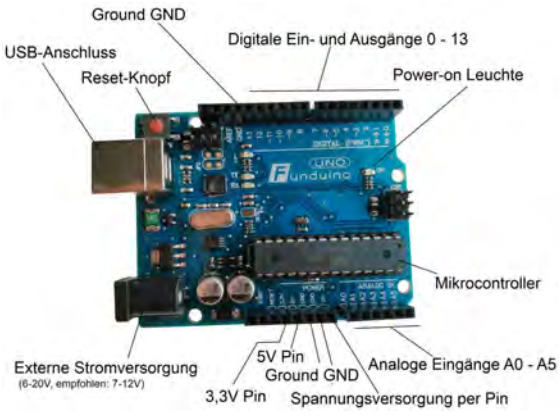
Erste Schritte

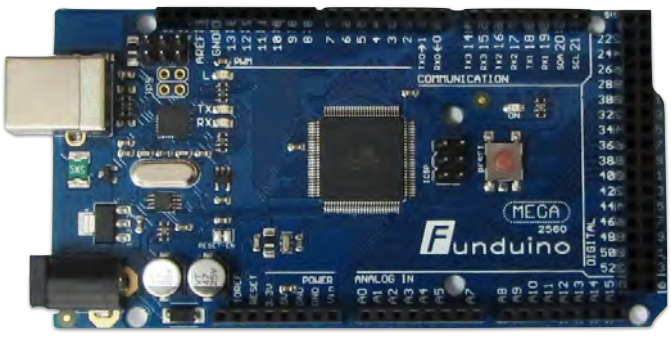
Programmieren für Ingenieure
Sommer 2015

Andreas Zeller, Universität des Saarlandes

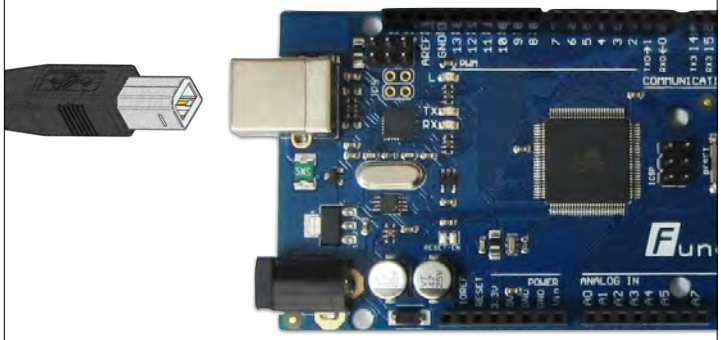
Das Arduino-Board



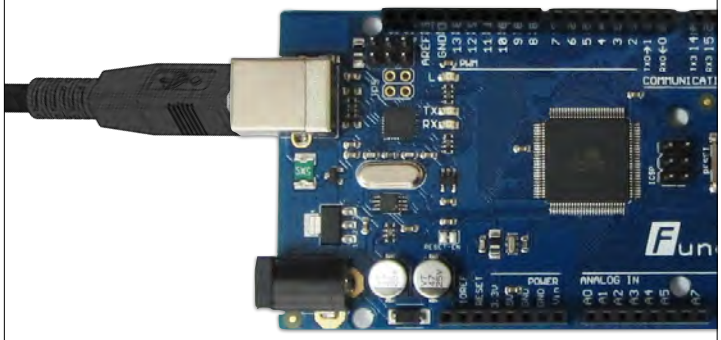
Ihr Fundino-Board



Anschluss über USB



Anschluss über USB



Programmierumgebung

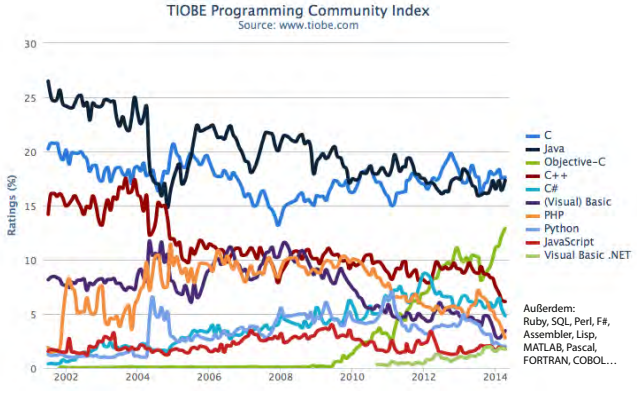
– Download über Vorlesungs-Webseite –

Demo

Ein Programm

- Bestimmt, was der Rechner tun soll
- Geschrieben in einer *Programmiersprache*
- Enthält und organisiert *Anweisungen*

Programmiersprachen



C

- Unsere Programmiersprache
- Entwickelt 1969–1973 in den Bell Labs für UNIX (als Nachfolger von B)
- Eine der verbreitetsten und einflussreichsten Sprachen
- Dialekte: C++, Objective-C

Ken Thompson und Dennis Ritchie, Erfinder der Programmiersprache C

Ein C-Programm

- besteht aus *Anweisungen:*

```
digitalWrite(led, HIGH);
```

- die wiederum in *Funktionen* zusammengefasst werden:

```
void setup() {
   pinMode(led, OUTPUT);
}
```

- *Kommentare* erläutern den Zweck:

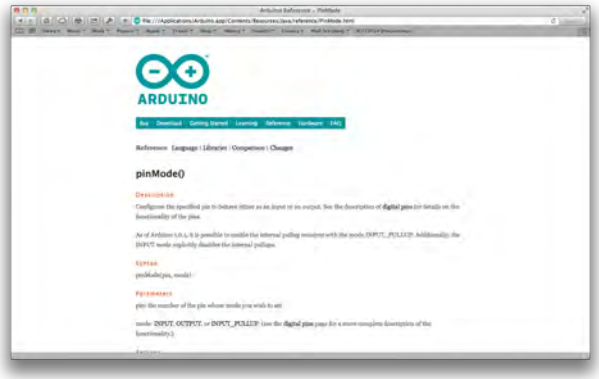
```
delay(1000);   // Eine Sekunde warten
```

Anweisungen

- Wir betrachten zunächst *Funktionsaufrufe.*
- Die Arduino-Plattform stellt Tausende von *Funktionen* zur Verfügung
- Jede Funktion bietet einen *Dienst* an.

`pinMode()`	Pin als Ein/Ausgang konfigurieren
`digitalWrite()`	Daten digital ausgeben
`delay()`	Warten

Alle Funktionen



In Arduino Menü: Hilfe → Referenz

Funktionsaufrufe

- Die meisten Funktionen haben *Parameter*, die ihre Funktionsweise bestimmen

```
digitalWrite(pin_number, value)
```

- Beim Aufruf muss für jeden Parameter ein Wert (*Argument)* angegeben werden

```
digitalWrite(13, HIGH);
```

Funktionsname

Wert für *value*

Wert für *pin_number*

Vorgegebene Funktionen

- Jedes Arduino-Programm *(Sketch)* beginnt mit zwei Funktionen:

`setup()`	Einmalig beim Start ausführen
`loop()`	Immer wiederholen

- In diesen Funktionen wird festgelegt, was im Programm passieren soll.

Funktionen definieren

- Eine Funktion wie setup() und loop() wird als Folge von Anweisungen definiert, eingeschlossen in {…}

```
void setup() {
   Anweisung 1;
   Anweisung 2;
   …
}
```

- Jede Anweisung endet in einem ";"

Kommentare

- *Kommentare* dienen dazu, das Programm für Menschen verständlich(er) zu machen
- Entweder // … bis Zeilenende oder /* … */

```
/* Pin 13 has an LED connected
on most Arduino boards. */

// setup() runs once when you press reset
```

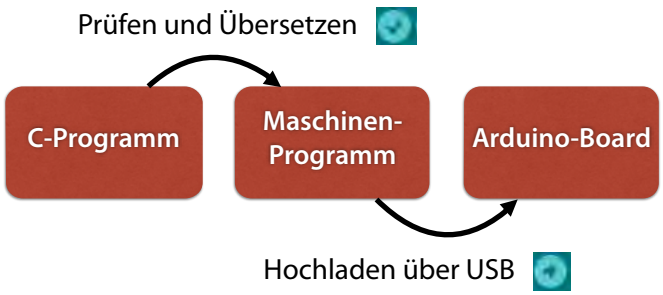
- Der Rechner *ignoriert* alle Kommentare

Beispiel: 3x Blinken

```
void setup() {
    // configure PIN 13 (built-in LED) as output
    pinMode(13, OUTPUT);

    // turn the LED on (HIGH is the voltage level)
    digitalWrite(13, HIGH);

    // wait for a second
    delay(1000);

    // turn the LED off by making the voltage LOW
    digitalWrite(13, LOW);

    // wait for a second
    delay(1000);

    // turn the LED on
    …
}
```

Demo

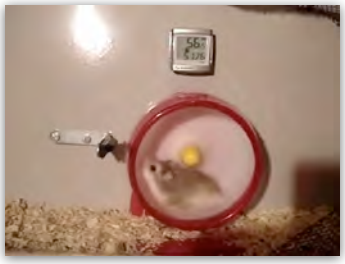
Vom Programm zum Prozessor



Demo

Wiederholung

- Nach setup() wird die loop()-Funktion immer und immer wieder aufgerufen

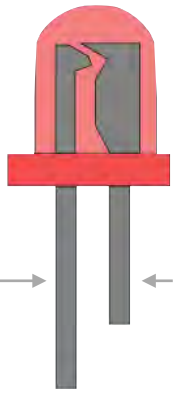


Beispiel: Ewig Blinken

```
void setup() {
    // configure PIN 13 (built-in LED) as output
    pinMode(13, OUTPUT);
}

void loop() {
    // turn the LED on (HIGH is the voltage level)
    digitalWrite(13, HIGH);

    // wait for a second
    delay(1000);

    // turn the LED off by making the voltage LOW
    digitalWrite(13, LOW);

    // wait for a second
    delay(1000);
}
```

Demo

Eine Leuchtdiode



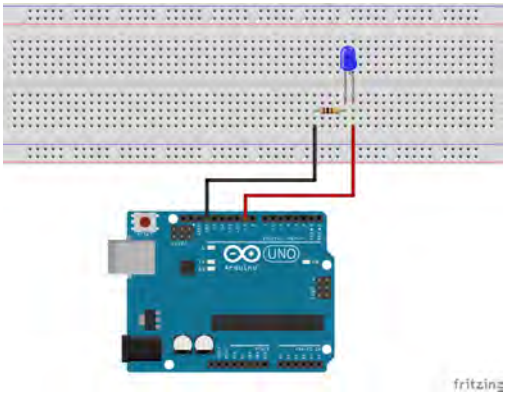
Anode (+)
- langes Bein
- runde Seite

Kathode (–)
- kurzes Bein
- abgeflachte Seite

LED anschließen

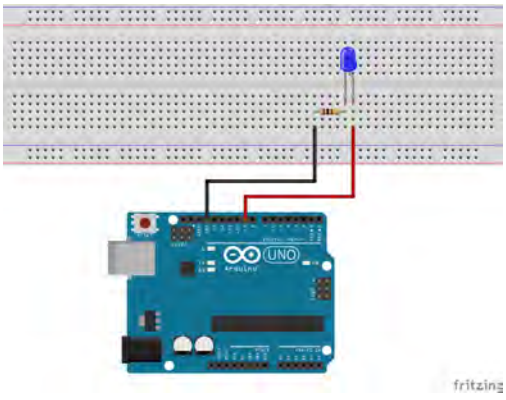
- Um eine LED an 5V anzuschließen, braucht man einen *Vorwiderstand:*
  - *200Ω* für rot, gelb
  - *100Ω* für weiß, grün, blau, IR
- Kathode (–, kurzes Bein) an GND, Anode (+, langes Bein) an Port

LED anschließen



Demo

Der richtige Port



Der richtige Port

- Soll die LED an einen anderen Port angeschlossen werden (etwa 9), muss man im gesamten Programm die Portnummer ändern
- In einem großen Programm wird das schnell zum Problem
- Lösung: *Variablen*

Variablen

- *Variablen* dienen dazu, *Werte* zu speichern.
- Mit der Anweisung

```
int led = 13;
```

  wird `led` als eine *Variable* eingeführt, die mit dem Wert 13 belegt ist.
- Nach der Anweisung steht der Name `led` stellvertretend für den Variablenwert

Typen

- Der *Typ* einer Variable bestimmt, welche Werte die Variable speichern kann
- `int` – ganzzahlige Werte (integer)
- Weitere Typen: `float`, `char`, `void`

Symbolisches Blinken

```
// Pin 13 has an LED connected on most
// Arduino boards.  Give it a name:
int led = 13;

void setup() {
   pinMode(led, OUTPUT);
}

void loop() {
    digitalWrite(led, HIGH);
    delay(1000);
    digitalWrite(led, LOW);
    delay(1000);
}
```

Schneller Blinken

```
// Pin 13 has an LED connected on most
// Arduino boards.  Give it a name:
int led = 13;

// Blinking delay (in ms)
int blink_delay = 250;

void setup() {
   pinMode(led, OUTPUT);
}

void loop() {
    digitalWrite(led, HIGH);
    delay(blink_delay);
    digitalWrite(led, LOW);
    delay(blink_delay);
}
```

Demo

Wechselblinken

```
int led_red   = 12;
int led_green = 13;

void setup() {
   pinMode(led_red, OUTPUT);
   pinMode(led_green, OUTPUT);
}

void loop() {
    digitalWrite(led_red, HIGH);
    digitalWrite(led_green, LOW);
    …
}
```

Demo

Bezeichner

- Alle Namen für Variablen und Funktionen *(Bezeichner)* bestehen aus a–z, A-Z, 0–9 und _ (Unterstrich)
- Bezeichner dürfen nicht mit 0–9 beginnen
- In einem Sketch darf jeder Bezeichner nur 1x vergeben werden

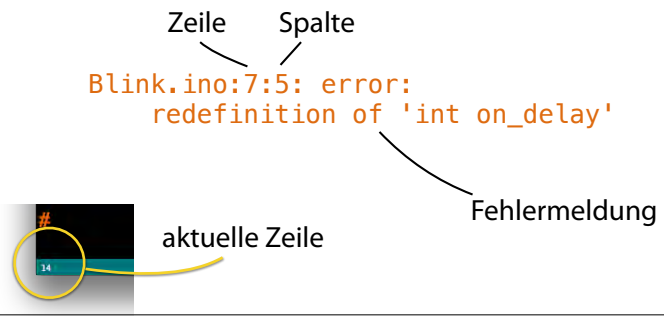
Bezeichner

- `delay`, `Delay` und `DELAY` sind unterschiedliche Bezeichner
- Konvention:
  - `Delay` – eine *Klasse*
  - `DELAY` – ein *Makro*
  - `_delay` – *intern*

} machen wir nicht!

So was dähmliches

- Bei Fehlern: *Fehlermeldung*



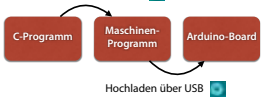
Demo

Vorschau

- Morse-Code
- Funktionen mit Parametern
- Kontrollstrukturen

Vom Programm zum Prozessor

Prüfen und Übersetzen



Hochladen über USB

Funktionsaufrufe

- Die meisten Funktionen haben *Parameter*, die ihre Funktionsweise bestimmen

```
digitalWrite(pin_number, value)
```

- Beim Aufruf muss für jeden Parameter ein Wert (*Argument*) angegeben werden

```
digitalWrite(13, HIGH);
```

Funktionsname — Wert für *value*

Wert für *pin_number*

Variablen

- *Variablen* dienen dazu, *Werte* zu speichern.
- Mit der Anweisung

```
int led = 13;
```

wird `led` als eine *Variable* eingeführt, die mit dem Wert 13 belegt ist.

- Nach der Anweisung steht `led` stellvertretend für den Variablenwert

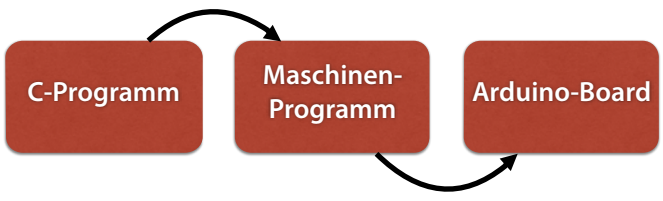
Symbolisches Blinken

```
// Pin 13 has an LED connected on most
// Arduino boards.  Give it a name:
int led = 13;

void setup() {
    pinMode(led, OUTPUT);
}

void loop() {
    digitalWrite(led, HIGH);
    delay(1000);
    digitalWrite(led, LOW);
    delay(1000);
}
```

Handouts

Vom Programm zum Prozessor

Prüfen und Übersetzen



Hochladen über USB

Ein C-Programm

- besteht aus *Anweisungen:*

```
digitalWrite(led, HIGH);
```

- die wiederum in *Funktionen* zusammengefasst werden:

```
void setup() {
   pinMode(led, OUTPUT);
}
```

- *Kommentare* erläutern den Zweck:

```
delay(1000);   // Eine Sekunde warten
```

Vorgegebene Funktionen

- Jedes Arduino-Programm *(Sketch)* beginnt mit zwei Funktionen:

`setup()`	Einmalig beim Start ausführen
`loop()`	Immer wiederholen

- In diesen Funktionen wird festgelegt, was im Programm passieren soll.

Funktionen definieren

- Eine Funktion wie setup() und loop() wird als Folge von Anweisungen definiert, eingeschlossen in {…}

```
void setup() {
   Anweisung 1;
   Anweisung 2;
   …
}
```

- Jede Anweisung endet in einem ";"

Anweisungen

- Wir betrachten zunächst *Funktionsaufrufe*.
- Die Arduino-Plattform stellt Tausende von *Funktionen* zur Verfügung
- Jede Funktion bietet einen *Dienst* an.

`pinMode()`	Pin als Ein/Ausgang konfigurieren
`digitalWrite()`	Daten digital ausgeben
`delay()`	Warten

Funktionsaufrufe

- Die meisten Funktionen haben *Parameter*, die ihre Funktionsweise bestimmen

```
digitalWrite(pin_number, value)
```

- Beim Aufruf muss für jeden Parameter ein Wert (*Argument)* angegeben werden

```
digitalWrite(13, HIGH);
```

Funktionsname

Wert für *pin_number*

Wert für *value*

Variablen

- *Variablen* dienen dazu, *Werte* zu speichern.
- Mit der Anweisung

```
int led = 13;
```

wird `led` als eine *Variable* eingeführt, die mit dem Wert 13 belegt ist.

- Nach der Anweisung steht der Name `led` stellvertretend für den Variablenwert

Symbolisches Blinken

```
// Pin 13 has an LED connected on most
// Arduino boards.  Give it a name:
int led = 13;

void setup() {
   pinMode(led, OUTPUT);
}

void loop() {
    digitalWrite(led, HIGH);
    delay(1000);
    digitalWrite(led, LOW);
    delay(1000);
}
```