

Cheat Sheet für ESP32/ESP32-S3

Für Board-Support ab Version 3.x

Hauptquelle: Arduino Language Reference
<https://arduino.cc/de/Reference/>

WLAN - Funktionen

```
WiFi.h - WLAN-Funktionen, allgemein
Zunächst das WLAN-Kennwort angeben:
const char* ssid = "NetworkName";
const char* pass = "Password";
Alternative (besser beim Weitergeben von Programmen):
#include <credentials.h> // im LIB-Verzeichnis anlegen,
// Inhalt: die beiden Zeilen von oben
WiFi.begin(ssid, pass); // Startet WLAN zu ssid als Client
WiFi.begin(ssid); // Verbindung mit offenem WLAN ssid
IPAddress myIP = WiFi.localIP(); // IP-Adresse auslesen
```

```
WiFi.status(); // Lese Status der Verbindung z.B. WL_CONNECTED
WiFi.BSSID(bssid); // Liest die MAC-Adresse des Routers
long rssi = WiFi.RSSI(); // Liest die Signalstärke aus
WiFi.disconnect(); // Verbindung trennen
```

WiFiAP.h - WLAN-Funktion Accesspoint

```
WiFi.SoftAP(ssid, pass); // Startet einen Accesspoint
Wifi.SoftAP(ssid); // offener Accesspoint
IPAddress myIP = WiFi.softAPIP(); // IP-Adresse auslesen

const char * softAPgetHostname(); // AP-Name auslesen
bool softAPsetHostname(const char * hostname); // AP-Name festlegen

String softAPmacAddress(void); // MAC-Adresse festlegen
uint8_t * softAPmacAddress(uint8_t * mac); // MAC-Adresse auslesen
```

HTTP(S) - Server

WebServer.h - für HTTP und HTTPS Verbindungen

Zunächst eine Objektinstanz erzeugen:
WebServer server(80); // Port 80 für HTTP, Port 443 für HTTPS

In Setup():
server.begin(); // Webserver starten, WLAN vorher verbinden
server.onNotFound(Send404); // Callback-Funktion für ungültige URL
server.on("/", HandleRoot); // Callback-Funktion für Web-Root
Weitere URLs: weitere Callback-Fkt. oder in HandleRoot abfragen

Webseiten ausliefern mit Status-Code und Inhaltstyp:
server.send(200, "text/html", WebPage()); // WebPage liefert String
void Send404(void) { // hier als Funktion Send404()
 server.send(404, "text/plain", "Internet kaputt");
 // sende unformatierten Text
}

Auswerten von Server-Argumenten (für Formulare, Benutzereingaben):

```
void HandleRoot(void) {
    // DEBUG Ausgabe, zeigt alle Parameter des Aufrufs
    Serial.printf("URI: %s, ", server.uri());
    Serial.print("Method: ");
    Serial.println((server.method() == HTTP_GET) ? "GET" : "POST");
    Serial.printf(", Arguments: %s\n", server.args());
    // Alle Server-Argumente ausgeben
    for (uint8_t i = 0; i < server.args(); i++) {
        Serial.printf("%s : %s\n", server.argName(i), server.arg(i));
    }
    // ENDE DEBUG, jetzt nach bekannten Argumenten schauen;
    if(server.hasArg("r") || server.hasArg("g") || server.hasArg("b"))
    { uint16_t red, green, blue; String Temp;
        Temp = server.arg("r"); // Server sendet Strings
        red = Temp.toInt(); // String in Zahl umwandeln
        // jetzt was damit machen, z.B. LED-Farbe setzen
    }
    server.send(200, "text/html", WebPage()); // neue Webseite senden
}
```

Weitere Möglichkeiten siehe Programmbeispiele in Arduino!

Bluetooth - Funktionen

BluetoothSerial.h - Bluetooth zur Kommunikation

Zunächst eine Objektinstanz erzeugen:

```
BluetoothSerial SerialBT; // Name ist beliebig
In Setup():
SerialBT.begin("Geräte_1234"); // Eindeutiger Name
```

Danach können die gleichen Funktionen wie bei der seriellen Schnittstelle genutzt werden:

```
SerialBT.available(); // Empfangsdaten vorhanden?
SerialBT.println("Text");
Zeichen = SerialBT.read();
StringVar = SerialBT.readString();
SerialBT.end();
```

Weitere Funktionen für die Verbindungssteuerung:

```
void enableSSP(); // Secure Pairing einschalten
bool setPin(*pin); // z.B. SerialBT.setPin("1234");
bool connect(remoteName); // neue Verbindung
bool connect(); // gespeicherte Verbindung
bool connected(timeout=0); // Verbindung ok?
bool disconnect();
bool unpairDevice(remoteAddress[]);
```

Callback-Funktion für Verbindungs-Ereignisse:

```
SerialBT.register_callback(BT_Comm); // in Setup()
void BT_Comm(esp_spp_cb_event_t event,
             esp_spp_cb_param_t *param) {
    if(event == ESP_SPP_SRV_OPEN_EVT){
        Serial.println("Client verbunden");
    }
    if(event == ESP_SPP_CLOSE_EVT){
        Serial.println("Client getrennt");
    }
}
```

Weitere Event-Nummern und Definitionen:

docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/bluetooth/esp_spp.html

BluetoothA2DPSource.h und BluetoothA2DPSink.h

Bluetooth zur Audio-Übertragung (Quelle, Senke)

z.B. github.com/pschatzmann/ESP32-A2DP

Sleep - Modes

Der ESP32 kann in einen Sleep-Mode geschickt werden, in dem die Stromaufnahme stark reduziert ist. Vorher muss eine Quelle für den Wakeup definiert werden. Man kann auch eine feste Zeit einstellen, nach der der ESP wieder aufwacht.

```
esp_sleep_enable_ext0_wakeup (pin, [HIGH, LOW]);
esp_sleep_enable_timer_wakeup (microSeconds);
esp_deep_sleep_start (); // schlafen schicken
```

Touch - Eingänge

Der ESP32 hat 10 nutzbare Touch-Eingänge (ESP32-S3: 14). Die Touch-Pins müssen nicht mit PinMode initialisiert werden. Die Werte haben teilweise Störungen und sollten gefiltert werden.

```
byte TouchValue = touchRead(T0); // T0-T9
```

Rückgabewert 0 - 100, Touch-Ereignis bei Schwellwert 20-30 (je nach Beschaltung):

```
bool digitalTouch(T_Chan) {
    if(touchRead(T_Chan) < 30) return true;
    else return false;
```

Der ESP32 /ESP32-S3 hat 4 Universaltimer mit 64 Bit Zählfang Eingangstakt 80MHz mit Vorteiler (:1 bis :65535)

```
hw_timer_t *MeinTimer = NULL; // Zeiger auf eine Timer-Struktur
MeinTimer = timerBegin (Frequenz); // Timerfrequenz in Hz, wenn
// MeinTimer danach noch NULL ist, liegt ein Fehler vor
TimerStart (MeinTimer); // Timer starten
TimerStop (MeinTimer); // Timer stoppen
timerAlarm(MeinTimer, Alarmwert, Reload, Count); // 64 Bit Alarmwert
// Reload true: nach Alarm bei 0 anfangen, Count -> Anzahl
// der automatischen Reloads (0 -> unendlich)
timerAttachInterrupt(MeinTimer, &onTimer); // ISR onTimer()
uint64_t timerRead(MeinTimer); // Zählerstand lesen (64Bit!)
timerWrite(MeinTimer, uint64_t Wert); // Zählerstand setzen
timerRestart(MeinTimer); // Zählerstand auf 0 setzen
uint64_t timerReadMicros(MeinTimer); // Mikrosekunden
uint64_t timerReadMillis(MeinTimer); // Millisekunden
double timerReadSeconds(MeinTimer); // Sekunden
```

