Knomi - ESP C3 Hack

Knomi ist ein extra Display für den Stealthburner das von BTT entwickelt wurde. Es zeigt während dem Druck diverse Statusinformationen, an die es über WLAN vom Drucker bezieht. Siehe dazu auch die Doku für mehr Informationen: https://bigtreetech.github.io/docs/KNOMI.html .

Der original Knomi basiert auf einem ESP32 Wroom. Nun gibt es aber diverse andere Runddisplays, die ebenfalls einen EPS32 verwenden. Eines dieser Displays ist das folgende Display:
https://de.aliexpress.com/item/1005005453515690.html
Zu haben für ca. 12€ und damit deutlich günstiger als ein original Knomi. Der Nachteil ist, dass dieses Display einen ESP32 C3 verwendet, der leider nur mit 400 anstatt 520kb RAM ausgestattet ist und zudem nur einen CPU Kern hat (und nicht zwei wie der Knomi).

Im Folgenden wird beschrieben, wie die Firmware kompiliert werden kann und was es ggf. noch an nützlichen Infos gibt ...

YouTube Video #65

WICHTIG

Diese Firmware funktioniert nur auf einem Display Modul mit einem ESP32 C3!
Die Firmware funktioniert nicht auf einem original Knomi!

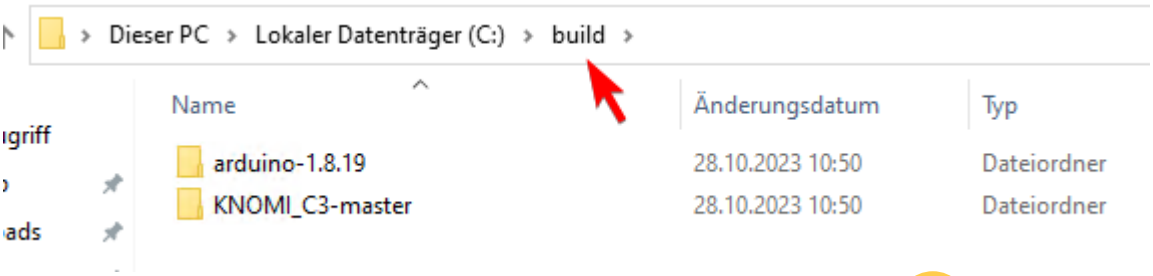
Vorbereitungen & Infos

Die Anleitung basiert darauf das die alte Arduino IDE (legacy) genutzt wird. Diese kann in einem portablen Modus genutzt werden - sprich alle nötigen Dateien & Ordner befinden sich in einem Ordner auf der Platte.

Als Vorbereitung muss also auf der Festplatte ein Ordner angelegt werden in dem später die Arduino IDE und der Quellcode abgelet werden. Ich werde im Folgenden beispielhaft den Ordner `C:\Build` verwenden. Wenn ihr einen anderen Ordner habt muss das ggf. angepasst werden.

Downloads

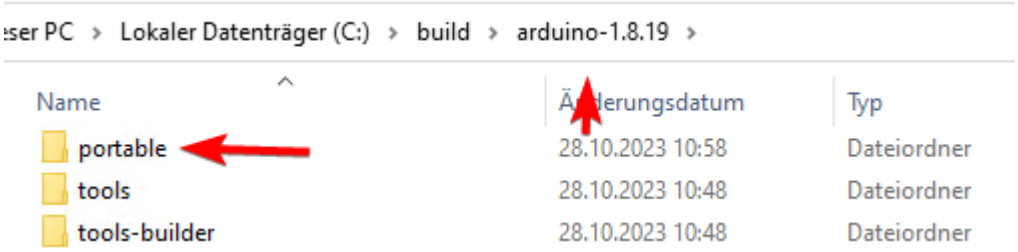
- Die Arduino IDE kann hier geladen werden
  https://www.arduino.cc/en/software
  Wir brauchen die Arduino IDE 1.8.19 und zwar als Windows ZIP.
  Direktlink : https://downloads.arduino.cc/arduino-1.8.19-windows.zip
- Die Datei `arduino-1.8.19-windows.zip` muss in den Ordner `c:\Build\` entpackt werden
- Als nächstes muss der Knomi C3 Quellcode von Github geladen werden. Am einfachsten geht das über den direkten Link
  https://codeload.github.com/DrKlipper/KNOMI_C3/zip/refs/heads/master
- Das geladene ZIP `KNOMI_C3-master.zip` wird dann ebenfalls in den `c:\Build\` Ordner entpackt
- Danach sollte euer Ordner folgendermaßen aussehen:



Hinweis: Groß- und Kleinschreibung ist bei dem Ordnernamen egal

Arduino IDE portabel

Damit die Arduino IDE portabel wird muss ein extra Ordner angelegt werden. Und zwar in `C:\Build\arduino-1.8.19\` der Ordner `portable`. Der muss auch wirklich genauso heißen!

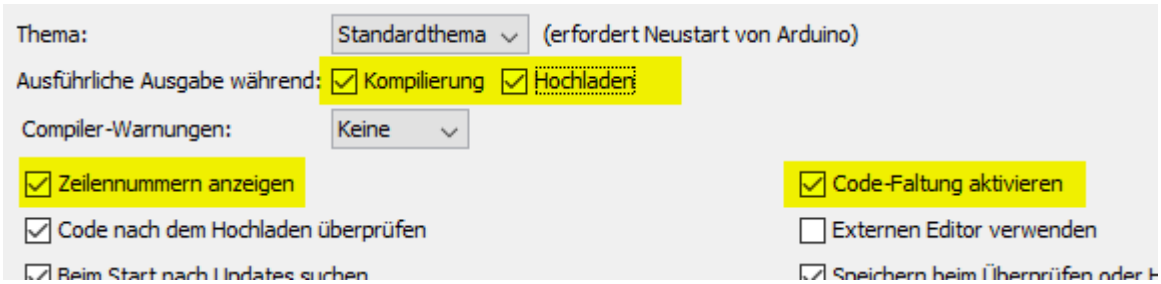


> **Die Arduino IDE starten**
> (über `arduino.exe`)

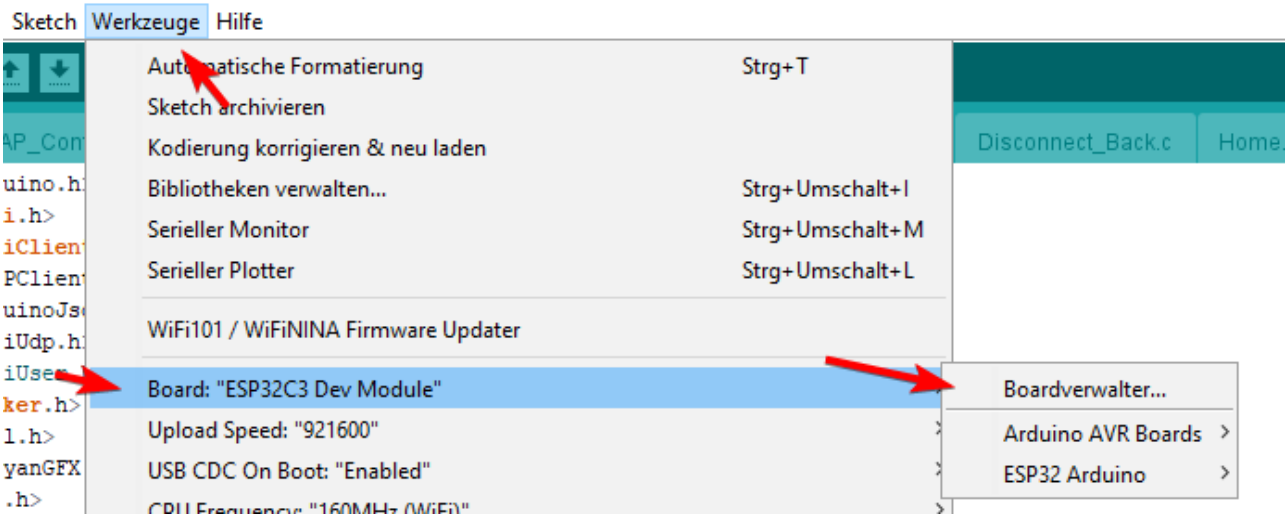
Arduino IDE einrichten

- Arduino IDE Starten und Einstellungen (Datei → Voreinstellungen) anpassen

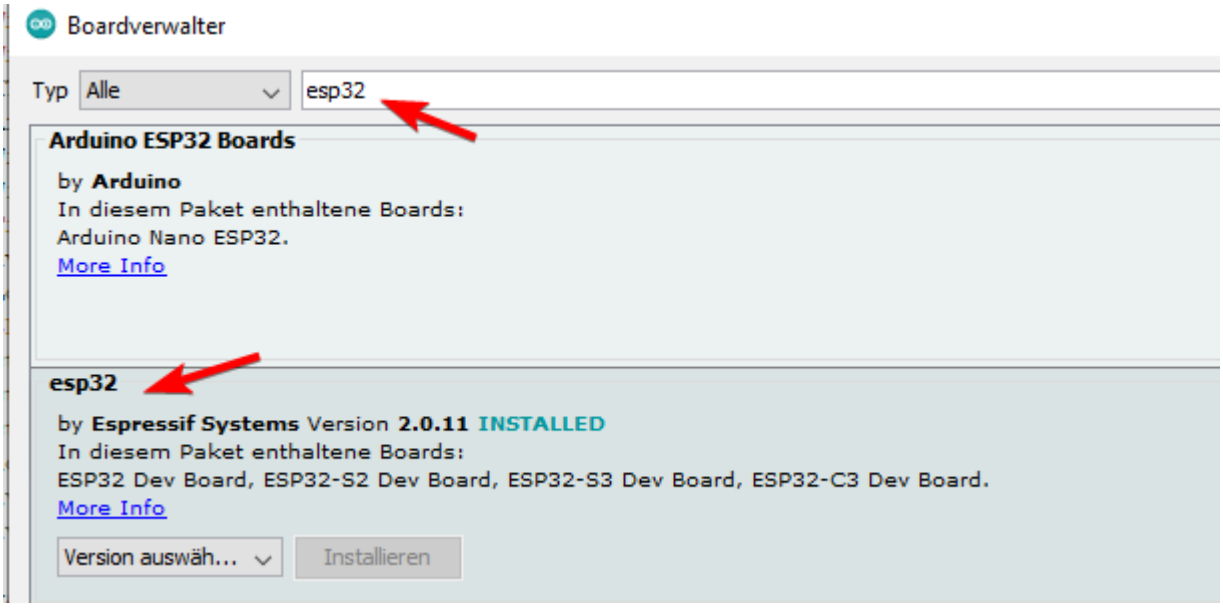
  Die Einstellungen sind nicht zwingend notwendig, helfen aber ggf. bei der Fehlersuche



- Jetzt muss im Board Manager ein extra Paket für die ESP32 installiert werden. Den Board Manager findet man im Menü über Werkzeuge.



Hier suchen wir nach dem Paket ESP32 und installieren (Button unten rechts) die letzte Version



> **Die Arduino IDE schließen**

Bibliotheken

Für das korrekte Kompilieren müssen noch 3 Bibliotheken kopiert werden.

- Die Bibliotheken (Libs) aus dem Ordner
  `C:\build\KNOMI_C3-master\Firmware\lib`
  in den Arduino IDE Ordner

`C:\build\arduino-1.8.19\portable\packages\esp32\hardware\esp32\2.0.11\libraries`
kopieren.

extra ESP32 Partition

Es muss eine neues Partition Layout hinzugefügt werden (beschreibt die Einteilung des Flash in bestimmte Bereiche). Sonst passt die compilierte Firmware nicht komplett in den Seicher.

- Die Partition Datei
  `C:\Build\KNOMI_C3-master\Firmware\c3_partitions.csv`
  muss in den Ordner
  `C:\Build\arduino-1.8.19\portable\packages\esp32\hardware\esp32\2.0.11\tools\partitions\`
  kopiert werden.
- Dann muss noch ein Eintrag im Partiton Menü hinzugefügt werden
  Die Datei
  `C:\Build\arduino-1.8.19\portable\packages\esp32\hardware\esp32\2.0.11\boards.txt` in einem Texteditor öffnen
- Im Bereich `esp32c3.menu.PartitionScheme` (Auf das `esp32c3` / `ESP32C3 Dev Module` achten) muss am Ende folgendes ergänzt werden:

```
esp32c3.menu.PartitionScheme.knomi_c3=Knomi C3 (3,3MB No OTA/1MB SPIFFS)
esp32c3.menu.PartitionScheme.knomi_c3.build.partitions=c3_partitions
esp32c3.menu.PartitionScheme.knomi_c3.upload.maximum_size=3342336
```

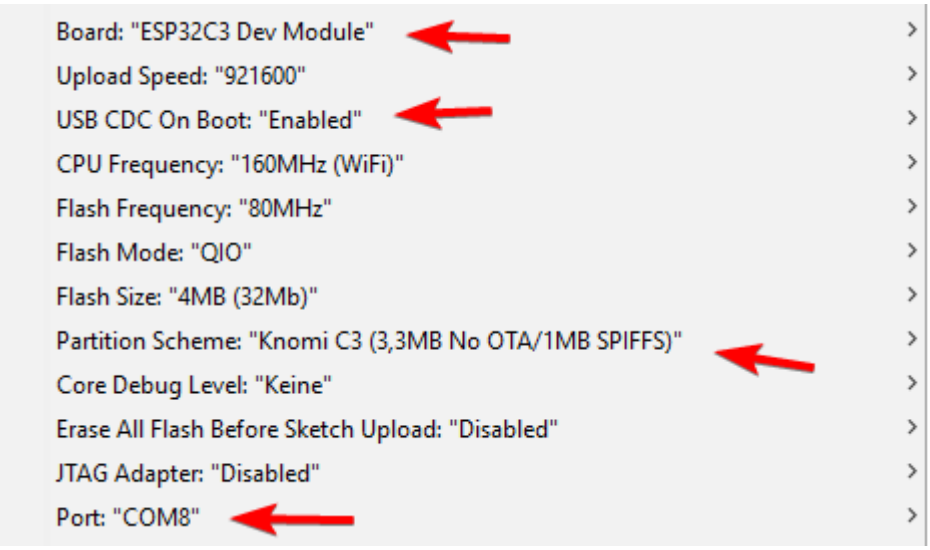
```
esp32c3.menu.PartitionScheme.app3M_fat9M_16MB.upload.maximum_size=3145728
esp32c3.menu.PartitionScheme.rainmaker=RainMaker
esp32c3.menu.PartitionScheme.rainmaker.build.partitions=rainmaker
esp32c3.menu.PartitionScheme.rainmaker.upload.maximum_size=3145728
esp32c3.menu.PartitionScheme.knomi_c3=Knomi C3 (3,3MB No OTA/1MB SPIFFS)
esp32c3.menu.PartitionScheme.knomi_c3.build.partitions=c3_partitions
esp32c3.menu.PartitionScheme.knomi_c3.upload.maximum_size=3342336
```

- Datei speichern und schließen.

> **Die Arduino IDE starten**
> (über `arduino.exe`)

Controller einstellen

- In der Arduino IDE muss ein ESP32 C3 Controller ausgewählt werden (Menü Tools bzw. Werkzeuge).
- Zusätzlich sind alle Einstellungen wie folgt zu setzen



> **Hinweis:**
> Der COM Port muss natürlich ermittelt werden! Das geht z.B. über den Geräte Manager von Windows.

Code laden und kompilieren

- In der Arduino IDE über öffnen aus dem Ordner `C:\Build\KNOMI_C3-master\Firmware\KnomiC3` die Datei `KnomiC3.ino` öffnen.
- In der Datei `WifiUser.cpp` eure Daten eintragen. Die Datei findet ihr in den Reitern in der Arduino IDE oben.
  Eintragen müsst ihr folgendes:

```
// WEB Config
String wifi_ssid  = "SSID";       // Wifi SSID
String wifi_pass  = "Password";   // Wifi Password
String klipper_ip = "192.168.xxx.xxx";    // Klipper IP
```

- Dann im Menü Sketch → Überprüfen/Kompilieren anklicken und warten .....
- Wenn am Ende dann sowas da steht:

```
Der Sketch verwendet 2999696 Bytes (89%) des Programmspeicherplatzes.
Das Maximum sind 3342336 Bytes.
Globale Variablen verwenden 184460 Bytes (56%) des dynamischen
Speichers, 143220 Bytes für lokale Variablen verbleiben. Das Maximum
sind 327680 Bytes.
```

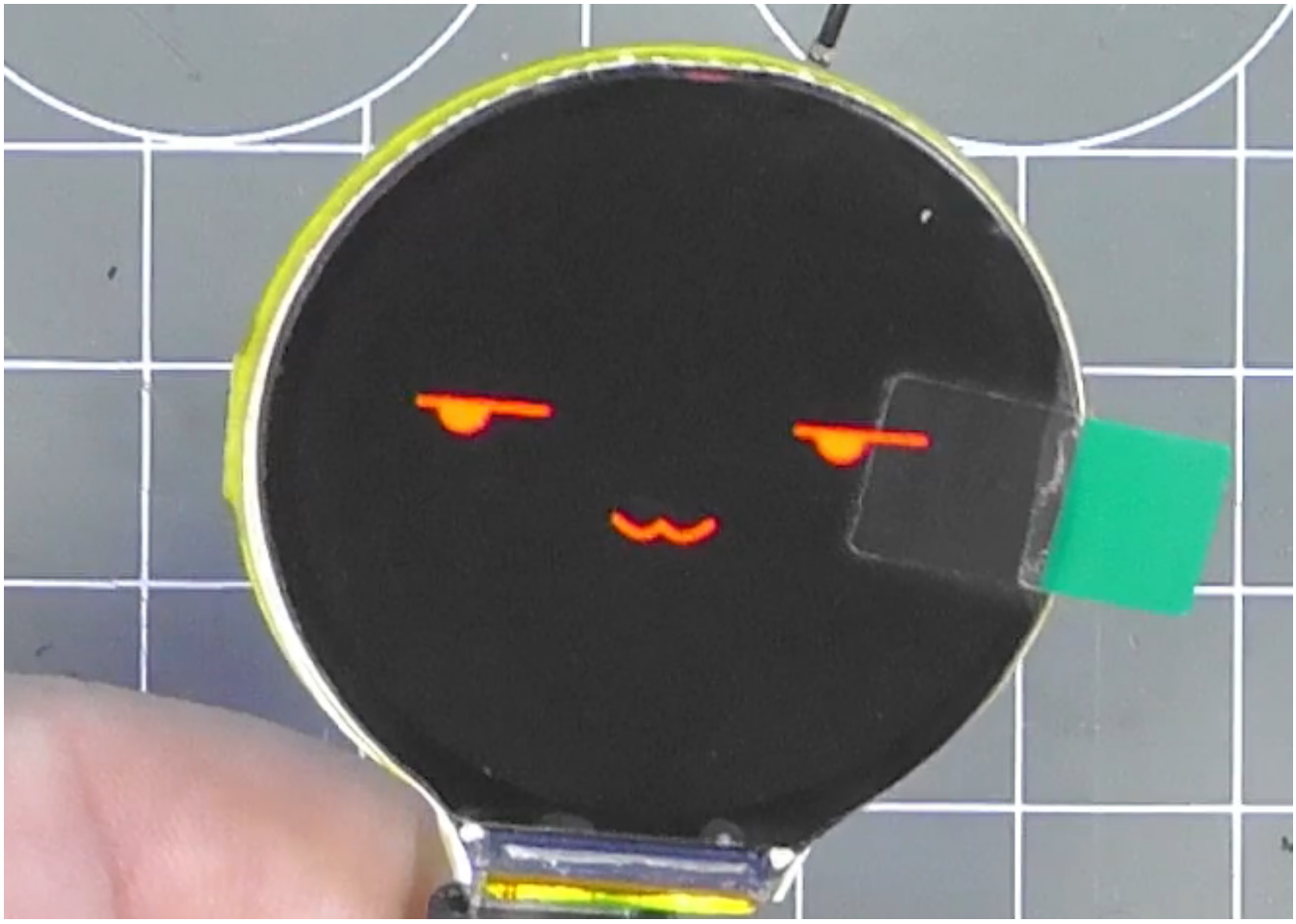
  ... hat das Kompilieren geklappt 🙂

- Jetzt könnt ihr den Code auf den Controller übertragen, mit dem Pfeil nach rechts oben bei den Buttons.
  Und ja, man hätte sich das Compilieren vorher auch sparen können, denn das würde er bei diesem Schritt sonst automatisch mit erledigen 🙂

Test

Wenn der Knomi C3 jetzt bootet sollte er erst einen Startschirm zeigen und dann die rollenden Augen.

Das bedeutet er ist verbunden 🙂



Klipper Konfig

- Eine neue Konfig Datei (z.B. knomi.cfg) mit folgendem Inhalt anlegen:

knomi.cfg

```
### KNOMI ADDON
[gcode_macro BedLevelVar]
variable_leveling : False

gcode:
  SET_GCODE_VARIABLE MACRO=BedLevelVar VARIABLE=leveling
VALUE=False

[gcode_macro HomeSetVar]
variable_homing : False

gcode:
  SET_GCODE_VARIABLE MACRO=HomeSetVar VARIABLE=homing VALUE=False

#[gcode_macro G28]
#rename_existing: G0028
#variable_homing:False
#
```

```
#gcode:
#  SET_GCODE_VARIABLE MACRO=HomeSetVar VARIABLE=homing VALUE=True
#  G0028 {rawparams}
#  SET_GCODE_VARIABLE MACRO=HomeSetVar VARIABLE=homing VALUE=False
```

- Wenn ihr einen Drucker ohne ein `[homing_override]` habt, dann müsst ihr das untere Makro ([gcode_macro G28]) noch aktivieren. Dazu alle # am Anfang der Zeilen entfernen
- Die Datei in der `printer.cfg` inkludieren :
  `[include knomi.cfg]`
- Jetzt müsst ihr an der Stelle wo euer Leveling passiert (bei mir ist das beim QGL) 2 Zeilen ergänzen:

```
SET_GCODE_VARIABLE MACRO=BedLevelVar VARIABLE=leveling VALUE=True
QUAD_GANTRY_LEVEL
SET_GCODE_VARIABLE MACRO=BedLevelVar VARIABLE=leveling VALUE=False
```

  Bei mir ist das in einem Printer_Start Makro. Die erste Zeile setzt den Status “Leveln” und die letzte Zeile setzt den Status wieder zurück. Wo ihr das bei euch genau eintragt müsst ihr mal selber ermitteln. Das hängt sicherlich auch vom Drucker und eurer Konfig ab.

- Dann müssen wir noch das Homing mit einem extra Status versehen. Wenn ihr kein `[homing_override]` in eurer Konfig habt, dann reicht dsas G28 Makro in der Extra Konfig. Habt ihr aber `[homing_override]`, dann kommt folgendes an den Anfang:
  `SET_GCODE_VARIABLE MACRO=HomeSetVar VARIABLE=homing VALUE=True`

```
[homing_override]

axes: xyz
gcode:
    SET_GCODE_VARIABLE MACRO=HomeSetVar VARIABLE=homing VALUE=True

    # collect user state variables
    _User_Variables
    {% set verbose = printer["gcode_macro _User_Variables"].verbose %}
    {% set safe_z = printer["gcode_macro _User_Variables"].safe_z|float %}
```

  und folgendes ans Ende:
  `SET_GCODE_VARIABLE MACRO=HomeSetVar VARIABLE=homing VALUE=False`

```
614     # park the toolhead
615     _Park_Toolhead
616
617     SET_GCODE_VARIABLE MACRO=HomeSetVar VARIABLE=homing VALUE=False
618
619     _exit_point function=homing_override
620
```

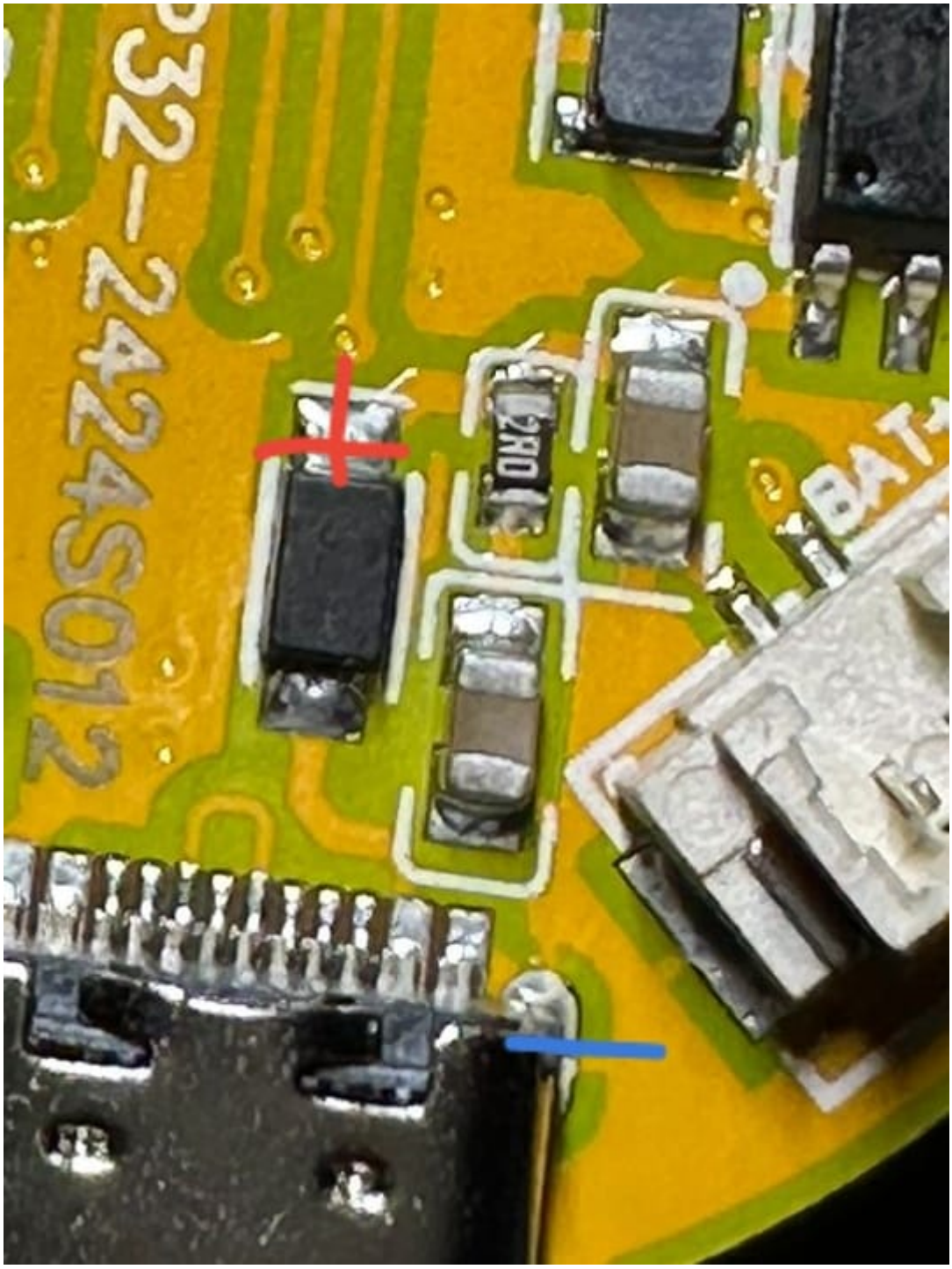
- Zum Schluss noch im Printer_Start Macro die Werte zum Druckstart zurücksetzen:

```
SET_GCODE_VARIABLE MACRO=BedLevelVar VARIABLE=leveling VALUE=False
SET_GCODE_VARIABLE MACRO=HomeSetVar VARIABLE=homing VALUE=False
```

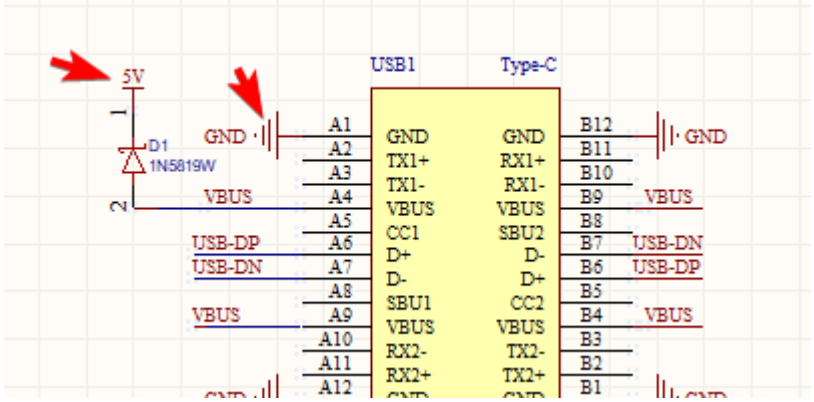
```
[gcode_macro PRINT_START_PLA]
gcode:
  ; Get Parameter from Slicer (Start_G-Code Filament)
  {% set T_EX = params.EXTRUDER|default(210)|int %}
  {% set T_BD = params.BED|default(50)|int %}

  SET_GCODE_VARIABLE MACRO=BedLevelVar VARIABLE=leveling VALUE=False
  SET_GCODE_VARIABLE MACRO=HomeSetVar VARIABLE=homing VALUE=False

  M117 Prepare Printer ...
  SET_FAN_SPEED FAN=Base SPEED=0.2      ; Fan Speed Elektronik auf 20%
```

Stromversorgung

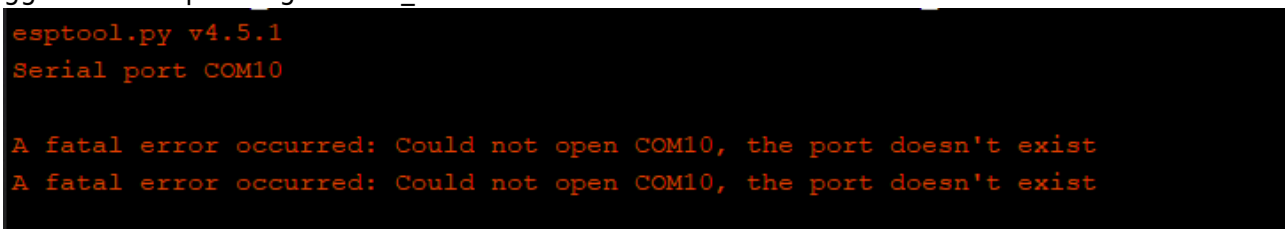
Da das Board keinen direketen nutzbaren Anschluss hat kann man sich so behelfen:



Im Schaltplan ist das an der Schottky Diode:



Probleme

- TFT_eSPI ist der fasche Grafiktreiber → lvyx… muss rein
  - Es muss auch eine extra konfig Klasse eingebaut werden !
- Das Startlogo (BTT_LOGO) ist zu groß
  Der C3 hat 400KB Ram, der Wroom 520kb
  Das Startbild ist aber ca. 110kb Groß und freier Speicher nach Knomi Start sind beim C3 nur ~76KB → Boot Loop !!
- Die Hintergrundbeleuchtung ist auf IO3 und nicht 2 oder 16
- Konvertierung des Sketch für Arduino IDE (Main Datei umbenennen)
- Zahlreiche Includes müssen umgebaut werden von <> nach “” → liegt an der Arduino IDE
- Es braucht das letzte Paket von ESP32 in der Arduino IDE, sonst ist der C3 nicht verfügbar
- Die Partition Table muss umgebaut werden. Sonst passt das BIN nicht in den Controller
- Die EEPROM Klasse funktioniert nicht richtig → Daten weren nicht gespeichert
- USB CDC macht ein wenig Probleme
  → Der Port ist nach dem Start in der Arduino IDE nicht mehr gültig (falsches Handle)
  → wenn sich der C§ aufhängt geht gar kein Upload mehr. Dann muss man den Controller manuell in den Boot Modus bringen → BOOT halten → Reset drücken → BOOT loslassen
  CDC muss auch extra aktiviert werden in der Arduino IDE !!
- Der Controller muss in der Arduino IDE umgestellt werden auf C3 Dev
- ggf. kleine Anpassungen an lv_conf.h
- 

Anpassungen / Script

- Neues Bootlogo
- Angepasste GIF Animationen
- Problem → HTTP Blockt wegen einem Kern
- Anpassung Status Abfrage Klipper !
  - https://www.klipper3d.org/Command_Templates.html#variables
- Teil drucken mit Aussparung
- Es gibt auch ein Display mit 2 kernen / 520k Ram → https://de.aliexpress.com/item/1005005644432428.html
- Voron Logo entfernt
- Logik angepasst
- lv_conf.h Anpassung wegen Farbne invert → #define LV_COLOR_16_SWAP 0
- GIF kann man extrahieren mit Seite …
  - http://tomeko.net/online_tools/hex_to_file.php?lang=en
  - Anpassen hiermit : https://ezgif.com/optimize
  - Zurückwandeln in Hex String hiermit : http://tomeko.net/online_tools/file_to_hex.php?lang=en
- Startbitmap
  - 240×240 Bixel RGB je 8 Farben → Export Gimp Optionen ! → 24 Bit
  - Converter : https://lvgl.io/tools/imageconverter
  - Color Format → CF_TRUE_COLOR

    - Output → C Array
    - Optionen aus
- Gif Tools
    - https://onlinegiftools.com/optimize-gif
- Moonraker Remote API
    - https://moonraker.readthedocs.io/en/latest/web_api/

Flashen

- D:\Projekte\Knomi\arduino-1.8.19\portable\packages\esp32\tools\esptool_py\4.5.1/esptool.exe –chip esp32c3 –port COM4 –baud 921600 –before default_reset –after hard_reset write_flash -z –flash_mode dio –flash_freq 80m –flash_size 4MB 0x0 C:\Users\DOMINI~1\AppData\Local\Temp\arduino_build_847170/KnomiC3.ino.bootloader.bin 0x8000 C:\Users\DOMINI~1\AppData\Local\Temp\arduino_build_847170/KnomiC3.ino.partitions.bin 0xe000 D:\Projekte\Knomi\arduino-1.8.19\portable\packages\esp32\hardware\esp32\2.0.11/tools/partitions/boot_app0.bin 0x10000 C:\Users\DOMINI~1\AppData\Local\Temp\arduino_build_847170/KnomiC3.ino.bin

```
esptool.exe --chip esp32c3 --port COM4 --baud 921600 --before default_reset --after hard_reset write_flash -z --flash_mode dio --flash_freq 80m --flash_size 4MB 0x0 KnomiC3.ino.bootloader.bin 0x8000 KnomiC3.ino.partitions.bin 0xe000 boot_app0.bin 0x10000 KnomiC3.ino.bin
```

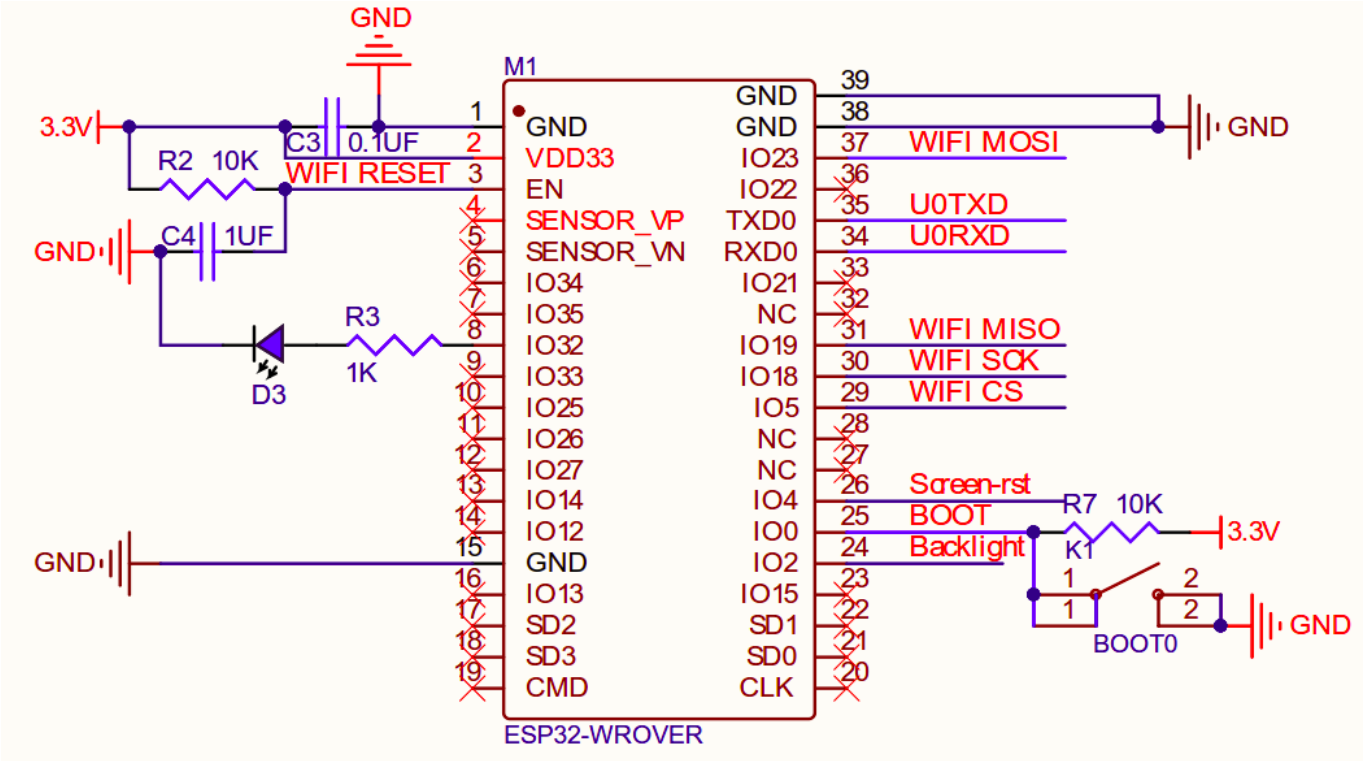
- ESP Tool EXE → https://github.com/espressif/esptool/releases

Hardware DIFF

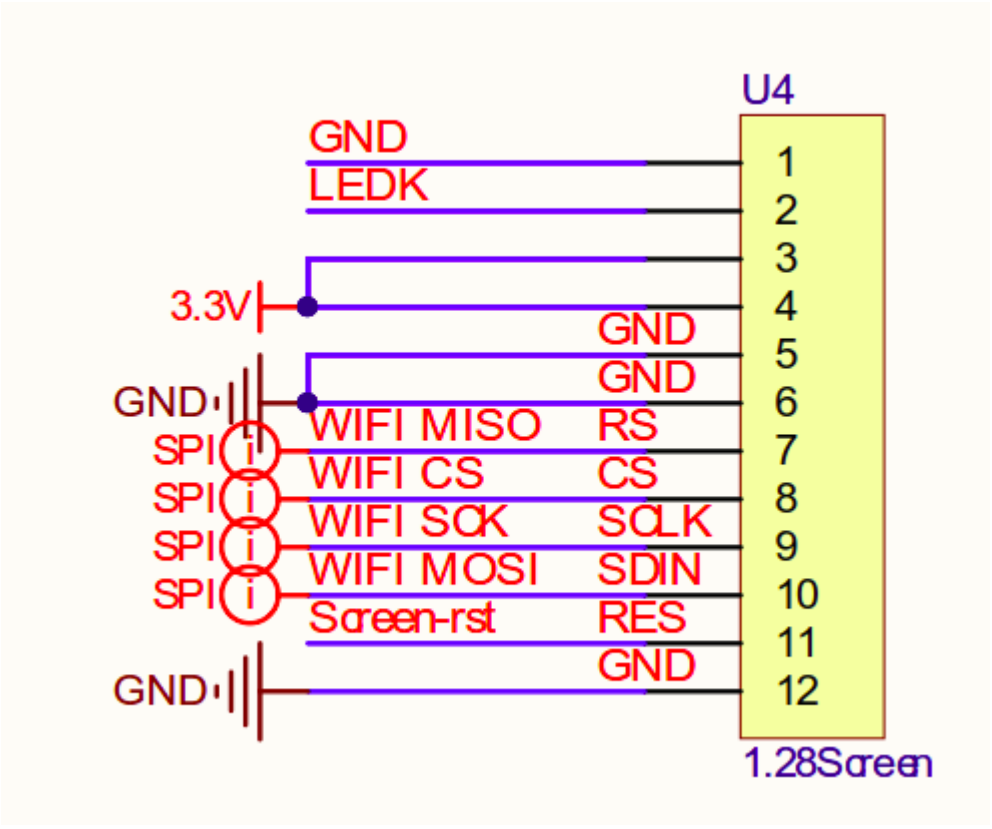
Funktion	Knomi Desc.	Knomi Pin	Knomi Display	Clone Desc.	Clone Pin	Clone Display
Boot Taster	BOOT	IO0		BOOT	IO9	
Backlight	Backlight	IO2	2 LEDK (vom Mosfet)		IO3	LED-
Bildschirm Reset	Screen-rst	IO4	11 RES	REST	EN	11 REST
WIFI Reset	WIFI RESET	EN				
WIFI Mosi	WIFI MOSI	IO23	10 SDIN	SDA	IO7	10 SPIMOSI
WIFI Miso	WIFI MISO	IO19	7 RS	D/C	IO2	7 SPIMISO
WIFI SCK	WIFI SCK	IO18	9 SCLK	SCL	IO6	9 SPICLK
WIFI CS	WIFI CS	IO5	8 CS	CS	IO10	8 SPICS
Uart TX	U0TXD	TXD0		USB_DN	IO18	
Uart RX	U0RXD	RXD0		USB-DP	IO19	
LED	??	IO32				

Hardware Knomi

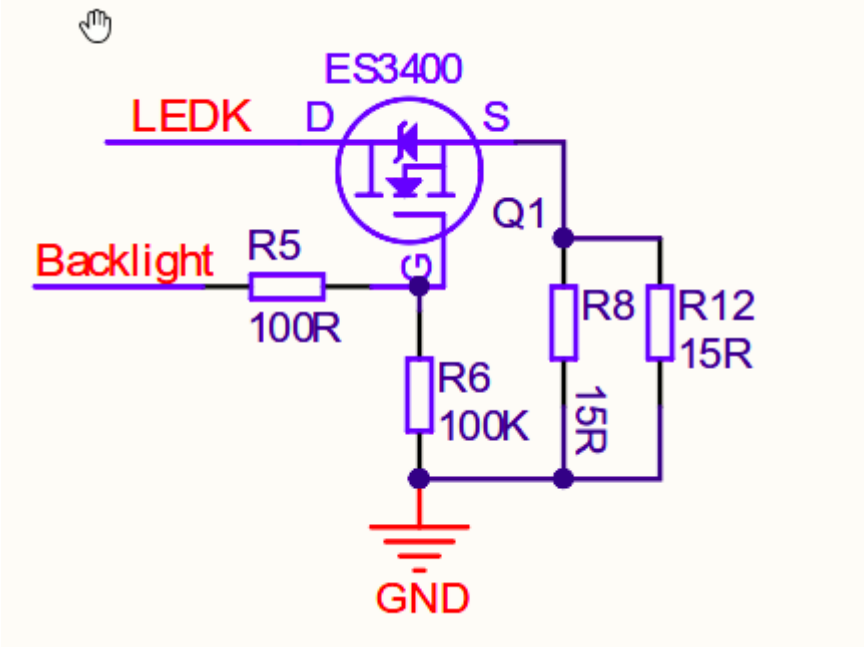
ESP



Display

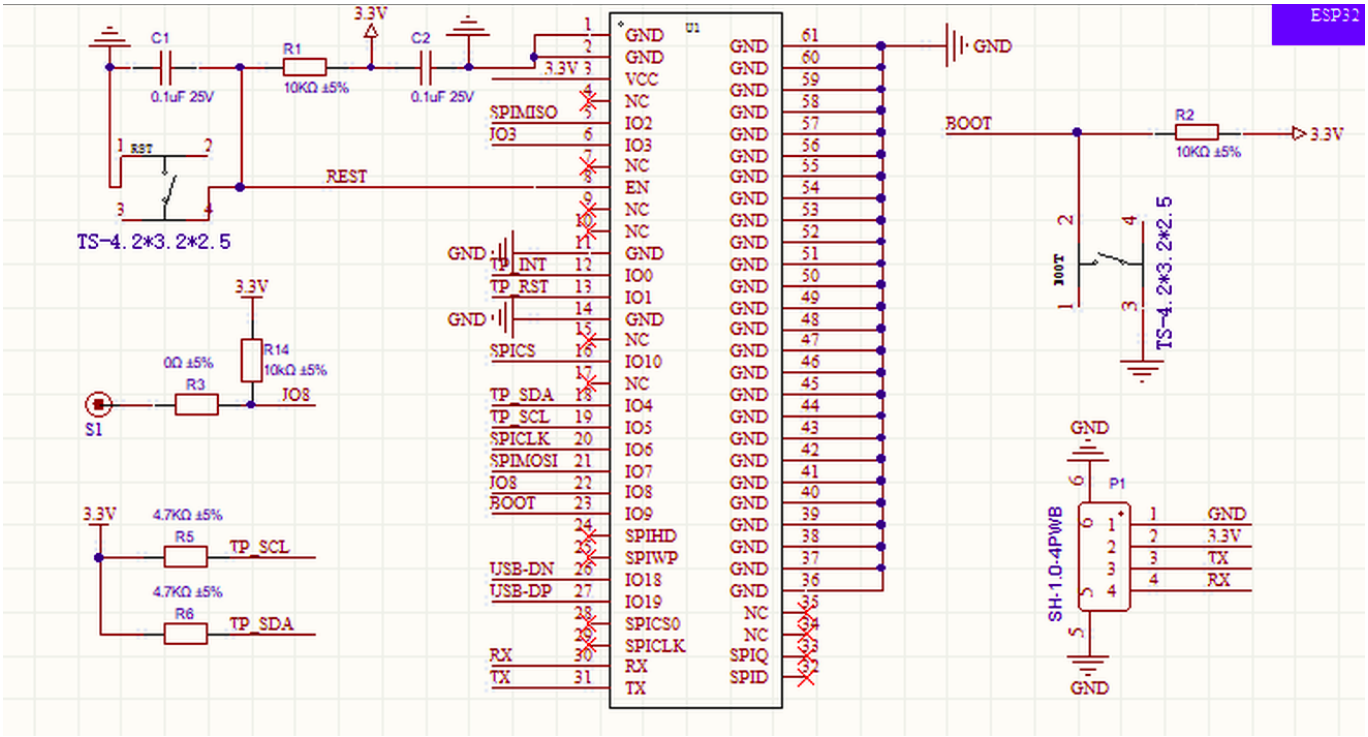


Backlight

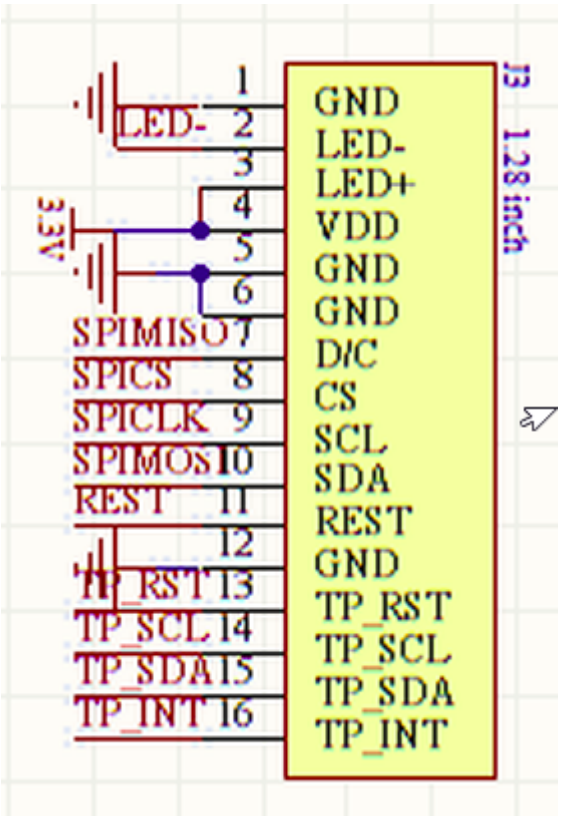


Hardware new

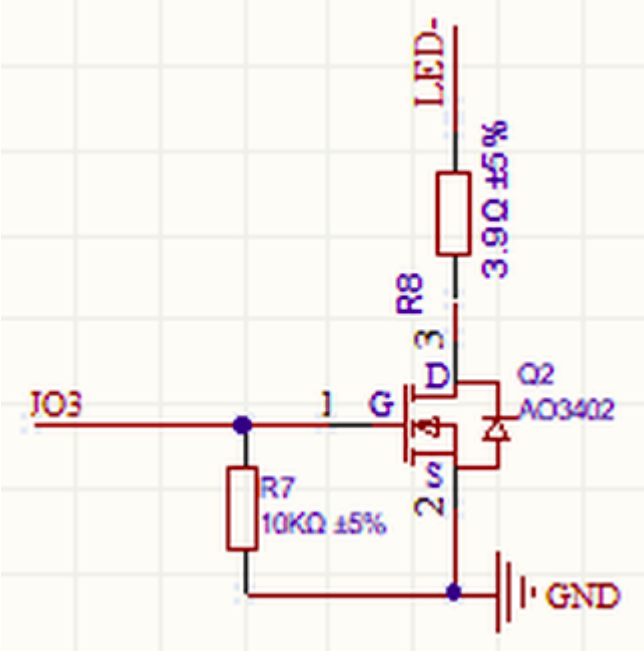
ESP



Display



Backlight



Links

- ESP Chip Info
  https://github.com/Jason2866/ESP32_Show_Info
  Beim C3 am Anfang ein dekay(4000) einbauen weil sonst keine seriellen Ausgaben kommen !
- https://github.com/bigtreetech/KNOMI
- https://bigtreetech.github.io/docs/KNOMI.html
- https://de.aliexpress.com/item/1005005453515690.html → Without Touch
- Display Libraries & Docs
  http://pan.jczn1688.com/directlink/1/ESP32%20module/1.28inch_ESP32-2424S012.zip?spm=a2

g0o.detail.1000023.1.1d16270bygRPys&file=1.28inch_ESP32-2424S012.zip
- https://bigtreetech.github.io/docs/KNOMI.html#
- https://github.com/DrKlipper/KNOMI_C3/

Direkt flashen

- `esptool.exe --chip esp32c3 --port COM4 --baud 921600 --before default_reset --after hard_reset write_flash -z --flash_mode dio --flash_freq 80m --flash_size 4MB 0x0 KnomiC3.ino.bootloader.bin 0x8000 KnomiC3.ino.partitions.bin 0xe000 boot_app0.bin 0x10000 KnomiC3.ino.bin`

Firmware compilieren

https://github.com/DrKlipper/KNOMI_C3

Klipper Konfig

Verbindung einrichten

Reset

`esptool.exe –port COM4 erase_flash`

Zukunft

520kb Display