

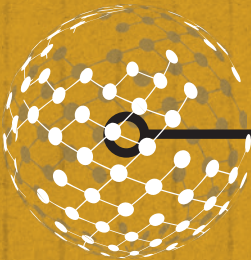
Der komplette Quellcode
der Projekte für Arduino™
und Raspberry Pi zum
Download.



FRUIT UP
YOUR
FANTASY

WILFRIED KLAAS

BUSSYSTEME IN DER PRAXIS



Serielle Schnittstelle, SPI, I²C,
1-Wire, USB, KNX und CAN:
Schnittstellen verstehen und in
Projekten mit Arduino™ und
Raspberry Pi direkt einsetzen.

FRANZIS

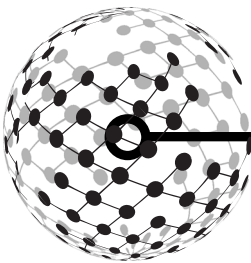
WILFRIED KLAAS

BUSSYSTEME IN DER PRAXIS

Wilfried Klaas studierte Elektrotechnik an der Ruhr Universität Bochum (Abschluss Dipl.-Ing.) und arbeitet bei der EASY Software AG als System-Architekt. Außer mit Musik beschäftigt er sich intensiv mit Mikrocontrollern sowie deren Programmierung und sucht dabei immer die Verbindung zu seinem zweiten großen Hobby, dem Modellbau. Als freier Autor hat er diverse Publikationen zu diesen Themen verfasst.
Kontakt: w.klaas@gmx.de, <http://www.rcarduino.tk>

FRUIT UP
YOUR
FANTASY

WILFRIED KLAAS



BUSSYSTEME

IN DER PRAXIS

Serielle Schnittstelle, SPI, I²C,
1-Wire, USB, KNX und CAN:
Schnittstellen verstehen und in
Projekten mit Arduino™ und
Raspberry Pi direkt einsetzen.

Bibliografische Information der Deutschen Bibliothek

Die Deutsche Bibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte Daten sind im Internet über <http://dnb.ddb.de> abrufbar.

Alle Angaben in diesem Buch wurden vom Autor mit größter Sorgfalt erarbeitet bzw. zusammengestellt und unter Einschaltung wirksamer Kontrollmaßnahmen reproduziert. Trotzdem sind Fehler nicht ganz auszuschließen. Der Verlag und der Autor sehen sich deshalb gezwungen, darauf hinzuweisen, dass sie weder eine Garantie noch die juristische Verantwortung oder irgendeine Haftung für Folgen, die auf fehlerhafte Angaben zurückgehen, übernehmen können. Für die Mitteilung etwaiger Fehler sind Verlag und Autor jederzeit dankbar. Internetadressen oder Versionsnummern stellen den bei Redaktionsschluss verfügbaren Informationsstand dar. Verlag und Autor übernehmen keinerlei Verantwortung oder Haftung für Veränderungen, die sich aus nicht von ihnen zu vertretenden Umständen ergeben. Evtl. beigefügte oder zum Download angebotene Dateien und Informationen dienen ausschließlich der nicht gewerblichen Nutzung. Eine gewerbliche Nutzung ist nur mit Zustimmung des Lizenzinhabers möglich.

Arduino ist ein eingetragenes Markenzeichen der Arduino S.r.l.

© 2015 Franzis Verlag GmbH, 85540 Haar bei München

Alle Rechte vorbehalten, auch die der fotomechanischen Wiedergabe und der Speicherung in elektronischen Medien. Das Erstellen und Verbreiten von Kopien auf Papier, auf Datenträgern oder im Internet, insbesondere als PDF, ist nur mit ausdrücklicher Genehmigung des Verlags gestattet und wird widrigenfalls strafrechtlich verfolgt.

Die meisten Produktbezeichnungen von Hard- und Software sowie Firmennamen und Firmenlogos, die in diesem Werk genannt werden, sind in der Regel gleichzeitig auch eingetragene Warenzeichen und sollten als solche betrachtet werden. Der Verlag folgt bei den Produktbezeichnungen im Wesentlichen den Schreibweisen der Hersteller.

Autor: Wilfried Klaas

Programmleitung: Dr. Markus Stäuble

Satz: DTP-Satz A. Kugge, München

art & design: www.ideehoch2.de

Druck: M.P. Media-Print Informationstechnologie GmbH, 33100 Paderborn

Printed in Germany

ISBN 978-3-645-65310-7

Vorwort

Schnittstellen und Bussysteme sind sehr wichtige Grundlagen der Mikrocontroller-Programmierung, ermöglichen sie doch die Kommunikation mit der »Außenwelt«.

In diesem Buch stelle ich Ihnen zunächst die verschiedenen relevanten Schnittstellen und Bussysteme vor, die für diese Kopplung wichtig sind.

Im zweiten Teil steigen Sie dann in die Programmierung ein. Ich werde mich auf zwei der wichtigsten Systeme beschränken: Arduino und Raspberry Pi. Die vermittelten Grundlagen können Sie später auf andere Mikrocontroller übertragen.

Alle hier vorgestellten Schaltungen und Programme wurden nach bestem Wissen erstellt und getestet. Jedoch kann für die korrekte Funktion keine Haftung übernommen werden.

Ich wünsche viel Spaß bei der Lektüre.

Wilfried Klaas

Inhaltsverzeichnis

I	Die Hardware.....	11
1.1	Arduino™	11
1.1.1	Schnittstellen	12
1.2	Raspberry Pi	14
2	Grundlagen der Schnittstellen	17
2.1	Serielle Schnittstelle	17
2.1.1	USART	18
2.1.2	RS232	19
2.1.3	RS485	22
2.1.4	DMX.....	23
2.1.5	Modbus	25
2.1.6	PROFIBUS	27
2.1.7	NMEA 0183	28
2.2	SPI	29
2.2.1	Protokoll	30
2.3	I ² C.....	32
2.3.1	Elektrisch.....	32
2.3.2	Takt und Zustände.....	33
2.3.3	Adressierung	33
2.3.4	Protokoll	33
2.4	1-Wire	34
2.4.1	Protokoll	34
2.5	USB	36
2.5.1	Übertragungsarten.....	39
2.5.2	Datenübertragungsraten.....	41
2.5.3	Verwendung	41
2.6	CAN.....	41
2.6.1	Protokoll	42
2.6.2	Elektrisch.....	44
2.6.3	Übertragungsraten.....	45
2.6.4	CANopen	45
2.6.5	NMEA 2000.....	46
2.7	KNX	46
3	Anwendungsbeispiele mit dem Arduino™	49
3.1	Seriell und USB.....	49
3.1.1	USB: Kommunikation mit dem Host	52
3.1.2	USB: Steuerung von Aufgaben	54
3.1.3	USB: Messwertaufzeichnung.....	62
3.1.4	NMEA 0183: GPS-Modul, Schnittstellenkonvertierung.....	70
3.1.5	DMX mit dem Arduino™	75
3.2	SPI	84

3.2.1	SPI: Schieberegister	84
3.2.2	SPI: Hardware	87
3.2.3	SPI: Vierstelliges Sieben-Segment-LED-Display	89
3.2.4	SPI: 16-Bit-Porterweiterung mit dem MCP23S17	93
3.2.5	SPI: Ansteuerung zweizeiliges LC-Display	97
3.2.6	SPI: Ansteuerung Grafik-LCD	98
3.2.7	SPI: 5x7-DOT-Matrix-LED mit MAX6952	106
3.2.8	SPI: 8-Digit-LED-Driver, MAX7221	109
3.2.9	SPI: Funkstrecke mit dem nRF24L01	111
3.2.10	SPI: SD-Karten und SDHC-Karten	116
3.3	I ² C	120
3.3.1	I ² C: Echtzeit-Uhrenbaustein DS3231	120
3.3.2	I ² C: A/D-Wandler 12 Bit, 8 ksps Sampling, MAX127	126
3.3.3	I ² C: D/A-Wandler 12 Bit, MCP4725	129
3.3.4	I ² C: Gyroskop und Beschleunigungssensor MPU6050	132
3.3.5	I ² C: Magnetometer HMC5883L	136
3.3.6	I ² C: EEPROM 24C64	138
3.3.7	I ² C: Temperaturmessung, DS1621 / LM75	141
3.3.8	I ² C: Luftdruckmessung, BMP085 / BMP180	143
3.4	1-Wire	145
3.4.1	1-Wire: Temperaturmessung mit dem DS18B20	145
3.5	CAN-Bus	147
3.6	Sonstige	150
3.6.1	Ultraschall mit dem HC-SR04	150
3.6.2	RGB-LED WS2812B	152
3.7	Design eines minimalen Arduino™-Mikrocontrollerboards	158
4	Anwendungsbeispiele mit dem Raspberry Pi	161
4.1	Seriell und USB	161
4.1.1	USB: Kommunikation mit dem Host	162
4.1.2	USB: Steuerung von Aufgaben	164
4.1.3	NMEA 0183: GPS-Modul	165
4.2	SPI	169
4.2.1	SPI: Schieberegister	171
4.2.2	SPI: 16-Bit-Porterweiterung mit dem MCP23S17	173
4.2.3	SPI: Ansteuerung Grafik-LCD, PiTFT mit Touchscreen	175
4.2.4	SPI: 8-Digit-LED-Driver, MAX7221	178
4.2.5	SPI: Funkstrecke mit dem nRF24L01	182
4.3	I ² C	187
4.3.1	I ² C: Der Echtzeit-Uhrenbaustein (RTC) DS3231	188
4.3.2	I ² C: A/D-Wandler 12 Bit, 8 ksps Sampling, MAX127	190
4.3.3	I ² C: D/A-Wandler 12 Bit, MCP4725	191
4.3.4	I ² C: Gyroskop und Beschleunigungssensor MPU6050	193
4.3.5	I ² C: Magnetometer HMC5883L	195
4.3.6	I ² C: Temperaturmessung, DS1621, LM75	197
4.3.7	I ² C: Luftdruckmessung mit BMP085, BMP180	198
4.4	1-Wire	200
4.4.1	1-Wire: Temperaturmessung mit dem DS18B20	200
4.5	Sonstige	203

4.5.1	Ultraschall mit dem HC-SR04	203
4.5.2	RGB-LED WS2812B.....	206
4.6	Internet.....	210
4.6.1	Installation und Test des Python-WebServers Flask	210
4.6.2	Benutzung von WebTemplates	212
4.7	Raspberry Pi und Arduino™, zwei Welten verbunden.....	214
5	Benötigte Hardware und Software	217
5.1	Programmierumgebung Arduino™.....	217
5.1.1	Installation der Arduino™-IDE	217
5.1.2	Projektinstallation.....	217
5.1.3	Sketchverzeichnis einstellen.....	218
5.1.4	Treiberinstallation - Arduino™ Uno unter Windows.....	219
5.1.5	Die Arduino™-IDE	220
5.1.6	Sketch 1: Blinklicht.....	223
5.1.7	Debugging	224
5.2	Programmierumgebung Raspberry Pi.....	226
5.2.1	Idle, integriertes Entwicklungssystem für Python	227
5.2.2	Ninja-IDE.....	228
6	Hardware	231
7	Internetseiten.....	233
	Stichwortverzeichnis	237

1

Die Hardware

1.1 Arduino™

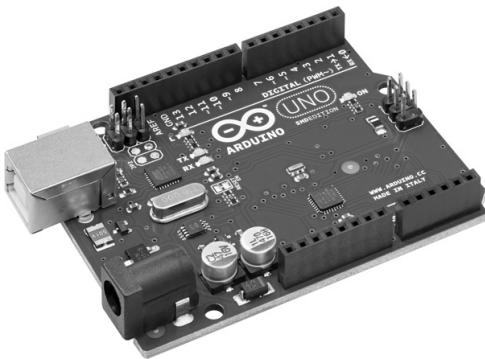


Bild 1.1: Arduino Uno

Wohl der am weitesten verbreitete Mikrocontroller im DIY-Bereich ist der Arduino. Er ist zwar schon etwas älter, hat aber nichts von seiner Attraktivität verloren. Im Kern handelt es sich um einen Atmel-Chip aus der ATmega-Reihe. Je nach Board findet ein ATmega328 (Arduino Uno), ein ATmega32U16 (Arduino Leonardo) oder ein ATmega256 (Arduino Mega) Verwendung. Die ATmegas sind 8-Bit-Prozessoren mit einer Taktfrequenz von 16 MHz. Die Ressourcen wie RAM und Flash sind natürlich begrenzt, reichen aber für die üblichen Anwendungen aus. Große Vorteile der Arduinos sind die weite Verbreitung, die große Gemeinschaft und die niedrigen Hard-

und Softwarekosten. Ein offizielles Board kostet zwar fast so viel wie ein Raspberry Pi, aber man bekommt ein komplettes Entwicklungssystem mit dazu. Und mittlerweile gibt es preisgünstige kompatible Varianten. Wer sich etwas eingearbeitet hat, kann später auch eigene Boards designen und direkt an seine Bedürfnisse anpassen. Ein ATmega-Basis-Board kann man dann selber auf einer Lochrasterplatine für weniger als 5 € herstellen.

Der Arduino hat bereits eine Vielzahl von Schnittstellen integriert. So finden Sie bei einem originalen Board neben SPI und I²C auch direkt eine funktionierende USB-Anbindung. Diese ist als reine Seriell/USB-Konvertierung ausgelegt.

Einzig der Arduino Leonardo kann als echter USB-Slave benutzt werden. Falls man einen echten USB-Slave verwenden möchte, müssen neben der Programmierung im Mikrocontroller auch die entsprechenden Treiber für das Betriebssystem erstellt werden. Und wer sich schon einmal mit der Treiberprogrammierung unter Windows oder Linux auseinandergesetzt hat, weiß, dass das kein einfaches Unterfangen ist. Der Arduino Leonardo kann aber z. B. direkt als HID (Human Interface Device) eingesetzt werden und die bereits vorhandenen HID-Treiber verwenden.

Dies ist zwar nicht seine bevorzugte Anwendung, aber selbst der Arduino kann mit entsprechender Hardware eine Internetverbindung aufbauen und so z. B. Messwerte übers Internet zur Verfügung stellen. Dabei sollte man allerdings stark auf den Ressourcenverbrauch achten.

Für dieses Buch sollten Sie schon etwas Übung im Umgang mit dem Arduino haben. Grundlegende Kenntnisse der IDE und Programmierkenntnisse sind Voraussetzung.

1.1.1 Schnittstellen

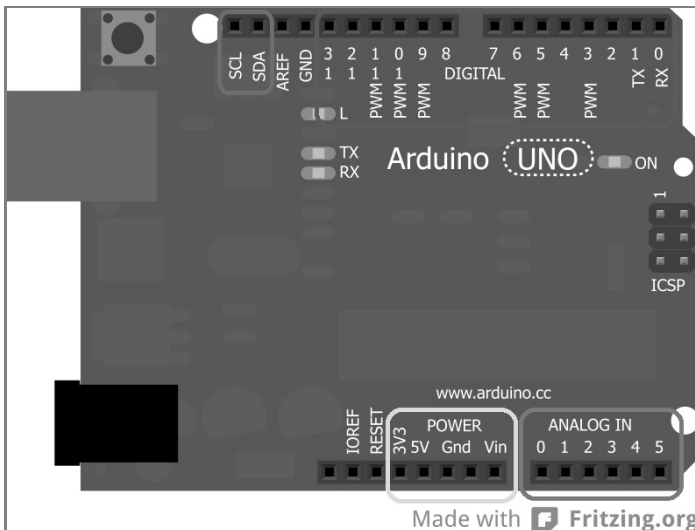


Bild 1.2:
Schnittstellen des
Arduino Uno

Die nachfolgende Tabelle beschreibt die einzelnen Schnittstellen auf der Arduino-Platine.

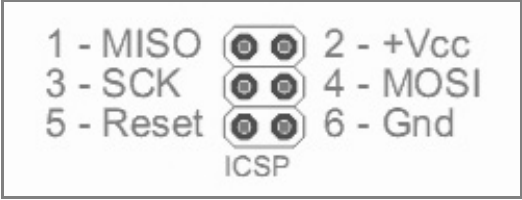
USB	Die große silberne Buchse links oben. Der USB-Anschluss ist automatisch mit der 1. seriellen Schnittstelle verbunden.
1. serielle Schnittstelle	rot, Signale TX und RX
2. serielle Schnittstelle	Beim Arduino Uno kann die 2. serielle Schnittstelle nur über eine zusätzliche Bibliothek per Softwareemulation erzeugt werden. Gut geeignet ist die AltSoftSerial-Bibliothek ¹ von Paul Stoffregen, da sie im Interruptmodus arbeitet. TX ist dann auf D9, RX auf D8. PWM auf D10 funktioniert dann wegen des in der Bibliothek benutzten Timers nicht mehr. Der Pin D10 kann aber für andere Aufgaben weiter benutzt werden.
SPI	rot oder ICSP, Signale MOSI-Pin 11 oder ICSP-4, MISO-Pin 12 oder ICSP-1, SCK-Pin 13 oder ICSP-3, SS-Pin 10. Kann normalerweise frei im roten (D0-D13) oder grünen Bereich (D14-D19) gewählt werden.
	 <p>Das Diagramm zeigt den ICSP-Header mit sechs Pins, die in zwei Spalten angeordnet sind. Die linke Spalte enthält die Pins 1 (MISO), 3 (SCK) und 5 (Reset). Die rechte Spalte enthält die Pins 2 (+Vcc), 4 (MOSI) und 6 (Gnd). Die gesamte Anordnung ist als ICSP beschriftet.</p>
	Bild 1.3: ICSP-Header
I²C	lila (grün), Signale SCL (A4), SDA (A5)
1-Wire	rot, Pin kann normalerweise frei im Bereich von D0-D19 gewählt werden.

Tabelle 1.1: Anordnung der Schnittstellen beim Arduino

¹ <https://github.com/PaulStoffregen/AltSoftSerial>

1.2 Raspberry Pi

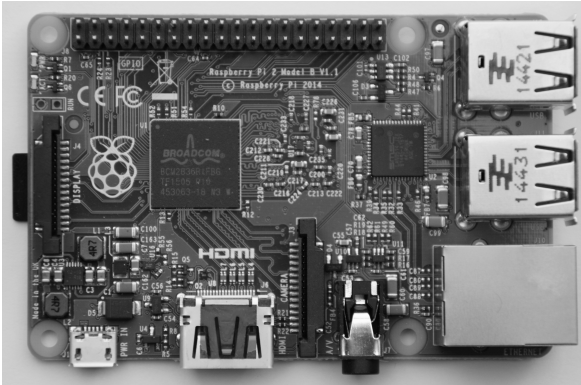


Bild 1.4: Raspberry Pi 2, Model B

Der Raspberry Pi ist ein vollständiger Computer. Als Betriebssystem wird normalerweise das Linux-Derivat Raspbian² verwendet. Der direkte Zugriff auf die Port-Pins (GPIO, General Purpose Input Output) ist möglich und ermöglicht die eigene Implementierung von Bussystemen. Doch Vorsicht, Linux ist kein Echtzeitbetriebssystem, d. h. es kann durchaus passieren, dass inmitten der eigenen Routine, die Zeichen von der seriellen Schnittstelle liest, das Betriebssystem plötzlich "etwas ganz wichtiges Anderes erledigen muss" und dann leider Zeichen verloren gehen. Das kann man durch direkte Treiberprogrammierung verhindern, die aber leider bei Linux nicht weniger komplex als in einem Windows-System ist. Zum Glück gibt es bereits fertige Treiber für die wichtigsten Schnittstellen. Den Raspberry Pi gibt es mittlerweile in verschiedenen Varianten. Ich verwende in diesem Buch hauptsächlich den Raspberry Pi 2, Model B. Er funktioniert im Prinzip genau wie der ältere Raspberry Pi B+, jedoch wurden der Arbeitsspeicher und die USB-Schnittstellen verdoppelt. Außerdem ist – sehr wichtig für unsere Anwendungen – der Prozessor ein Quadcore-Prozessor. D. h. statt bisher nur einer CPU stehen dem Raspberry Pi nun vier CPUs zur Verfügung. Das macht sich in der Geschwindigkeit aber auch in der Verfügbarkeit deutlich bemerkbar. Denn nun kann das Betriebssystem nebenbei Operationen ausführen, ohne unmittelbar die Ausführung unseres Codes zu beeinflussen. Mittelbar natürlich schon, weil die Ressourcen geteilt werden. Schnittstellen für SPI, I²C und Seriell sind schon vorhanden.

² <https://www.raspbian.org/>

Raspberry A+B					Raspberry B+ & 2				
Pin					Pin				
	3,3V	1	2	5V		3,3V	1	2	5V
I ² C: SDA	GPIO 02	3	4	5V	I ² C: SDA	GPIO 02	3	4	5V
I ² C: SCL	GPIO 03	5	6	GND	I ² C: SCL	GPIO 03	5	6	GND
GCLK	GPIO 04	7	8	GPIO 14 TX 0	GCLK	GPIO 04	7	8	GPIO 14 TX 0
	GND	9	10	GPIO 15 RX 0		GND	9	10	GPIO 15 RX 0
GEN0	GPIO 17	11	12	GPIO 18 GEN1	GEN0	GPIO 17	11	12	GPIO 18 GEN1
GEN2	GPIO 27	13	14	GND	GEN2	GPIO 27	13	14	GND
GEN3	GPIO 22	15	16	GPIO 23 GEN4	GEN3	GPIO 22	15	16	GPIO 23 GEN4
	3,3V	17	18	GPIO 24 GEN5		3,3V	17	18	GPIO 24 GEN5
SPI: MOSI	GPIO 10	19	20	GND	SPI: MOSI	GPIO 10	19	20	GND
SPI: MISO	GPIO 09	21	22	GPIO 25 GEN6	SPI: MISO	GPIO 09	21	22	GPIO 25 GEN6
SPI: CLK	GPIO 11	23	24	GPIO 08 SPI: SS0	SPI: CLK	GPIO 11	23	24	GPIO 08 SPI: SS0
	GND	25	26	GPIO 07 SPI: SS1		GND	25	26	GPIO 07 SPI: SS1
					I ² C: ID EEPROM	ID_SD	27	28	ID_SC I ² C ID EEPROM
						GPIO 05	29	30	GND
						GPIO 06	31	32	GPIO 12
						GPIO 13	33	34	GND
						GPIO 19	35	36	GPIO 16
						GPIO 26	37	38	GPIO 20
						GND	39	40	GPIO 21

Tabelle 1.2: Belegung der GPIO-Pins

Für den Raspberry-Pi-Teil sind Kenntnisse der Grundlagen im Umgang mit dem Raspberry Pi und dem Betriebssystem Raspbian Voraussetzung, ebenso Grundkenntnisse der Sprache Python. In diesem Buch wird abgesehen von ein paar Terminalscripten hauptsächlich mit der Sprache Python gearbeitet.

4

Anwendungsbeispiele mit dem Raspberry Pi

Das Betriebssystem des Raspberry Pi, Raspbian, hat als Basis ein Debian-Linux. Es gibt verschiedene Möglichkeiten, für den Raspberry Pi Programme zu erstellen. Im Folgenden werden hauptsächlich Programme in der Sprache Python beschrieben, weil die Struktur dieser Sprache recht einfach ist und sich daher auch für den weniger geübten Programmierer eignet. Viele der hier beschriebenen Beispiele und Bibliotheken gibt es natürlich auch in C oder einer anderen Sprache.

4.1 Seriell und USB

Der Raspberry Pi hat zwei bzw. vier USB-Host-Anschlüsse. An diese lassen sich diverse USB-Geräte anschließen. Beim Modell A lässt sich ein Anschluss auf OTG (USB On-The-Go) umschalten und der Raspberry Pi damit als USB-Slave betreiben. Die Programmierung dazu ist allerdings recht komplex. Noch dazu unterstützt diese Vorgehensweise nur das Modell A des Raspberry Pi. Deswegen verzichte ich an dieser Stelle auf eine weitere Vertiefung. Anders sieht das bei der seriellen Schnittstelle aus. Der Raspberry Pi besitzt eine serielle Schnittstelle auf dem GPIO-Stecker.

```
068 # mit den entsprechenden Einstellungen für den 24er-Ring
069 strip = Adafruit_NeoPixel(LED_COUNT, LED_PIN, LED_FREQ_HZ, LED_DMA,
LED_INVERT, LED_BRIGHTNESS)
070 # Initialisierung der Bibliothek. Muss einmal
071 # aufgerufen werden, bevor die Bibliothek
072 # verwendet werden kann.
073 strip.begin()
074
075 print 'Press Ctrl-C to quit.'
076 while True:
077     strip.setBrightness(255)
078     showTime()
079     time.sleep(0.2)
```

4.6 Internet

Der Raspberry Pi kann noch viel mehr. Wie wäre es mit einer Darstellung von Messwerten im WWW oder auch der Steuerung von Aktoren? Dazu benötigt man nicht viel. In den vorherigen Kapiteln haben sich einige Python-Scripts angesammelt, die nun ins WWW gebracht werden. Dazu benötigt man natürlich zunächst einen Web-Server. Hier wird das Python-Webframework Flask vorgestellt und verwendet.

4.6.1 Installation und Test des Python-WebServers Flask

Zunächst muss Flask erst einmal installiert werden. Dazu benötigt man den Python-Paket-Installer PIP. Auch der muss zunächst installiert werden:

```
001 sudo apt-get install python-pip
```

Jetzt kann man auch `flask` installieren:

```
001 sudo pip install flask
```

Zum Testen wird dieses kleines Pythonscript dienen:

```
001 from flask import Flask
002 app = Flask(__name__)
003
004 @app.route("/")
005 def hello():
006     return "Hello World!"
007
008 if __name__ == "__main__":
009     app.run(host='0.0.0.0', port=80, debug=True)
```


Zunächst werden die für dieses Script benötigten Flask-Module geladen.

Dann wird ein neues Flask-Objekt erzeugt. Mit `@app.route("/")` wird der Einstiegspunkt der Webapplikation erzeugt, in diesem Fall also das Webroot.

Es folgt eine Funktion, die einen einfachen String zurückgibt.

Mit `if __name__ == "__main__":` wird dann geprüft, ob das Python-Modul direkt aus einer Kommandozeile ausgeführt wird. Wenn ja, wird der WebServer gestartet. Dieser wird auf alle verfügbaren IP-Adressen gebunden und horcht auf Port 80. Zusätzlich werden alle Fehler und Debug-Ausgaben ausgegeben.

Und schon kann es losgehen: Einfach das Script mit `sudo python test.py` starten. Danach einen Browser öffnen und die Adresse `http://127.0.0.1/` eingeben. Jetzt sollte ein "Hello World!" im Browser stehen.

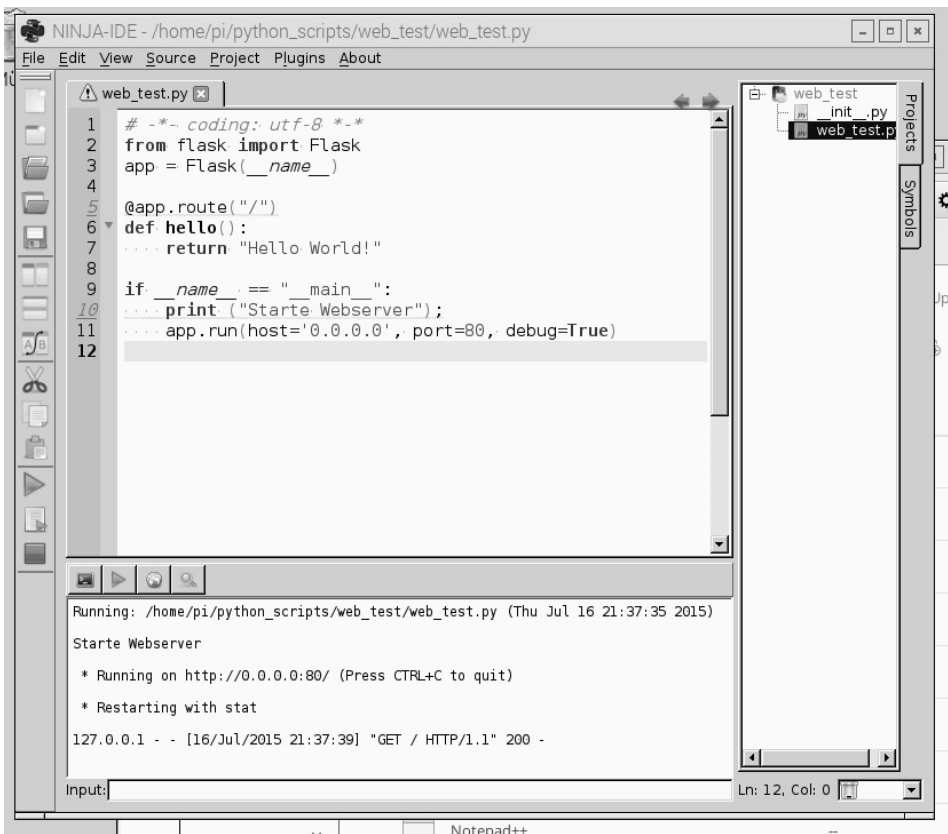


Bild 4.28: Ninja-IDE mit dem Hello-World-Testprogramm

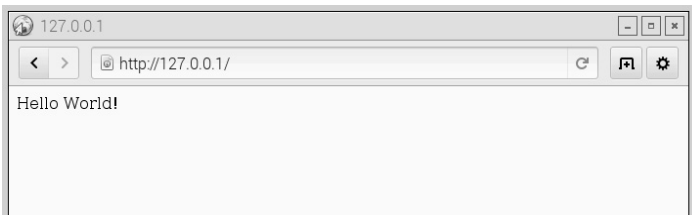


Bild 4.29:
Hello-World-
Browsersausgabe

4.6.2 Benutzung von WebTemplates

Ganze Webseiten mit einem Python-Script zu erzeugen ist sehr aufwendig. Deswegen ist im Flask-Framework auch die Template-Engine Jinja2 (siehe Kapitel 7, Internetseite 27) enthalten. Mit dieser Engine lassen sich Vorlagen von Webseiten mit Inhalten verknüpfen. Z. B. erstellt man sich eine Webseite zur Ausgabe der Temperatur eines Sensors. Die Webseite soll folgendermaßen aussehen:

Datei `index.html`:

```
001 <!DOCTYPE html>
002 <html>
003 <head>
004 <meta charset="ISO-8859-1">
005 <title>Mein Raspberry: {{ title }}</title>
006 </head>
007 <body>
008 Die Temperatur um {{ time }} beträgt {{ temperatur }} °C.
009 </body>
010 </html>
```

Diese Datei wird in ein Unterverzeichnis mit dem Namen »templates« unterhalb unserer Python-Datei gestellt. Die Python-Datei wird wie folgt geändert:

```
001 from flask import Flask, render_template
002 import datetime
003 app = Flask(__name__)
004
005 def getTemperatur():
006     ... # hier Temperatur besorgen und zurückgeben
007
008 @app.route("/")
009 def temp():
010     now = datetime.datetime.now();
011     strTime = now.strftime("%H:%M %d.%m.%Y");
012     temperatur = getTemperatur();
013     strTemperatur = '{:02.2f}'.format(temperatur);
014     templateData = {
015         'title': 'Tempsensor 1',
```

```
016     'temperatur' : strTemperatur,  
017     'time': strTime  
018     };  
019     return render_template('index.html', **templateData);  
020  
021 if __name__ == "__main__":  
022     app.run(host='0.0.0.0', port=80, debug=True);
```

Vieles ist schon bekannt. Die Bibliothek `datetime` enthält Datums- und Zeitmethoden. Da das aktuelle Datum und die Uhrzeit ausgegeben werden sollen, wird diese Bibliothek benötigt.

`getTemperatur()` ist die Funktion, in der der Temperatursensor angesprochen wird. Wie das funktioniert, ist in den vorherigen Kapiteln bereits behandelt worden.

In der Hauptmethode `temp()` werden zunächst das aktuelle Datum und die Uhrzeit geholt und entsprechend formatiert. Dann wird die Temperatur besorgt und auch entsprechend formatiert (hier mit zwei Vor- und zwei Nachkommastellen).

Es wird eine Struktur (Record) mit den verschiedenen Werten zusammengebaut und mithilfe der Renderengine zu einer HTML-Seite verarbeitet. Dabei werden die Platzhalter in der HTML-Datei (gekennzeichnet durch das `{...}`) durch die entsprechenden Texte im Record ersetzt.

Das Ergebnis sieht dann so aus:

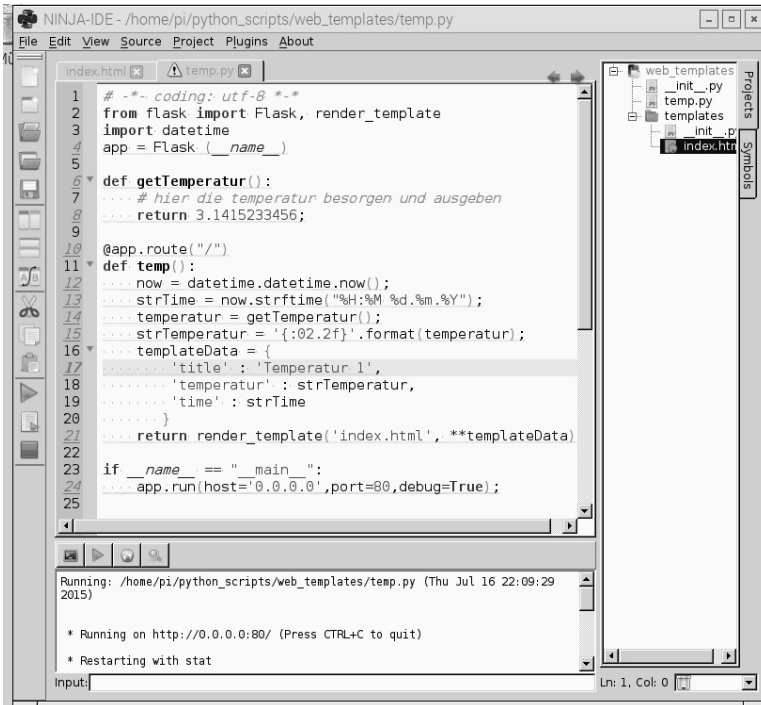


Bild 4.30: Ninja-IDE mit dem Web-Temperatur-Programm

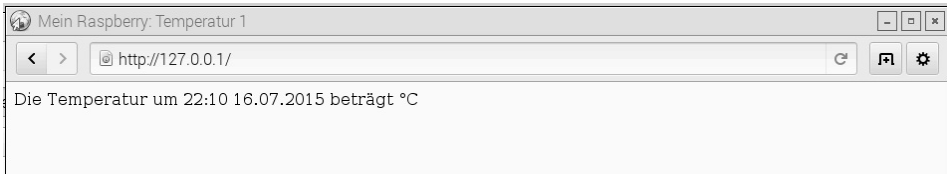


Bild 4.31: Browserausgabe Web-Temperatur

4.7 Raspberry Pi und Arduino™, zwei Welten verbunden

Bereits im Kapitel 4.1.2 wurde eine mögliche Kopplung von Arduino und Raspberry Pi beschrieben. Dabei wurde die serielle Schnittstelle verwendet. Auch mittels einer Funkstrecke (Kapitel 3.2.9 und 4.2.5) kann eine Kopplung beider Systeme erfolgen. Diese Kopplung wird z.B. dann benötigt, wenn der Raspberry Pi den Echtzeitanforderungen einer Schnittstelle nicht mehr genügt. Das Linux-Betriebssystem ist eben kein Echtzeitbetriebssystem, so kann es sein, dass mitten in der Routine zur Aktualisierung einer WS2812B-Kette ein höher priorisierter Systemtask dazwischen funkt und das

zeitkritische Protokoll stört. Das macht sich durch Flackern der LED-Kette bemerkbar. Natürlich kann man um diese Probleme herum programmieren. Im Falle der WS2812B wird per DMA und PWM eine vom Betriebssystem ungestörte Steuerung ermöglicht. Manchmal ist das aber nicht möglich. Mithilfe eines Arduinos, der die Echtzeitsteuerung übernimmt, kann man dann das Update und die Schnittstellenkoordination dem Arduino überlassen. Die Kontrolle aber kann weiterhin der Raspberry Pi behalten. Exemplarisch für eine solche Schnittstelle sei hier das bereits im Kapitel 4.1.2 gezeigte Beispiel genannt.

Wie die Kopplung tatsächlich aussieht, kann je nach Anwendungsfall variieren: seriell oder SPI oder eventuell per Funk. Die Grundlagen dazu sind in den vorherigen Kapiteln beschrieben worden. Nun gilt es nur noch, neben der hardware technischen Anbindung ein entsprechendes Software-Protokoll zu finden oder selber zu entwickeln.

Stichwortverzeichnis

Numerisch

1-Wire 34, 145, 152, 200
 Datenleitung 34
 DS18B20 146
 Protokoll 34
 Temperaturmessung 145
 Übertragungsmodi 35
1-Wire-Overdrive-Modus 35
24C64 138
5-x-7-DOT-Matrix-LED 106
74HC165 84
74HC595 84, 87
74HCT125 206
8-Digit-LED-Driver 109, 178

A

A/D-Modus 92
A/D-Wandler 63, 90, 126, 190
Adafruit-NeoPixel-Ring 81
Aktoren 47
AltSoftLibrary 50
Arduino 11, 49
 1-Wire 13, 145
 A/D-Wandler 63
 Arduino Leonardo 11
 Arduino Mega 11
 Arduino Uno 11
 CAN-Bus 147
 Debugging 224
 DMX 75
 DMX-Master 78
 DMX-Shield 76

DMX-Slave 81
Entwicklungsumgebung 217, 220
GPS-Modul 70
I2C 13, 120
IDE-Upload 223
Internetverbindung 12
Messwertaufzeichnung 62
Minimales Mikrocontrollerboard 158
Porteinstellung 222
Projektinstallation 217
Raspberry-Pi-Verbindung 164, 214
Rollladen-Steuerung 54, 56
Schnittstellen 12
Schnittstellenanordnung 13
Serielle Schnittstelle 13
Serielle Schnittstelle API 50
Sketch 224
Sketchverzeichnis 218
SPI 13, 84
SPI-Hardware 87
Treiberinstallation 219
USB 13
Windows 219
Arduino Uno
 2. serielle Schnittstelle 71
Arduino-IDE 217, 220
ATmega 11
Atmel 11, 32

ATmega256 11
ATmega328 11, 158, 159
ATmega32U16 11

B

Baudrate 21
Beschleunigungssensor 132, 193
BitBanging-Verfahren 85
Blink 220
BMP085 143, 198, 199
BMP180 143, 198
BMP180-Modul 143

C

CAN-Bus 41, 147
 ACK 43
 CANopen 45
 Carrier Sense Multiple Access/Collision Resolution 42
 CSMA/CR-Verfahren 42
 Daten-Frame 42
 Elektrisch 44
 Error-Frame 43
 Highspeed-CAN 44
 Lowspeed-CAN 45
 NMEA 2000 46
 Overload-Frame 44
 Protokoll 42
 Remote-Frame 43
 Übertragungsraten 45
CANopen 45
Centronics 17
Clock-Stretching 33

Controller Area Network
41

Counter-Modus 92

D

D/A-Wandler 129, 191

Debian-Linux 161

Debugging 224

Delta-Komprimierung 65

DMX 23, 75

DMX512-A 23

Master 77

Protokoll 24

RDM-Modus 75

RDM-Protokoll 24

Slave 78

DS1307 120

DS1621 141, 197

DS18B20 145, 200, 201

DS3231 120, 188

DS3231-Modul 121

E

EA DOGM128 98

EA DOG-Serie 98

Echtzeit-Uhrenbaustein
120, 188

Echtzeituhren-Modul 189

EEPROM 121, 138

EEPROM M24C64 139

F

Flask 210

Funkstrecke 111, 182, 214

G

Geräteklassen 38

GPS-Modul 70, 165

Grafik-LCD 98, 175

Gyrosensor 132

Gyroskop 132, 193

H

Hardware 217, 231

HC-SR04 150, 203

Anschlüsse 151, 203

HD44780 97

Highspeed-CAN 44

HMC5883 195

HMC5883L 136, 195

HTerm 52, 62

Human Interface Device 12

I

I2C 32, 187

A/D-Wandler 126, 127

Adressierung 33

BMP085 143

Clock-Stretching 33

D/A-Wandler 129

DS1621 141

Echtzeit-Uhrenbaustein
120, 188

EEPROM 138, 139

Elektrisch 32

Gyroskop und
Beschleunigungs-
sensor 132

LM75 141

Luftdruckmessung 143

Magnetometer 136

MPU6050 133

Protokoll 33

Spezifikation 32

Systemtakt 33

Takt und Zustände 33

Temperaturmessung
141

Idle 227

IEEE-488 17

Internet 210

Inverter 207

ISP-Anschluss 159

ISP-Programmer 159

K

KNX 46

Nachteile 47

SIM-KNX 48

Struktur 47

Vorteile 47

KNX-Busankoppler 232

L

Laufängenkodierung 63

LCD-Backpack 231

LC-Display 97

LED-Anzeigen 109

LedControl-Bibliothek 110

LED-Display 231

LED-Matrix 111

LEDs

Kaskadierung 152

Levelshifter 190, 206

74HCT125 206

NPN-Transistor 206

LM75 141, 197, 198

Lowspeed-CAN 45

LSB 18

Luftdruckmessung 143,
198

M

Magnetfeldsensoren 136,
195

Magnetometer 136, 195

MAX127 126, 190

MAX6952 106

MAX7219 109

MAX7221 109, 178

MCP23S10 231

MCP23S17 93, 173

MCP2515 148

MCP4725 129, 191

Fast-Modus 130

Normaler Modus 130

Mikrocontrollerboard 158

- minicom 167
- Modbus 25
 - Modbus ASCII 26
 - Modbus RTU 25
 - Modbus/TCP 27
- MPU6050 132, 193
 - Beschleunigungs-
umrechnung 135
 - Gyrosensorumrechnung
135
- MPU6050-Modul 133
- MSB 18
- N**
- NeoPixel Jewel 154, 231
- NeoPixel-Ring 121, 126, 231
- Ninja-IDE 228, 229
- NMEA 0183 28, 165
 - GPS-Modul 70, 165
 - optischer Eingang 70
 - physikalische
Schnittstelle 70
 - Protokoll 66
 - Protokolldaten 28
- NMEA 2000 46
- NMEA-Protokoll 66
- NPN-Transistor 206
- nRF24L01 111, 112, 182
- nRF24L01-Module 112, 183
- O**
- One-Wire Siehe 1-Wire
- OTG 161
- P**
- Parallele Schnittstelle 17
- Paritätsbit 21
- Pegeldifferenz 23
- PiTFT 175, 176, 178, 232
- PROFIBUS 27
 - elektrischer 28
 - PROFIBUS DP 27
 - PROFIBUS FMS 27
 - PROFIBUS PA 27
- PuTTY 162
- Python 161, 226
 - Entwicklungssystem
227
 - Python-WebServer 210
- Q**
- quick2wire 196
- R**
- Raspberry Pi 14, 161
 - 1-Wire 200
 - Arduino-Verbindung
164, 214
 - BMP085 199
 - DS18B20 201
 - GPIO 14
 - GPIO-Port 162
 - GPS-Modul 166
 - HC-SR04 204
 - I2C 187
 - I2C-Schnittstelle 187
 - Internet 210
 - LM75 198
 - MAX127 190
 - MCP4725 191
 - MPU6050 193
 - NMEA 0183 165
 - nRF24L01 183
 - PiTFT 175
 - Programmierungsumgebung
226
 - Raspberry Pi 2 Model B
14
 - Serielle Kommunikation
161
 - SPI 169, 171, 173
 - SPI-Schnittstelle 169
 - SPI-Treiber 170
 - Steuerung von Aufgaben
164
 - USB 161
 - USB/Seriell-Adapter
162
 - WS2812B 206
- Raspbian 161
- Remote Device
Management 75
- RF24-Bibliothek 112
- RGB-LED 152, 206
- RGB-LED-Kette 81
- RGB-LEDs 164
- RGB-LED-Scheinwerfer 75
- RLE 63
- Rollladen-Steuerung 54,
56
- RS232 19, 70
 - elektrisch 19
 - NRZ 19
 - Protokoll 21
- RS422 22, 70
- RS485 22
 - Tokenverfahren 22
- RTC 188
- Runtime Length Encoding
63
- S**
- SATA 17
- Schieberegister 171
- Schnittstellenkonvertierung
70
- SCSI 17
- SDHC-Karte 116
- SD-Karte 116
- Sensoren 47, 60
 - barometrische 143
 - Luftdrucksensor 198
 - Magnetfeldsensoren
195
 - Temperatursensor 145,
201
- Serielle Schnittstelle
Universal asynchronous
receiver transmitter
18

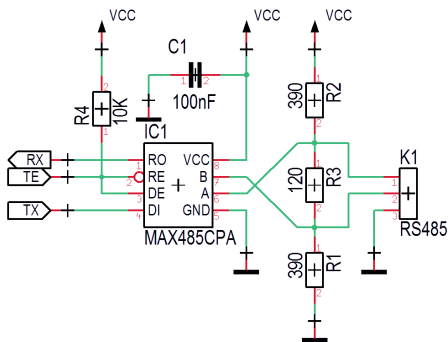
- Universal synchronous asynchronous receiver transmitter 18
- Serielle Schnittstelle 17
 - Asynchron 18
 - DMX Siehe DMX
 - Modbus Siehe Modbus
 - NMEA 0183 Siehe NMEA 0183
 - PROFIBUS Siehe PROFIBUS
 - RS232 Siehe RS232
 - RS485 Siehe RS485
 - Synchron 18
 - UART Siehe UART
 - USART Siehe Modbus
- Sieben-Segment-LED-Display 89, 231
- SIM-KNX 48
- Sketch 224
- Sketchverzeichnis 218
- SkypeLight 154
- SlaveSelect 100
- Software 217
- SPI 29, 84, 169
 - 16-Bit-Porterweiterung 93, 173
 - 5x7-DOT-Matrix-LED 106
 - 8-Digit-LED-Driver 109, 178
 - Ansteuerung Grafik-LCD 175
 - Bibliothek 87
 - CPHA 31
 - CPOL 31
 - CS 30
 - Funkstrecke 111
 - Grafik-LCD 98
 - Hardware 87
 - LC-Display 97
 - MCP23S17 93
 - MISO 30
 - MOSI 30
 - Porterweiterung 84, 93
 - Protokoll 30
 - Schieberegister 84, 171
 - SCK 30
 - SCLK 30
 - SDHC-Karten 116
 - SDI 30
 - SD-Karte 117
 - SD-Karten 116
 - SDO 30
 - Sieben-Segment-LED-Display 89
 - SS 30
 - SPI-Porterweiterung 231
 - Steuerung von Aufgaben 164
- T**
- Temperaturmessung 141, 145, 197, 200
- Temperatursensor 145, 201
- Template-Engine 212
- Terminalprogramm 162, 163
- Think Bowl 196
- Touchdisplay 98
- Touchpanel 104
- Touchscreen 175
- Treiberinstallation 219
- Tresorsteuerung 110
- TWI 32
- Typ-C-Steckverbindung 37
- U**
- UART 18, 19
- Ultraschallentfernungsmessung 150, 203
- USART 18
- USB 36, 49
 - Bulk-Transfer 40
 - Control-Transfer 40
 - Datenübertragungsraten 41
 - Endpunkt/Endpunkt-Kommunikation 39
 - Geräteklassen 38, 39
 - Interrupt-Transfer 40
 - Isochroner Transfer 39
 - Kommunikation mit dem Host 52, 162
 - Messwertaufzeichnung 62
 - Steuerung von Aufgaben 54, 164
 - Treiber 49
 - Übertragungsarten 39
 - USB 3.1 Profile 37
 - Verwendung 41
- USB 3.1 37
 - Typ-C-Steckverbindung 37
- USB On-The-Go 161
- USB/Seriell-Adapter 162
- USB/Seriell-Bridge 162
- USB-Slave 161
- W**
- Web-Temperatur-Programm 214
- WebTemplates 212
- WS2812B 121, 152, 164, 206
- X**
- X Desktop 177, 178



WILFRIED KLAAS

BUSSYSTEME IN DER PRAXIS

Geringe Anschaffungskosten, eine große Community und gut zugängliche Schnittstellen sind die Erfolgsfaktoren von Arduino™ und Raspberry Pi. Ob Sensoren, Displays oder andere Hardware – die genaue Kenntnis von Schnittstellen wie SPI und I²C ist für erfolgreiche Projekte unumgänglich. Egal ob Sie Arduino™ oder Raspberry Pi bevorzugen, in diesem Buch lernen Sie die Funktionsweise dieser Schnittstellen im Detail kennen und nutzen sie in Praxisprojekten für beide wichtigen Plattformen. Für den schnellen Einstieg steht der komplette Quellcode zum Download bereit.



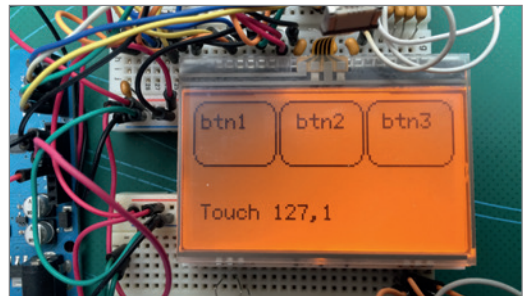
Alle wichtigen Details zu den Schnittstellen werden erklärt: Aufbau, Funktionsweise und Programmierung.

Projektpraxis für Arduino™ und Raspberry Pi

Mit Stromlaufplänen, Blockdiagrammen und Protokollbeschreibungen werden die wichtigsten Schnittstellen für eigene Maker-Projekte beschrieben. Damit Sie das Wissen auch direkt für Arduino und Raspberry Pi anwenden können, stellt Ihnen Klaas zahlreiche nachvollziehbare Praxisprojekte mit Schaltplan und Quellcode vor: Der Anschluss von Sensoren über I²C, die Ansteuerung von Displays über SPI oder die Ansteuerung eines GPS-Moduls über die serielle Schnittstelle sind nur einige Beispiele aus dem großen Projektteil. In einem extra Kapitel stellt Ihnen Klaas außerdem die Programmierumgebungen für Arduino™ und Raspberry Pi vor.

Aus dem Inhalt:

- Schnittstellen des Arduino™ und des Raspberry Pi
- Bussysteme: Serielle Schnittstelle, SPI, I²C, 1-Wire, USB, CAN und KNX
- SB-Kommunikation mit dem Arduino™ und dem Raspberry Pi
- DMX mit dem Arduino™
- Schieberegister, Sieben-Segment-Display, Grafik-LCD und Funkverbindung über SPI
- Gyroskop, Beschleunigungssensor, Temperatur- und Luftdruckmessung über I²C
- Temperaturmessung mittels 1-Wire
- Arduino™ und Raspberry Pi miteinander verbinden
- Arduino™ und Raspberry Pi programmieren



Projekte für den Arduino™ und den Raspberry Pi verdeutlichen die Nutzung der einzelnen Schnittstellen.

Der komplette Quellcode aus dem Buch auf www.buch.ch

Besuchen Sie unsere Website www.franzis.de



FRANZIS