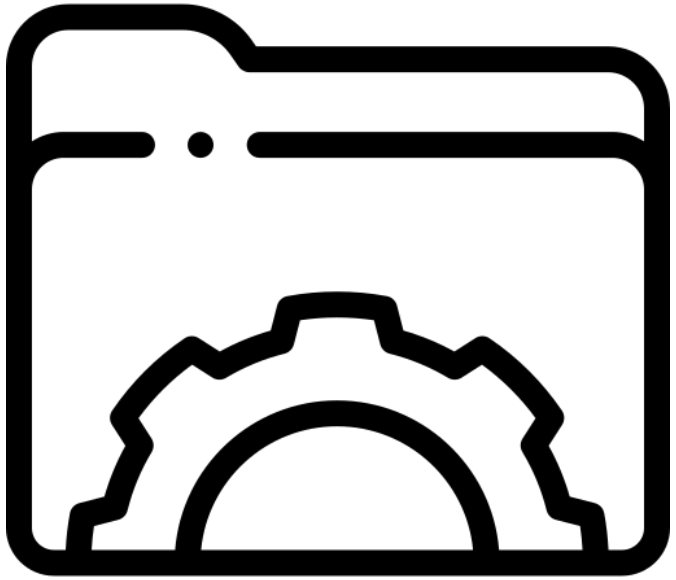
Dateisystem – SPIFFS

HTL ST. PÖLTEN



Quelle: Freepik on www.flaticon.com

SPIFFS

HTL ST. PÖLTEN

- Daten müssen auf dem ESP32 dauerhaft gespeichert werden
  - Konfigurationen
  - Logfiles
  - Sensordaten
  - Webserver Daten
  - …
- Benötigt Dateisystem → Serial Peripheral Interface Flash File System (SPIFFS)

SPIFFS – Funktionen

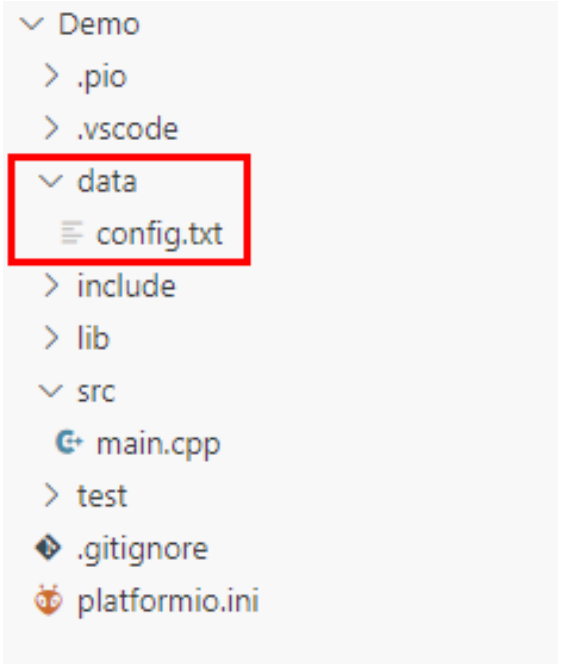
HTL ST. PÖLTEN

- Befehle in **`SPIFFS.h`**
- Dateiverwaltung
  - **C**reate
  - **R**ead
  - **U**pdate
  - **D**elete
- Flache Dateistruktur → keine Ordner
- Featuremäßig EEPROM überlegen

SPIFFS – PlatformIO

HTL ST. PÖLTEN

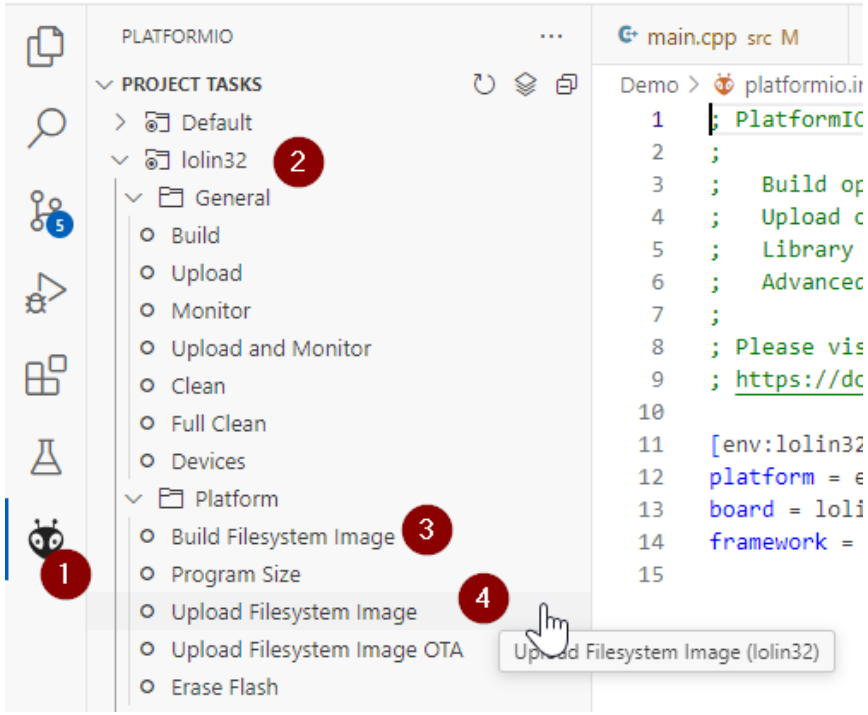
- Daten müssen in einen Projektordner „data“ abgelegt werden
- Ordner wird im Projekt (in VSCode) manuell angelegt
- Benötigte Dateien werden (in VSCode) manuell angelegt



SPIFFS – PlatformIO Upload

HTL ST. PÖLTEN

1. PlatformIO Plugin auswählen
2. Projekt Tasks öffnen
3. Build Filesystem Image
4. Upload Filesystem Image



Nach Fertigstellung SUCCESS Message im Terminal

SPIFFS – Initialisierung

HTL ST. PÖLTEN

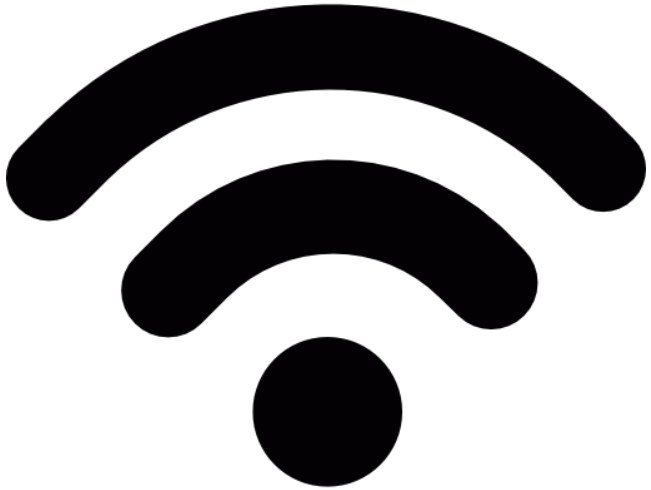
- `#include <SPIFFS.h>`
- Initialisierung von SPIFFS mit dem Befehl: `SPIFFS.begin()`
- Liefert true/false je nach Erfolg
- Vorzugsweise in einer eigenen Funktion `initSPIFFS()`, um die `setup()` Funktion klein zu halten

Demo > src > main.cpp > ...

```
1   #include <Arduino.h>
2   #include <SPIFFS.h>
3
4   void initSPIFFS();
5
6   void setup() {
7     Serial.begin(115200);
8     initSPIFFS();
9   }
10
11  void loop() {
12  }
13
14  void initSPIFFS() {
15    if (SPIFFS.begin())
16      printf("Successfully setup SPIFFS\n");
17    else
18      printf("Error setting up SPIFFS!\n");
19  }
```

WiFi

HTL ST. PÖLTEN



Quelle: Freepik on www.flaticon.com

WiFi

HTL ST. PÖLTEN

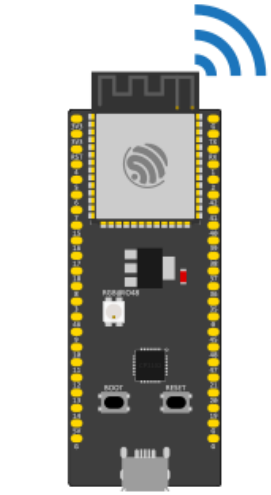
- Für Verbindung mit Clients notwendig
- Bietet Unterstützung für 802.11b/g/n
- Betrieb im Station mode bzw. AP mode möglich

- Library WiFi.h
- Dokumentation: https://espressif-docs.readthedocs-hosted.com/projects/arduino-esp32/en/latest/api/wifi.html

WiFi – Station Mode

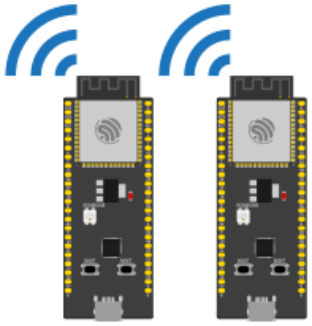
HTL ST. PÖLTEN

- Station Mode – STA
- Verwendung wenn sich ESP32 zu bereits bestehendem Netzwerk verbinden soll
- Beispielsweise bei Anbindung an Internet, Heimnetz, etc.



ESP32 as Wi-Fi Station (STA)

Wi-Fi Access Point (AP)



ESP32' as Wi-Fi Station (STA)

WiFi – STA Funktionen

HTL ST. PÖLTEN

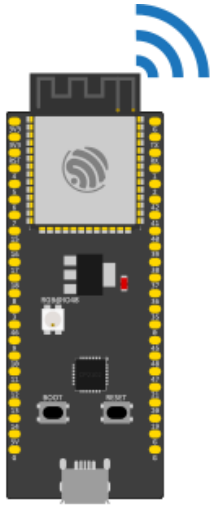
- WiFi Modus setzen → `WiFi.mode(WIFI_STA);`
- Verbindung zu bestehendem WLAN aufbauen → `WiFi.begin(ssid, password);`
- SSID und Passwort sollten zumindest nicht direkt im main Code sein → eventuell in separate Source Datei auslagern, die nicht weitergegeben wird (Git)

- Ausgabe der zugewiesenen IP → `WiFi.localIP().toString();`

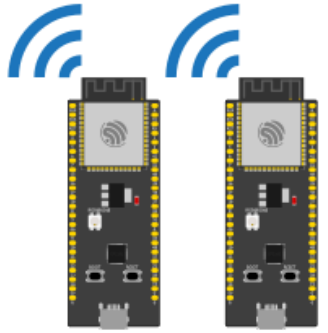
WiFi – AP mode

HTL ST. PÖLTEN

- ESP32 ist als Access Point (AP) konfiguriert
- ESP32 stellt eigenes WLAN bereit und vergibt IPs per DHCP
- Beispielsweise bei Verwendung als WebServer per HTTP oder auch HTTPS



ESP32 as Wi-Fi Access Point (AP)



ESP32' as Wi-Fi Station (STA)

WiFi – AP Funktionen

HTL ST. PÖLTEN

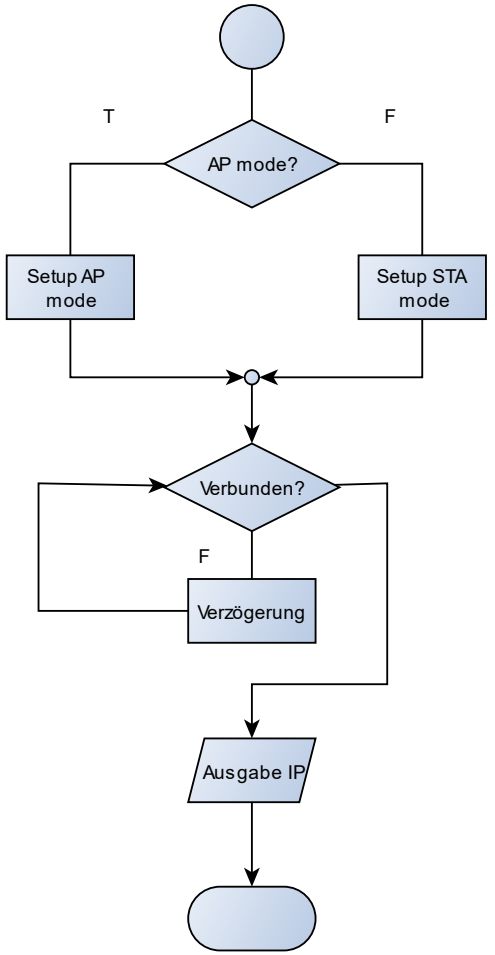
- WiFi Modus setzen → `WiFi.mode(WIFI_AP);`
- Neues WLAN aufbauen → `WiFi.softAP(ssid, password);`
- Bei unkritischen Anwendungen kann SSID und Passwort hard coded werden

- Ausgabe der zugewiesenen IP → `WiFi.softAPIP().toString()`

WiFi – Verbindungsaufbau

HTL ST. PÖLTEN

- Code zum Aufbauen/Initialisieren sollte in eigene Funktion ausgelagert werden → z.B: `initWiFi();`
- Sinnvollerweise inkl. Flag, das den Modus des WLAN-Adapters steuert (STA/AP) → z.B: `initWifi(bool apMode);`
- Funktion wird in setup() aufgerufen und kümmert sich um Verbindungsausbau



WiFi – Verbindungsaufbau

HTL ST. PÖLTEN

- Nach Start des Verbindungsaufbaus muss verzögert werden, bis eine erfolgreiche Verbindung besteht
- WiFi Statusabfrage → `WiFi.status()`
- Nach Erfolg ist Status `WL_CONNECTED`

```
Serial.println("Connection to WiFi . . .");
while ((WiFi.status() != WL_CONNECTED) && (startWiFi < 21))
{
    delay(1000);
    printf(" .");
    startWiFi++;
}
```