STM32103C8T6 aka PluePill mit PlatformIO auf Linux

Ich verwende für C+ + sehr gerne den Qt Creator. Dieser kann auch mit PlatformIO bzw. umgekehrt.

Das BluePill-Board (Universelles STM32F103-Board) lässt sich damit recht einfach programmieren.

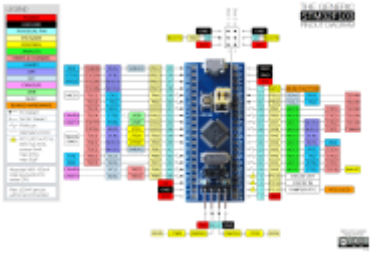
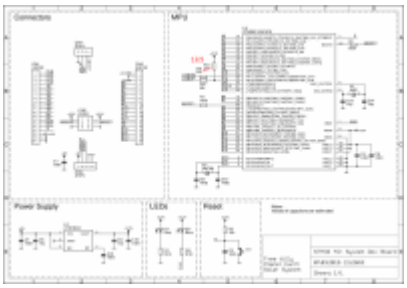




Foto und Pinout von https://commons.wikimedia.org/

Man beachte den geänderten Pullup-Widerstand. Den braucht USB, um richtig erkannt zu werden. Bei meinem BluePill Board war der schon richtigerweise 1k5 (statt fälschlicherweise 10k).

Notwendige/Hinreichende Hardware

- STLink V2
- BluePill mit einem originalen STM32F103C8T6 [1)]
- Micro-USB Kabel für Stromversorgung und SerialUSB-Verbindung

Notwendige Schritte

1. Qt Creator installieren

```
sudo apt install build-essential clang-8 qt5-default qt5-doc qt5-doc-html
qtcreator''
```

2. PlatformIO Core installieren:

```
python3 -c "$(curl -fsSL
https://raw.githubusercontent.com/platformio/platformio/develop/scripts/get-
platformio.py)"
sudo ln -s ~/.platformio/penv/bin/platformio /usr/local/bin/platformio
sudo ln -s ~/.platformio/penv/bin/pio /usr/local/bin/pio
sudo ln -s ~/.platformio/penv/bin/piodebuggdb /usr/local/bin/piodebuggdb''
```

3. Zugehörige udev-rules installiern:

```
curl -fsSL
https://raw.githubusercontent.com/platformio/platformio-core/master/scripts/
99-platformio-udev.rules | sudo tee /etc/udev/rules.d/99-platformio-
udev.rules
```

4. platformio.ini anlegen

Im Projektverzeichnis braucht es die Datei platformio.ini

platformio.ini

```
; Please visit documentation for the other options and examples
; https://docs.platformio.org/page/projectconf.html

[env:genericSTM32F103R8]
platform = ststm32
board = genericSTM32F103R8
framework = arduino
upload_protocol = stlink

# refer to
https://github.com/platformio/platform-ststm32/issues/178#issuecomment-
467286677
build_flags = -D USBCON
              -D PIO_FRAMEWORK_ARDUINO_ENABLE_CDC
              -D HAL_PCD_MODULE_ENABLED
              -D USBD_VID=0x1EAF
              -D USBD_PID=0x0004
              -D USB_PRODUCT=\"bluepill\"
```

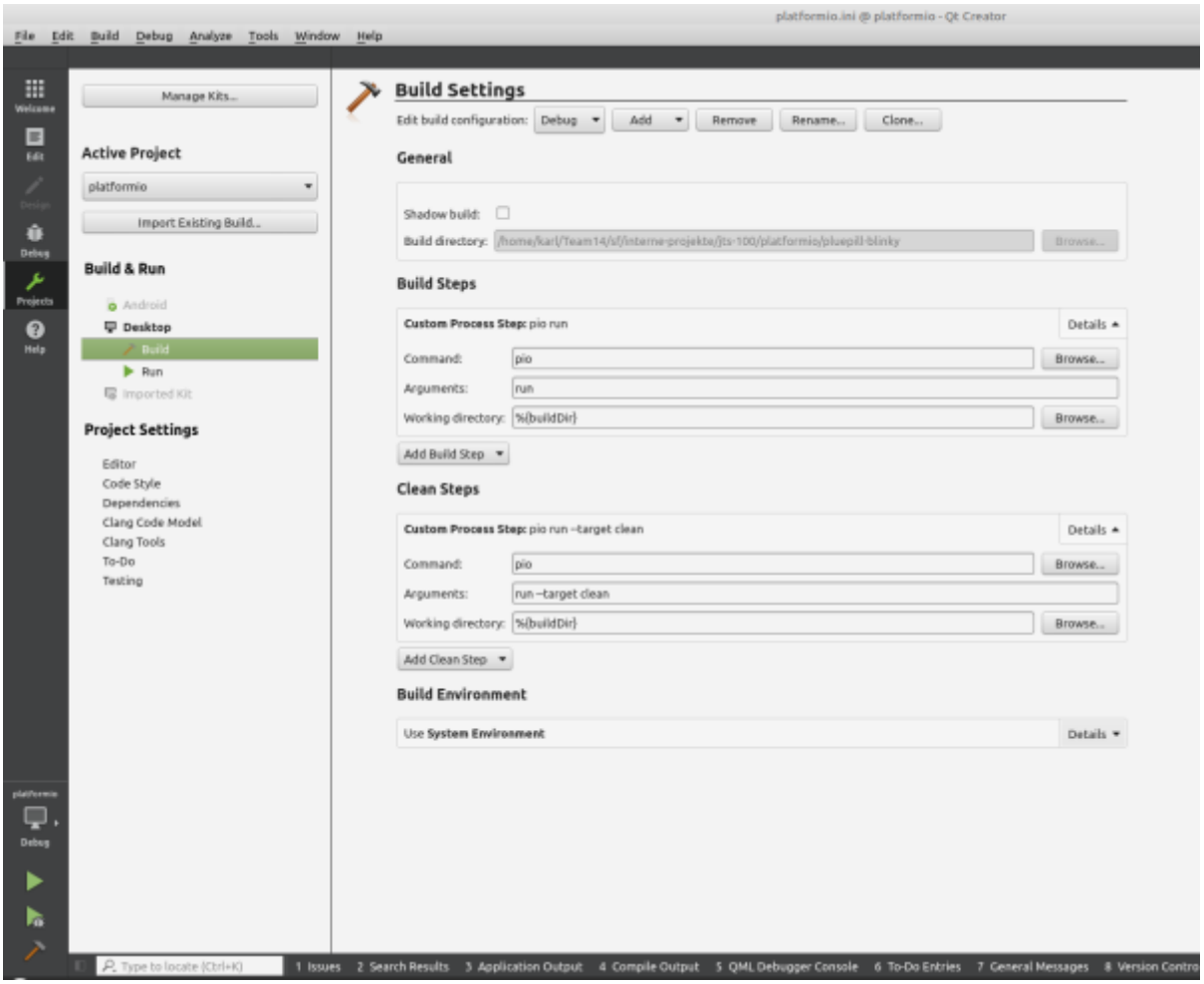
5. Qt Creator Projekt erstellen

```
pio init --ide qtcreator
```

6. Qt Creator Projekt öffnen und anpassen:

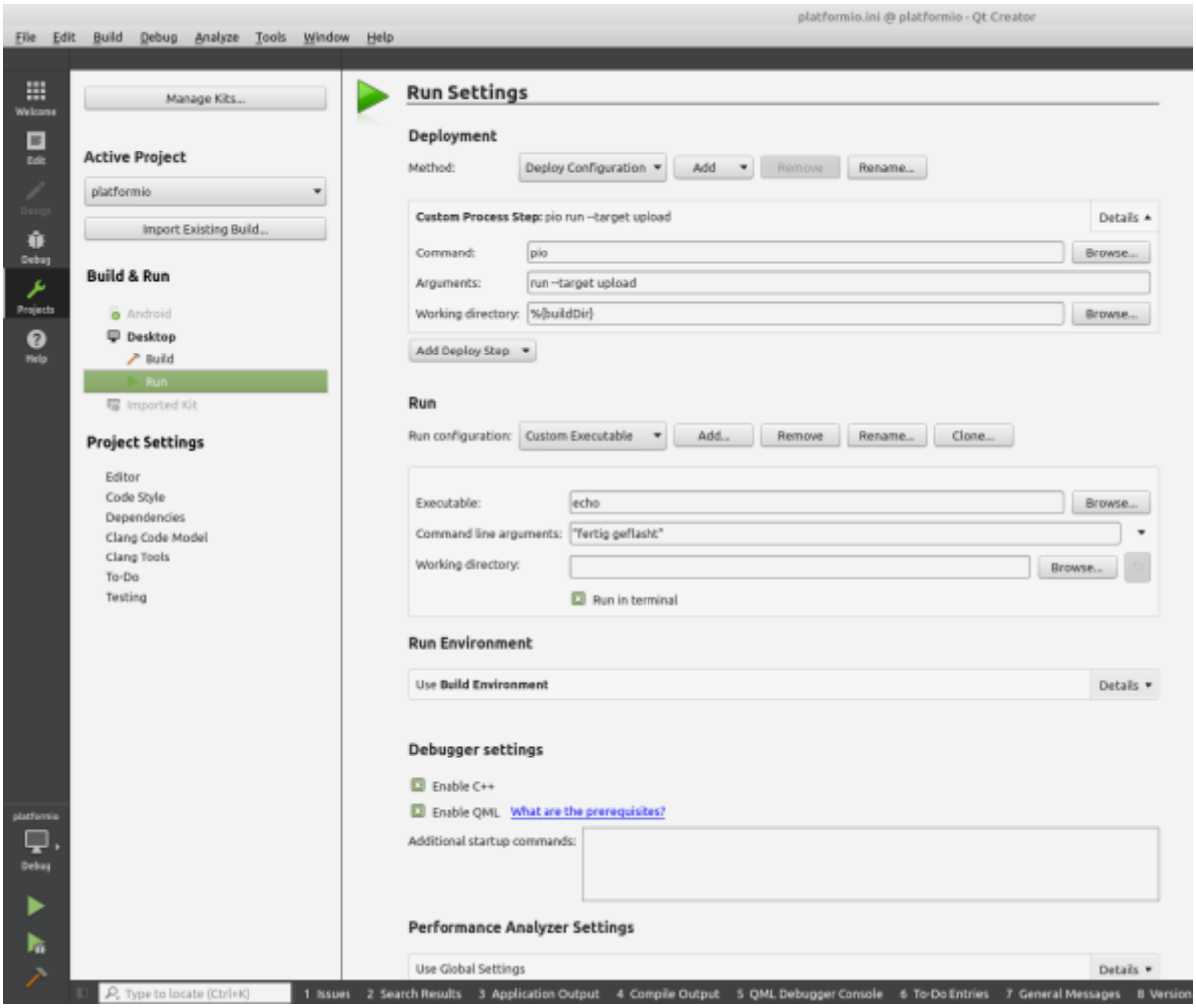
Im Bereich „Build“ muss angepasst werden:

- Build Directory
- Custom Build command: `pio run`
- Custom Clean command: `pio run –target clean`



Im Bereich „Run“ passen wir folgendes an:

- Custom Deploy Step: `pio run –target upload`
- Custom Run Configuration: `echo „fertig geflasht“`
- echo muss im Terminal gestartet werden (Checkbox setzen)



7. Blinky Demo

Wir legen eine neue cpp-Datei an im Verzeichnis `src`:

main.cpp

```
#include <Arduino.h>

#undef LED_BUILTIN
#define LED_BUILTIN PC13


void setup()
{
  pinMode(LED_BUILTIN, OUTPUT);
  Serial.begin(9600);
  SerialUSB.begin(9600);
}

void loop()
{
  digitalWrite(LED_BUILTIN, HIGH);
  Serial.println("Serial LED OFF");
```

```
  SerialUSB.println("SerialUSB LED OFF");

  delay(1000);

  digitalWrite(LED_BUILTIN, LOW);
  Serial.println("Serial LED ON");
  SerialUSB.println("SerialUSB LED ON");

  delay(1000);
}
```

8. Bauen und Hochladen

Mit **STRG+B** kann man das Projekt bauen. Hierzu ist eine Internetverbindung notwendig, da sich PlatformIO bei Bedarf die entsprechende Toolchain holt. Dieses Feature macht PlatformIO so besonders.

Mit **STRG+R** starten wir Run, und Qt Creator führt den zuvor entsprechend konfigurierten Befehl

```
pio run --target upload
```

aus. Hat alles geklappt, sollte ein Terminalfenster erscheinen mit dem Test „fertig geflasht“.

9. Hinzufügen von Bibliotheken

Ordner der Arduino-Bibliotheken [2] sollten im Projektverzeichnis unter ./lib abgelegt werden. Ein Aktualisieren des Qt Creator-Projektes mittels

```
pio init --ide qtcreator
```

ist notwendig, damit der Qt Creator dann die Bibliotheken auch findet [3].

STM32duino

Es gibt einen Bootloader, der das Board rein über USB programmierbar macht - ganz nach dem Geschmack von Arduino. Viele Arduino-Boards verwenden hierfür einen zweiten Mikrocontroller. Der Arduino DUE mit dem Atmel SAM3x8e kann aber auch direkt per USB programmiert werden. Ein im SAM3 fix eingebauter Bootloader ermöglicht dies. STM sieht sowas leider nicht vor. Daher muss ein eigener Bootloader mit einem STlink reingespielt werden. Ob dieser Bootloader schon produktreif ist, muss erst evaluiert werden. In einem Produkt, wo nur der USB-Anschluss zugänglich ist, soll man keinen Reset-Taster drücken oder gar einen Boot0-Jumper setzen müssen.

Artikel, Deutsch, Arduino, C++

1)

hier werden auch Nachbau-MCUs verkauft, darüber beschwert sich dann OpenOCD wegen falscherm

ID-Code: hla_swd Warn : UNEXPECTED idcode: 0x2ba01477 Error: expected 1 of 1: 0x1ba01477
[2)]

sind nach der Installation mit der Arduino IDE im Home-Verzeichnist unter ~./Arduino/libraries zu finden
[3)]

hierbei wird das Project-File mit den entsprechenden Abhängigkeiten aktualisiert