

AUTOMADUINO

PROGRAMMIEREN

ARDUINO FÜR MAKERINNEN

Präsentiert von Deborah Fehr



TAGESPLAN

PROGRAMMIEREN

- Programmieren mit Text
- Aufbau Arduino Code
- Programmierstile
- Schleifen
- Ressourcen

9:00

Einführung

11:00

Pause!

11:20

Werkstatt

13:00

PROGRAMMIERUNG

GRAFISCH

Programmieren mit
Blöcken statt Text,
z.B. Ardublock,
Automaduidino

TEXT

klassisches
Programmieren,
schwieriger aber
flexibler

↑
Heute! 😊



AUTOMADUINO AUFBAU

The screenshot displays the Automaduno Editor interface, which is divided into several sections:

- Top Bar:** Features the "Automaduno Editor" title, a settings gear icon, and three mode tabs: "Funktionsmodus", "Kurzmodus", and "Schaltermodus".
- Left Panel (Komponenten):** A vertical list of components including "LED An", "LED Aus", "Summer An", and "Summer Aus".
- Central Canvas:** A visual programming workspace showing a flowchart. It starts with a "Start" node (a play button) leading to an "LED An" block (Pin: 7). This is followed by a "nach" (delay) block set to "1000 ms". The flow then branches to an "LED Aus" block (Pin: 7), which is followed by another "nach" (delay) block set to "1000 ms". The flow returns to the "LED An" block. Labels with arrows identify parts of the diagram: "Einstieg" (Start) points to the start node, "Zustand" (State) points to the LED blocks, and "Übergang" (Transition) points to the delay blocks.
- Right Panel (Code):** A code editor showing the corresponding C++ code. A context menu is open over the code, with options: "Pins zuweisen", "Code kopieren", and "Ende anzeigen". The code includes comments and function definitions for the LED states.

```
1 //Imports:
2 //Pins:
3 int pin_0_led = 7;
4 void setup() {
5   pinMode(pin_0_led, OUTPUT);
6 }
7 void loop() {
8   function_0_led();
9 }
10 void function_0_led(){
11   digitalWrite(pin_0_led, HIGH);
12   delay(1000);
13   function_1_led();
14 }
15 void function_1_led(){
16   digitalWrite(pin_0_led, LOW);
17   delay(1000);
18 }
```

VARIABLEN



INT

für ganze Zahlen, z.B. 5

BOOL

Wahr oder falsch

CHAR

Charakter, entspricht Zeichen



FUNKTIONEN



VERHALTEN

Was soll die Komponente tun?

VORDEFINIERT

z.B. writeDigital, readDigital...

EIGENE FUNKTIONEN

Tun das, was wir wollen



AUFBAU



1

INIT

Definiert Pins und
importiert Bibliotheken

2

SETUP

Komponenten
konfigurieren

3

LOOP

Hier wird der Arduino
Code ausgeführt

4

FUNKTIONEN

Extra Funktionen,
besseres Verständnis

INIT



- lege die Nummern für die Pins fest
- Vorteil: einfaches Ändern bei Umstecken der Pins
- festlegen von Variablen: z.B. für Zähler

- für manche Bauteile (z.B. Servos) brauchen wir Bibliotheken, die wir hier importieren

//Pins:

```
int pin_0_led = 7;
```

//Import:

```
#include <Servo.h>
```




SETUP



- Setup ist eine vordefinierte Funktion und **muss vorhanden** sein!
- läuft genau 1x bei Start des Programms
- legt Belegung der Pins fest: Ist Bauteil INPUT (= Sensor oder Button) oder OUTPUT?
- kann z.B. auch genutzt werden um LED gleich einzuschalten

```
void setup() {  
  pinMode(pin_0_led,  
    OUTPUT);  
}
```



LOOP



- Loop ist eine vordefinierte Funktion und **muss vorhanden** sein!
- Loop ("Schleife") startet am Ende der Funktion wieder von vorne
- Alle Anweisungen werden auf dem Arduino ausgeführt und dann wiederholt
- häufig wird der Code nur in die Loop geschrieben


```
void loop() {  
  function_0_led();  
}
```




FUNKTIONEN



- Wir können eigene Funktionen schreiben und dann aufrufen!
- Vorteil: Code ist gut lesbar, einfache Fehlersuche
- Nachteil: mehr Schreibarbeit
- Code kann auch nur in Loop geschrieben werden, dieser Teil ist daher nicht unbedingt notwendig



```
void function_0_led(){  
  digitalWrite(pin_0_led, HIGH);  
  delay(1000);  
  
  function_1_led();  
}
```





Blink – Programm
nochmal!

AUFGABE

BLINK?



```
pinMode(LED, OUTPUT);
```

```
delay(1000);
```

```
digitalWrite(LED, LOW);
```

```
digitalWrite(LED, HIGH);
```

```
int LED = 7;
```

```
delay(1000);
```



?

```
void setup() {
```

?

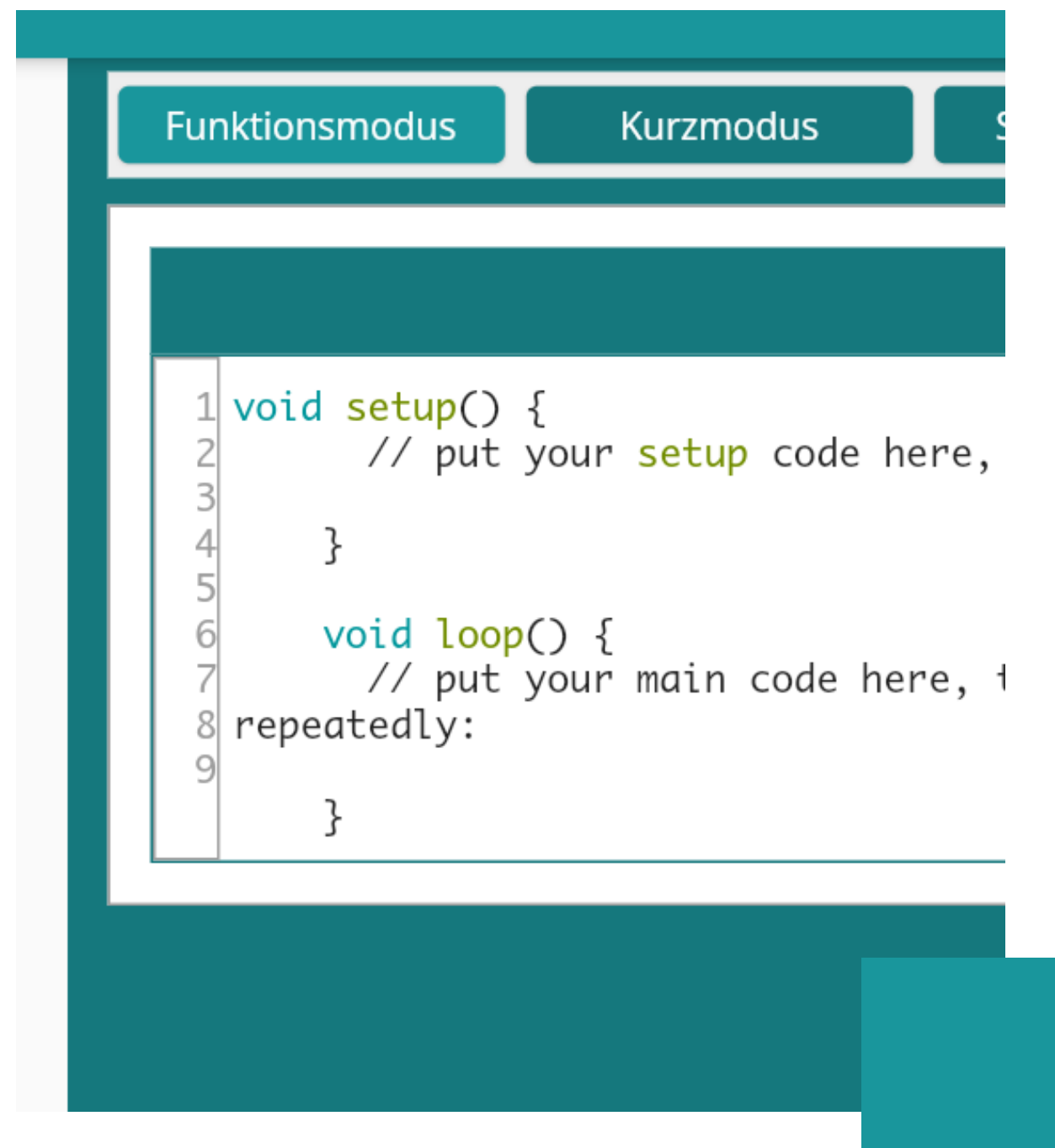
```
}
```

```
void loop() {
```

?

```
}
```


PROGRAMMIER STIL



FUNKTIONEN

bisher unsere
Standardeinstellung,
fügt für jeden Block
eine eigene Funktion
hinzu

KURZ

heute Kurzmodus: fügt
den Code in der Loop
hinzu,
Stil häufig in Tutorials
zu finden

Tipp: Highlight-Funktion benutzen!

KURZMODUS

```
1 //Pins:
2 int pin_0_led = 7;
3
4 void setup() {
5   pinMode(pin_0_led, OUTPUT);
6 }
7
8 void loop() {
9   function_0_led();
10 }
11
12
13 void function_0_led(){
14   digitalWrite(pin_0_led, HIGH);
15   delay(1000);
16   function_1_led();
17 }
18
19 void function_1_led(){
20   digitalWrite(pin_0_led, LOW);
21   delay(1000);
22   function_0_led();
23 }
```

```
1 //Pins:
2 int pin_0_led = 7;
3
4 void setup() {
5   pinMode(pin_0_led, OUTPUT);
6 }
7
8 void loop() {
9   while(true){
10    digitalWrite(pin_0_led, HIGH);
11    delay(1000);
12    digitalWrite(pin_0_led, LOW);
13   }
14 }
```

- Änderung ab Loop Funktion!

SCHLEIFEN

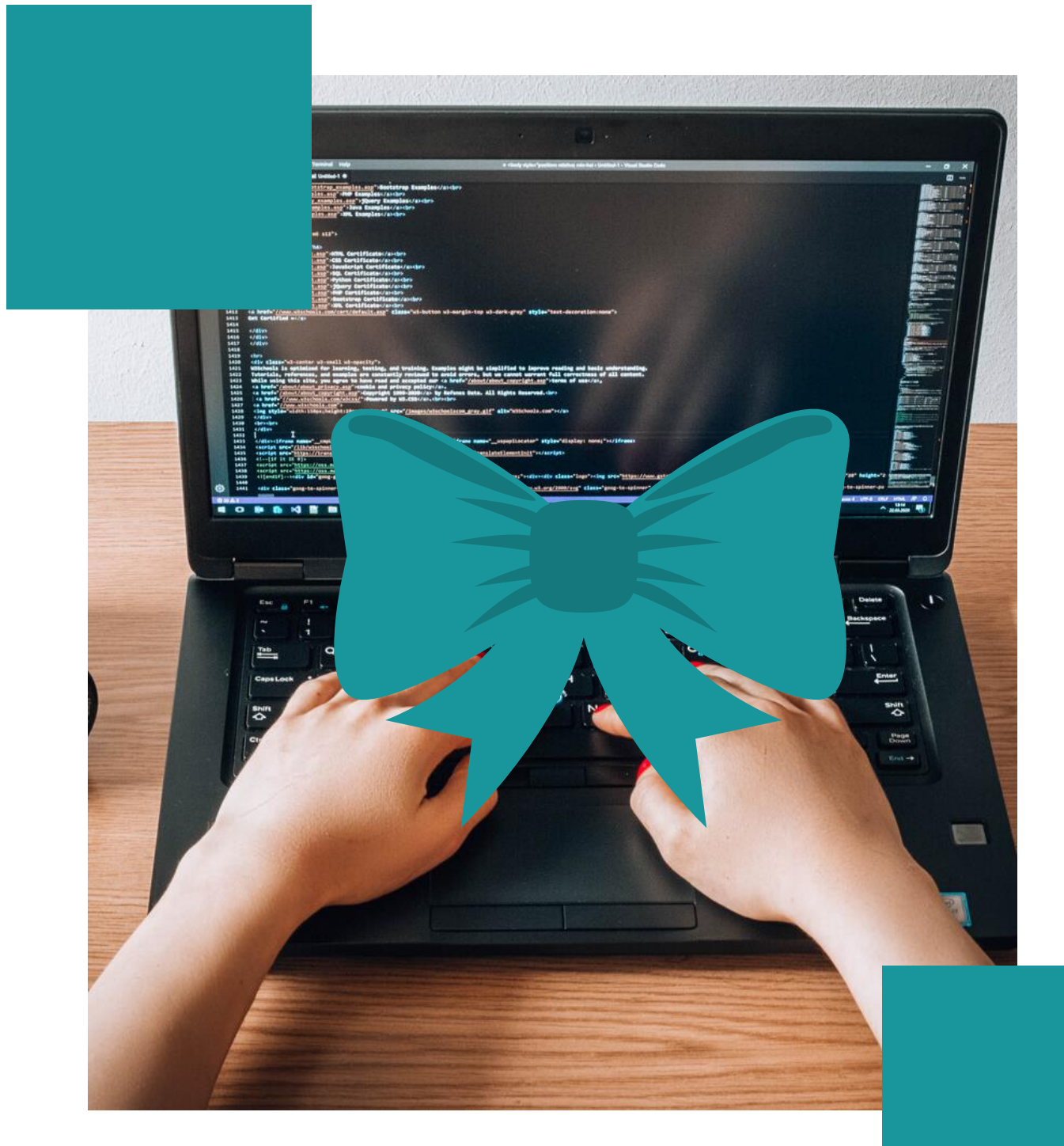
Schleifen werden häufig in der Informatik verwendet und wiederholen Codeblöcke!

FOR

Programmieren mit
Blöcken statt Text,
z.B. Ardublock,
Automaduoino

WHILE

klassisches
Programmieren,
schwieriger aber
flexibler




FOR SCHLEIFE



- führt Code eine bestimmte Anzahl an Wiederholungen aus
- hier: i fängt bei 0 an
- Bedingung: $i \leq 25$, falls wahr führe Code in Klammer aus
- Zähler: nach jedem Durchlauf erhöhe i um 1 (i++)
- Achtung: For Schleife fängt wieder an! Benutze Variable für Ende

```
for (int i = 0; i <= 25; i++) {  
    digitalWrite(summer, HIGH);  
    delay(10);  
}  
  
bool end = false;  
...  
if(end == false){  
    ... Code ...  
    end = true;  
}
```




WHILE SCHLEIFE



- führt Code aus solange eine Bestimmung zutrifft
- hier: messung ist negativ (=0)
- solange der Sensor auf Pin 6 kein Signal bekommt: nochmal messen
- wenn Signal: Schleife wird beendet und LED eingeschaltet
- Achtung: While Schleife kann Programm "lahm legen" wenn sie ewig läuft

```
int messung = 0;  
while (messung == 0){  
  messung = digitalRead(6);  
}  
digitalWrite(7, HIGH);
```





10x
Blink!

AUFGABE

LÖSUNG

```
1 int led = 10;
2 bool end = false;
3
4 void setup() {
5     pinMode(led, OUTPUT);
6 }
7
8 void loop() {
9     if(end == false) {
10         for (int i = 0; i <= 10; i++) {
11             digitalWrite(led, HIGH);
12             delay(500);
13         }
14         end = true;
15     }
16 }
```

MEHR KOMPONENTEN!

- Automaduidino deckt nicht alle Komponenten ab...
- Wenn wir selbst programmieren können wir noch mehr nutzen!
- z.B. Tastenfeld, LCD Bildschirm, Infrarot-Sender ...
- Anleitungen für Teile beispielsweise auf *funduino.de*





BIBLIOTHEK

verwendet Keypad Bibliothek

VARIABLEN

Variablen für Größe und Tasten müssen
festgelegt werden

SERIAL

Serial Monitor: Möglichkeit
Messergebnisse am PC anzuzeigen

TASTENFELD

```
#include <Keypad.h>
//Hier wird die gröÙe des Keypads definiert
const byte COLS = 3; //3 Spalten
const byte ROWS = 4; //4 Zeilen
//Die Ziffern/Zeichen:
char hexaKeys[ROWS][COLS]={
  {'#', '0', '*'},
  {'9', '8', '7'},
  {'6', '5', '4'},
  {'3', '2', '1'}
};
```





Tastenfeld mit Passwort

AUFGABE

LÖSUNG

```
1 #include <Keypad.h>
2 char P1='1';char P2='2';char P3='3';char P4='A'; // Passwort
3 char C1, C2, C3, C4; // Eingabe
4
5 int roteLED = 12; //Die rote LED ist an Pin 12 angeschlossen
6 int grueneLED = 13; //Die grüne LED wird an Pin 13 angeschlossen
7 const byte COLS = 4; //4 Spalten
8 const byte ROWS = 4; //4 Zeilen
9 int z1=0, z2, z3, z4;
10 char hexaKeys[ROWS][COLS]={
11 {'D','#','0','*'},
12 {'C','9','8','7'},
13 {'B','6','5','4'},
14 {'A','3','2','1'}}
15 };
16
17 byte colPins[COLS] = {2,3,4,5}; //Definition der Pins für die 4 Spalten
18 byte rowPins[ROWS] = {6,7,8,9}; //Definition der Pins für die 4 Zeilen
19 char Taste; // = gedrückte Tastae
20 Keypad Tastenfeld = Keypad(makeKeymap(hexaKeys), rowPins, colPins, ROWS, COLS);
21
22 void setup() {
23   Serial.begin(9600);
24   pinMode(roteLED, OUTPUT);
25   pinMode(grueneLED, OUTPUT);
```

```
28 void loop() {
29   Anfang: // Dies ist eine Markierung, zu der per "goto-"Befehl
30   Taste = Tastenfeld.getKey();
31   if (Taste)
32   {
33     if (Taste=='#') // Wenn die Rautetaste gedrückt wurde...
34     {
35       if (C1==P1&&C2==P2&&C3==P3&&C4==P4)
36       {
37         digitalWrite(roteLED, LOW);
38         digitalWrite(grueneLED, HIGH);
39       }
40       {
41         digitalWrite(roteLED, HIGH);
42         digitalWrite(grueneLED, LOW);
43         delay(3000);
44         z1=0; z2=1; z3=1; z4=1;
45         goto Anfang;
46       }
47     }
48     if (z1==0) {
49       C1=Taste;
50       Serial.print(C1);
51       z1=1; z2=0; z3=1; z4=1;
```


Mittagspause



UMFRAGE + QUIZ



Bitte Umfrage ausfüllen

<https://forms.gle/GH4Mj4ZM44CLHHh39>

oder unter

www.automaduino.com

Quiz verfügbar! Teste dein Wissen



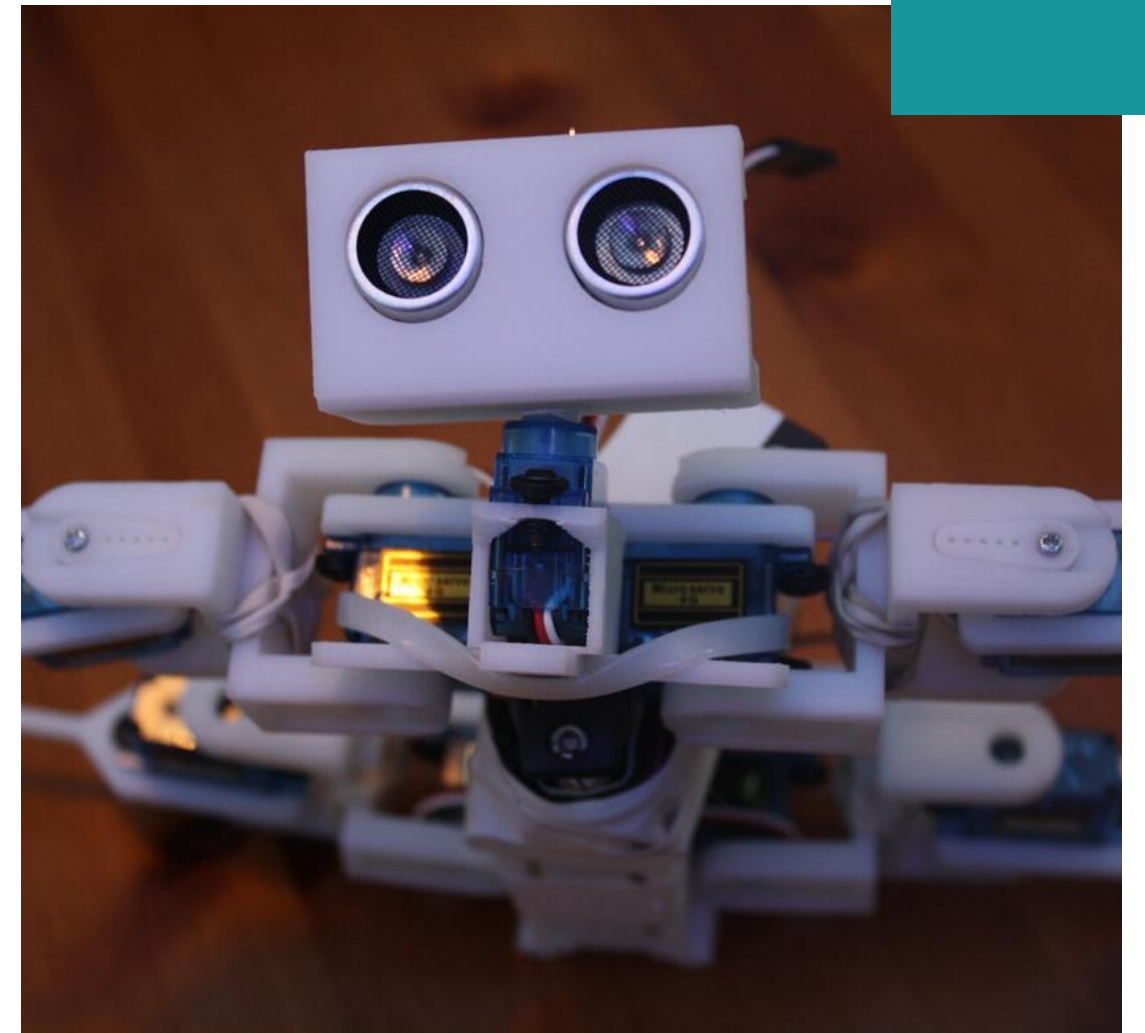
ARBEITSBLÄTTER



LIEDER MIT SUMMER

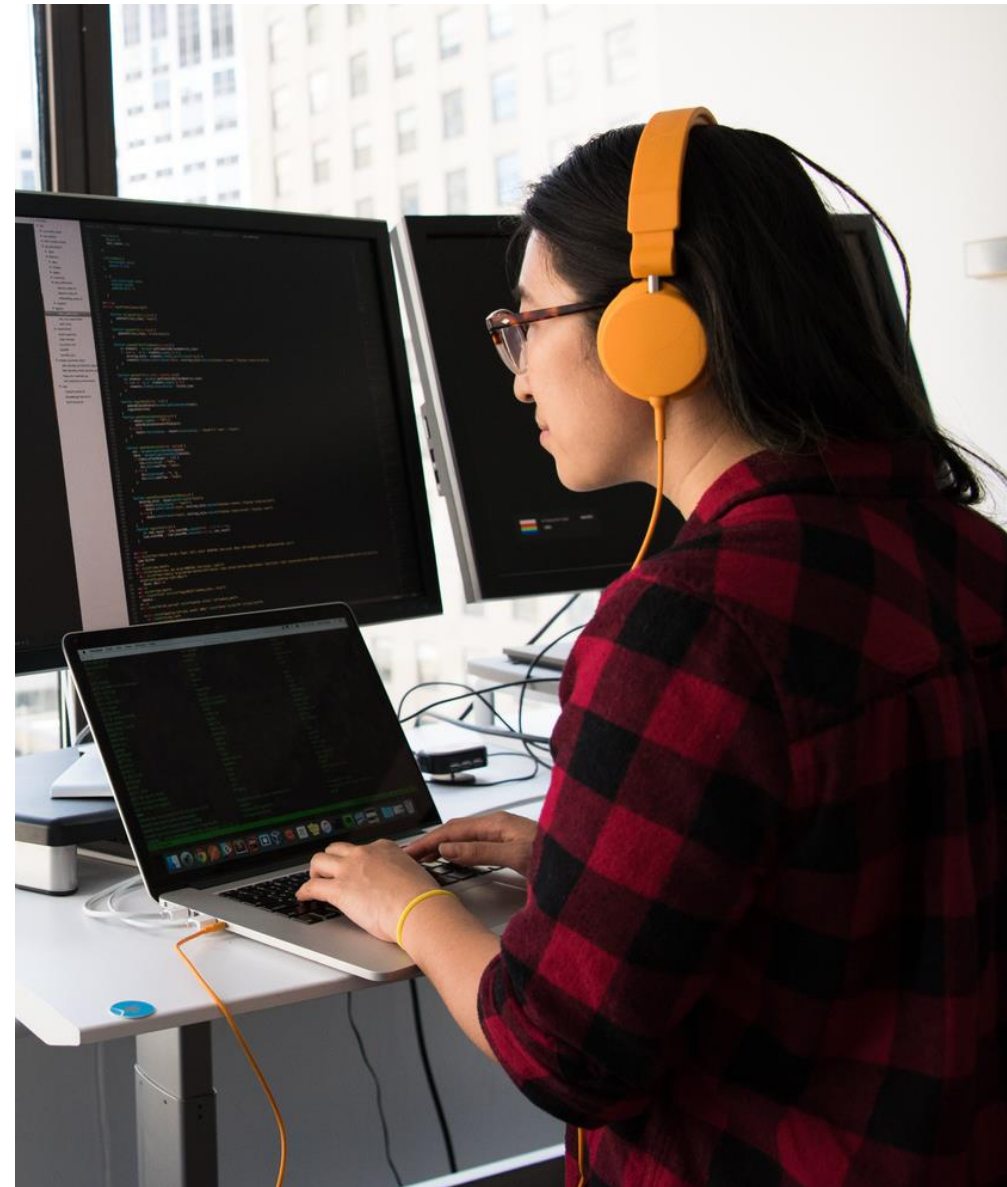


**STRASSEN-
KREUZUNG**



SPASS MIT SERVOS

WAS IST EIN... MAKER?





**DANKE FÜR EURE
TEILNAHME!**

—
Wir heißen euch gerne wieder
im KI-Makerspace willkommen!

