Felder und Schleifen

Programmieren für Ingenieure
Sommer 2014

Andreas Zeller, Universität des Saarlandes

Sensor abfragen

```
int ledPin = 13;   // Die LED
int buttonPin = 8; // Der Taster

void setup() {
    pinMode(ledPin, OUTPUT);
    pinMode(buttonPin, INPUT);
}

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

Zuweisung

- Die Anweisung

  name = wert

  bewirkt, dass die Variable *name* den neuen Wert *wert* hat.
- Im weiteren Programmablauf liefert jeder spätere Zugriff auf die Variable den Wert *wert* (bis zur nächsten Zuweisung)

Blinken mit Millis

```
int ledPin = 13;     // Pin LED
int buttonPin = 8;   // Pin Taster
int ms = 0;          // Zeit

void setup() {
    pinMode(ledPin, OUTPUT);
    pinMode(buttonPin, INPUT);
}

void loop() {
    int ms = millis() % 1000;
    if (ms < 500)
        digitalWrite(ledPin, LOW);
    if (ms > 500)
        digitalWrite(ledPin, HIGH);
}
```

Entprellen mit Millis

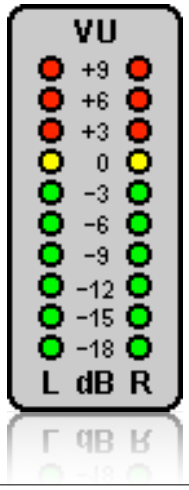
```
unsigned long previousPush = 0;

void loop() {
  if (millis() - previousPush >= 20) {
    if (digitalRead(buttonPin) == HIGH) {
      previousPush = millis();
      ledStatus = !ledStatus;
      pushed = 1;
    }
    else if (pushed && digitalRead(buttonPin) == LOW)
      pushed = 0;
  }

  // Blinken...
}
```

Themen heute

- Felder
- Schleifen
- Heartbleed

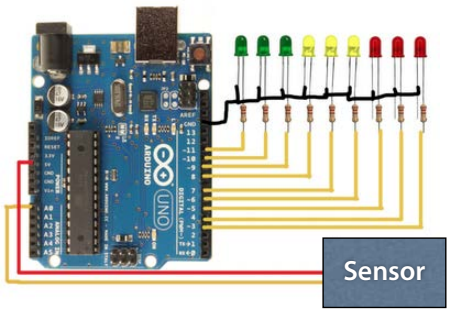
Pegel

LED-Kette soll Pegel anzeigen



Plan

- Wir schließen einen Sensor an – etwa einen *Photowiderstand*
- Wir messen dessen Wert
- Wir schließen 7 LEDs an
- Je nach Pegel lassen wir 0–7 LEDs leuchten



http://learningthearduino.blogspot.de/2012/11/arduino-vu-meter-audio-sound-meter-with.html

Sensor abfragen

- Wir haben bereits digitalRead() kennengelernt, das Daten digital liest
- Neu: analogRead() liest Daten *analog* ein

```
analogRead(pin_number)
```

liefert einen Wert von 0…1023,
je nach Spannung am Pin (0…5V)

Sensor abfragen

```
int inputPin = 0;   // Der Eingabe-PIN

void setup() {
    Serial.begin(9600);
}

void loop() {
    int level = analogRead(inputPin);

    Serial.println(level);
    delay(100);
}
```

Demo

- Photowiderstand anschließen

Sensorwert ausgeben

- Je nach Wert am Analog-Eingang sollen 0…7 LEDs leuchten

Sensorwert ausgeben

```
int inputPin = 0;   // Der Eingabe-PIN

int led1 = 13;   // Die LED-PINs
int led2 = 12;
int led3 = 11;
// usw. für 4 weitere LEDs

void setup() {
    pinMode(led1, OUTPUT);
    pinMode(led2, OUTPUT);
    // usw.
}
```

```
int inputPin = 0;   // Der Eingabe-PIN

int led1 = 13;   // Die LED-PINs
int led2 = 12;
int led3 = 11;
// usw. für 4 weitere LEDs

void setup() {
    pinMode(led1, OUTPUT);
    pinMode(led2, OUTPUT);
    // usw.
}

void loop() {
    int level = analogRead(inputPin);

    if (level < 100)
        digitalWrite(led1, LOW);
    else
        digitalWrite(led1, HIGH);
```

```
void loop() {
    int level = analogRead(inputPin);

    if (level < 100)
        digitalWrite(led1, LOW);
    else
        digitalWrite(led1, HIGH);

    if (level < 200)
        digitalWrite(led2, LOW);
    else
        digitalWrite(led2, HIGH);

    if (level < 300)
        digitalWrite(led3, LOW);
    else
        digitalWrite(led3, HIGH);

    // usw. für 4 weitere LEDs
}
```

```
int inputPin = 0;   // Der Eingabe-PIN

int led1 = 13;   // Die LED-PINs
int led2 = 12;
int led3 = 11;
// usw. für 4 weitere LEDs

void setup() {
    pinMode(led1, OUTPUT);
    pinMode(led2, OUTPUT);
    // usw.
}

void loop() {
    int level = analogRead(inputPin);

    if (level < 100)
        digitalWrite(led1, LOW);
    else
        digitalWrite(led1, HIGH);

    if (level < 200)
        digitalWrite(led2, LOW);
    else
        digitalWrite(led2, HIGH);

    if (level < 300)
        digitalWrite(led3, LOW);
    else
        digitalWrite(led3, HIGH);

    // usw. für 4 weitere LEDs
}
```

Alle Variablen werden gleich behandelt

Ziel

Wir brauchen einen Weg, *unterschiedliche* Variablen auf *dieselbe Weise* zu behandeln

Felder

- Ein *Feld (Array)* fasst mehrere Variablen zu einer Einheit zusammen
- Jedes *Element* (jede Variable) wird mit einer Nummer angesprochen

Felder

0	1	2	3	4	5	6	7	8	9	10

- Die Notation *name*[*index*] bezieht sich auf das Element mit der Nummer *index*
- Das erste Element hat die Nummer 0

Felder deklarieren

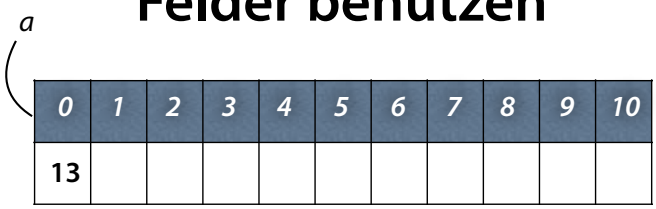
a

0	1	2	3	4	5	6	7	8	9	10

```
int a[11];    // 11 Elemente
```

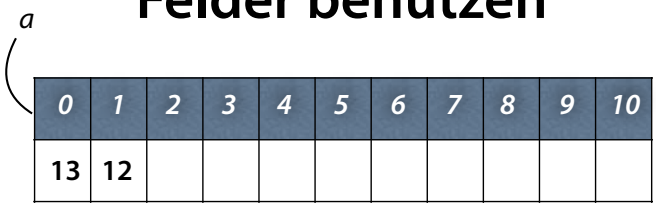
Typ der Elemente — Name des Feldes — Größe des Feldes

Felder benutzen



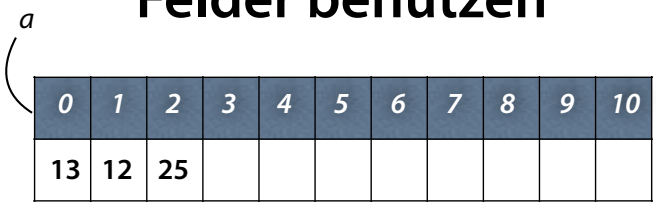

```
int a[11];    // 11 Elemente

void setup() {
   a[0] = 13;


}
```

Felder benutzen




```
int a[11];    // 11 Elemente

void setup() {
   a[0] = 13;
   a[1] = 12;


}
```

Felder benutzen




```
int a[11];    // 11 Elemente

void setup() {
   a[0] = 13;
   a[1] = 12;
   a[2] = a[0] + a[1];

}
```

Felder benutzen

a

0	1	2	3	4	5	6	7	8	9	10
					1					

```
int a[11];    // 11 Elemente

void setup() {
   int i = 5;
   a[i] = 1;


}
```

Geht auch mit symbolischem Index

Felder benutzen

a

0	1	2	3	4	5	6	7	8	9	10
	2				1					

```
int a[11];    // 11 Elemente

void setup() {
   int i = 5;
   a[i] = 1;
   a[a[i]] = 2;

}
```

Geht auch mit symbolischem Index

Felder initialisieren

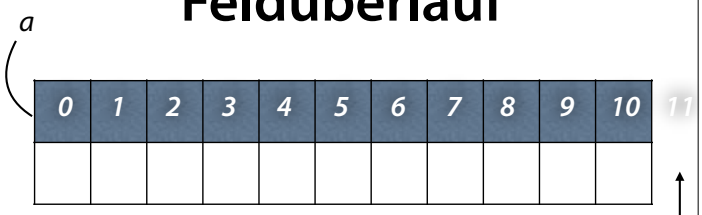
leds

0	1	2	3	4	5	6
13	12	11	10	9	8	7

```
int leds[7];
void setup() {
   leds[0] = 13;
   leds[1] = 12;
   leds[2] = 11;
   leds[3] = 10;
   leds[4] = 9;
   leds[5] = 8;
   leds[6] = 7;
}
```

```
int leds[] =
    { 13, 12, 11, 10, 9, 8, 7 };

void setup() {
   // Bereits initialisiert
}
```

Feldüberlauf




```
int a[11];    // 11 Elemente

void setup() {
   int x = a[11];
}
```

Was passiert hier?

Undefiniertes Verhalten

- Greift das Programm außerhalb der Grenzen auf ein Feld zu, ist sein Verhalten undefiniert (= alles ist möglich!)
- Später mehr dazu

http://openclipart.org/image/2400px/svg_to_png/189185/elefante_in_corsa_pink.png

Bei jedem Feldzugriff *Feldgrenzen* beachten!

Schleifen

- Wir brauchen immer noch einen Weg, um alle Elemente einheitlich zu behandeln

```
void setup() {
    pinMode(leds[0], OUTPUT);
    pinMode(leds[1], OUTPUT);
    pinMode(leds[2], OUTPUT);
    pinMode(leds[3], OUTPUT);
    pinMode(leds[4], OUTPUT);
    pinMode(leds[5], OUTPUT);
    pinMode(leds[6], OUTPUT);
}
```

wie sieht das aus bei 100 LEDs?

While-Schleifen

- Mit einer *Schleife* kann man ein Programmstück beliebig oft *wiederholen:*

```
while (Bedingung) {
   Anweisungen…;
}
```

- Die *Anweisungen* werden wiederholt, *solange* die *Bedingung* erfüllt ist

While-Schleifen

```
i = 1;
while (i < 5) {
   Serial.println(i);
   i = i + 1;
}
Serial.println("ENDE");
```

Ausgeführte Anweisungen

```
i = 1;
Serial.println(i);
i = i + 1; // 2
Serial.println(i);
i = i + 1; // 3
Serial.println(i);
i = i + 1; // 4
Serial.println(i);
i = i + 1; // 5
Serial.println("ENDE");
```

Setup mit Schleife

- Der setup()-Code kann jetzt so aussehen:

```
void setup() {
    int i = 0;
    while (i < 7) {
        pinMode(leds[i], OUTPUT);
        i = i + 1;
    }
}
```

Schleife mit Schleife

```
void loop() {
    int level = analogRead(inputPin);

    if (level < 100)
        digitalWrite(led1, LOW);
    else
        digitalWrite(led1, HIGH);

    if (level < 200)
        digitalWrite(led2, LOW);
    else
        digitalWrite(led2, HIGH);

    if (level < 300)
        digitalWrite(led3, LOW);
    else
        digitalWrite(led3, HIGH);

    // usw. für 4 weitere LEDs
}
```

```
void loop() {
    int level = analogRead(inputPin);

    int i = 0;
    while (i < 7)
    {
        if (level < 100 * (i + 1))
            digitalWrite(leds[i], LOW);
        else
            digitalWrite(leds[i], HIGH);
        i = i + 1;
    }
}
```

Demo

- LEDs und (erst einmal) Potentiometer anschließen

Endlosschleife

- setup() und loop() werden vom System als *Endlosschleife* aufgerufen:

```
void main() {
  setup();
  while (1) {
    loop();
  }
}
```

Feinheiten

- Anstelle von `x = x + y` geht auch `x += y` (analog `x *= y, x /= y, x -= y`)

```
void setup() {
    int i = 0;
    while (i < 7) {
        pinMode(leds[i], OUTPUT);
        i += 1;
    }
}
```

Inkrement

- Anstelle von `x = x + 1` oder `x += 1` geht auch `++x` oder `x++` (analog `--x` oder `x--` für `x -= 1`)

```
void setup() {
    int i = 0;
    while (i < 7) {
        pinMode(leds[i], OUTPUT);
        i++;
    }
}
```

Prä- und Postinkrement

- *++x* erhöht *erst x* und liefert *dann* den Wert
- *x++* liefert *erst* den Wert von *x* und erhöht *dann*

```
int x = 0;
int y = 0;

y = x++;
// x = 1, y = 0

y = ++x;
// x = 2, y = 2

y = x--;
// x = 1, y = 2
```

Inline Inkrement

```
void setup() {
    int i = 0;
    while (i < 7) {
        pinMode(leds[i], OUTPUT);
        i++;
    }
}
```

```
void setup() {
    int i = 0;
    while (i < 7)
        pinMode(leds[i++], OUTPUT);
}
```

For-Schleifen

- Weiß man, wie oft eine Schleife wiederholt werden soll, setzt man oft *for-Schleifen* ein:

```
for (Initialisierung; Bedingung; Inkrement) {
   Anweisungen…;
}
```

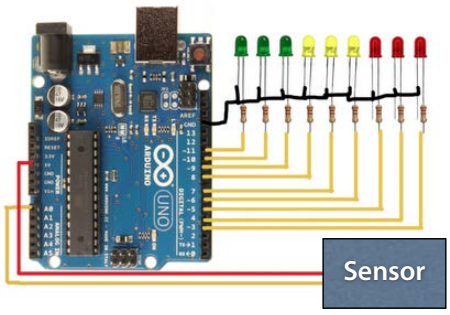
ist dasselbe wie

```
Initialisierung;
while (Bedingung) {
   Anweisungen…;
   Inkrement;
}
```

Setup mit For-Schleife

```
void setup() {
    int i = 0;
    while (i < 7) {
        pinMode(leds[i], OUTPUT);
        i++;
    }
}
```

```
void setup() {
    for (int i = 0; i < 7; i++)
       pinMode(leds[i], OUTPUT);
}
```



http://learningthearduino.blogspot.de/2012/11/arduino-vu-meter-audio-sound-meter-with.html

Pegel anzeigen

```
void range(int d) {
  for (int i = 0; i < 7; i++)
    if (i < d)
      digitalWrite(leds[i], HIGH);
    else
      digitalWrite(leds[i], LOW);
}


void loop() {
    int level = analogRead(inputPin);
    range(level / 100);
}
```

Demo

- Erneut Belichtung messen

Punkt anzeigen

```
void dot(int d) {
  for (int i = 0; i < 7; i++)
    if (i == d)
      digitalWrite(leds[i], HIGH);
    else
      digitalWrite(leds[i], LOW);
}


void dot() {
    int level = analogRead(inputPin);
    dot(level / 100);
}
```

Demo

- Belichtung per Punkt messen

Lauflicht

```
void dot(int d) {
  for (int i = 0; i < 7; i++)
    if (i == d)
      digitalWrite(leds[i], HIGH);
    else
      digitalWrite(leds[i], LOW);
}

void loop() {
    for (int i = 0; i < 7; i++) {
        dot(i);
        delay(20);
    }
}
```

Der Titel dieser Folie enthält keine Ligatur. Wem so etwas auffällt: bei mir melden :-)

Demo

- Demo Lauflicht (ggf. variieren, auch über Sensor)

Sichtbarkeit

- Jede Variable ist nur innerhalb der sie einschließenden Klammern {…} (einem *Block)* definiert

```
void loop() {
  for (int i = 0; i < 7; i++) {
    int n = i + 1;
    dot(n);
  }
  // n und i sind hier nicht mehr sichtbar
}
```

Sichtbarkeit

- In einem Block muss jede Variable einen eindeutigen Namen tragen
- Verschiedene Blöcke dürfen den gleichen Namen für "ihre" Variablen benutzen

```
void dot(int d) {
  for (int i = 0; i < 7; i++) {
    ...
  }
}

void loop() {
  for (int i = 0; i < 7; i++) {
    dot(i);
  }
}
```

Zwei verschiedene i's

Sichtbarkeit

- Beim Zugriff auf eine Variable gilt der *innerste* Name.

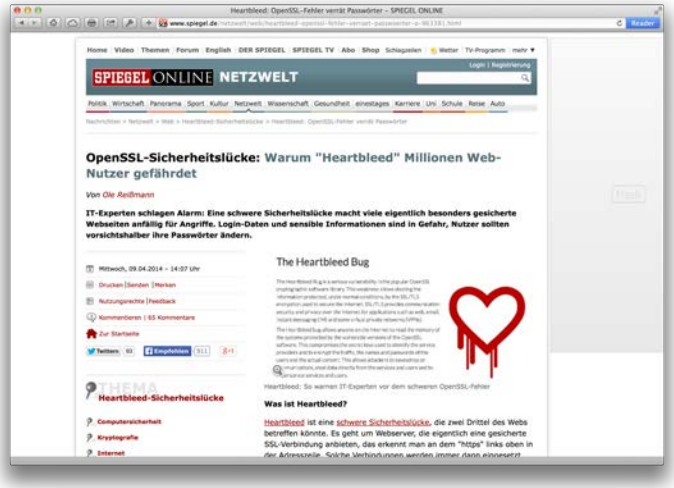
```
int i = 1;
void loop() {
  Serial.println(i);
  for (int i = 2; i < 10; i++) {
    if (i < 10)
    {
      int i = 3;
      Serial.println(i);
    }
  }
}
```

Globale Variablen

- Variablen, die nicht in einem Block definiert werden, heißen *globale Variablen*
- Globale Variablen sind *risikoreich*, da man sie unabsichtlich benutzen kann

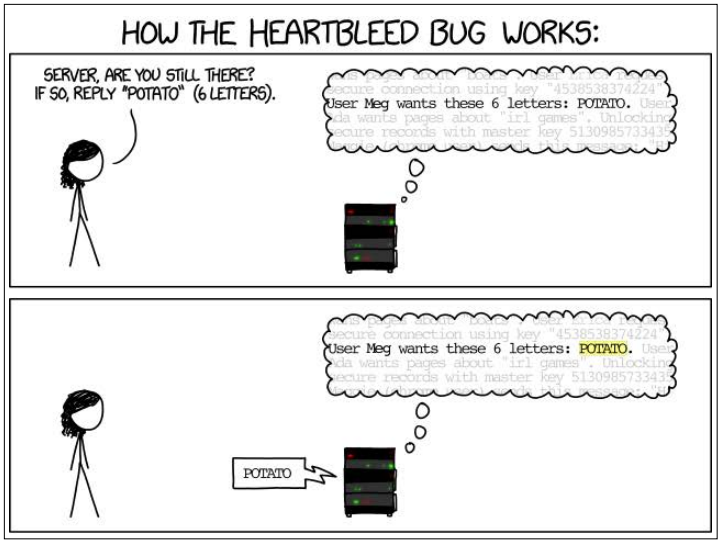
```
float pi = 3.141526535;

void loop() {
  int py = 25;
  pi += 1;  // sollte py sein
}
```

Heartbleed



Heartbeat-Protokoll

- Die *Heartbeat*-Funktion dient zu prüfen, ob ein Server im Internet noch läuft
- Man schickt eine Zeichenkette hin –
- und bekommt sie in gleicher Form zurück.



Zeichen in C

- Ein einzelnes Zeichen wird in C in einzelne Hochkomma eingeschlossen:

```
char c = 'a';
Serial.println(c);
```

- Die wichtigste Verwendung ist als *Feld* von Zeichen (*Zeichenkette,* auch *String*)
- Zeichenketten enden mit einem speziellen "Null-Zeichen", geschrieben als '\0'

Zeichenkette

s

0	1	2	3	4	5	6	7	8	9	10
H	a	l	l	o	!	\0				

```
char s[] = { 'H', 'a', 'l', 'l', 'o', '!', '\0'};
```

oder kürzer

```
char s[] = "Hallo!";
```

Was ist s[0]?

Zeichen einlesen

- Die Funktion Serial.parseInt() liefert eine Zahl von der seriellen Schnittstelle:

```
int n = Serial.parseInt();
```

- Mit Serial.readBytes(*buffer*, *n*) kann man *n* Zeichen in das Feld *buffer* einlesen:

```
char buffer[20];
Serial.parseInt(buffer, n);
```

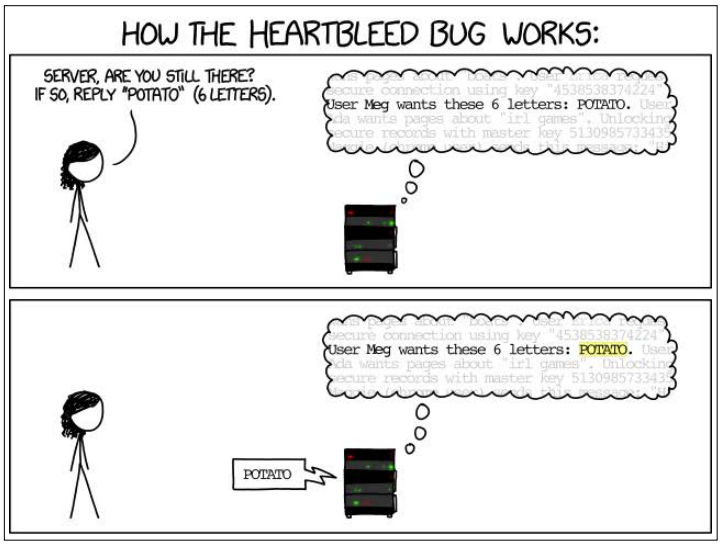
Heartbeat

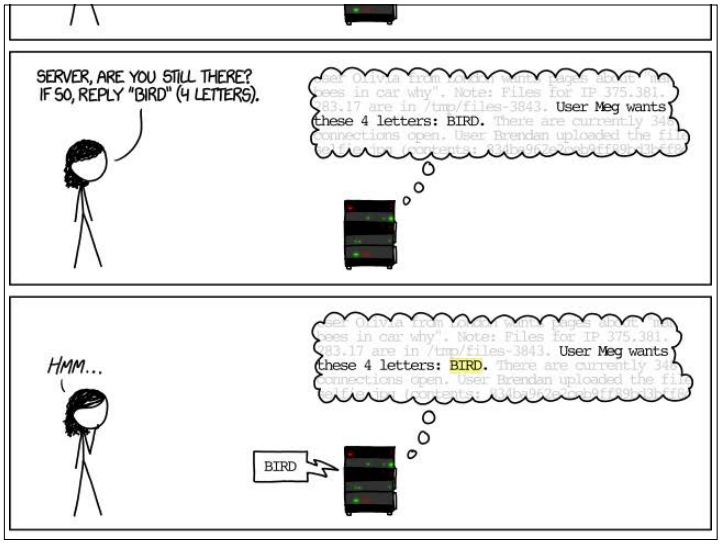
```
void setup() {
  Serial.begin(9600);

  char buffer[10];                // Puffer
  char secret[20] = "Joshua";     // Geheime Daten

  while (1) { // Endlosschleife
    if (Serial.available() > 0) { // Daten verfügbar
     int n = Serial.parseInt();   // Anzahl Zeichen

     Serial.readBytes(buffer, n); // Zeichen einlesen

     for (int i = 0; i < n; i++)  // Zeichen ausgeben
        Serial.print(buffer[i]);
      Serial.println();
    }
  }
}
```

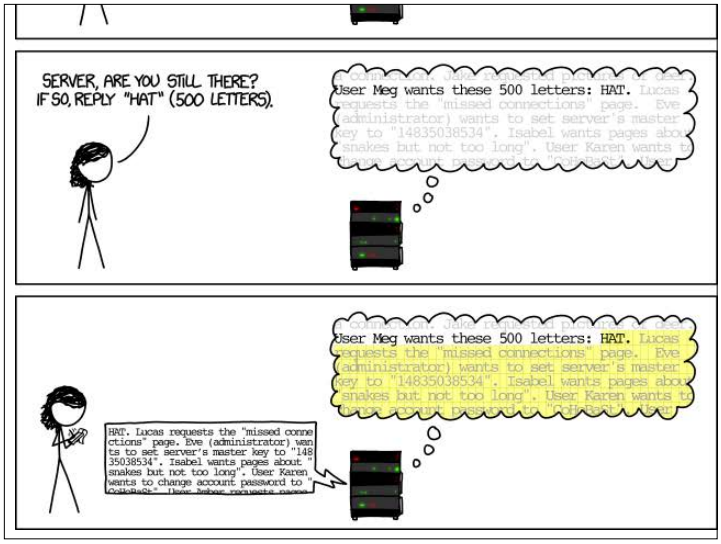
Demo

- Erst Standard-Eingabe; dann zu großen Wert für n angeben

Bei jedem Feldzugriff
Feldgrenzen beachten!







Heartbeat

```
void setup() {
  Serial.begin(9600);

  char buffer[10];                // Puffer
  char secret[20] = "Joshua";     // Geheime Daten

  while (1) { // Endlosschleife
    if (Serial.available() > 0) { // Daten verfügbar
     int n = Serial.parseInt();   // Anzahl Zeichen

     Serial.readBytes(buffer, n); // Zeichen einlesen

     for (int i = 0; i < n; i++)  // Zeichen ausgeben
        Serial.print(buffer[i]);
      Serial.println();
    }
  }
}
```

Heartbleed

```
void setup() {
  Serial.begin(9600);

  char buffer[10];                // Puffer
  char secret[20] = "Joshua";     // Geheime Daten

  while (1) { // Endlosschleife
    if (Serial.available() > 0) { // Daten verfügbar
     int n = Serial.parseInt();   // Anzahl Zeichen

     Serial.readBytes(buffer, n); // Zeichen einlesen

     for (int i = 0; i < n; i++)  // Zeichen ausgeben
        Serial.print(buffer[i]);
      Serial.println();
    }
  }
}
```

Korrektur

```
void setup() {
  Serial.begin(9600);

  char buffer[10];                // Puffer
  char secret[20] = "Joshua";     // Geheime Daten

  while (1) { // Endlosschleife
    if (Serial.available() > 0) { // Daten verfügbar
     int n = Serial.parseInt();   // Anzahl Zeichen
     if (n > 10) { n = 10; }      // Überlauf vermeiden
     Serial.readBytes(buffer, n); // Zeichen einlesen

     for (int i = 0; i < n; i++)  // Zeichen ausgeben
        Serial.print(buffer[i]);
      Serial.println();
    }
  }
}
```

Bei jedem Feldzugriff *Feldgrenzen* beachten!

Bei jedem Feldzugriff *Feldgrenzen* beachten!

Nicht = und == verwechseln!

Wertüberlauf vermeiden!

Ariane 5 Absturz – wegen Überlauf

Vorschau

- Mehr über Zeichenketten
- Interaktion mit LCD-Anzeige

Felder initialisieren

leds

0	1	2	3	4	5	6
13	12	11	10	9	8	7

```
int leds[7];
void setup() {
   leds[0] = 13;
   leds[1] = 12;
   leds[2] = 11;
   leds[3] = 10;
   leds[4] = 9;
   leds[5] = 8;
   leds[6] = 7;
}
```

```
int leds[] =
    { 13, 12, 11, 10, 9, 8, 7 };
void setup() {
  // Bereits initialisiert
}
```

While-Schleifen

```
i = 1;
while (i < 5) {
   Serial.println(i);
   i = i + 1;
}
Serial.println("ENDE");
```

Ausgeführte Anweisungen

```
i = 1;
Serial.println(i);
i = i + 1; // 2
Serial.println(i);
i = i + 1; // 3
Serial.println(i);
i = i + 1; // 4
Serial.println(i);
i = i + 1; // 5
Serial.println("ENDE");
```

Zeichenkette

s

0	1	2	3	4	5	6	7	8	9	10
H	a	l	l	o	!	\0				

```
char s[] = { 'H', 'a', 'l', 'l', 'o', '!', '\0'};
```

oder kürzer

```
char s[] = "Hallo!";
```

Was ist s[0]?

Heartbeat

```
void setup() {
  Serial.begin(9600);

  char buffer[10];                // Puffer
  char secret[20] = "Joshua";     // Geheime Daten

  while (1) { // Endlosschleife
    if (Serial.available() > 0) { // Daten verfügbar
      int n = Serial.parseInt();  // Anzahl Zeichen

      Serial.readBytes(buffer, n); // Zeichen einlesen

      for (int i = 0; i < n; i++)  // Zeichen ausgeben
        Serial.print(buffer[i]);
      Serial.println();
    }
  }
}
```

Handouts

Sensor abfragen

```
int inputPin = 0;   // Der Eingabe-PIN

void setup() {
    Serial.begin(9600);
}

void loop() {
    int level = analogRead(inputPin);

    Serial.println(level);
    delay(100);
}
```

Felder

0	1	2	3	4	5	6	7	8	9	10

- Die Notation *name*[*index*] bezieht sich auf das Element mit der Nummer *index*
- Das erste Element hat die Nummer 0

While-Schleifen

- Mit einer *Schleife* kann man ein Programmstück beliebig oft *wiederholen:*

```
while (Bedingung) {
   Anweisungen…;
}
```

- Die *Anweisungen* werden wiederholt, *solange* die *Bedingung* erfüllt ist

Schleife mit Schleife

```
void loop() {
    int level = analogRead(inputPin);

    if (level < 100)
        digitalWrite(led1, LOW);
    else
        digitalWrite(led1, HIGH);

    if (level < 200)
        digitalWrite(led2, LOW);
    else
        digitalWrite(led2, HIGH);

    if (level < 300)
        digitalWrite(led3, LOW);
    else
        digitalWrite(led3, HIGH);

    // usw. für 4 weitere LEDs
}
```

```
void loop() {
    int level = analogRead(inputPin);

    int i = 0;
    while (i < 7)
    {
        if (level < 100 * (i + 1))
            digitalWrite(leds[i], LOW);
        else
            digitalWrite(leds[i], HIGH);
        i = i + 1;
    }
}
```

Inkrement

- Anstelle von `x = x + 1` oder `x += 1` geht auch `++x` oder `x++`
  (analog `--x` oder `x--` für `x -= 1`)

```
void setup() {
    int i = 0;
    while (i < 7) {
        pinMode(leds[i], OUTPUT);
        i++;
    }
}
```

For-Schleifen

- Weiß man, wie oft eine Schleife wiederholt werden soll, setzt man oft *for-Schleifen* ein:

```
for (Initialisierung; Bedingung; Inkrement) {
   Anweisungen…;
}
```

ist dasselbe wie

```
Initialisierung;
while (Bedingung) {
   Anweisungen…;
   Inkrement;
}
```

Pegel anzeigen

```
void range(int d) {
  for (int i = 0; i < 7; i++)
    if (i < d)
      digitalWrite(leds[i], HIGH);
    else
      digitalWrite(leds[i], LOW);
}


void loop() {
    int level = analogRead(inputPin);
    range(level / 100);
}
```

Heartbeat

```
void setup() {
  Serial.begin(9600);

  char buffer[10];                // Puffer
  char secret[20] = "Joshua";     // Geheime Daten

  while (1) { // Endlosschleife
    if (Serial.available() > 0) { // Daten verfügbar
     int n = Serial.parseInt();   // Anzahl Zeichen

     Serial.readBytes(buffer, n); // Zeichen einlesen

     for (int i = 0; i < n; i++)  // Zeichen ausgeben
        Serial.print(buffer[i]);
      Serial.println();
    }
  }
}
```