5 Exkurs: Zeit-Dienst-Client mit TTGO-Display

Mit den Ergebnissen der letzten Kapitel ist es nicht schwer, den ESP32 so zu programmieren, dass er Datum und Uhrzeit auf dem Display des TTGO-Moduls anzeigt. Damit können wir die Anzeige auch "stand-alone", d.h. ohne Anschluss an einen PC, betreiben. Als elektrische Quelle bietet sich eine Powerbank an.

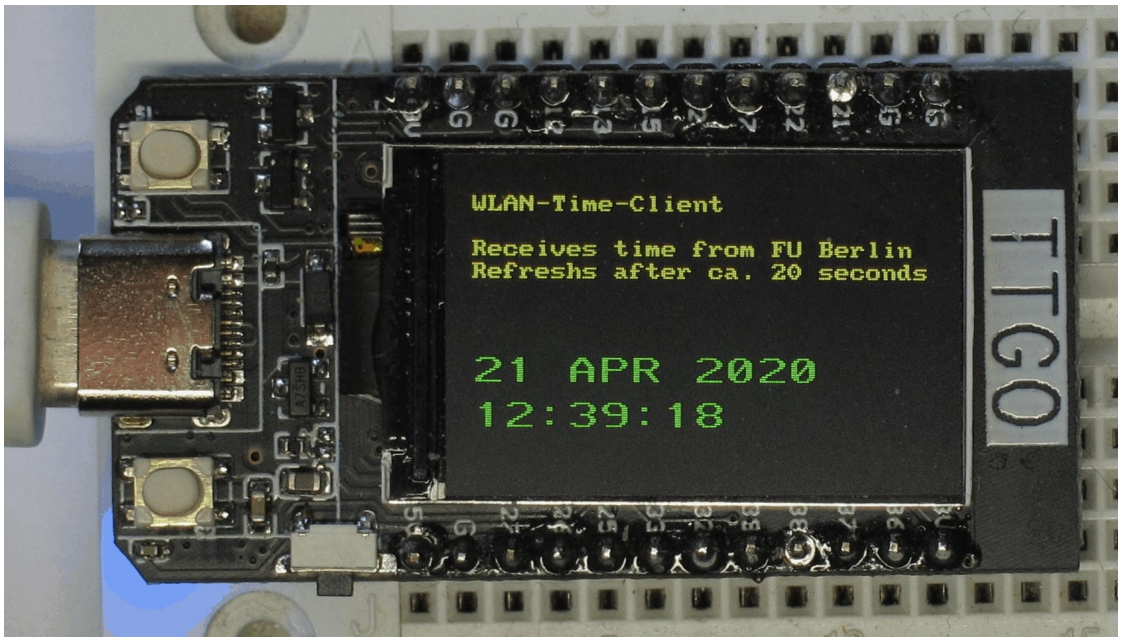


Abb. 1

Im Wesentlichen können wir das Programm `sta_client_time_3.py` übernehmen. Zunächst entfernen wir alle `print`-Befehle, bis auf die Ausgabe von Datum und Zeit. Als einzige Fehlermeldung benutzen wir nur die pauschale Meldung `'Disconnected!'`.

Jetzt müssen wir

- das Display initialisieren,
- die übrig gebliebenen `print`-Befehle durch entsprechende Ausgabe-Anweisungen für das Display ersetzen.

Wer mag kann auch einen Titel und kurze Erläuterungen zum Programm wie in der Abb. 1 einfügen.

Beachten Sie: Alle folgenden Informationen beziehen sich nur auf das TTGO-T-Display-Board und die im Kapitel V angegebene Firmware!

Initialisierung

Zur Initialisierung unseres Displays geht man folgendermaßen vor:

```
from machine import Pin, SPI
import st7789

spi = SPI(2, baudrate=20000000, polarity=1, sck=Pin(18), mosi=Pin(19))
display = st7789.ST7789(spi, 135, 240,  reset=Pin(23, Pin.OUT), cs=Pin(5,
      Pin.OUT), dc=Pin(16, Pin.OUT), backlight=Pin(4, Pin.OUT), rotation=3)
      #landscape
display.init()
display.fill(0)
```

Das Display besitzt einen Treiber-Baustein vom Typ ST7789; dieser wird über SPI angesteuert. Zunächst erzeugen wir die beiden Objekte `spi` und `display`. Dabei teilen wir mit, an welchen Pins des ESP32 die Steuerleitungen (sck, mosi, ...) des Displays angeschlossen sind. Die benötigten Klassen importieren wir vorher aus den Modulen `machine` und `st7789`. Anschließend wird das Display mit `display.init()` initialisiert und mit `display.fill(0)` gelöscht; genauer gesagt erfolgt das Löschen, indem sämtliche Pixel des Displays den Farbwert 0 erhalten. Dieser entspricht der Farbe Schwarz. Detaillierte Hinweise zur Bedeutung der Parameter bei den Funktionen `SPI` und `ST7789` befinden sich am Ende dieses Kapitels.

Ein Blick hinter die Kulissen: Farben beim ST7789-Display

Der Display-Treiber arbeitet mit 16-Bit-Farben in der RGB565-Notation. Glücklicherweise stellt die Klasse `st7789` eine Methode zur Verfügung, welche die üblichen r-g-b-Werte in das benötigte Format umwandelt:

```
st7789.color565(r, g, b)
```

Noch einfacher ist es, mit den Farben zu arbeiten, die als Konstanten fester Bestandteil der Klasse `st7789` sind, z. B. `RED, GREEN,` usw.

Durch den Befehl `display.fill(st7789.RED)` wird z. B. das ganze Display mit Rot ausgefüllt.

Tipp: Wer sich alle Funktionen und Konstanten der Klasse `st7786` anzeigen lassen möchte, gibt im Terminal von Thonny ein:

```
import st7786
dir(st7786)
```

Ausgabe von Zeit und Datum auf dem TTGO-Display

Zur Ausgabe von Zeichenketten werden **Zeichensätze** benötigt. Die von uns benutzte Firmware stellt bereits eine Reihe von Zeichensätzen zur Verfügung; sie müssen also nicht mehr auf dem ESP32 installiert werden. Die Zeichensätze, welche wir in unserem Programm einsetzen wollen, müssen allerdings importiert werden. Ich habe mich hier für die Zeichensätze `vga1_8x8` (kleine Schriftgröße, keine Sonderzeichen) und `vga1_16x16` (große Schriftgröße, keine Sonderzeichen) entschieden:

```
import vga1_8x8 as font1
import vga1_16x16 as font2
```

Die Zeichenkette `s = str(data,'UTF-8')` aus unserem Daytime-Programm soll nun mit dem `font1` auf dem Display ausgegeben werden. Dies geschieht mit dem Befehl

```
display.text(font1, s , 20, 40)
```

Dabei bilden 20 und 40 die x- bzw. y-Koordinate der linken oberen Ecke des Textfeldes. Man beachte, dass die oberste Zeile des Displays die y-Koordinate 0 besitzt.

Da die vom Server empfangenen Zeichenketten immer gleich lang sind, wird die alte Ausgabe beim Durchlaufen der Request-Schleife immer vollständig durch die neue überschreiben werden. Somit ist es nicht erforderlich, vor jeder neuen Daytime-Ausgabe das Display zu löschen.

Die Ausgabe von `'Disconnected!'` soll mit dem font2 unterhalb der Daytime-Anzeige erfolgen; sie soll nun in der Schriftfarbe Rot mit der Hintergrundfarbe Grün geschrieben werden. Dazu benutzen wir die Anweisung

```
display.text(font2, 'Disconnected!' , 20, 80, st7789.RED, st7789.GREEN)
```

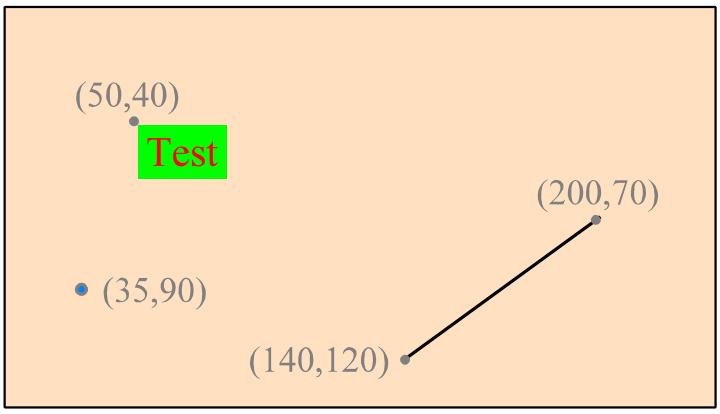
Hinter den Koordinaten tauchen in der Parameterliste zwei Farbwerte auf: Der erste Farbwert gibt die Schriftfarbe an, der zweite die Hintergrundfarbe. Das Modul st7789 stellt eine Reihe von häufig benutzten Farben zur Verfügung: BLACK, BLUE, CYAN, GREEN, MAGENTA, RED, WHITE, YELLOW. Weitere Farben kann man mit Hilfe der r-, g- und b-Werte bilden (s. o.).

Das Ergebnis ist in Abb. 2 zu sehen.

Abb. 2

Ein Blick hinter die Kulissen: **Koordinaten und wichtige Befehle**

Das Display ist 240x135 Pixel groß. Es lässt sich im Landscape- und im Portrait-Modus betreiben. Die oben angegebene Initialisierung stellt den Landscape-Modus ein. Einzelne Pixel, Strecken und Texte können Pixel-genau gesetzt werden. Das Pixel in der linken oberen Ecke hat die Koordinaten (0,0), das Pixel in der rechten unteren Ecke hat die Koordinaten (239,134).



In der obigen Abbildung sind drei Objekte auf das Display gezeichnet worden: Eine Textbox, ein Pixel, eine Strecke (line). Die benutzten Befehle lauten:

```
display.text(font2, 'Test', 50, 40, st7789.RED, st7789.GREEN)

display.pixel(35, 90, st7789.BLUE)

display.line(140, 120, 200, 70, st7789.BLACK)
```

Die Angaben zur Farbe bei der Text-Ausgabe können auch entfallen; in diesem Fall werden hier die Standardwerte benutzt (Weiß für die Textfarbe, Schwarz für die Hintergrundfarbe). Wird eine Textzeile über eine bereits bestehende (mit der gleichen Länge) geschrieben, so wird diese durch die Hintergrundfarbe der Textbox vollständig überschrieben.

Wer Datum und Uhrzeit in einer größeren Schrift (z. B. mit font2) anzeigen lassen möchte, der muss dazu Datum und Uhrzeit in zwei Zeichenketten zerlegen (Der Platz reicht sonst nicht aus!) und diese dann in jeweils einer eigenen Textbox zur Anzeige bringen. Ein entsprechendes Programm (wie in Abb. 1 benutzt) findet man in der Datei `wlan_client_time_to_display_1.py`.

Ein Blick hinter die Kulissen: Die Parameter von SPI und ST7789

Der ESP32-Baustein stellt vier SPI-Interfaces zur Verfügung; zwei davon (SPI0 und SPI1) stehen dem Benutzer generell nicht zur Verfügung; die anderen beiden (SPI2 und SPI3) werden häufig als HSPI und VSPI (mit den Identifiern 1 bzw. 2) bezeichnet. Die Anschlüsse dieser beiden Interfaces sind standardmäßig:

	HSPI (id=1)	VSPI (id=2)
`sck`	14	18
`mosi`	13	23
`miso`	12	19

Schaut man sich die Verdrahtung des Display-Controllers ST7789 auf dem TTGO-T-Display-Board an (s. Schematic.pdf), so stellt man fest: Der SCK-Eingang des Displays (SCLK) ist mit GPIO18 verbunden. Der MOSI-Eingang (SDA) ist mit GPIO19 verbunden; hier liegt also eine Abweichung von der Standardbelegung vor. Die gefundenen Zuordnungen des Schaltplans stimmen auch mit den Angaben auf S. 1 von Kapitel V überein.) Wir deswegen nun das VSPI-Interface (id = 2) und erzeugen eine zugehörige Instanz mit den gefundenen Verbindungen:

```
spi = SPI(2, baudrate=20000000, polarity=1, sck=Pin(18), mosi=Pin(19))
```

Der erste Parameter gibt hier den Identifier an, der zweite die Taktrate des Clock-Signals. Die Kennziffer 1 beim polarity-Parameter besagt, dass das Clock-Signal im Ruhezustand auf Low liegt und die Datenbits jeweils bei einer negativen Flanke des Clock-Signals gelesen werden.

Kommen wir nun zu den Parametern der Methode ST7789:

```
display = st7789.ST7789(spi, 135, 240, reset=Pin(23, Pin.OUT), cs=Pin(5,
      Pin.OUT),  dc=Pin(16,  Pin.OUT),  backlight=Pin(4,  Pin.OUT),
      rotation=3)
```

Der erste Parameter bezeichnet das vom Display-Controller zu benutzende SPI-Interface-Objekt. Die beiden nächsten Parameter geben die Höhe und die Breite (im Landscape-Modus) an. Dieser Landscape-Modus (so wie er in Abb. 1 zu sehen ist), wird durch den letzten Parameter `rotation = 3` eingestellt. Über die restlichen Parameter werden die Ausgänge für die folgenden Steuersignale festgelegt:

`reset`	Über dieses Signal kann der Display-Controller zurückgesetzt werden.
`cs`	Über diesen Signal wird das Display für die SPI-Kommunikation aktiviert (**C**hip **S**elect).
`dc`	Low- bzw. High-Zustand legen fest, ob es sich bei den gerade übermittelten Bytes um Daten (**D**ata) oder Befehle (**C**ommands) handelt.
`backlight`	Über den Ausgang GPIO4 wird die Hintergrund-Beleuchtung gesteuert.