ONSITE XMAS: PROGRAMMING GUIDE

T-Systems on site Services GmbH

Version: 1.0

Onsite Xmas

T · · Systems ·

Inhaltsverzeichnis

HARDWARE KOMPONENTEN

ESP 32

Der ESP32 ist eine kostengünstige und mit geringem Leistungsbedarf ausgeführte 32-Bit-Mikrocontrollerfamilie der Firma espressif. Er kommt standardmäßig mit Wifi und BLE an Bord.

Power supply Modul

Wird benötigt, um die 9 Volt des Akkus auf konstante 5 Volt zu bringen, so kommt es nicht zu Verbindungsabbrüchen des BLE Chips. Der Spannungswandler verfügt über einen USB Output, wodurch eine einfache Stromversorgung des ESP32 ermöglicht wird.

LED Stripe

WS2812b LED Stripe: Bei den WS2812 LEDs handelt es sich um adressierbare RGB-LEDs. Sie verfügen über einen integrierten Chip und belegen daher nur einen einzigen digitalen Output

1 SETUP

1.1 Ein Ausflug in die Bibliotheken

Um den Programmcode für unsere Christbaumkugel zu schreiben werden einige externe Bibliotheken benötigt, welche mittels „include“ eingebunden werden müssen.

Folgende Bibliotheken sind bereits durch Arduino selbst oder durch das „esp32“-Paket mitgeliefert:

- **ESP_WiFi:** Stellt allgemeine Funktionen für WiFi-Verbindungen zu Verfügung. Wir benötigen sie aber nur um WiFi abzuschalten, damit der Stromverbrauch reduziert wird.
- **BLEDevice:** Klasse für ein BLE Device. Die Grundlage für die BLE Kommunikation.
- **BLEServer:** Klasse für einen BLE Server. Hierauf kann sich später das Smartphone verbinden.
- **BLEUtils:** Allgemeine Funktionen für BLE.

Um die LED-Stripes anzusteuern wird zusätzlich die „NeoPixel“-Bibliothek von Adafruit benötigt. Eigentlich ist diese Bibliothek dafür gedacht einen LED-Ring von Adafruit anzusteuern, aber sie funktioniert auch wunderbar mit unseren LED-Stripes.

Um sie zu installieren muss diese zunächst aus den Git-Repository von Adafruit heruntergeladen werden: https://github.com/adafruit/Adafruit_NeoPixel. Das ZIP-Paket kann nun an einem beliebigen Ort auf dem Computer ablegen werden. Als nächstes muss das Paket in die Arduino IDE eingebunden werden. Dies funktioniert wie folgt:
„Sketch -> Bibliothek einbinden -> ZIP-Bibliothek hinzufügen…“ anschließend das heruntergeladene Paket auswählen.

Das Paket liefert zusätzlich zur Bibliothek einige Beispiele, die in der IDE unter „Datei -> Beispiele -> Adafruit NeoPixel“ zu finden sind.

1.2 Genau die richtige Debugging App – nRFConnect

Um die Bluetooth-Verbindung von Smartphone aus zu testen kann die App nRFConnect genutzt werden. Die App ist für Android sowie iOS kostenlos verfügbar:

- https://play.google.com/store/apps/details?id=no.nordicsemi.android.mcp&hl=en_US
- https://apps.apple.com/us/app/nrf-connect/id1054362403

1.3 Arduino - Was sonst?

Um das Board über euren Computer zu programmiere, müsst ihr es per Mikro-USB anschließen. Das Board wird über einen virtuellen COM Port angesprochen. Hierzu benötigt der Computer noch die passenden Treiber. Diese sind hier zu finden:

https://www.silabs.com/products/development-tools/software/usb-to-uart-bridge-vcp-drivers

Anschließend kann man sich die Arduino IDE unter https://www.arduino.cc/ für alle gängigen Betriebssysteme downloaden und installieren. Zunächst ist es wichtig, alle benötigten Boardkonfigurationen und Bibliotheken einzubinden, um den ESP32 über die IDE nutzen zu können. Dazu geht man unter „Datei -> Voreinstellungen“ und fügt eine „zusätzliche Boardverwalter-URL“ ein: *https://dl.espressif.com/dl/package_esp32_index.json*

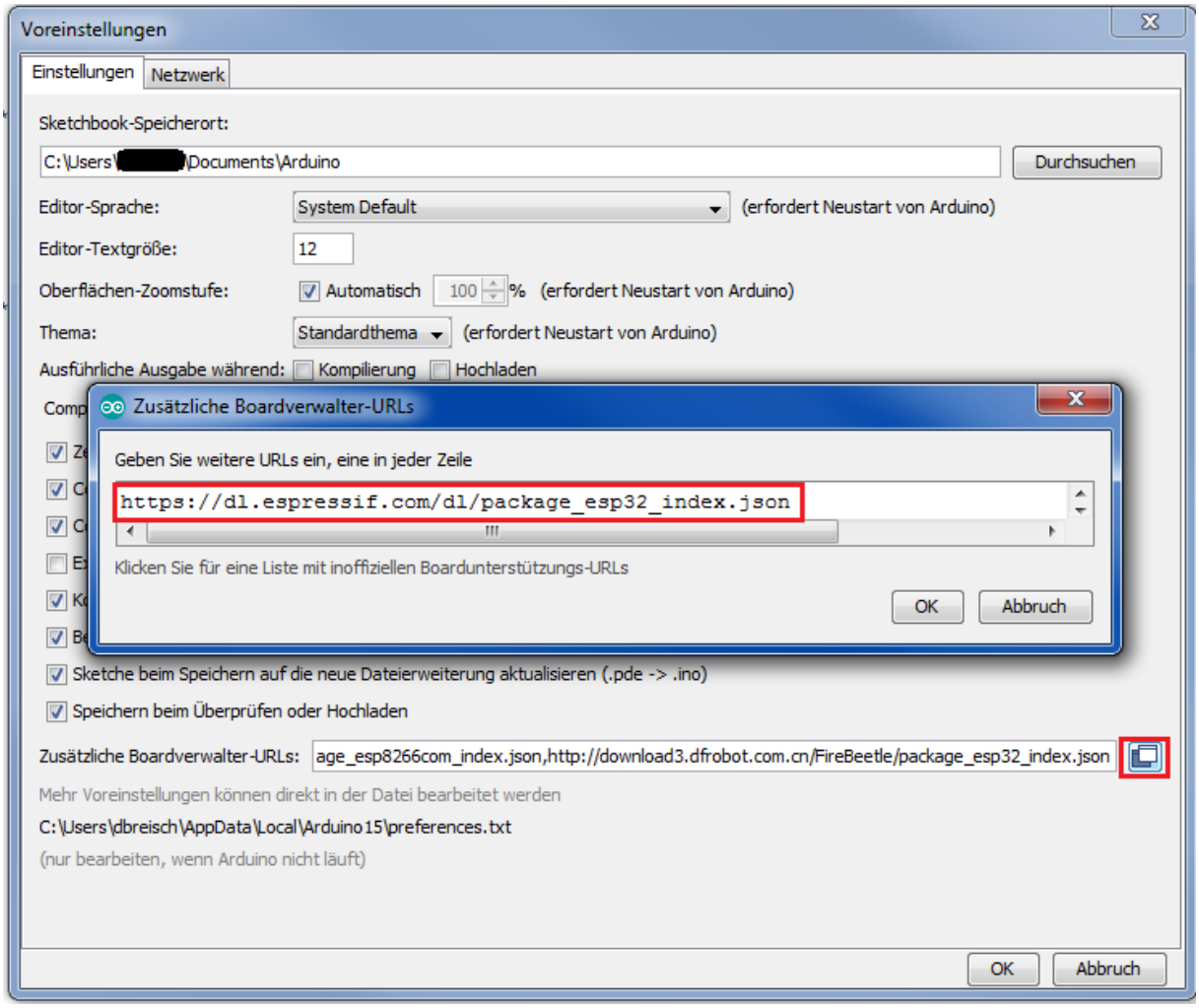

Abbildung 1: Zusätzliche Boardverwalter hinzufügen

Nun kann unter „Werkzeuge -> Board -> Boardverwalter“ das „esp32“ Paket installiert werden.

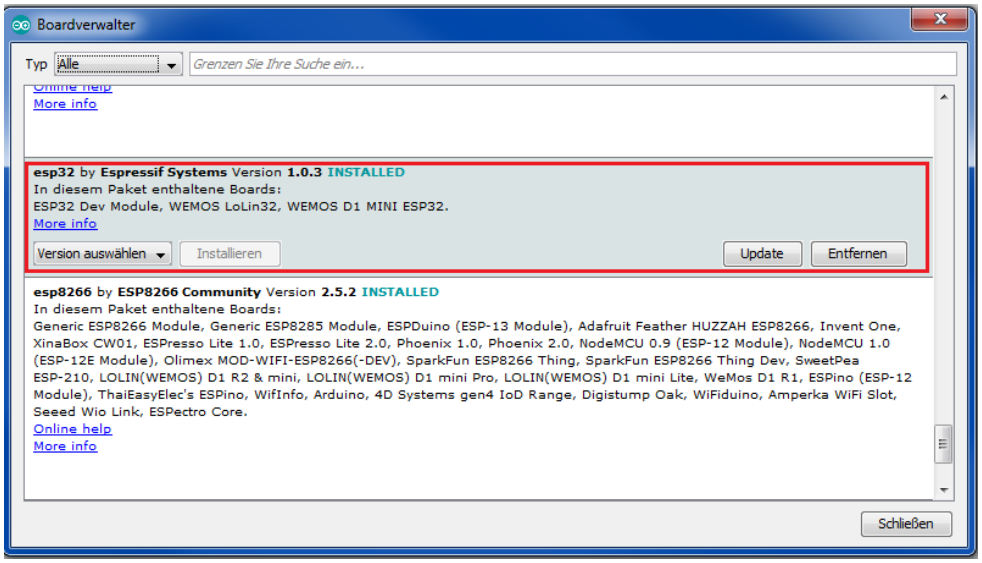

Abbildung 2: esp32 Paket installieren

Damit nun die IDE weiß für welches Board tatsächlich entwickelt wird, gibt man abschließend unter „Werkzeuge“ die korrekte Konfiguration wie in Abbildung 3 dargestellt an. Wichtig hierbei ist zudem das Auswählen des richtigen virtuelle COM Ports, an dem das Board angeschlossen ist.

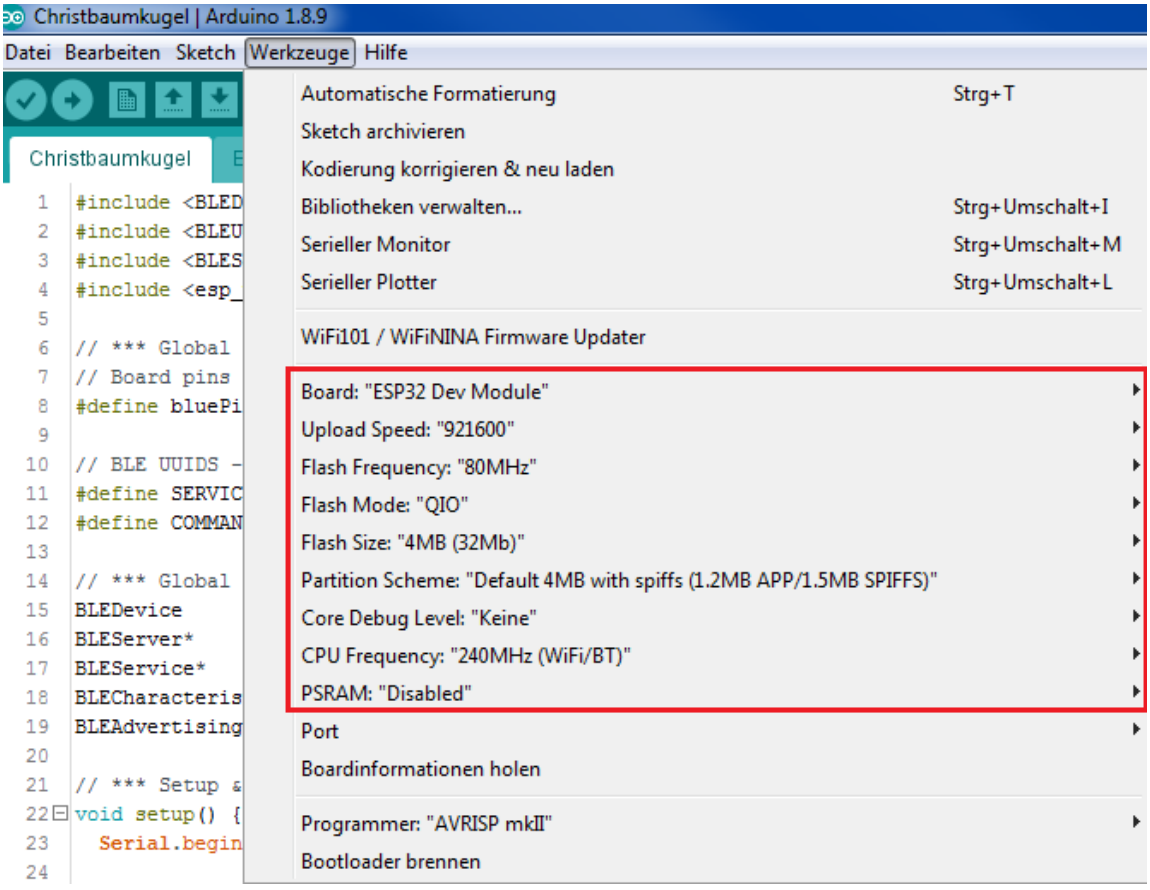

Abbildung 3: Boardkonfiguration

Will man den Sketch auf das Board flashen, klickt man auf den Pfeil „Hochladen“ (oben links in der IDE). Daraufhin wird der Sketch kompiliert und das Flashen beginnt. Will man lediglich kompilieren, klickt man auf den Haken „Überprüfen“.

Während des Flashvorgangs wird am unteren Bildschirm der aktuelle Vorgang in der Konsole ausgegeben. Sobald dort „Connecting…“ steht, muss in einigen Fällen der „Boot“ Knopf des Boards gedrückt werden, bis die Übertragung startet. Danach kann der Knopf losgelassen werden.

Abbildung 4: Konsolenausgabe beim Flashen

Eine praktische Debugmöglichkeit bietet der „Serielle Monitor“, der in der rechten oberen Ecke zu finden ist. Hier werden alle seriellen Ausgaben des Controllers angezeigt, die der Entwickler erstellt hat.

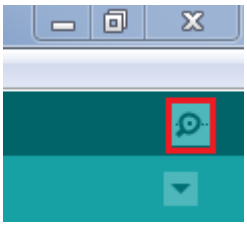

Abbildung 5: Button "Serieller Monitor"

Für Linux Nutzer:

Unter Linux muss zunächst die Versionskontrollsoftware Git und das Python-Modul Serial mittels

```
sudo apt install git python-serial
```

installiert werden.

Damit nun die IDE weiß für welches Board tatsächlich entwickelt wird, gibt man abschließend unter „Werkzeuge“ die korrekte Konfiguration wie in Abbildung 3 dargestellt an. Wichtig hierbei ist zudem das Auswählen des richtigen virtuelle COM Ports, an dem das Board angeschlossen ist.

1.4 Sag Hallo, zur Welt!

Zum Testen der Konfiguration kann ein kleiner „Hello-World“ Sketch implementiert werden. Hierzu muss über „Datei -> Neu“ ein neuer Sketch angelegt werden. Mithilfe des in Abbildung 6 dargestelltem Sketch kann eine Serielle Ausgabe geschrieben werden, welche im Seriellen Monitor verfolgt werden kann. Dabei ist darauf zu achten, dass die Baudrate im Seriellen Monitor mit der im Sketch hinterlegten Baudrate (115200) übereinstimmt.

```
void setup() {
 Serial.begin(115200);
 Serial.println("Hello ESP32 World!");
}

void loop() {
Serial.println("Hello");
delay(500);
}
```

Abbildung 6 Hello World Sketch

2 PROGRAMMING

Der Base Sketch beinhalt bereits das Bluetooth Profil, welches für die erfolgreiche Kommunikation mit der Onsite Xmas App benötigt wird. Dieses besteht aus einem Service und drei Charakteristiken. Ein Kernkonzept von BLE ist der Begriff der Charakteristik. Diese kann als zustandsbehafteten Datensatz betrachtet werden, der eine Identität und einen Wert hält. Ein BLE Client kann den Wert der Charakteristik lesen oder einen neuen setzen (falls zulässig). Der Wert der Charakteristik liefert demnach die Informationen. Wenn die Charakteristik z.B. die Herzfrequenz repräsentiert, gemessen von einem Sensor, dann kann ein entfernter Client die aktuelle Herzfrequenz abrufen, indem er den aktuellen Wert der Charakteristik liest. In diesem Fall wird ein Remote-Client den Wert nicht neu schreiben. Der Wert wird ausschließlich intern vom BLE-Server geändert, entweder jedes Mal, wenn eine Leseanforderung von einem Client gestellt wird oder wenn ein neuer Wert vom Sensor gemessen wird.

Um nun auf einen solchen Zugriff der Charakteristik reagieren zu können muss jeweils eine entsprechende Callback Methode registriert werden. Die Klasse namens BLECharacteristicCallbacks bietet hierfür zwei Methoden, die überschrieben werden können:

onRead(BLECharacteristic* pCharacteristic) - Wird aufgerufen, wenn eine Leseanforderung von einem Client eingeht. Ein neuer Wert für das Merkmal kann vor der Rückkehr aus der Funktion gesetzt werden und wird als der vom Client empfangene Wert verwendet.

onWrite(BLECharacteristic* pCharacteristic) - Wird aufgerufen, wenn eine Schreibanfrage eines Clients eingeht. Der neue Wert wurde bereits im Merkmal gesetzt und Ihre eigene Codelogik kann den Wert lesen und eine Aktion durchführen.

Diese Callback Methoden sind bereits im Sketch definiert. Hier muss demnach nun das von euch angepasste Verhalten implementiert werden 😉

Verbindungstest mit dem Smartphone durchführen

Um die BLE Konfiguration zu testen, kann nun eine Verbindung zum Erstellen BLE Gerät hergestellt werden. Der definierte Name sollte Dazu nach dem Scannen in der Liste der gefundenen Geräte sichtbar werden. Nach dem erfolgreichen Verbindungsaufbau ist das definierte Profil nun auf der Oberfläche sichtbar.

Durch einen Klick auf den definierten Service, können die gekapselten Charakteristiken ausgeklappt werden. Um sicherzustellen, dass alles funktioniert kann der definierte default-Wert der Charakteristik ausgelesen werden.

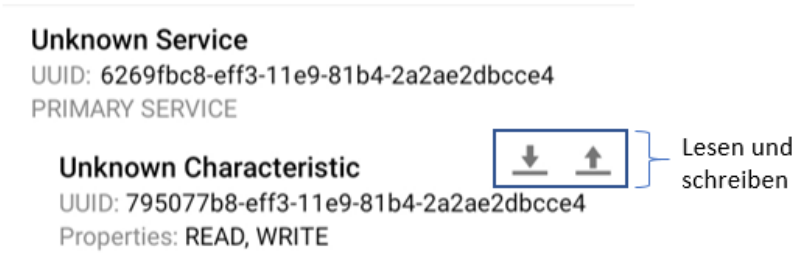

Abbildung 7 Lesen und Schreiben einer Charakteristik mithilfe der App nRF Connect

LED-Stripe initialisieren und verschiedene Lichtspiele programmieren

Zunächst muss mit folgender Zeile eine Neo-Pixel Objekt initialisiert werden. Dabei müssen mehrere Parameter definiert werden:

- Die Anzahl der verwendeten LEDs auf dem Stripe. Diese können im folgendem über einen Index angesprochen werden.
- Die Pin-Nummer an dem der LED-Stripe am Mikrocontroller angeschlossen ist.
- Ein Wert, der den Typ der angeschlossenen NeoPixel angibt

```
Adafruit_NeoPixel stripe(LED_COUNT, LED_PIN, NEO_GRB + NEO_KHZ800);
```

Anschließend kann in der setup() Funktion begin() aufgerufen werden, um den Datenpin für die NeoPixel-Ausgabe vorzubereiten.

```
stripe.begin();
```

Optional kann hier auch die Helligkeit der LEDs definiert werden. Dies kann allerdings auch für jedes Lichtmuster separat definiert werden. Die Helligkeit wird wie folgt konfiguriert:

```
stripe.setBrightness(BRIGHTNESS);
```

Durch folgenden Code kann nun die Farbe einer LED an einem spezifischen index definiert werden. Durch show() wird die definierte Pixelfarbe an den Stripe gesendet, sodass sich diese aktualisiert.

```
stripe.setPixelColor(index, color);
stripe.show();
```

Durch das separate Ansprechen jeder einzelnen LED können nun beliebige Lichtspiele entworfen werden.

Tipp: Lasst euch durch die mitgelieferten Beispiele der Bibliothek inspirieren und seid kreativ.

A.1 LED-Stripe: Nützliche Funktionen

Adafruit_NeoPixel stripe(LED_ANZAHL, LED_PIN, NEO_GRB + NEO_KHZ800);	Erstellt das Objekt “stripe” vom Typ “Adafruit_NeoPixel”. Parameter: - LED_ANZAHL: Anzahl der LEDs auf dem Stripe. - LED_PIN: Date input Pin am Board. - NEO_GRB + NEO_KHZ800: Konstanter Wert.
stripe.begin();	Startet den LED-Stripe.
stripe.show();	Änderungen am LED-Stripe übernehmen. Z.B. bei neuer Farbe.
stripe.setBrightness(int HELLIGKEIT);	Setzt die Helligkeit der LEDs. Wert von 0 – 255.
stripe.Color(RED, GREEN, BLUE);	Erstellt Farbcode anhand von RGB-Code. Gibt den Farbcode als uint32_t zurück.
stripe.setPixelColor(int POSITION_LED, uint32_t FARBE);	Setzt LED an POSITION_LED auf Farbe FARBE.
stripe.fill(uint32_t FARBE, VonLED, BisLED);	Setzt mehrere LEDs auf Farbe FARBE
stripe.clear();	Schaltet alle LEDs aus. Funktioniert nur in Verbindung mit stripe.show().