem 6. Zeichen wegschneiden.

VarianteII:



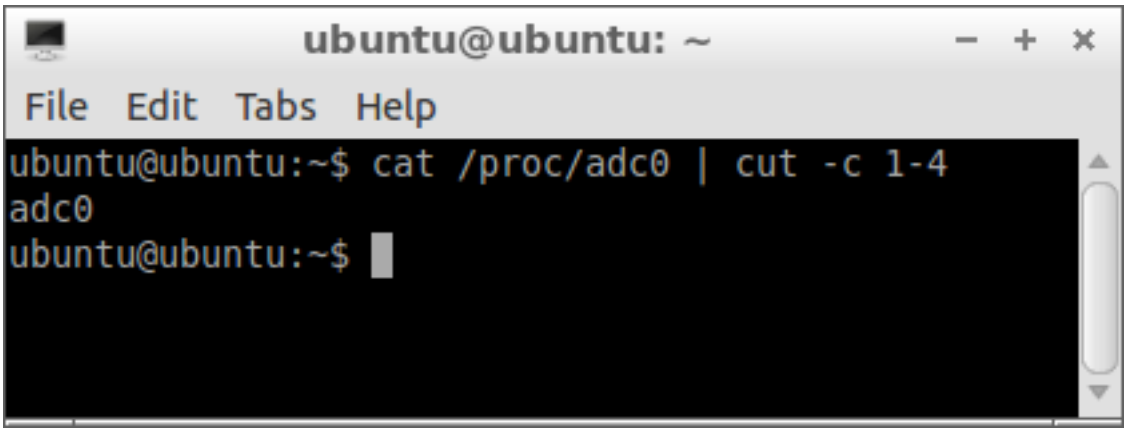
Das Zweite Feld nach dem ":" stehen lassen.

Um diese Möglichkeiten von cut kennenzulernen

cut --help

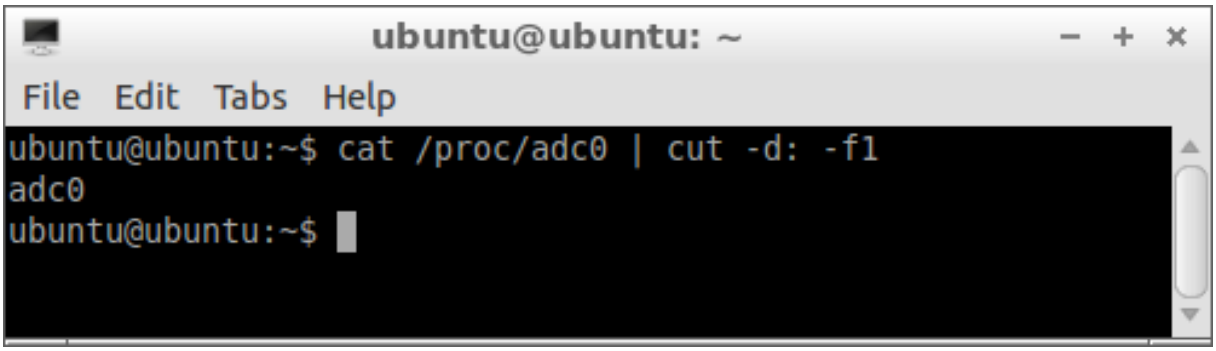
ins LXTerminal eingeben.

Hier noch Varianten um Teile vor dem Doppelpunt auszugeben:
Variante I:



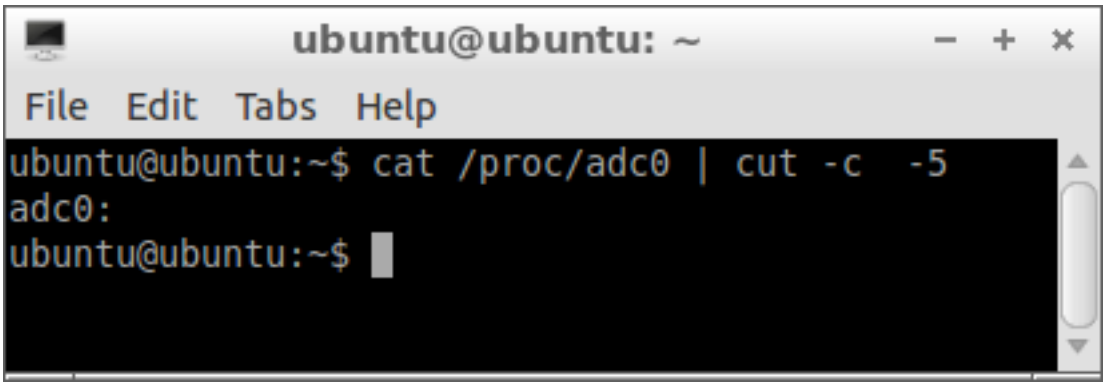
Das erste bis zum vierten Zeichen stehen lassen.

Variante II:



Das erste Feld vor dem ":" stehen lassen.

Variante III:



Alles bis zum 5. Zeichen stehen lassen.

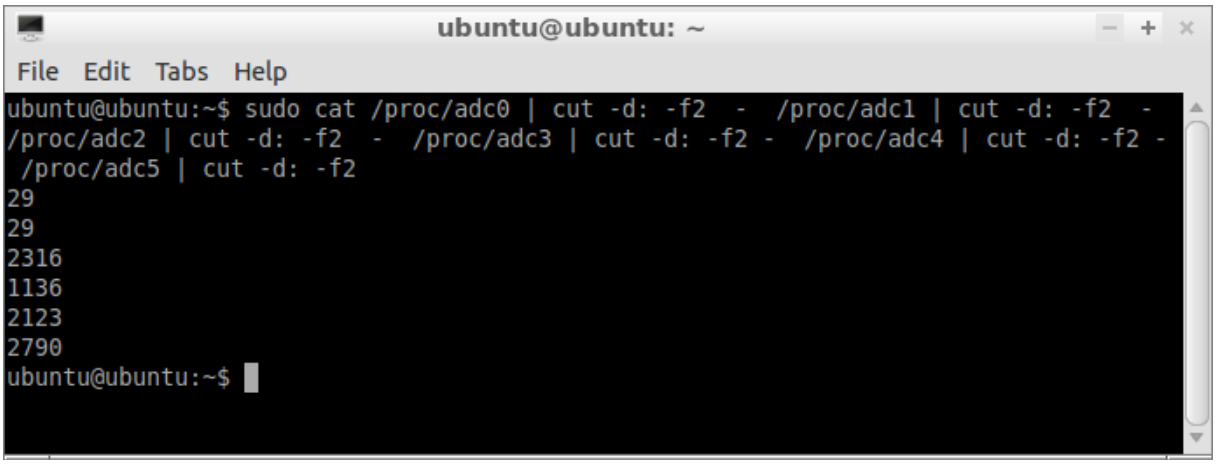
Wir haben also mit cat und mit anderen Linux-Kommandos schöne Möglichkeiten um Zeichenketten beliebig aufzusplitten.
Das Pipe-Symbol "|" wird unter Lubuntu mit der rechten Alt-Taste (ALT GR) + < erzeugt. (anders bei OSX mit Alt+7)

• Auslesen aller ADC-Pin-Werte:

Wenn wir mehrere Sensoren abfragen wollen, ist mein Vorschlag, dass wir es gleichzeitig tun.
Hier möchte das LXTerminal, dass wir sudo eingeben:

sudo cat /proc/adc0 | cut -d: -f2 - /proc/adc1 | cut -d: -f2 .... usw

Die einzelnen Sensor-Abfragen werden gleichzeitig mit cat abgefragt indem sie durch einen Bindestrich "-" aneinandergekettet werden.



• Schreiben des PWM0-Pin-Wertes

Es könnte heißen:

echo 127 > /sys/class/leds/pwm0/brightness

(PWM funktioniert noch nicht, wie es sollte
... irgendwo ist hier noch ein Fehler begraben
... diese Files sollten existieren damit sie beschrieben werden können... tun sie aber nicht)

• Steuern mit Shell-Skript

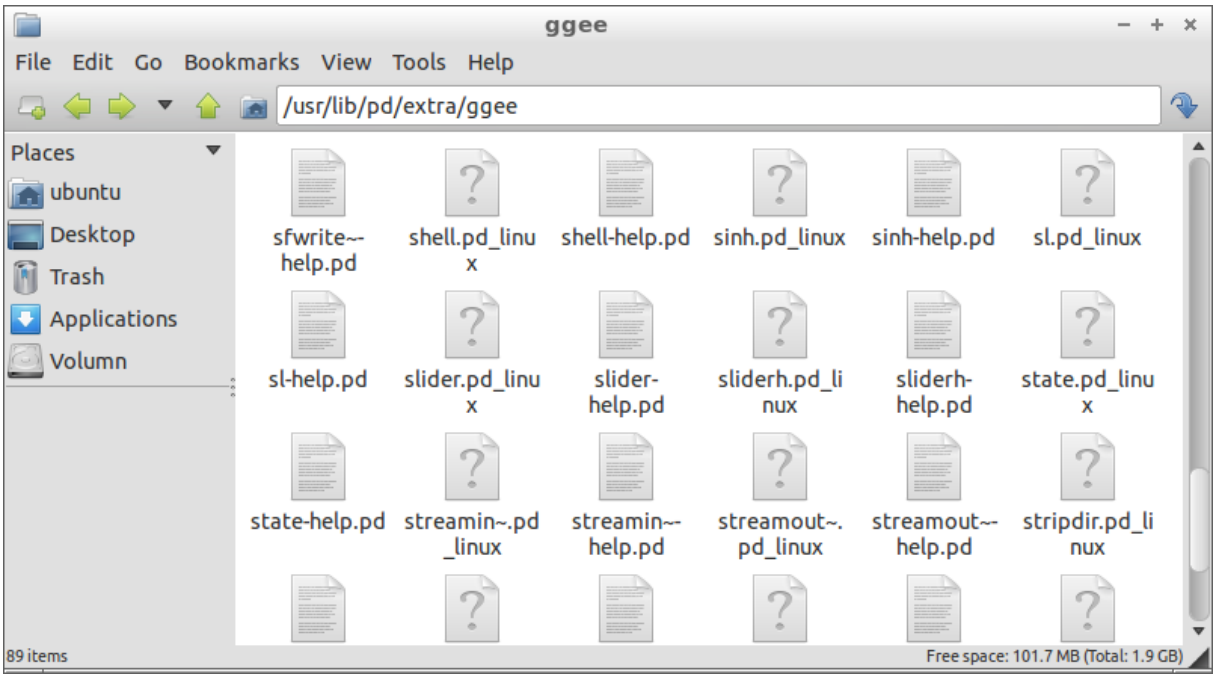
Abläufe können mit Skripts geschrieben, und diese dann in der Kommandozeile aufgerufen werden ... in der nächsten Folge ...

Vorgangsweise zum Steuern mit PD:

Es gibt in PD ein shell-Object.
Zu finden in der Ggee-Library.
(diese kann einfach über den Synaptik Package Manager installiert werden)

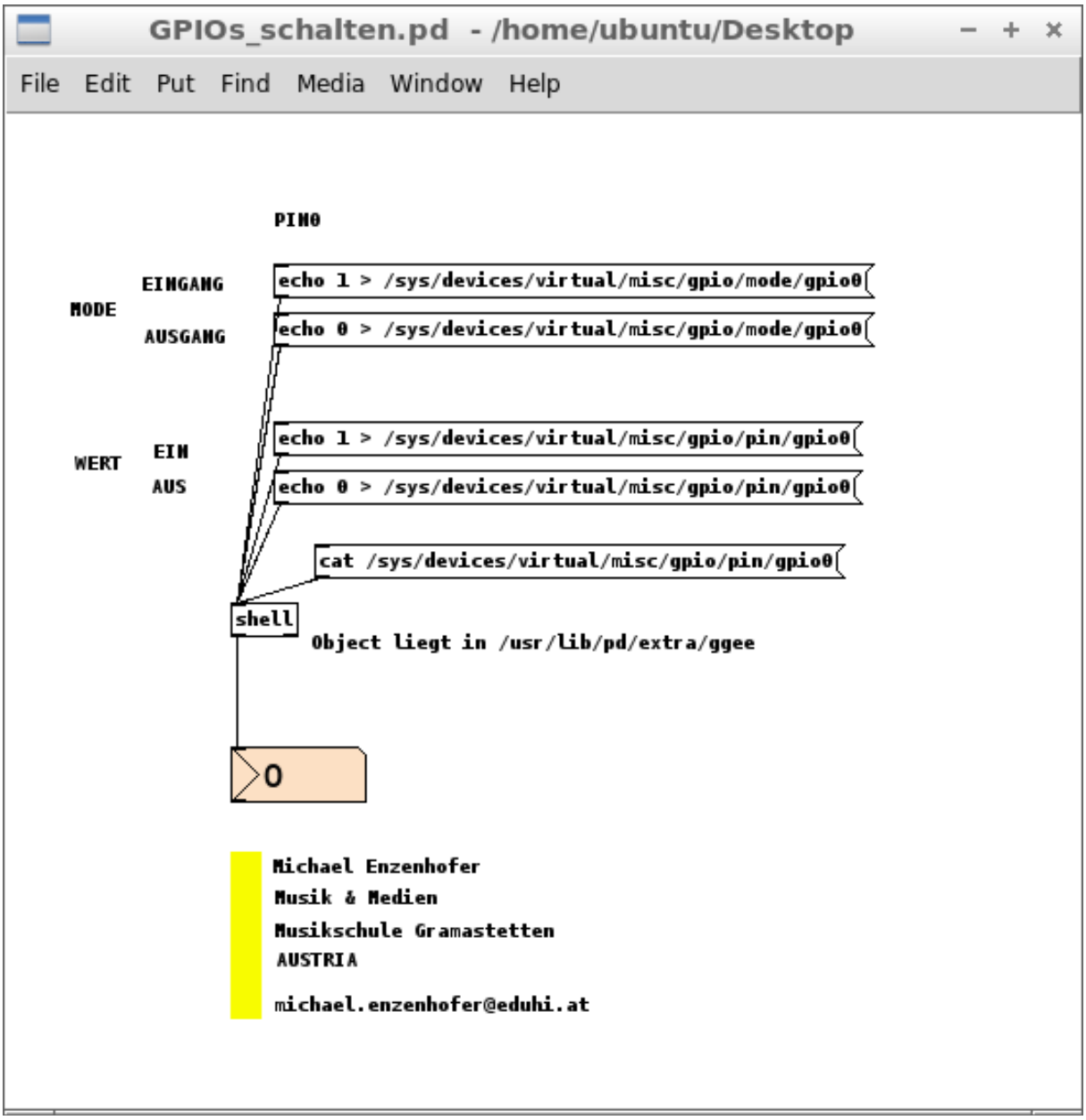
Suche nach pd-ggee

Das installierte pd-ggee liegt in diesem Pfad und shell-pd_linux liegt in diesem Ordner:



In PD können Massage-Boxen mit den Terminal-Befehlen an das Shell-Object gesendet werden und bewirken daher dasselbe, wie wenn die Befehle direkt vom Terminal gesendet würden.

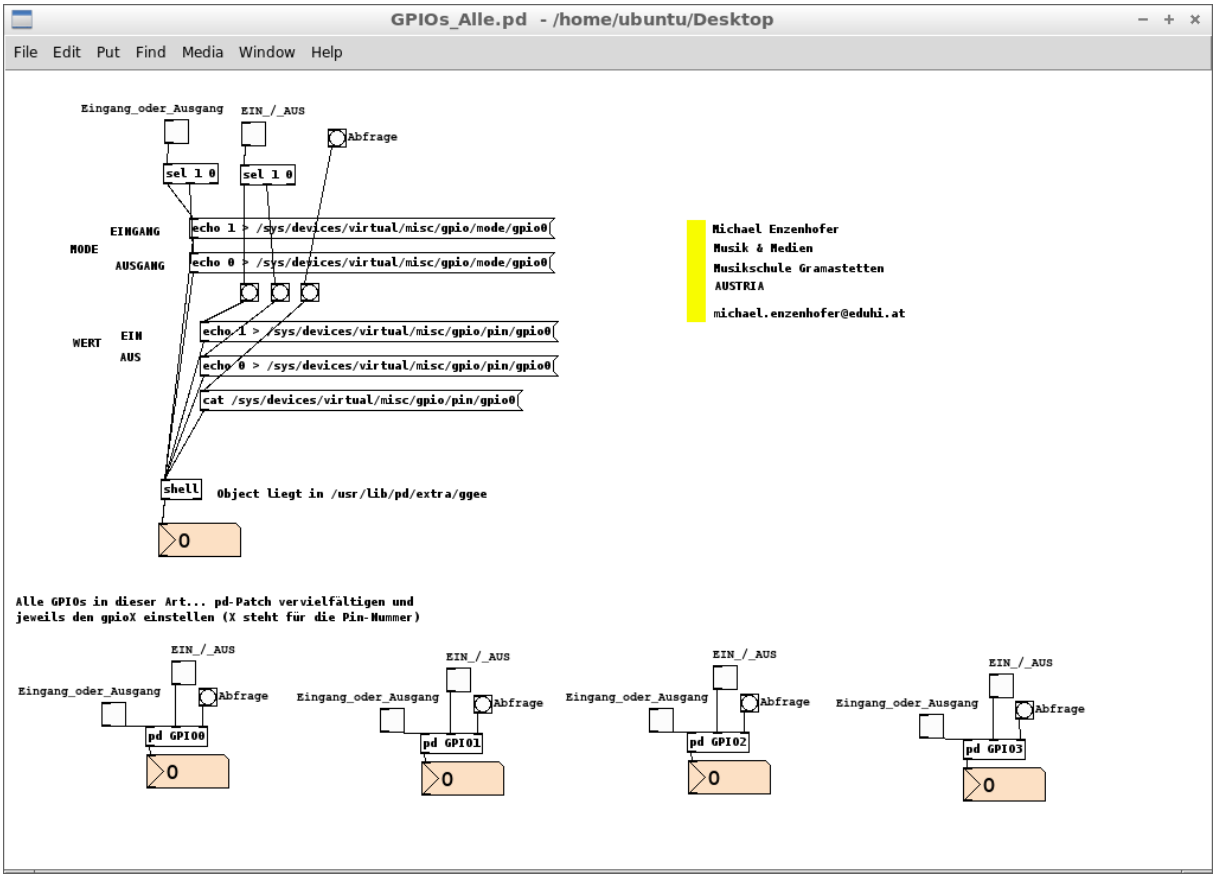
• Ansteuern der GPIO´s
• Abfragen der GPIO´s



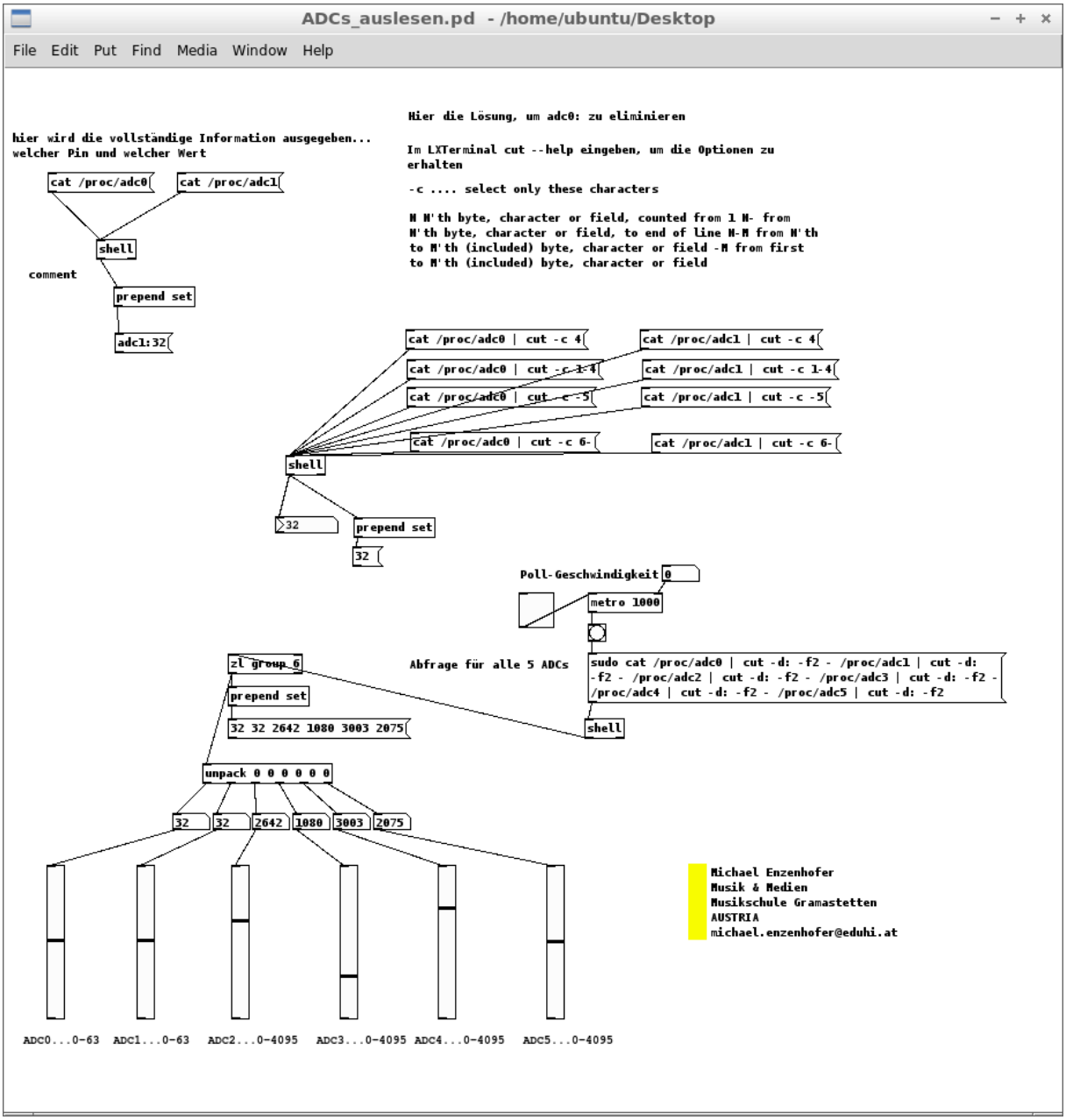
Zum laufenden Auslesen könnte mit einem metro-Object "gepollt" werden.
(Die Poll-Geschwindigkeit muss mit Versuchen ermittelt werden.
Je nachdem wie viele Aus- und Eingänge angeschlossen werden, wird es Grenzen geben. Mein Vorschlag 500ms.

Hier ist zu bedenken, dass in der Folge mit Shell-Skripts noch optimiert werden kann.
Weiters ist die Abfrage der Pins durch Pollen noch nicht optimal.
Abhilfe müsste die Definition der GPIOs als Interrupts zu Verbesserungen führen. (Hier habe ich noch keine zulänglichen Infos gefunden)

Hier wurde der Patch in ein pd-Object gesteckt.
Dort muss jeweils der letzte /gpioX - wobei X für die Pin-Nummer steht ausgetauscht werden.
Es sollten die Massages direkt mit dem shell-Object, und nicht mit send- und receive-Objekten verbunden werden.
Es könnte zu Konflikten kommen wenn gleichzeitig auch die Analogen Eingänge abgefragt werden.



• Analog-Ein



ACHTUNG: Die Eingangspannung von 3,3V nicht überschreiten!

• Analog-Abfrage
(hier muss für´s Erste auch "gepollt" werden)

• ACHTUNG: Zumindest bei mir liefern die ersten beiden Analogeingänge die gleichen Werte, unabhängig davon, bei welchem Eingang (ADC0 oder ADC1) ich die Potentiometer bewege.

• PWM (Analog-Aus)
(noch nicht geklärt)

Zusammenfassung der verwendeten Terminal-Kommandos für unsere Zwecke:

• echo
• cat
• cut

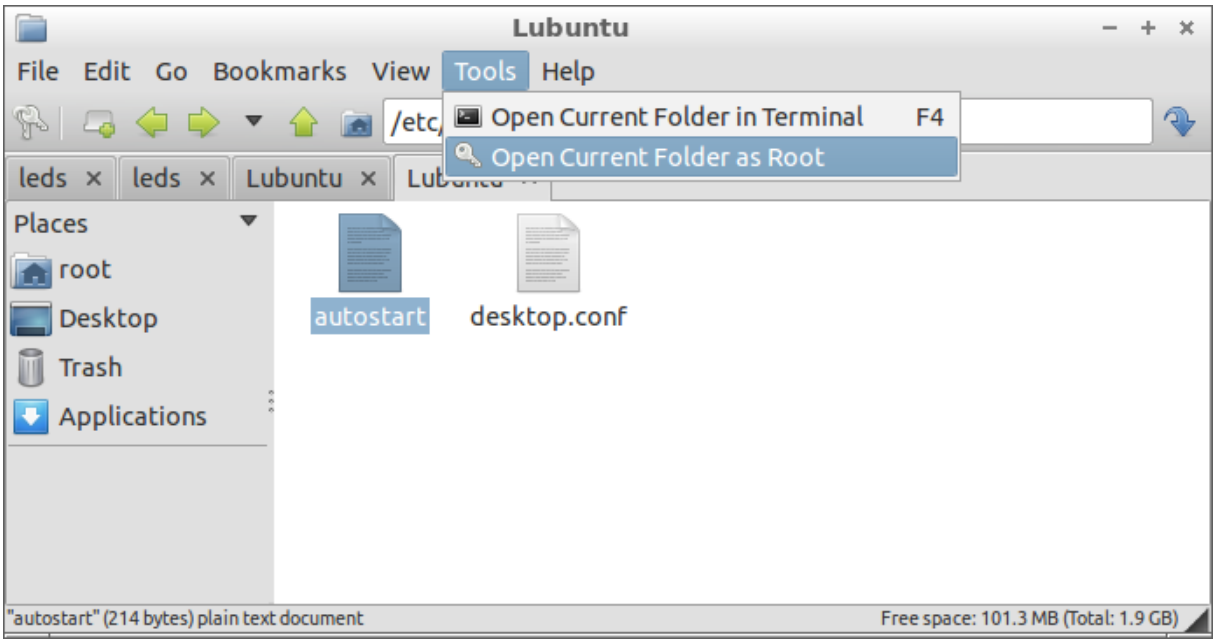
Sonstige Nützlichkeiten:

• Automatischer Programmstart

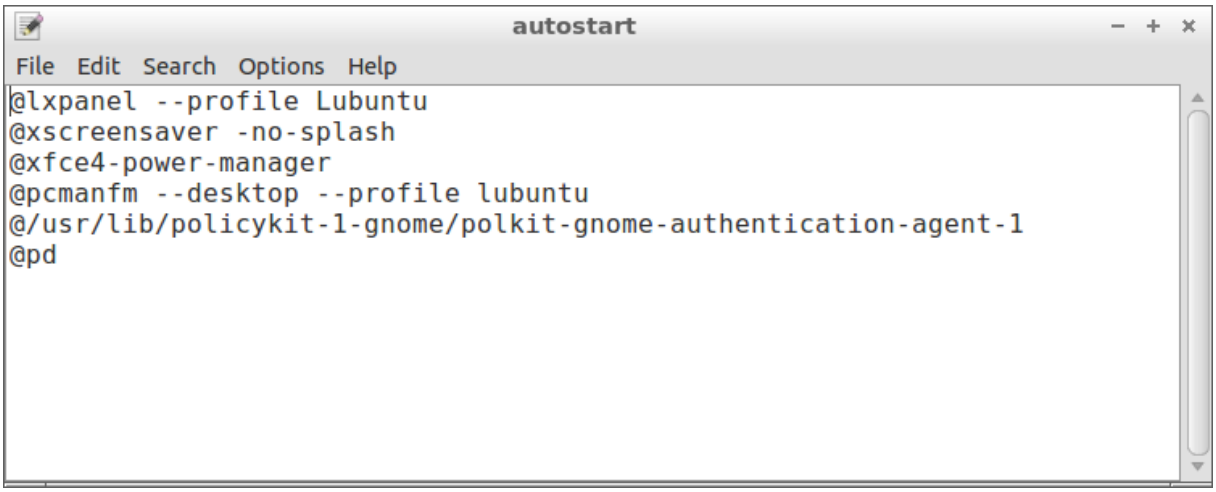
Den autostart-Textfile in

/etc/xdg/lxsession/Lubuntu/autostart

als Root öffnen und das gewünschte Programm im Text eingeben

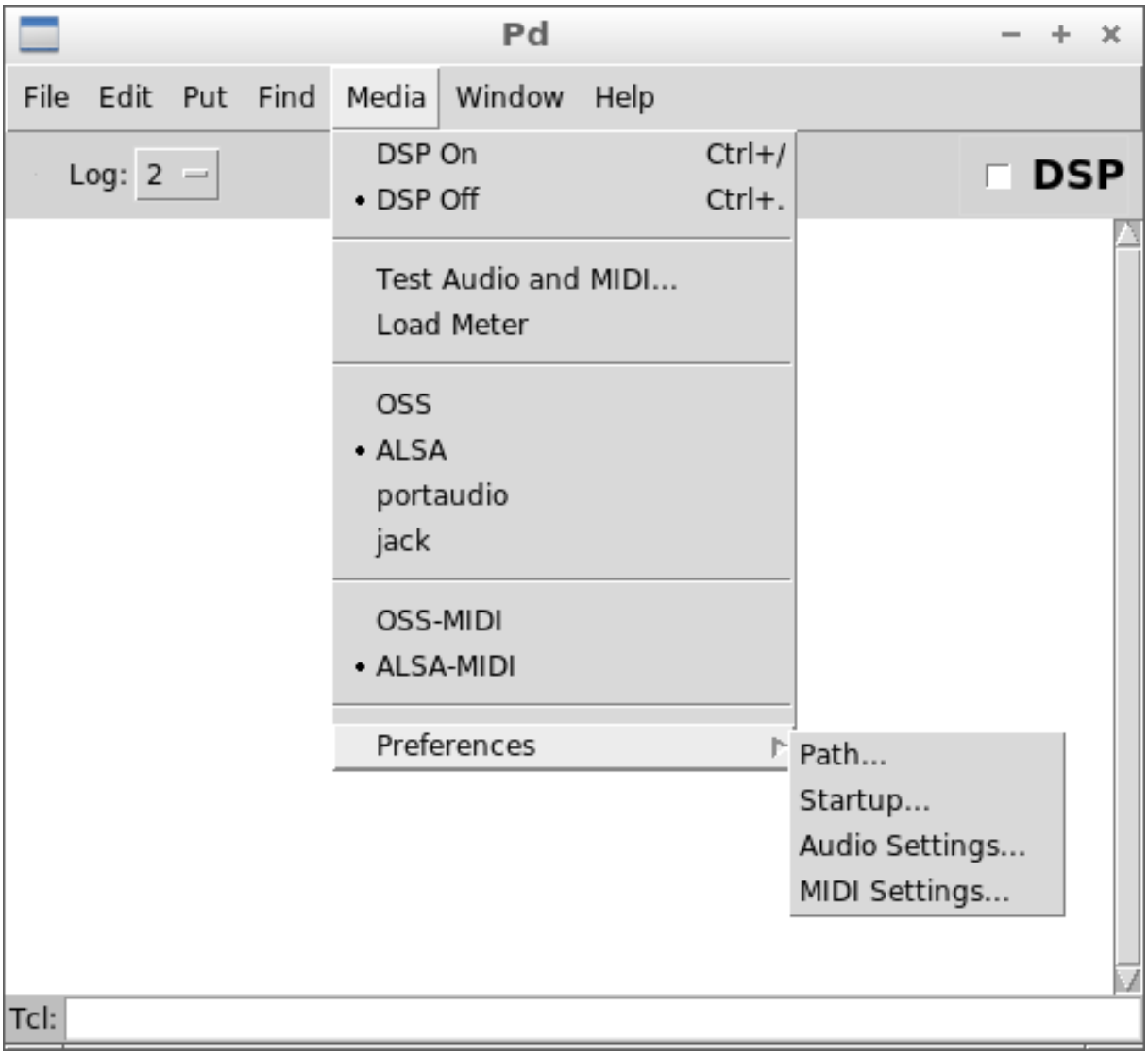


Es wird in unserem Fall @pd in die nächste Zeile eingegeben.



Beim nächsten Boot-Vorgang wird PD automatisch gestartet.

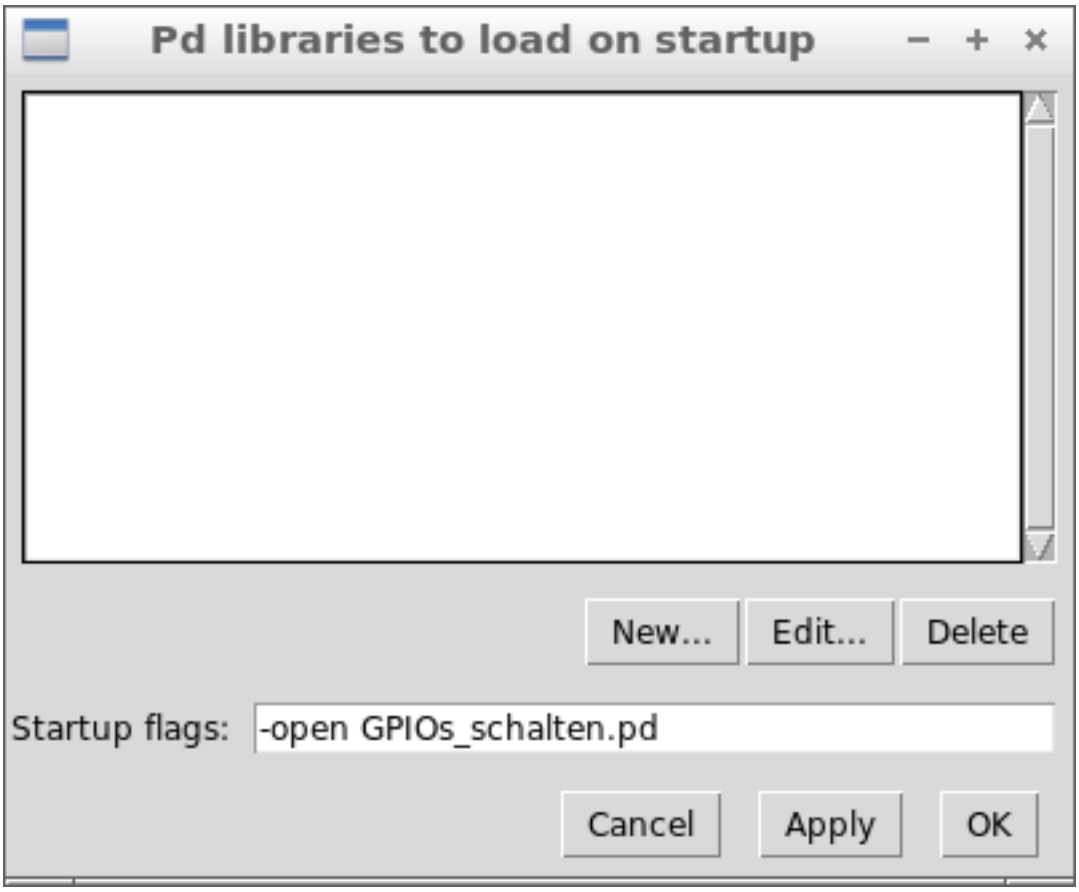
Damit nun auch ein PD-File automatisch startet, muss in den PD-Preferences ein sogenannter Startup-Flag gesetzt werden.



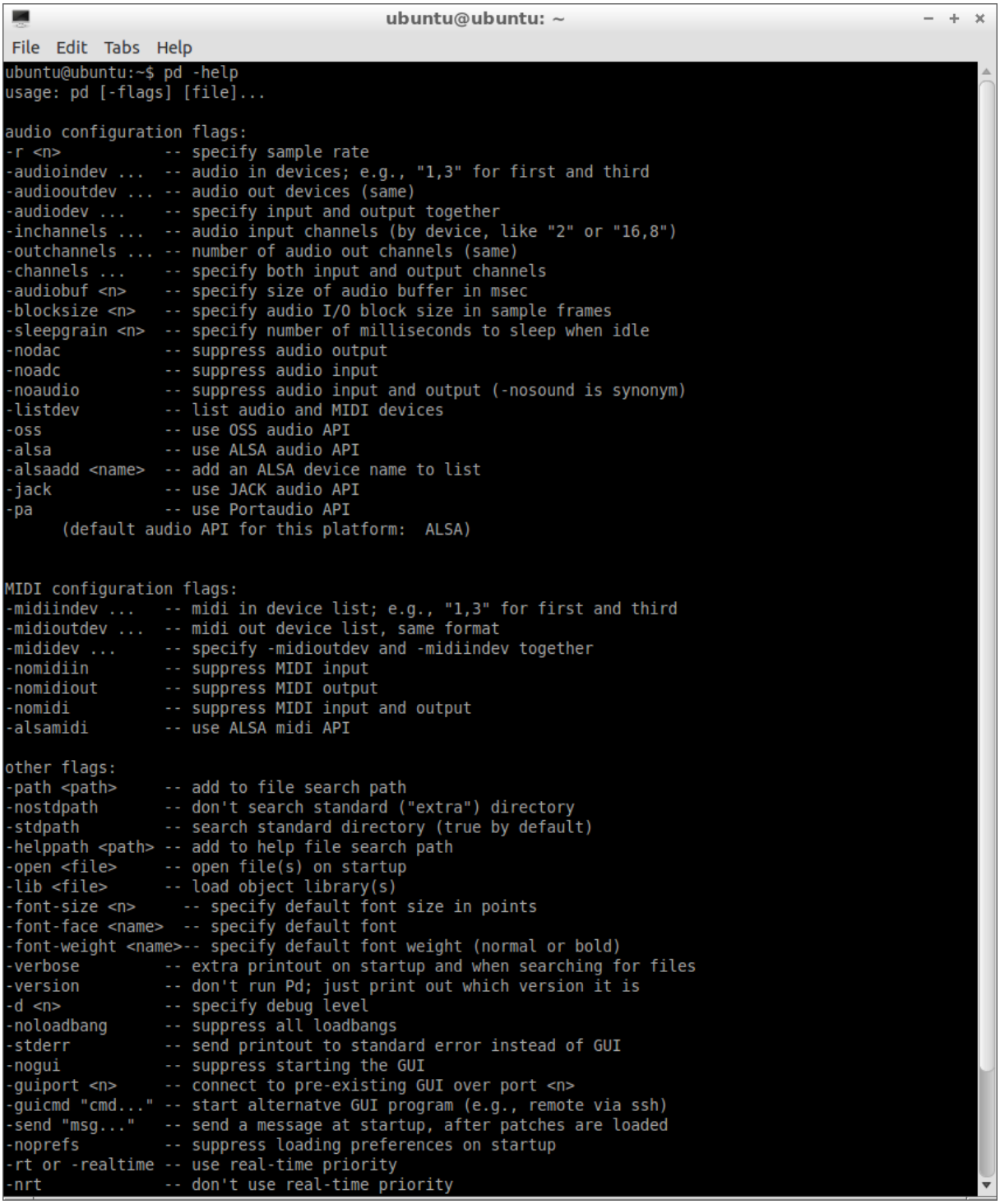
Media -> Preferences -> Startup wählen und -open Startup flag setzen.

Der PD-File, der hier automatisch starten soll hat die Bezeichnung:

GPIOs_schalten.pd



Alle möglichen Startup flags können ausgelesen werden, indem im LXTerminal pd -help eingegeben wird:


ubuntu@ubuntu: ~
File Edit Tabs Help
ubuntu@ubuntu:~$ pd -help
usage: pd [-flags] [file]...
audio configuration flags:
-r <n> -- specify sample rate
-audioindev ... -- audio in devices; e.g., "1,3" for first and third
-audiooutdev ... -- audio out devices (same)
-audiodev ... -- specify input and output together
-inchannels ... -- audio input channels (by device, like "2" or "16,8")
-outchannels ... -- number of audio out channels (same)
-channels ... -- specify both input and output channels
-audiobuf <n> -- specify size of audio buffer in msec
-blocksize <n> -- specify audio I/O block size in sample frames
-sleepgrain <n> -- specify number of milliseconds to sleep when idle
-nodac -- suppress audio output
-noadc -- suppress audio input
-noaudio -- suppress audio input and output (-nosound is synonym)
-listdev -- list audio and MIDI devices
-oss -- use OSS audio API
-alsa -- use ALSA audio API
-alsaadd <name> -- add an ALSA device name to list
-jack -- use JACK audio API
-pa -- use Portaudio API
(default audio API for this platform: ALSA)
MIDI configuration flags:
-midiindev ... -- midi in device list; e.g., "1,3" for first and third
-midioutdev ... -- midi out device list, same format
-mididev ... -- specify -midioutdev and -midiindev together
-nomidiin -- suppress MIDI input
-nomidiout -- suppress MIDI output
-nomidi -- suppress MIDI input and output
-alsamidi -- use ALSA midi API
other flags:
-path <path> -- add to file search path
-nostdpath -- don't search standard ("extra") directory
-stdpath -- search standard directory (true by default)
-helppath <path> -- add to help file search path
-open <file> -- open file(s) on startup
-lib <file> -- load object library(s)
-font-size <n> -- specify default font size in points
-font-face <name> -- specify default font
-font-weight <name>-- specify default font weight (normal or bold)
-verbose -- extra printout on startup and when searching for files
-version -- don't run Pd; just print out which version it is
-d <n> -- specify debug level
-noloadbang -- suppress all loadbangs
-stderr -- send printout to standard error instead of GUI
-nogui -- suppress starting the GUI
-guiport <n> -- connect to pre-existing GUI over port <n>
-guicmd "cmd..." -- start alternatve GUI program (e.g., remote via ssh)
-send "msg..." -- send a message at startup, after patches are loaded
-noprefs -- suppress loading preferences on startup
-rt or -realtime -- use real-time priority
-nrt -- don't use real-time priority

Noch zu erledigen:

- PWM-Ansteuerung
- Die serielle Ansteuerung

Vermutung:

Es müssten die GPIO-Files von GPIO0 und GPIO1 eliminiert werden, damit sie frei für serielle Daten werden.
Das gelingt mir derzeit noch nicht.
Als Superuser funktioniert es auch nicht.
Die Files im /sys - Verzeichnis sind virtuelle Files, welche eigentlich gar nicht vorhanden sind. Diese können auf andere Files verweisen, die auch andere Namen tragen können... oder so ähnlich...
Hier fehlt mir noch das genaue Verständnis der Linux-Welt.
Sogenannte export - und unexport - Befehle sind hier noch wirkungslos??