

Das Arduino Nano ESP32-Board entdecken

BLE und ESP-NOW anwenden, die Bus-Systeme verstehen,
MicroPython lernen

Erik Bartmann


bombini
verlag

Die Informationen in diesem Buch wurden mit größter Sorgfalt erarbeitet. Dennoch können Fehler nicht vollständig ausgeschlossen werden. Verlag, Autoren und Übersetzer übernehmen keine juristische Verantwortung oder irgendeine Haftung für eventuell verbliebene Fehler und deren Folgen.

Alle Warennamen werden ohne Gewährleistung der freien Verwendbarkeit benutzt und sind möglicherweise eingetragene Warenzeichen. Der Verlag richtet sich im wesentlichen nach den Schreibweisen der Hersteller. Das Werk einschließlich aller seiner Teile ist urheberrechtlich geschützt. Alle Rechte vorbehalten einschließlich der Vervielfältigung, Übersetzung, Mikroverfilmung sowie Einspeicherung und Verarbeitung in elektronischen Systemen.

Kommentare und Fragen können Sie gerne an uns richten:

Bombini Verlags GmbH
Richard-Wagner-Str. 11
53757 Sankt Augustin

E-Mail: service@bombini-verlag.de

Copyright:

© 2024 by Bombini Verlag

Bibliografische Information Der Deutschen Nationalbibliothek

Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über <http://dnb.d-nb.de> abrufbar.

Satz: Bombini Verlag

Belichtung, Druck und buchbinderische Verarbeitung: Print Group Sp. z.o.o., Polen

ISBN 978-3-946496-36-6

Inhalt

Einleitung	7
Das Arduino-Nano-ESP32-Board	9
Ein knapper Vergleich beider Boards	9
Die Spezifikationen	11
Die Speicherbereiche - Flash und SRAM	12
Die Pin-Belegung	12
Die Ports	14
Der Bootloader	15
Die Arduino-Entwicklungsumgebung öffnen	18
Die Spannungsversorgung	19
C++ und MicroPython	21
Arbeiten mit der Arduino-IDE 2	23
Die Arduino-IDE	23
Ein erstes Arduino-Projekt unter Arduino-IDE 2 erstellen	25
Arduino-Pins und GPIO-Pins	28
Programmieren mit der Arduino IDE 2	31
Die Analyse des Blink-Sketches	32
Kommentare	32
Variablen	34
Funktionen	35
Die Datentypen	41
Die Arduino-IDE unterstützt dich	42
Projekt 1: Hello World	47
Die RGB-LED	47

Projekt 2: Die digitalen Pins.....	51
Die digitalRead- und digitalWrite-Funktion	51
PWM-Signale	58
PWM-Steuerung ohne Timer	63
Projekt 3: Die analogen Pins	65
Die analogRead- und analogWrite-Funktion	66
Der Spannungsteiler	67
Projekt 4: Der störrische Taster.....	77
Das Entprellen (Debouncing) - Variante 1	81
Das Entprellen (Debouncing) - Variante 2	81
Das Entprellen (Debouncing) - Variante 3	83
Das Entprellen (Debouncing) - Variante 4	84
Projekt 5: Bus-Systeme.....	87
Die serielle Schnittstelle	87
Der SPI-Bus - noch eine serielle Schnittstelle.....	120
Der I2C-Bus.....	132
Der MCP23017-Schaltplan - Eingänge und Ausgänge	143
Der I3C-Bus.....	151
Projekt 6: ESP32-NOW.....	153
Die Ein-Weg-Kommunikation.....	154
Die Ein-Weg-Fernsteuerung	166
Die Zwei-Wege-Kommunikation.....	179
Weitere Wege der Kommunikation	190
Projekt 7: Human Interface Device	191
Die Computermaus über HID	192
Die Computertastatur über HID	198

Spaß mit der HID-Maus	203
Projekt 8: Synthesizer-Steuerung über BLE	213
Der Arduino-Sketch	214
Kommen wir zur Android-Seite	218
Projekt 9: Node-RED.....	221
Die Installation von Node-RED	222
Der Aufruf von Node-RED	223
Die Installation einer Erweiterung über die Palette.....	224
Ein erster serieller Test	225
Der Arduino Talker in Node-RED.....	229
Das Blinken der LED.....	232
Projekt 10: Temperaturmessung über WiFi.....	237
Der Aufbau einer WiFi-Verbindung.....	237
Die Temperaturmessung	240
Der ThingSpeak-Server	245
Projekt 11: Messwerte visualisieren	255
Die Vorbereitungen	256
Ein erster Test	258
Die Darstellung eines Messwerts	261
MicroPython-Workshop: Installationen	267
Die Installation der MicroPython-Firmware.....	268
Die Arduino-MicroPython-Entwicklungsumgebung	270
Die Thonny-MicroPython-Entwicklungsumgebung.....	273
Die Installation des Arduino Bootloaders.....	275
MicroPython-Workshop: Grundlagen.....	281
Die Groß- und Kleinschreibung.....	281

Die Variablen	282
Das Casting	283
Zeichenketten und Arrays.....	285
Die Kommentare	286
Die Schleifen.....	287
Die Operatoren	288
MicroPython-Workshop: Die Nano-Board-Programmierung ...	291
Die Ansteuerung eines I/O-Pins	292
Die LED soll blinken.....	296
MicroPython-Skripte	299
MicroPython-Workshop: Pins ansteuern	309
Die Abfrage eines Tasters	309
Einen Status vorgeben.....	311
Eine Zeitverzögerung	312
Ein Lauflicht.....	312
Ein einfaches Roulettespiel	315
Das verbesserte Roulettespiel	317
MicroPython-Workshop: WLAN und OLED	325
Die Paketverwaltung unter Thonny	325
Der erste Test des SSD1306-OLED	326
Das WiFi aktivieren	328
Die aktuelle Zeit ermitteln.....	333
Die Anzeige der Zeit auf dem OLED	335
Stichwortverzeichnis	339

Projekt 8: Synthesizer-Steuerung über BLE

Wer Spaß an elektronischer Musik und Sphärenklängen hat, der kennt auch die elektronischen Musikinstrumente, die sich Synthesizer nennen. Auf der folgenden Abbildung ist ein Synthesizer mit integriertem Keyboard und vielen Drehreglern zu sehen.



Abbildung 1: Der Synthesizer MatrixBrute von Arturia

Dies soll keine Einführung in den Umgang mit professionellen Synthesizern werden, sondern das Interesse wecken, sich mit derartigen Instrumenten und Klängen näher zu befassen. Viel ist für den Anfang nicht erforderlich, denn wir wollen in diesem Kapitel die MIDI-Steuerung über das Arduino-Nano-ESP32-Board realisieren.

Was ist MIDI?

MIDI ist die Abkürzung für Musical Instrument Digital Interface und es handelt sich dabei um einen Industriestandard für den Austausch von musikalischen Steuerinformationen zwischen elektronischen Instrumenten, wie zum Beispiel Keyboards oder Synthesizern. Es werden von MIDI keinerlei Audio-Daten versendet oder verarbeitet. Eine MIDI-Information enthält nicht direkt die Musik, wie sie zum Beispiel auf einer CD oder einer MP3-Datei enthalten ist, sondern nur Steueranweisungen, was zu tun ist.

Ich möchte einen freien Synthesizer in Form einer Android-App mit dem Namen SynprezFM rudimentär ansteuern und dafür das Arduino-Nano-ESP32-Board verwenden.



Abbildung 2: Die Synthesizer-App SynprezFM

Die Steuerung wird über BLE erfolgen.

Was ist BLE?

BLE ist die Abkürzung für Bluetooth Low Energy und ist eine Funktechnik, mit der sich Geräte in einer Umgebung von etwa zehn Metern vernetzen lassen.

Ich möchte zur Realisierung kein komplettes Keyboard nachbauen, sondern lediglich einen einzigen Taster mit zwei Potentiometern. Eine Taste ist zur Generierung des Tons erforderlich, das erste Potentiometer steuert die Tonhöhe und das zweite die Anschlaggeschwindigkeit. Recht simpel also, doch es steckt Potential in diesen Anfängen.

Und wer weiß, vielleicht regt den ein oder anderen dieses Kapitel auch dazu an, den Synthesizer auszubauen und noch ganz andere Dinge damit zu veranstalten.

Fangen wir mit dem Arduino-Sketch an, bevor es auf der Smartphone-Seite unter Android weitergeht.

Der Arduino-Sketch

Um die Funktionalität von MIDI über BLE nutzen zu können, ist es erforder-

lich, eine Library hinzuzufügen, die sich ESP32-BLE-MIDI nennt. Wir rufen also wieder den Library Manager auf und geben den markierten Suchbegriff ein.

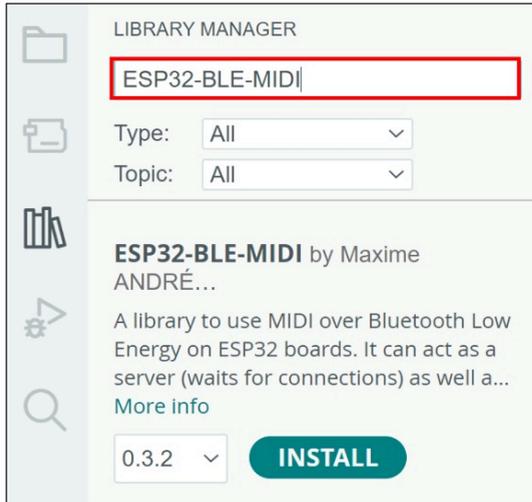


Abbildung 3: Die Installation der ESP32-BLE-MIDI-Library

Nähere Informationen zur Library sind unter dem folgenden Link zu finden.

<https://www.arduino.cc/reference/en/libraries/esp32-ble-midi/>

Nun können wir starten. In Zeile 1 wird die erforderliche BLE-MIDI-Library eingebunden, die gerade eben installiert wurde. In Zeile 3 wird der Pin definiert, an dem sich der Taster befindet. In den Zeilen 4 und 5 werden Variablen definiert, um den Zustand des Tasters zu überwachen, damit sowohl ein LOW-HIGH- als auch ein HIGH-LOW-Pegelwechsel erkannt werden kann. Die zu spielende Note wird als Integer-Wert in der Variablen `midiNote` gespeichert, die in Zeile 6 deklariert ist. Die Anschlagstärke wird in der Variablen `velocity` (Geschwindigkeit) gespeichert und in Zeile 7 deklariert.

```

1  #include <BLEMidi.h>
2
3  int buttonPin    = 2; // Button-Pin-Note
4  int btnState     = 0; // Actual button state
5  int lastBtnState = 0; // Last button state
6  int midiNote;    // MIDI-Note
7  int velocity;    // MIDI-Velocity

```

In der `setup`-Funktion in Zeile 12 wird der Taster über die `pinMode`-Funktion

```

9   void setup() {
10      Serial.begin(115200);
11      Serial.println("Initializing bluetooth");
12      pinMode(buttonPin, INPUT_PULLDOWN);
13      BLEMidiServer.begin("Eriks-ESP32-MIDI-Device");
14      Serial.println("Waiting for connections...");
15  }

```

konfiguriert und in der darauffolgenden Zeile 13 der BLEMidiServer über die begin-Methode gestartet. Über den angegebenen Namen kann das BLE-Device später auf dem Smartphone auffindig gemacht werden.

In der loop-Funktion kommt es nun in Zeile 18 zur Abfrage, ob der Server verbunden wurde, was über die isConnected-Methode erreicht wird. Bei erfolgreicher Verbindungsaufnahme kommt es zur kontinuierlichen Abfrage des Tasters (Zeile 19) und der beiden Potentiometer (Zeilen 20 und 22). Für die analogen Werte findet wieder ein Mapping über die map-Funktion statt, die wir schon kennengelernt haben. Findet ein Pegelwechsel des Tasters von LOW nach HIGH statt, kommt es zum Aufruf der noteOn-Methode in Zeile 23, mit der Angabe des MIDI-Kanals 0, der MIDI-Note (midiNote) und der Anschlagstärke (velocity). Die Note wird hörbar. Wird die Taste wieder losgelassen, erfolgt ein Pegelwechsel von HIGH nach LOW und die noteOff-Methode in Zeile 26 wird mit den gleichen Werten wie bei noteOn aufgerufen. Die Note verklingt.

```

17  void loop() {
18      if(BLEMidiServer.isConnected()) {
19          btnState = digitalRead(buttonPin);
20          velocity = map(analogRead(A6), 0, 4095, 0, 127);
21          if(btnState == 1 && lastBtnState == 0){ // LOW-HIGH-Level
22              midiNote = map(analogRead(A7), 0, 4095, 10, 90);
23              BLEMidiServer.noteOn(0, midiNote, velocity); // Note on
24              Serial.println("On");
25          } else if(btnState == 0 && lastBtnState == 1){ // HIGH-LOW-Level
26              BLEMidiServer.noteOff(0, midiNote, velocity); // Note off
27              Serial.println("Off");
28          }
29          lastBtnState = btnState; // Save last button state
30      }
31  }

```

Sowohl das Drücken als auch das Loslassen einer Taste löst ein entsprechendes MIDI-Ereignis aus.

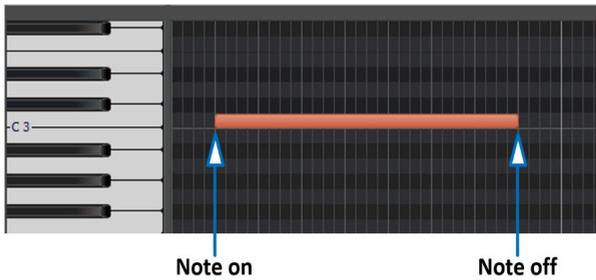


Abbildung 4: Note on und Note off in MIDI

Die Länge der Note wird durch die Länge des grafischen Balkens abgebildet und befindet sich genau zwischen den beiden MIDI-Befehlen Note on und Note off. Nun sind aber noch weitere Informationen erforderlich, denn es muss hinsichtlich der Note noch die Notennummer und die Velocity (Anschlagstärke) übermittelt werden. Hinsichtlich MIDI ist dann noch der Kanal wichtig, denn es gibt beim MIDI-Standard sechzehn verschiedene Kanäle. Eine Nachricht an ein MIDI-Gerät würde dann ungefähr wie folgt lauten: „Spiele auf MIDI-Kanal 0 die Note C3 mit der Länge, die zwischen Note on und Note off liegt sehr laut, also mit einem hohen velocity-Wert.“ Der velocity-Wert kann dabei im Bereich von 0 bis 127 liegen und deckt somit 128 unterschiedliche Werte ab.

Sehen wir uns sowohl den Schaltplan als auch den Schaltungsaufbau an.

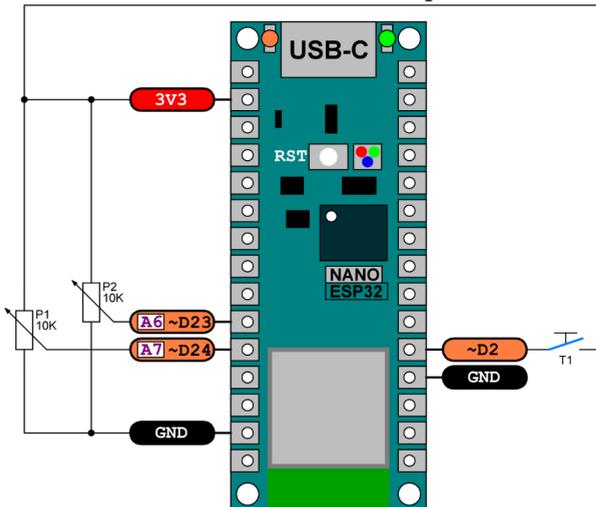


Abbildung 5: Der Schaltplan zur Synth-Ansteuerung

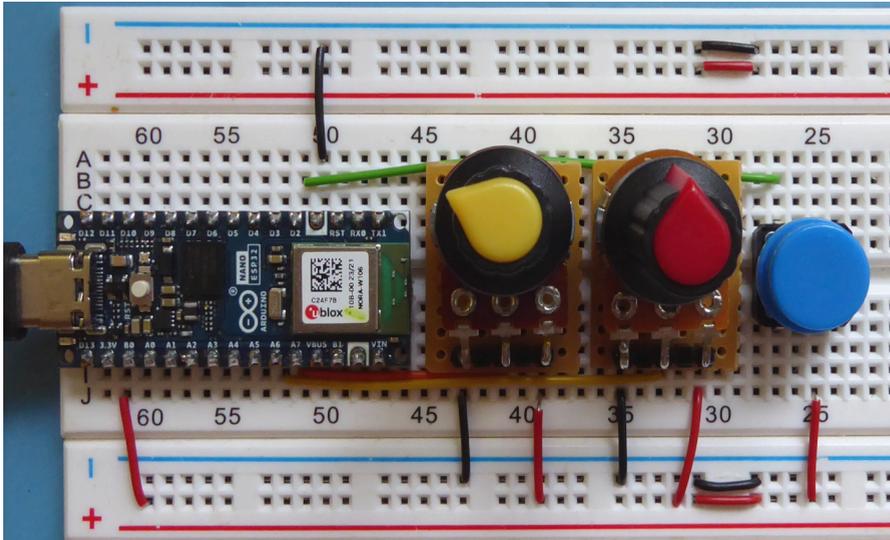


Abbildung 6: Der Schaltungsaufbau zur Synth-Ansteuerung

Jetzt laden wir den Sketch hoch und wenden uns dem Smartphone zu.

Kommen wir zur Android-Seite

Um auf der Android-Seite mit BLE zu arbeiten, muss eine entsprechende App mit Namen MIDI BLE Connect installiert werden. Dazu wird Google Playstore aufgerufen und in das Suchfeld der entsprechende Begriff eingegeben.

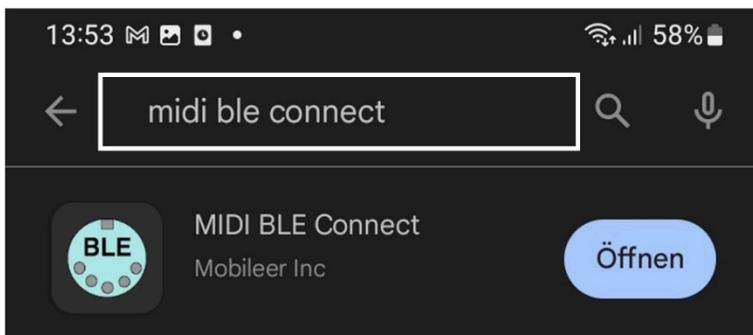


Abbildung 7: Die Installation von MIDI-BLE-Connect

Im Anschluss installieren wir den Synthesizer SynprezFM über das gleiche Verfahren.

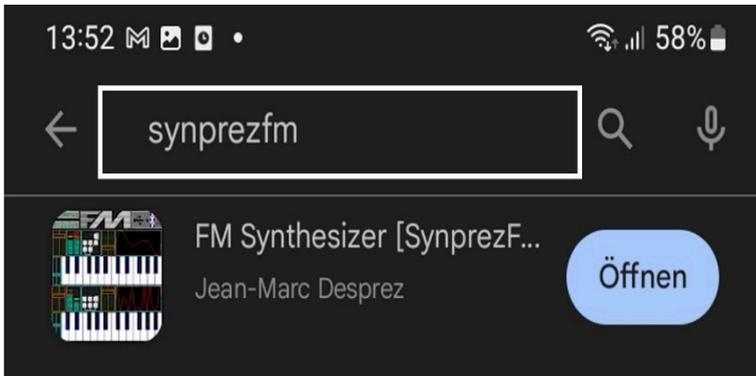


Abbildung 8: Die Installation von SynprezFM

Nach den beiden Installationen starten wir MIDI BLE Connect und stellen vorher sicher, dass auf dem Smartphone Bluetooth aktiviert ist. Nach der Auswahl der START BLUETOOTH SCAN-Schaltfläche sollte das ESP32-BLE-Device zur Auswahl angezeigt werden. Durch die Auswahl mit einem Fingertipp wird es dann wie in der Abbildung dargestellt.

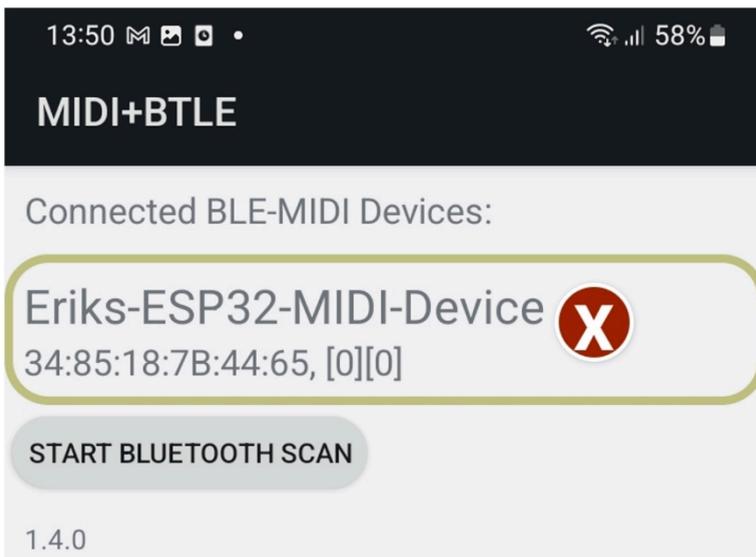


Abbildung 9: Das ESP32-BLE-Device wurde ausgewählt und aktiviert

Nun kann die SynprezFM-App gestartet werden, was mit dem unmittelbaren Erkennen des ESP32-BLE-Devices einhergehen sollte. Die Meldung sollte ungefähr so aussehen in der rot umrandeten Box auf der folgenden Abbildung.

Projekt 8: Synthesizer-Steuerung über BLE

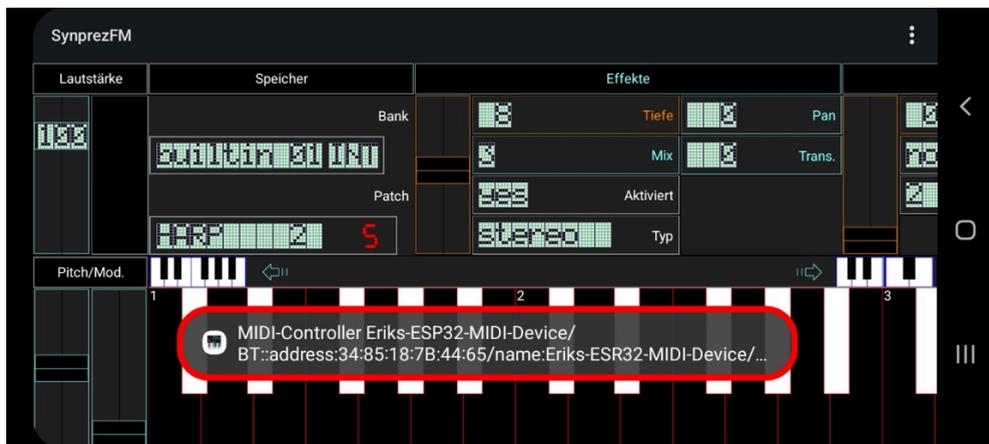


Abbildung 10: Der MIDI-Controller wurde erkannt

Nun kann es losgehen und auf der Arduino-Seite der Taster mit unterschiedlichen Potentiometerstellungen gedrückt werden. Die SynprezFM-App sollte reagieren und die gedrückten Tasten sollten farbig aufleuchten und Töne sollten hörbar sein.