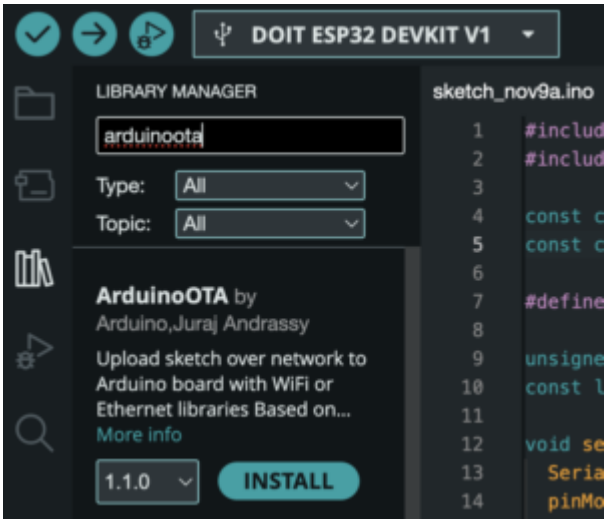
OTA Updates für den ESP32 – Aktualisierungen aus der Ferne

Der ESP32 unterstützt OTA (Over-the-Air), mit dem du den Sketches drahtlos aktualisieren kannst. OTA ist besonders hilfreich, wenn der Mikrocontroller schwer zugänglich ist oder du Änderungen ohne physische Verbindung übertragen möchtest. **In diesem Tutorial erfährst du, wie du OTA einrichtest und lernst Schritt für Schritt ein Beispielprojekt kennen: Eine blinkende LED, deren Blinkfrequenz per OTA-Update verändert wird.**

Die Bibliothek ArduinoOTA

Falls du die ESP32-Boards über den Boardverwalter der Arduino-IDE hinzugefügt hast, ist die Bibliothek **ArduinoOTA** normalerweise schon dabei. Die Bibliothek ermöglicht dir eine unkomplizierte Integration der OTA-Funktionalität.

Sollte die Bibliothek dennoch fehlen, kannst du sie über den **Bibliotheksverwalter** installieren. Gehe dazu in der Arduino-IDE zu **Sketch > Bibliothek einbinden > Bibliotheken verwalten** und suche nach „ArduinoOTA“. Stelle sicher, dass die neueste Version installiert ist, damit du alle aktuellen Features und Sicherheitsverbesserungen nutzen kannst.



Schritt 1: Der erste Sketch für eine blinkende LED

Du startest mit einem einfachen Sketch, der eine LED am ESP32 im Sekundentakt blinken lässt. **Die LED ist dabei über einen passenden Vorwiderstand an den GPIO-Pin 2 des ESP32 angeschlossen.** Dieser Sketch dient als Grundlage für das OTA-Update.

```
#include <WiFi.h>
#include <ArduinoOTA.h>

const char* ssid = "Dein_WLAN-NETZWERK";          // Ersetze durch deinen WLAN-Namen
const char* password = "Dein_WLAN_Passwort";  // Ersetze durch dein WLAN-Passwort

#define LED_PIN 2 // GPIO-Pin der LED

unsigned long previousMillis = 0;
const long interval = 1000; // Blinkintervall in Millisekunden (1 Sekunde)

void setup() {
  Serial.begin(115200);
```

```
  pinMode(LED_PIN, OUTPUT);

  // WLAN-Verbindung herstellen
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("\nVerbunden mit WiFi");

  // OTA-Setup
  ArduinoOTA.setHostname("esp32_led_ota");
   ArduinoOTA.setPassword("Dein_Sicheres_Passwort"); // Setze hier ein starkes Passwort für OTA-Updates, um unbefugten Zugriff zu verhindern
  ArduinoOTA.begin();
}

void loop() {
  ArduinoOTA.handle(); // Nach OTA-Updates suchen

  unsigned long currentMillis = millis();
  if (currentMillis - previousMillis >= interval) {
    previousMillis = currentMillis;
     digitalWrite(LED_PIN, !digitalRead(LED_PIN)); // Zustand der LED umschalten
  }
}
```

So funktioniert der Sketch

1. **Bibliotheken einbinden**: Der Sketch beginnt mit dem Einbinden der notwendigen Bibliotheken WiFi.h und ArduinoOTA.h. Erstere ermöglicht die Verbindung des ESP32 mit einem WLAN-Netzwerk, während ArduinoOTA.h die OTA-Funktionalität bereitstellt. Diese Bibliotheken sind notwendig, um die gewünschten Netzwerk- und Update-Funktionen auf dem ESP32 zu implementieren.
2. **Netzwerk-Konfiguration**: Die Variablen ssid und password speichern die Zugangsdaten für dein WLAN. Diese werden

verwendet, um den ESP32 mit deinem Netzwerk zu verbinden.

3. **GPIO-Pin-Definition**: #define LED_PIN 2 definiert den Pin, an den die LED angeschlossen ist. In diesem Fall nutzt du den GPIO-Pin 2 des ESP32.
4. **WLAN-Verbindung herstellen**: Im setup()-Teil des Codes wird die Verbindung zum WLAN hergestellt. Mit WiFi.begin(ssid, password) verbindet sich der ESP32 mit dem Netzwerk. Die Schleife while (WiFi.status() != WL_CONNECTED) sorgt dafür, dass das Programm wartet, bis die Verbindung hergestellt ist.
5. **OTA-Setup**: Im setup()-Teil wird auch die OTA-Funktionalität initialisiert. Mit ArduinoOTA.setHostname("esp32_led_ota") wird der Name des ESP32 im Netzwerk festgelegt. Dieser Name erleichtert es, das Gerät im Netzwerk zu identifizieren, besonders wenn du mehrere ESP32-Geräte verwendest. Ein (möglichst sicheres) Passwort hinterlegst du mit Hilfe der Funktion ArduinoOTA.setPassword("Mein_OTA_Passwort"), damit nur du Updates durchführen kannst. ArduinoOTA.begin() startet den OTA-Service, damit der ESP32 auf eingehende Updates wartet.
6. **LED-Blinken**: Die Funktion loop() enthält den Code, der die LED im Sekundentakt blinken lässt. Mit der Funktion millis() wird überprüft, ob der festgelegte Intervall (1000 Millisekunden) vergangen ist. Wenn dies der Fall ist, wird der Zustand der LED umgeschaltet mit digitalWrite(LED_PIN, !digitalRead(LED_PIN)).
7. **OTA-Handler**: In der loop()-Funktion wird ArduinoOTA.handle() aufgerufen, um kontinuierlich nach OTA-Updates zu suchen. Dies ermöglicht es, jederzeit ein Update zu empfangen, während das Hauptprogramm weiterläuft.

Schritt 2: OTA-Update zur Änderung des intervalls

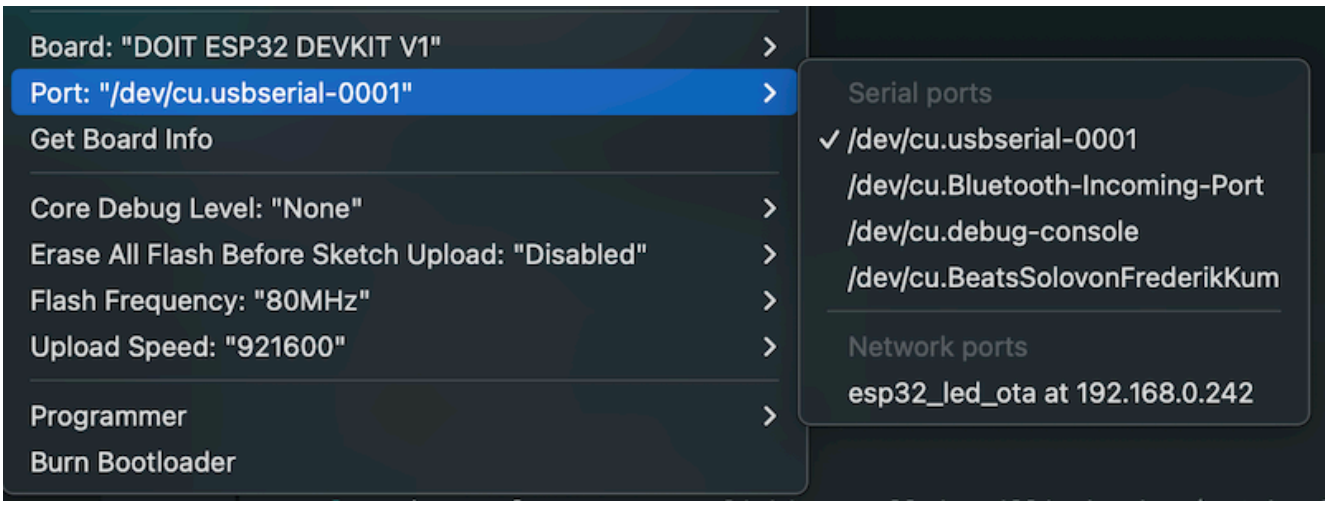
Nun nimmst du eine Änderung am Code vor, um die Blinkfrequenz der LED auf 500 ms zu reduzieren. Diese Änderung überträgst du drahtlos per OTA auf den ESP32.

Ändere im obigen Sketch den Wert des Intervalls von 1000 auf 500:

```
const long interval = 500; // Blinkintervall in Millisekunden (500 ms)
```

Update via OTA durchführen

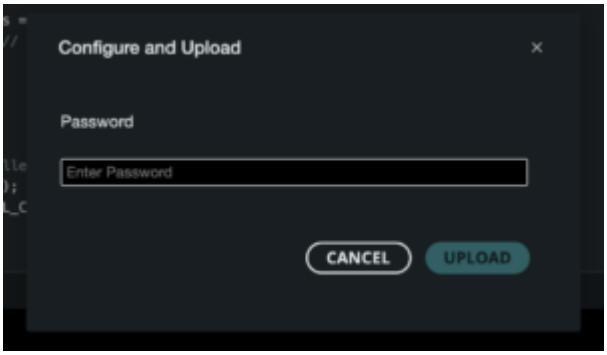
Nun folgt der Upload des aktualisierten Sketchs. Sofern dein ESP32 mit deinem WLAN-Netzwerk verbunden ist, sollte er in der Arduino-IDE als Netzwerkport sichtbar sein. Gehe hierfür zu **Werkzeuge > Port** und wähle den ESP32 (esp32_led_ota) aus.



OTA-Update hochladen: Lade den neuen Sketch (mit 500 ms Blinkintervall) über den Netzwerkport hoch. Klicke hierfür einfach wie gewohnt auf den Upload-Button – so wie du es auch

machst, wenn dein ESP32 über ein USB-Kabel verbunden ist. **Achte darauf, dass in deinem Update wieder die WLAN-Zugangsdaten und dein Passwort hinterlegt sind.**

Die Arduino-IDE fordert dich auf, das OTA-Passwort einzugeben. Gib das definierte Passwort („Mein_OTA_Passwort“) ein und das Update wird drahtlos übertragen.



Hinweis: Solltest du Updates mit verschiedenen Passwörtern machen, kann es sein, dass die Arduino IDE ein falsches Passwort automatisch verwenden möchte. Schließe in diesem Fall die IDE und öffne sie erneut – dann wirst du wieder nach dem Passwort für deinen ESP32 gefragt.

Nachdem das Update drahtlos auf deinen ESP32 übertragen wurde, sollte die LED nun im neuen Halbsekundentakt aufleuchten. Und das war es auch schon – du kennst nun eine Möglichkeit, Programme zu aktualisieren, ohne den Microcontroller irgendwo ausbauen und an deinen Computer anschließen zu müssen. Weitere Informationen zum Thema findest du bei Espressif.