PiZero WLAN-debug

Wer schon mehr mals ein Bastelprojekt mit Microcontrollern, hier speziell Pi Pico umgesetzt hat wird auch die Erfahrung gemacht haben, das die Kabelverbindung zu den GPIO's des Entwicklungs-rechners mit Monitor, Tastatur und Maus eine nervenaufreibende Behinderung beim Umgang der am Pico angeschlossenen Hardware bei der Fehlersuche im Programmcode sind. Auch fest verbaute Hardware lässt sich so nur sehr umständlich mit Verlängerungen debuggen.
Im von den Entwicklern des Pico veröffentlichen „getting startet with pico.pdf" Tutorial wird anhand der Software „openocd" ein Aufbau gezeigt der auch Prozessorinternes Debuggen ermöglicht.

Wer nun aber die Programmiersprache C hardwarenah lernen und auch Ergebnisse sehen/messen will, muss zuvor Zeit in das wesentlich „anspruchsvollere" Setup für die Entwicklung investieren bevor er auch nur eine Zeile Programmcode schreiben kann, und oft verheddert man sich bei Umfangreichen Softwarepaketen mit deren Konfiguretion. Und auch wenn das Setup fertig ist vergisst man es weil das Pico-c-sdk und der eigene Programmcode wichtiger sind. Aus diesem Anlass habe ich zwei Skripte geschrieben die mir einen RPi headless als wlan-Debug-Bridge, und eine andere Linuxmaschine/Pi im gleichem W-LAN mit Geany als IDE so konfigurieren, das sich nach deren Auführung direkt Geany mit einem „Hallo Welt"-Projekt öffnet, so das ich durch drücken von „F9" den Buildprozess, dann durch „shift-F9" die Kompilierung und mit anschließendem F5 die Übertragung der elf-Datei auf den Pico durchführe. Die Serielle Konsole des Pico muss zuvor im Geanymenu „Erstellen→Seriell" geöffnet werden um die Funktion des Setups und Programmausgabe zu Überprüfen. Das Setup erfordert das der Benutzer auf beiden Maschinen ein Konto mit „/home"-Stammverzeichnis hat in dem die Skripte zur Ausfürung gebracht werden und sie einen Ordner mit Unterordnern anlegen. Und das in der raspi-config der wlan-debug-Bridge, die Loginshell über die serielle Konsole deaktiviert und der Hardwarezugriff aktiviert sind.

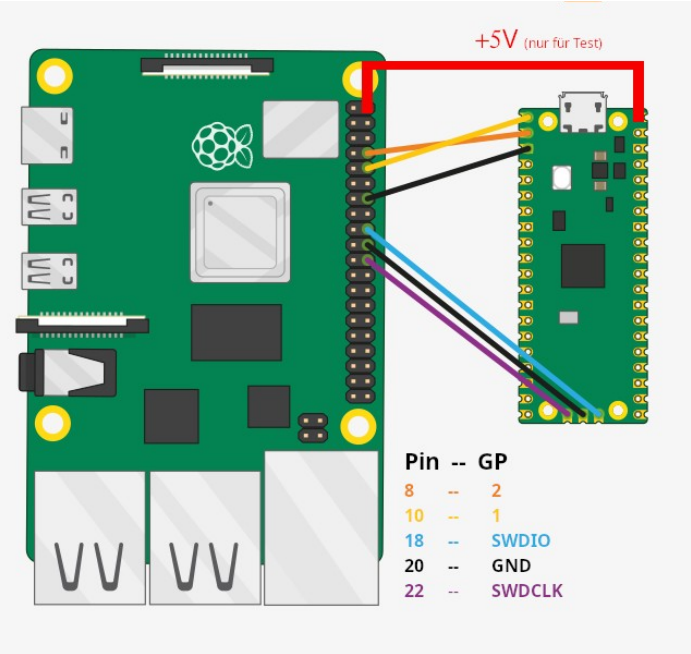
Das Setup installiert auf der Bridge openocd und anderes, was bei einem PiZeroW eine Stunde dauert und legt nur ein Skript an das mittels ssh vom Entwicklungsrechner ausgeführt wird. Auf dem Linux-Rechner werden das sdk und alles benötigte, sowie drei Skripte (transfer, seriell und new_projekt) Installiert. Wenn für das pico-sdk programmiert wird muss geany immer aus der Konsole heraus aufgerufen werden. Und ein neues pico-sdk Projekt muss immer durch Aufruf von „./new_projekt.sh projektname" aus dem „/home/user/pico/"- Verzeichnis in der Konsole angelegt werden, mit dem Geany dann automatisch startet. Um sich die wiederholende Passworteingabe bei der Übertragung der elf-Datei zum Pico zu ersparen sollten ssh-schlüssel übertragen sein; wenn auf der Entwicklermaschine kein /home/user/.ssh/id_rsa.pub existiert legt man das mit

```
ssh-keygen -t rsa
```

an um es dann mit

```
ssh-copy-id -i ~/.ssh/id_rsa.pub user@debug-bridge
```

auf die debug-bridge zu übertragen



Die im Bild rot eingezeichnete +5V verbindung dient zu Testzwecken ohne stromhungriger Hardware am Pico, und muss bei externer Spannungsversorgung des Pico/Schaltung getrennt oder mit einer Diode Rückfluss-gesichert werden um Potentialverschiebungen zu verhindern.
Die wlan-debug-bridge ist durch einen „Variablentausch“ im transfer.sh Skript im „/home/user/-Verzeichnis, auch auf Andere als das Pico board zu Konfigurieren.

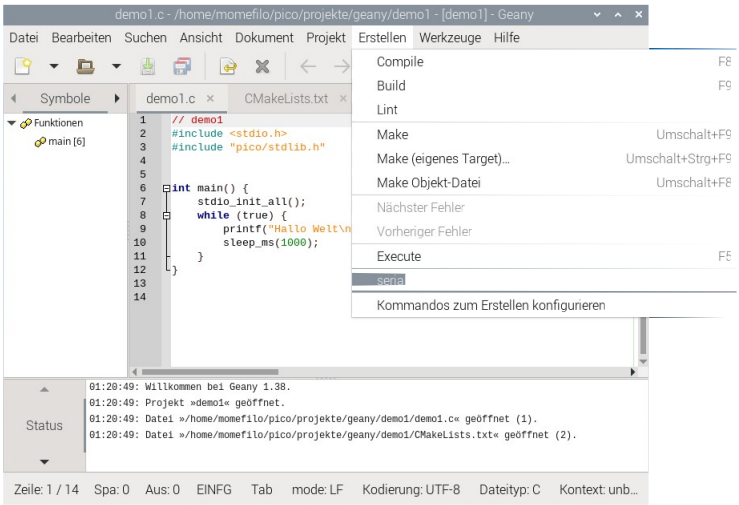
Der Link für das Skript des wlan-debug Pi in der Schaltung oben, das Skript sollte mit sudo aufgerufen werden

```
wget https://raw.githubusercontent.com/momefilo/Entwicklungsumgebung/refs/heads/main/install_bridge.sh
```

Beim Skriptaufruf auf der Enwicklermaschine muss dem Skript der hostname der wlan-debug-bridge als Parameter mit übergeben werden.

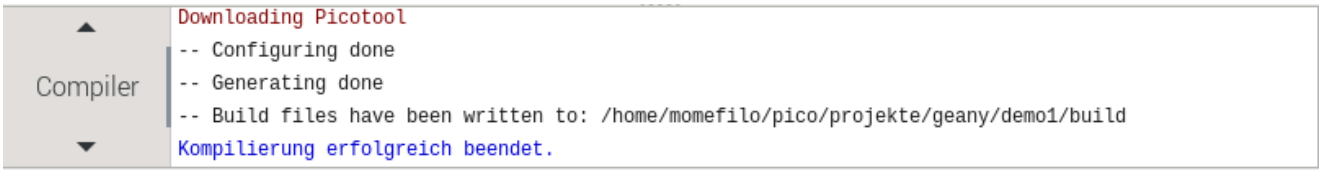
```
wget https://raw.githubusercontent.com/momefilo/Entwicklungsumgebung/refs/heads/main/install_desktop.sh
```

Nicht vergessen die Skripte zu vor mit chmod +x ausführbar zu machen um am deren Ende mit,



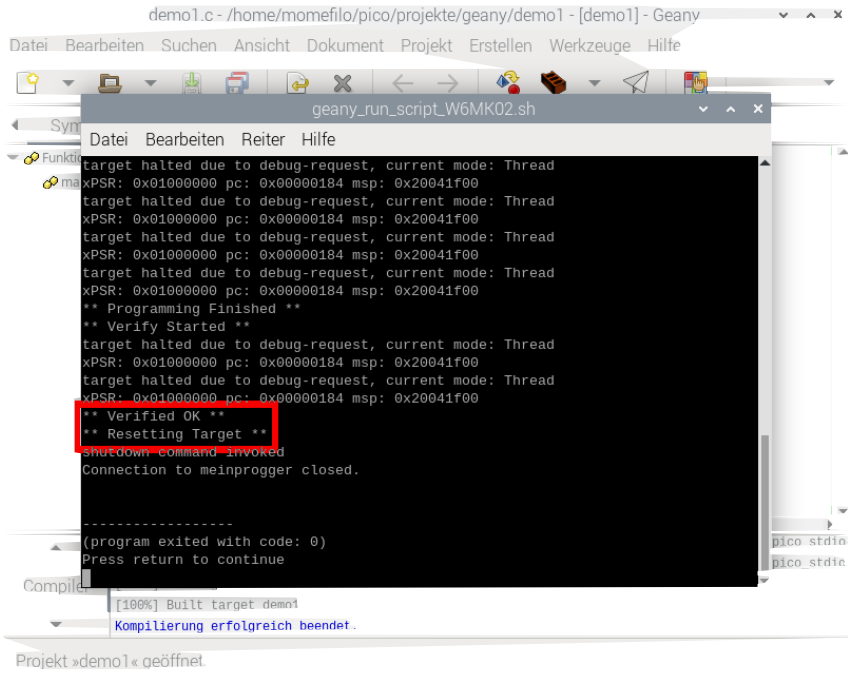
wie im Bild oben gezeigt die serielle Konsole aufzurufen die wir bei Seite ziehen und während der gesamten Debugsession göffnet lassen, um die printf-Ausgaben des noch zu builden, kompilieren und zu Übetragenden Beispiel-Programms zu verfolgen.

Das Builden geschieht durch drücken der F9-Taste , und Geany zeigt uns dessen Erfolg im Meldungsfenster unten an

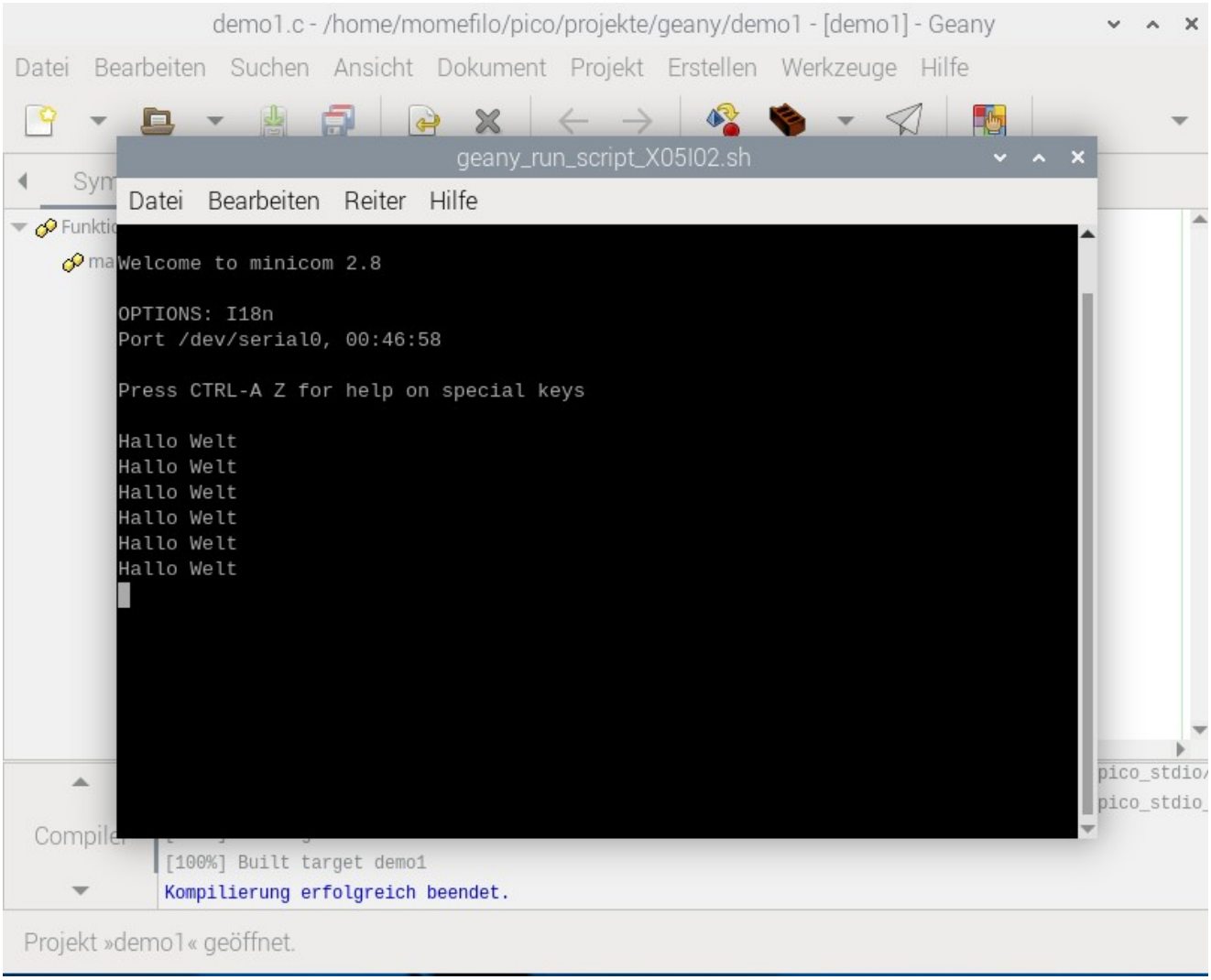


Mit shift-F9 sterten wir die nur erstmals länger andauernde, und vor jeder Übertragung zu wiederholenden Kompilerung, die uns Geany so anzeigt

Dann wird mit F5 die Übertagung auf den Pico gestartet worauf hin sich ein Terminalfenster öffnet das uns den Erfolg im rot markierten Bereich zeigt



Mit Enter schließen wir das Fenster und sehen in der zuvor geöffneten seriellen Konsole das Hallo Welt


demo1.c - /home/momefilo/pico/projekte/geany/demo1 - [demo1] - Geany
Datei Bearbeiten Suchen Ansicht Dokument Projekt Erstellen Werkzeuge Hilfe
geany_run_script_X05l02.sh
Datei Bearbeiten Reiter Hilfe
Welcome to minicom 2.8
OPTIONS: I18n
Port /dev/serial0, 00:46:58
Press CTRL-A Z for help on special keys
Hallo Welt
Hallo Welt
Hallo Welt
Hallo Welt
Hallo Welt
Hallo Welt
[100%] Built target demo1
Kompilierung erfolgreich beendet.
Projekt »demo1« geöffnet.