

Beispielprogramm I2C mit RASP32

- RUN WiringPi mit Hardware-Aufbau BMP180

Samstag, 16. März 2024 02:36

Generelle Vorbedingungen für die Installation des RASP32:

- RASP32 update auf Linux Kernel 6.6.21 mit Bulleyes!
- Zugriffsrechte für GPIO neu setzen:

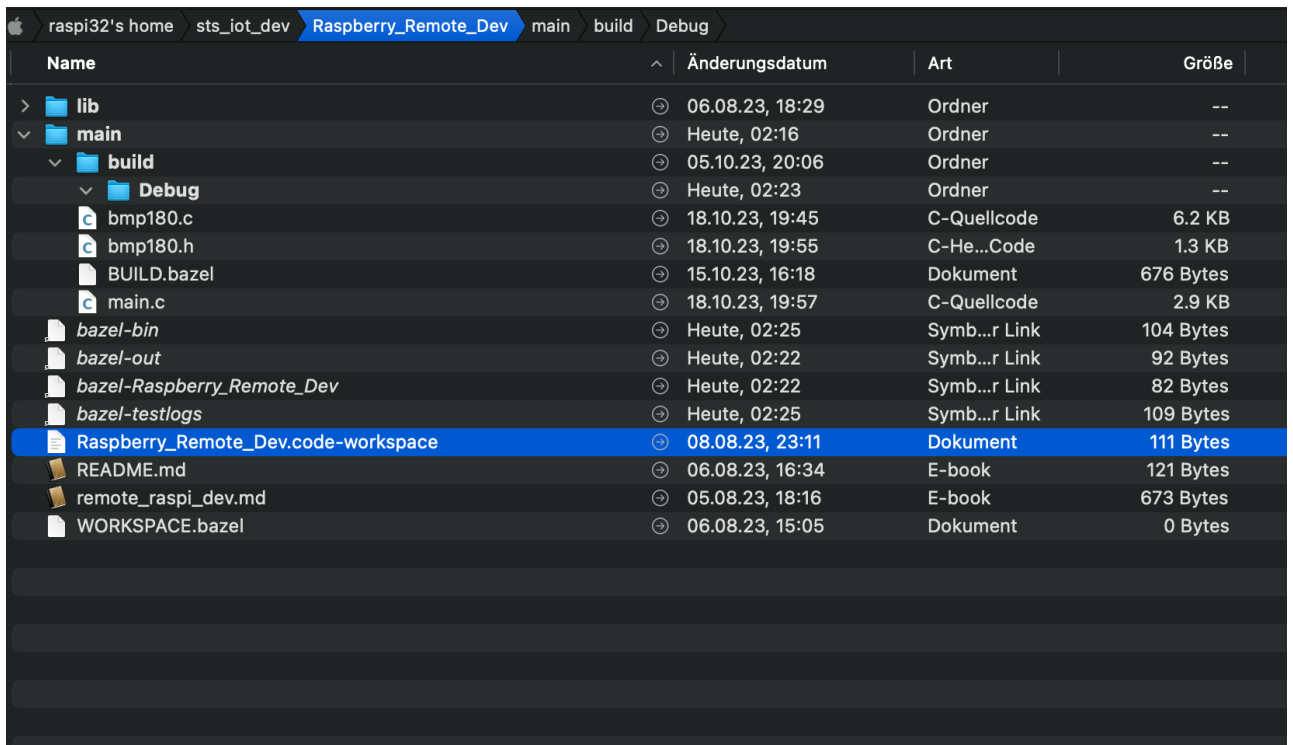
```
pi@raspi32bit:~$ sudo rpi-update
*** Raspberry Pi firmware updater by Hexxeh, enhanced by AndrewS and Dom
*** Performing self-update
*** Relaunching after update
*** Raspberry Pi firmware updater by Hexxeh, enhanced by AndrewS and Dom
FW_REV:3c75a2f715be18c8aaa990f9e2d5508cd586372a
BOOTLOADER_REV:b745226b41ac202976ee8307fcb179a1193fab3c
rpi-eeeprom firmware package appears to be too old. Skipping bootloader updates
WANT_32BIT:1 WANT_64BIT:1 WANT_PI4:1 WANT_PI5:0
#####
WARNING: This update bumps to rpi-6.6.y linux tree
See: https://forums.raspberrypi.com/viewtopic.php?p=2191175

'rpi-update' should only be used if there is a specific
reason to do so - for example, a request by a Raspberry Pi
engineer or if you want to help the testing effort
and are comfortable with restoring if there are regressions.

DO NOT use 'rpi-update' as part of a regular update process.
#####
Would you like to proceed? (y/N)
*** As requested, not updating kernel
*** As requested, not updating kernel modules
*** As requested, not updating VideoCore libraries
*** As requested, not updating SDK
*** Running ldconfig
*** Storing current firmware revision
*** Deleting downloaded files
*** Syncing changes to disk
*** If no errors appeared, your firmware was successfully updated to 3c75a2f715be18c8aaa990f9e2d5508cd586372a
*** A reboot is needed to activate the new firmware
pi@raspi32bit:~$ sudo adduser pi gpio
The user 'pi' is already a member of 'gpio'.
pi@raspi32bit:~$ sudo chown root:gpio /dev/gpiomem
pi@raspi32bit:~$ sudo chmod g+rw /dev/gpiomem
pi@raspi32bit:~$
```

Start des Programms im Debug-Modus

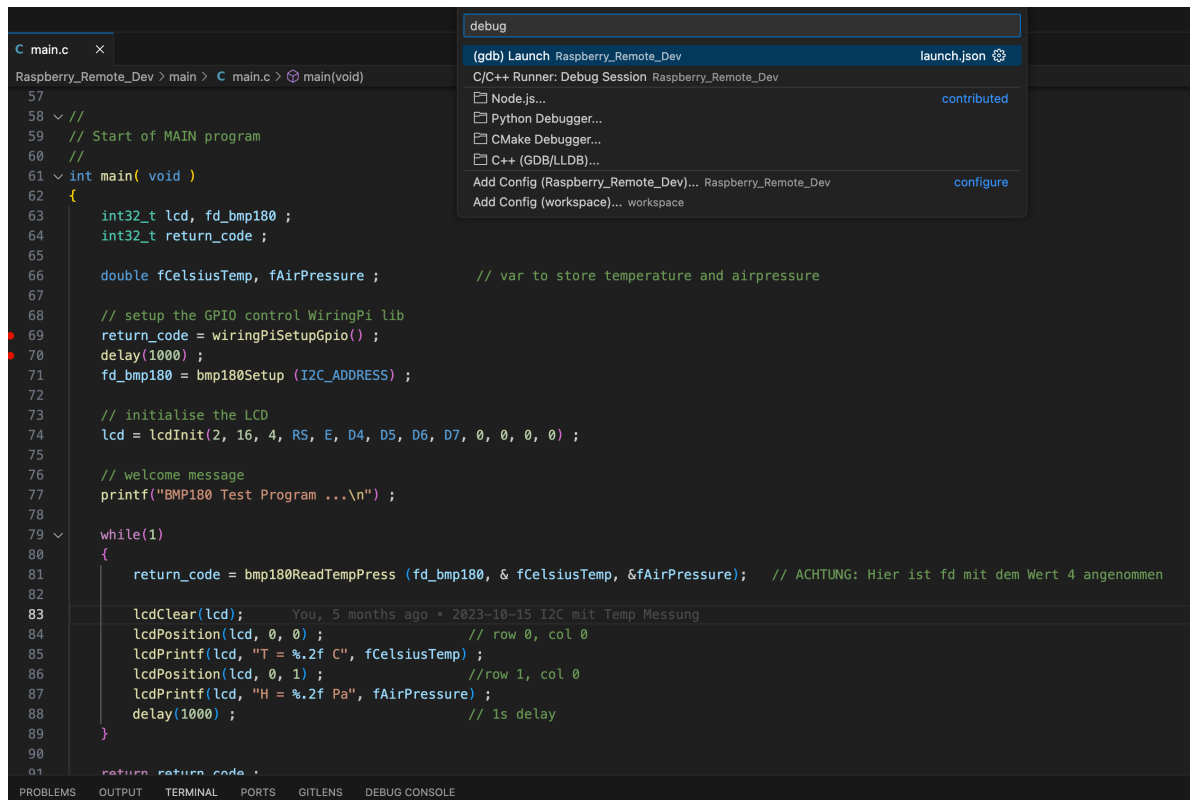
1. Home Directory über SMB mappen
Anmerkung: Auf dem Mac habe ich dazu einen Shortcut angelegt. Weitere Information un dem OneNote Reiter Mac OS X CMD
2. Start Visual Studio über VS Workspace Datei. Hier ein Screenshot auf das gemappte Home-Verzeichnis des RASP32
3. Visual Code verlangt das Raspberry Password für "pi@192.168.178.41"
Anmerkung: Das RASP32 Password wird zweimal verlangt



Name	Änderungsdatum	Art	Größe
> lib	06.08.23, 18:29	Ordner	--
▼ main	Heute, 02:16	Ordner	--
▼ build	05.10.23, 20:06	Ordner	--
▼ Debug	Heute, 02:23	Ordner	--
bmp180.c	18.10.23, 19:45	C-Quellcode	6.2 KB
bmp180.h	18.10.23, 19:55	C-He...Code	1.3 KB
BUILD.bazel	15.10.23, 16:18	Dokument	676 Bytes
main.c	18.10.23, 19:57	C-Quellcode	2.9 KB
bazel-bin	Heute, 02:25	Symb...r Link	104 Bytes
bazel-out	Heute, 02:22	Symb...r Link	92 Bytes
bazel-Raspberry_Remote_Dev	Heute, 02:22	Symb...r Link	82 Bytes
bazel-testlogs	Heute, 02:25	Symb...r Link	109 Bytes
Raspberry_Remote_Dev.code-workspace	08.08.23, 23:11	Dokument	111 Bytes
README.md	06.08.23, 16:34	E-book	121 Bytes
remote_raspi_dev.md	05.08.23, 18:16	E-book	673 Bytes
WORKSPACE.bazel	06.08.23, 15:05	Dokument	0 Bytes

4. Bazel Build starten
Z.B. über Terminal oder innerhalb Visual Code. Ich bin mir hier nicht sicher, ob über den Start des Debuggers der Build Prozess automatisch gestartet wird. Aus diesem Grund führe ich den ersten Bazel Build über das Terminal durch:
`bazel build //main:main --compilation_mode=dbg`

1. Start Debugging über Visual Code.
-> Commandzeilenfenster im Visual Code (oben in der Titelleiste)
-> Start Debugging (debug)
-> (gdb) Launch



2. Zugriffrechte müssen für jeden Start korrekt gesetzt werden

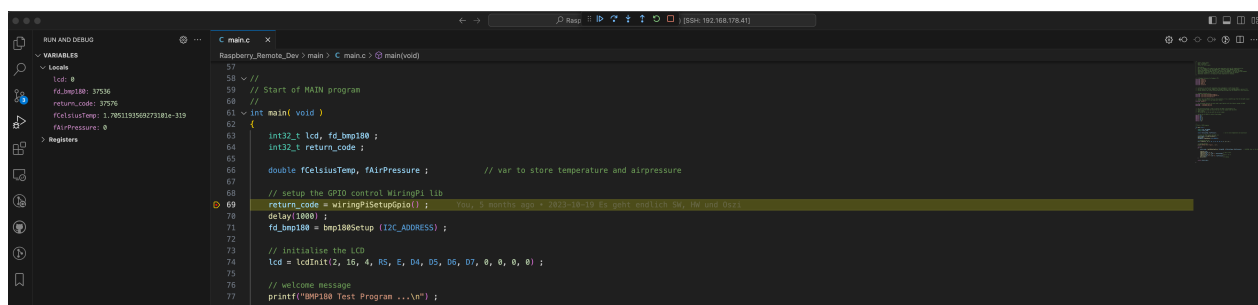
- > Zugriffsrechte für GIOMEM setzen
 - > sudo chown root.gpio /dev/gpiomem
 - > sudo chmod g+rw /dev/gpiomem

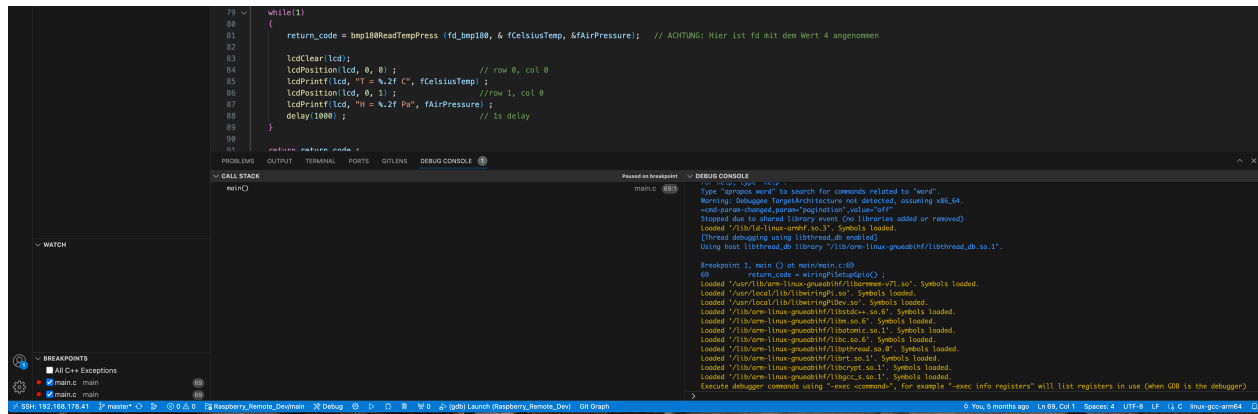
3. -> Breakpoint vorher setzen

- > In der Menüzeile ist, wenn alles funktioniert hat, nun die Steuerung für den Debugger



- > Der Debugger hat am ersten Breakpoint gestoppt





-> Weitere Ausführung über das oben gezeigte kleine Debugger Dreieck.

7) Output

