



# Vernetzte "Dinge" mit Raspberry Pi und Arduino

Marcel Bernet, mc-b

# Haftung



- Bei den Bauanleitungen bzw. irgendwelcher Software gibt es keine Haftung für irgendwelche Schäden oder Funktionsgarantie, bitte immer nur als Anregung auffassen.
- Ich hafte nicht für Schäden, die der Anwender oder Dritte durch die Verwendung der Software oder Hardware verursachen oder erleiden. In keinem Fall hafte ich für entgangenen Umsatz oder Gewinn oder sonstige Vermögensschäden die bei der Verwendung oder durch die Verwendung dieser Programme oder Anleitungen entstehen können.
- Und wer mit Strom umgeht, soll sich bitte bei höheren Spannung und Strömen der Gefahren bewusst sein. Modellbahn gehört VDE-technisch zur Kategorie Spielzeug, dementsprechend streng sind die Vorschriften.
- Siehe hierzu auch die Sicherheitshinweise des Fremo.
  
- Die Schaltungen und die Software werden als Anregung und Hilfe unter Modellbahnhern veröffentlicht. Sie sind auf Grund von Beispielen aus dem Netz bzw. eigenen Ideen entstanden.
- Natürlich sind sinnvolle Anregungen, Fehlermeldungen und Verbesserungen zu den Schaltung immer willkommen. Allerdings möchte ich aus gegebenen Anlass darauf hinweisen, dass ich leider keine Zeit für langwierige Diskussionen des Typs "Ich habe die Schaltung nachgebaut, warum funktioniert sie bei mir nicht" habe. Auch die Frage "Ich habe noch diesen oder jenen Chip in der Schublade, kann ich den auch verwenden", möge sich bitte jeder selbst beantworten.

# Agenda

- Vormittag
  - Einleitung „Internet der Dinge“, Sensoren, Aktoren – 20'
  - Die Arduino Plattform (Hardware, einfache Anwendungen, Sensoren) – 90'
  - Die Raspberry Pi Plattform – 20'
- Nachmittag
  - Arduino's und Raspberry Pi verbinden – 10'
  - Aufbau einer Steuerung anhand zweier konkreter Beispiele (Hausautomation und Modelleisenbahn-Steuerung) – 120'
  - Raspberry Pi, Arduino mit dem Internet verbinden – 30'

# Was haben diese Dinge gemeinsam?

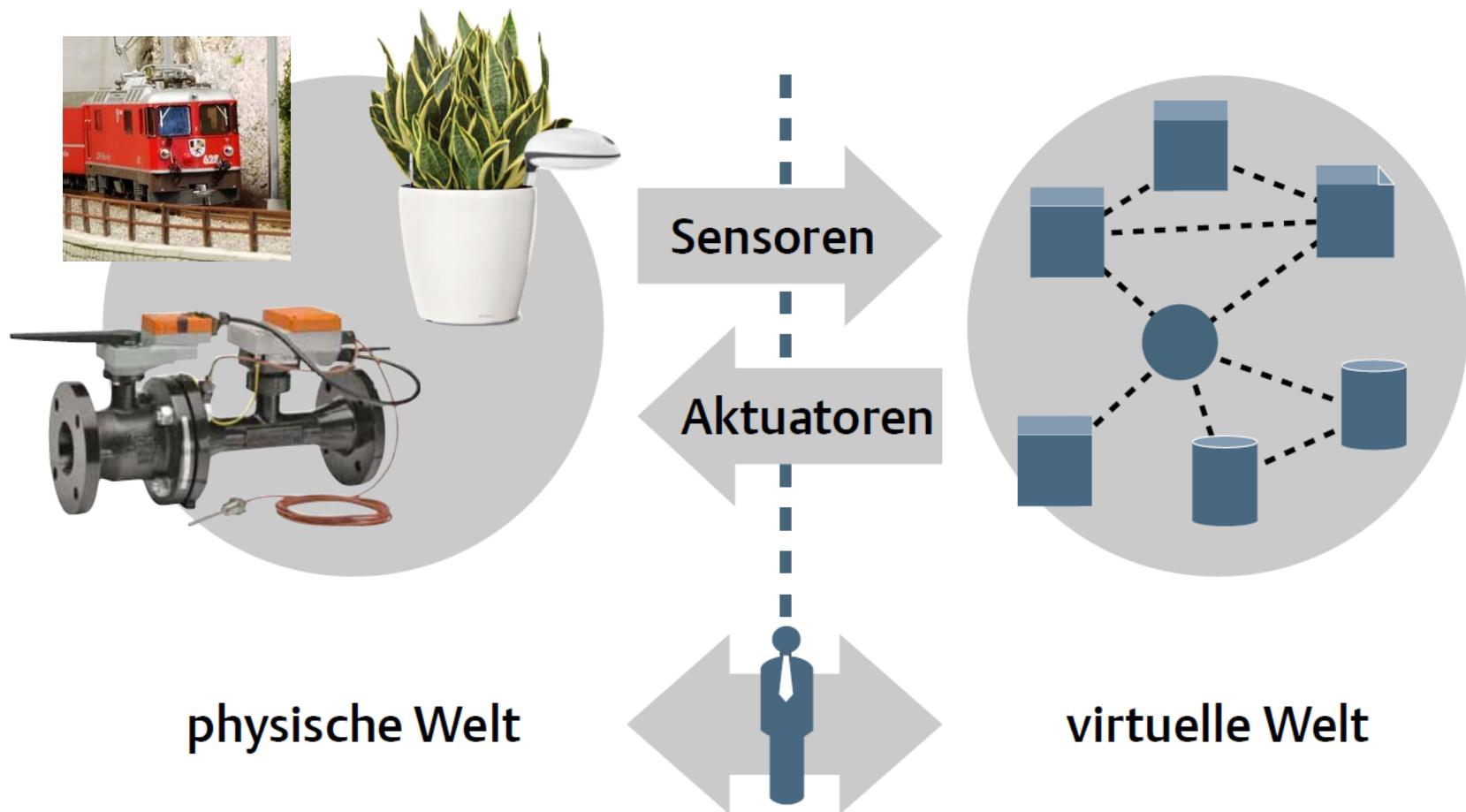


Sie sind mit dem Internet verbunden  
Es sind fertig entwickelte Produkte

Quelle: /ch/open OBL vom 2.5.13 - Internet of Things @ Ergon (<http://www.ch-open.ch/events/obl/oblosl-2013/>)

Copyright (C) Marcel Bernet. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

# Das Internet der Dinge

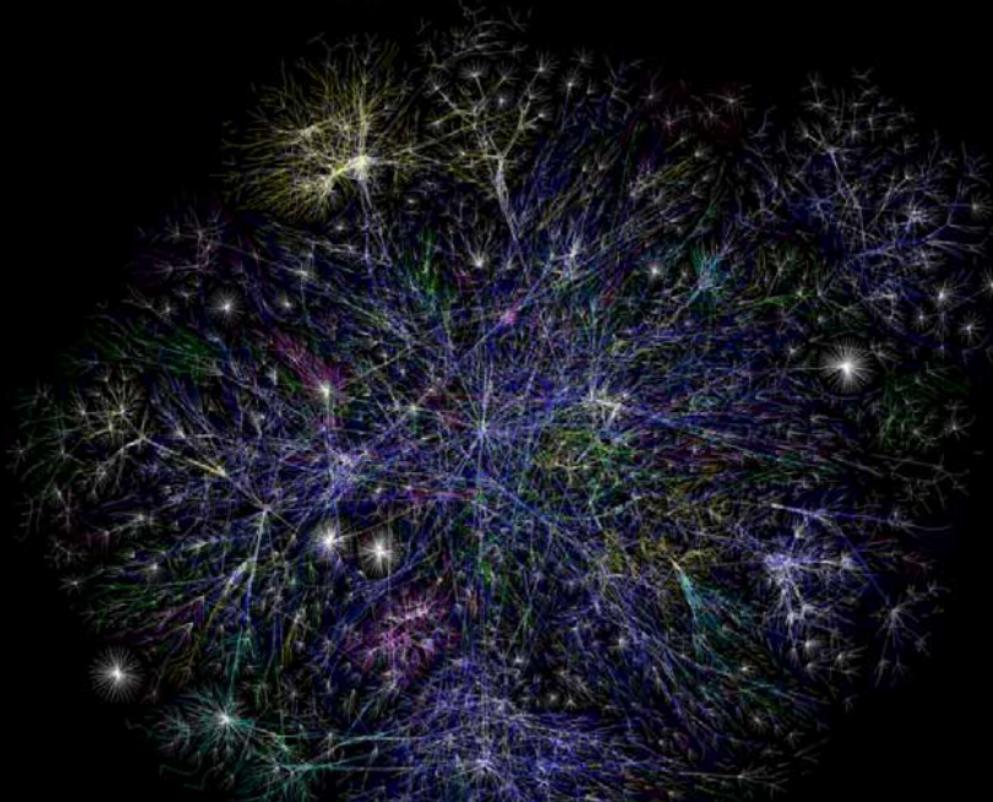


Quelle: /ch/open OBL vom 2.5.13 - Internet of Things @ Ergon (<http://www.ch-open.ch/events/obl/oblosl-2013/>)

Copyright (C) Marcel Bernet. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

*«During 2008, the number of things connected to the Internet exceeded the number of people on earth.»*

[Cisco]



Quelle: /ch/open OBL vom 2.5.13 - Internet of Things @ Ergon (<http://www.ch-open.ch/events/obl/oblosl-2013/>)

Copyright (C) Marcel Bernet. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

# Rasante Entwicklung der Hardware



Fortschritt getrieben durch die Verbreitung von Smartphones



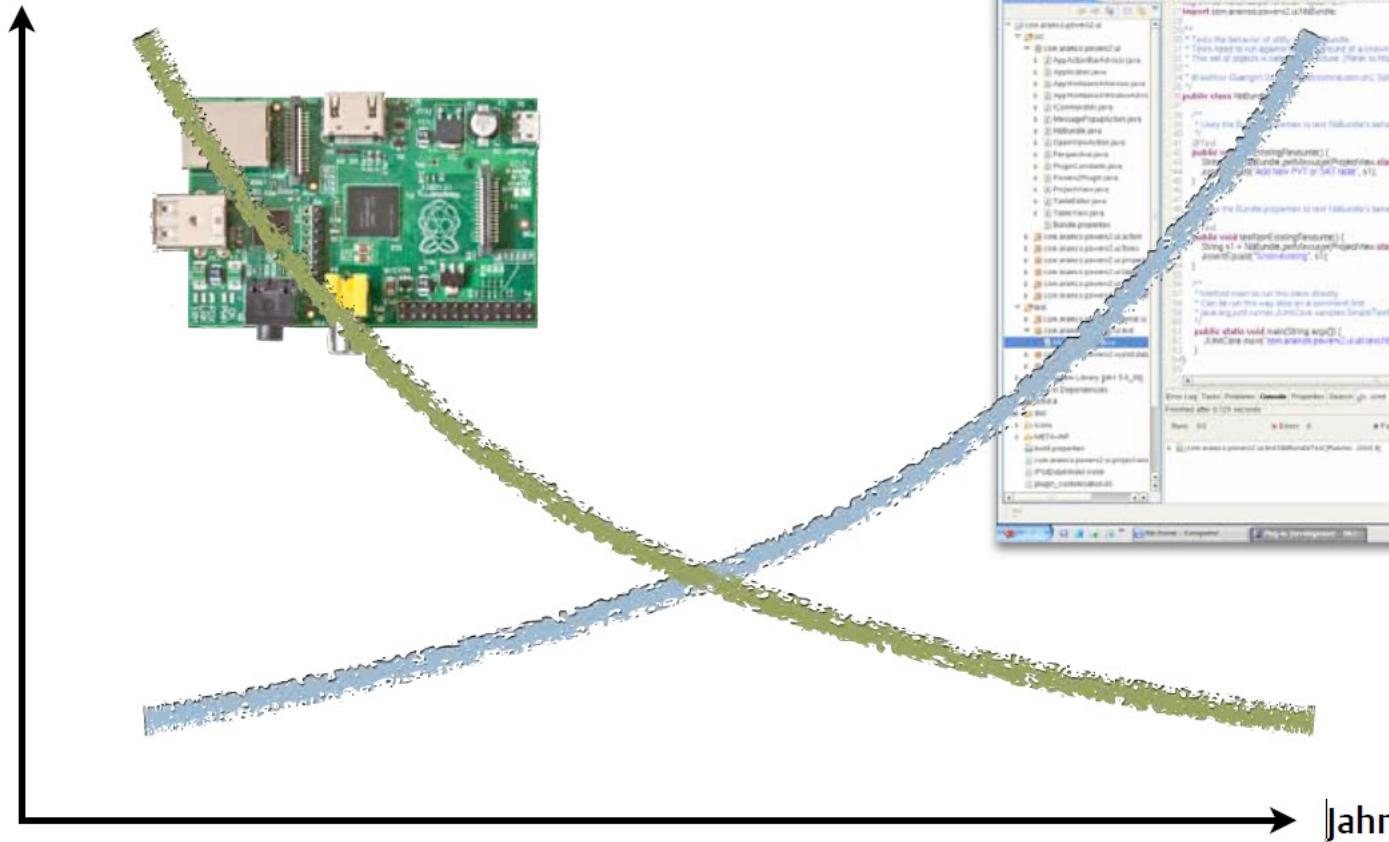
Quelle: /ch/open OBL vom 2.5.13 - Internet of Things @ Ergon (<http://www.ch-open.ch/events/obl/oblosl-2013/>)

Copyright (C) Marcel Bernet. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

# Hardware und Software — Yin und Yan?



Anteil Kosten



Screenshot of an IDE showing Java code for a test class named `com.armeria.protocol.test.BundlesTest`. The code includes annotations like `@Test` and `@TestTemplate`, and methods like `void test()` and `void testBundles()`. The IDE interface shows the project structure on the left and the code editor on the right.

```
public class BundlesTest {
    @Test
    public void test() {
        // ...
    }

    @TestTemplate
    void testBundles() {
        // ...
    }
}
```

Quelle: /ch/open OBL vom 2.5.13 - Internet of Things @ Ergon (<http://www.ch-open.ch/events/obl/oblosl-2013/>)

Copyright (C) Marcel Bernet. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

# 1. Herausforderung: Komplexität



Quelle: /ch/open OBL vom 2.5.13 - Internet of Things @ Ergon (<http://www.ch-open.ch/events/obl/oblosl-2013/>)

Copyright (C) Marcel Bernet. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

## 2. Herausforderung: Qualität



Zuverlässigkeit

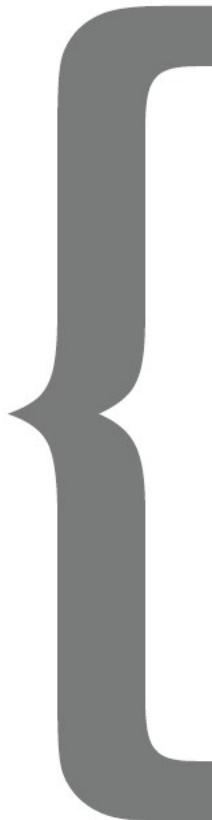
Bedienbarkeit



Quelle: /ch/open OBL vom 2.5.13 - Internet of Things @ Ergon (<http://www.ch-open.ch/events/obl/oblosl-2013/>)

Copyright (C) Marcel Bernet. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

# 3. Herausforderung: Sicherheit



Schutz vor Missbrauch

Nachvollziehbarkeit



Quelle: /ch/open OBL vom 2.5.13 - Internet of Things @ Ergon (<http://www.ch-open.ch/events/obl/oblosl-2013/>)

Copyright (C) Marcel Bernet. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

# Zusammenfassung (aus Wikipedia)



- **Ein Sensor** (von lateinisch sentire, dt. „fühlen“ oder „empfinden“), (Messgrößen-) Aufnehmer oder (Mess-)Fühler **ist ein technisches Bauteil, das bestimmte physikalische oder chemische Eigenschaften** (z. B.: Wärmestrahlung, Temperatur, Feuchtigkeit, Druck, Schall, Helligkeit oder Beschleunigung) und/oder die stoffliche Beschaffenheit seiner Umgebung qualitativ oder **als Messgröße quantitativ erfassen kann**. Diese Größen werden mittels physikalischer oder chemischer Effekte erfasst **und in ein weiterverarbeitbares elektrisches Signal umgeformt**.
- **Aktoren (Wandler; Antriebselemente)**, oft auch wegen des englischen Begriffs „actuator“ als Aktuatoren bezeichnet, **setzen die elektrischen Signale** (z. B. vom Steuerungscomputer ausgehende Befehle) **in mechanische Bewegung** oder andere physikalische Größen (z. B. Druck oder Temperatur) **um** und greifen damit aktiv in das Regelungssystem ein und/oder geben Sollgrößen vor.
- **Das Internet der Dinge** (auch englisch Internet of Things) **beschreibt, dass der (Personal) Computer zunehmend als Gerät verschwinden und durch „intelligente Gegenstände“ ersetzt wird**. Statt – wie derzeit – selbst Gegenstand der menschlichen Aufmerksamkeit zu sein, soll das „Internet der Dinge“ den Menschen bei seinen Tätigkeiten unmerklich unterstützen. Die immer kleineren Computer sollen Menschen unterstützen ohne abzulenken oder überhaupt aufzufallen.

# Agenda



- Vormittag
  - Einleitung „Internet der Dinge“, Sensoren, Aktoren – 20'
  - Die Arduino Plattform (Hardware, einfache Anwendungen, Sensoren) – 90'
  - Die Raspberry Pi Plattform – 20'
- Nachmittag
  - Arduino's und Raspberry Pi verbinden – 10'
  - Aufbau einer Steuerung anhand zweier konkreter Beispiele (Hausautomation und Modelleisenbahn-Steuerung) – 120'
  - Raspberry Pi, Arduino mit dem Internet verbinden – 30'

# Arduino - Übersicht



Arduino Uno



Arduino Leonardo



Arduino Due



Arduino Yún



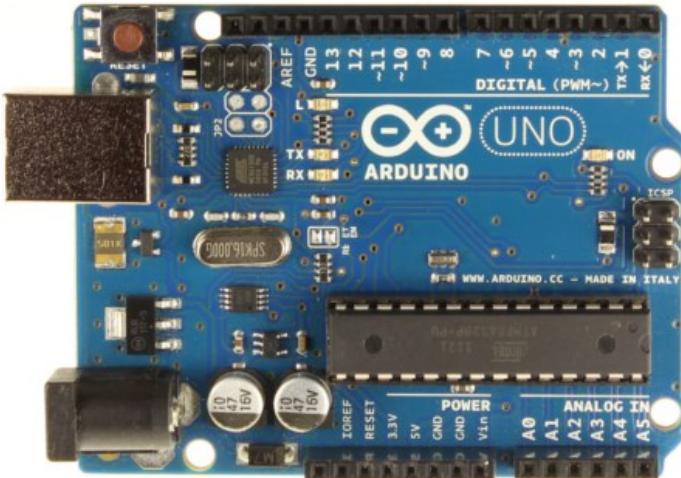
Arduino Robot



Arduino Esplora

- [www.arduino.cc](http://www.arduino.cc)
- Kostengünstig (ab 19\$)
- Cross-Plattform (Windows, Linux, Mac) - Entwicklungsumgebung
- Schnell erlernbare Programmiersprache, basierend auf Wiring (<http://wiring.org.co/>)
- Gesteuert mittels Codestücken, sogenannten **Sketches**.
- Offene und Erweiterbare Software-Libraries auf Basis C++
- Offene und Erweiterbare Hardware. Basierend auf Atmel's Mikrocontrollern.
- Verfügbar in verschiedenen Varianten Duemilanove (2009), Uno, Mega, Due, YÚN

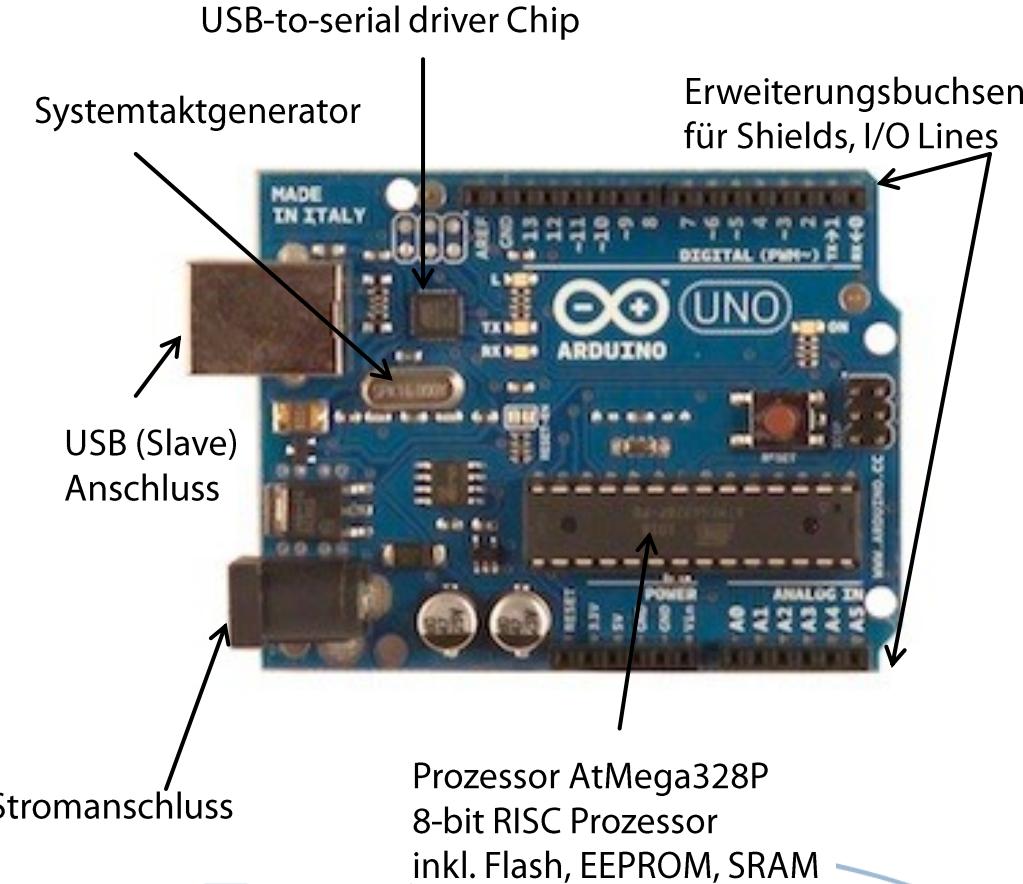
# Arduino Uno Spezifikationen



Microcontroller	ATmega328
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
Analog Input Pins	6
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328) of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328)
EEPROM	1 KB (ATmega328)
Clock Speed	16 MHz

Zum Vergleich: Der erste vollwertige Mikroprozessor von 1971: [http://de.wikipedia.org/wiki/Intel\\_8080](http://de.wikipedia.org/wiki/Intel_8080)

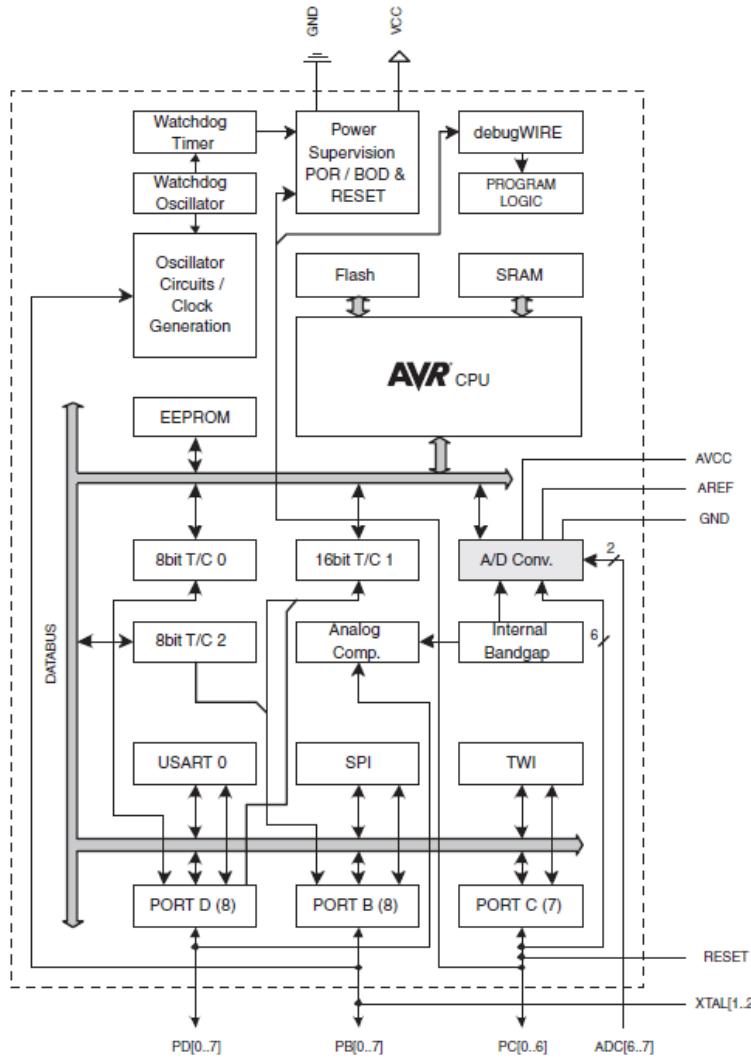
# Arduino - Aufbau



- Die Arduino-Plattform ist eine aus Soft-und Hardware bestehende **Physical-Computing-Plattform**.
- Physical Computing bedeutet im weitesten Sinne, interaktive, physische Systeme durch die Verwendung von Hardware und Software zu erstellen. Diese Systeme reagieren auf Ereignisse (Sensoren) in der realen, analogen Welt und/oder wirken (Aktoren) auf sie ein.

<http://de.wikipedia.org/wiki/Arduino-Plattform>

# Arduino – Prozessor ATmega328P

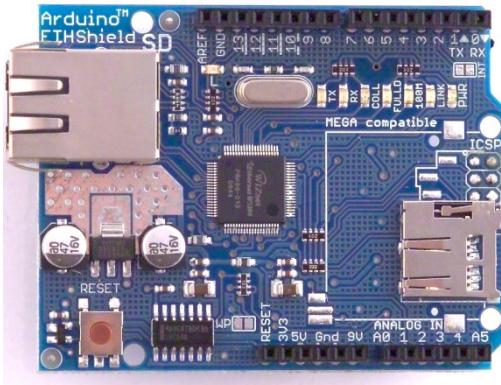


- High Performance, Low Power AVR® 8-Bit Microcontroller
- **Advanced RISC Architecture**
  - 131 Powerful Instructions – Most Single Clock Cycle Execution
  - 32 x 8 General Purpose Working Registers
  - Fully Static Operation
  - Up to 20 MIPS Throughput at 20 MHz
  - On-chip 2-cycle Multiplier
- High Endurance Non-volatile Memory Segments
  - 2K Bytes of In-System Self-Programmable Flash program memory (ATmega48PA/88PA/168PA/328P)
  - **256/512/512/1K Bytes EEPROM (ATmega48PA/88PA/168PA/328P)**
  - **512/1K/2K Bytes Internal SRAM (ATmega48PA/88PA/168PA/328P)**
  - **Write/Erase Cycles: 10,000 Flash/100,000 EEPROM**
  - Data retention: 20 years at 85°C/100 years at 25°C(1)
- **Optional Boot Code Section with Independent Lock Bits**
- In-System Programming by On-chip Boot Program
- True Read-While-Write Operation
- Programming Lock for Software Security
- **Peripheral Features**
  - Two 8-bit Timer/Counters with Separate Prescaler and Compare Mode
  - One 16-bit Timer/Counter with Separate Prescaler, Compare Mode, and Capture
- **Mode**
  - Real Time Counter with Separate Oscillator
- Six PWM Channels
- **I/O and Packages**
  - **23 Programmable I/O Lines**
  - 28-pin PDIP, 32-lead TQFP, 28-pad QFN/MLF and 32-pad QFN/MLF

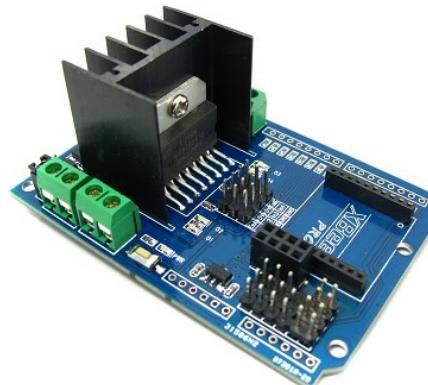
# Shields (<http://arduino.cc/en/Main/Products>)



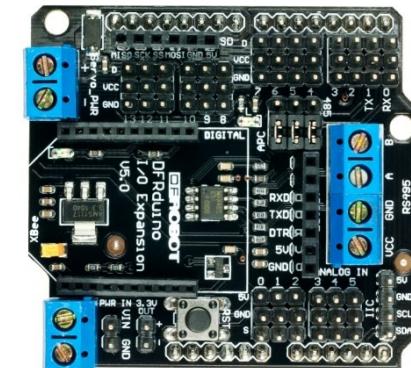
Ethernet-Shield



MotoMama (Motor) Shield



IO Expansion Shield

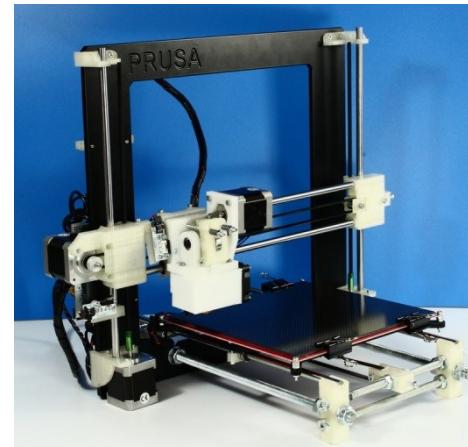


- Mittels Erweiterungsboards, sogenannten Shields, kann die Funktionalität des Board erweitert werden.
- Diese werden einfach in die Buchsenleiste des Arduino Board gesteckt und mittels entsprechender Software angesteuert.
- Es existieren eine Vielzahl von Shields, u.a. für
  - Netzwerkanbindung mittels RJ45 / Ethernet
  - Ansteuerung von Motoren (Lokomotiven).
  - Ein Shield zum Abspielen von Sounddateien (u.a. MP3) z.B. Für Bahnhofs durchsagen
  - Schalten von 220V Verbrauchern
  - IO Expansion Shields für Anschluss von Servo, LED, Rückmelder, I2C, RS 485 etc.

# Arduino - Einsatzgebiete



NinjaBlocks: Hausautomation



RAMPS: Steuerung von 3D Druckern



LiliPad: integriert in Kleidung



Arduino ADK: Steuerung von USB Geräten

# Arduino

+

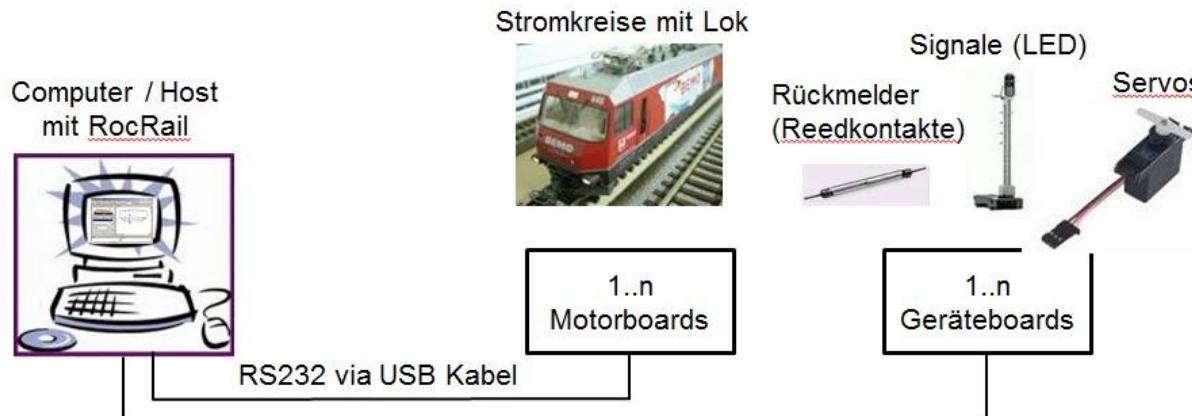
# Modelleisenbahn



+

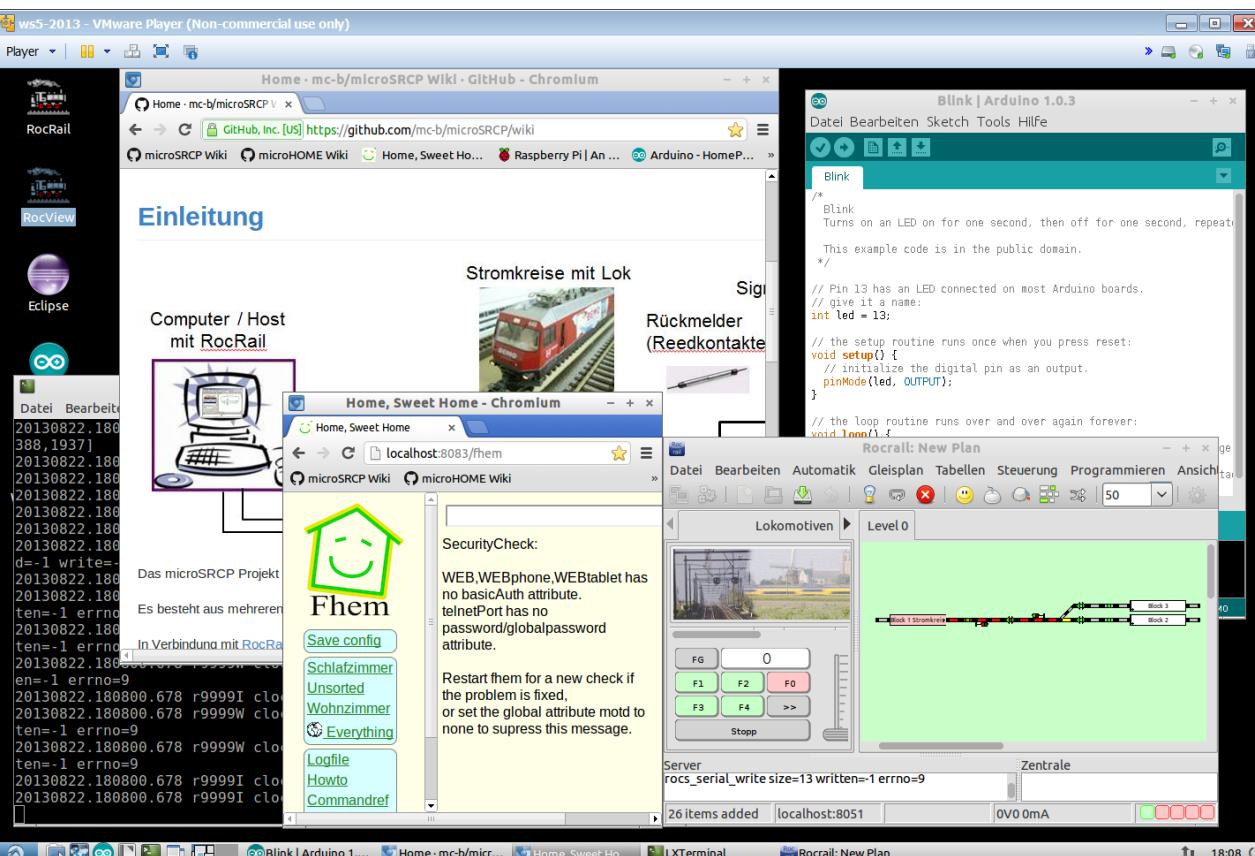


# = microSRCP



- Das microSRCP Projekt dient zum Steuern von Modelleisenbahnen.
- Es besteht aus mehreren Sketches welche aus [Arduino](#) [Microcontrollerboards](#) Modelleisenbahn Zentralen macht.
- In Verbindung mit Steuerungsprogramm [RocRail](#), welches auf Windows/Mac/Linux/RaspberryPi läuft, entsteht eine vollständige Modelleisenbahn Steuerung. RocRail Clients für Smartphones (z.B. andRoc) und Tablets ergänzen die Lösung.

# VMware Umgebung



**Bei Fehler:** This virtual machine was created with a newer version ...  
 Settings -> Compatibility Kompatibilität auf VMware Fusion 4 setzen.

- Xubuntu Linux mit xfce Window Manager analog Raspberry Pi.
- Arduino IDE, RocRail, VNCViewer (für die Verbindung zum RaspberryPi) sind auf dem Desktop als Verknüpfungen zu finden.
- fhem und die Dokumentationen zu microSRCP etc. im Chromium Browser als Lesezeichen

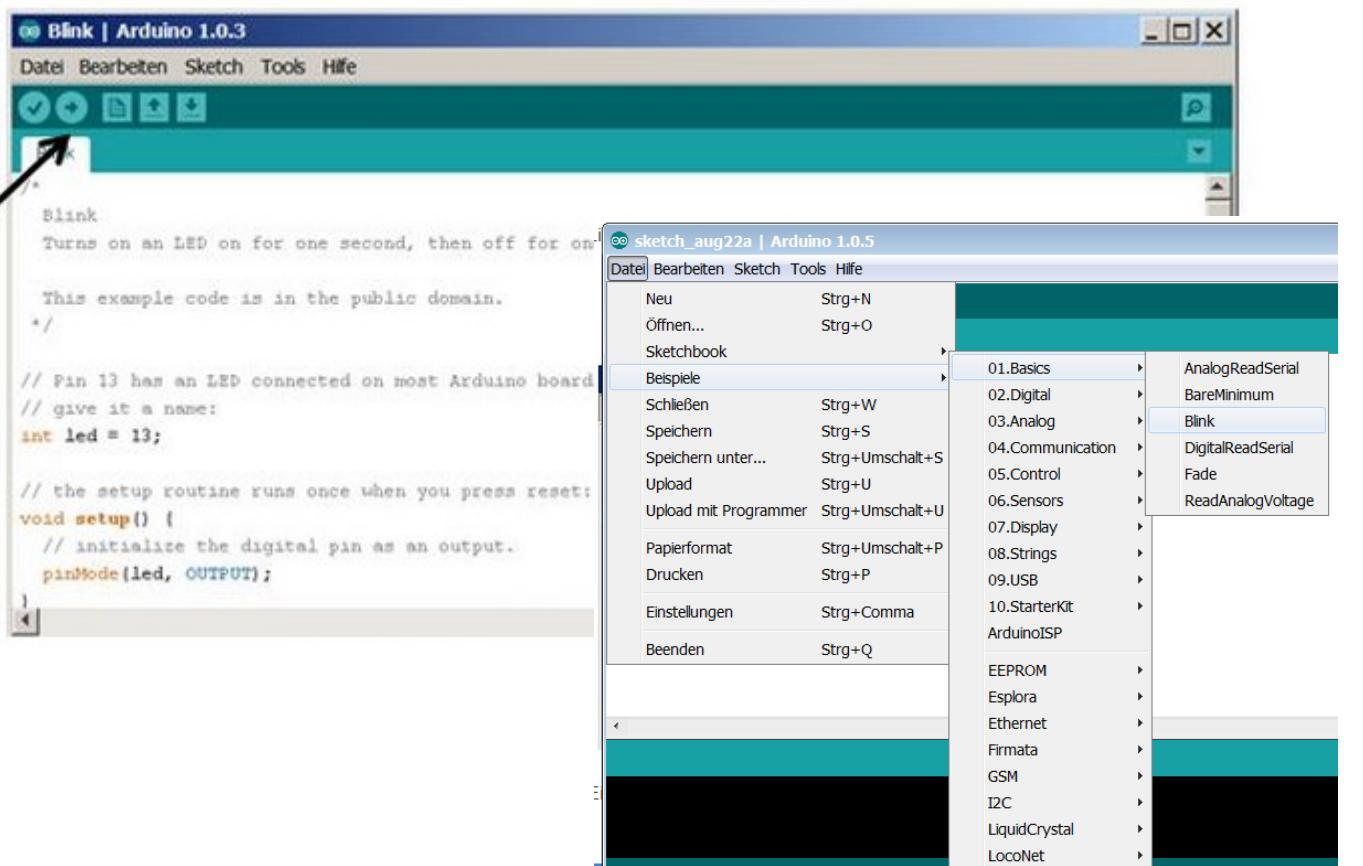
# Arduino Entwicklungsumgebung



Compilieren



UpLoad



- <https://github.com/mc-b/microSRCP/wiki/Grundkomponenten#wiki-Entwicklungsumgebung>

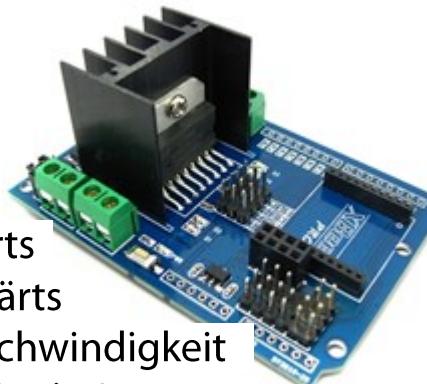
# Arduino Testumgebung



## MotoMama Shield

2 Motordriver mit je 2A

- Pin 8 - Input 1: Vorwärts
- Pin 9 - Input 2: Rückwärts
- Pin 10 - Enable A: Geschwindigkeit
- 11 wie 10, 12 wie 8, 13 wie 9
- (<http://de.wikipedia.org/wiki/Vierquadrantensteller>)
- ([http://rn-wissen.de/index.php/Getriebemotoren\\_Ansteuerung](http://rn-wissen.de/index.php/Getriebemotoren_Ansteuerung))

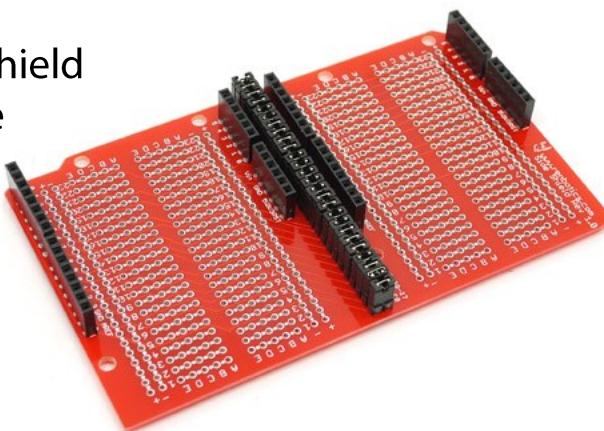


## IO Expansion Shield

- 4 LED an Pin 4-7
- 1 Servo an Pin 3
- 3 Sensoren an Pin A0-A2

## Seedstudio Side Shield

- Verdoppelt die Shield Fläche



## Arduino Duemilanove (2009)

- oder
- Arduino Mega
- mit Atmega1280

Uno machen Probleme mit  
Ubuntu/Debian

# Übung 1 (Kennenlernen)



- Installiert die benötigte Software (Entwicklungs-umgebung, Treiber) bzw. startet die VMWare Umgebung
- Schliesst das Arduino Board an den Computer an.
- Wählt unter Datei-> Beispiele -> 01. Basics den Blink Sketch aus.
- Wählt unter Tools -> Board das Board (Arduino 2009 oder Uno) und unter Tool -> Serieller Port den Port (com? Windows, ansonsten /dev/ttyUSB? Arduino 2009, /dev/ttyACM? Arduino Uno) aus.
- Ändert 'int led' = 13 auf 'int led = 4'
- Nach Überprüfen und Upload, sollte das Led an Pin 4 Blinken.

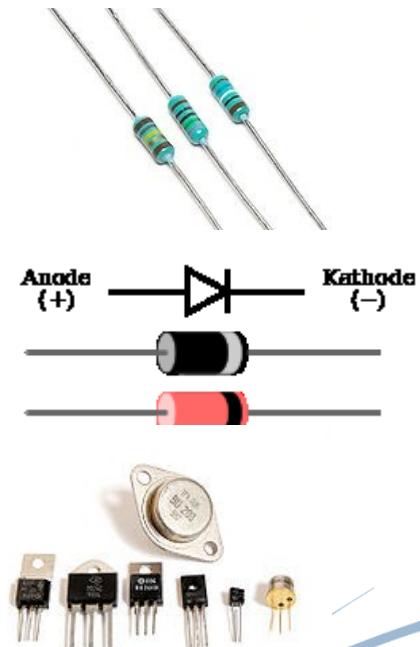
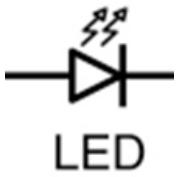
# Elektronik 1 mal 1



- Unter Elektronik (Lehre von der Steuerung von Elektronen) werden alle Vorgänge in Steuer-, Regel- und Verstärkerschaltungen sowie die Vorgänge in den hierfür verwendeten Bauelementen verstanden  
(<http://de.wikipedia.org/wiki/Elektronik>)



- Ein Widerstand ist ein zweipoliges passives elektrisches Bauelement zur Realisierung eines ohmschen Widerstandes in elektrischen und elektronischen Schaltungen
- Eine Diode (bzw. Leuchtdiode) ist ein elektrisches Bauelement, das Strom nur in einer Richtung passieren lässt und in der anderen Richtung als Isolator wirkt.
- Ein Transistor (Spezialform H-Brücke) ist ein elektronisches Bauelement zum Schalten und Verstärken von elektrischen Signalen,



# Struktur eines Sketches



Eine Methode setup() wo allgemeine Initialisierungen erledigt werden

```
void setup()
{
    // allgemeine Initialisierungen wie z.B. ein Pin auf Output oder Input zu setzen, Bsp:
    pinMode( 13, OUTPUT );
    pinMode( A0, INPUT_PULLUP );
}
```

Und eine Methode loop() wo die eigentliche Verarbeitung stattfindet.

```
void loop()
{
    if ( digitalRead( A0 ) == 0 )      // wenn A0 gegen GND verbunden ist
        digitalWrite( 13, HIGH );      // LED an
    else                                // sonst
        digitalWrite( 13, LOW );       // LED aus
}
```

Die setup() Methode wird nur einmal, am Anfang, durchlaufen, die loop() Methode endlos.

# AVR-GCC vs Arduino



## Programm in AVR-GCC

```
// Testprogramm: Blinken auf Pin PC0
//
#ifndef MCU           // Welcher AVR genutzt wird, wird i.A. im
#define MCU  atmega32
#endif

#ifndef F_CPU          // kann auch im Makefile definiert sein
#define F_CPU 1000000UL // Takt als LONG definieren, da zu groß für
#endif

#include <avr/io.h>    // Namen der IO Register
#include <util/delay.h> // Funktionen zum warten
// Achtung, damit delay richtig funktioniert muß mit Optimierung C
int main(void)
{
    DDRC = _BV(0);      // Nur PC0 als output, _BV(0) = (1<<0) =
    PORTC = 254;         // Pullups auf allen anderen Pins

    while (1)
    {
        PORTC &= 255-_BV(0); // 0 auf Bit 0 Ausgeben, Rest so lassen
        _delay_ms(100);     // 100 ms Warten
        PORTC |= _BV(0);   // 1 auf Bit 0 Ausgeben, Rest so lassen
        _delay_ms(100);
    }
}
```

Quelle: [http://www.rn-wissen.de/index.php/AVR-Einstieg\\_leicht\\_gemacht](http://www.rn-wissen.de/index.php/AVR-Einstieg_leicht_gemacht)

## Sketch mit Arduino Libraries

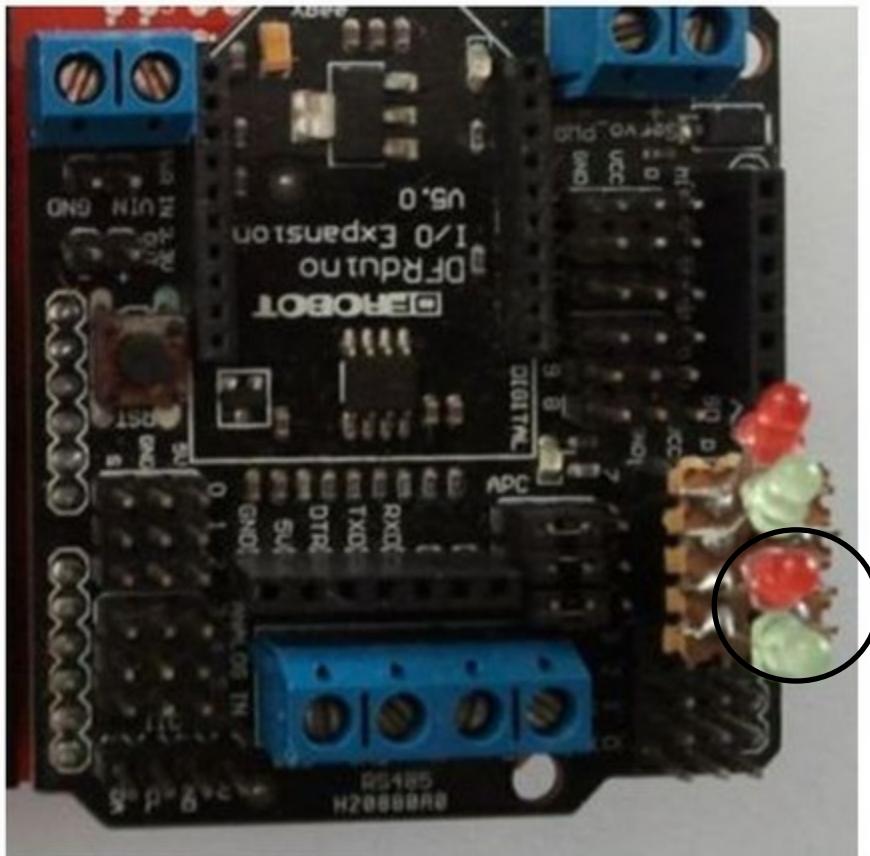
```
/*
Blink
Turns on an LED on for one second, then off for one second,
This example code is in the public domain.
*/

void setup() {
    // initialize the digital pin as an output.
    // Pin 13 has an LED connected on most Arduino boards:
    pinMode(13, OUTPUT);
}

void loop() {
    digitalWrite(13, HIGH);      // set the LED on
    delay(1000);                // wait for a second
    digitalWrite(13, LOW);       // set the LED off
    delay(1000);                // wait for a second
}
```

Referenz: <http://arduino.cc/en/Reference/HomePage>

# Ansteuerung von LED's (Aktoren)

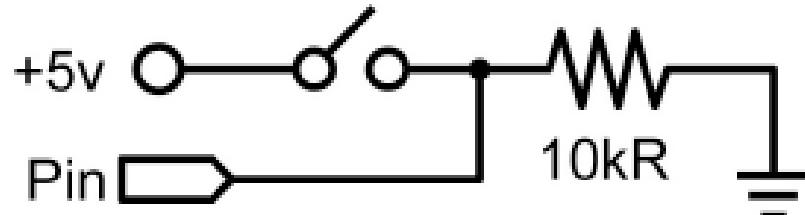


```
int rot = 4; // LED Lichtsignal
int gruen = 5;
// Die setup() Methode, wird einmal beim
// Start des Sketches durchlaufen
void setup()
{
    // verwende die nachfolgenden Pins als Output:
    pinMode(rot, OUTPUT);
    pinMode(gruen, OUTPUT);
}
// Die loop() Methode wird immer und immer aufgerufen,
// solange das Board am Strom angeschlossen ist.
void loop()
{
    digitalWrite(gruen, HIGH); // freie Fahrt
    digitalWrite(rot, LOW);
    delay(1000); // 1ne Sekunde warten
    digitalWrite(gruen, LOW); // Stop
    digitalWrite(rot, HIGH);
    delay(1000); // 1ne Sekunde warten
}
```

# Digital Input (Sensor)



## digital input



**ACHTUNG:** +5v und GND sind im obiger Zeichnung vertauscht.

```
int sensorPin = A0;           // Input Pin, z.B. mit Taster
int ledPin = 4;                // LED

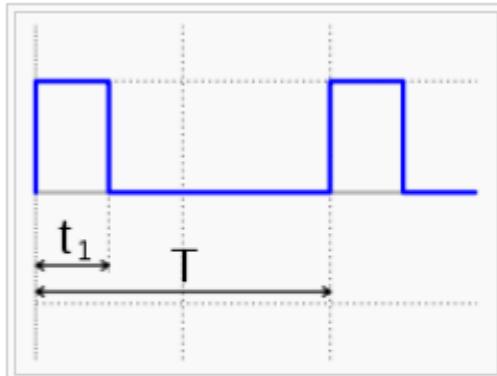
void setup()
{
    pinMode( sensorPin, INPUT_PULLUP ); // setzt den internen Widerstand
                                         // Pin ist gegen +5V geschaltet
    pinMode(ledPin, OUTPUT);
}

void loop()
{
    if ( digitalRead( sensorPin ) == LOW )
    {
        digitalWrite( ledPin, HIGH );
        delay( 1000 );
        digitalWrite( ledPin, LOW );
    }
}
```

# LED Dimmen - PWM (Aktor)



## pwm output



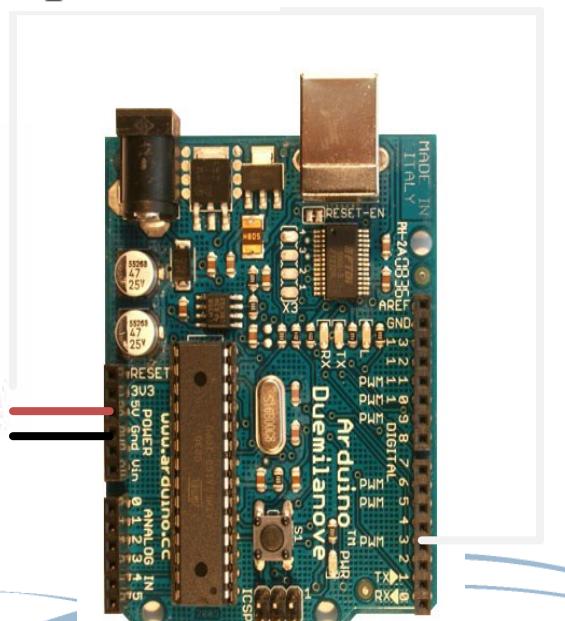
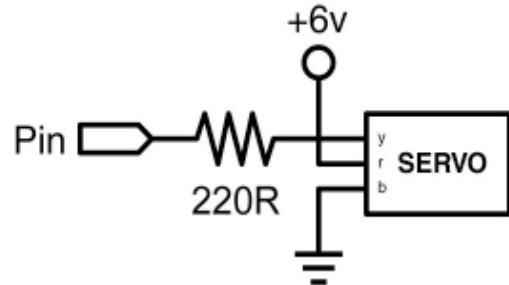
```
void setup() {}  
void loop()  
{  
    for(int value = 0; value < 255; value += 1) // langsam LED hell dimmen  
    {  
        analogWrite( 5, value );  
        delay(15); // Verzögerung um den Effekt sichtbar zu machen  
    }  
    delay( 500 );  
    for(int value = 255; value >= 1; value -=1) // langsam LED wieder abdunkeln  
    {  
        analogWrite( 5, value );  
        delay(15);  
    }  
    delay( 500 );  
}
```

Siehe: <http://de.wikipedia.org/wiki/Pulsweitenmodulation>

# Ansteuerung von Servo's (Aktor), Libraries



servo output



```
// Sweep
#include <Servo.h>          // Bekanntmachung der Servo Library
Servo myservo;               // Erstellt ein Servo Object zur Steuerung des Servos

int pos = 0;                  // Variable zur Speicherung der akt. Pos. des Servos

void setup()
{
    myservo.attach(3);        // Servo dem Pin 3 zuweisen.
}

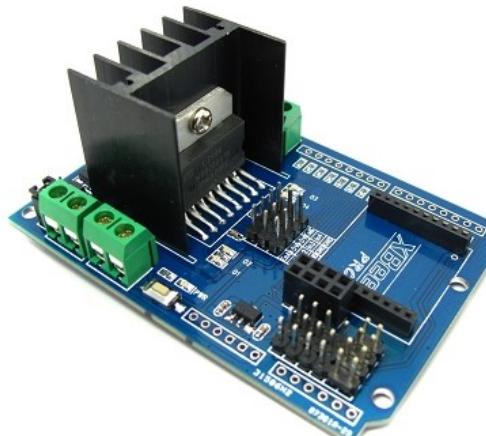
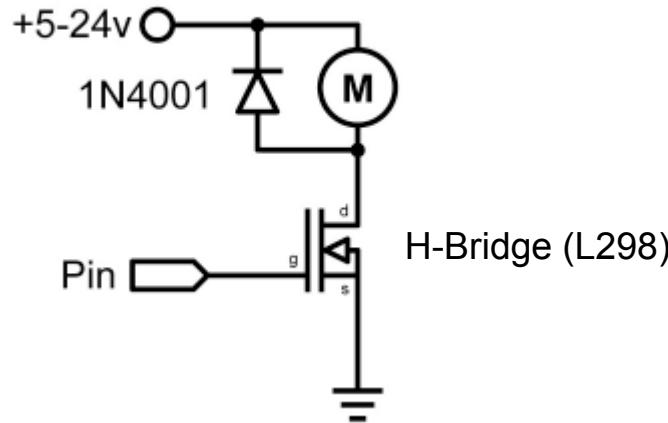
void loop()
{
    for(pos = 0; pos < 180; pos += 1)
        // Servo vom linken zum rechten Anschlag
    {
        // in Schritten von 1 schalten
        myservo.write(pos);
        delay(15); // 15 Millisekunden warten
    }
    for(pos = 180; pos>=1; pos-=1)
        // Servo vom rechten zum linken Anschlag
    {
        // in Schritten von 1 schalten
        myservo.write(pos);
        delay(15);
    }
}
```

ACHTUNG: vorher Servo-Hebel aussstecken und  
richtigen Pin verwenden!

# Ansteuerung von Motoren (Aktor), Shields



high current output



[http://www.rn-wissen.de/index.php/Getriebemotoren\\_Ansteuerung](http://www.rn-wissen.de/index.php/Getriebemotoren_Ansteuerung)

```
// Motor Shield test // mit MotoMama Shield // Test Ausgang 2
int dirpin1 = 12; // Fahrrichtung für Treiber 2, Pin 12 + 13
int dirpin2 = 13;
int speedpin = 11; // Geschwindigkeit für Treiber 2, Pin 11
int speed = 200;

void setup()
{
    pinMode(dirpin1, OUTPUT); // Pin als Output setzen
    pinMode(dirpin2, OUTPUT);
    pinMode(speedpin, OUTPUT);
}
void loop()
{
    digitalWrite(dirpin1,HIGH); // Fahrrichtung vorwaerts
    digitalWrite(dirpin2,LOW);
    analogWrite(speedpin, speed); // Geschwindigkeit setzen
    delay( 500 );
    analogWrite( speedpin, 0 );
    delay( 1000 );

    digitalWrite(dirpin1,LOW); // Fahrrichtung rueckwaerts
    digitalWrite(dirpin2,HIGH);
    analogWrite(speedpin, speed); // Geschwindigkeit setzen
    delay( 500 );
    analogWrite( speedpin, 0 );
    delay( 1000 );
}
```

# Arduino – Sprachreferenz



Structure	Variables	Functions
+ <code>setup()</code>	<b>Constants</b> + HIGH   LOW + INPUT   OUTPUT INPUT_PULLUP + true   false + integer constants + floating point constants	Digital I/O + pinMode() + digitalWrite() + digitalRead()
+ if	<b>Data Types</b> + void + boolean + char + unsigned char + byte + int + unsigned int + word + long + unsigned long + short + float + double + string - char array + String - object + array	Analog I/O + analogReference() + analogRead() + analogWrite() - PWM  Due only + analogReadResolution() + analogWriteResolution()
+ if...else	<b>Conversion</b> + char0 + byte0 + int0 + word0 + long0 + float0	Advanced I/O + tone() + noTone() + shiftOut() + shiftIn() + pulseIn()
+ for	<b>Variable Scope &amp; Qualifiers</b> + variable scope + static + volatile + const	Time + millis() + micros() + delay() + delayMicroseconds()
+ switch case	<b>Utilities</b> + sizeof()	Math + min() + max() + abs() + constrain() + map() + pow() + sqrt()
+ while		Trigonometry + sin() + cos() + tan()
+ do...while		Random Numbers + randomSeed() + random()
+ break		Bits and Bytes + lowByte() + highByte() + bitRead() + bitWrite() + bitSet() + bitClear() + bit0
+ continue		External Interrupts + attachInterrupt() + detachInterrupt()
+ return		Interrupts + interrupts() + noInterrupts()
+ goto		

## Standard Libraries

- + EEPROM - reading and writing to "permanent" storage
- + Ethernet - for connecting to the internet using the Arduino Ethernet Shield
- + Firmata - for communicating with applications on the computer using a standard serial protocol.
- + GSM - for connecting to a GSM/GRPS network with the GSM shield.
- + LiquidCrystal - for controlling liquid crystal displays (LCDs)
- + SD - for reading and writing SD cards
- + Servo - for controlling servo motors
- + SPI - for communicating with devices using the Serial Peripheral Interface (SPI) Bus
- + SoftwareSerial - for serial communication on any digital pins. Version 1.0 and later of Arduino incorporate Mikal Hart's NewSoftSerial library as SoftwareSerial.
- + Stepper - for controlling stepper motors
- + TFT - for drawing text , images, and shapes on the Arduino TFT screen
- + WiFi - for connecting to the internet using the Arduino WiFi shield
- + Wire - Two Wire Interface (TWI/I2C) for sending and receiving data over a net of devices or sensors.

<http://arduino.cc/en/Reference/HomePage>

<http://arduino.cc/en/Reference/Libraries>

# Übung 2 (Grundschaltungen)



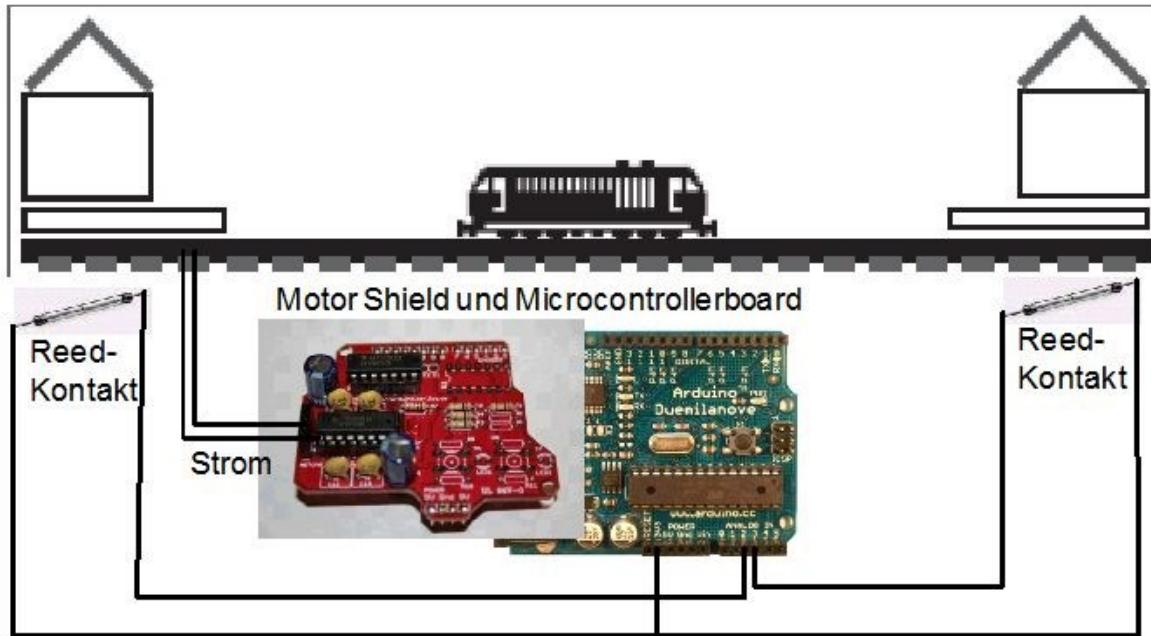
- Sucht mindestens zwei der vorherigen Beispiele aus und erweitert oder kombiniert sie, z.B.:
  - Lauflicht
  - Led rot an, Servo auf **60**, Licht grün an, Servo auf **100**
  - ...
- Die detaillierten Beschreibungen und Lösungen zu den Beispielen sind unter folgenden Link zu finden:
  - <https://github.com/mc-b/microSRCP/wiki/Grundkomponenten>

# Sensoren



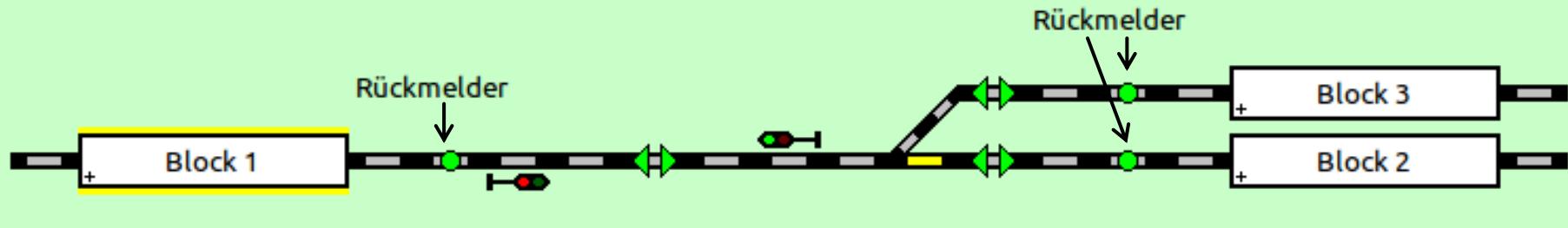
- Sensoren erlauben es auf externe Meldungen zu reagieren und ermöglichen so automatisierte Abläufe (Pendelzug)
- Arduino kann Analoge, Digitale, Serielle (RS-232C) und I2C Meldungen auswerten.
- Beispiele für Sensoren sind:
  - Temperatur-Sensor
  - 2/3 Axis Sensor (analog Smartphones), Tilt-Sensor
  - Licht-Sensor (Infrarot)
  - Taster, Potentiometer, Push-Button, Reed-Kontakt, Joystick
  - Hall Sensor (Metall-Detektor)

# Rückmelder, Sensoren



- Rückmelder ermöglichen automatischen Betrieb auf der Anlage. Sie melden eine bestimmte Position einer Lokomotive und können so z.B. für die Steuerung einer Pendelzugstrecke eingesetzt werden. Die Steuerung verwendet einfache Reedkontakte welche direkt an das Board angeschlossen werden können.

# Aufbau Übungsmodul



- Das Übungsmodell besteht aus einer Strecke mit einem Abstellgleis
- Drei Rückmelder (Sensoren) ermöglichen einen Pendelzugbetrieb
- Mittels einer Weiche mit Servo kann das Abstellgleis angesteuert werden.
- Zwei Lichtsignale (mit 2 LED) erlauben visuellen Feedback

# Übung 3 (Pendelzug)



- Implementiert einen Pendelzugbetrieb (fahrt nach links, stopp, warten, fahrt nach rechts, stopp, warten)
- Dabei könnt Ihr wählen ob die Lokomotive nur auf der Hauptstrecke von links nach rechts fährt oder das Abstellgleis mit einbezogen wird (Achtung nur Motor Treiber 2 verwenden, wegen Überschneidung von Treiber 1 mit Servo Library).
- Die detaillierten Beschreibungen und Lösungen zu den Beispielen sind unter folgenden Link zu finden:
  - [https://github.com/mc-b/microSRCP/wiki/Grundkomponenten#wiki-Rckmelder\\_Sensoren](https://github.com/mc-b/microSRCP/wiki/Grundkomponenten#wiki-Rckmelder_Sensoren)

# Arduino – mehr Kilobytes



- **SPI** – (Serial Peripheral Interface) ist ein von Motorola entwickeltes Bus-System ([http://de.wikipedia.org/wiki/Serial\\_Peripheral\\_Interface](http://de.wikipedia.org/wiki/Serial_Peripheral_Interface))
  - Wird für die Kommunikation mit dem Ethernet Shield, SD Card etc. verwendet.
- **ICSP Stecker**
  - Mittels ICSP Stecker kann, ohne Bootloader, Code auf den Arduino uploaded werden. Der Stecker dient, zusammen z.B. mit einem USBTiny (<http://learn.adafruit.com/usbtinyisp>), auch zum Aufspielen eines Bootloaders.
- **I2C** (<http://de.wikipedia.org/wiki/I2C> auch TWI) lokaler 2 Draht Bus (<https://github.com/mc-b/microSRCP/wiki/I2c-bus>)
  - Besser Arduino's direkt mittels USB Verbinden.
- **UART** – (Universal Asynchronous Receiver Transmitter - <http://de.wikipedia.org/wiki/UART>) – Serieller Bus, welcher, über FTDI USB-to-serial driver Chip oder Atmega16U2, mit der USB Schnittstelle verbunden ist. Über diesen werden neue Sketche uploaded. Belegt Pin 0 und 1.
- **Interrupts** – werden von den Arduino Libraries abstrahiert, siehe Reference <http://arduino.cc/en/Reference/HomePage>
- **Advanced Arduino Programming** – siehe <http://arduino.cc/en/Reference/Libraries> und <http://arduino.cc/en/Hacking/LibraryTutorial>.

# Zusammenfassung

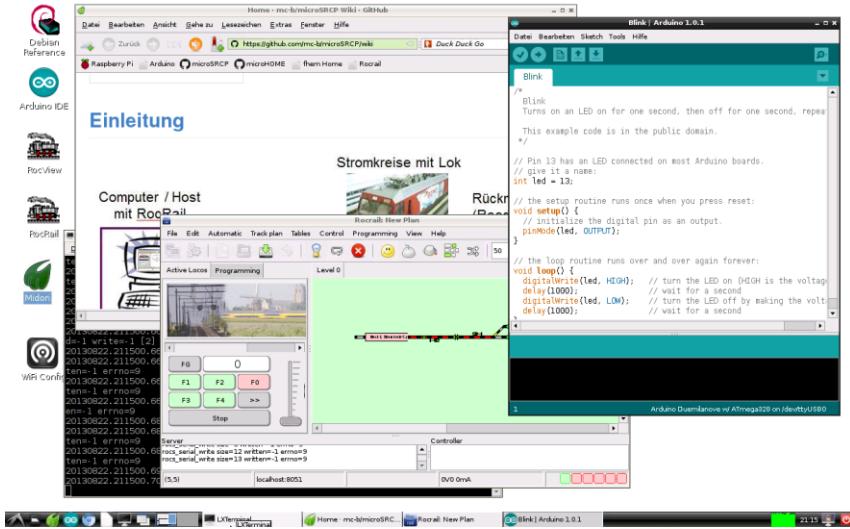
- Die Arduino-Plattform ist eine aus Soft-und Hardware bestehende Physical-Computing-Plattform
- Wikipedia ([http://de.wikipedia.org/wiki/Physical\\_Computing](http://de.wikipedia.org/wiki/Physical_Computing)):
  - Physical Computing bedeutet im weitesten Sinne, **interaktive, physische Systeme** durch die Verwendung von Hardware und Software zu erstellen. Diese Systeme **reagieren auf Ereignisse in der realen, analogen Welt und/oder wirken auf sie ein**. Obwohl diese Definition auch auf automatische Verkehrskontrollsysteme zutrifft, wird sie in diesem Zusammenhang nicht verwendet. Unter Physical Computing werden Systeme verstanden, die sich mit der Beziehung zwischen dem Menschen und der digitalen Welt befassen. Der Begriff wird meistens für Projekte mit einem künstlerischen oder Designhintergrund oder für Do-it-yourself-Hobbyprojekte verwendet. Dabei werden Sensoren und Mikrocontroller verwendet, um analoge Eingaben Software-Anwendungen verfügbar zu machen und/oder elektromechanische Geräte, wie Motoren, Servos, Leuchtdioden oder andere Hardware steuern.
- Einfache Programmierung mittels Sketches und einer Vielzahl von Shield's machen die Arduino zu einem universell einsetzbaren I/O Gerät.

# Agenda



- Vormittag
  - Einleitung „Internet der Dinge“, Sensoren, Aktoren – 20'
  - Die Arduino Plattform (Hardware, einfache Anwendungen, Sensoren) – 90'
  - Die Raspberry Pi Plattform – 20'
- Nachmittag
  - Arduino's und Raspberry Pi verbinden – 10'
  - Aufbau einer Steuerung anhand zweier konkreter Beispiele (Hausautomation und Modelleisenbahn-Steuerung) – 120'
  - Raspberry Pi, Arduino mit dem Internet verbinden – 30'

# Raspberry Pi, Model B (<http://www.raspberrypi.org/>)



- Der Raspberry Pi ist ein kreditkarten-grosser **Einplatinencomputer**
- Die Platine enthält das Ein-Chip-System BCM 2835 von Broadcom mit dem 700-MHz-Hauptprozessor ARM1176JZF-S sowie 512 MB Arbeitsspeicher. Das Modell B hat zudem eine Ethernet-Schnittstelle und einen zweiten USB-Anschluss.
- Linux und andere Betriebssysteme, welche die ARM-Architektur unterstützen, können installiert werden.
- Eine Festplattenschnittstelle ist nicht vorhanden. Stattdessen können Speicherkarten (SD bzw. MMC) als nicht-flüchtiger Speicher oder externe Festplatten und USB-Sticks über den USB-Port benutzt werden.
- [http://de.wikipedia.org/wiki/Raspberry\\_Pi](http://de.wikipedia.org/wiki/Raspberry_Pi)

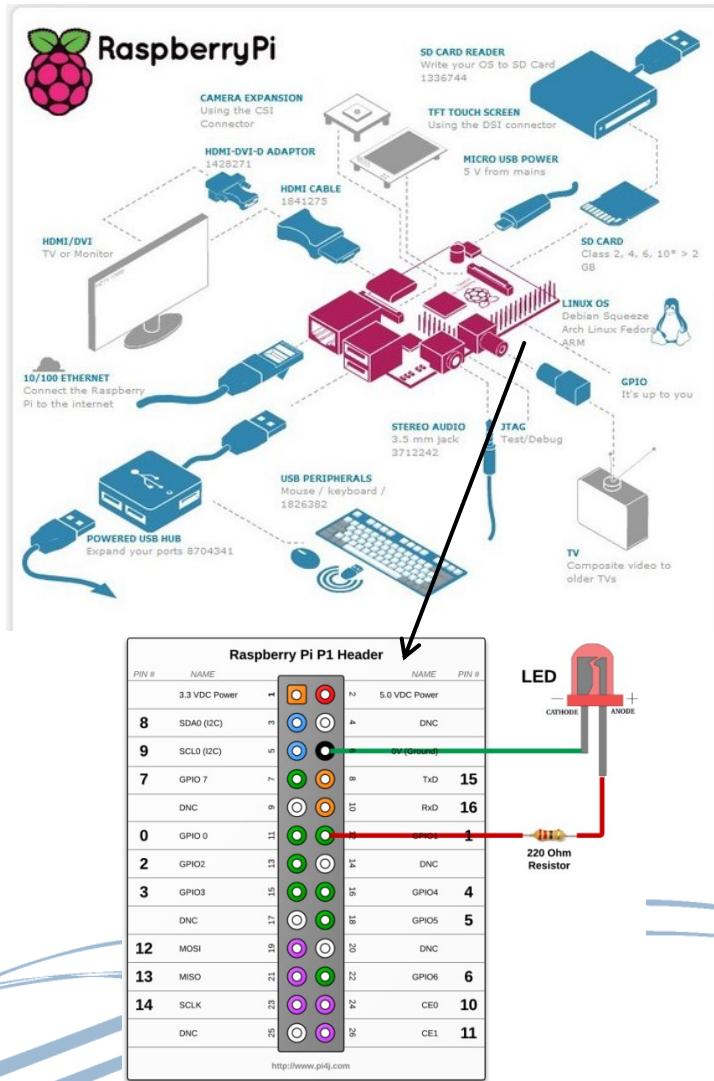
# Raspberry Pi, Spezifikation



	Modell A	Modell B
Preisempfehlung:	25 US-\$ (exkl. Mehrwertsteuer)	35 US-\$ (exkl. Mehrwertsteuer)
Größe:	Kreditkartengröße 85,60 mm × 53,98 mm × 17 mm	
SoC:	Broadcom BCM2835	
CPU:	ARM1176JZF-S (700 MHz)	
GPU:	Broadcom VideoCore IV	
Arbeitsspeicher (SDRAM):	256 MB	512 MB (bis Oktober 2012 256 MB)
USB 2.0 Anschlüsse:	1	2 (über integrierten Hub)
Videoausgabe:	FBAS, HDMI	
Tonausgabe:	3,5 mm-Klinkenstecker (analog), HDMI (digital)	
Nicht-flüchtiger Speicher:	SD (SDHC und SDXC)/MMC/SDIO-Kartenleser	
Netzwerk:	–	10/100 MBit Ethernet-Controller (LAN9512 des Herstellers SMSC <sup>[19]</sup> )
Schnittstellen:	Bis zu 16 GPIO-Pins, SPI, I²C, UART	
Echtzeituhr:	–	
Leistungsaufnahme: <sup>[20]</sup>	5 V, 500 mA (2,5 Watt)	5 V, 700 mA (3,5 Watt)
Stromversorgung: <sup>[20]</sup>	5 V Micro-USB-Anschluss (Micro-B), alternativ 4 × AA-Batterien	
Betriebssysteme:	GNU/Linux, BSD, RISC OS <sup>[21]</sup> , Plan 9 <sup>[22]</sup>	

Quelle: [http://de.wikipedia.org/wiki/Raspberry\\_Pi](http://de.wikipedia.org/wiki/Raspberry_Pi)

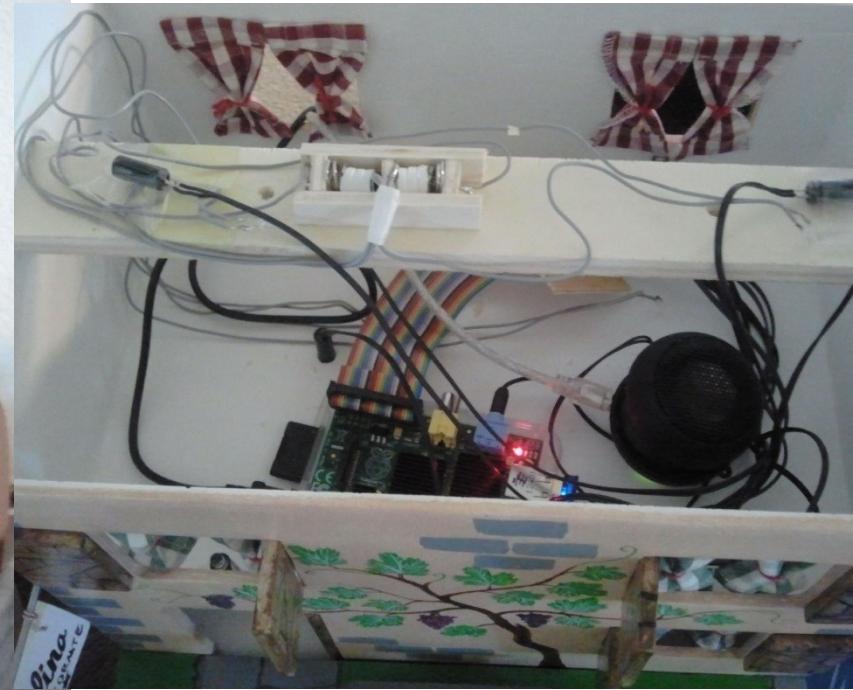
# Raspberry Pi I/O



- USB, LAN, HDMI, SD Card onBoard,
- WLAN mittels USB Adapter anschliessbar
- GPIO Port mit 26 Pin's

- 2 Pins eine Spannung von 5 Volt bereitstellen, aber auch genutzt werden können, um den Raspberry Pi mit Strom zu versorgen.
- 1 Pin eine Spannung von 3,3 Volt bereitstellt
- 1 Pin als Masse dient, sowie 4 derzeit mit der Masse verbundene Pins, die zukünftig eine andere Belegung bekommen könnten
- 17 Pins frei programmierbar sind, wovon einige Sonderfunktionen übernehmen können
  - 5 Pins können als SPI-Schnittstelle verwendet werden
  - 2 Pins haben einen 1,8-kΩ-Pull-Up-Widerstand (auf 3,3 V) und können als I<sub>C</sub>-Schnittstelle verwendet werden
  - 2 Pins können als UART-Schnittstelle verwendet werden.

# Raspberry Pi im Einsatz



[Hotel Mulino in Ascona, Schweiz](http://www.hotel-mulino.ch/)  
(<http://www.hotel-mulino.ch/>)



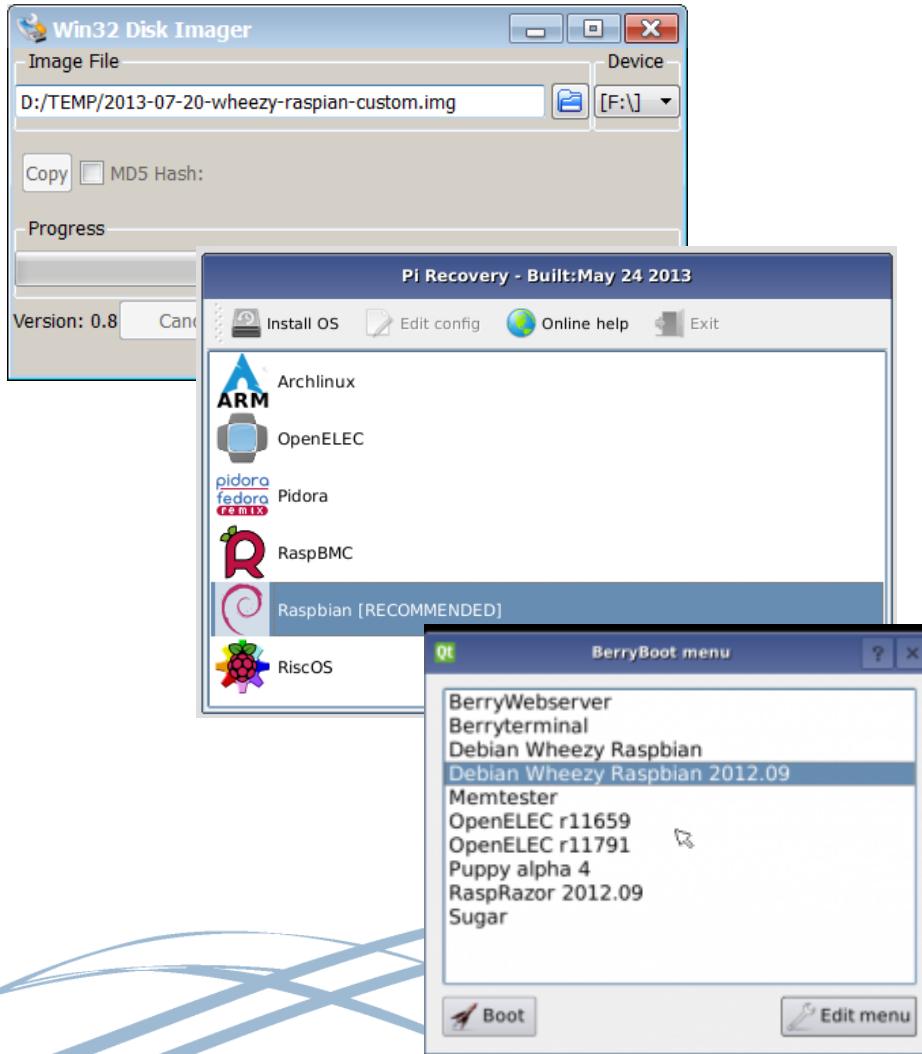
Wetterstation  
Temperatur  
Wind  
Regen  
Luftfeuchtigkeit  
  
F520  
  
gang\_licht  
mat\_computer  
mat\_halogen  
wz\_fernseher

# Raspberry Pi, weitere Einsatzmöglichkeiten



- Wegen des günstigen Preises und der geringen Leistungsaufnahme des Raspberry Pi ergeben sich abseits der vorgesehenen Nutzung als Schulrechner noch andere Möglichkeiten:
  - Musik-Streaming-Client
  - Media Center (**OpenElec**, RaspBMC)
  - Thin Client oder Server (z.B. als Reverse Proxy Server)
  - als Steuerungsplatine in einem Quadrocopter
  - Wetterstation
  - UKW-Radiosender
  - WLAN Access Point (<http://www.rpiblog.com/2012/12/turn-raspberry-pi-into-wireless-access.html> oder <https://help.ubuntu.com/community/WifiDocs/WirelessBroadcastSystem>)
  - Modelleisenbahn Steuerung (**RocRail**)
  - Hausautomation Steuerung (**fhem**)

# Raspberry Pi, Betriebssysteme



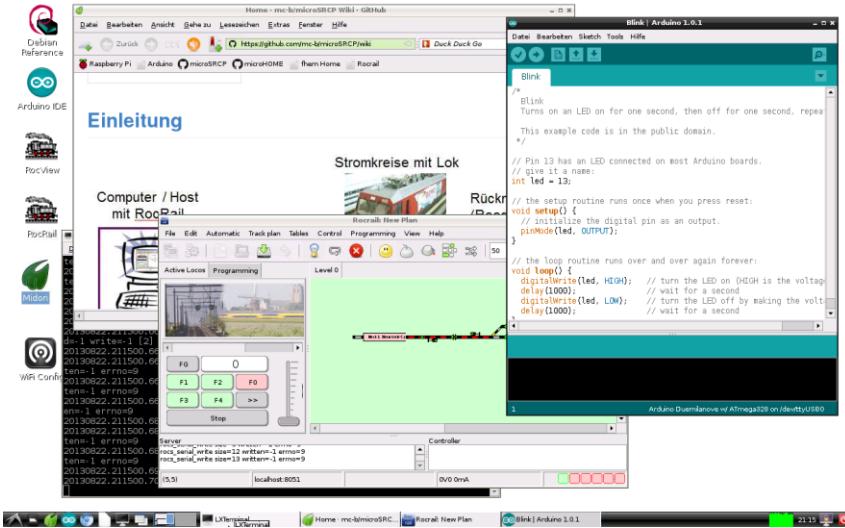
- Für den Raspberry Pi sind mehrere Open-Source-Betriebssysteme verfügbar. So kann sich der Käufer selbst entscheiden, welches er installieren möchte. Die Installation geschieht entweder durch das Einspielen eines **Images auf die SD-Karte** oder auch mit der einfacher zu verwendenden Eigenentwicklung **NOOBS**, die nur auf die Karte kopiert werden muss. Mit **BerryBoot** gibt es einen gleich einfach zu installierenden Bootloader, der es ermöglicht mehrere Betriebssysteme auf einer Karte parallel zu installieren.

# Raspberry Pi, Installationsvarianten



- Images auf SD-Karte
  - LowLevel: dd auf Linux und Mac, Win32DiskImager auf Windows.
  - Es wird nur der Platz für das installierte Image verwendet.
  - Kein Recovery System
- NOOBS
  - Einfach
  - Auswahl zwischen verschiedenen Systemen
  - Recovery System
  - Braucht bis 2 GB Platz für nicht installierte Systeme
- BerryBoot
  - Sehr platzsparend, weil Images komprimiert und on-Demand geholt werden
  - Installation mittels WLAN möglich
  - Auswahl zwischen verschiedenen Systemen
  - Recovery System
  - Kein Overclocking, mittels raspi-config, möglich

# Raspberry Pi, Installiertes Betriebssystem

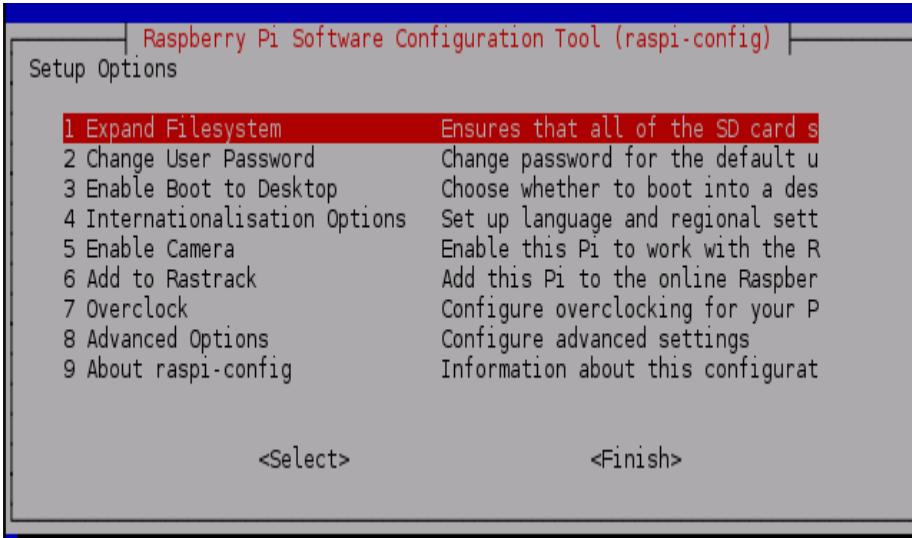


## Installation in der Linux Shell:

- sudo apt-get update
- sudo apt-get install samba samba-common-bin arduino \chromium-browser iw tightvncserver vncviewer
- fhem und RocRail Debian bzw. Raspberry Pi Pakete downloaden und jeweils wie folgt installieren:
  - sudo dpkg -i <paket>
  - sudo apt-get -f install

- Die empfohlene Linux-Distribution ist das auf **Debian** basierende Raspbian, welches auch installiert ist.
- Zusätzlich sind folgende Produkte installiert:
  - Samba – File/Print Server
  - Arduino – Entwicklungsumgebung
  - chromium-browser - Chrome
  - iw – WLAN Tools
  - tightvncserver vncviewer – VNC Server und VNC Viewer (Vergleichbar mit Remote Desktop auf Windows)
  - Fhem – Freundliche Hausautomatisierung und Energie-Messung
  - RocRail – Modelleisenbahn Steuerung

# Raspberry Pi, Erstkonfiguration



- Raspian bietet ein Konfigurationstool, welches beim ersten Start automatisch gestartet wird, an.
- Mit raspi-config können u.a.
  - Hostname
  - TimeZone/Tastaturlayout
  - Overclock (übertakten)
- eingestellt werden

# Raspberry Pi, wichtige Kommandos in LXTerminal



- **sudo [-i]**
  - Superuser werden, ein Password ist nicht erforderlich. -i = permanent
- **sudo service <servicename> start|stop**
  - Starten und Stoppen eines Services z.B. fhem, vncserver
- **sudo shutdown -r now**
  - Restart des Betriebssystems
- **sudo shutdown -h now**
  - Stoppen des Betriebssystems.
- **sudo apt-get install <paket>**
  - Installation neuer Software
- **ls, rm, mv, cd**
  - Anzeigen, löschen, verschieben von Dateien und Wechsel Verzeichnis
- **df -h, free -m, w**
  - Diskbelegung, Speicherbelegung, Auslastung CPU
- **scp <datei> <server>:<datei> oder scp <server>:<datei> <datei>**
  - Kopieren von Daten von einem System zu einem anderen, Alternativ Samba verwenden.
- **ifconfig – Ausgabe der eigenen IP-Adresse**

# Raspberry Pi, was ist wo?



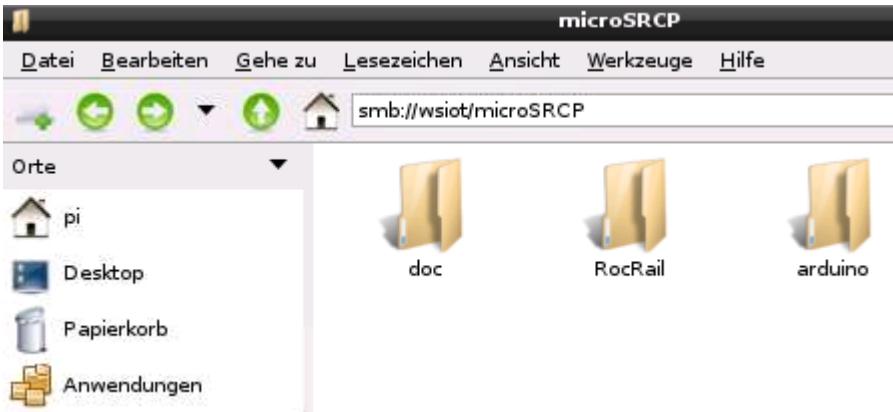
- fhem
  - /opt/fhem – Konfigurationsdateien, auch via Weboberfläche änderbar
    - fhem.cfg - Hauptkonfigurationsdatei
- microSRCP
  - /home/pi/microSRCP – Hauptverzeichnis
  - /home/pi/microSRCP/microSRCP/arduino – Arduino Sketchbook Speicherort
  - microSRCP mit GitHub abgleichen – *git pull* im microSRCP Verzeichnis
- RocRail
  - ....../microSRCP/RocRail/RaspberryPi – Konfigurationsdateien RocRail
  - rocrail.ini – Zentrale, Anschluss etc.
  - plan.xml – Gleisplan, Lokomotiven etc.
- Ino –Compilierung von Sketches mittels Kommandozeile
  - /home/pi/inobuild – microSRCPServer und StandardFirmata Sketch

# Raspberry Pi, Ino

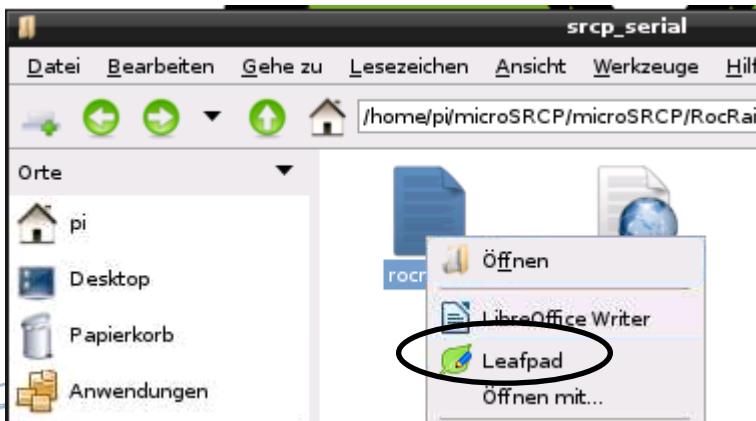


- Kommandlinetool um Arduino Sketches zu compilieren und upzuladen.
  - <http://inotool.org/quickstart>
- Compilieren und uploaden
  - cd inobuild/microSRCPServer
  - ino build –m [uno|atmega328|mega]
  - ino upload –m [uno|atmega328|mega]
- -m atmega328 steht für den Arduino 2009. mega für den AtMega mit 1280 Prozessor
- Gleiche Variante funktioniert auch für StandardFirmata

# Raspberry Pi, Dateimanager



- Zugriff auf freigegebenes Laufwerk
  - smb://<host>/microSRCP



- Datei editieren
  - Rechte Maustaste - Leafpad

# Raspberry Pi, Zugriff (ohne Bildschirm)



ws5-2013 - VMware Player (Non-commercial use only)

Player | II | ⌂ | ⌂ | ⌂ | ⌂

RocRail

Datei Bearbeiten Reiter Hilfe  
pi@raspiX: ~  
pi@wsiot:~\$ ssh pi@192.168.178.66  
pi@192.168.178.66's password:  
Linux raspix 3.6.11+ #506 PREEMPT Fri Jul 19 20:01:57 BST 2013 armv6l  
  
The programs included with the Debian GNU/Linux system are free software;  
the exact distribution terms for each program are described in the  
individual files in /usr/share/doc/\*copyright.  
  
Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent  
permitted by applicable law.  
Last login: Wed Aug 21 19:15:21 2013  
pi@raspix ~ \$

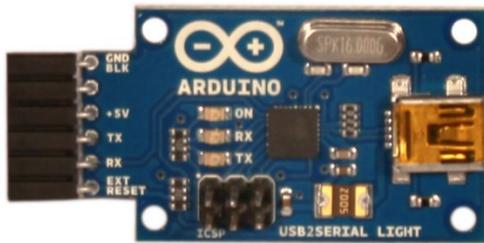
ws5-2013 - VMware Player (Non-commercial use only)

Player | II | ⌂ | ⌂ | ⌂ | ⌂

RocRail

Datei Bearbeiten Reiter Hilfe  
pi@wsiot:~\$ vncviewer  
serve...alog - + x  
VNC server:  
192.168.178.66:1

RocView



- ssh, putty
  - ssh pi@<ip-adresse>
  - User: pi, Password: arduino
  
- VNCViewer (UltraVNC Viewer)
  - vncviewer
  - Host: <ip-adresse>:1
  - Password: arduino
  
- GPIO Schnittstelle (Seriell)
  - Durch Anschluss eines USB Serial Adapters an Pin 4 und 5

# Übung 4 (Raspberry Pi)



- Verbindet Euch mit einem Raspberry Pi, mit einer der vorher vorgestellten Methoden.
- Startet den midori Browser, sucht die Dokumentation Grundschaltungen vom microSRCP Projekt.

# Zusammenfassung

- Der Raspberry Pi ist ein kreditkartengrosser **Einplatinencomputer**.
- **Entwicklung:**
  - Ein **Prototyp** mit einem **Atmel-ATmega644-Mikrocontroller** wurde im Jahr 2006 produziert. Die Schaltpläne der Platine wurden veröffentlicht.
  - Die Leistungen des Gerätes überzeugten die Entwickler nicht, durch den zu diesem Zeitpunkt eintretenden Booms von Smartphones kamen geeignete **ARM-Prozessoren** auf den Markt und man fand mit dem BCM2835 einen billigen Prozessor mit verhältnismäßig hoher Leistung und entwarf für diese CPU eine neue Mehrlagenplatine.

# Agenda



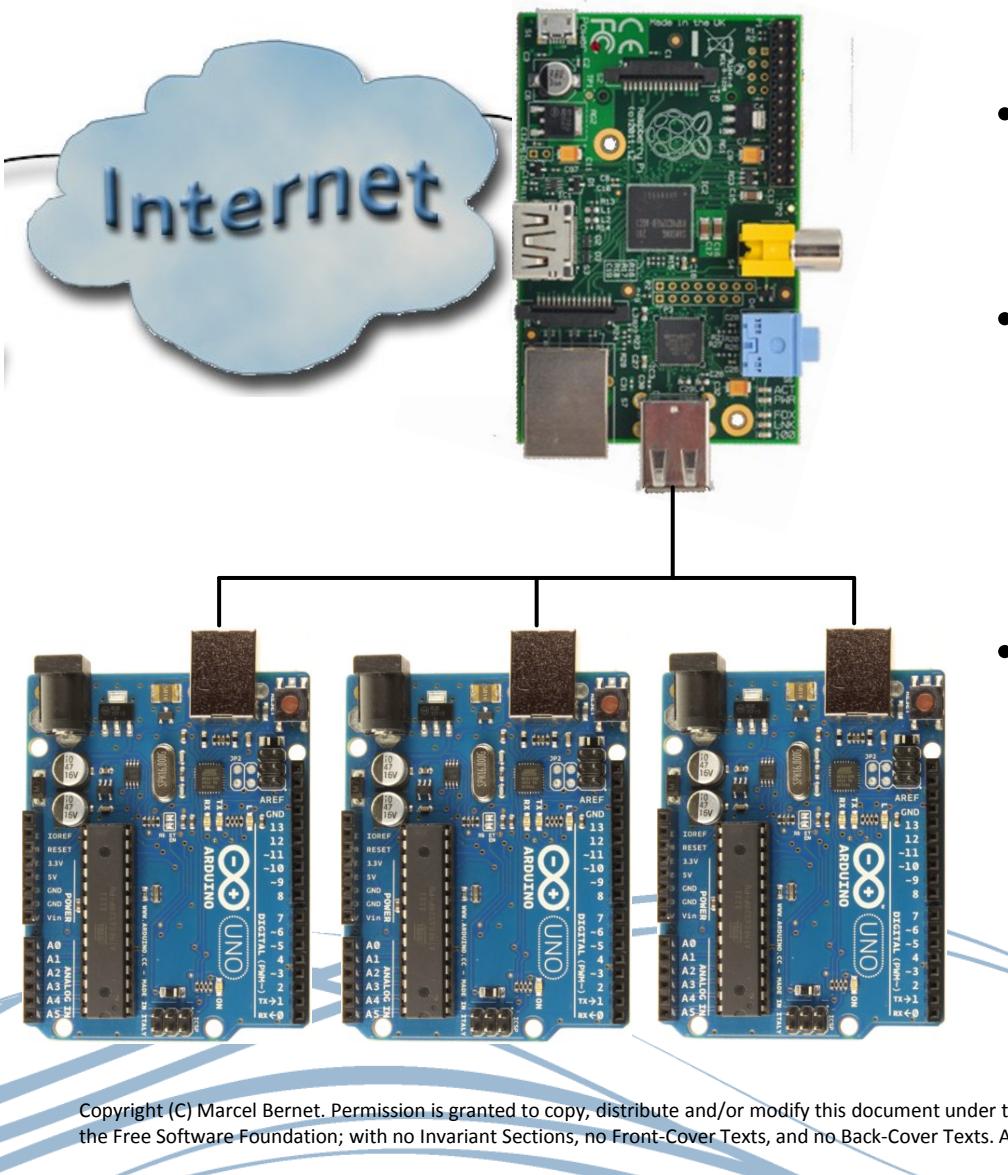
- Vormittag
  - Einleitung „Internet der Dinge“, Sensoren, Aktoren – 20'
  - Die Arduino Plattform (Hardware, einfache Anwendungen, Sensoren) – 90'
  - Die Raspberry Pi Plattform – 20'
- Nachmittag
  - Arduino's und Raspberry Pi verbinden – 10'
  - Aufbau einer Steuerung anhand zweier konkreter Beispiele (Hausautomation und Modelleisenbahn-Steuerung) – 120'
  - Raspberry Pi, Arduino mit dem Internet verbinden – 30'

# Raspberry Pi - Arduino



- **Das Internet der Dinge** (auch englisch Internet of Things) **beschreibt, dass der (Personal) Computer zunehmend als Gerät verschwinden und durch „intelligente Gegenstände“ ersetzt wird.**
- Der **Raspberry Pi** ist ein kreditkartengrosser **Einplatinencomputer**.
- Die **Arduino-Plattform** ist eine aus Soft-und Hardware bestehende Physical-Computing-Plattform (I/O Gerät).
- Kombiniert man physische Dinge (Modelleisenbahn, Lampe etc.) mit Raspberry Pi und Arduino's – entsteht ein Netzwerk von „**intelligenten Gegenständen**“ welches mit dem Internet verbunden werden kann.

# Raspberry Pi – Arduino im Detail



- Arduino's werden an USB Schnittstelle vom Raspberry Pi angeschlossen.
- Der Raspberry PI bildet die Schnittstelle nach aussen und die Arduino's übernehmen lokale Steuerungs- (Aktor) und Rückmelde- (Sensor) Aufgaben
- Vorteile:
  - Flexibel, Universell
  - Eine Zentrale Steuerung (Raspberry Pi)
  - Einen Arduino ist billiger zum Ersetzen als einen Raspberry Pi
  - USB Kabel und Hub's sind billig und zuverlässig

# Raspberry Pi – Arduino - Anschlüsse



```
pi@raspix ~ $ lsusb
Bus 001 Device 002: ID 0424:9512 Standard Microsystems Corp.
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 001 Device 003: ID 0424:ec00 Standard Microsystems Corp.
Bus 001 Device 004: ID 148f:5370 Ralink Technology, Corp. RT5370 Wireless Adapter
Bus 001 Device 005: ID 1a40:0101 Terminus Technology Inc. 4-Port HUB
Bus 001 Device 010: ID 2341:0001 Arduino SA Uno (CDC ACM)
Bus 001 Device 008: ID 0403:6001 Future Technology Devices International, Ltd FT232 USB-Serial (UART)
Bus 001 Device 006: ID 049f:0070 Compaq Computer Corp.
Bus 001 Device 007: ID 046d:c05b Logitech, Inc. M-U0004 810-001317 [B110 Optical USB Mouse]
pi@raspix ~ $ ls -l /dev/ttyUSB0 /dev/ttyACM0
crw-rw---T 1 root dialout 166, 0 Aug 23 15:01 /dev/ttYACM0
crw-rw---T 1 root dialout 188, 0 Aug 23 14:57 /dev/ttYUSB0
```

- lsusb – gibt die installierten Geräte aus
  - Arduino Sa Uno = Arduino Uno an /dev/ttYACM?
  - FTDI FT232 USB-Serial = Arduino 2009 an /dev/ttYUSB?
- Die Arduino Entwicklungs-umgebung erkennt die Devices automatisch.



# Übung 5 (Raspberry Pi – Arduino)



- Verbindet mindestens einen Arduino mittels USB Kabel mit dem Raspberry Pi
- Startet die Arduino Entwicklungsumgebung, wählt einen Beispiel Sketch (z.B Blink) und uploadet diesen auf den/die Arduino's.

# Zusammenfassung

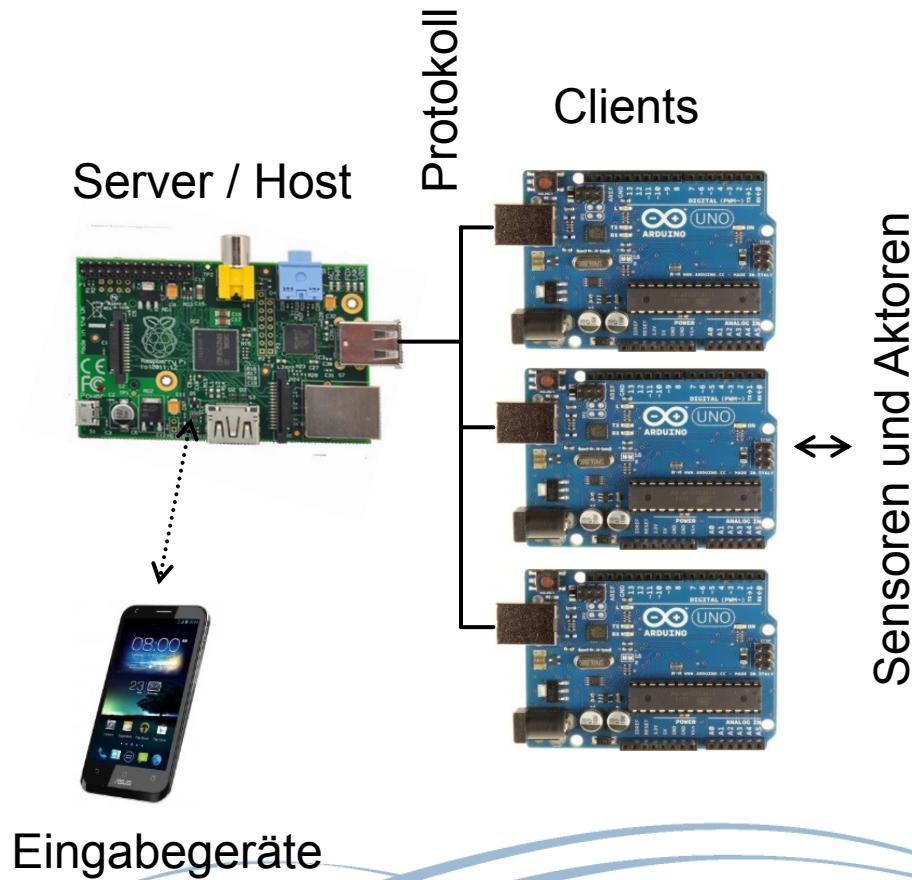
- Der **Raspberry Pi** ist ein kreditkartengrosser **Einplatinencomputer**.
- Die **Arduino-Plattform** ist eine aus Soft- und Hardware bestehende Physical-Computing-Plattform (I/O Gerät).
- Beide Geräte zusammen ergänzen sich gegenseitig.

# Agenda



- Vormittag
  - Einleitung „Internet der Dinge“, Sensoren, Aktoren – 20'
  - Die Arduino Plattform (Hardware, einfache Anwendungen, Sensoren) – 90'
  - Die Raspberry Pi Plattform – 20'
- Nachmittag
  - Arduino's und Raspberry Pi verbinden – 10'
  - Aufbau einer Steuerung anhand zweier konkreter Beispiele (Hausautomation und Modelleisenbahn-Steuerung) – 120'
  - Raspberry Pi, Arduino mit dem Internet verbinden – 30'

# Grundsätzlicher Aufbau



- **Server** welche die Clients überwacht und steuert
- **Protokoll** zwischen Client und Server
- **Clients** welche die Befehle vom Server entgegennehmen und weiterleiten an
- **Sensoren und Aktoren**
  - Readkontakte, LED, Servo's, Motoren etc.
- **Eingabegeräte** wie:
  - PC Programm oder APP's auf dem Smartphone

# Modelleisenbahn vs Hausautomation



- Modelleisenbahn Komponenten
  - Server: RaspberryPi/RocRail
  - Client: Arduino/microSRCPServer
  - Protokoll: SRCP
  - Sensoren und Aktoren = Geräte
  - Eingabegeräte: RocView, andRoc, iRoc ...
- <https://github.com/mc-b/microSRCP/wiki>
- <http://de.wikipedia.org/wiki/Rocrail>
- <http://srcpd.sourceforge.net/>
- **Rocrail** ist eine freie Software zur Steuerung von digitalen Modelleisenbahnen. Die Züge können manuell, voll-automatisch oder in einem Mischbetrieb gesteuert werden.
- Hausautomation Komponenten
  - Server: RaspberryPi/fhem
  - Client: Arduino/StandardFirmata
  - Protokoll: Firmata
  - Sensoren und Aktoren = Arduino Pin's
  - Eingabegeräte: Browser, andFhem ...
- <http://www.fhemwiki.de/wiki/Arduino>
- <http://de.wikipedia.org/wiki/FHEM>
- [http://firmata.org/wiki/Main\\_Page](http://firmata.org/wiki/Main_Page)
- **FHEM** ist ein Perl-basiertes Serverprogramm für die Hausautomation, der zur automatisierten Bedienung von Aktoren wie zum Beispiel Lichtschaltern oder Heizung sowie der Aufzeichnung von Sensorinformationen wie Raumtemperatur oder Luftfeuchtigkeit dient.

# Modelleisenbahn: Protokoll SRCP



- Das Simple Railroad Command Protocol (SRCP) ist ein auf TCP/IP basierendes Protokoll, das eine einheitliche Kommunikation zwischen Modellbahn-Software (Host/RocRail) und -Hardware (Arduino) ermöglicht.

## Gerätegruppen

Das SRCP Protokoll ist u.a. unterteilt in folgende Gerätegruppen

- GL Generic Loco - Lokomotiven
- GA Generic Accessoire - Zubehör z.B. Signal, Weiche
- FB Feed Back – Rückmelder, Sensor
- SM Service Mode – zum setzen von Konfigurationsvariablen (CV's).
- POWER Energieversorgung
- SERVER SRCP Server
- SESSION SRCP Clientsession

Jede Gerätegruppe hat ihren eigenen Adressbereich, so kann es zu keinen Konflikten zwischen Gerätegruppen kommen.

## Kommunikation

Eine Standard Kommunikation sieht, verkürzt, so aus:

```
// Initialisierung
recv: 1, SET PROTOCOL SRCP 0.8
send: 1, 25817 201 OK PROTOCOL SRCP
recv: 1, SET CONNECTIONMODE SRCP COMMAND
send: 2, 25921 202 OK
recv: 2, GO
send: 2, 26022 200 OK GO 1
recv: 2, GET 1 POWER
send: 2, 26124 100 INFO 0 POWER 460

// Geraete (GA) Schalten:
recv: 2, SET 5 GA 1 0 1 -1
send: 2, 42051 200 OK

// Lokomotiven (GL) steuern
recv: 2, SET 4 GL 1 0 84 128 1 0 0 0 0
send: 2, 50576 200 OK

// Sensoren (FB) melden - Arduino an Host
send: 2, 66698 100 INFO 0 FB 2 1
send: 2, 68402 100 INFO 0 FB 2 0
```

<https://github.com/mc-b/microSRCP/wiki/Srcp-protokoll>

# Modelleisenbahn: Geräte



- Die Geräte werden direkt in der setup() Methode des Sketches initialisiert bzw. registriert.
- Zur Registrierung dient die zentrale Klasse DeviceManager. Geräte welche dem DeviceManager nicht bekannt gemacht werden, werden nicht gefunden und können auch nicht angesteuert werden.

```
void setup()
{
    BEGIN( 9600 );
    // Geraete initialisieren, je nach Board und Verwendung
    DeviceManager.addAccessoire( new dev::GASignal( ADDR(1), 4, 5 ) );           // 2 Signale mit 2 LED an Ports 4 - 7.
    DeviceManager.addAccessoire( new dev::GASignal( ADDR(2), 6, 7 ) );
    DeviceManager.addAccessoire( new dev::GASlowServo( ADDR(3), 2, 60, 90 ) );      // Servo mit Addr 3 an Pin 2, min. Stel
    DeviceManager.addAccessoire( new dev::GASlowServo( ADDR(4), 3, 60, 90 ) );      // und Weiterschalten um 1ne Position a
    DeviceManager.addFeedback( new dev::FBSwitchSensor( ADDR(1), A0, A3 ) );        // Sensoren, jeweils in Gruppen von 8 (
    DeviceManager.addLoco( new dev::GLMotoMamaAnalog( ADDR(2), 11, 12, 13 ) );
}
```

[https://github.com/mc-b/microSRCP/wiki/Steuerungskonfiguration#wiki-Gerte\\_definieren](https://github.com/mc-b/microSRCP/wiki/Steuerungskonfiguration#wiki-Gerte_definieren)

# Modelleisenbahn: Motordriver Gerät



```
GLArduinoMotor::GLArduinoMotor( int addr, uint8_t pin, uint8_t dir )
{
    this->addr = addr;
    this->pin = pin;
    this->dir = dir;

    pinMode( dir, OUTPUT );
    digitalWrite( dir, HIGH );

    // ohne Funktion?
    //setPwmFrequency( pin, 1024 );
    pinMode( pin, OUTPUT );
}

int GLArduinoMotor::set( int addr, int drivemode, int v, int v_max, int fn[] )
{
    // rueckwärts?
    if ( drivemode == 0 )
        digitalWrite( dir, LOW );
    // vorwärts
    else
        digitalWrite( dir, HIGH );

    if ( v > v_max )
        v = v_max;
    if ( v < 0 )
        v = 0;

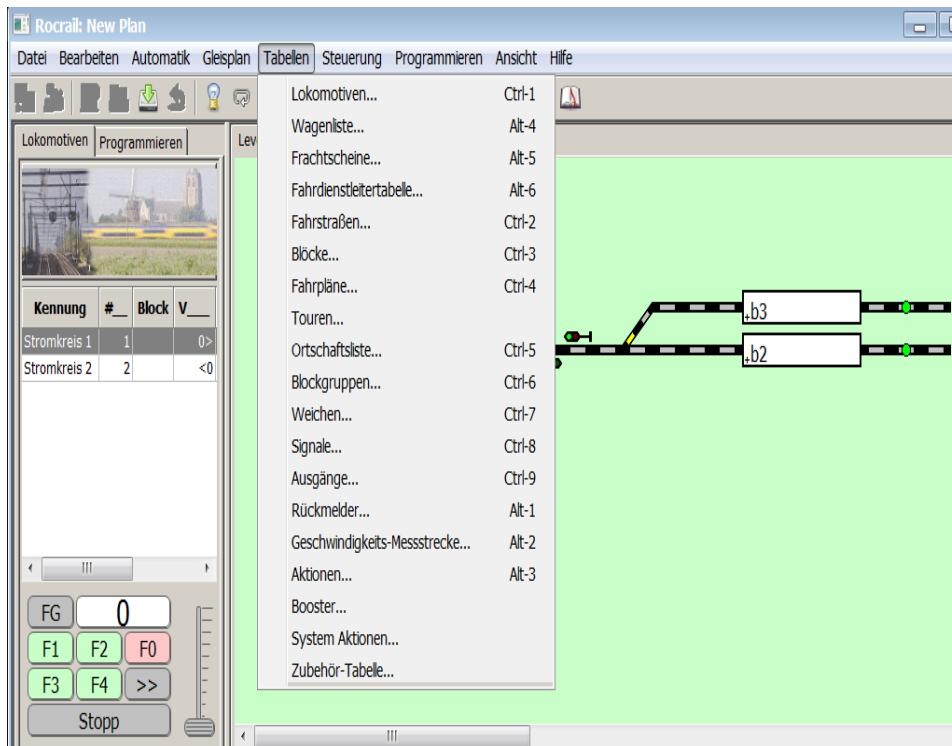
    v = map( v, 0, v_max, 0, 255 );
    analogWrite( pin, v );

    return (200);
}
```

- Gerät zur Ansteuerung eines Motordrivers.
- Im **Konstruktor** werden die Pin's initialisiert
- In der Methode **set** wird der SRCP Befehl von RocRail verarbeitet.

<https://github.com/mc-b/microSRCP/wiki/Steuerungerweiterungen>

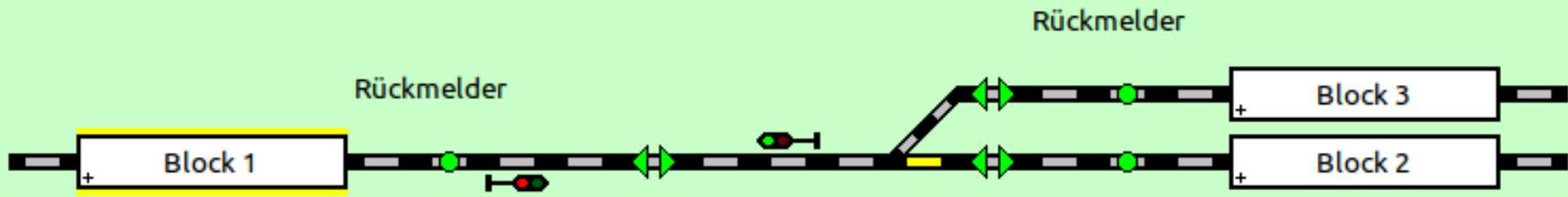
# Modelleisenbahn: RocRail



- **Rocrail** ist eine freie Software zur Steuerung von digitalen Modelleisenbahnen. Die Züge können manuell, vollautomatisch oder in einem Mischbetrieb gesteuert werden.
- Es verwaltet Lokomotiven, Sensoren, Signale etc. in Tabellen.
- Bietet einen komfortablen Gleisplaneditor
- Und bietet Automatische Abläufe mittels Blöcken, Fahrstrassen und Fahrplänen.

<http://wiki.rocrail.net/doku.php?id=stepbystep-de>

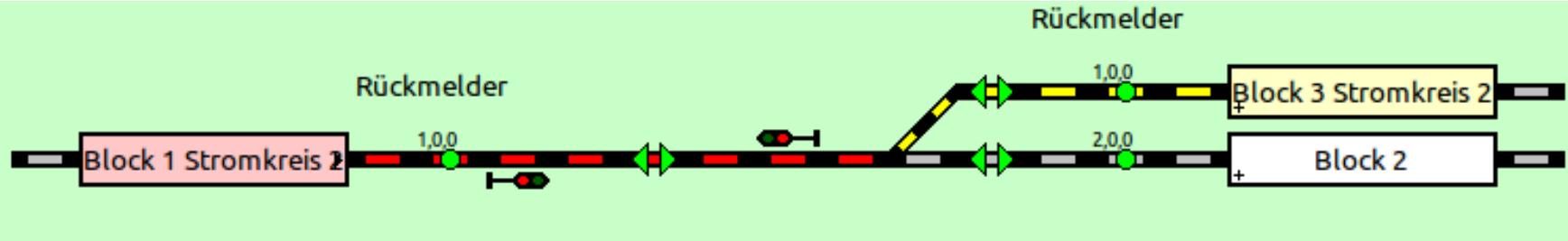
# Aufgabe 1a: Modelleisenbahn



Baut mit Rocrail, Raspberry Pi (alternativ VMWare Umgebung) und Arduino eine funktionsfähige Modelleisenbahnsteuerung auf.

- microSRCPServer Sketch in Arduino uploaden
- Rocrail und RocView starten
- Arduino (mit microSRCPServer Sketch) in RocRail als srcp Zentrale einrichten
- 2 Stromkreise (Lokomotiven) anlegen (Stromkreis 1 mit Adresse 1 und Stromkreis 2 mit Adresse 2)
- Obiger Gleisplan erstellen
  - 3 Rückmelder mit den Adressen 1 – 3
  - 2 Signale mit Adresse 1 und 2, die Led sind an Port 0 und 1
  - 1 Weiche (Servo) mit Adresse 3
- Schritt für Schritt Anleitung: <http://wiki.rocrail.net/doku.php?id=stepbystep-de>
- **ACHTUNG:** Servo schaltet langsam. Unter Datei -> RocRail Eigenschaften -> Schaltzeiten (Lichtsignale) erhöhen!

# Aufgabe 1b: Modelleisenbahn



- *Blöcke und Fahrstrassen sind die Grundlage von automatischen Abläufen auf der Modelleisenbahn.*
- Erweitert den Gleisplan um Blöcke und Fahrstrassen
  - Erstellt zuerst die 3 Blöcke mit den Bezeichnungen Block 1 – 3 und die Richtungszeiger.
  - Setzt bei Block/Fahrstrassen jeweils den Rückmelder des Blocks als enter2in Aktion
  - Lasst anschliessend automatisch die Fahrstrassen erstellen mittels dem Pulldownmenue Datei -> Analysieren -> Analysieren.
  - Werden keine Fehler angezeigt, könnt Ihr Block 1 mit Stromkreis 2 (Lokbelegung setzen) belegen, Fahrstrom und Automatikmodus einschalten und dann auf dem Block 1 die Lok starten.
- Siehe auch Schritt für Schritt Anleitung ab Punkt 7.3

# Aufgabe 1c: Modelleisenbahn



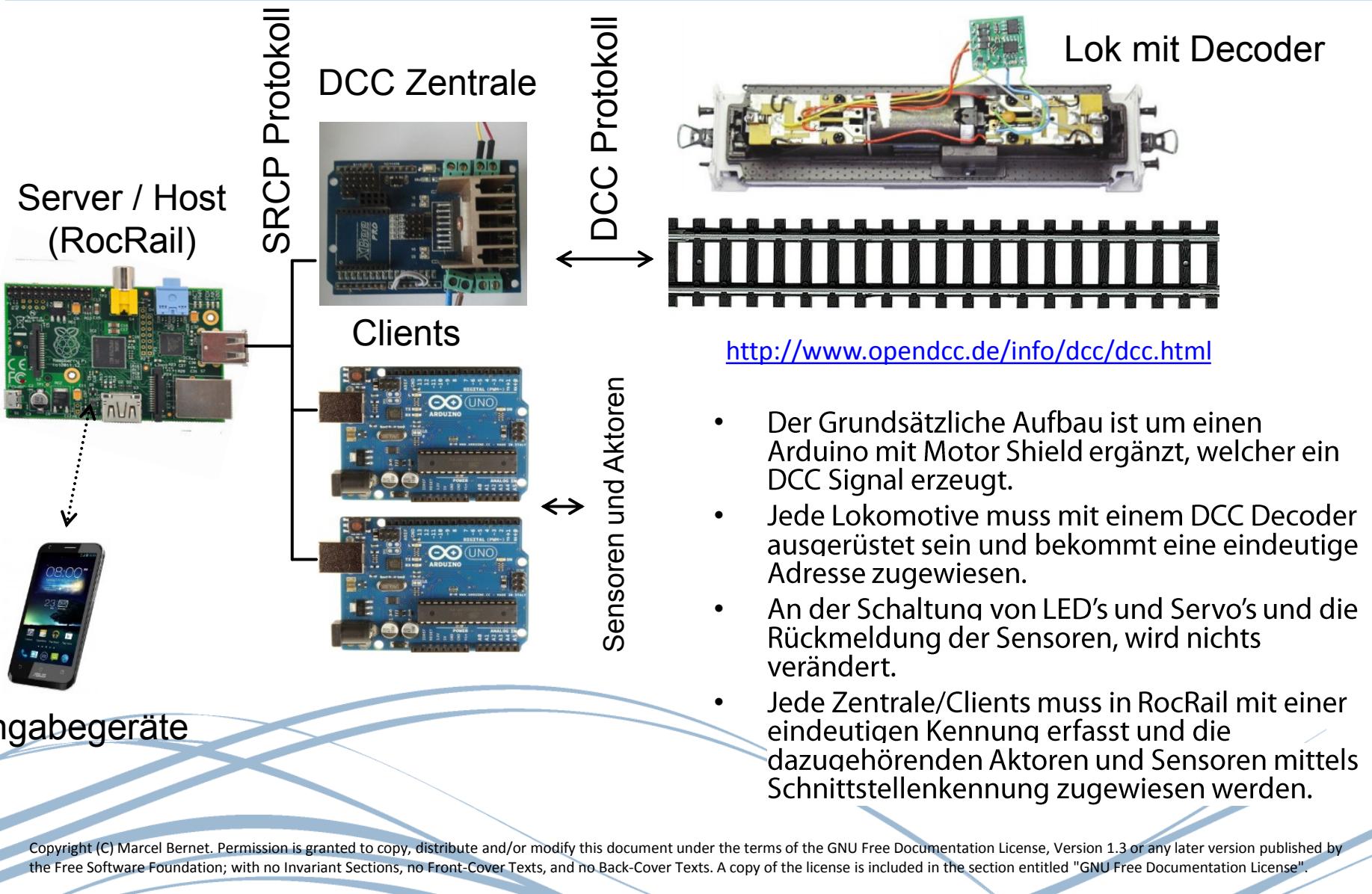
- Erweitert die Steuerung durch weitere Eingabegeräte wie ein Smartphone.
  - Installiert andRoc o.ä. auf Eurem Smartphone
  - Startet die App mit folgenden Verbindungseinstellungen:
    - Host: <IP-Adresse des Raspberry Pi>
    - Port: 8051

# Modelleisenbahn – mehr Kilobytes

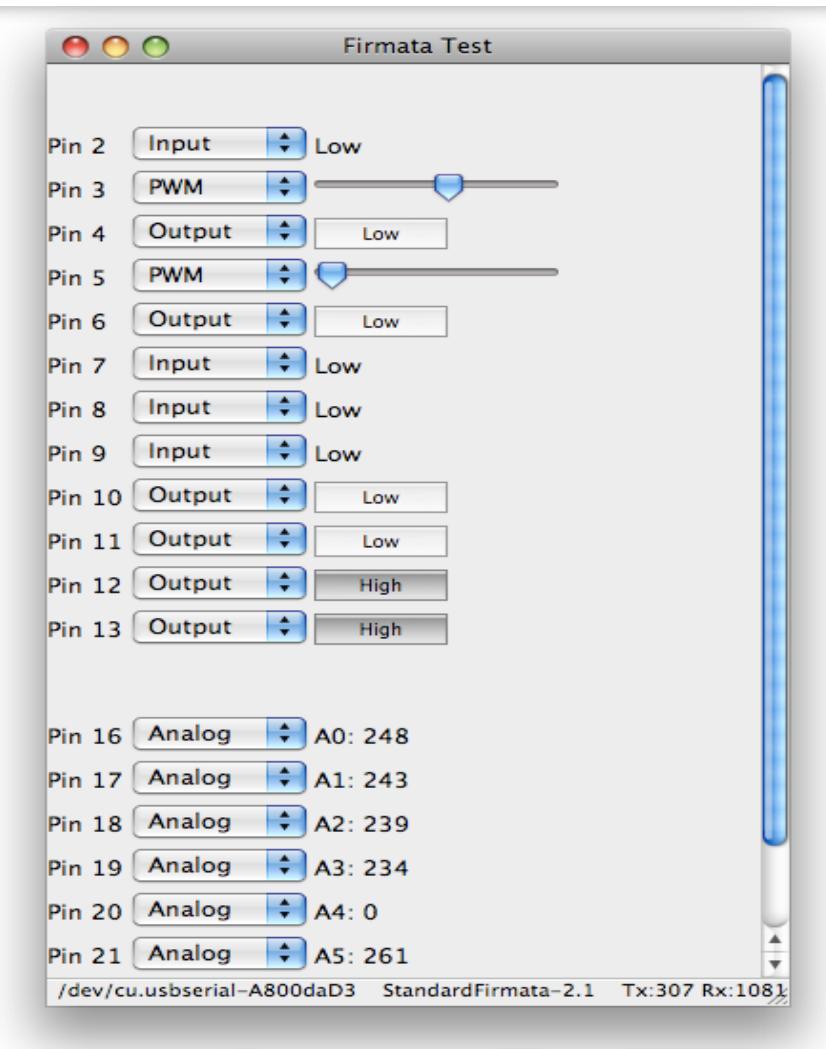


- Digitale Modelleisenbahnsteuerung von Lokomotiven:
  - <https://github.com/mc-b/microSRCP/wiki/Steuerungdigital>
- Ändern der bereitgestellten Geräte auf dem Arduino Board und neue Shield's einbinden:
  - <https://github.com/mc-b/microSRCP/wiki/Steuerungkonfiguration>
  - <https://github.com/mc-b/microSRCP/wiki/Steuerungerweiterungen>
- Debugging, Logging, Layout der Sourcen ...
  - <https://github.com/mc-b/microSRCP/wiki/Entwickler>
- Details zum SRCP Protokoll und Verwendung des I2C Buses:
  - <https://github.com/mc-b/microSRCP/wiki/Srcp-protokoll>
  - <https://github.com/mc-b/microSRCP/wiki/I2c-bus>

# Grundsätzlicher Aufbau - Digital



# Hausautomation: Firmata



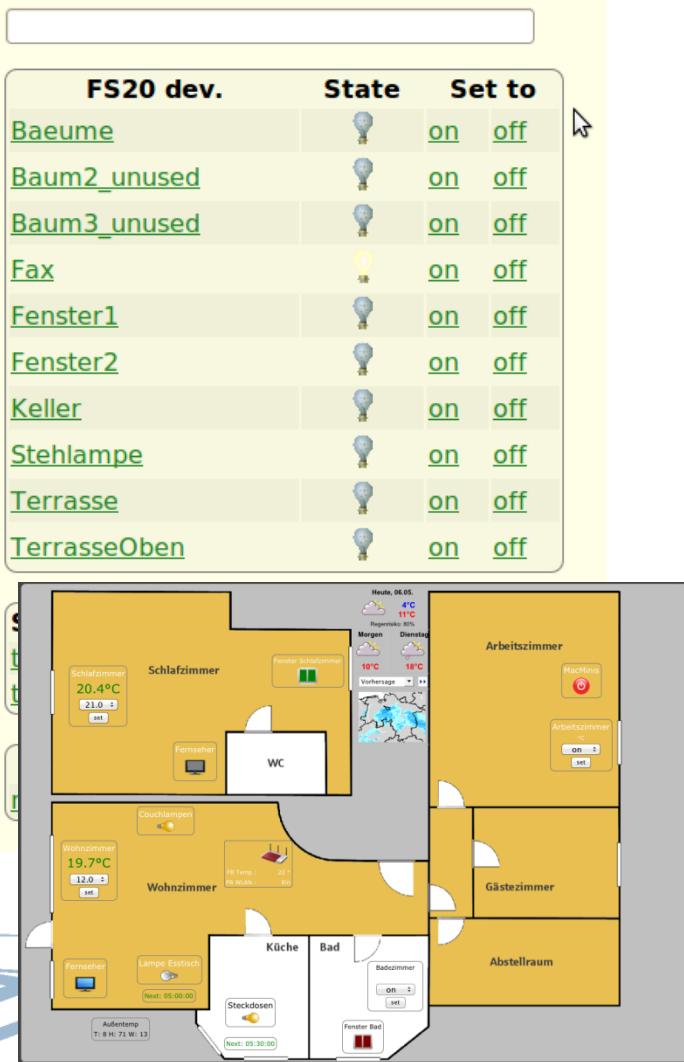
- Firmata ist ein generisches Protokoll für die Kommunikation mit Mikrocontrollern und einem Host-Computer.
- Firmata Libraries existieren für verschiedene Sprachen C, Java ..
- Das Ziel von Firmata ist es, Menschen die vollständige Kontrolle über die Arduino Plattform vom Host-Computer zu geben. D.h. nicht der Firmata Sketch legt fest welche Geräte vom Arduino angesprochen werden sollen, sondern die Konfiguration auf dem Host (hier fhem).
- Das Bild links zeigt das Firmata Testprogramm, wo für jeden Pin dessen Funktion (Input, Output, Servo etc). festgelegt werden kann.

# Hausautomation: fhem



[Alarm](#)  
[Bewaesserung](#)  
[Energiemonitor](#)  
[Heizung](#)  
[Lampen](#)  
[Meteo](#)  
[Meteo-archive](#)  
[Plots](#)  
[Rolladen](#)  
[GEN](#)  
[All together](#)

[Howto](#)  
[FAQ](#)  
[Details](#)  
[Examples](#)  
[Edit files](#)



The screenshot shows the fhem web interface. On the left, there is a sidebar with various links. In the center, there are two main sections: a table of device status and a floor plan visualization.

FS20 dev.	State	Set to
Baeume		<a href="#">on</a> <a href="#">off</a>
Baum2_unused		<a href="#">on</a> <a href="#">off</a>
Baum3_unused		<a href="#">on</a> <a href="#">off</a>
Fax		<a href="#">on</a> <a href="#">off</a>
Fenster1		<a href="#">on</a> <a href="#">off</a>
Fenster2		<a href="#">on</a> <a href="#">off</a>
Keller		<a href="#">on</a> <a href="#">off</a>
Stehlampe		<a href="#">on</a> <a href="#">off</a>
Terrasse		<a href="#">on</a> <a href="#">off</a>
TerrasseOben		<a href="#">on</a> <a href="#">off</a>

The floor plan visualization shows a house layout with rooms including Schlafzimmer, Wohnzimmer, Küche, Bad, Arbeitszimmer, and Gästezimmer. Various sensors and actuators are represented by icons and labeled with their current state or value. For example, the Schlafzimmer has a temperature of 20.4°C and a set point of 21.0. The Arbeitszimmer has a light switch labeled "Arbeitszimmer" with "on" and "off" buttons. The Küche has a temperature of 19.7°C and a set point of 12.0. The Bad has a temperature of 20.0°C and a set point of 20.0. The Gästezimmer has a light switch labeled "Gästezimmer" with "on" and "off" buttons. The floor plan also includes a WC, a Fernseher, a Steckdose, and a Fenster. Weather information for the day (Heute, Morgen, Dienstag) and a forecast are also displayed at the top right of the floor plan.

- **FHEM** ist ein Perl-basiertes Server-programm für die Hausautomation, der zur automatisierten Bedienung von Aktoren wie zum Beispiel Lichtschaltern oder Heizung sowie der Aufzeichnung von Sensorinformationen wie Raumtemperatur oder Luftfeuchtigkeit dient.
- [http://fhem.de/fhem\\_DE.html](http://fhem.de/fhem_DE.html)
- <http://www.fhemwiki.de/wiki/Arduino>

# Aufgabe 2a: Hausautomation



The screenshot shows the fhem web interface. On the left, there's a sidebar with links: Save config, Schlafzimmer, Unsorted, Wohnzimmer, Everything, Logfile, Howto, Commandref, Wiki / Forum, Edit files (circled in red), Select style, and Event monitor. The main area has tabs for SecurityCheck, Schlafzimmer, and Wohnzimmer. A central panel shows "Own modules and helper files" with a file named "99\_Utils.pm". Below it is a "styles" section with several CSS files: darkfloorplanstyle.css, darkstyle.css, darksvg\_defs.svg, darksvg\_style.css, darktouchpadstyle.css, darktouchpadsvg\_defs.svg, darktouchpadsvg\_style.css, and floorplanstyle.css. At the bottom, there's a code editor window displaying Arduino configuration code:

```
# definiere FRM als IO-Device - Baudrate 57600
# ist default im StandardFirmata Sketch
define Arduinol FRM /dev/ttyUSB0@57600
attr Arduinol loglevel 6
attr Arduinol sampling-interval 1000

# Led 4 an Device Arduinol im Wohnzimmer
define Led4 FRM_OUT 4
attr Led4 IODev Arduinol
attr Led4 stateFormat value
attr Led4 room Wohnzimmer
```

- Baut mit fhem, Raspberry Pi (Alternativ VMWare Umgebung) und Arduino eine kleine Haussteuerung auf.

- Startet den Browser und die fhem Oberfläche.
- Wählt Edit Files und fhem.cng
- Ergänzt mindestens die nebenstehenden Zeilen (je nach Arduino ist der Serielle Port zu ändern)
- Startet fhem neu, falls es nicht selber startet  
sudo service fhem stop  
sudo service fhem start

<http://www.fhemwiki.de/wiki/Arduino>

# Aufgabe 2b: Hausautomation



```
# Pin A1 und A2 als Input zum ein und
# ausschalten von Led 4
define Input1 FRM_IN 15
attr Input1 room Wohnzimmer
define Input2 FRM_IN 16
attr Input2 room Wohnzimmer

define Input1Notify notify Input1 set Led4 on
define Input2Notify notify Input2 set Led4 off

# Led 4 alle 5 Sekunden ein und ausschalten
define at1 at +*00:00:10 set Led4 on
define at2 at +*00:00:15 set Led4 off
```

- Erweitert die Hausautomation um:
  - Reaktion auf externe Ereignisse (Input, Sensor)
  - automatisierte Abläufe (z.B. LED alle 5 Sekunden ein-/ausschalten)
- Weitere Ideen:
  - [http://www.fhemwiki.de/wiki/Kategorie:Code Snippets](http://www.fhemwiki.de/wiki/Kategorie:Code_Snippets)

# Aufgabe 2c: Hausautomation



- Erweitert die Steuerung durch weitere Eingabegeräte wie ein Smartphone.
  - Installiert andFhem o.ä. auf Eurem Smartphone
  - Startet die App mit folgenden Verbindungseinstellungen:
    - URL: <IP-Adresse des Raspberry Pi>:8083/fhem

# Hausautomation – 220 Volt!



- Raspberry Pi + Arduino Rboard oder irgend ein Relais Shield
  - <http://shop.boxtec.ch/rboard-arduino-mit-relais-kanaelen-p-40972.html>
- Raspberry Pi + 433MHz RF Link Kit + <https://github.com/r10r/rcswitch-pi> und z.B. <http://shop.boxtec.ch/codec-adaptive-wireless-relais-433mhz-p-41487.html>
- Oder kommerzielle Produkte wie:
  - <http://www.busware.de/tiki-index.php>
  - <http://conrad.ch> – nach FS20 suchen
  - Weitere: [weitere http://fhem.de/fhem\\_DE.html#Hardware](http://fhem.de/fhem_DE.html#Hardware)
- **Alle Angaben ohne Gewähr und nicht geprüft!**



# Hausautomation- mehr Kilobytes



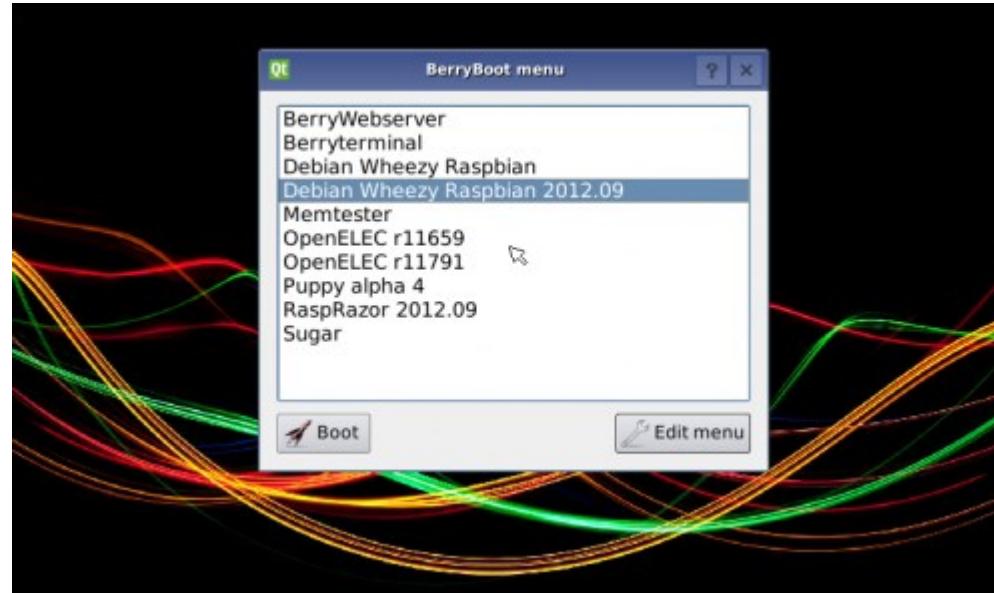
- <https://github.com/mc-b/microHOME/wiki>



# Aufgabe 3: Raspberry Pi Installieren

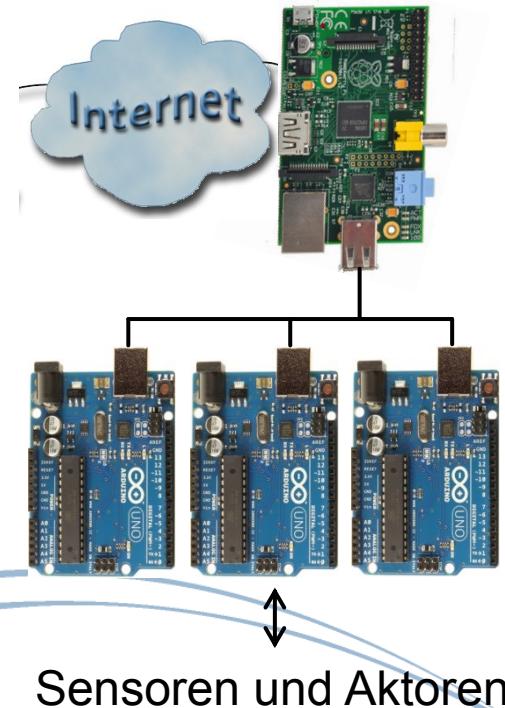


- Für eine Headless Installation eines Raspberry Pi mittels Berry Boot durch
  - [http://www.berryterminal.com/doku.php/berryboot/headless\\_installation](http://www.berryterminal.com/doku.php/berryboot/headless_installation)



# Zusammenfassung

- Kombiniert man physische Dinge (Modelleisenbahn, Lampe etc.) mit Raspberry Pi und Arduino's – entsteht ein Netzwerk von „**intelligenten Gegenständen**“ welches mit dem Internet verbunden werden kann.



# Agenda



- Vormittag
  - Einleitung „Internet der Dinge“, Sensoren, Aktoren – 20'
  - Die Arduino Plattform (Hardware, einfache Anwendungen, Sensoren) – 90'
  - Die Raspberry Pi Plattform – 20'
- Nachmittag
  - Arduino's und Raspberry Pi verbinden – 10'
  - Aufbau einer Steuerung anhand zweier konkreter Beispiele (Hausautomation und Modelleisenbahn-Steuerung) – 120'
  - Raspberry Pi, Arduino mit dem Internet verbinden – 30'

# Variante A: Dynamisches DNS

([http://de.wikipedia.org/wiki/Dynamic\\_DNS](http://de.wikipedia.org/wiki/Dynamic_DNS))



## DYNDNS HOSTNAMES

HOSTNAME	SERVICE	DETAILS
[REDACTED]	Host	188.154.159
[REDACTED]	Host	188.154.159

Portfreigaben Speicher Fernwartung Dynamic DNS VPN

An FRITZ!Box angeschlossene Computer sind sicher vor unerwünschten Zugriffen aus dem Internet. Für einige Anwendungen wie z.B. Online-Spiele oder das Filesharing-Programm eMule muss Ihr Computer jedoch für andere Teilnehmer des Internets erreichbar sein. Durch Portfreigaben erlauben Sie solche Verbindungen.

Aktiv	Bezeichnung	Protokoll	Port	an Computer	an Port	
<input checked="" type="checkbox"/>	HTTPS-Server	TCP	443	mcbpx01	443	
<input type="checkbox"/>	SSH	TCP	22	mcbpx01	22	

Portfreigaben Speicher Fernwartung Dynamic DNS VPN

Über Dynamic DNS können Anwendungen und Dienste, für die in der FRITZ!Box-Firewall Portfreigaben eingerichtet wurden, über einen festen Domännamen aus dem Internet erreicht werden, obwohl sich die öffentliche IP-Adresse des Internetanschlusses ändert.

Dynamic DNS benutzen  
Geben Sie die Anmelddaten für Ihren Dynamic DNS-Anbieter an.

Dynamic DNS-Anbieter:  Neuen Domännamen anmelden

Domainname:

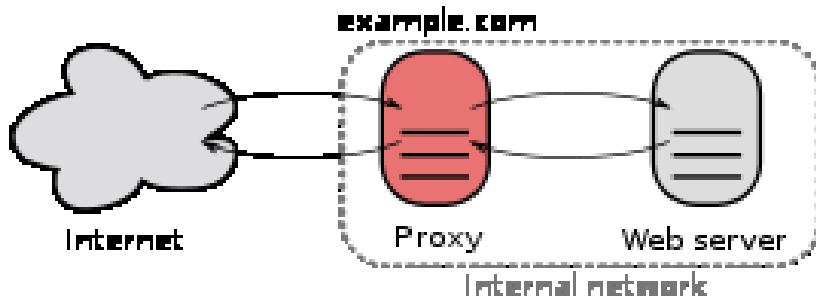
Benutzername:

Kennwort:

- Dynamisches DNS oder DDNS ist eine Technik, um Domains im Domain Name System dynamisch zu aktualisieren. So ist der Rechner immer unter demselben Domänennamen erreichbar, auch wenn die aktuelle IP-Adresse für den Nutzer unbekannt ist.
- Voraussetzung
  - Account bei einem Dynamic DNS Anbieter (z.B. DynDNS).
  - DynDNS fähiger Router, z.B. Fritz!Box
- Installation
  - Account, z.B. bei DynDNS einrichten
  - Port (fhem: 8083, RocRail: 8051) im Router freigeben
  - Router mit Dynamic DNS Anbieter verbinden.

# Variante B: mit vorgesetztem Reverse Proxy

([http://de.wikipedia.org/wiki/Reverse\\_Proxy](http://de.wikipedia.org/wiki/Reverse_Proxy))

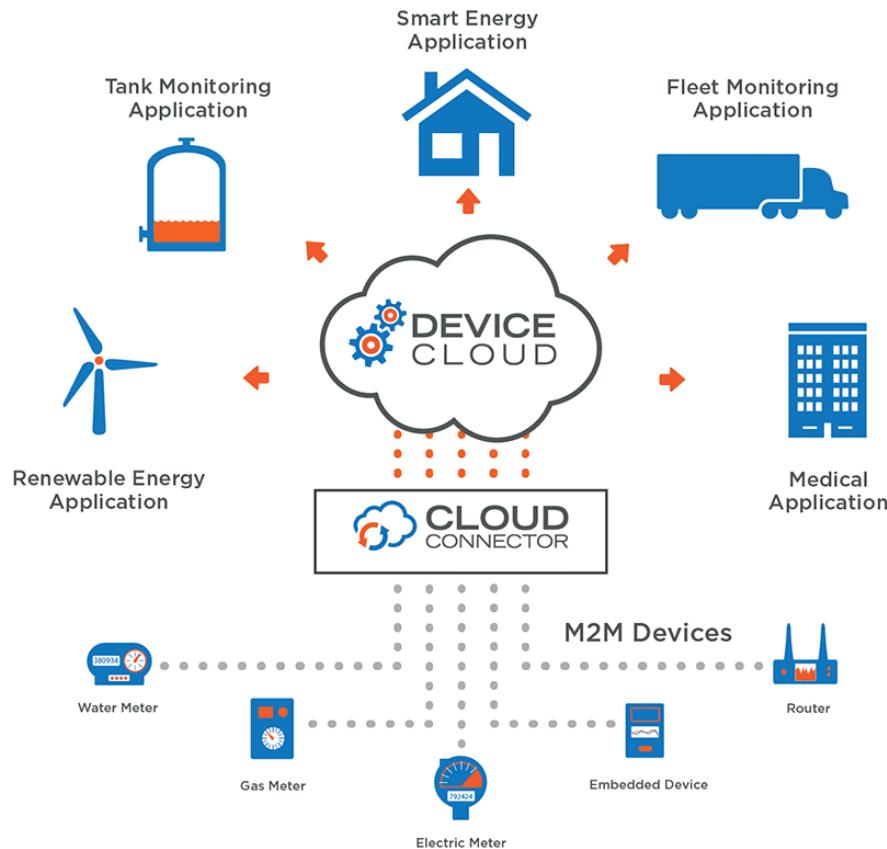


Apache Konfiguration in sites-enabled/default.ssh für fhem:

```
<IfModule mod_ssl.c>
<VirtualHost *:443>
    # Allgemeine Proxy Einstellungen
    ProxyRequests Off
    <Proxy *>
        Order deny,allow
        Allow from all
    </Proxy>
    # Weiterleitungen fhem
    ProxyPass /fhem http://<server>:8083/fhem
    ProxyPassReverse /fhem http://<server>:8083/fhem
    ....
```

- Grundsätzlich gleiche Variante wie A.
- Aus Sicherheits-gründen wird ein Apache Server, Konfiguriert als Reverse Proxy vorgesetzt.
- Voraussetzungen:
  - Wie Variante A, erweitert um einen zusätzlichen Gerät (z.B. Raspberry Pi) als Reverse Proxy.
- Installation:
  - Erster Teil wie Variante A, aber es wird nur Port 443 (https) auf dem Reverse Proxy freigegeben.
  - Installation minimales Linux System mit Apache Server (`apt-get install apache2`)

# Variante C: Cloud



- Verbindung erfolgt vom lokalen Gerät in die Cloud. Eine Verbindung Cloud -> Gerät ist ausgeschlossen (Security).
- Vorgehen (Grob)
  - Account bei Cloud Anbieter lösen
  - Gerät(e) eintragen
  - Client Software und i.d.R. Key herunterladen
  - Software auf Geräte laden und mit Cloud verbinden.

# Aufgabe 4: fhem absichern



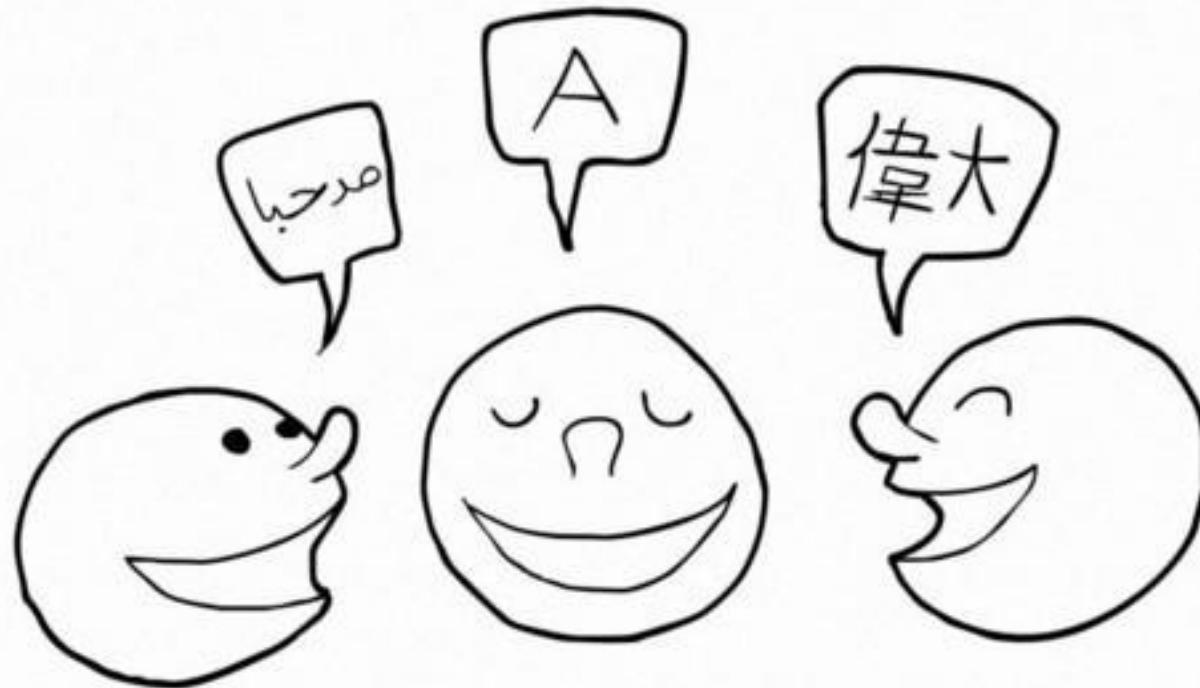
```
<Proxy *>
    AuthType Basic
    AuthName "Password Required"
    AuthUserFile /home/httpd/etc/passwd
    Require valid-user
    Allow from 127.0.0.1
</Proxy>
ProxyPass        /fhem  http://localhost:8083/fhem
ProxyPassReverse /fhem  http://localhost:8083/fhem
```

- Sichert fhem durch Voranschaltung eines Apache Servers ab.
  - [http://fhem.de/HOWTO\\_DE.html](http://fhem.de/HOWTO_DE.html) - Kapitel Sicherheit

# Zusammenfassung

- **Das Internet der Dinge** (auch englisch Internet of Things) **beschreibt, dass der (Personal) Computer zunehmend als Gerät verschwinden und durch „intelligente Gegenstände“ ersetzt wird.**
- Mit dem Raspberry Pi und der Arduino Plattform wird diese Vision Wirklichkeit!

# Fragen?



# Fragen nach dem Kurs



- Für Fragen nach dem Kurs sind folgende Links und Foren hilfreich:
- Arduino
  - <http://www.arduino.cc> - Hauptseite
  - <http://forum.arduino.cc/> - Forum
  - <http://playground.arduino.cc//Deutsch/HomePage> - Playground
- fhem
  - [http://fhem.de/fhem\\_DE.html](http://fhem.de/fhem_DE.html) - Hauptseite
  - <http://forum.fhem.de/> - Forum
  - <http://www.fhemwiki.de/wiki/Hauptseite> - Wiki
- microSRCP
  - <https://github.com/mc-b/microSRCP/wiki> - Wiki
  - <https://github.com/mc-b/microSRCP> - Code
  - <http://forum.rocrail.net/viewtopic.php?t=5930&highlight=%> - Forum RocRail
- Raspberry Pi
  - <http://www.raspberrypi.org/> - Hauptseite
  - [http://www.elinux.org/R-Pi\\_Hub](http://www.elinux.org/R-Pi_Hub) - Wiki
  - <http://www.raspberrypi.org/phpBB3/> - Forum