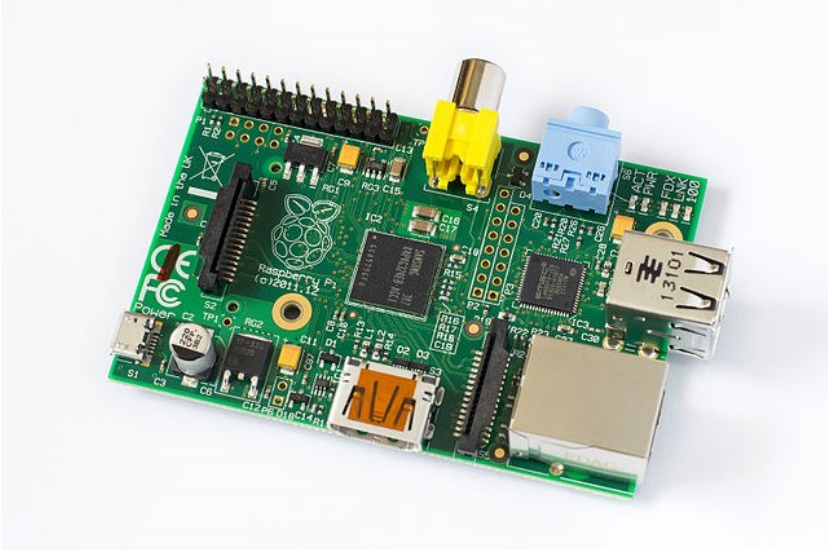
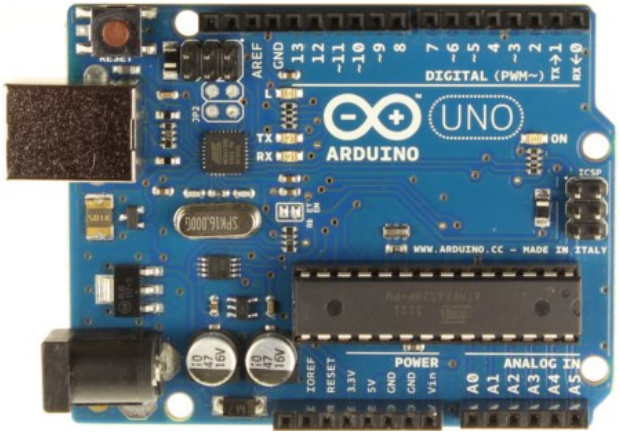




Vernetzte "Dinge" mit Raspberry Pi und Arduino

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 Modellbahnern 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

- Einleitung „Internet der Dinge“, Sensoren, Aktoren – 5′
- Die Arduino Plattform (Hardware, einfache Anwendungen, Sensoren) – 45′
- Die Raspberry Pi Plattform – 5′
- Arduino’s und Raspberry Pi verbinden – 5′
- Aufbau einer Steuerung anhand zweier konkreter Beispiele (Hausautomation und Modelleisenbahn-Steuerung) – 60′

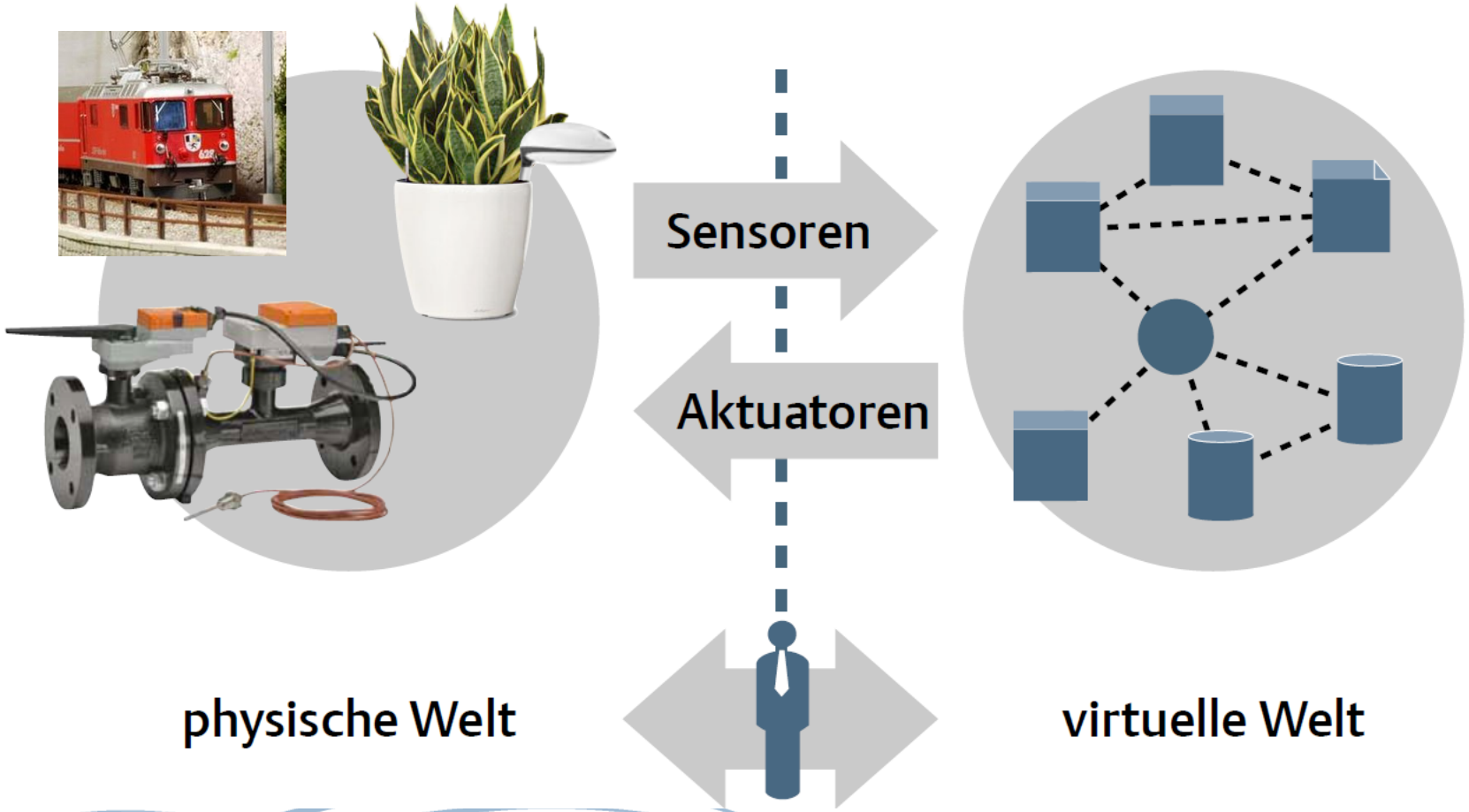
Was haben diese Dinge gemeinsam?

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/)

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/)

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

- Einleitung „Internet der Dinge“, Sensoren, Aktoren – 5′
- Die Arduino Plattform (Hardware, einfache Anwendungen, Sensoren) – 45′
- Die Raspberry Pi Plattform – 5′
- Arduino’s und Raspberry Pi verbinden – 5′
- Aufbau einer Steuerung anhand zweier konkreter Beispiele (Hausautomation und Modelleisenbahn-Steuerung) – 60′

Arduino - Übersicht

Arduino Uno

Arduino Leonardo

Arduino Due

Arduino Yún

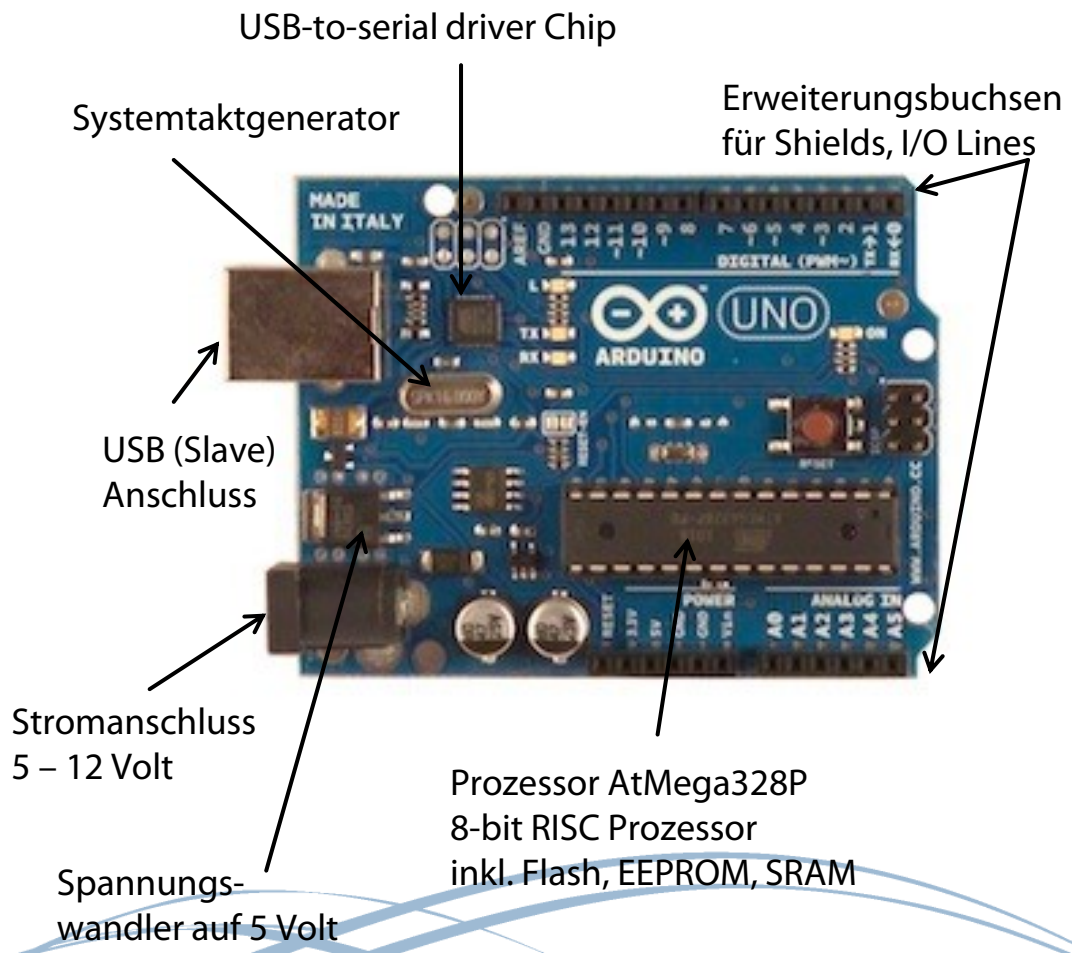
Arduino Robot

Arduino Esplora

- 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

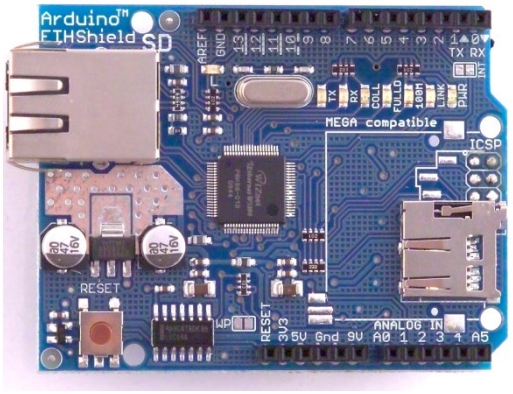
Arduino - Aufbau



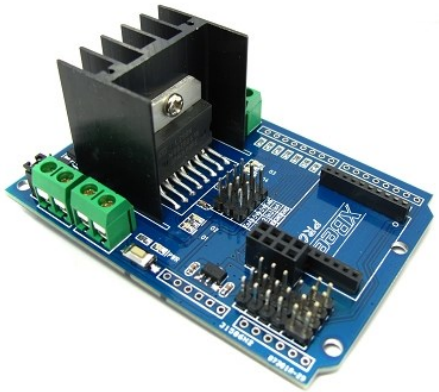
- Die Arduino-Plattform ist eine aus Soft-und Hardware bestehende **Physical-Computing-Plattform**.
- Physical Computing bedeutet im weitesten Sinne, interaktive, physische Systeme durch die Verwendung von Hardware und Software zu erstellen. Diese Systeme reagieren auf Ereignisse (Sensoren) in der realen,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

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

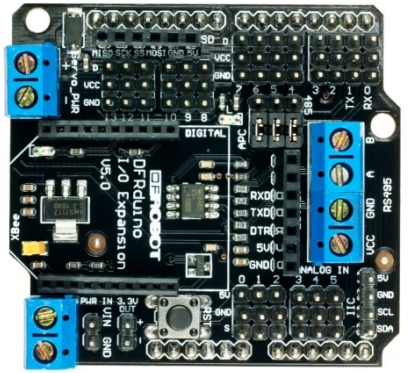
Ethernet-Shield



MotoMama (Motor) Shield



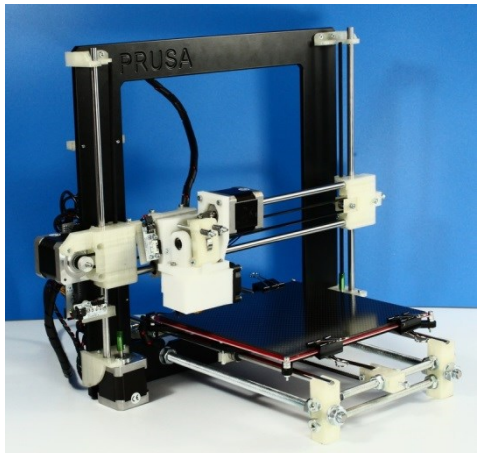
IO Expansion Shield



- 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 Bahnhofsdurchsagen
  - Schalten von 220V Verbrauchern
  - IO Expansion Shields für Anschluss von Servo, LED, Rückmelder, I2C, RS 485 etc.

Arduino - Einsatzgebiete

NinjaBlocks: Hausautomation

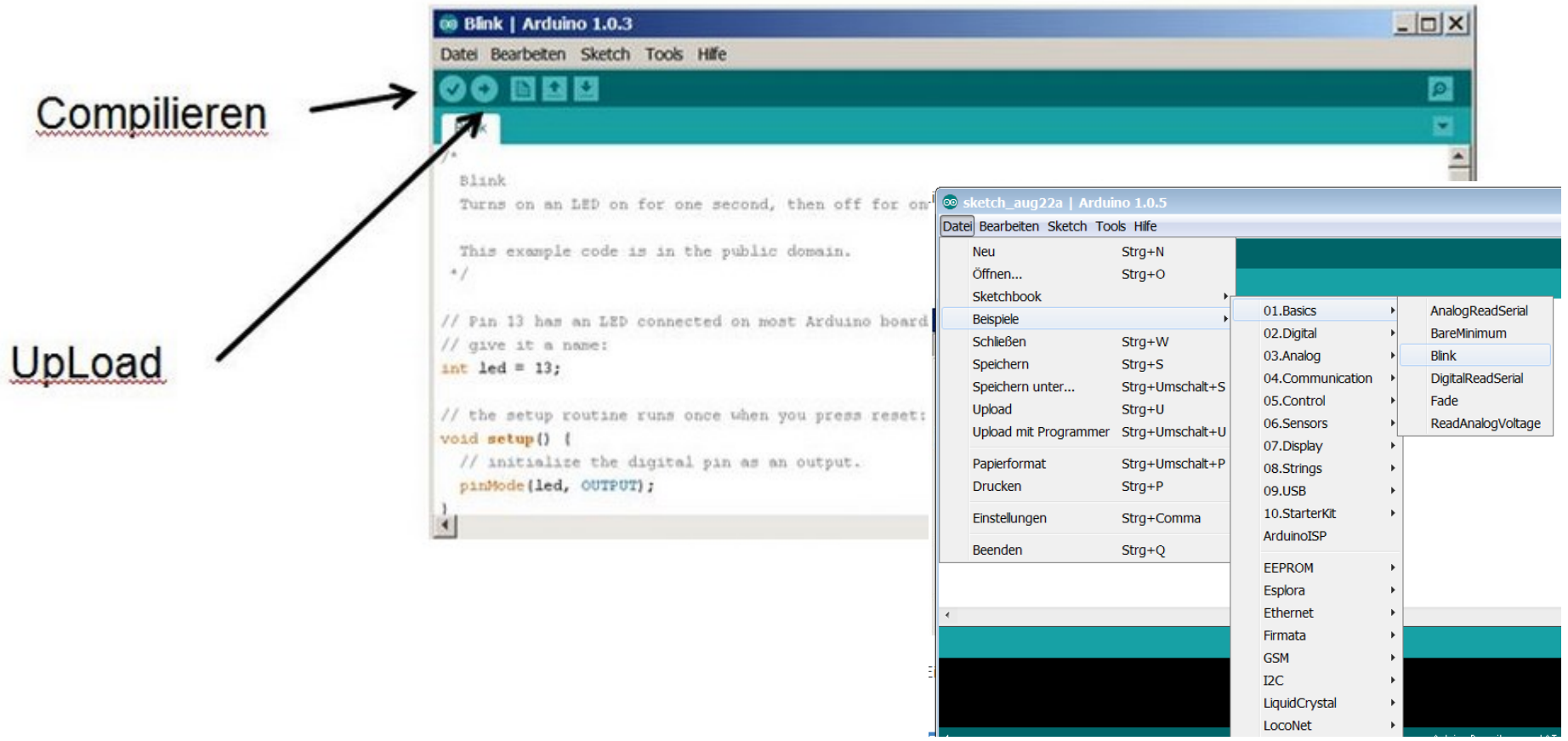


RAMPS: Steuerung von 3D Druckern

LiliPad: integriert in Kleidung

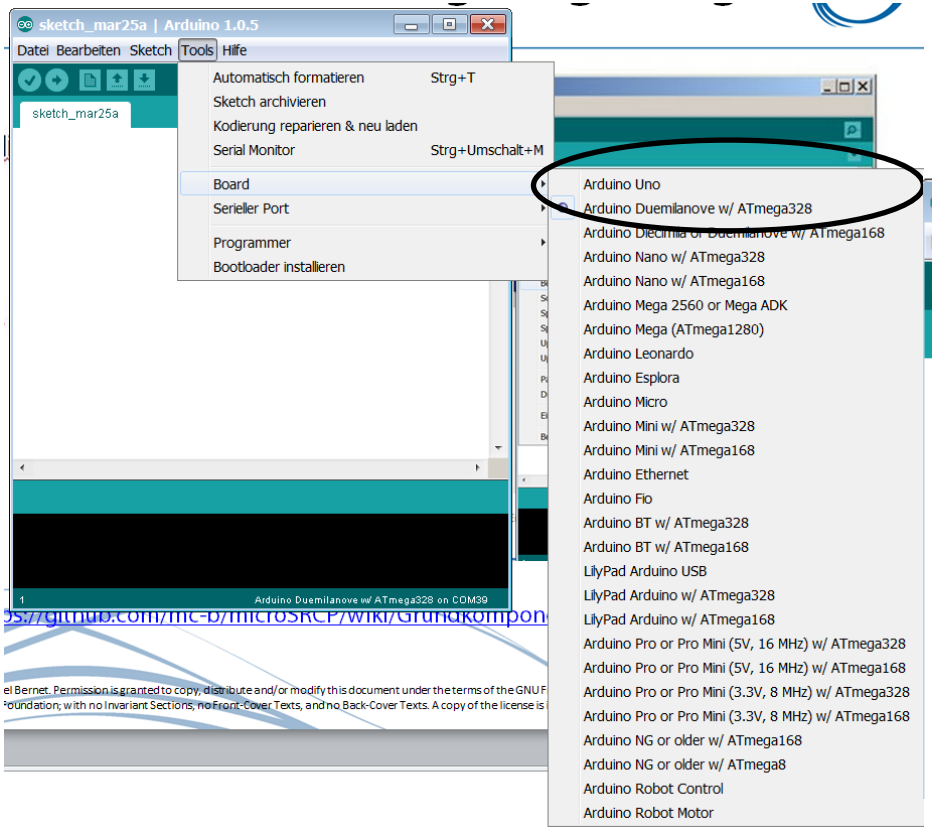
Arduino ADK: Steuerung von USB Geräten

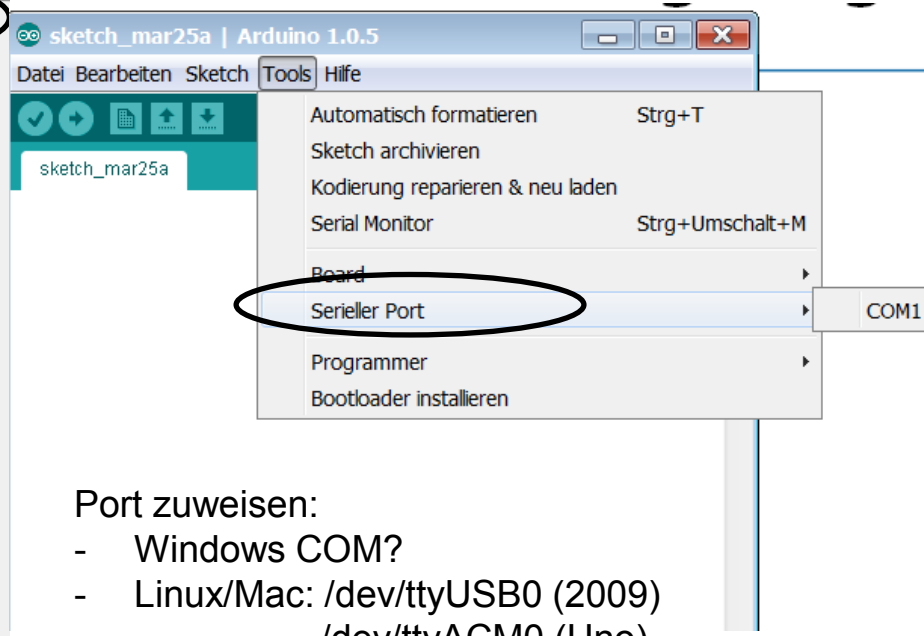
Arduino Entwicklungsumgebung (1)



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

Arduino Entwicklungsumgebung (2)



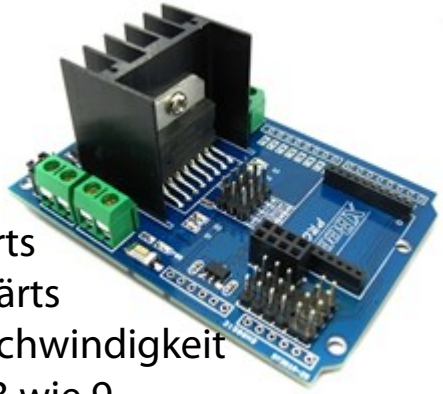


Port zuweisen:

- Windows COM?
- Linux/Mac: /dev/ttyUSB0 (2009)
  /dev/ttyACM0 (Uno)

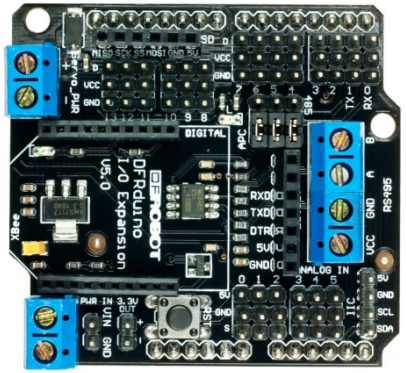
- https://github.com/mc-b/microSRCP/wiki/Grundkomponenten#wiki-Entwicklungsumgebung

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)

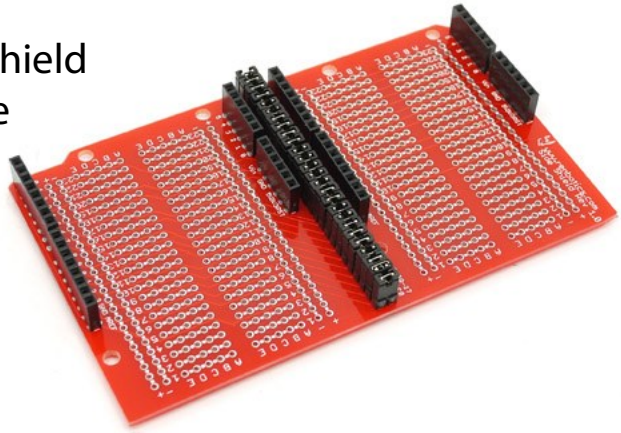


IO Expansion Shield

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

Seedstudio Side Shield

- Vordoppelt 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.

Struktur eines Sketches

Eine Methode setup() wo allgemeine Initialisierungen erledigt werden

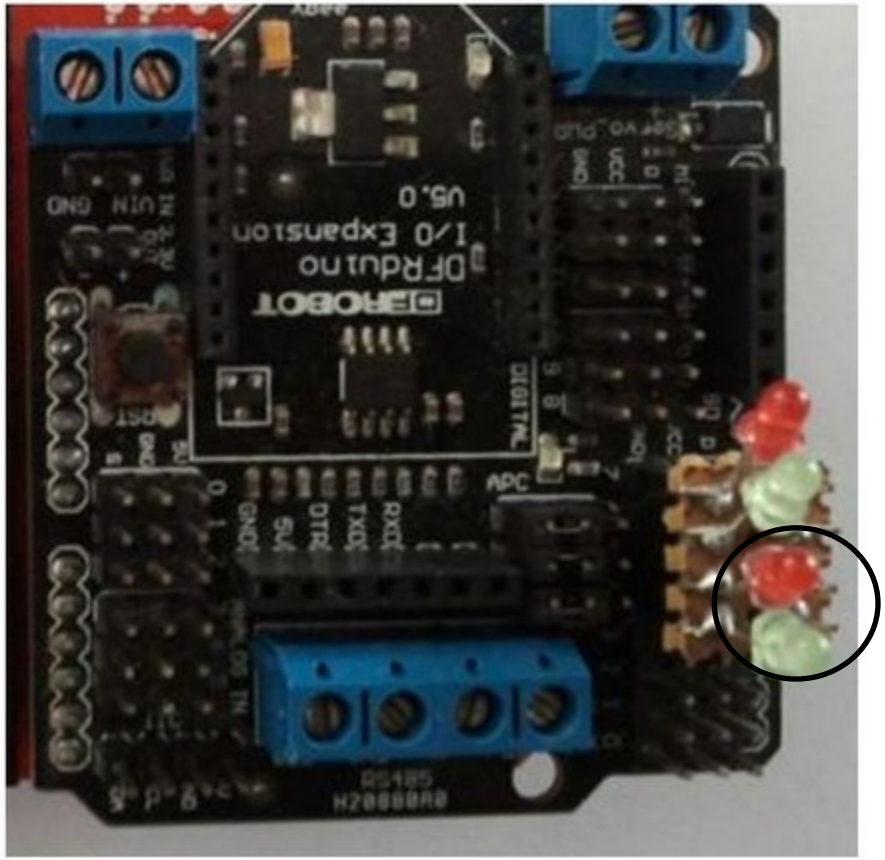
```
void setup()
{
    // allgemeine Initialisierungen wie z.B. ein Pin auf Output oder Input zu setzen, Bsp:
    pinMode( 13, OUTPUT );
    pinMode( A0, INPUT_PULLUP );
}
```

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
}
```

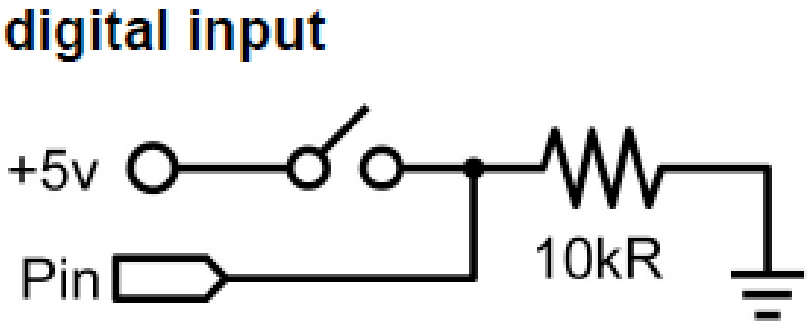
Die setup() Methode wird nur einmal, am Anfang, durchlaufen, die loop() Methode endlos.

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)

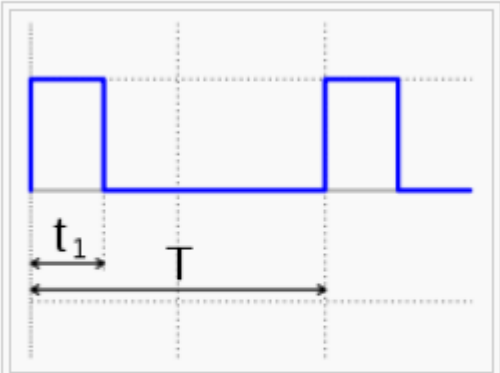


```
int sensorPin = A0;                    // Input Pin, z.B. mit Taster
int ledPin = 4;                        // LED

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

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

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

LED Dimmen - PWM (Aktor)

pwm output



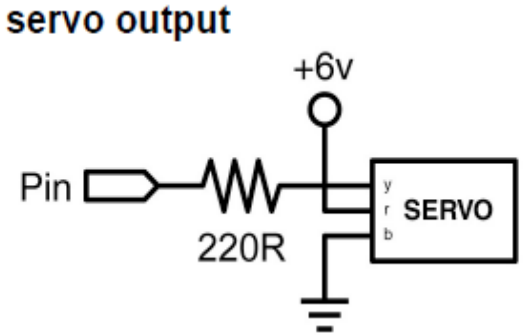
PWM-Signal mit einem Tastgrad

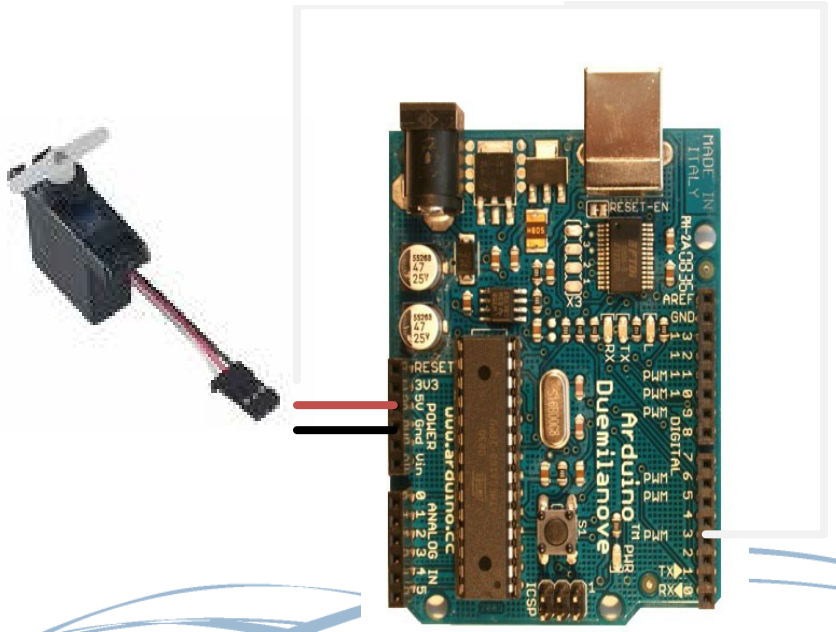
$\frac{t_1}{T} = 0{,}25 = 25\,\%$

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

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

Ansteuerung von Servo's (Aktor), Libraries

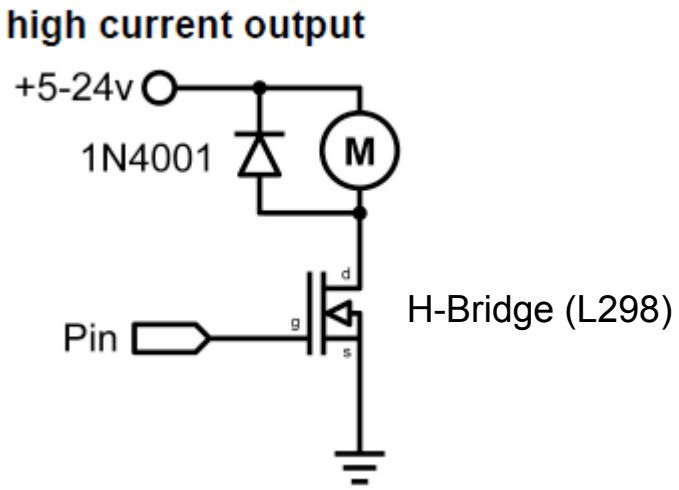




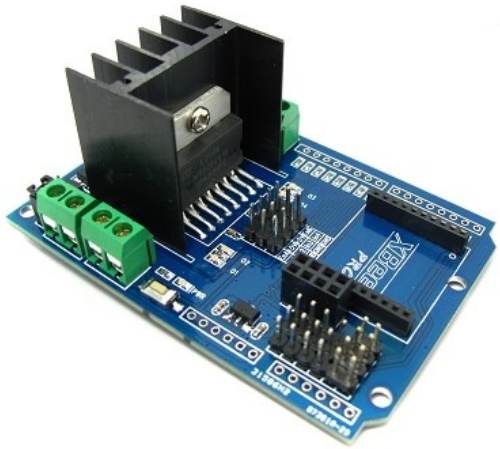
```
// Sweep
#include <Servo.h>   // Bekanntmachung der Servo Library
Servo myservo;       // Erstellt ein Servo Object zur Steuerung des Serv

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 Millissekunden 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 ausstecken und richtigen Pin verwenden!

Ansteuerung von Motoren (Aktor), Shields



H-Bridge (L298)



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(speed, 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

- setup()
- loop()

Control Structures

- if
- if...else
- for
- switch case
- while
- do... while
- break
- continue
- return
- goto

Further Syntax

- ; (semicolon)
- {} (curly braces)
- // (single line comment)
- /* */ (multi-line comment)
- #define
- #include

Arithmetic Operators

- = (assignment operator)
- + (addition)
- - (subtraction)
- * (multiplication)
- / (division)
- % (modulo)

Comparison Operators

- == (equal to)
- != (not equal to)
- < (less than)
- > (greater than)
- <= (less than or equal to)
- >= (greater than or equal to)

Boolean Operators

- && (and)
- || (or)
- ! (not)

Pointer Access Operators

- * dereference operator
- & reference operator

Bitwise Operators

- & (bitwise and)
- | (bitwise or)
- ^ (bitwise xor)
- ~ (bitwise not)
- << (bitshift left)
- >> (bitshift right)

Compound Operators

- ++ (increment)
- -- (decrement)
- += (compound addition)
- -= (compound subtraction)
- *= (compound

Variables

Constants

- HIGH | LOW
- INPUT | OUTPUT | INPUT_PULLUP
- true | false
- integer constants
- floating point constants

Data Types

- void
- boolean
- char
- unsigned char
- byte
- int
- unsigned int
- word
- long
- unsigned long
- short
- float
- double
- string - char array
- String - object
- array

Conversion

- char()
- byte()
- int()
- word()
- long()
- float()

Variable Scope & Qualifiers

- variable scope
- static
- volatile
- const

Utilities

- sizeof()

Functions

Digital I/O

- pinMode()
- digitalWrite()
- digitalRead()

Analog I/O

- analogReference()
- analogRead()
- analogWrite() - *PWM*

Due only

- analogReadResolution()
- analogWriteResolution()

Advanced I/O

- tone()
- noTone()
- shiftOut()
- shiftIn()
- pulseIn()

Time

- millis()
- micros()
- delay()
- delayMicroseconds()

Math

- min()
- max()
- abs()
- constrain()
- map()
- pow()
- sqrt()

Trigonometry

- sin()
- cos()
- tan()

Random Numbers

- randomSeed()
- random()

Bits and Bytes

- lowByte()
- highByte()
- bitRead()
- bitWrite()
- bitSet()
- bitClear()
- bit()

External Interrupts

- attachInterrupt()
- detachInterrupt()

Interrupts

- interrupts()
- noInterrupts()

Standard Libraries

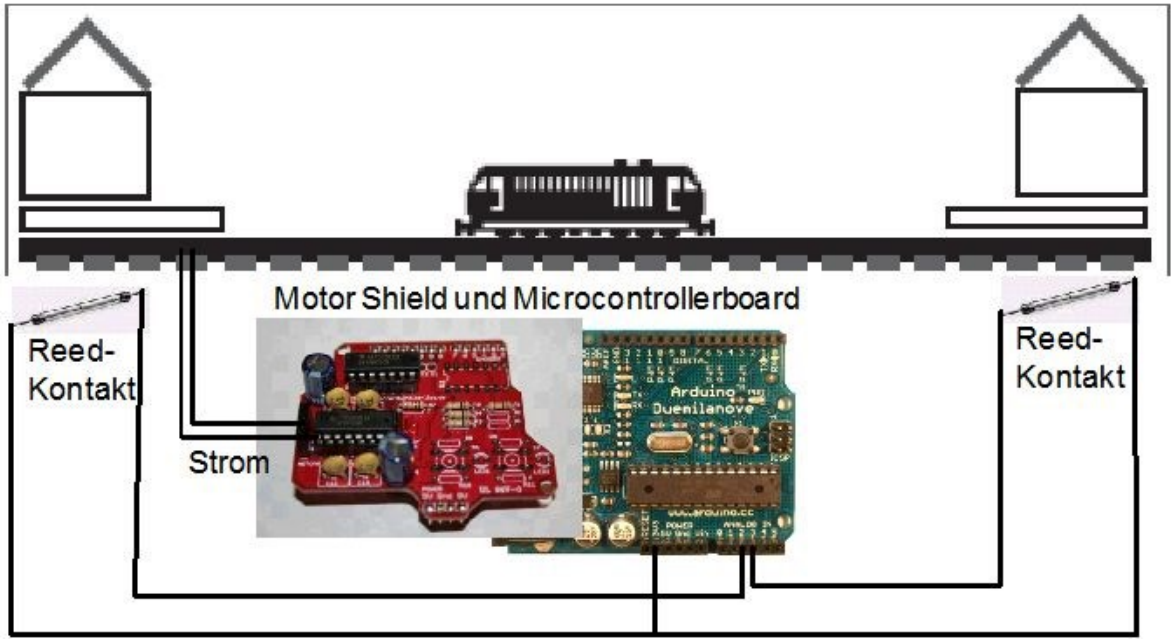
- EEPROM - reading and writing to "permanent" storage
- Ethernet - for connecting to the internet using the Arduino Ethernet Shield
- Firmata - for communicating with applications on the computer using a standard serial protocol.
- GSM - for connecting to a GSM/GRPS network with the GSM shield.
- LiquidCrystal - 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

Sensoren

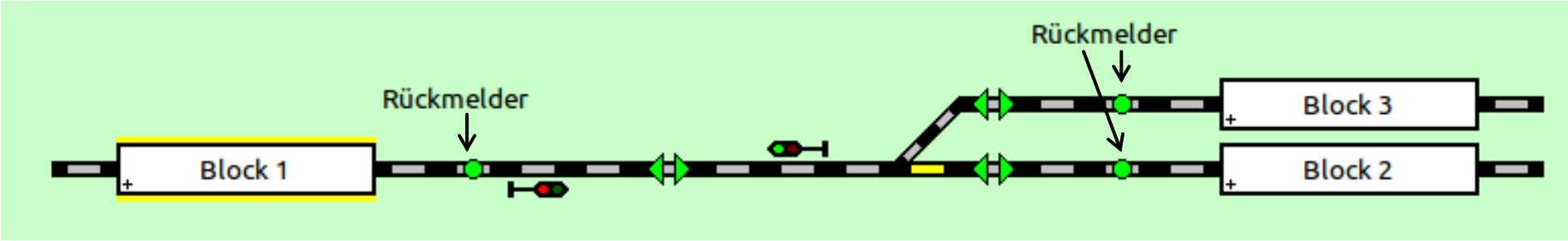
- Sensoren erlauben es auf externe Meldungen zu reagieren und ermöglichen so automatisierte Abläufe (**Pendelzug**)
- Arduino kann Analoge, Digitale, Serielle (RS-232C) und I2C Meldungen auswerten.
- Beispiele für Sensoren sind:
  - Temperator-Sensor
  - 2/3 Axis Sensor (analog Smartphones), Tilt-Sensor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

Sensoren (Rückmelder) - Übungsmodul



- 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 Reed-Kontakte (Reed vom englischen Röhrchen) welche direkt an das Board (A0-A2) angeschlossen und mittels Magnet geschaltet werden.

Aufbau Übungsmodul



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

Zusammenfassung

- Die Arduino-Plattform ist eine aus Soft-und Hardware bestehende Physical-Computing-Plattform

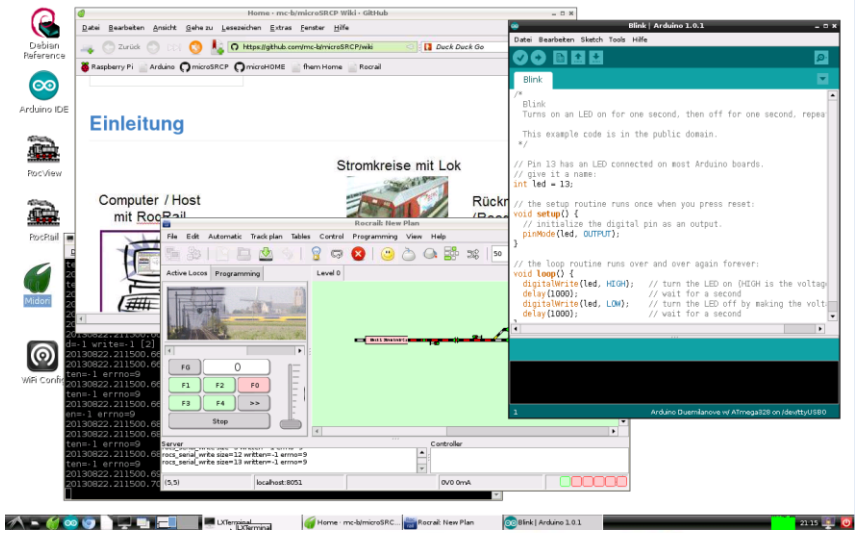
- Wikipedia (http://de.wikipedia.org/wiki/Physical_Computing):
  - Physical Computing bedeutet im weitesten Sinne, **interaktive, physische Systeme** durch die Verwendung von Hardware und Software zu erstellen. Diese Systeme **reagieren auf Ereignisse in der realen, analogen Welt und/oder wirken auf 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.

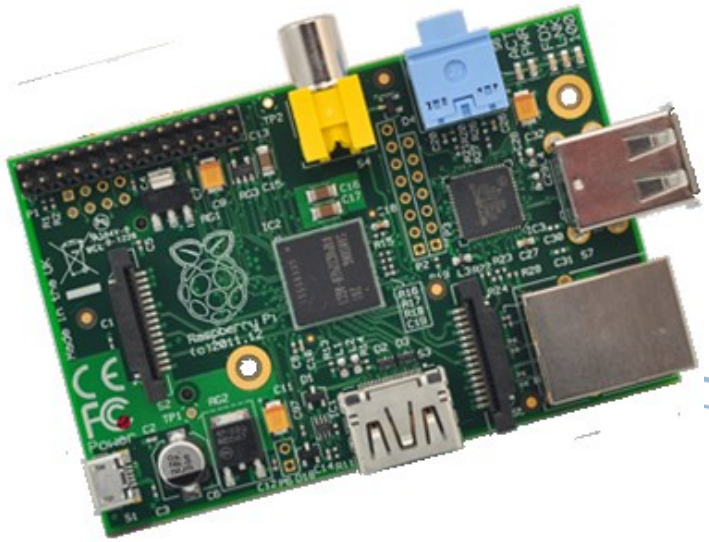
- Einfache Programmierung mittels Sketches und einen Vielzahl von Shield's machen die Arduino zu einem universell einsetzbaren I/O Gerät.

Agenda

- Einleitung „Internet der Dinge“, Sensoren, Aktoren – 5′
- Die Arduino Plattform (Hardware, einfache Anwendungen, Sensoren) – 45′
- Die Raspberry Pi Plattform – 5′
- Arduino’s und Raspberry Pi verbinden – 5′
- Aufbau einer Steuerung anhand zweier konkreter Beispiele (Hausautomation und Modelleisenbahn-Steuerung) – 60′

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



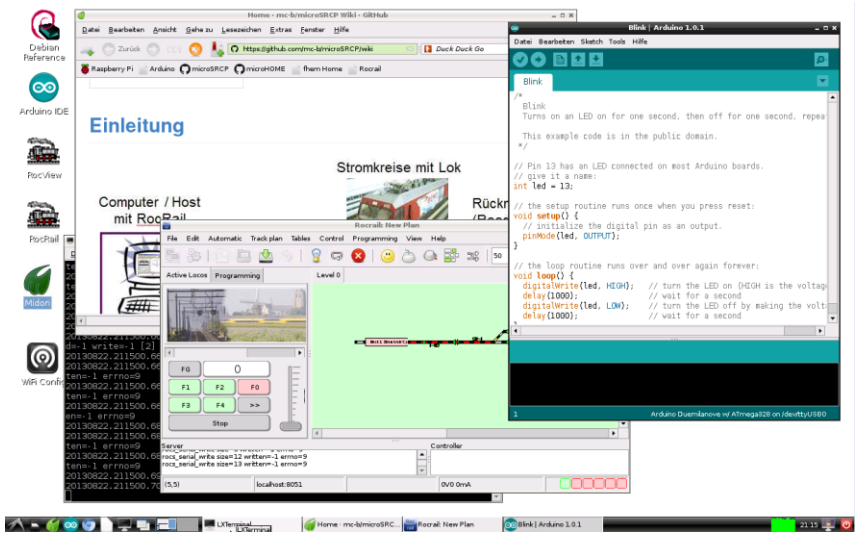


- Der Raspberry Pi ist ein kreditkarten-grosser **Einplatinencomputer**
- Die Platine enthält das Ein-Chip-System BCM 2835 von Broadcom mit dem 700-MHz-Hauptprozessor ARM1176JZF-S sowie 512 MB Arbeitsspeicher. Das Modell B hat zudem eine Ethernet-Schnittstelle und einen zweiten USB-Anschluss.
- Linux 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

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, Installiertes Betriebssystem



Installation in der Linux Shell:

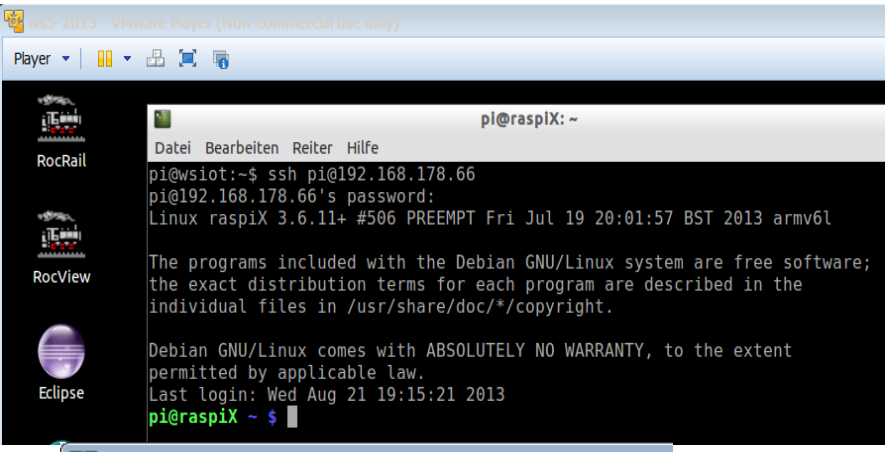
- sudo apt-get update
- sudo apt-get install samba samba-common-bin arduino \
- chromium-browser iw tightvncserver vncviewer

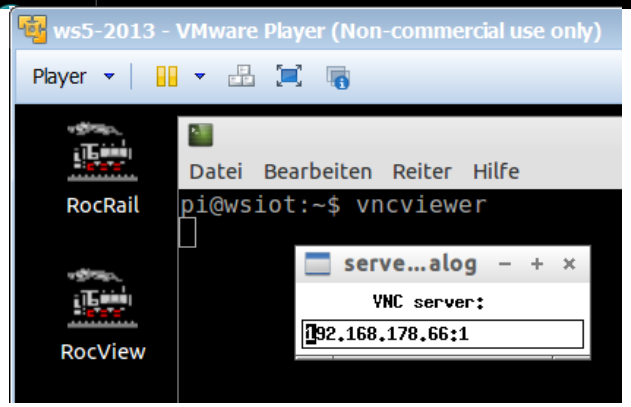
fhem und RocRail Debian bzw. Raspberry Pi Pakete downloaden und jeweils wie folgt installieren:

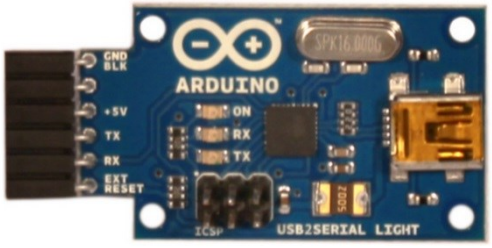
- sudo dpkg –i <paket>
- sudo apt-get –f install

- 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 Hausautomat-isierung und Energie-Messung
  - RocRail – Modelleisenbahn Steuerung

Raspberry Pi, Zugriff (ohne Bildschirm)







- 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

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

- Einleitung „Internet der Dinge“, Sensoren, Aktoren – 5′
- Die Arduino Plattform (Hardware, einfache Anwendungen, Sensoren) – 45′
- Die Raspberry Pi Plattform – 5′
- Arduino’s und Raspberry Pi verbinden – 5′
- Aufbau einer Steuerung anhand zweier konkreter Beispiele (Hausautomation und Modelleisenbahn-Steuerung) – 60′

Raspberry Pi - Arduino

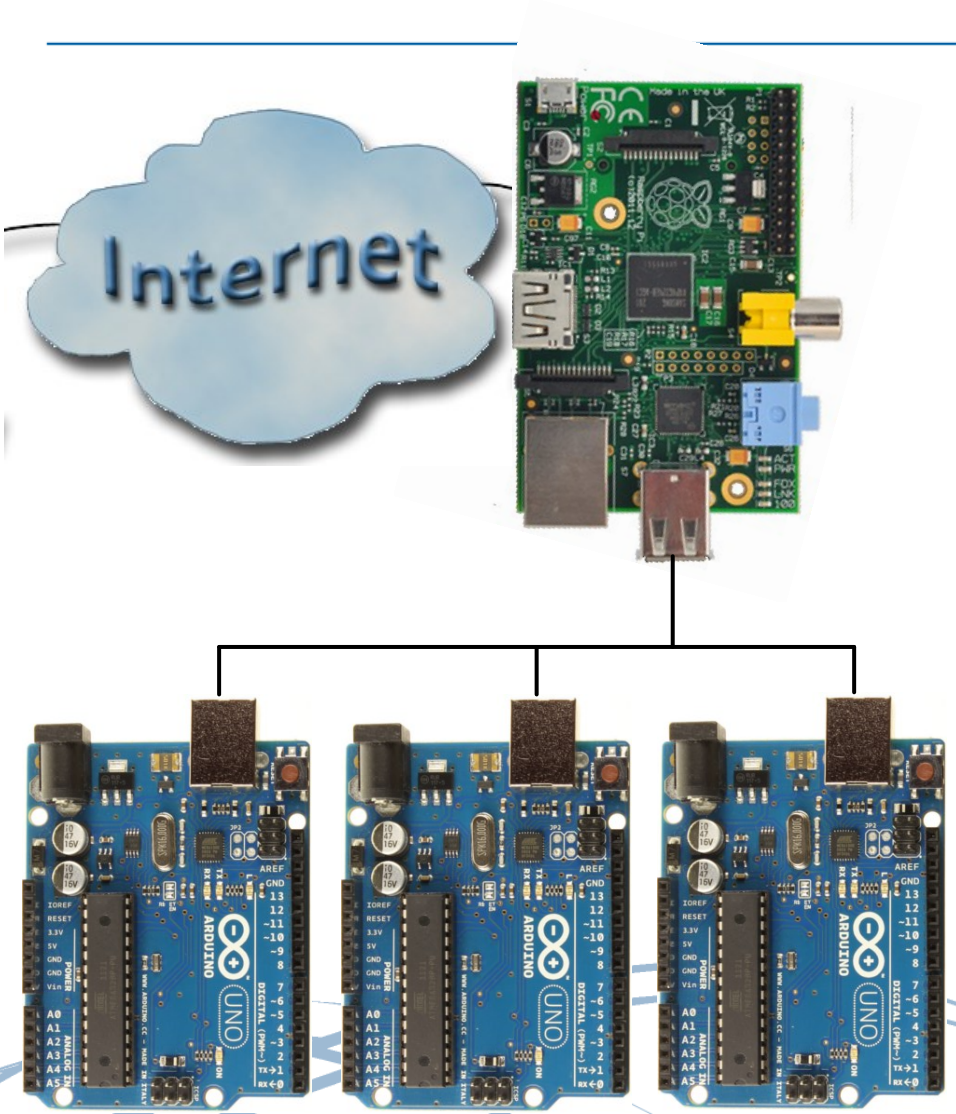
- **Das Internet der Dinge** (auch englisch Internet of Things) **beschreibt, dass der (Personal) Computer zunehmend als Gerät verschwinden und durch „intelligente Gegenstände“ ersetzt wird**.

- 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 phyische 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 Raspbery Pi
  - USB Kabel und Hub's sind billig und zuverlässig

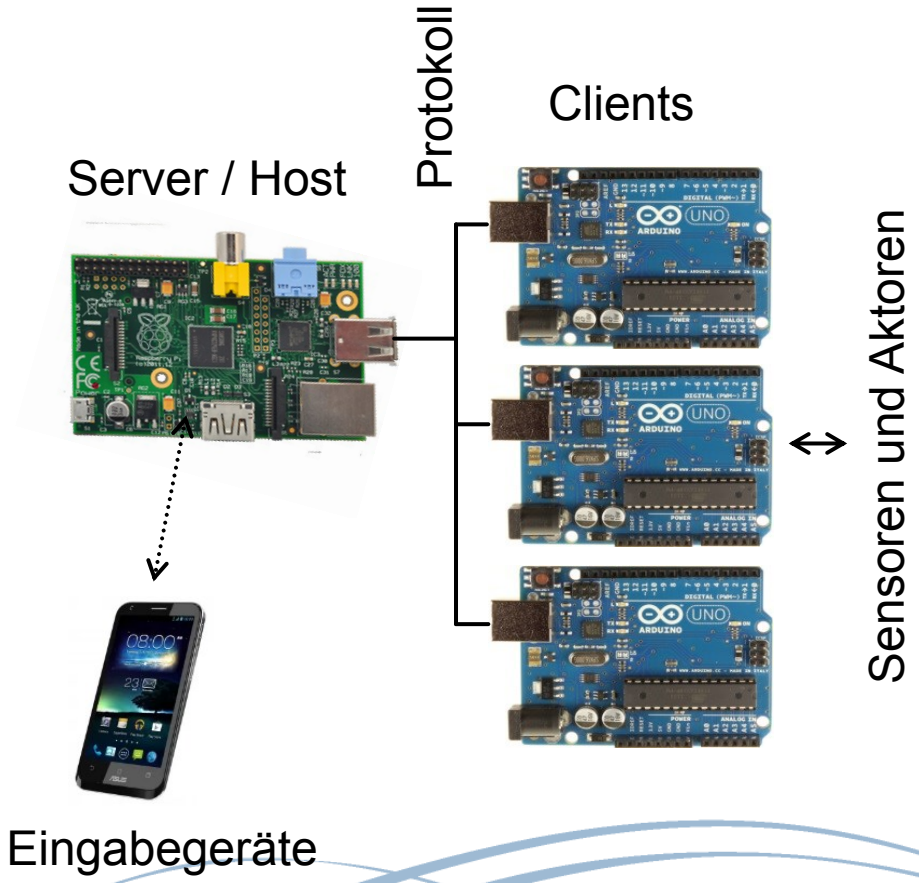
Zusammenfassung

- 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

- Einleitung „Internet der Dinge“, Sensoren, Aktoren – 5′
- Die Arduino Plattform (Hardware, einfache Anwendungen, Sensoren) – 45′
- Die Raspberry Pi Plattform – 5′
- Arduino’s und Raspberry Pi verbinden – 5′
- Aufbau einer Steuerung anhand zweier konkreter Beispiele (Hausautomation und Modelleisenbahn-Steuerung) – 60′

Grundsätzlicher Aufbau



- **Server** welche die Clients überwacht und steuert
- **Protokoll** zwischen Client und Server
- **Clients** welche die Befehle vom Server entgegen-nehmen 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

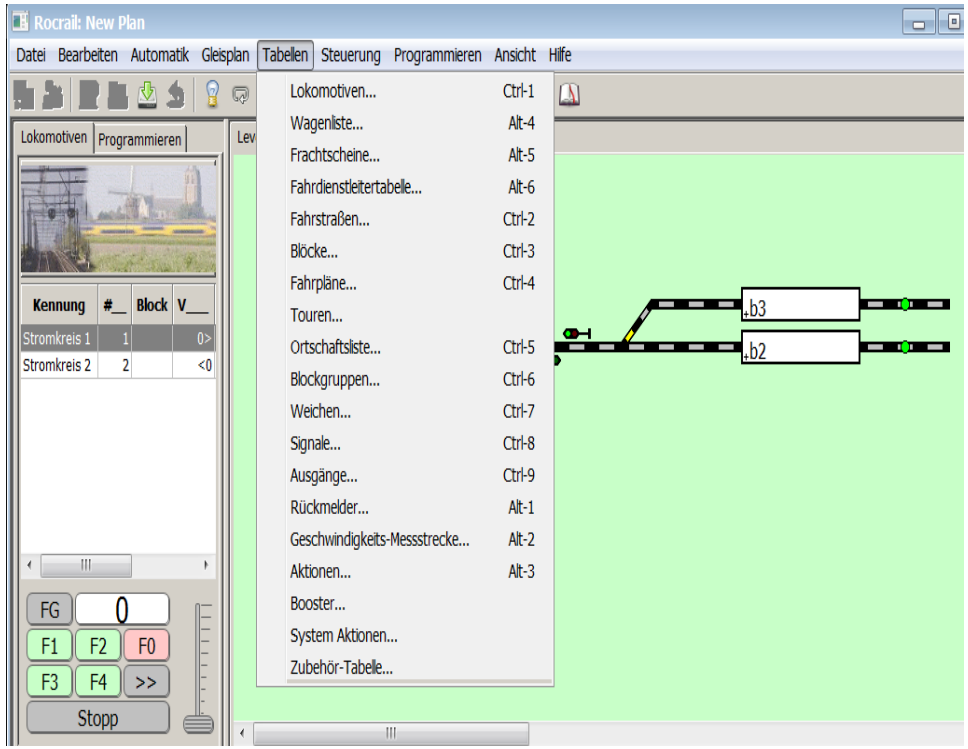
- Modelleisenbahn Komponenten
  - Server: RaspberryPi/RocRail
  - Client: Arduino/microSRCPServer
  - Protokoll: SRCP
  - Sensoren und Aktoren = Geräte
  - Eingabegeräte: RocView, andRoc, iRoc …
- https://github.com/mc-b/microSRCP/wiki
- http://de.wikipedia.org/wiki/Rocrail
- http://srcpd.sourceforge.net/

- **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

- **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 Raum-temperatur oder Luftfeuchtigkeit dient.

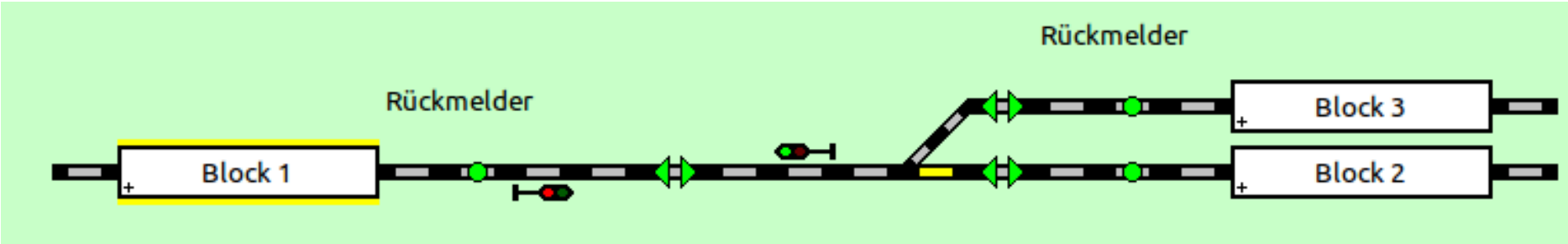
Modelleisenbahn: RocRail



- **Rocrail** ist eine freie Software zur Steuerung von digitalen Modelleisenbahnen. Die Züge können manuell, voll-automatisch 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

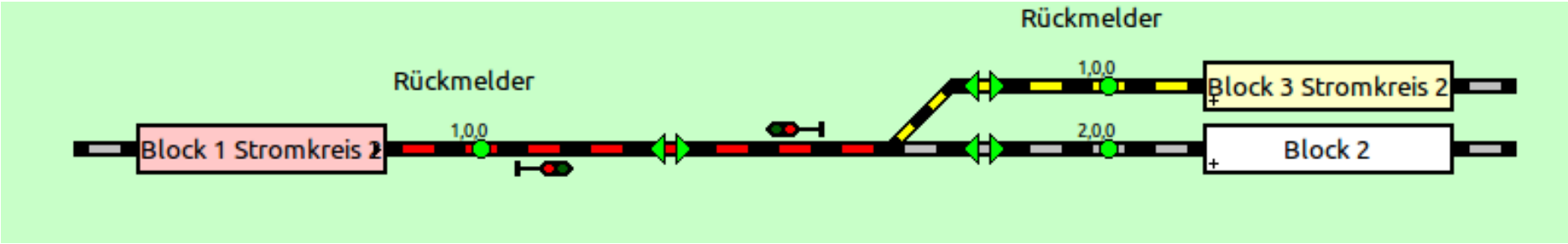
Aufgabe 1a: Modelleisenbahn



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

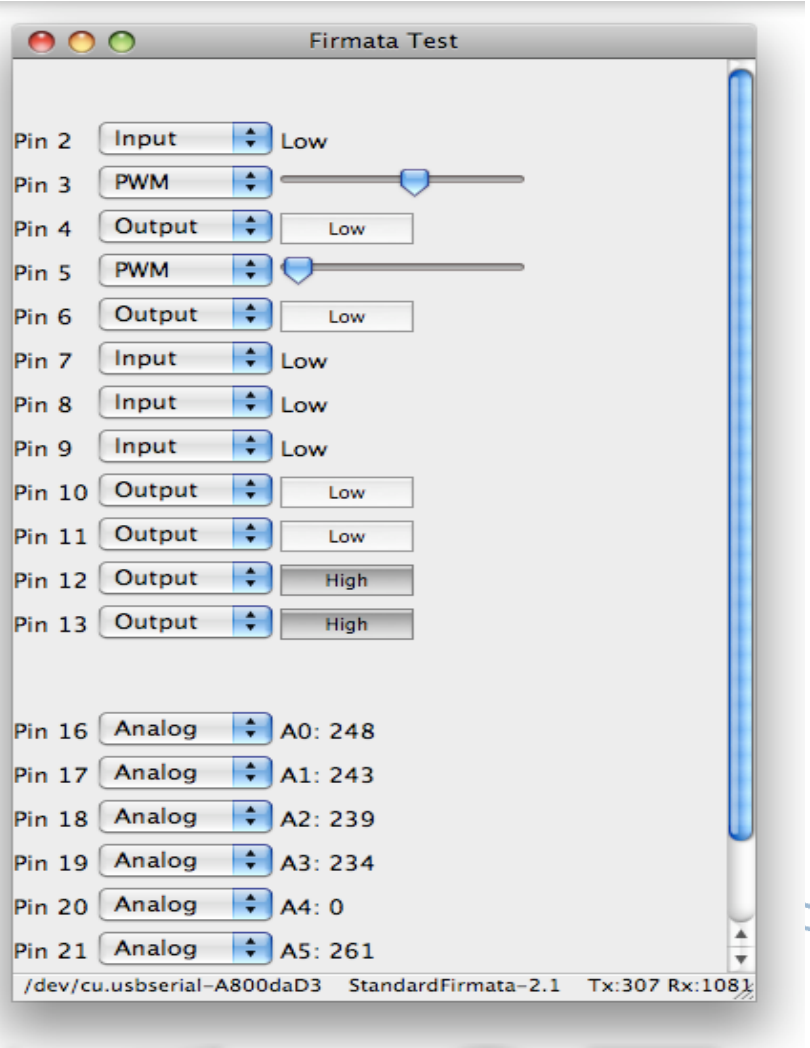


- *Blöcke und Fahrstrassen sind die Grundlage von automatischen Abläufen auf der Modelleisenbahn.*
- Erweitert den Gleisplan um Blöcke und Fahrstrassen
  - Erstellt zuerst die 3 Blöcke mit den Bezeichnungen Block 1 – 3 und die Richtungszeiger.
  - Setzt bei Block/Fahrstrassen jeweils den Rückmelder des Blocks als enter2in Aktion
  - Lasst anschliessend automatisch die Fahrstrassen erstellen mittels dem Pulldownmenu 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

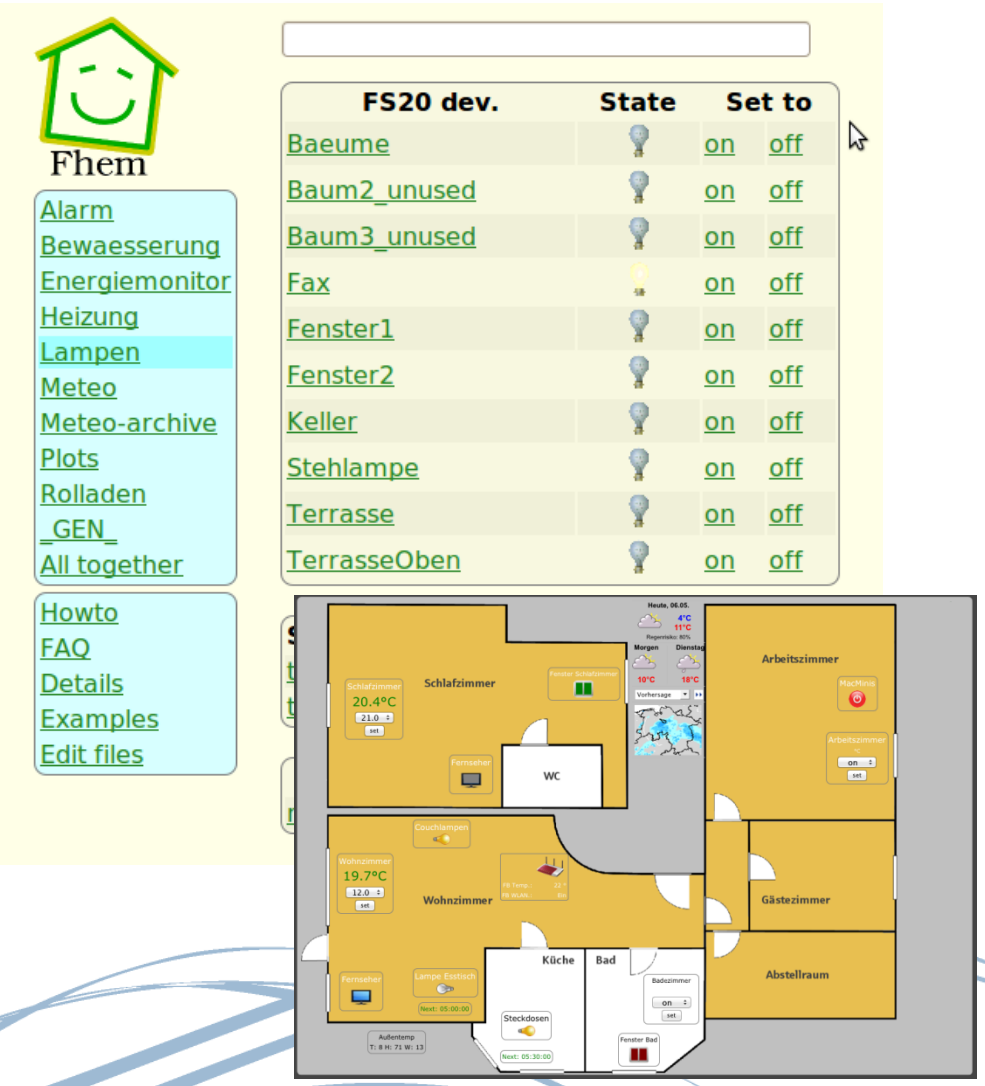
- Erweitert die Steuerung durch weitere Eingabegeräte wie ein Smartphone.
  - Installiert andRoc o.ä. auf Eurem Smartphone
  - Startet die App mit folgenden Verbindungseinstellungen:
    - Host: <IP-Adresse des Raspberry Pi>
    - Port: 8051

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.

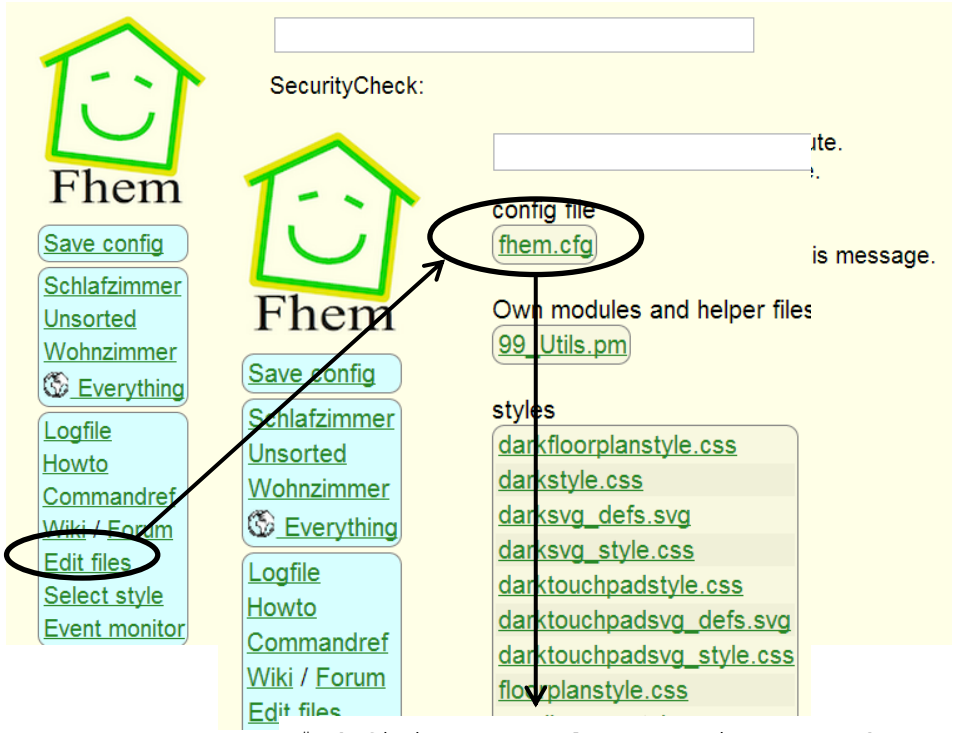
Hausautomation: fhem



- **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://www.fhemwiki.de/wiki/Arduino

Aufgabe 2a: Hausautomation




```
# definiere FRM als IO-Device - Baudrate 57600
# ist default im StandardFirmata Sketch
define Arduino1 FRM /dev/ttyUSB0@57600
attr Arduino1 loglevel 6
attr Arduino1 sampling-interval 1000

# Led 4 an Device Arduino1 im Wohnzimmer
define Led4 FRM_OUT 4
attr Led4 IODev Arduino1
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

Aufgabe 2c: Hausautomation

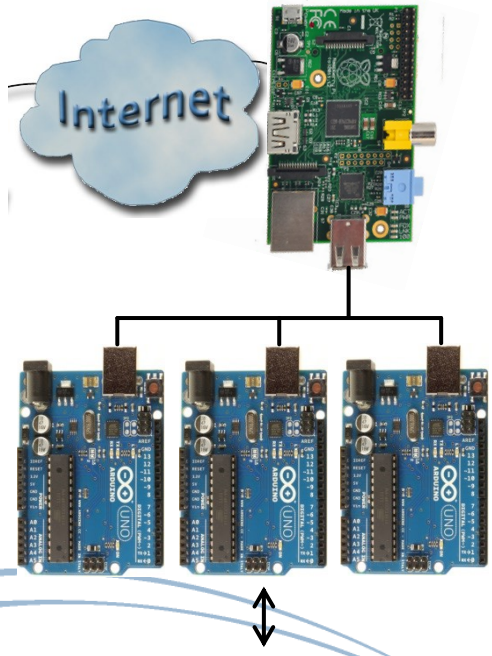
- Erweitert die Steuerung durch weitere Eingabegeräte wie ein Smartphone.
  - Installiert andFhem o.ä. auf Eurem Smartphone
  - Startet die App mit folgenden Verbindungseinstellungen:
    - URL: <IP-Adresse des Raspberry Pi>:8083/fhem

Hausautomation– mehr Kilobytes

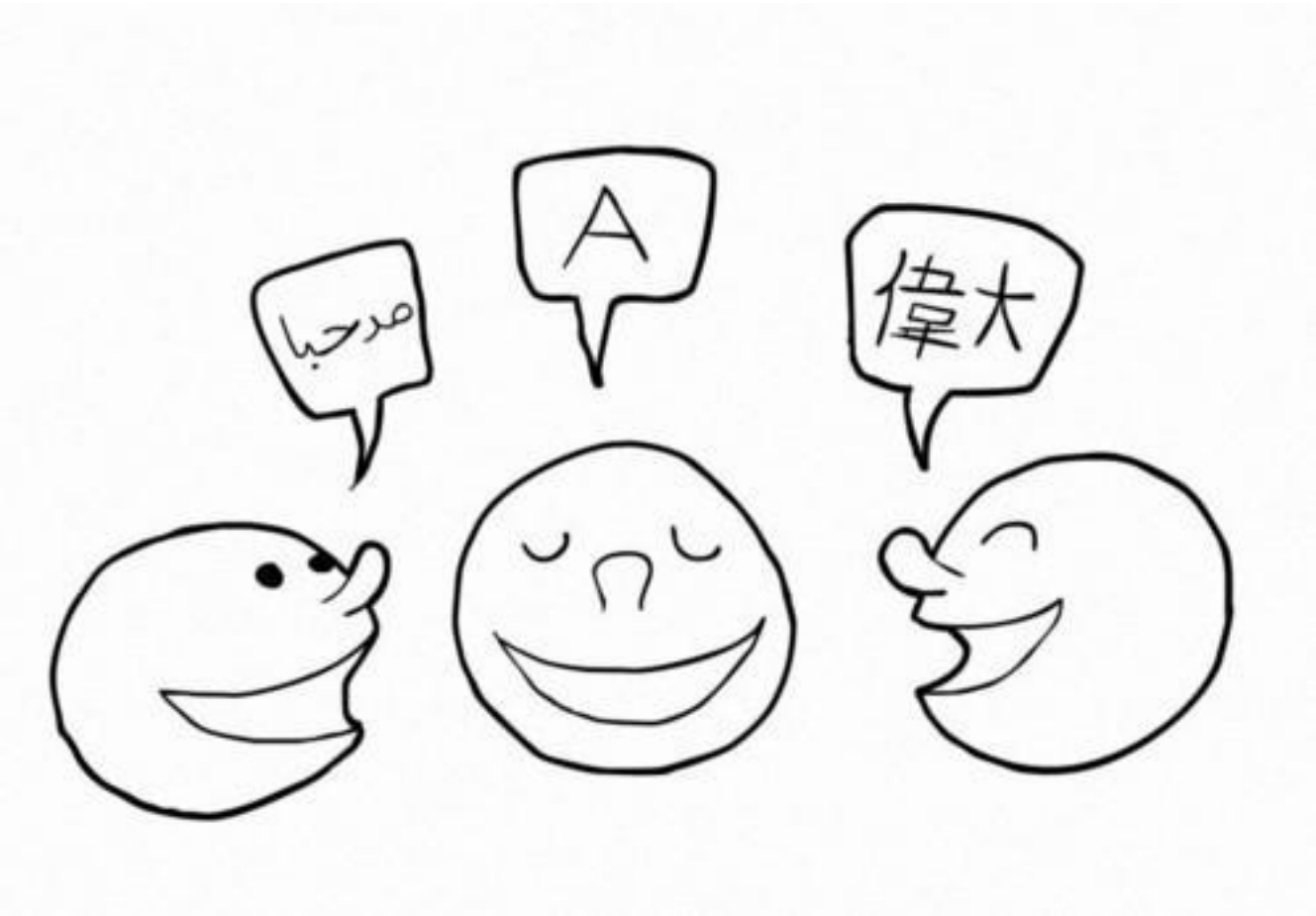
- https://github.com/mc-b/microHOME/wiki

Zusammenfassung

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



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 - 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 - Wiki
  - http://www.raspberrypi.org/phpBB3/ - Forum