

# Mikrocontroller-IDE Anleitung für OSX

Um auch auf einem Apple Produkt einen Mikrocontroller (STM32F3Discovery) mit Hilfe einer IDE (Eclipse) Programmieren zu können, sind etwaige „workarounds“ nötig, aber im Prinzip ist es möglich.

## Voraussetzungen Installieren

### Installation der IDE

Dazu muss man als Allererstes die IDE herunterladen. Diese Anleitung bezieht sich ausschließlich auf Eclipse:



In meinem Fall habe ich die neuste Eclipse-Version für C/C++ heruntergeladen.

Link: <http://www.eclipse.org/downloads/packages/eclipse-ide-cc-developers/marsr>

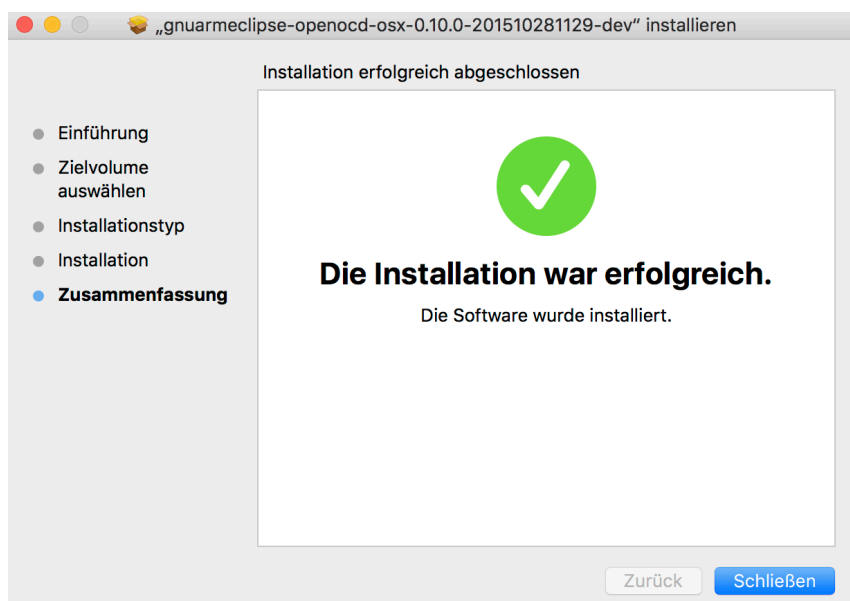
Als nächsten Schritt, entpackt man die Heruntergeladene App und schiebt sie in den Ordner „Programme“, womit es zu der Installation nicht mehr zu sagen gibt.

### Installation von GNU ARM Eclipse OpenOCD

Hierzu einfach auf folgenden Link klicken und durch Doppelklick installieren:

<https://github.com/gnuarmclipse/openocd/releases/download/gae-0.10.0-20151028/gnuarmclipse-openocd-osx-0.10.0-201510281129-dev.pkg>

Das Endergebnis sollte in etwa so aussehen:



## Installation von GCC ARM Embedded

In diesem Fall kann man das nicht wirklich „Installation“ nennen, denn es ist einfach nur ein Ordner, welchen man herunterladen muss.

Link: <https://launchpad.net/gcc-arm-embedded>

Ich habe Folgende Version heruntergeladen:

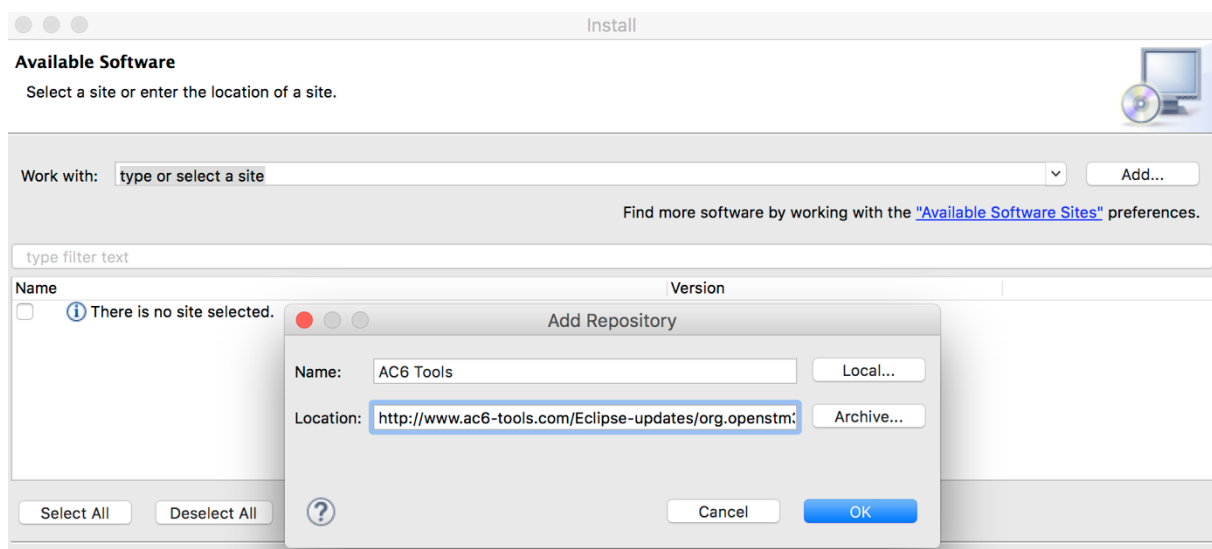
gcc-arm-none-eabi-4\_9-2015q3-20150921-mac.tar.bz2 (md5)

Nach dem Entpacken habe ich den Ordner in meine Programme verschoben.

## Eclipse Konfigurieren

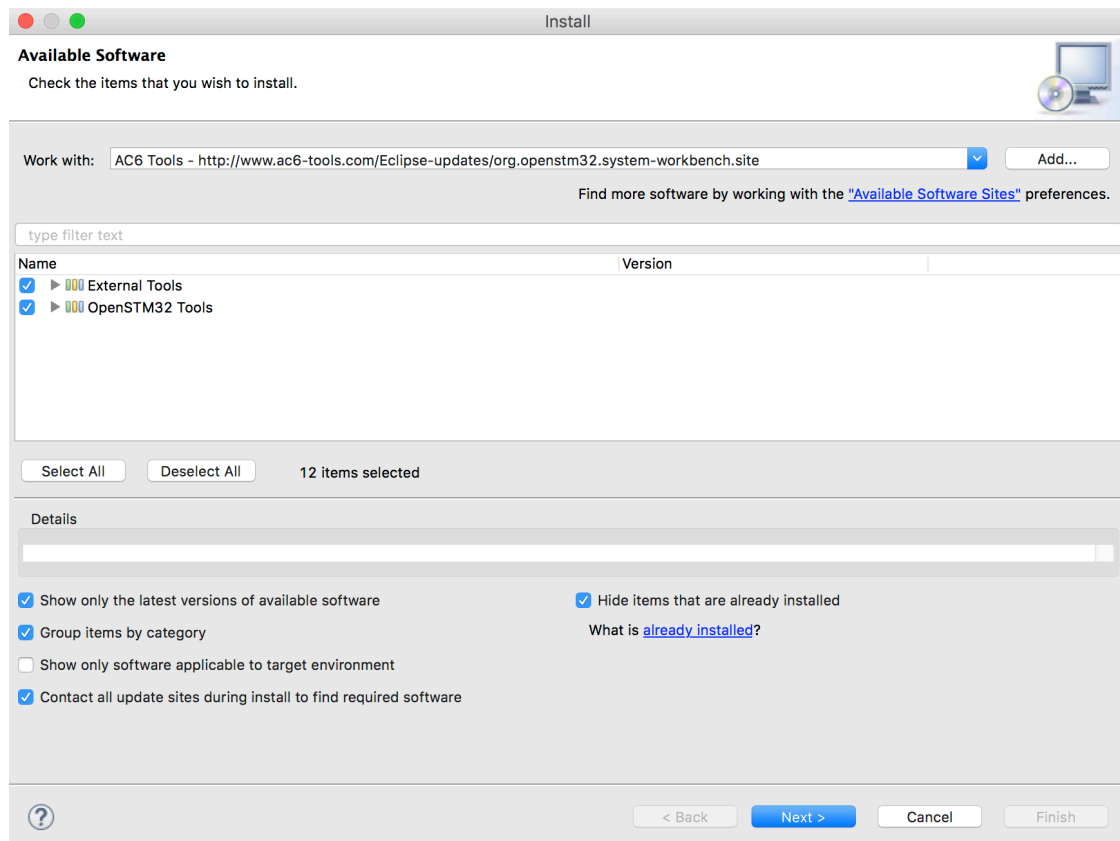
- Eclipse starten und Workspace festlegen
- Help -> Install New Software
- Auf „Add...“ klicken und folgenden Link eingeben

<http://www.ac6-tools.com/Eclipse-updates/org.openstm32.system-workbench.site>

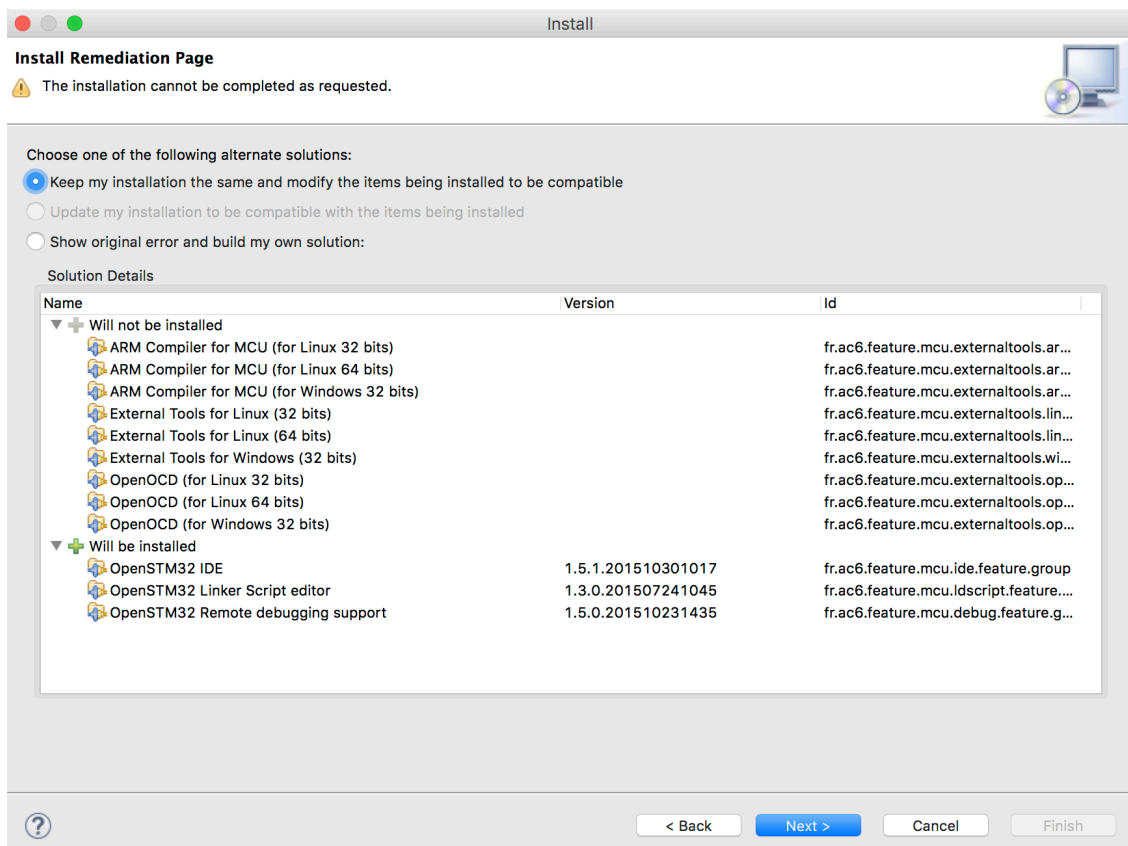


(Es ist im Prinzip egal welchen Namen man vergibt)

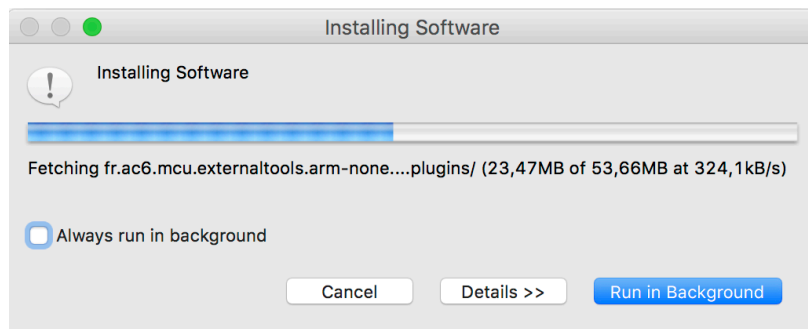
Alle Pakete auswählen und auf „Next >“ klicken



Nicht erschrecken! Nun kommt folgendes Fenster.

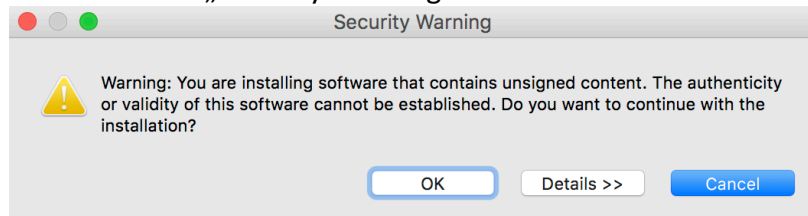


Einfach ignorieren und so lange auf „Next >“ drücken bis schlussendlich installiert wird.

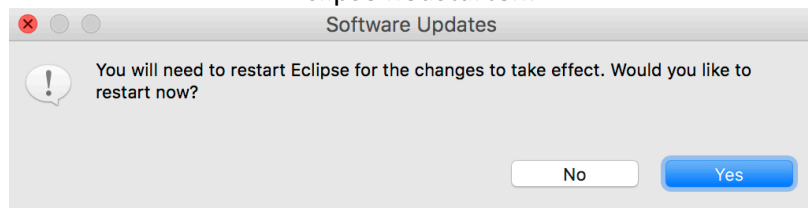


Komischer Weise wird, obwohl die Pakete laut Eclipse inkompatibel sind, etwas heruntergeladen und installiert.

Bei einem „Security Warning“ einfach auf OK klicken.



Eclipse neustarten.



Gratulation! Nun haben wir eine völlig kaputte Installation von OpenSTM32!  
Ok, beheben wir das ...

## Reparatur des ARM Compilers

Wenn wir auf Eclipse einen Rechtsklick machen und „Paketinhalt anzeigen“ auswählen sollten wir irgendwo die Pakete finden, die wir zuvor installiert haben.

In meinem Fall: /Applications/Eclipse\_Microcontroller.app/Contents/Eclipse/plugins/

Gehen wir nun in folgenden Ordner:

```
/Applications/Eclipse_Microcontroller.app/Contents/Eclipse/plugins/fr.ac6.mcu.externaltools.arm-none.linux64_1.3.0.201507241112/tools
```

Wenn sich hier nun ein Ordner namens „compiler“ befindet, löschen wir diesen und erstellen einen neuen mit selbigem Namen. Falls keiner Vorhanden ist, erstellt man einfach einen neuen.

In diesen Ordner kopieren wir nun den ganzen Inhalt aus unserem, zu Anfang installierten GCC ARM Compiler hinein.

Der Inhalt von .../tools/compiler sollte nun wie folgt aussehen:

```
drwxr-xr-x@ 6 jakubkopec  staff   204B  21 Sep 23:45 arm-none-eabi/
drwxr-xr-x@ 29 jakubkopec  staff   986B  21 Sep 23:48 bin/
drwxr-xr-x@ 3 jakubkopec  staff   102B  21 Sep 20:57 lib/
drwxr-xr-x@ 4 jakubkopec  staff   136B  21 Sep 22:36 share/
```

So weit, do gut!

## Reparatur von OpenOCD

Eine ähnliche Prozedur wie im vorherigen Kapitel erwartet uns auch hier. Diesmal handelt es sich um folgenden Pfad:

```
/Applications/Eclipse_Microcontroller.app/Contents/Eclipse/plugins/fr.ac6.mcu.externaltools.openocd.linux64_1.5.0.201510231513/tools
```

Hier sollte sich ein Ordner namens „openocd“ befinden oder wenigstens ein Archiv welches so ähnlich heißt. Falls es nur ein Archiv ist, einfach entpacken.

- Im Verzeichnis openocd/bin alle vorhandenen Files löschen.
- Nun muss man alle Files aus /Applications/GNU ARM Eclipse/OpenOCD/0.10.0-xxxxxxxxxx-dev/bin kopieren und in openocd/bin (dort wo gerade alles gelöscht wurde) einfügen.

Es sollte im Verzeichnis openocd/bin nun in etwa so aussehen:

```
-rwxr-xr-x  1 jakubkopec  staff    58K 28 0kt 12:32 libftdi1.2.dylib*
-rwxr-xr-x  1 jakubkopec  staff    25K 28 0kt 12:32 libusb-0.1.4.dylib*
-rwxr-xr-x  1 jakubkopec  staff   106K 28 0kt 12:32 libusb-1.0.0.dylib*
-rwxr-xr-x  1 jakubkopec  staff   2,1M 28 0kt 12:32 openocd*
```

Eine Kleinigkeit müssen wir aber noch machen um OpenOCD zum laufen zu bringen:

- Man muss die ausführbare Datei „openocd“ zu „openocd2“ umbenennen
- Daraufhin erstellt man eine neue ausführbare Datei „openocd“
- In diese Datei fügt man folgenden Code ein

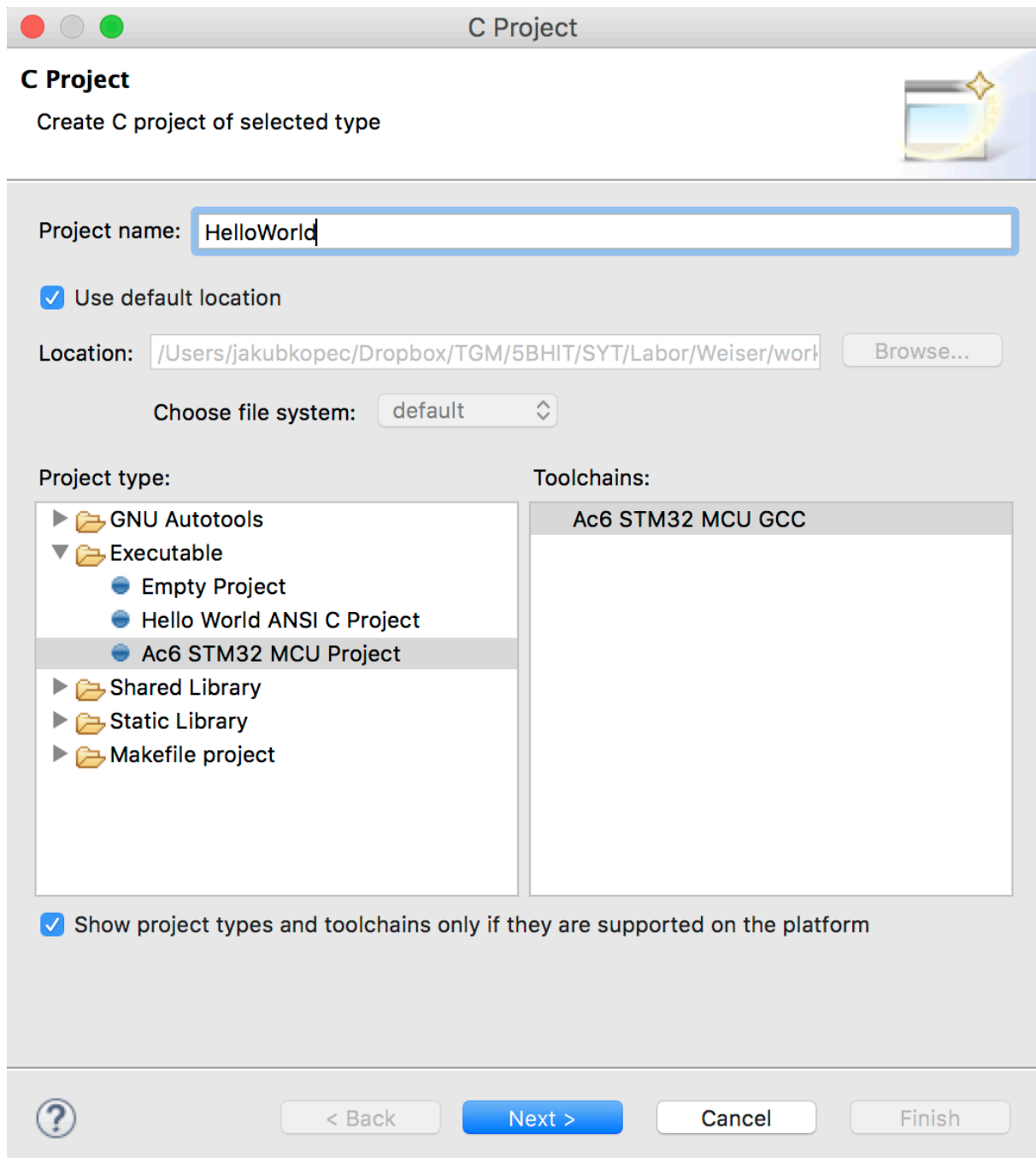
```
#!/bin/sh
echo "Fixing parameters for openocd"
echo "Input $@"
temp="${8%\"}"
temp="${temp#\"}"
"$02" $1 $2 $3 "$4" $5 "$6" $7 "$temp"
```

- Die Datei anschließend mit `chmod +x openocd` ausführbar machen

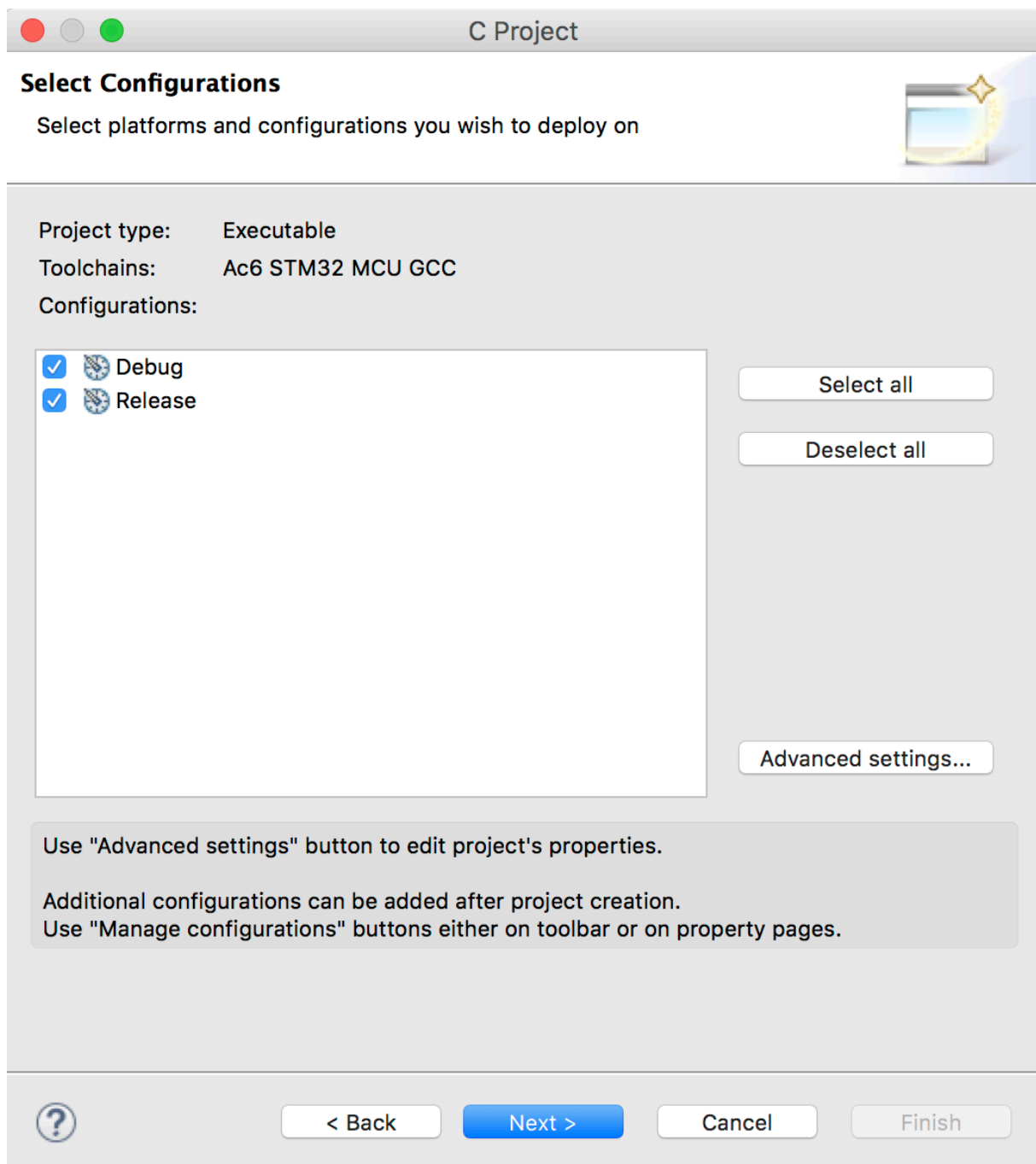
## Der Moment der Wahrheit

Schauen wir mal ob das alles was wir jetzt gemacht haben funktioniert.

Erstellen wir erstmal ein Projekt und wählen dabei ein Ac6 STM32 MCU Projekt.



Auf „Next >“ klicken.






Unseren gewünschten Mikrocontroller auswählen.

C Project

### MCU Configuration

Select the board and configurations



☒ Show ST Discovery boards

☒ Show ST EVAL boards

☒ Show ST NUCLEO boards

☒ Show custom boards

Series :

STM32F3

⌵

Board :

STM32F3DISCOVERY

⌵

Create a new custom board

Remove this custom board

Mcu	STM32F303VCTx	
Core	ARM Cortex-M4	
Package	LQFP100	
Memory	RAMEXEC - size : 0xA000 (@0x20000000)	
Memory	FLASH - size : 0x40000 (@0x8000000)	

?

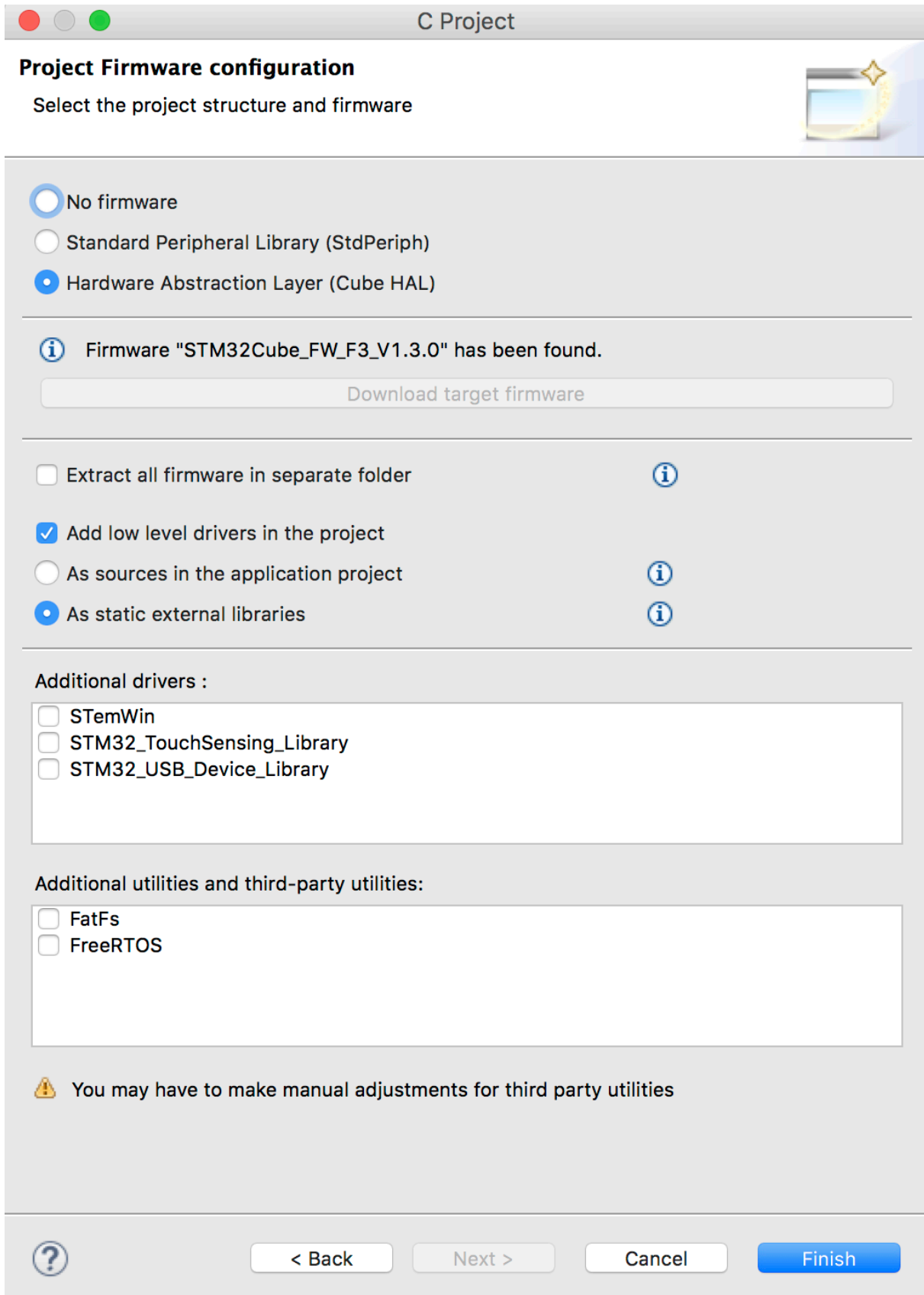
< Back

Next >

Cancel

Finish

**Wichtig!** Nicht auf „Finish“ sondern „Next >“ klicken und den Hardware Abstraction Layer auswählen.



**C Project**

**Project Firmware configuration**  
Select the project structure and firmware

☐ No firmware  
☐ Standard Peripheral Library (StdPeriph)  
☒ Hardware Abstraction Layer (Cube HAL)

**i** Firmware "STM32Cube\_FW\_F3\_V1.3.0" has been found.  
[Download target firmware](#)

☐ Extract all firmware in separate folder **i**  
☒ Add low level drivers in the project  
☐ As sources in the application project **i**  
☒ As static external libraries **i**

**Additional drivers :**

☐ STemWin  
☐ STM32\_TouchSensing\_Library  
☐ STM32\_USB\_Device\_Library

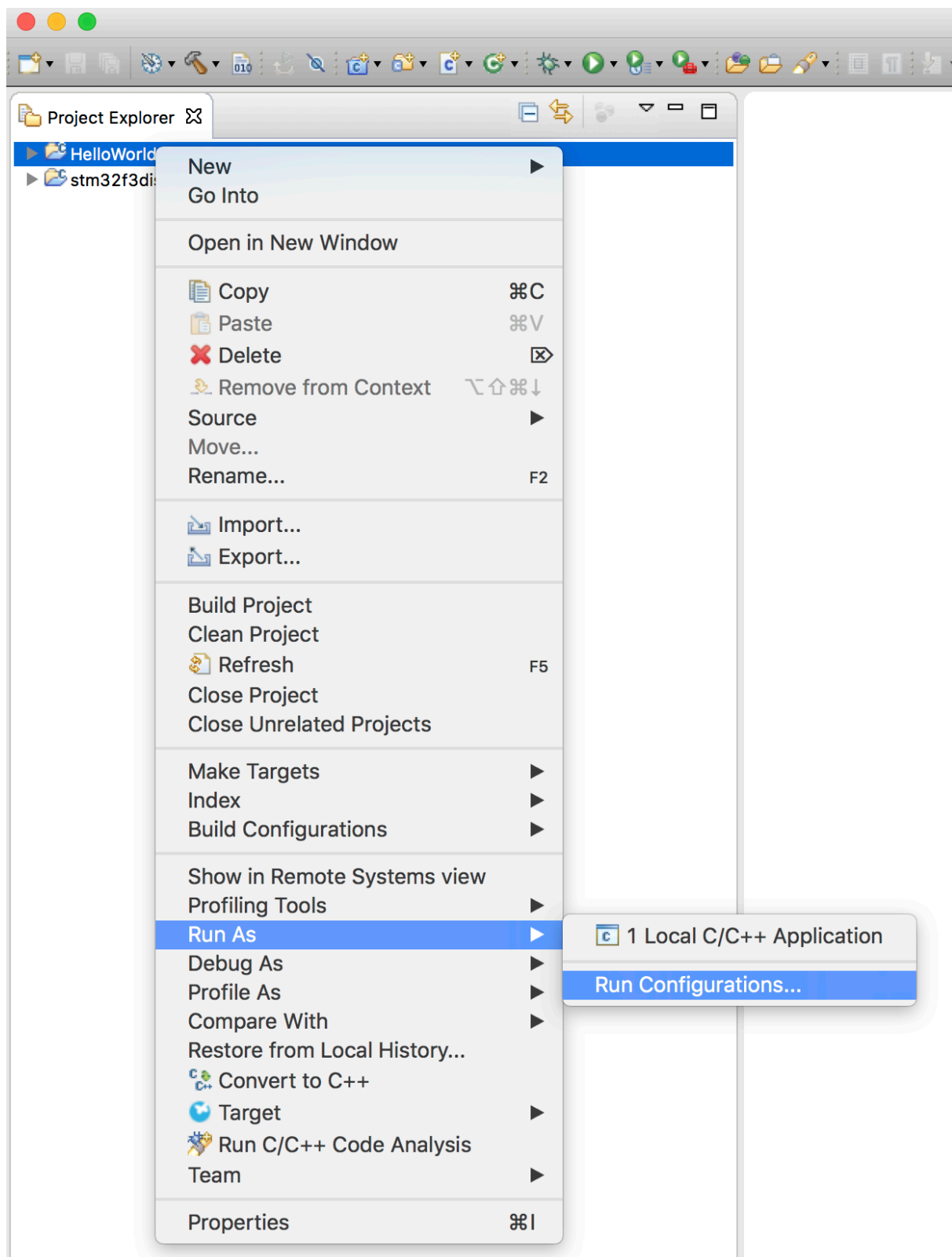
**Additional utilities and third-party utilities:**

☐ FatFs  
☐ FreeRTOS

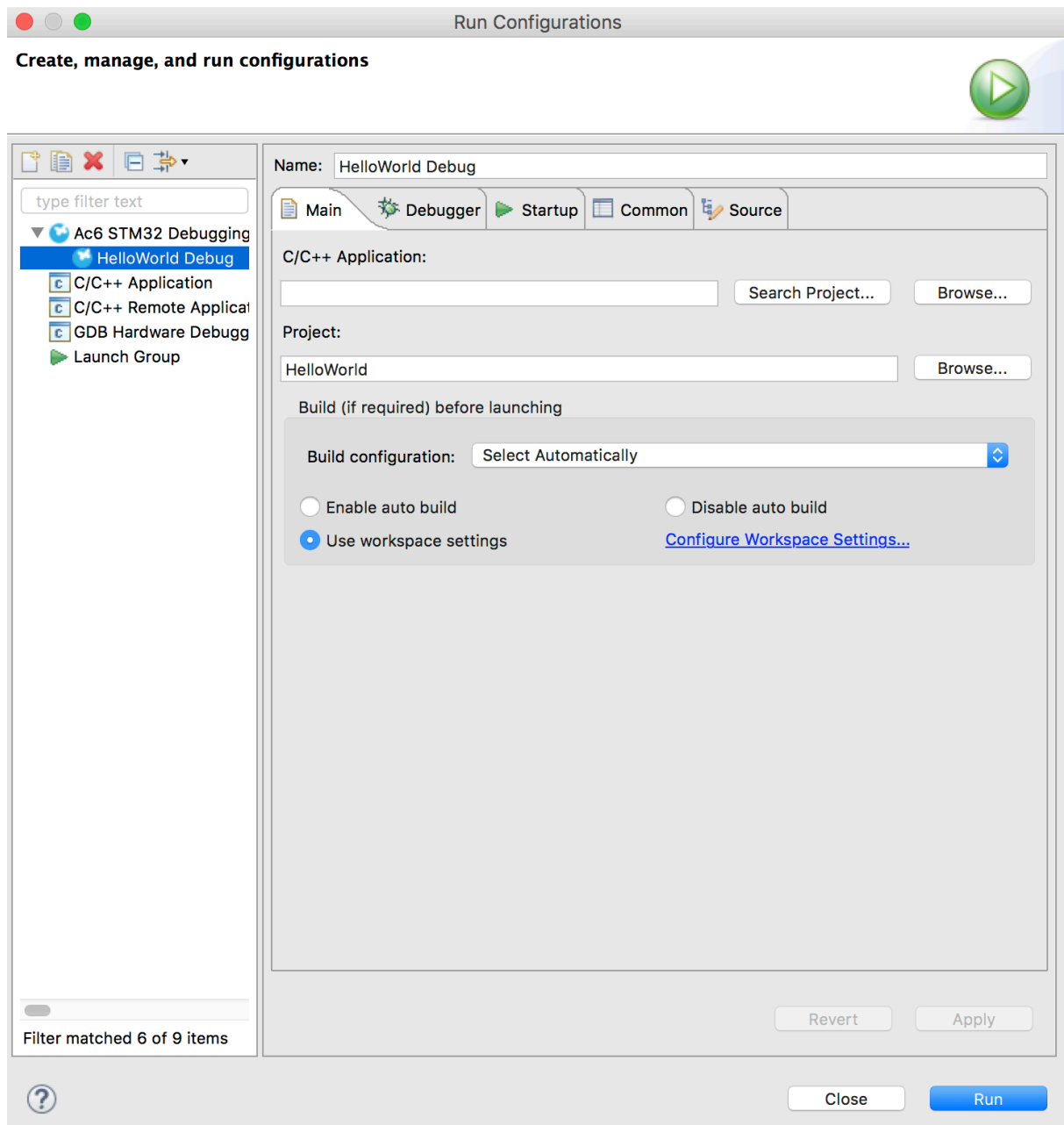
**!** You may have to make manual adjustments for third party utilities

**?** < Back Next > Cancel Finish

Einen Blick in die Run Configurations werfen.



Einen Doppelklick auf „Ac6 STM32 Debugging“ machen und anschließend auf „Run“ klicken.



Wenn es funktioniert hat, sollte der Output wie folgt sein:

```
Fixing parameters for openocd
Input -f stm32f3discovery.cfg -s
/Applications/Eclipse.app/Contents/Eclipse/plugins/fr.ac6.mcu.debug_1.5.0.201510231
435/resources/openocd/scripts/st_board -s
/Applications/Eclipse.app/Contents/Eclipse/plugins/fr.ac6.mcu.debug_1.5.0.201510231
435/resources/openocd/scripts -c "program Debug/Ampelsteuerung.elf verify reset
exit"
GNU ARM Eclipse 64-bits Open On-Chip Debugger 0.10.0-dev-00141-g09aeb96 (2015-10-
28-13:32)
Licensed under GNU GPL v2
For bug reports, read
    http://openocd.org/doc/doxygen/bugs.html
Info : auto-selecting first available session transport "hla_swd". To override use
'transport select <transport>'.
adapter speed: 1000 kHz
adapter_nsrst_delay: 100
Info : The selected transport took over low-level target control. The results might
differ compared to plain JTAG/SWD
none separate
srst_only separate srst_nogate srst_open_drain connect_deassert_srst
Info : Unable to match requested speed 1000 kHz, using 950 kHz
Info : Unable to match requested speed 1000 kHz, using 950 kHz
Info : clock speed 950 kHz
Info : STLINK v2 JTAG v16 API v2 SWIM v0 VID 0x0483 PID 0x3748
Info : using stlink api v2
Info : Target voltage: 2.912456
Info : stm32f3x.cpu: hardware has 6 breakpoints, 4 watchpoints
adapter speed: 1000 kHz
target state: halted
target halted due to debug-request, current mode: Thread
xPSR: 0x01000000 pc: 0x0800062c msp: 0x2000a000
adapter speed: 8000 kHz
** Programming Started **
auto erase enabled
Info : device id = 0x10036422
Info : flash size = 256kbytes
target state: halted
target halted due to breakpoint, current mode: Thread
xPSR: 0x61000000 pc: 0x2000003a msp: 0x2000a000
wrote 6144 bytes from file Debug/Ampelsteuerung.elf in 0.433777s (13.832 KiB/s)
** Programming Finished **
** Verify Started **
target state: halted
target halted due to breakpoint, current mode: Thread
xPSR: 0x61000000 pc: 0x2000002e msp: 0x2000a000
target state: halted
target halted due to breakpoint, current mode: Thread
xPSR: 0x61000000 pc: 0x2000002e msp: 0x2000a000
verified 4532 bytes in 0.113052s (39.148 KiB/s)
** Verified OK **
** Resetting Target **
adapter speed: 1000 kHz
shutdown command invoked
```