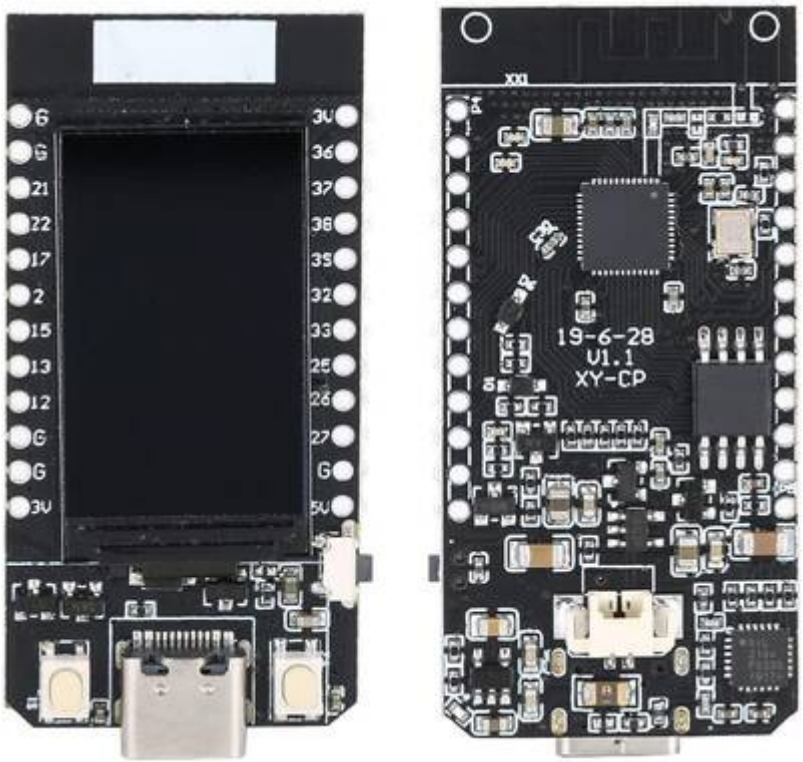
code, ESP32, ESP8266, Hardware, Nuetzliche Links, Projekte, Display

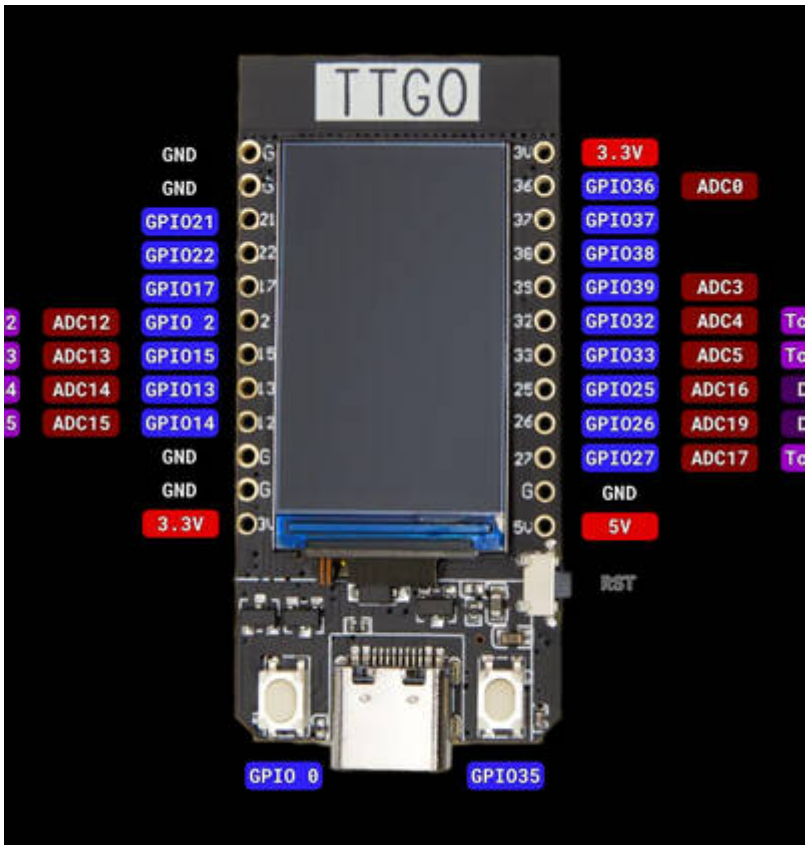
ESP32 TTGO T-Display

Spezifikation



Material: PCB Größe: Ca.51,52 x 25,04 x 8,54 mm/2,03 x 0,99 x 0,34 Zoll Gewicht: Ca.7,81 g
Hardware-Spezifikationen: Chipsatz: Für ESPRESSIF-ESP32 240MHz Xtensa?Einzel-/Doppelkern-32-Bit-LX6-Mikroprozessor
Flash: Für QSPI-Flash 4 MB SRAM: 520 kB SRAM-Taste: USB auf TTL zurücksetzen: CP2104
Modulare Schnittstelle: UART, SPI, SDIO, I2C, LED-PWM, TV-PWM, I2S, IRGPIO, ADC, Kondensator-Berührungssensor, DACLNA-Vorverstärker
Anzeige: IPS ST7789V 1,14 Zoll Arbeitsspannung: 2,7 V-4,2 V Arbeitsstrom: Ca.60 MA
Schlafstrom: Ca.120uA Arbeitstemperaturbereich: -40 ℃ - + 85 ℃ Netzteilspezifikationen:
Stromversorgung: USB 5V/1A Ladestrom: 500mA Batterie: 3,7V Lithiumbatterie JST-Anschluss: 2Pin 1,25 mm
USB: WLAN-Standard Typ C: FCC/CE-ROT/IC/TELEC/KCC/SRRC/NCC (esp32-Chip)
Protokoll: 802.11 b/g/n (802.11n, Geschwindigkeit bis zu 150 Mbit/s) A-MPDU- und A-MSDU-Polymerisation, Unterstützung von 0,4 µS Schutzintervall
Frequenzbereich: 2,4 GHz-2,5 GHz (2400 M-2483,5 M) Sendeleistung: 22 dBm
Kommunikationsentfernung: Ca.300 m Bluetooth-Protokoll: Erfüllen Sie die Bluetooth-Standards v4.2BR/EDR und BLE
Radiofrequenz: Mit einer Empfindlichkeit von -97 dBm. NZIF-Empfänger Klasse 1, Klasse 2, Klasse 3, Sender AFH
Audiofrequenz: CVSD SBC-Audiofrequenz Software-Spezifikation: Wi-Fi-Modus: Station/SoftAP/SoftAP + Station/P2P
Sicherheitsmechanismus: WPA/WPA2/WPA2-Enterprise/WPS-Verschlüsselungstyp: AES/RSA/ECC/SHA
Firmware-Upgrade: UART-Download/OTA (Über Netzwerk/Host, um Firmware herunterzuladen und zu

schreiben)
Software-Entwicklung: Unterstützung für Cloud-Server-Entwicklung/SDK für die Entwicklung von Benutzer-Firmware
Netzwerkprotokoll: IPv4, IPv6, SSL, TCP/UDP/HTTP/FTP/MQTT
Benutzerkonfiguration: AT + Befehlssatz, Cloud-Server, für Android/iOS-App
Betriebssystem: Für FreeRTOS



Nützliche Links

- Xinyuan-LilyGO / TTGO-T-Display (Library github)
- LilyGO TTGO T-display ESP32 (Arduino)
- ESPHome ST7789V TFT LCD

Video

Code

ttgo-display.yaml

```
# ESPHome code for the LilyGO TTGO Display
# Copyright 2023 by Smart Home Junkie
#
# Visit my website at https://www.smarthomejunkie.net
# Watch the tutorial for this display and code at
https://youtu.be/LJCeelAzlS0
substitutions:
  devicename: ttgo-display
  upper_devicename: TTGO Display

esphome:
  name: $devicename
  friendly_name: $upper_devicename

esp32:
  board: esp32dev
  framework:
    type: arduino

# Enable logging
logger:
  level: DEBUG

packages:
  wifi: !include packages/wifi.yaml
  mqtt: !include packages/mqtt.yaml
  ota:  !include packages/ota.yaml
  time: !include packages/time.yaml
  font: !include packages/font.yaml

# Enable Home Assistant API
api:

captive_portal:

spi:
  clk_pin: GPIO18
  mosi_pin: GPIO19

# Define the cycle variable. This indicates if the pages should be
cycled or not
globals:
  - id: cycle
    type: boolean
    initial_value: "true"
```

```
# Define binary sensors
binary_sensor:
  - platform: gpio          # Short Press button 0
    pin:
      number: GPIO0
      inverted: true
      mode:
        input: true
        pullup: true
    name: "Short Press Button 0"
    id: short_press_button_0
    on_click:
      min_length: 1ms
      max_length: 1000ms
      then:
        - display.page.show_previous: my_display
        - component.update: my_display

  - platform: gpio         # Long Press button 0
    pin:
      number: GPIO0
      inverted: true
    name: "Long Press Button 0"
    id: long_press_button_0
    on_click:
      min_length: 1001ms
      max_length: 5000ms
      then:
        - switch.toggle: backlight

  - platform: gpio          # Short Press button 1
    pin:
      number: GPIO35
      inverted: true
    name: "Short Press Button 1"
    id: short_press_button_1
    on_click:
      min_length: 1ms
      max_length: 1000ms
      then:
        - display.page.show_next: my_display
        - component.update: my_display

  - platform: gpio          # Long Press button 1
    pin:
      number: GPIO35
      inverted: true
    name: "Long Press Button 1"
    id: long_press_button_1
    on_click:
```

```
      min_length: 1001ms
      max_length: 5000ms
      then:
        - if:
            condition:
              lambda: |-
                return id(cycle);
            then:
              globals.set:
                id: cycle
                value: "false"
            else:
              globals.set:
                id: cycle
                value: "true"

# Allow dimmable control of the backlight (pin GPIO4) - Currently not
working
output:
  - platform: ledc
    pin: GPIO4
    id: gpio4

light:
  - platform: monochromatic
    output: gpio4
    name: "Backlight"

switch:
  - platform: gpio
    pin: GPIO4
    id: backlight
    internal: true

# Define all the numeric sensors used
sensor:
  - platform: homeassistant
    entity_id: sensor.youtube_subscriber
    id: subscriptions

  - platform: homeassistant
    entity_id: sensor.youtube_views
    id: views

  - platform: homeassistant
    entity_id: sensor.plug_02_power
    id: homelab

  - platform: homeassistant
    entity_id: sensor.wohnklima_temperature_2
    id: wohnklima_temperature
```

```
  - platform: homeassistant
    entity_id: sensor.wohnklima_humidity_2
    id: wohnklima_humidity

  - platform: homeassistant
    entity_id: sensor.wohnklima_equivalent_sea_level_pressure_2
    id: wohnklima_pressure

  - platform: homeassistant
    entity_id: sensor.dw2_02_luminosity
    id: wohnklima_light_sensor

  - platform: homeassistant
    entity_id: sensor.openweathermap_temperature
    id: outside_temperature

  - platform: homeassistant
    entity_id: sensor.openweathermap_wind_speed
    id: wind_speed

  - platform: homeassistant
    entity_id: sensor.xbt_eur_bid
    id: bitcoin

# Define all the string sensors used
text_sensor:
  - platform: homeassistant
    entity_id: sensor.openweathermap_condition
    id: weather_condition
    filters:
      - to_upper:

# Define colors
color:
  - id: RED
    red: 100%
    green: 0%
    blue: 0%
  - id: GREEN
    red: 0%
    green: 100%
    blue: 0%
  - id: BLUE
    red: 0%
    green: 0%
    blue: 100%
  - id: YELLOW
    red: 100%
    green: 100%
    blue: 0%
```

```
  - id: WHITE
    red: 100%
    green: 100%
    blue: 100%
  - id: ORANGE
    red: 100%
    green: 67%
    blue: 20%

# Define all the images used. Store the images in the images folder
within the esphome folder (esphome/images/)
image:
  - file: "images/youtube.png"
    id: youtube_image
    resize: 80x80
    type: RGB24
  - file: "images/logo.jpg"
    id: logo
    resize: 120x120
    type: RGB24
  - file: "images/electricity-icon.png"
    id: electricity_image
    resize: 80x80
    type: RGB24
  - file: "images/bitcoin_logo.png"
    id: bitcoin_logo
    resize: 80x80
    type: RGB24
  - file: "images/wind_icon.png"
    id: wind_icon
    resize: 30x30
    type: RGB24
  - file: "images/thermometer_icon.png"
    id: thermometer_icon
    resize: 30x30
    type: RGB24

# Define animations
animation:
  - file: "images/weather.gif"
    id: weather_animation
    resize: 80x80
    type: RGB24

# Define the graphs for sensors that you want to show on the display
graph:
  - id: wohnklima_temperature_graph
    sensor: wohnklima_temperature
    duration: 4h
    width: 220
    height: 90
```

```
    x_grid: 1h
    y_grid: 5
    min_range: 5
    max_range: 35
    min_value: 5
    max_value: 35
    color: GREEN
  - id: wohnklima_humidity_graph
    sensor: wohnklima_humidity
    duration: 4h
    width: 220
    height: 90
    x_grid: 1h
    y_grid: 25
    min_range: 1
    max_range: 100
    min_value: 1
    max_value: 100
    color: BLUE
  - id: wohnklima_pressure_graph
    sensor: wohnklima_pressure
    duration: 4h
    width: 220
    height: 90
    x_grid: 1h
    y_grid: 100.0
    color: YELLOW
  - id: wohnklima_light_sensor_graph
    duration: 4h
    width: 220
    height: 90
    x_grid: 1h
    traces:
      - sensor: wohnklima_light_sensor
        color: ORANGE
        line_type: SOLID
        line_thickness: 5

# Define qr code locations
qr_code:
  - id: qrcode_wiki
    value: https://gatonero-wiki.duckdns.org

  - id: qrcode_wlan
    value: WIFI:S:Mettigel;T:WPA;P:start123;H:false;

# Set up the display. This is the main part of the code
display:
  - platform: st7789v
    model: TTGO_TDISPLAY_135x240
```

```
    backlight_pin: GPIO4
    cs_pin: GPIO5
    dc_pin: GPIO16
    reset_pin: GPIO23
    rotation: 270°
    update_interval: 1s
    id: my_display
    pages:
# Define the pages
      - id: showintro
# Intro
        lambda: |-
          it.image(0, 10, id(logo));
          it.printf(120, 10, id(latoblack_intro), WHITE, "GATONERO");
          it.printf(145, 50, id(latoblack_intro), WHITE, "HOME");
          it.printf(125, 90, id(latoblack_intro), WHITE, "ASSISTANT");

      - id: showtime
# Time'
        lambda: |-
          it.strftime(45, 20, id(latoblack), "%d-%m-%Y",
id(esptime).now());
          it.strftime(25, 55, id(latoblackheading1), "%H:%M:%S",
id(esptime).now());

      - id: showsubscribers
# YouTube Subscribers
        lambda: |-
          it.printf(0,0,id(latoblack), WHITE, "SUBSCRIBERS");
          it.image(0, 40, id(youtube_image));
          if (id(subscriptions).has_state()) {
            it.printf(95, 60, id(latoblack), WHITE, "%.0f",
id(subscriptions).state);
          } else {
            it.printf(95, 65, id(lato), WHITE, "LOADING...");
          }

      - id: showviews
# YouTube Subscribers
        lambda: |-
          it.printf(0,0,id(latoblack), WHITE, "VIEWS");
          it.image(0, 40, id(youtube_image));
          if (id(views).has_state()) {
            it.printf(95, 60, id(latoblack), WHITE, "%.0f",
id(views).state);
          } else {
            it.printf(95, 65, id(lato), WHITE, "LOADING...");
          }

      - id: showhomelab
# HomeLab
```

```
        lambda: |-
          it.printf(0,0,id(latoblack), WHITE, "HomeLab");
          it.image(0, 40, id(electricity_image));
          if (id(homelab).has_state()) {
            if (id(homelab).state> -1000) {
              it.printf(95, 60, id(latoblack), WHITE, "%.0f Watt",
id(homelab).state);
            } else {
              it.printf(95, 60, id(latobold), WHITE, "%.0f Watt",
id(homelab).state);
            }
          } else {
            it.printf(95, 65, id(lato), WHITE, "LOADING...");
          }

      - id: showbitcoin
# bitcoin
        lambda: |-
          it.printf(0,0,id(bitcoin_font), WHITE, "bitcoin");
          it.image(0, 40, id(bitcoin_logo));
          if (id(bitcoin).has_state()) {
            it.printf(95, 60, id(bitcoin_font), WHITE, "%.0f",
id(bitcoin).state);
          } else {
            it.printf(95, 65, id(lato), WHITE, "LOADING...");
          }

      - id: show_wohnklima_temperature_graph
# Wohnzimmer Temperatur
        lambda: |-
          if (id(wohnklima_temperature).has_state()) {
            it.printf(0,0,id(latoblack), WHITE, "TEMP: %.1f °C",
id(wohnklima_temperature).state);
            it.graph(10, 40, id(wohnklima_temperature_graph));
          } else {
            it.printf(0,0,id(latoblack), WHITE, "TEMPERATURE");
            it.printf(80, 65, id(lato), WHITE, "LOADING...");
          }

      - id: show_wohnklima_humidity_graph
# Wohnzimmer Feuchte
        lambda: |-
          if (id(wohnklima_humidity).has_state()) {
            it.printf(0,0,id(latoblack), WHITE, "HUM: %.0f %%",
id(wohnklima_humidity).state);
            it.graph(10, 40, id(wohnklima_humidity_graph));
          } else {
            it.printf(0,0,id(latoblack), WHITE, "HUMIDITY");
            it.printf(80, 65, id(lato), WHITE, "LOADING...");
          }
```

```
      - id: show_wohnklima_pressure_graph
# Wohnzimmer Luftdruck
        lambda: |-
          if (id(wohnklima_pressure).has_state()) {
            it.printf(0,0,id(latoblack), WHITE, "PRS: %.0f hPA",
id(wohnklima_pressure).state);
            it.graph(10, 40, id(wohnklima_pressure_graph));
          } else {
            it.printf(0,0,id(latoblack), WHITE, "PRESSURE");
            it.printf(80, 65, id(lato), WHITE, "LOADING...");
          }

      - id: show_wohnklima_light_sensor_graph
# Wohnzimmer Lichtsensor
        lambda: |-
          if (id(wohnklima_light_sensor).has_state()) {
            it.printf(0,0,id(latoblack), WHITE, "LUX: %.0f lx",
id(wohnklima_light_sensor).state);
            it.graph(10, 40, id(wohnklima_light_sensor_graph));
          } else {
            it.printf(0,0,id(latoblack), WHITE, "LUX");
            it.printf(80, 65, id(lato), WHITE, "LOADING...");
          }

      - id: show_weather
# Wetter
        lambda: |-
          id(weather_animation).next_frame();
          it.image(0, 0, id(weather_animation), COLOR_ON, COLOR_OFF);
          if (id(weather_condition).state == "CLOUDY" ||
              id(weather_condition).state == "RAINY" ||
              id(weather_condition).state == "FOG" ||
              id(weather_condition).state == "HAIL" ||
              id(weather_condition).state == "SNOWY" ||
              id(weather_condition).state == "SUNNY" ||
              id(weather_condition).state == "WINDY" ) {
            it.printf(110,0,id(latoblack),
WHITE,"%s",id(weather_condition).state.c_str());
          }
          if (id(weather_condition).state == "LIGTNING" ||
              id(weather_condition).state == "POURING" ) {
            it.printf(110,0,id(latobold),
WHITE,"%s",id(weather_condition).state.c_str());
          }
          if (id(weather_condition).state == "PARTLYCLOUDY") {
            it.printf(110,0,id(latoblack), WHITE,"PARTLY");
            it.printf(110,35,id(latoblack), WHITE,"CLOUDY");
          }
          if (id(weather_condition).state == "SNOWY-RAIN") {
            it.printf(110,0,id(latoblack), WHITE,"SNOWY");
            it.printf(110,35,id(latoblack), WHITE,"RAIN");
```

```
          }
          if (id(weather_condition).state == "LIGHTNING-RAINY") {
            it.printf(110,0,id(latobold), WHITE,"LIGHTNING");
            it.printf(110,35,id(latobold), WHITE,"RAINY");
          }
          if (id(weather_condition).state == "WINDY-VARIANT") {
            it.printf(110,0,id(latoblack), WHITE,"WINDY");
          }
          if (id(weather_condition).state == "CLEAR-NIGHT") {
            it.printf(110,0,id(latoblack), WHITE,"CLEAR");
            it.printf(110,35,id(latoblack), WHITE,"NIGHT");
          }
          if (id(weather_condition).state == "EXCEPTIONAL") {
            it.printf(110,0,id(latobold), WHITE,"EXCEPTIONAL");
          }
          it.image(0, 100, id(thermometer_icon));
          it.printf(35,100,id(latobold), WHITE,"%.1f °C",
id(outside_temperature).state);
          it.image(110, 100, id(wind_icon));
          it.printf(145,100,id(latobold), WHITE,"%.2f m/s",
id(wind_speed).state);

      - id: showqrcode
# QR-Code dokuwiki MyWiki
        lambda: |-
          it.qr_code(60, 5, id(qrcode_wiki), WHITE, 5);

      - id: showqrwlan
# QR-Code dokuwiki Wlan
        lambda: |-
          it.qr_code(60, 5, id(qrcode_wlan), WHITE, 5);

# Define the cycle interval of the pages.
interval:
  - interval: 5s
    then:
      if:
        condition:
          lambda: 'return id(cycle);'
        then:
          - display.page.show_next: my_display
          - component.update: my_display
```