toobo

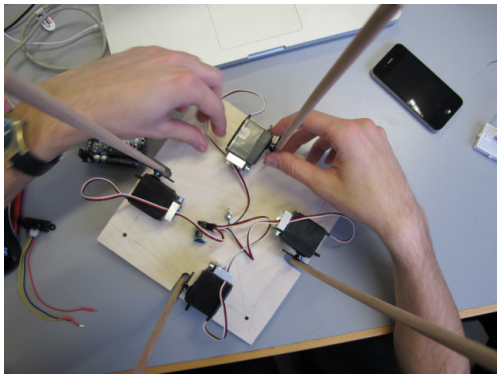
zHdK Interaction Design
Physical Computing
3.Semester
Martin Feigel | Stefan Wanner
Rouven Bühlmann | Sabrina Brunner

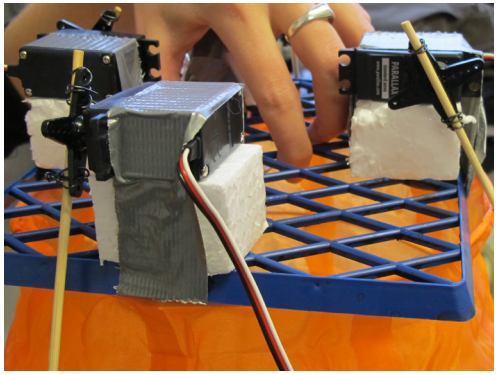
Inhalt:

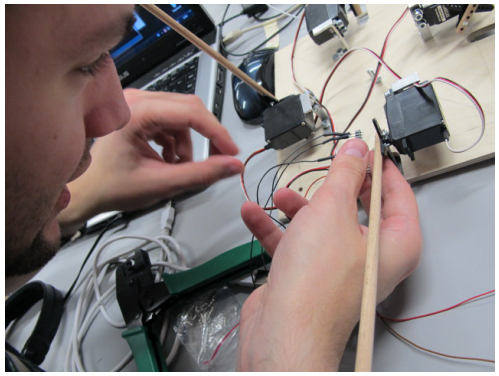
Recherche

Die Grundlage der Ideensuche bildete das Brainstorming. Wir überlegten uns welche Fähigkeiten oder Eigenschaften das Wesen haben könnte. Später überlegten wir uns wie wir diese Fähigkeiten umsetzen können, und entwickelten zusammen neue Ideen die zu dem späteren Konzept von Toobo führten.

Arbeitsprozess







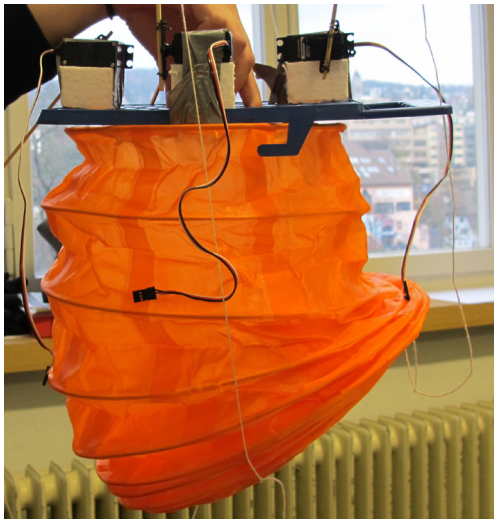
Konzeptbeschrieb

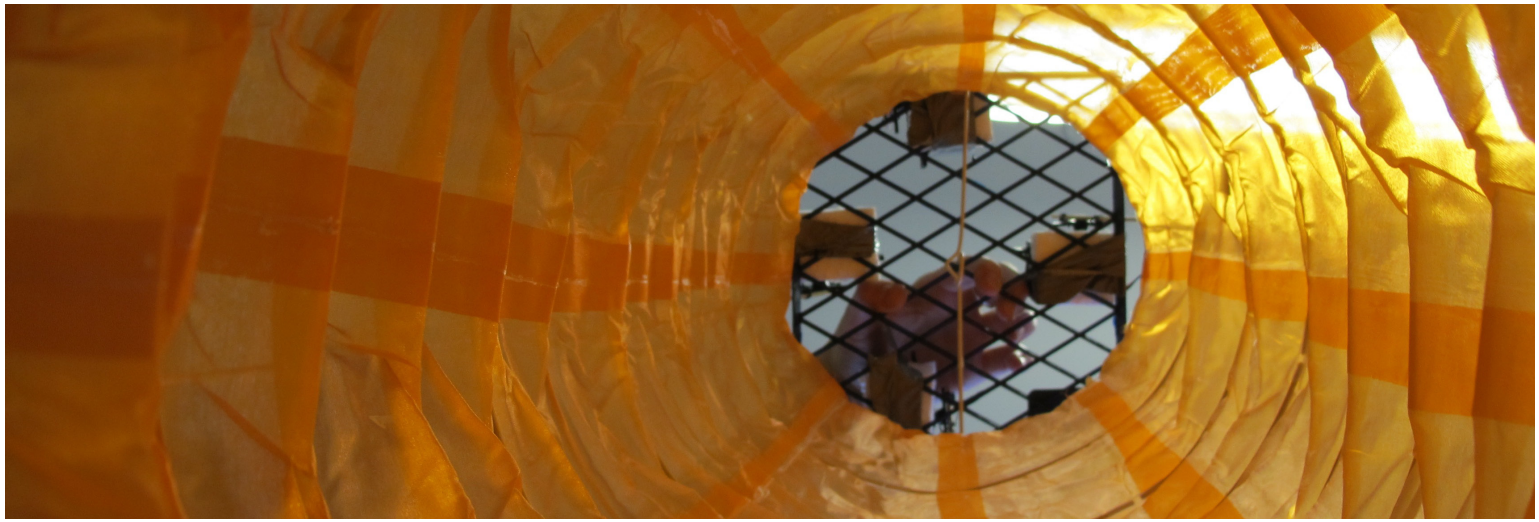
Das Objekt sieht aus wie ein wurmähnliches Wesen. Eine Röhre aus Stoff wird von Oben mit vier Servo-Motoren in Bewegung versetzt. Der Wurm wird wie eine Marionette die an Fäden hängt gesteuert. Er nimmt Bewegungen und Geräusche wahr, und kann auch auf diese reagieren.

Charaktereigenschaften:

scheu | ängstlich | schreckhaft | neugierig | zurückhaltend | misstrauisch | verspielt | vermittelnd | verbindend | sensibel | unterhaltend

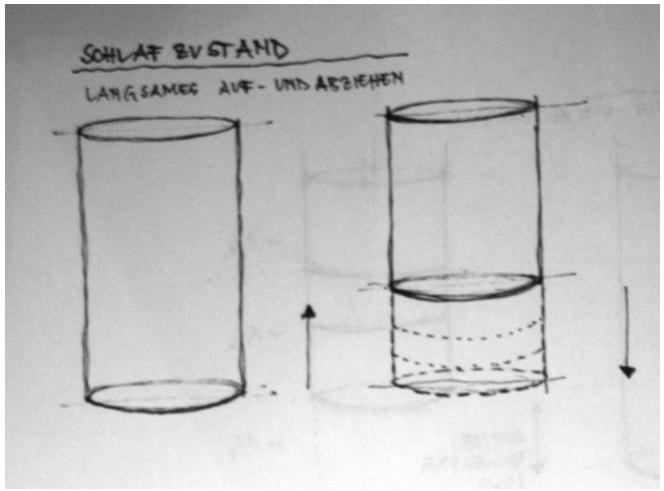
Input		*Processing*				*Output*
eine Person nähert sich langsam	→	Distanzsensor	→	ein Servomotor bewegt sich nach oben	→	er bewegt sich auf die Person zu
eine Person nähert sich schneller	→	Distanzsensor	→	ein Servomotor bewegt sich nach oben	→	er bewegt sich von der Person weg
es passiert nichts	→	alle Senoren sind inaktiv	→	alle Servos bewegen sich ruckartig nach oben und wieder nach unten	→	er schläft ein
ein lautes Geräusch	→	Mikrophon/Klatschschalter	→	alle Servos bewegen sich nach oben	→	er erschreckt sich
...	→	...	→	die gegenüberliegenden Servos bewegen sich abwechsungsweise	→	er sagt nein



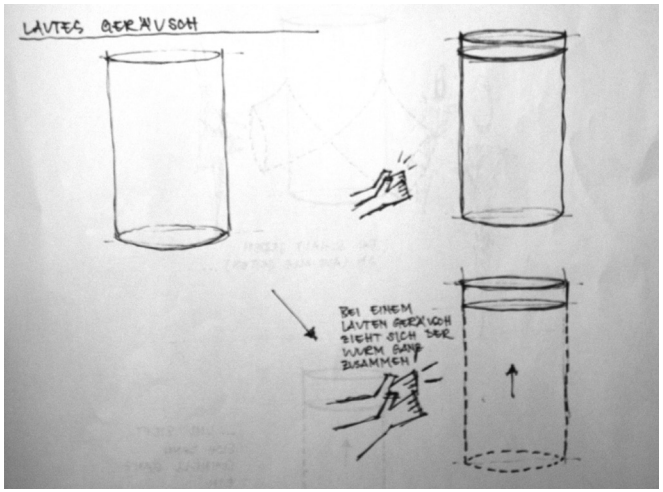


Bewegungsmöglichkeiten

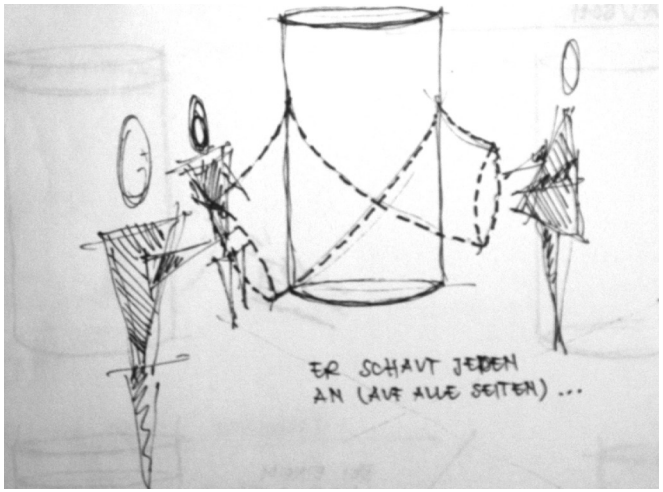
Zustände



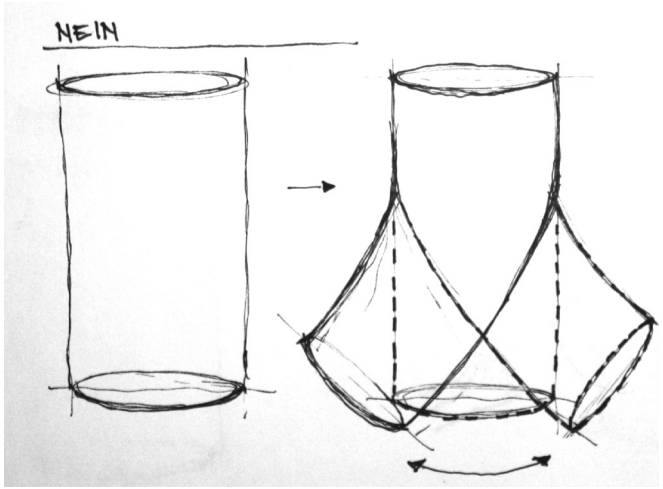
schlafen



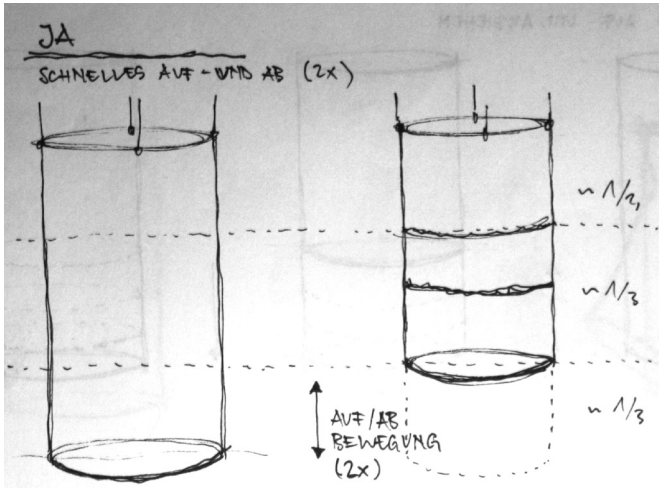
sich erschrecken



eine Person anschauen



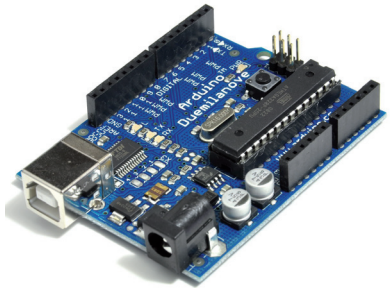
Nein sagen



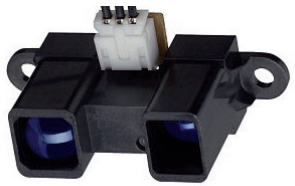
Ja sagen

Elektronik

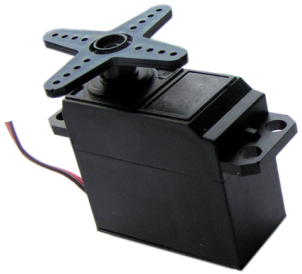
Hauptbestandteile der Elektronik:



Arduino

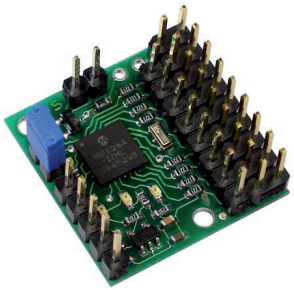


Distanzsensor

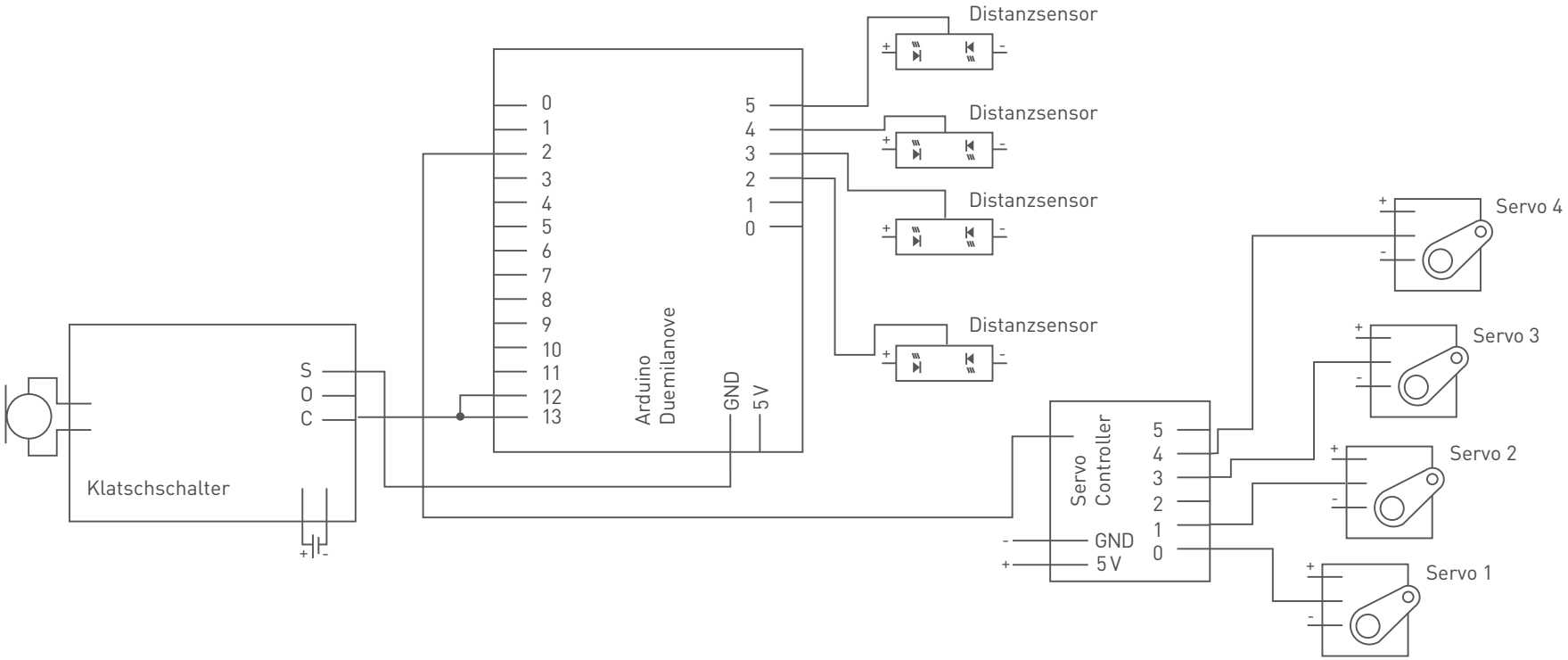


Servomotor

Klatschschalter



Servocontroller



Programmierung

Funktion (auslesen der Sensordaten):

```
void readSensors(boolean inSwitch){
int sensorWert1 = analogRead(SENSOR1);
int sensorWert2 = analogRead(SENSOR2);
int sensorWert3 = analogRead(SENSOR3);
int sensorWert4 = analogRead(SENSOR4);
sensor1A = (sensorWert1 > threshold);
sensor2A = (sensorWert2 > threshold);
sensor3A = (sensorWert3 > threshold);
sensor4A = (sensorWert4 > threshold);

if(inSwitch){
  fillSensorArray(sensorWert1, sensorWert2, sensorWert3, sensorWert4);
 }

 if(clappingAllowed){ clapping = digitalRead(CLAPIN); }
 else{ clapping = false; }

 if(clapping == true){
  Serial.print( „Klatsch-Sensor: „);
  Serial.println( clapping );
 }

 if( sensor1A || sensor2A || sensor3A || sensor4A ){
  Serial.println( „SENSOREN: „ );
  if(sensor1A){
   Serial.print( „Sensor 1: „);
   Serial.println( sensorWert1 );
  } if(sensor2A){
   Serial.print( „Sensor 2: „);
   Serial.println( sensorWert2 );
  } if(sensor3A){
   Serial.print( „Sensor 3: „);
   Serial.println( sensorWert3 );
  } if(sensor4A){
   Serial.print( „Sensor 4: „);
   Serial.println( sensorWert4 );
  }
 }
}
```

Funktion (Servosteuerung):

```
void moveTowards(int inDelay){

 if((scare && scareBack == false)
&& (sensor1A || sensor2A || sensor3A || sensor4A)){
  sleepCounter = 0;
  if(sensor1A && (median(1, true) < threshold)){
   Serial.println(„MOVING AWAY! - @SENSOR1 „);
//    position(3, startPos, 0);
    position(1, startPos, 0);
  }
  if(sensor2A && (median(2, true) < threshold)){
   Serial.println(„MOVING AWAY! - @SENSOR2 „);
//    position(4, startPos, 0);
   position(2, startPos, 0);
  }
  if(sensor3A && (median(3, true) < threshold)){
   Serial.println(„MOVING AWAY! - @SENSOR3 „);
//    position(1, startPos, 0);
   position(3, startPos, 0);
  }
  if(sensor4A && (median(4, true) < threshold)){
   Serial.println(„MOVING AWAY! - @SENSOR4 „);
//    position(2, startPos, 0);
   position(4, startPos, 0);
  }
  scareBack = true;
 }

 if(scare && scareBack == true){
  if(!recent(1, threshold) || !recent(2, threshold) ||
!recent(3, threshold) || !recent(4, threshold) ){
    delay(3000);
    readSensors(false);
    goToPos(endPos, 0);
    Serial.println(„TURNING BACK!“);
  }
  scareBack = false;
 }
}
```

Funktion (Zusammenziehen):

```
void fullRetreat(){
 if( clapping ){
  sleepCounter = 0;
  goToPos(startPos, 3000);
  lookAround();
 }
}
```