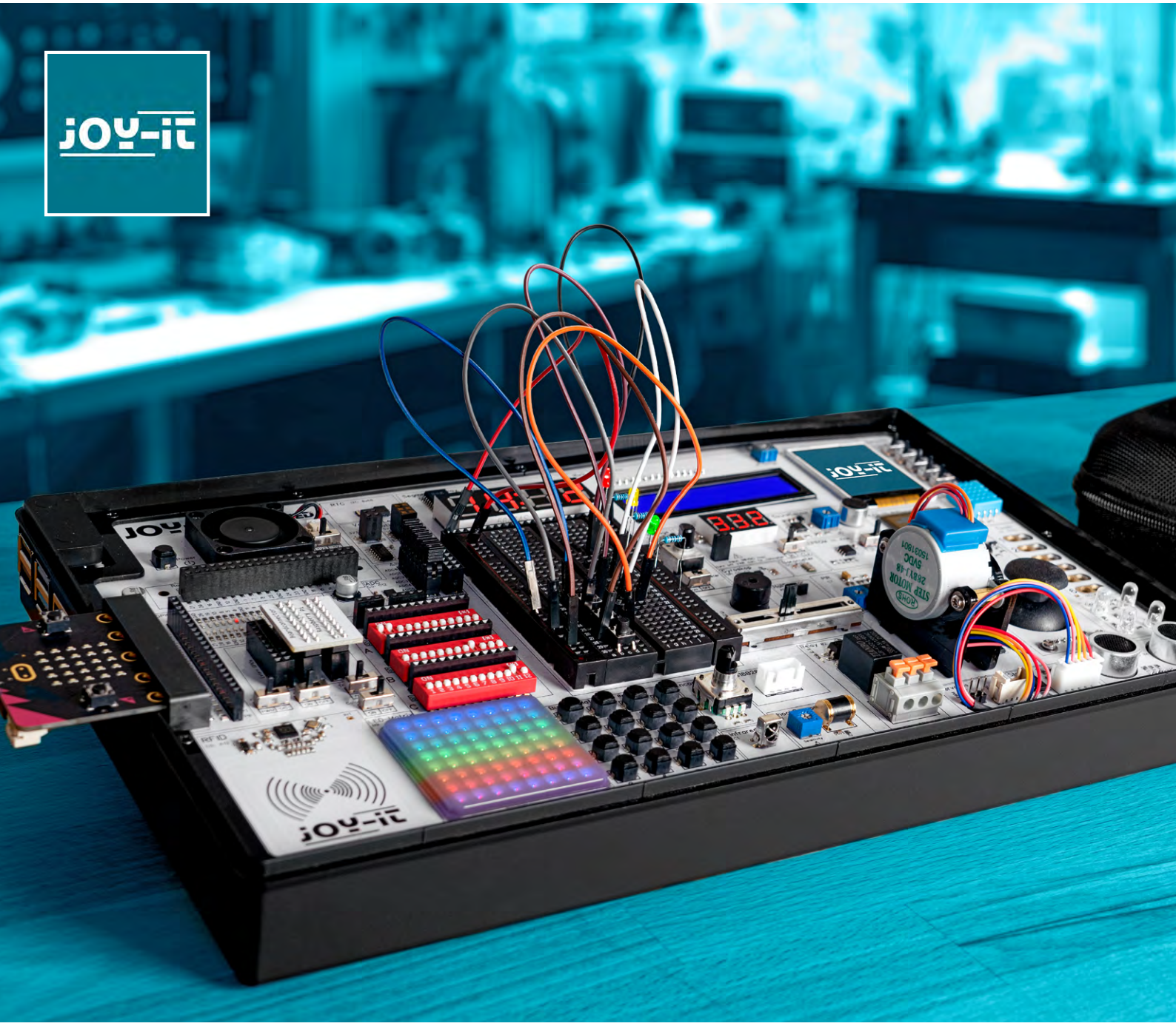
JOY-PI Advanced im Praxiseinsatz

Ein Erfahrungsbericht

von Dr. Peter Tschulik

Die Entwicklungsplattform und Lernzentrale JOY-PI Advanced verspricht, ein kompaktes und leistungsstarkes Gerät zu sein, mit dem sich Projekte schnell und einfach realisieren lassen. Sie spricht sowohl Anfänger als auch erfahrene Maker an und sollte ebenfalls gut im technischen Unterricht einsetzbar sein. Die Entwicklungsplattform ist kompatibel mit einer Vielzahl von Mikroprozessorboards wie dem Raspberry PI, Raspberry PI Pico, Arduino Nano, BBC micro:bit und NodeMCU ESP32 und bietet 39 integrierte Sensoren und Baugruppen. Der folgende Artikel beschreibt meine Erfahrungen mit dieser Entwicklungsplattform und gibt wertvolle Praxistipps für den Einstieg.



Joy-IT

Mit der Marke Joy-IT verbinde ich innovative Geräte zu einem fairen Preis, die genau auf die Bedürfnisse von Makern zugeschnitten sind. So besitze ich das vielseitige 3-in-1-Gerät DMSO2D72 aus Oszilloskop, Signalgenerator und Multimeter, das einen fixen Platz auf meinem Labortisch hat. Mit dem DPS5015-Modul habe ich mir ein Hochleistungs-Labornetzteil gebaut. Mit der Mini-Wireless-Tastatur mit integriertem Maus-Touchpad und RGB-Beleuchtung und dem tragbaren 15,6"-Touchscreen-Monitor steuere ich meine Medienzentrale basierend auf einem Raspberry PI und Kodi. Mit allen diesen Geräten bin ich sehr zufrieden. Da ich auch nebenberuflich auf einer Fachhochschule unterrichte und meinen SchülerInnen meine Freude an der Elektronik und Programmierung weitergeben möchte, habe ich mir die Entwicklungsplattform und Lernzentrale JOY-PI Advanced zusammen mit einem Raspberry 4 bestellt.

Unpacking

Innerhalb weniger Tage kam das Paket von ELV an. Die Entwicklungsplattform und Lernzentrale JOY-PI Advanced wird in einem schön bedruckten Karton geliefert. Öffnet man diesen, finden sich die überraschend kompakte Evaluierungsplattform in Form eines bedruckten schwarzen Kunststoffkoffers (L x B x H: 327 x 200 x 52 mm) und eine schöne Hartschalenbox, die das Zubehör und die Bedienungsanleitung enthält.

Beginnen wir mit der Hartschalenbox: Diese beinhaltet zuerst einmal drei Adapterboards für NodeMCU ESP32, Arduino Nano und Raspberry PI Pico. Für das BBC micro:bit gibt es aus gutem Grund kein eigenes Adapterboard, da die anderen Adapterboards den micro:bit Connector verwenden. Um eines der Adapterboards zu verwenden, wird das entsprechende Mikrocontrollerboard aufgesteckt und danach in der linken Seite der Entwicklungsplattform eingesteckt. Das BBC micro:bit kann direkt ohne Adapterboard eingesteckt werden (Bild 1).

Hoffentlich werden in Zukunft für neue Mikrocontrollerboards neue Adapterboards von Joy-IT zu erwerben sein.

Weitere Teile der Hartschalenbox sind zwei Boardconnectoren, einer für den Raspberry PI und einer für die externen Boards, wobei der Boardconnector für den Raspberry bereits angesteckt ist (Bild 2). Möchte man hingegen eines der Adapterboards oder das BBC micro:bit Board verwenden, so muss der „Boardconnector Extern" gesteckt sein (Bild 3).

Ein weiterer Beutel der Hartschalenbox beinhaltet folgende Bauteile für Experimente, die am Steckbrett aufgebaut werden können: je zehn Widerstände mit 100 Ω, 220 Ω, 330 Ω, 1 kΩ und 10 kΩ, je drei grüne, gelbe und rote LEDs, ein NPN-Transistor S8050, fünf Taster, ein Potentiometer mit 10 kΩ und drei Kondensatoren mit 100 nF.

Für den Einbau und den Betrieb des Raspberry PI befinden sich folgende Teile in der Hartschalenbox: ein Säckchen mit Schrauben, eine 32-GB-microSD-Karte, ein Micro-auf-SD-Kartenadapter, ein microSD-USB-Lesegerät und ein Kreuzschlitzschraubenzieher. Weitere beigefügte elektronische Komponenten sind eine Infrarot-Fernbedienung, ein RFID-Chip, eine RFID-Karte, ein wasserfester DS18B20-Temperatursensor mit Anschlusskabel, ein Schrittmotor inklusive Montagehalter und Zahnscheibe sowie ein Servo. Sechs Kabel mit Krokoklemmen, 15 verschiedenfarbige Steckkabel sowie ein Netzkabel runden die überkomplette Ausstattung ab.

Öffnet man den Deckel der Entwicklungsplattform, so ist man zuerst einmal von der Vielzahl der verbauten Sensoren, Aktoren und Komponenten überwältigt. Als Displays stehen ein 4-stelliges 7-Segment-Display, ein blaues LCD-Display mit 16 Zeichen und zwei Zeilen, ein 1,8"-TFT-Display, ein 0,96"-OLED-Display sowie eine 8x8-RGB-Matrix zur Verfügung. Als Sensoren stehen ein DS18B20-Temperatursensor, ein Schocksensor, ein Hallsensor, ein Barometer, ein Soundsensor, ein Gyroskop, ein PIR-Sensor, eine Lichtschranke, ein NTC, ein Lichtsensor, sechs Touchsensoren, ein Farbsensor, ein Ultraschall-Abstandssensor und ein DHT11-Temperatur- und Feuchtigkeitssensor zur Verfügung. Desweiteren stehen noch ein Joystick, fünf Schalter, ein Potentiometer, ein Dreh-Encoder, eine 4x4-Button-Matrix, ein Relais, ein PWM-Lüfter, eine Servo- und eine Schrittmotor-Schnittstelle, ein Vibrationsmotor, eine RTC-Echtzeituhr, ein Buzzer, ein EEPROM-Speicher, ein Infrarot-Empfänger, ein Steckbrett und ein RFID-Lesegerät zur Verfügung. Schließlich befinden sich auch noch Mess- und Wandelmodule wie Analog-digital-Konverter und Pegelwandler auf der Platine der Entwicklungsplattform.

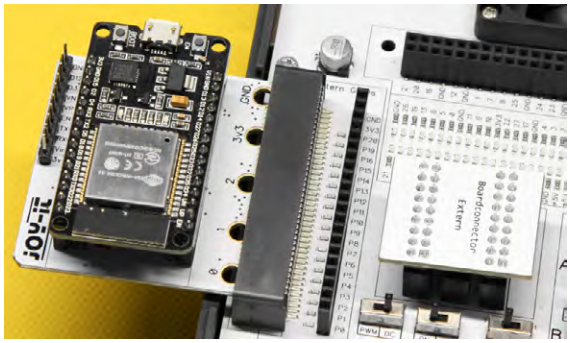


Bild 1: Adapterboard NodeMCU ESP32 angesteckt an der Entwicklungsplattform (Quelle: Joy-IT)

In das Gehäuse eingebaut ist außerdem ein 36-W-Netzteil, das die Spannungen 12 V, 5 V und 3,3 V sowie eine variable Spannung von 2 V bis 11 V mit bis zu 1 A zur Verfügung stellt. Leider habe ich keine Informationen gefunden, wie sich die 36 W auf die jeweiligen Spannungen aufteilt. Interessanterweise konnte ich bei der variablen Spannungsquelle entgegen der Bedienungsanleitung Spannungen bis hinunter auf 0,76 V einstellen. Ein integriertes Voltmeter kann interne und externe Spannungen bis 30 V messen und anzeigen. Bild 4 zeigt die Anzeige am Voltmeter bei Verwendung der variablen Spannungsquelle. Weitere Details sind in Kapitel 7 „Sonstige Funktionen" der Bedienungsanleitung beschrieben.

Bild 2: Vormontierter Boardconnector für den Raspberry Pi (Quelle: Joy-IT)

Bild 3: Aufgesteckter Boardconnector für Adapterboards und den BBC micro:bit (Quelle: Joy-IT)

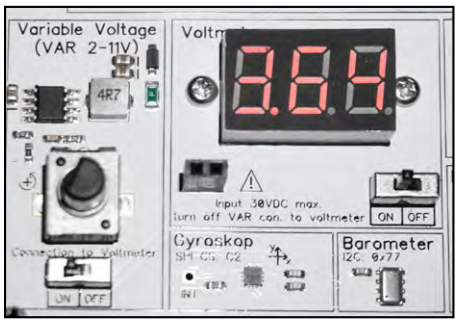


Bild 4: Variable Spannungsversorgung und Anzeige am Voltmeter

Da nicht alle Sensoren gleichzeitig ansteuerbar sind, lassen sich diese über drei DIP-Schalter mit jeweils zwölf Schaltern, die mit A, B und C beschriftet sind, zu- und abschalten. Dabei ist es sehr wichtig zu beachten, dass jeweils nur ein Schalter pro Kanalnummer auf ON steht, um Kurzschlüsse zu vermeiden. Wird beispielsweise der Servomotor auf Kanal A9 aktiviert (ON), so sollten die Schalter B9 und C9 stets deaktiviert (in der unteren Stellung) bleiben. Im Beispiel in Bild 5 sind die Lichtschranke (B1) und die vier Steuerleitungen für den Schrittmotor (B5–B8) aktiviert.

Die Funktion der Schalter ist sowohl in der Bedienungsanleitung als auch auf der Innenseite des Deckels der Evaluierungsplattform abgedruckt. Auf der Platine findet man praktischerweise bei jedem Sensor die Schalterstellungen bzw. eine etwaige I2C-Adresse aufgedruckt. Weitere Details sind in der Bedienungsanleitung in Kapitel 3 „Sensorik" beschrieben.

Der beigelegte Schrittmotor muss zuerst wie in der Bedienungsanleitung in Kapitel 7 beschrieben zusammengebaut und auf die Platine des Entwicklungsboards aufgesteckt werden.

Vier wichtige Schalter befinden sich unterhalb des Boardconnectors, wie Bild 6 zeigt.

Mit dem Schalter „Fan Control" wird der Lüfter des Raspberry PI gesteuert, in der Stellung PWM kann dieser vom Raspberry PI oder einem anderen Mikrocontroller gesteuert werden, in der Stellung DC läuft er dauerhaft.

Der Schalter „TFT Backlight" schaltet die Hintergrundbeleuchtung des TFT-Displays ein oder aus, und der Schalter „ColorSensor LEDs" schaltet die vier Beleuchtungs-LEDs für den Farbsensor.

Einen weiteren Schalter, beschriftet mit „GPIO LEDs" gibt es noch, der eine der genialsten Funktionen der Evaluierungsplattform aktiviert bzw. deaktiviert. Alle Ein- und Ausgänge des Raspberry PI als auch des micro:bit-Connectors zum Anschluss von NodeMCU ESP32,

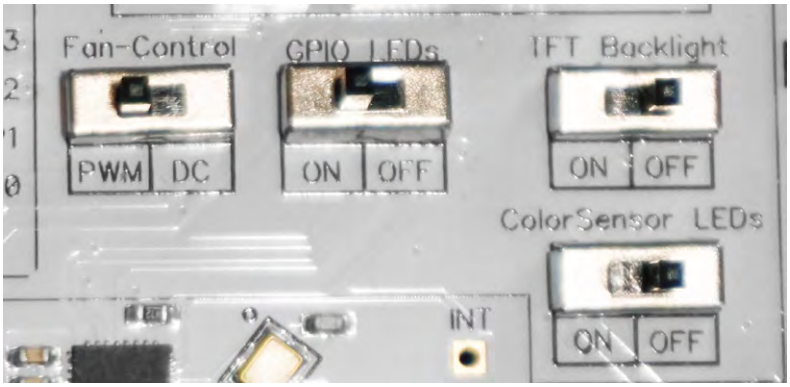


Bild 6: Wichtige Schalter

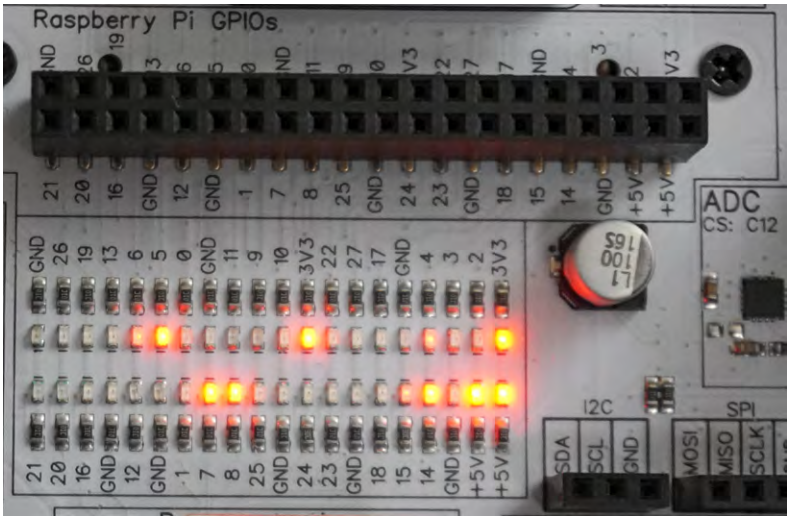


Bild 7a: LEDs und Buchsenleiste für den Raspberry PI

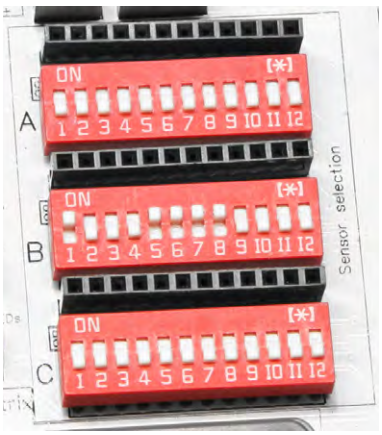


Bild 5: DIP-Schalter für die Auswahl der Sensoren (Quelle: Joy-IT)

Arduino Nano, Raspberry PI Pico und BBC micro:bit besitzen LEDs, die den aktuellen Logikpegel anzeigen. In vielen Fällen erspart man sich dadurch eine Fehlersuche mit einem Multimeter bzw. Logic-Analyser. Ebenso sind alle Ein- und Ausgänge über Buchsenleisten zugänglich (Bild 7a und 7b). Dieser Schalter aktiviert bzw. deaktiviert die entsprechenden LEDs.

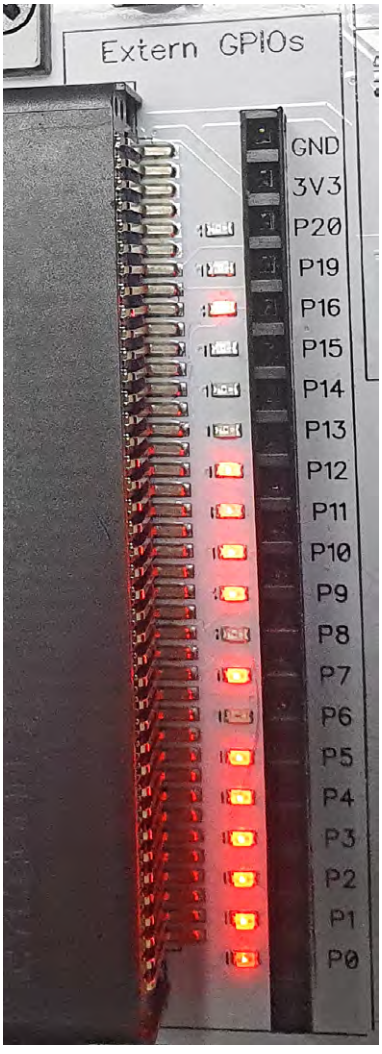


Bild 7b: LEDs und Buchsenleiste für die Anschlüsse an das Adapterboard

Erste Schritte

Voll Erwartung steckte ich die Evaluierungsplattform an den Strom an und – es passierte nichts. Keine LEDs oder Displays leuchteten. Erst nach dem Drücken des Power-Tasters links oben neben dem Lüfter leuchteten einige LEDs auf und der Raspberry PI-Lüfter begann sich zu drehen. Mit einem längeren Druck der Power-Taste (ca. 2 Sekunden) schaltet sich die Evaluierungsplattform wieder aus.

Bevor ich die Verwendung des Raspberry PI beschreibe, macht es Sinn, sich die Windows-Version der Entwicklungsumgebung downzuloaden und zu installieren – für IoS ist leider keine eigene Version verfügbar. Nach dem Start der Software zeigt sich der Startbildschirm wie in Bild 8 zu sehen.

Dort stehen unter „Sprachauswahl" Deutsch und Englisch zur Verfügung. Klickt man auf Hilfe, so kommt man in den Helpdesk-Bereich von Joy-IT für den JOY-PI, wo sich viele nützliche Tipps finden.

Im Kapitel „JOY-PI ADVANCED" finden sich die folgenden wichtigen Punkte:

Unter „Konfigurationen" finden sich die Bibliotheken, die für das Ansprechen der einzelnen Sensoren notwendig sind, sowie das Shutdown-Skript, das den Raspberry PI beim Druck auf die Power-Taste automatisch herunterfährt. Unabhängig davon, ob ein Raspberry PI installiert ist oder nicht, schaltet sich die Stromversorgung nach dem langen Tastendruck erst nach ca. 10 Sekunden ab. Das Skript war auf meinem Image bereits vorinstalliert.

Im Menüpunkt „Geräteauswahl" wird das verwendete Board ausgewählt. Zum Zeitpunkt der Erstellung dieses Artikels standen in der Version 3.0.1 der Raspberry PI Pico und der ESP32 noch nicht zur Verfügung und konnten daher nicht getestet werden. Seit Ende September werden in der Version 3.1.0 auch diese beiden populären Mikrocontrollerboards unterstützt. Das Update erkennt die alte Version und überschreibt diese mit der neuen Version. Übrigens kann immer nur ein Mikroprozessorsystem aktiv sein, wie Bild 9 zeigt.

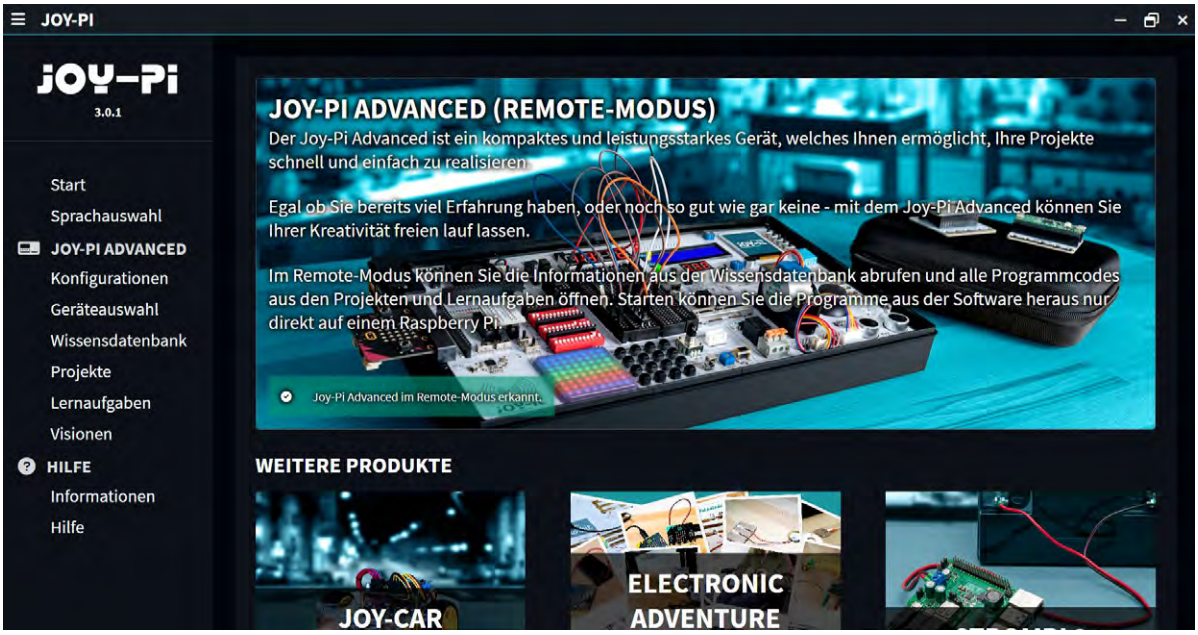


Bild 8: Startbildschirm des Programms JOY-PI

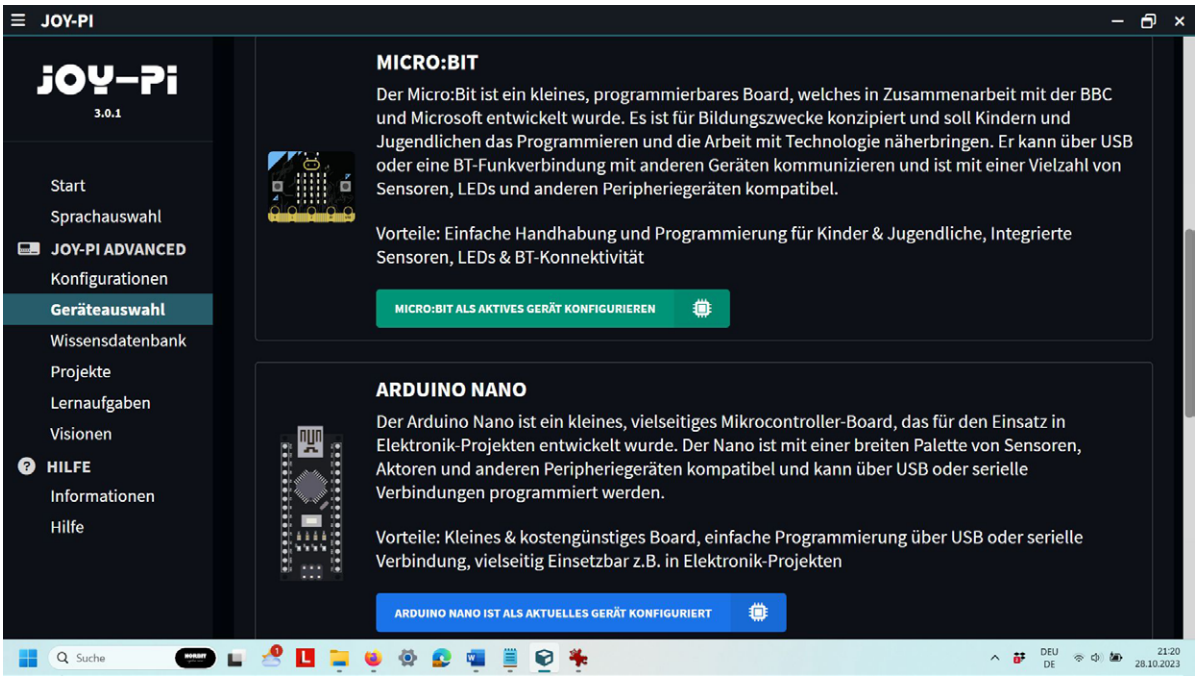


Bild 9: Arduino Nano als aktives Board eingestellt

Bild 10: Übersicht Barometer-Sensor

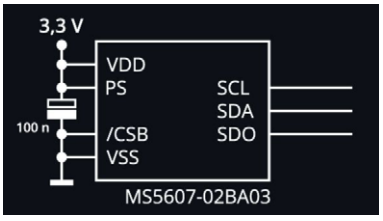


Bild 11: Schaltplan des Barometer-Sensors

Ein echtes Highlight ist die „Wissensdatenbank". Hier wird zuerst einmal ein Bild der Evaluierungsplattform angezeigt, und darunter wird jeder einzelne Bereich genau erklärt. Ich möchte dies am Beispiel des Arduino Nano und des Barometer-Sensors zeigen. Schon in der Übersicht werden die Grundfunktion sowie die notwendigen Schaltereinstellungen bzw. eine I2C-Adresse angezeigt. Dieser Sensor ist über I2C angeschlossen (keine Einstellungen am DIP-Schalter notwendig) und benutzt die I2C-Adresse 0x77.

Klickt man auf den Sensor, so wird zuerst eine allgemeine Erklärung eines Barometers, gefolgt von einer technischen Erklärung angezeigt (Bild 10). Danach ist der Schaltplan des Barometer-Sensors ersichtlich (Bild 11), gefolgt von Hinweisen, wie die Höhenmessung mit dem Sensor verbessert werden kann.

Basierend auf der Bezeichnung des Sensors kann man im Internet herausfinden, dass es sich um einen Sensor von TE-Connectivity handelt, und so auch leicht ein Datenblatt finden, falls man tiefer in die Materie dieses Sensors einsteigen möchte.

Da ich den Arduino Nano ausgewählt habe, folgt eine Einweisung, wie dieser am Arduino zu verwenden ist. Man erfährt, welche Bibliothek man einbinden muss, und die unterschiedlichen Programmteile in C++ für die Arduino IDE werden genau vorgestellt und beschrieben (Bild 12).

Schlussendlich wird das komplette Programm C++ aufgelistet, und man kann dies auch über eine Schaltfläche direkt in die Zwischenablage und danach in die Arduino IDE übertragen.

Schauen wir uns an, wie diese Sensor-Beschreibung aussieht, wenn der Raspberry PI anstelle des Arduino Nano ausgewählt wird: Die Beschreibung inklusive Schaltplan ist gleich, nur beim Code wird anstelle von C++ der Python-Code für des Raspberry PI angezeigt (Bild 13).

Wird die dritte bereits unterstützte Mikroprozessorplattform BBC micro:bit ausgewählt, wird die Programmierung in Scratch angezeigt (Bild 14). Hier muss man leider selbst Hand anlegen, da die Scratch-Dateien nicht kopiert werden können.

Alle 42 Kapitel dieses Abschnitts laden regelrecht dazu ein, die Funktionen nacheinander kennenzulernen. Die Aufbereitung und die Tiefe dieses Abschnitts „Wissensdatenbank" sind definitiv ein Highlight des JOY-PI Advanced.

```
#include <MS5607.h>

// defining object for barometer
MS5607 barometer(0x77);
```

Die Fließkommavariablen werden später im Programm mit den Messwerten befüllt.

```
// defining parameters for printing
float temperature;
float pressure;
float altitude;
```

Wir öffnen den seriellen Port, um Statusausgaben in Textform in der Arduino-IDE (*Werkzeuge/Serieller Monitor*) anzuzeigen und initialisieren danach den Sensor (wobei wir darauf warten, bis er wirklich bereit ist):

```
void setup() {
  // setup serial communication
  Serial.begin(9600);
  // wait until barometer is recognized
  while(!barometer.begin()){}
```

Bild 12: Programmteile des Barometer-Sensors für den Arduino Nano

```
# import necessary libraries
from JoyPiAdvanced import barometer
import busio
import board
```

Dann können die Objekte für den Buszugriff und das Barometer initialisiert werden:

```
# initilize barometer
i2c = busio.I2C(board.SCL, board.SDA)
baro = barometer(i2c)
```

Das Programm gibt dann eine Tabelle aus, in der die Temperatur, der Luftdruck und die sich daraus ergebene Höhe angezeigt wird, wobei als Referenzwert der bereits erwähnte Normaldruck genutzt wird:

```
# prepare table for printing values
print("| temperature |   pressure    | altitude  |")
print(45 * "-")
# change the reference_pressure to your local pressure to calculate your altitude
print("|   ", baro.get_temperature(), "   |   ", baro.get_pressure(), "   |   ",
```

Bild 13: Programmteile des Barometer-Sensors für den Raspberry PI

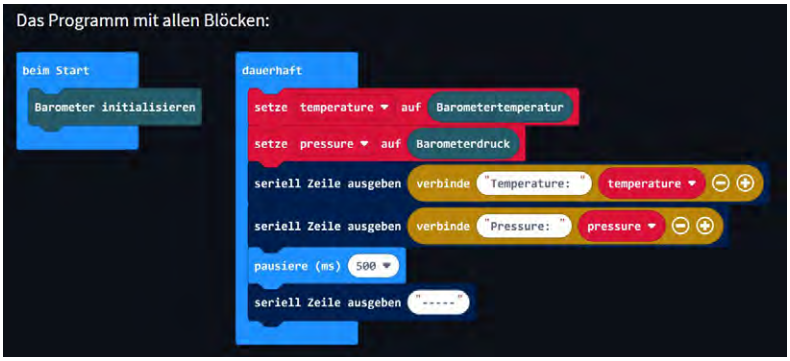


Bild 14: Programmteile des Barometer-Sensors für den micro:bit

Während das Kapitel „Wissensdatenbank" jede einzelne Funktion eigenständig behandelt, geht es in der Rubrik „Projekte" darum, verschiedene Bausteine der Entwicklungsplattform miteinander interagieren zu lassen.

Zehn Projekte warten hier auf die Erkundung durch den Nutzer und sind für die derzeit drei unterstützten Mikroprozessorplattformen verfügbar.

- Beim **Projekt Farberkennung** wechselt die Farbe der RBG-Matrix , sobald ein Objekt über den Farbsensor gehalten wird, und zeigt die primäre Farbe des Objekts an.
- Beim **Projekt Interaktives Gyroskop** zeigt ein Pfeil auf der LED-Matrix an, in welche Richtung der Gyrosensor auf dem JOY-PI Advanced geneigt wird.
- Beim **Projekt Reaktionstester** wird eine zufällige Würfelzahl auf der LED-Matrix angezeigt, und die dazugehörige Sensortaste muss danach so rasch wie möglich gedrückt werden. Die gemessene Reaktionszeit wird auf dem LC-Display angezeigt.
- Beim **Projekt RGB-Cobra** wird wie beim Spiel „Snake" eine Schlange auf der RGB-Matrix zum Futterplatz mittels des Joysticks gelenkt.
- Beim **Projekt Fernsteuerung** werden Tastendrücke der mitgelieferten Fernbedienung decodiert und der Tastencode am LC-Display angezeigt. Beim Erkennen eines validen Codes ertönt außerdem der Buzzer.
- Beim **Projekt Bewegungsalarm** fängt der Vibrationsmotor an zu rattern und der Summer ertönt, sobald der Bewegungssensor eine Bewegung erkannt hat.
- Beim Projekt „Magnetfeld-Visualisierung" werden das Magnetfeld und die Richtung am TFT-Display über einen Kreis visualisiert.
- Beim **Projekt Schrittmotoren- Steuerung** wird die Geschwindigkeit des Schrittmotors über das Potentiometer und die Drehrichtung über einen Schalter gesteuert.
- Beim **Projekt Systemüberwachung** wird die Temperatur zusammen mit den Systemdaten des Mikroprozessors am OLED-Display angezeigt und der Lüfter entsprechend der Temperatur gesteuert.
- Beim **Projekt Interaktiver Safe** muss ein Zahlencode korrekt eingegeben werden, wobei die Zahlen am TFT-Display und die Belohnung an OLED-Display angezeigt werden.

Ist der Raspberry PI ausgewählt und klickt man auf die Schaltfläche „Projekt öffnen", so wird ein Windows-Explorer mit der entsprechenden Python-Datei geöffnet.

Bei der Auswahl des Arduino Nano öffnet sich ein Directory mit einer C++-Ino-Datei und beim BBC micro:bit ein Directory mit einer Java-Script-Datei. Die jeweiligen Programmdateien sind sehr gut in englischer Sprache dokumentiert.

Einen Schritt weiter geht der Reiter „Lernaufgaben". Dieser enthält zehn Aufgaben, die sich je nach eingesetzter Mikroprozessorplattform ein wenig unterscheiden. Ziel ist es wohl, dass man versucht, diese Lernaufgaben selbst zu lösen und bei Bedarf Hilfe in den beigefügten Lösungen zu finden. Dies beginnt bei einfacheren Projekten wie der Temperaturmessung mittels externem Temperatursensor und geht bis zu einer Wetterstation und einem Bilderrahmen.

Der Reiter „Visionen" enthält als letzten Übungsschritt 14 Projektideen, wo zwar die Aufgaben beschrieben werden, nicht jedoch die Lösungen enthalten sind. Die Vorgangsweise beginnend bei der „Wissensdatenbank", wo die Programmierung einzelner Elemente geübt wird, hin zu „Projekten", wo die Neugierde geweckt wird, was man nicht alles mit dem JOY-PI Advanced anfangen kann, bis hin zu „Lernaufgaben" und „Visionen", wo schon komplexe Programmieraufgaben warten, erscheint mir logisch und für einen nachhaltigen Lerneffekt sehr gut geeignet.

In den weiteren beiden Abschnitten meines Erfahrungsberichts befasse ich mich damit, Sensoren aus der „Wissensdatenbank" sowie „Projekte" am Raspberry PI und am Arduino Nano zum Laufen zu bringen. Lediglich der BBC micro:bit bleibt außen vor, da ich kein entsprechendes Board besitze.

Verwendung mit dem Raspberry PI

Der Einbau des Raspberry PI (siehe Kapitel 4 der Bedienungsanleitung) ist kinderleicht und erfolgt über eine Klappe auf der Unterseite der Evaluierungsplattform, die mittels Magneten mit dem Gehäuse verbunden ist. Unter dieser Klappe verbergen sich noch zwei weitere interessante Elemente. Eine Knopfzelle des Typs CR2032 als Pufferbatterie für die Echtzeituhr sowie der Hauptanschluss der Stromversorgung an die Hauptplatine über eine Hohlbuchse.

Bild 15: Desktop des Raspberry PI

Bild 16: Wenn der Raspberry PI erkannt wird, ist der Start gelungen.

Nun ging es um die Inbetriebnahme des Raspberry PI mit der von Joy-IT eigens entwickelten Lernzentrale, die schon auf der beigefügten SD-Karte vorinstalliert ist. Der Raspberry PI wird von der internen Stromquelle versorgt, man muss lediglich eine Maus und Tastatur z. B. über USB sowie einen externen Monitor anschließen. Dies reduziert ein wenig die Portabilität des Systems und könnte in der Bedienungsanleitung besser beschrieben sein. Wer eine kompaktere Lösung sucht, die allerdings weniger Experimentiermöglichkeiten anbietet, ist mit der Joy-IT-Experimentierlösung JoyPi Note besser bedient.

Nach dem Anschluss der Maus an den USB-Anschluss, der Tastatur und des Monitors bootete dann der Raspberry PI wie gewohnt und es erschien der Desktop wie in Bild 15 zu sehen.

Bevor man loslegen kann, benötigt es noch einige Vorarbeiten. Rechts oben in der Menüleiste sollte man den Raspberry PI mit seinem eigenen WLAN verbinden, um an Updates zu kommen. Ist das erledigt, startet man die Anwendung JOY-PI, indem man auf diese doppelt klickt. Die Anwendung schaut exakt gleich aus wie die Windows-Applikation, die ich im vorhergehenden Kapitel beschrieben habe. Links unten im Bild des JOY-PI Advanced sollte „JOY-PI Advanced erkannt" stehen (Bild 16).

Falls der Raspberry PI nicht erkannt wird, liegt es meistens daran, dass der Boardconnector (siehe Beginn des Artikels) nicht oder falsch gesteckt ist, da die Software die Präsenz einiger Sensoren beim Start prüft.

Danach habe ich unter „Konfiguration" noch die Bibliotheken installiert, da ich mir nicht sicher war, ob diese schon installiert waren. Dies dauert eine ganze Weile. Unter „Konfiguration" sollte noch kontrolliert werden, ob das Shutdown-Skript installiert ist. Zur Sicherheit habe ich in einer Konsole noch ein Update/Upgrade mit „sudo apt-get update" und „sudo apt-get upgrade" gemacht.

Ich empfehle, mit den Programmen für die Sensoren zu beginnen, und beschreibe hier im Detail, wie man das Relais testet.

Unter „Wissensdatenbank" wählt man „Relais" aus und stellt den Schalter B2 auf ON und alle anderen auf OFF. Danach scrollt man ganz nach unten zum Eintrag „Vollständige Programmdatei" und klickt auf „Copy", um die Daten in die Zwischenablage zu kopieren.

Doch wie geht es jetzt weiter? Genau wie die gesamte Inbetriebnahme des Raspberry PI mit den externen Komponenten wie Tastatur, Bildschirm etc. ist das nirgendwo beschrieben und eröffnet sich für einen Raspberry-PI-Anfänger wie mich nicht automatisch.

Die Lösung liegt darin, auf die Raspberry-Himbeere links oben zu klicken und dann Entwicklung → Thonny auszuwählen. Dies ist eine einfach zu bedienende Oberfläche für Python-Code. Mit der rechten Maustaste und „Einfügen" wird der Beispielcode von der Zwischenablage in die Anwendung Thonny kopiert (Bild 17).

Drückt man nun den grünen Pfeil „Ausführen", so schaltet sich das Relais im Sekundentakt ein und aus. Ein großartiges erstes Erfolgserlebnis – der Weg dahin war allerdings für mich sehr steinig. Die Ausführung kann mit „Ausführung beenden" wieder gestoppt werden.

Die Tests für den Vibrationsmotor, die Schalter, den Geräuschsensor, das Barometer, das Gyroskop, die Touchsensoren und des EEPROM-Speichers verliefen ohne Probleme.

Beim DHT-Temperatur- und Feuchtigkeitssensor gab es bei mir die erste Fehlermeldung „Unable to set line 18 to input". Erst nach einem Neustart, bei dem C9 eingeschaltet war, lief das Programmbeispiel einwandfrei.

Auch bei der RGB-Matrix gab es eine kryptische Fehlermeldung. Hier half die Hilfefunktion, da auch andere User schon das gleiche Problem hatten. Die Lösung findet sich beim JOY-PI-Helpdesk, allerdings musste ich statt

sudo nano /usr/share/applications/
Thonny.desktop

dies eingeben:

sudo nano /usr/share/applications/
org.thonny.Thonny.desktop

Wenn das Demoprogramm läuft, sind die Effekte der RGB-Matrix allerdings sehr spektakulär.

Die Tests des Erschütterungssensors, des Analog-digital-Converters, des Joysticks, des NTC-Temperatursensors, des Potentiometers, des Magnetfeld-/Hallsensors, des Lichtsensors, des Farbsensors und des 7-Segment-Displays klappten auf Anhieb.

Das Beispiel der Echtzeituhr erscheint mir nicht sehr sinnvoll, da in diesem keine Zeit in der RTC gesetzt wird und daher auch die Ausgabe leer ist.

Das Demoprogramm für das OLED-Display lief anfangs nur teilweise, da die Datei „joy-it.jpg" nicht gefunden wurde.

Diese befindet sich im Verzeichnis
/opt/JOY-PI/joypiadvanced/data/
knowledgebase/data/oled/raspberrypi/code

Öffnet man diesen Pfad im Dateimanager, klickt mit der rechten Maustaste auf „oled.py" und wählt dort Thonny aus, dann funktioniert das Testprogramm, wie es soll.

Die Testprogramme für den Servomotor und den Ultraschall-Abstandssensor liefen problemlos, während das Testprogramm für das TFT-Display wie beim OLED-Display aus dem Directory

/opt/JOY-PI/joypiadvanced/data/knowledgebase/data/tft/raspberrypi/code

ausgeführt werden muss (Datei „tft.py").

Das Testprogramm für das 16x2-LCD-Display funktioniert gut, wobei gegebenenfalls der Kontrast einzustellen ist. Die Testprogramme für den Schrittmotor, die Lichtschranke und den Buzzer liefen auf Anhieb problemlos.

Das Testprogramm für den DS18B20 bekam ich nicht zum Laufen. Es blieb zuerst in der Initialisierungsroutine mit der Nachricht „Waiting for initialization..." hängen, auch ein Neustart half nicht, und die Suche in der Hilfe war ebenso ergebnislos.

Die Testprogramme für den Dreh-Encoder, den Infrarot-Empfänger, den PIR-Bewegungssensor, den PWM-Lüfter, das RFID-Modul und die Button-Matrix funktionierten hingegen problemlos.

Die zehn Projekte bekam ich bis auf das „Interaktive Gyroskop", das immer ein paar kryptische Fehlermeldungen ausgab, zum Laufen.

Diese Projekte zeigen anschaulich, welch komplexe Funktionen mit ein wenig Programmcode zu realisieren sind, und machen alle enorm viel Spaß.

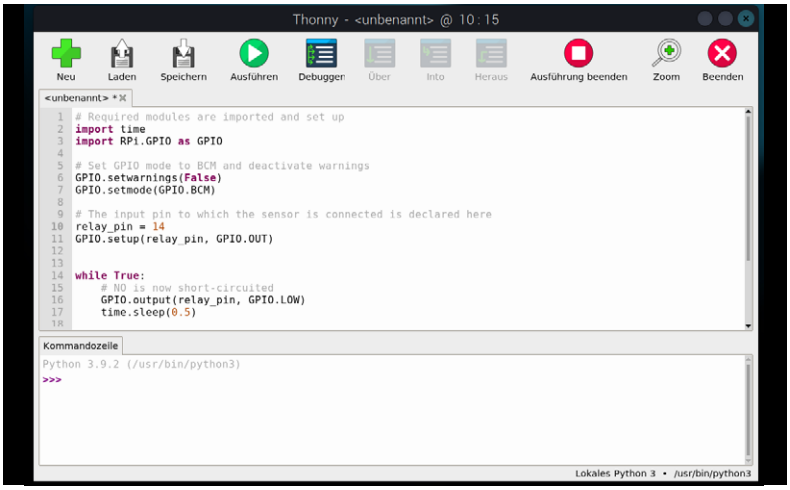


Bild 17: Relais-Beispielcode in Thonny geladen

Verwendung des Arduino Nano

Als erfahrener Programmierer von verschiedenen Arduino-Plattformen und ESP32-Boards liegt mir dieser Zugang viel näher als der des Raspberry PI mit seiner Python-Programmierung.

Für den Arduino Nano verwendet man die Windows-Software und stellt unter „Geräteauswahl" den Arduino Nano als aktives Gerät ein.

Der Arduino Nano wird auf das zugehörige Adapterboard gesteckt und dieses dann links an die Entwicklungsplattform gesteckt. Der Boardconnector Extern muss wie in Bild 3 gesteckt sein. Danach schaltet man die Entwicklungsplattform über den Power-Button wieder ein.

Leider ist nicht dokumentiert, ob man Adapterboards unter Spannung ein- oder ausstecken darf!

Mittels eines passenden USB-Kabels wird der Arduino Nano dann mit dem PC verbunden. Für einen ersten Test suchte ich mir den DS18B20-Temperatursensor aus, den ich beim Raspberry PI nicht zum Laufen brachte. Ich startete die Arduino IDE, legte eine neues Projekt an, kopierte den Code in die Arduino IDE und speicherte das Projekt ab. Unter „Board" wählte ich den Nano und die korrekte Schnittstelle aus (Bild 18).

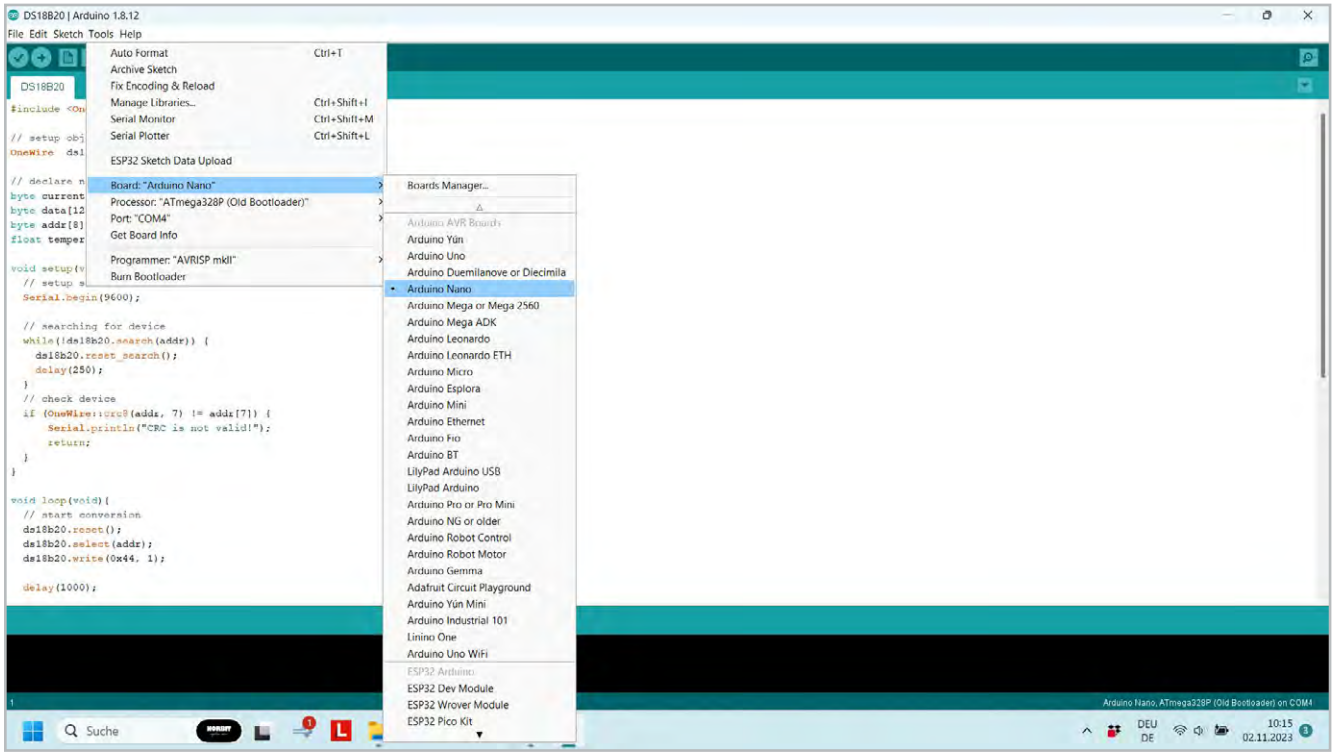


Bild 18: Auswahl des Nano in der Arduino IDE

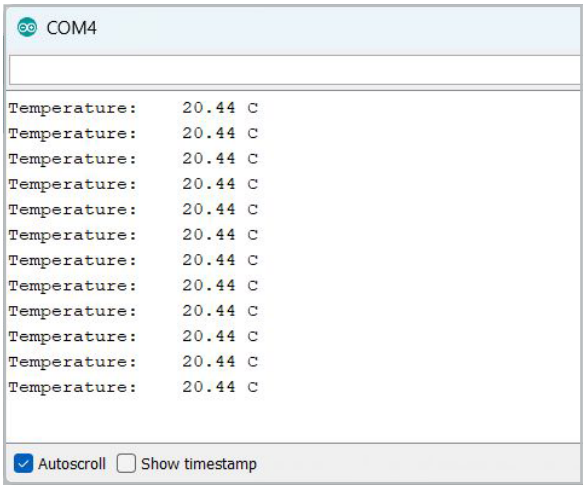


Bild 19: Temperaturanzeige im Seriellen Monitor der Arduino IDE

Danach drückte ich auf Uploading. Der Sketch kompilierte und übertrug sich ohne Probleme. Nach dem Öffnen des Seriellen Monitors und der Umstellung auf 9600 Baud wurde die Temperatur jede Sekunde korrekt angezeigt (Bild 19). Das bedeutet, dass die Ursache für das Nichtfunktionieren am Raspberry PI mit Problemen im Treiber oder dem Python-Code zusammenhängen muss.

Im Folgenden beschreibe ich noch die Erfahrungen mit einigen Testprogrammen aus der „Wissensdatenbank" und einem „Projekt", das beim Raspberry PI Probleme bereitet hat.

Beim Testprogramm „DHT Temperatur- und Feuchtigkeitssensor" muss zuerst eine Library für den DHT11 eingebunden werden.

Dazu geht man auf „Konfigurationen" und dann zum Eintrag „Bibliotheken (Arduino Nano)". Dort sucht man nach dem Eintrag „DHT Sensor Library" und klickt darauf.

Es öffnet sich ein GIThub-Link mit der entsprechenden Library. Rechts oben klickt man auf Code und wählt „Download ZIP" aus. Das ZIP-File kopiert man anschließend in das Installationsverzeichnis der Arduino-IDE in das Unterverzeichnis „libraries". Dort entpackt man das ZIP-File, sodass das Unterverzeichnis „DHT-sensor-library-master" entsteht.

Jetzt sollte – eventuell nach einem Neustart der Arduino IDE – sich das Testprogramm kompilieren und übertragen lassen. Der Serielle Monitor zeigt dann Feuchtigkeit und Temperatur an.

Noch ein Hinweis: Die Libraries können auch alle als eine große ZIP-Datei heruntergeladen werden, die dann nur mehr im entsprechenden Verzeichnis für die Libraries zu entpacken ist.

Beim Testprogramm für die RGB-Matrix muss die Library „Adafruit Neopixel Library" wie im vorhergehenden Beispiel beschrieben installiert werden, damit alles reibungslos funktioniert.

Auch das Testprogramm „Echtzeituhr" benötigt eine neue Library, sie hat den Namen „RTCDS1307 Library". Das Ergebnis ist hier viel besser als beim Raspberry PI: Die Uhr wird zuerst auf ein gewisses Datum und eine gewisse Uhrzeit gesetzt (im Programm änderbar), dann jede Sekunde aus der RTC abgerufen und im Seriellen Monitor angezeigt.

Beim Testprogramm OLED ist das Bild schon im Code eingefügt, und mit den korrekt installierten Libraries läuft es einwandfrei.

Beim Testprogramm TFT-Display wird zwar gezeigt, wie man verschiedene Formen und Muster generiert, aber im Gegensatz zum Raspberry PI nicht, wie man ein Bild anzeigt. Das dürfte den limitierten Ressourcen des Arduino Nano geschuldet sein.

Schließlich habe ich mir noch das Projekt „Interaktives Gyroskop" angesehen. Nach dem Klick auf „Projekt öffnen" zeigt sich im Windows-Explorer ein Ordner mit der entsprechenden Datei

„Arduino interactiveGyroscopeProject.ino"

In der Arduino IDE funktionierte auch dieses Projekt perfekt.

Fazit

Die Entwicklungsplattform und Lernzentrale JOY-PI Advanced ist eine umfangreiche und umfassende Plattform mit nahezu unbegrenzten Möglichkeiten. Hier meine ganz persönliche Pro- und Contraliste:

Pro

+ Von Makern für Maker gemacht
+ Riesiger Funktionsumfang
+ Einsetzbar für die wichtigsten Mikroprozessorplattformen am Markt
+ Zukunftssicher durch das Konzept der Adapterboards
+ Port-LEDs erleichtern die Fehlersuche ungemein und ersetzen in vielen Fällen einen teuren Logic-Analyser
+ Software für Raspberry PI und Windows
+ Kompakt und portabel

Contra

- Spärliches Handbuch, das die Grundschritte beim Einsatz eines Raspberry PI oder Arduino Nano nicht enthält
- Für Anfänger ist eine relativ lange Einarbeitungszeit nötig

Da ich nun das JOY-PI Advanced kennen- und schätzen gelernt habe, werde ich mir in Zukunft eine Plattform für meine schulischen Anwendungen und eine Plattform für meine privaten Elektronikprojekte zulegen. Obwohl ich nun schon einiges probiert habe, bleiben noch viele spannende Dinge zu entdecken. ELV