

Der Arduino reicht mir aus: Microcontroller statt Microprocessor

Mattias Schlenker

28. Juni 2014

Inhalt

1 Arduino vs. RaspberryPi

- Technische Daten
- Einsatzbereiche

2 Die Arduino-Zukunft

- „Starke“ Arduinos mit Linux
- Neue Microcontroller mit ARM-Kern
- Problem Fragmentierung

3 Gemeinsam stark!

- Raspberry Pi mit Arduino huckepack
- Raspberry Pi als Zentrale

4 Fazit

Worin unterscheiden sich die beiden Bastlerplatinen?

Auf den ersten Blick bietet der Raspberry Pi mehr als 1000 mal soviel Rechenleistung wie ein Arduino Uno für zehn Euro Aufpreis.
Da fällt die Entscheidung leicht, oder?
Sehen wir uns die Unterschiede doch einmal näher an:

Speicher und Rechenleistung

Plattform	Arduino	Raspberry Pi	Faktor
RAM/Variablenspeicher	2048	536870912	262144

Speicher und Rechenleistung

Plattform	Arduino	Raspberry Pi	Faktor
RAM/Variablenspeicher	2048	536870912	262144
Festspeicher	32768	8589934592	262144

Speicher und Rechenleistung

Plattform	Arduino	Raspberry Pi	Faktor
RAM/Variablenspeicher	2048	536870912	262144
Festspeicher	32768	8589934592	262144
Wortbreite	8Bit	32Bit	4

Speicher und Rechenleistung

Plattform	Arduino	Raspberry Pi	Faktor
RAM/Variablenspeicher	2048	536870912	262144
Festspeicher	32768	8589934592	262144
Wortbreite	8Bit	32Bit	4
„Rechenleistung“ (MIPS)	16	1000	64

Speicher und Rechenleistung

Plattform	Arduino	Raspberry Pi	Faktor
RAM/Variablenspeicher	2048	536870912	262144
Festspeicher	32768	8589934592	262144
Wortbreite	8Bit	32Bit	4
„Rechenleistung“ (MIPS)	16	1000	64

IO und Buses

Arduino heisst in diesem Kontext Atmega328P, Raspberry Pi meint das Model B...

Plattform	Arduino	Raspberry Pi	Faktor
GPIO digital	17	8	unfair! Äpfel vs Birnen!

IO und Buses

Arduino heisst in diesem Kontext Atmega328P, Raspberry Pi meint das Model B...

Plattform	Arduino	Raspberry Pi	Faktor
GPIO digital	17	8	unfair! Äpfel vs Birnen!
analog	6	0	0

IO und Buses

Arduino heisst in diesem Kontext Atmega328P, Raspberry Pi meint das Model B...

Plattform	Arduino	Raspberry Pi	Faktor
GPIO digital	17	8	unfair! Äpfel vs Birnen!
analog	6	0	0
SPI	1	2	2

IO und Buses

Arduino heisst in diesem Kontext Atmega328P, Raspberry Pi meint das Model B...

Plattform	Arduino	Raspberry Pi	Faktor
GPIO digital	17	8	unfair! Äpfel vs Birnen!
analog	6	0	0
SPI	1	2	2
UART (seriell)	1	1	1

IO und Buses

Arduino heisst in diesem Kontext Atmega328P, Raspberry Pi meint das Model B...

Plattform	Arduino	Raspberry Pi	Faktor
GPIO digital	17	8	unfair! Äpfel vs Birnen!
analog	6	0	0
SPI	1	2	2
UART (seriell)	1	1	1
I2C	1	1	1

IO und Buses

Arduino heisst in diesem Kontext Atmega328P, Raspberry Pi meint das Model B...

Plattform	Arduino	Raspberry Pi	Faktor
GPIO digital	17	8	unfair! Äpfel vs Birnen!
analog	6	0	0
SPI	1	2	2
UART (seriell)	1	1	1
I2C	1	1	1

Interrupts

Arduino heisst in diesem Kontext Atmega328P, Raspberry Pi meint das Model B...

Plattform	Arduino	Raspberry Pi	Faktor
Steigend/Fallend	2	8? Publikum?	?

Interrupts

Arduino heisst in diesem Kontext Atmega328P, Raspberry Pi meint das Model B...

Plattform	Arduino	Raspberry Pi	Faktor
Steigend/Fallend	2	8? Publikum?	?
Wakeup-Timer	2	beliebig viele	?

Interrupts

Arduino heisst in diesem Kontext Atmega328P, Raspberry Pi meint das Model B...

Plattform	Arduino	Raspberry Pi	Faktor
Steigend/Fallend	2	8? Publikum?	?
Wakeup-Timer	2	beliebig viele	?

Leistungsaufnahme

Arduino heisst in diesem Kontext Atmega328P, Raspberry Pi meint das Model B...

Plattform	Arduino	Raspberry Pi	Faktor
ohne Optimierungen	40mA bei 5V	300mA bei 5V	7,5

Leistungsaufnahme

Arduino heisst in diesem Kontext Atmega328P, Raspberry Pi meint das Model B...

Plattform	Arduino	Raspberry Pi	Faktor
ohne Optimierungen	40mA bei 5V	300mA bei 5V	7,5
optimiert	4 μ A bei 3,0V	60mA bei 3,6V	15000

Leistungsaufnahme

Arduino heisst in diesem Kontext Atmega328P, Raspberry Pi meint das Model B...

Plattform	Arduino	Raspberry Pi	Faktor
ohne Optimierungen	40mA bei 5V	300mA bei 5V	7,5
optimiert	4 μ A bei 3,0V	60mA bei 3,6V	15000

- Optimierungen beim Arduino: BOD abschalten, ADC abschalten, Tiefschlaf, Batteriebetrieb mit 2xAAA, Takt auf 8MHz reduziert.
- Optimierungen beim Raspberry Pi: Abschalten nicht benötigter Schnittstellen, Auflöten eines optimierten Spannungswandlers, Akkubetrieb mit 3,6V.

Leistungsaufnahme

Arduino heisst in diesem Kontext Atmega328P, Raspberry Pi meint das Model B...

Plattform	Arduino	Raspberry Pi	Faktor
ohne Optimierungen	40mA bei 5V	300mA bei 5V	7,5
optimiert	4 μ A bei 3,0V	60mA bei 3,6V	15000

- Optimierungen beim Arduino: BOD abschalten, ADC abschalten, Tiefschlaf, Batteriebetrieb mit 2xAAA, Takt auf 8MHz reduziert.
- Optimierungen beim Raspberry Pi: Abschalten nicht benötigter Schnittstellen, Auflöten eines optimierten Spannungswandlers, Akkubetrieb mit 3,6V.

Preise

Auf den ersten Blick liegen Arduino und Raspberry Pi preislich eng beieinander. Auf den zweiten Blick:

- Raspberry Pi: 35 bis 40 €
- Arduino Uno oder Zero: 25 bis 30 €

Preise

Auf den ersten Blick liegen Arduino und Raspberry Pi preislich eng beieinander. Auf den zweiten Blick:

- Raspberry Pi: 35 bis 40 €
- Arduino Uno oder Zero: 25 bis 30 €
- Arduino Pro Mini (Sparkfun oder Watterott): 10 €

Preise

Auf den ersten Blick liegen Arduino und Raspberry Pi preislich eng beieinander. Auf den zweiten Blick:

- Raspberry Pi: 35 bis 40 €
- Arduino Uno oder Zero: 25 bis 30 €
- Arduino Pro Mini (Sparkfun oder Watterott): 10 €
- Arduino Pro Mini (China-Klon): 3 bis 4 €

Preise

Auf den ersten Blick liegen Arduino und Raspberry Pi preislich eng beieinander. Auf den zweiten Blick:

- Raspberry Pi: 35 bis 40 €
- Arduino Uno oder Zero: 25 bis 30 €
- Arduino Pro Mini (Sparkfun oder Watterott): 10 €
- Arduino Pro Mini (China-Klon): 3 bis 4 €
- Atmega328P im DIL28-Gehäuse: 3 bis 5 €

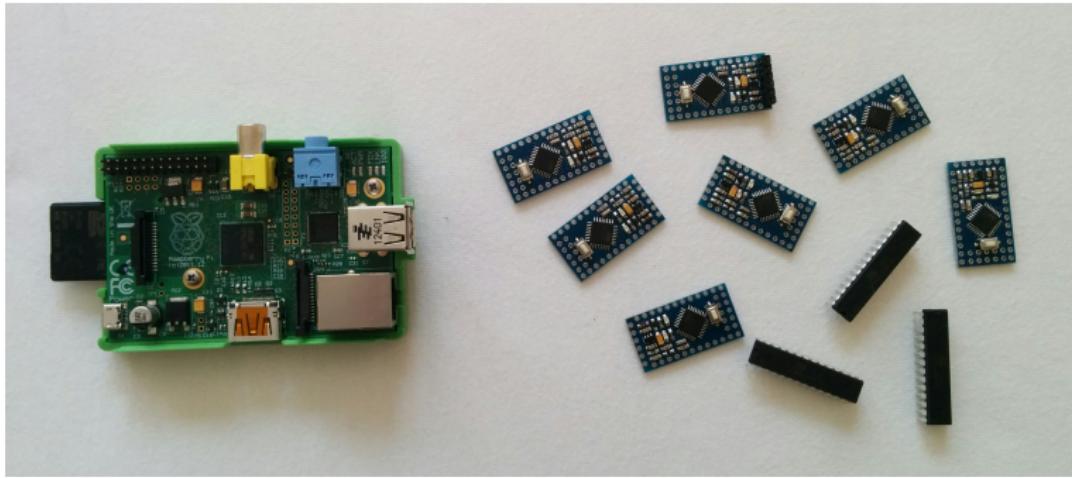
Preise

Auf den ersten Blick liegen Arduino und Raspberry Pi preislich eng beieinander. Auf den zweiten Blick:

- Raspberry Pi: 35 bis 40 €
- Arduino Uno oder Zero: 25 bis 30 €
- Arduino Pro Mini (Sparkfun oder Watterott): 10 €
- Arduino Pro Mini (China-Klon): 3 bis 4 €
- Atmega328P im DIL28-Gehäuse: 3 bis 5 €

Preise visualisiert

Der Raspberry links war teurer als die Arduino-Klone und nackten Atmega328P rechts!



Arduino

- *Energiesparende* Sensoren oder Aktoren: Jahrelanger Betrieb auf zwei AAA-Zellen möglich
- *Billige* Sensoren: Totalverlust ist zu verschmerzen
- *Billige* Peripherie: Der integrierte ADC erlaubt den Anschluss von Thermistoren statt One-Wire-Temperatursensoren
- *Einfache* Sensoren und Aktoren: auch größere Stückzahlen (Sensornetze, Kunstprojekte) lassen sich mit wenigen Bauteilen schnell fertigen
- *Robuste* Softwareentwicklung: Weit weniger Abstraktionsschichten, bessere Vorhersehbarkeit

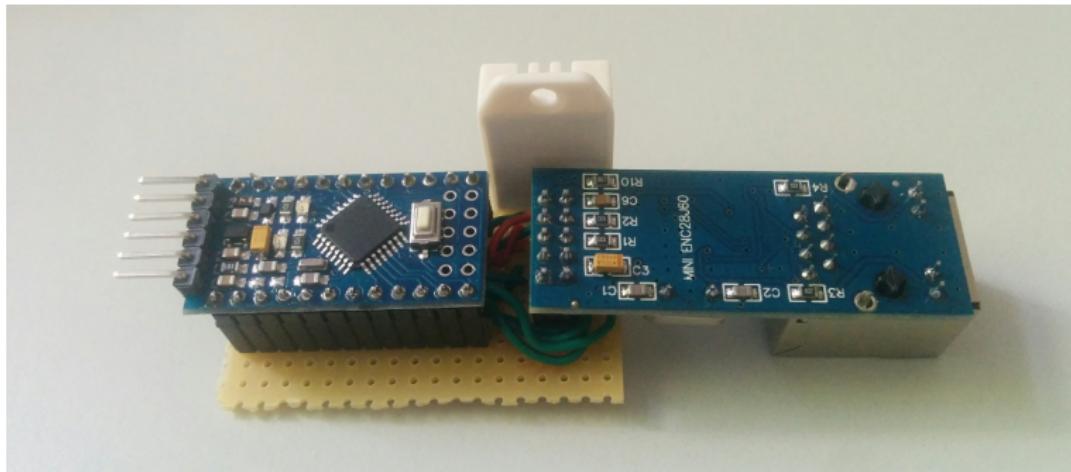
Temperatursensor mit Webserver

Der simpelste aller Webserver: Jedes auf Port 80 eingehende Paket - egal, was drinsteht - wird einfach mit einer Webseite (Temperatur und Luftfeuchte) beantwortet. Weniger als 20k Binärcode gesamt!

- Arduino Pro Mini oder nackter Atmega328P
- Enc28J60 Ethernet (SPI-Anbindung)
- DHT11/22 Sensor
- Gesamtkosten: ca. 12 €

Temperatursensor mit Webserver

So sieht der verlötete Aufbau aus:



Code auf <https://github.com/mschlenker>

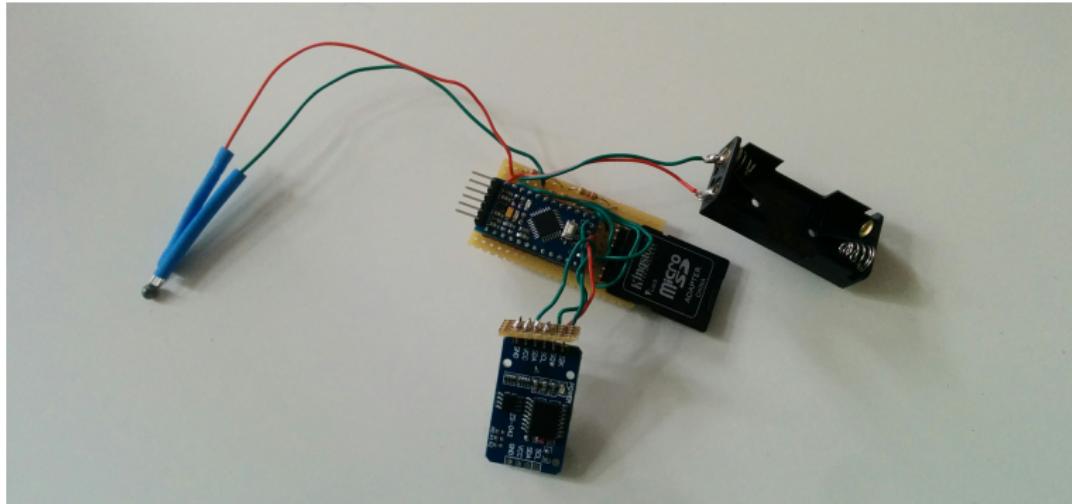
```
21 #include <UIPEthernet.h>
22 #include <dht.h>
23
24 #define DHT22_PIN 6
25
26 const char header[] = "HTTP/1.0 200 OK\r\n";
27   "Content-Type: text/plain\r\n\r\n\r\nLuftfeuchte: ";
28 const byte myMac[6] = { 0x23, 0x23, 0xDE, 0xAD, 0xBE, 0xEF };
29 const IPAddress myIP(192, 168, 2, 5);
30
31 EthernetServer server = EthernetServer(80);
32 dht DHT;
33
34 void setup() {
35   Serial.begin(9600);
36   Ethernet.begin(myMac, myIP);
37   server.begin();
38   Serial.println("\nListening...");
39 }
40
41 void loop() {
42   if (EthernetClient client = server.available()) {
43     Serial.println("\nAnswering...");
44     client.print(header);
45     int ch = DHT.read22(DHT22_PIN);
46     client.print(DHT.humidity, 1);
47     client.print("\r\nTemperatur: ");
48     client.print(DHT.temperature, 1);
49     client.print("\r\n");
50     client.stop();
51     Serial.println("\nDisconnect...");
52   }
53 }
```

Loggender Sensor (Temperatur, Erdfeuchte, Licht)

- Arduino Pro Mini oder nackter Atmega328P
- µSD-Kartenadapter (geschenkt!)
- diverse Analogsensoren (NTC, resistive Erdfeuchte, Photo-Widerstand...)
- hier zusätzlich: RTC
- Gesamtkosten: 5 bis 10 Euro €
- ca. ein Jahr Laufzeit auf 3xAA bei Optimierung

Sensor mit SD-Logging

So sieht der verlötete Aufbau aus:

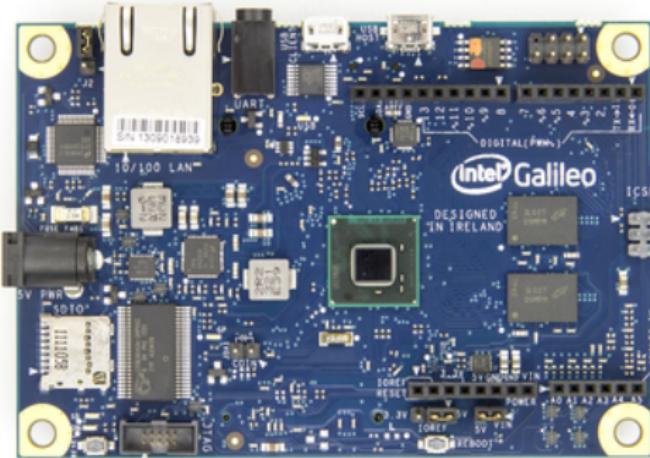


Raspberry Pi

- *Billige Basis für Sensornetze*
- *Stromsparende „Außenstelle“: Solarbetrieb mit relativ kleinem Panel möglich*
- *Große Flexibilität dank „normalem“ Linux*
- *Kleine Server für den Hausgebrauch*
- *Stark genug um bspw. DVB-T oder Webcam zu streamen*
- *Lautloser Thin Client für Windows- oder Linux-Server*

Galileo

Intel verschenkt an Unis Galileo-Bords im Arduino-Shield-Layout.
Das besondere: Statt Microcontroller kommt ein Pentium zum Einsatz:



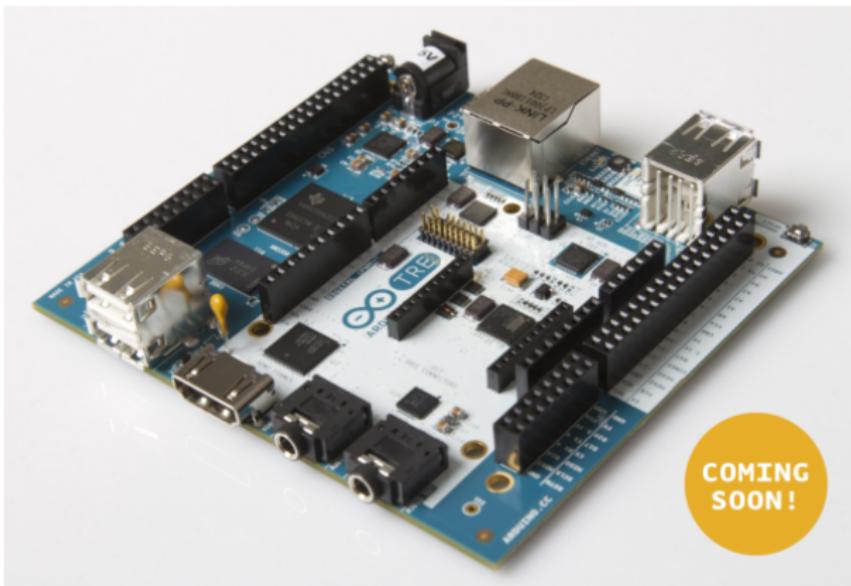
Galileo

Intel verschenkt an Unis Galileo-Boards im Arduino-Shield-Layout.
Das besondere: Statt Microcontroller kommt ein Pentium zum Einsatz:

- Intels Versuch, mit Embedded x86 in die Nähe von Microcontrollern zu kommen
- Referenzboard für Intels Quark SoC x1000
- Arduino-Code wird in ELF-Objekte kompiliert
- recht langsames IO
- *Mattias' Meinung: The worst of both worlds*

Arduino Tre

Gemeinsam mit TI und dem BeagleBone-Projekt entwickelter „janusköpfiger“ Arduino aus TI Sitara und Atmega32u4, Leistung der Linux-Seite höher als beim Raspberry Pi:



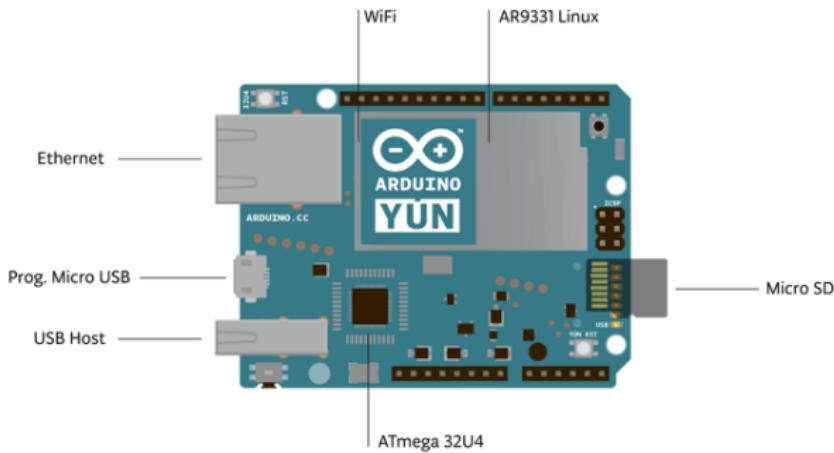
Arduino Tre

Gemeinsam mit TI und dem BeagleBone-Projekt entwickelter „janusköpfiger“ Arduino aus TI Sitara und Atmega32u4, Leistung der Linux-Seite höher als beim Raspberry Pi:

- Von TI als Referenzplattform betrachtet
- 100% Code-Kompatibilität zu Arduino Leonardo
- 100% Code-Kompatibilität zu Beaglebone Black
- recht teuer - Entwicklerkit 149 €- final 60 bis 80 €
- *Mattias' Meinung:* Tolles Konzept, hohe Leistung, aber hoher Preis und große Platine

Arduino Yún

Janusköpfiger Arduino aus MIPS-Prozessor und Atmega32u4,
Formfaktor des Uno, Leistung der Linux-Seite deutlich kleiner als
beim Raspberry Pi, dafür incl. WLAN und schnell angebundenem
Ethernet:



Arduino Yún

Janusköpfiger Arduino aus MIPS-Prozessor und Atmega32u4, Formfaktor des Uno, Leistung der Linux-Seite deutlich kleiner als beim Raspberry Pi, dafür incl. WLAN und schnell angebundenem Ethernet:

- Eigenentwicklung mit Fokus auf guter Nutzbarkeit
- 100% Code-Kompatibilität zu Arduino Leonardo
- OpenWRT basierte Distribution für die MIPS-Seite
- Straßenpreis 60 bis 75 €- kann parallel als WLAN Access Point genutzt werden
- *Mattias' Meinung:* Tolles Konzept, passende Leistung, sollte etwas günstiger sein
- *Der Clou:* Die Software - einfacher geht es nicht!

Arduino Yún - Software-Integration

Eine neue Sammlung von Bibliotheken für die Arduino-IDE abstrahiert die Kommunikation zwischen Microcontroller- und Microprozessorseite:

- Bridge-Bibliothek für Zugriff auf viele Netzwerkfunktionen aus Arduino-Sketches - nur eine Programmiersprache nötig!
- „Mailbox“ (toter Briefkasten) für den Datenaustausch zwischen Microcontroller und Microprocessor
- Integration von Temboo - abstrahiert verschiedene Webdienste
- Voller Paketumfang von OpenWRT
- Einfaches Webinterface für die Einrichtung - Luci bekannt von OpenWRT

Arduino Yún - Twitter-Beispiel

```
void loop() {
    String tweetText("TEST: Der Yun laeuft seit " + String(millis()) + " ms.");
    // Temboo-Objekt initialisieren
    TembooChoreo StatusesUpdateChoreo;
    StatusesUpdateChoreo.begin();
    StatusesUpdateChoreo.setAccountName(TEMBOO_ACCOUNT);
    StatusesUpdateChoreo.setAppKeyName(TEMBOO_APP_KEY_NAME);
    StatusesUpdateChoreo.setAppKey(TEMBOO_APP_KEY);
    // Temboo mit Twitter verknüpfen
    StatusesUpdateChoreo.setChoreo("/Library/Twitter/Tweets/StatusesUpdate");
    StatusesUpdateChoreo.addInput("AccessToken", TWITTER_ACCESS_TOKEN);
    StatusesUpdateChoreo.addInput("AccessTokenSecret", TWITTER_ACCESS_TOKEN_SECRET);
    StatusesUpdateChoreo.addInput("ConsumerKey", TWITTER_CONSUMER_KEY);
    StatusesUpdateChoreo.addInput("ConsumerSecret", TWITTER_CONSUMER_SECRET);
    // Tweet absetzen
    StatusesUpdateChoreo.addInput("StatusUpdate", tweetText);
    // Return Code ermitteln
    unsigned int returnCode = StatusesUpdateChoreo.run();
    if (returnCode == 0) {
        Serial.println(":D Tweet erfolgreich gesendet!");
        StatusesUpdateChoreo.close();
        delay(3600000); // Zehn Stunden warten
    } else {
        Serial.println(":-( Fehler!");
        while (StatusesUpdateChoreo.available()) {
            char c = StatusesUpdateChoreo.read();
            Serial.print(c);
        }
        StatusesUpdateChoreo.close();
        delay(90000); // 90 Sekunden warten
    }
}
```

Arduino Yún - Tweets ganz praktisch



Horst von Forst
@horstvonforst

 Folgen

TEST: Der Yun laeuft seit 8166 ms.

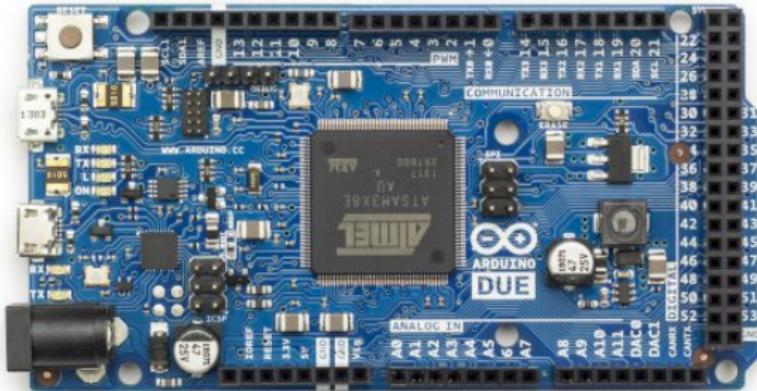
12:56 PM - 9 Dez 2013



Wozu? „Horst von Forst“, unser Weihnachtsbaum twitterte, wenn er zuwenig Wasser hatte. Der vollständige Code und Erklärungen zum verwendeten Sensor unter:
<http://www.arduino-hausautomation.de/>

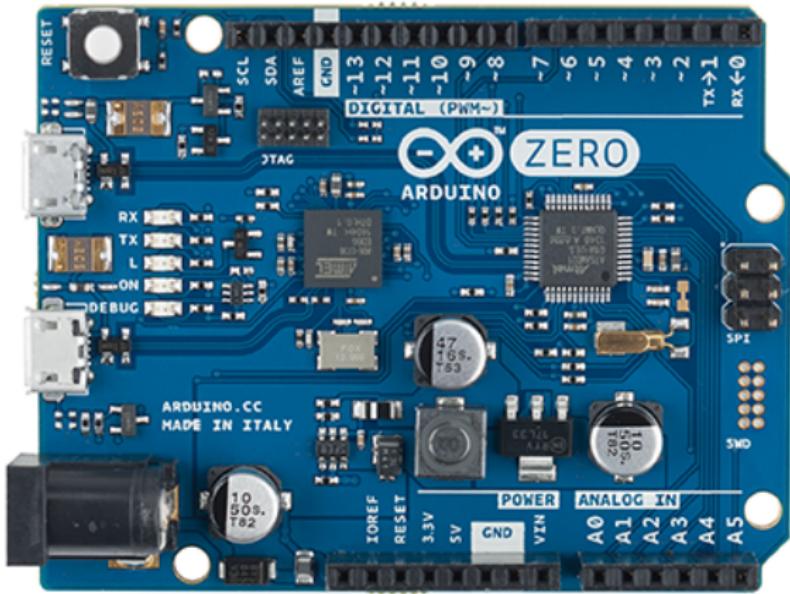
Arduino Due

Erstes Arduino-Board mit 32Bit-ARM-Microcontroller. Ca. 45 €. CAN-Interface. Blick in die Zukunft ohne weitere Relevanz. Daher keine technischen Details:



Arduino Zero

Designierter Nachfolger des Arduino Uno mit SAMD21G18, höhere Integration verspricht günstigere Preise:



Arduino Zero

Designierter Nachfolger des Arduino Uno mit SAMD21G1, höhere Integration verspricht günstigere Preise:

- 32 Bit Microcontroller mit ARM-Kern
- 256kB Flash
- 32kB SRAM
- Hohe Integration mit Debugger und 2xUSB
- Tendenziell Preise auf Level von Atmega32u4 oder Atmega328P - kleine Klone für weniger als 5 € möglich
- **Achtung!** 3,3 Volt
- **Achtung!** kein DIL Package des Controllers erhältlich

Bitte unterbrechen!

Falls Mattias in diesem Abschnitt ausschweift, bitte unterbrechen!

Microcontroller-Familien von Arduino

Im Laufe der Jahre wurden von Arduino immer mehr Microcontroller-Familien unterstützt, nicht nur kleine Sprünge wie von Atmega168 auf Atmega328, sondern mehrere Brüche.
Offizielle Familien:

- Atmega328: Arduino Uno, Pro Mini
- Atmega32u4: Leonardo, Yún, Micro
- Atmega2560: Mega2560
- *neu* ATSAMD21: Arduino Zero, Due
- *neu* Intel Galileo - noch nicht eingepflegt in Arduino IDE

Kompatibilitätsprobleme

Daraus ergeben sich Kompatibilitätsprobleme, beispielsweise wenn Code von einem Microcontroller auf den anderen portiert werden soll:

- Spannungsunterschiede: Mal 3,3V, mal 5V
- Unterschiedliche Position der Pins für SPI
- Code zum extremen Energiesparen ist Controllerspezifisch
- Viele Bibliotheken unterstützen bislang nur 328 und 32u4

Hoffnung?

Kann der ARM-Kern die Fragmentierung stoppen?

- Ablösung der 32u4 basierten Controller durch ARM
- Fokus der Entwickler schnell auf ARM?
- Atmega328P wird noch eine Weile bleiben (DIL-Package!)
- Andere Controllerhersteller werden zu Arduino stoßen
- Billige Klone im Mini-Format machen Verlust der DILs wett
- Arduino wird offener und emanzipiert sich von Atmel

Das Problem

Raspberry Pi ist billig und leistungsstark, aber:

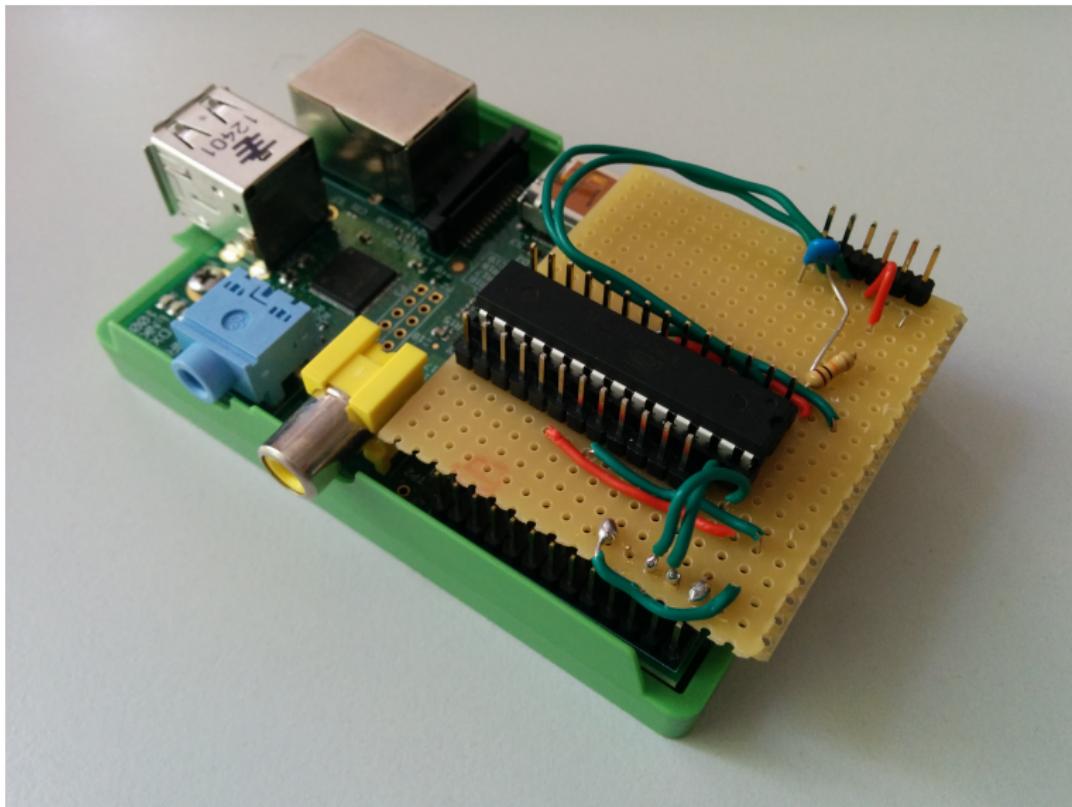
- Viele Bibliotheken nur für Arduino erhältlich
- Hoher Aufwand, Bibliotheken und Protokolle zu portieren
- Keine analogen Ports

Die Lösung

Wir kombinieren einfach Raspberry Pi und Arduino:

Best of both worlds!

- Alle Arduino-Bibliotheken nutzbar
- GPIO des Raspberry bleibt erhalten
- Weniger als 10 Euro Investition



Und wie geht das?

- Nackter Atmega328P huckepack
- Spannungsversorgung durch 3,3V-Schiene des Raspberry
- Kommunikation über I2C-Bus - weitere Bus-Teilnehmer möglich
- **Bonus:** Atmega328P kann direkt vom Raspberry Pi programmiert werden

Beispielcode Arduino

```
#include <Wire.h>
#define SLAVE_ADDRESS 0x03
int number = 0;
int state = 0;

void setup() {
    pinMode(13, OUTPUT);
    Serial.begin(9600);
    Wire.begin(SLAVE_ADDRESS);
    Wire.onReceive(receiveData);
    Wire.onRequest(sendData);
    Serial.println("Los geht's!");
}

void loop() { delay(50); }

void receiveData(int byteCount){
    while(Wire.available()) {
        number = Wire.read();
        Serial.print("data received: ");
        Serial.println(number);
        if (number == 1){
            digitalWrite(13, HIGH);
            state = 1;
        } else {
            digitalWrite(13, LOW);
            state = 0;
        }
    }
}

void sendData(){ Wire.write(number); }
```

Beispielcode Raspberry

```
import smbus
import time
import math
import random

bus = smbus.SMBus(1) # 0 bei Revision 1
address = 0x03      # Adresse des Arduino

def writeToArduino(value):
    bus.write_byte(address, value)
    return -1

def readFromArduino():
    number = bus.read_byte(address)
    return number

while True:
    val = int(random.random() * 5.0)
    writeToArduino(val)
    print "RPI: Hi Arduino, ich schick Dir ", val
    time.sleep(1)
    number = readFromArduino()
    print "Arduino: Hi RPI, ich hab empfangen ", number
    print
```

Vor- und Nachteile

Vorteile

- Billig: Atmega328P gibt es in der Fuzo beim blauen C
- Schnell realisiert
- Sehr flexibel

Nachteile

- Zwei Sprachen involviert: Python und Arduino
- Kein fertiges Protokoll: Handarbeit angesagt
- Nur bei 3,3V-Betrieb bleiben alle Vorteile von I²C erhalten

Kommunikation über's Netz

Arduinos mit Enc28J60-Ethernetmodul kommunizieren mit dem Raspberry Pi:

- UDP-Kommunikation bei Arduino simpel realisierbar
- UDP-Empfang auf Raspberry-Seite nur wenige Zeilen Python
- Billige Hardware am Arduino (3 bis 5 € für Enc28J60)
- Weniger Einschränkungen bei Buslänge ggü. direktem Onewire

Äpfel mit Birnen vergleichen lohnt nicht!

- Verschiedene Tools für verschiedene Zwecke
- Kenntnisse beider Tools helfen, das richtige auszuwählen
- Wer intensiv bastelt, muss kombinieren
- Konvergenz verwirrt gelegentlich - leider auch die Hersteller (siehe Galileo!)
- Kommt rüber auf die andere Seite!

Revolutionkunde

- Arduino: 2004 - die Microcontroller-Revolution
- Raspberry Pi: 2012 - die Microprocessor-Revolution + Popularisierung von Linux

Beides zusammen: Die Maker-Revolution!

Danke für die Aufmerksamkeit

- Mattias Schlenker - ms@mattiasschlenker.de
- Fork me on Github! <https://github.com/mschlenker>
- Webseite <https://www.arduino-hausautomation.de/>
- Google+ <https://plus.google.com/+MattiasSchlenker1>