Technische Universität Berlin

**Projekt Robotik**
Sommersemester 2014

# Trinkspiel-Bot

**Abschlussbericht**

Niklas Frank Maksim Dill

16. August 2014

Betreuer: Felix Bonowski

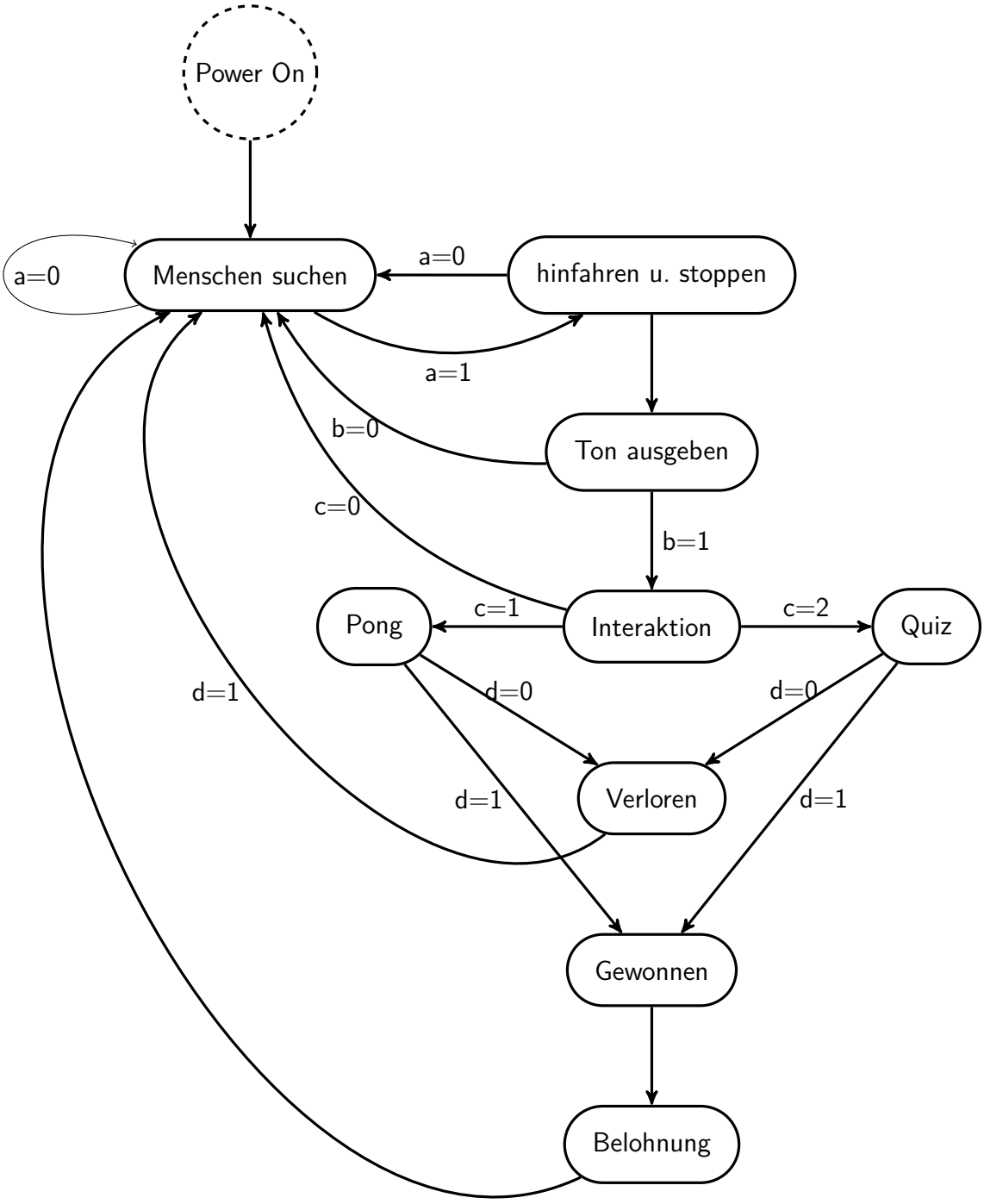
# Inhaltsverzeichnis

# 1 Einleitung

Wir haben uns die Aufgabe gesetzt einen Roboter zu bauen, der in der Lage ist Menschen zu erkennen, zu diesen sich fortzubewegen und anschließend eine Interaktion auszuführen. Die Idee ist dabei, dass nach erfolgreicher Teilnahme an einem Gewinnspiel, ein Getränk ausgeschenkt wird.
Zu Beginn erstellten wir ein Zustandsdiagramm, um die einzelnen Abläufe unseres Roboters gut darzustellen.
Um diese Idee zu verwirklichen, haben wir uns überlegt, welche einzelne Schritte nötig sind, damit die Abläufe nahezu immer ausgeführt werden, um ein geschlossenes System zu erhalten.

Das Zustandsdiagramm sieht dabei wie folgt aus:

## 1.1 Zustandsdiagramm



a = 1 Skeletterkennung
c für Entscheidung der Person
b = 1 Gesichtserkennung
d = 1 Gewonnen

Sobald unser Roboter eingeschaltet wird, fängt er an Leute zu suchen, indem er sich im Kreis schrittweise dreht. Falls jemand getrackt wurde, fährt er zur Person hin. Danach würde der Roboter durch einen Ton auf sich aufmerksam machen und eine Gesichtserkennung starten, um festzustellen, ob die Person zum Roboter gewandt ist. Falls die Person nicht mehr anwesend ist, wird wieder nach Menschen gesucht. Sonst wird die Person gefragt, ob ein Quiz oder Pong gestartet werden soll. Ebenfalls kann man auch ablehnen, woraufhin die Suche nach Menschen wieder beginnt. Wenn man verloren hat, wird wieder nach neuen Menschen gesucht, andernfalls erhält die Person eine Belohnung in Form eines Getränks und danach beginnt es von vorn.

Das Diagramm verdeutlicht, wie ungefähr unsere Umsetzung erfolgen soll. Jedoch ist zu erwähnen, dass das Quiz und die Gesichtserkennung nicht realisiert wurden, aufgrund von Zeitmangel. Der Code dazu ist in Abschnitt 4.9.

# 2 Umsetzung

## 2.1 Einführung

Zu Beginn ist es nötig, sich mit den vorhandenen Materialien vertraut zu machen.
Für unsere Idee verwenden wir die Kinect Kamera und ein im Labor vorhandenes fahrbares Gestell.
Das Gestell besitzt 6 Räder, jeweils 3 auf der rechten und auf der linken Seite. Die Steuerung efolgt über die Motoren, dessen Drehzahl mit einem Arduino gesteuert werden kann.
Aufgrund der besseren Eigenschaften, haben wir uns für den Arduino Uno entschieden, da dieser mehr Ausgänge/Eingänge besitzt.
Für die Kinect Kamera gibt es bereits Bibliotheken, die eine Personerkennung beinhaltet und das Bild auf dem Bildschirm ausgibt. Für die Datenverabeitung benutzen wir ein Notebook. Der Arduino und die Kamera sind per USB an den Laptop angeschlossen.
Um den Code zu verstehen, war es zuerst nötig diesen zu analysieren.

## 2.2 Personenerkennung/Kommunikation

Für die Personenerkennung verwenden wir die Bibliothek SimpleOnenNI. Der Code trackt bereits eine Person und gibt diese anschließend auf dem Bildschirm des Notebooks aus. Der Code kann das Skelett einer Person erkennen und anschließend einen Mittelpunkt des menschlichen Körpers erzeugen. Der Code ist im Abschnit 4.3 zu sehen.

Nachdem unser System ein Skelett erkannt hat, wird sozusagen ein "center of mass"generiert. Die Koordinaten dieses Punktes werden dann weiter an den Arduino versendet.
Es ist somit nötig, dass eine Kommunikation zwischen dem Arduino und dem Processing Programm konfiguriert wird. Damit sich unser Konstrukt zu einer Person hinbewegen kann, senden wir vom Processing Programm die Koordinaten des Mittelpunktes an den Arduino,

welcher wiederum die erhaltenen Daten so verarbeitet, dass das Fahrgestell sich zu der Person fortbewegen kann. Aufgrund des kartesischen Koordinatensystems der Kinect, können wir den Winkel berechnen und diesen zusammen mit der Distanz an den Arduino senden. Näheres zum Winkel folgt in Abschnitt 2.3.2.

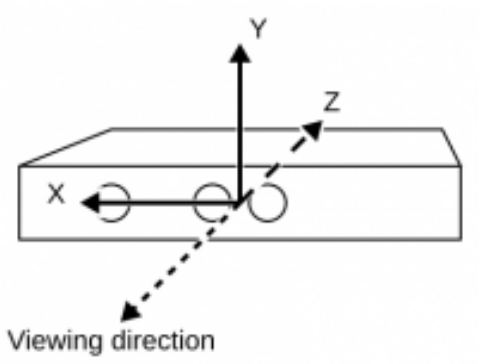



Abbildung 2.1: Kinect System

In Abb. 2.1 ist das System veranschaulicht. Für die serielle Schnittstelle haben wir bereits ein Beispiel auf der Mint Wiki-Seite, was die benötigte Kommunikation aufbaut. Es wird ArduPar auf dem Arduino installiert und anschließend das Code-Beispiel so verändert, dass die Koordinaten empfangen werden können. Dazu erzeugt man ein Objekt des Typs ArduPar und verwendet die write Methode, die dann das Objekt auf den Port schreibt. Das Objekt beinhaltet dabei z. B. den gewüschten Wert. Das Objekt kann dann vom Arduino mit der read Methode gelesen werden. Der dazugehörige, kommentierte Code ist in Abschnitt 4.1, 4.2, 4.6 und in 4.7 zu finden.

## 2.3 Fortbewegung

### 2.3.1 Motorsteuerung

Für die Fortbewegung wird ein 9 V Akku benutzt. Um das Gestell mit den Motoren anzusteuern, wurde dafür wieder eine Bibliothek von der Wiki-Seite benutzt. Die Steuerung der Motoren erfolgt über den TReX. Dieser reguliert die Stromzufuhr des Akkus an die Motoren, um die Drehzahl zu beeeinflussen. Der TReX kommuniziert mit dem Arduino Uno über UART. In Abbildung 2.2 ist die benötigte Verbindung zwischen Arduino und TReX dargestellt.

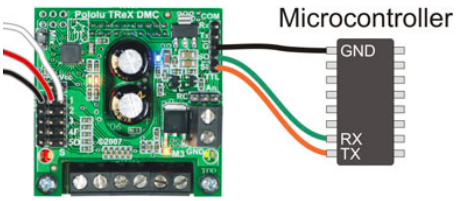



Abbildung 2.2: Verbindung zwischen Arduino und TReX

Die verwendeten Pins vom Arduino sind D12 und D13, da neben der 13 sich GND befindet und mit einem 3 Pin-Kabel beides sich leicht verbinden lässt, wie in Abb. 2.3 verdeutlicht

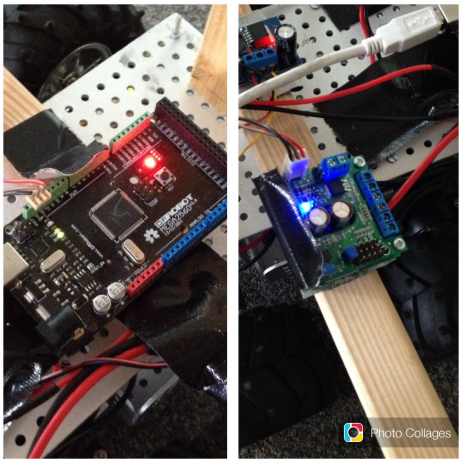



Abbildung 2.3: Verbindung zwischen Arduino und TReX verdeutlicht

Da bereits eine Bibliothek vorhanden ist, können wir die vorgegebenen Methoden, die einen Parameter für die Intensität erhalten, verwenden, um vorwärts, rückwärts, rechts oder links zu fahren. Der Code ist im Abschnitt 4.8 zu finden Es gibt 2 Möglichkeiten, damit sich das Gerät nach rechts oder links hinbewegt. Zu Beginn haben wir die rechte und linke Seite entgegengesetzt drehen lassen. So rotiert das Gestell auf einem Punkt entweder nach rechts oder links. Danach bewegt sich das Gerät vorwärts. Aufgrund aber der schlechten Handhabung, der Instabilität des Kamera-Trackings und von Vibrationen, wurden Funktionen geschrieben, die das Gerät Kurven fahren lässt, um diese Probleme zu umgehen.

### 2.3.2 Bewegung in Abhängigkeit des Winkels

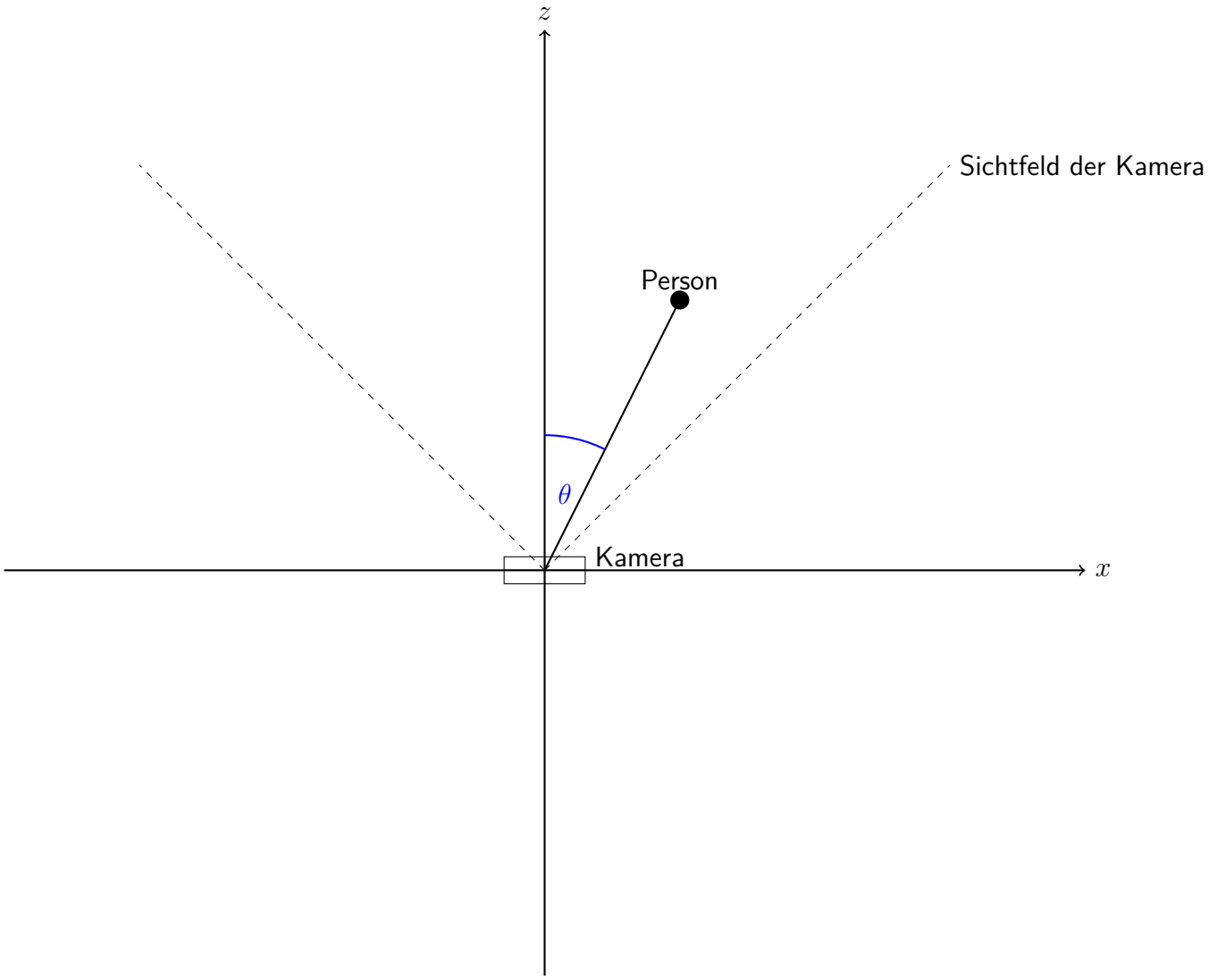



Abbildung 2.4: Darstellung für den gesuchten Winkel

Dieses Bild soll verdeutlichen wie der Roboter zu seinem Ziel kommt. Es ist eine Draufsicht auf die Kinect Kamera bzw. dem ganzen Roboter. Durch die Mitte der Kamera legt Processing ein kartesisches Koordinatensystem. Wenn Kinect eine Person erkennt kann Processing den Anteil auf der z- bzw. x-Achse ausgeben. Diese beiden Informationen und der rechte Winkel reichen aus, um den Winkel Gamma zu berechnen. Umso größer dieser Winkel ist ums so mehr Geschwindigkeit wird auf die linken bzw. rechten Motoren aufaddiert. Diese Rechnung wird immer wieder ausgeführt, sodass der Winkel immer kleiner und die Geschwindigkeit

immer niedriger wird, bis der Roboter im besten Fall sein Ziel mit einem Winkel von 0° erreicht.

## 2.4 Gehäuse

Damit sich der Roboter im Raum frei bewegen kann, schrauben wir auf der gelochten Metallplatte des fahrbahren Untersatz 3 Holzstäbe von ungefähr 80 cm Höhe, um eine angenehme Höhe für den Benutzer zu erreichen. Auf diesen 3 Stäbe wurde eine dünne Holzplatte von 30 cm x 30 cm Fläche angebracht, auf dem sich der Laptop befinden wird. Für die Kamera wird ebenfalls ein Holzstab von 35 cm Höhe angeschraubt, worauf dann die Kamera befestigt wird. Da bei dieser Konstruktion der Roboter leicht nach rechts oder links kippen kann, wurden Stützräder, rechts, links und hinten angebracht. Die Konstruktion ist in Abbildung 2.5 und in Abbildung 2.6 zu sehen.

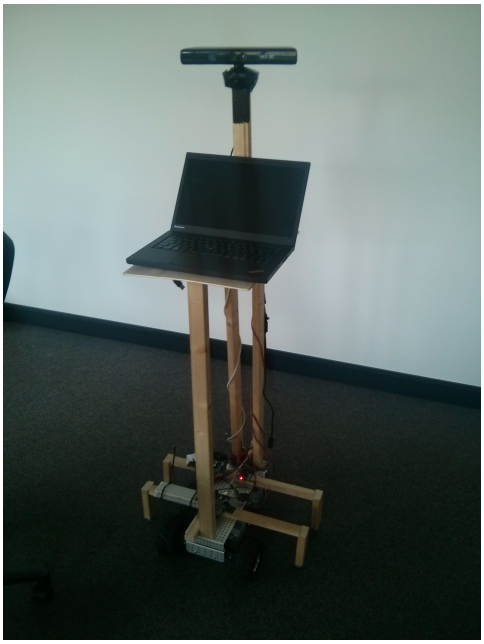


Abbildung 2.5: Trinkbot

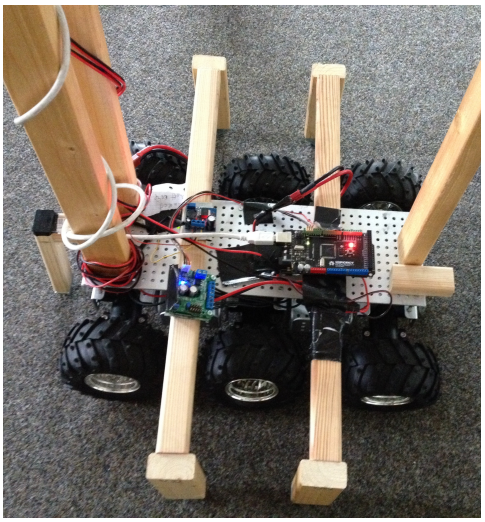


Abbildung 2.6: Stützräder

Der Holzstab für die Kamera wurde nochmals an die Platte befestigt, indem rechts und links Holzblöcke angebracht und miteinaner mit Schrauben verbunden wurden, um das Schwanken zu reduzieren.

Die Kamera wurde fest an den Holzstab getapet, damit die Konstruktion stabiler wird.

Abbildung 2.7: Kinect Kamera

Mit dieser Konstruktion entstand ein robustes Gehäuse. Um aber das Schwanken der Kamera zu reduzieren, damit das Tracking besser funktioniert, wurden funktionen geschrieben, die ein langsames Anfahren und leichtes bremsen ermöglichen. Im Abschnitt 4.8 ist der dazugehörige Code zu sehen.

## 2.5 Elektronik

### 2.5.1 Schalter

Wir haben einen Schalter eingebaut, damit die Stromzufuhr vom Akku zu dem TReX, der die Stromstärke regelt, getrennt werden kann. Einen Vorteil hat dies, dass falls das Gerät in die falsche Richtung fährt, es einfach ausgeschaltet werden kann.
Der Schaltplan dazu:

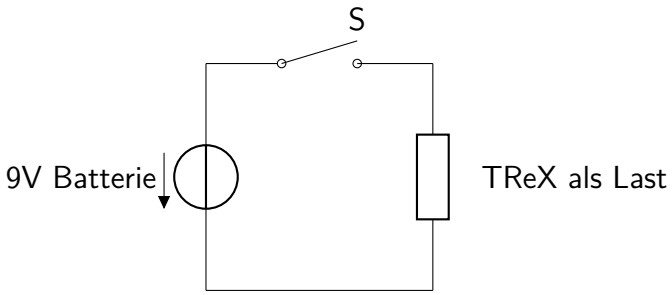



Abbildung 2.8: Schalter für den TReX

Man muss das Kabel zwischen dem Akku und dem TRex durchtrennen und dort einen Schalter an den durchtrennten Kabelenden anlöten, der beim Betätigen den Stromkreis wieder schließt. Der Schalter wurde oben auf der Platte angebracht:

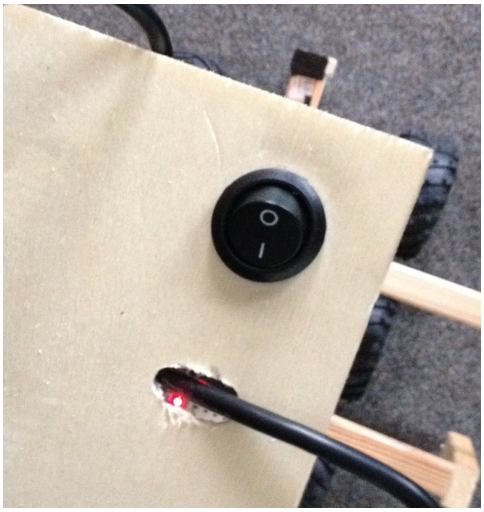


Abbildung 2.9: Schalter

### 2.5.2 Akkuversorgung für die Kinect Kamera

Die Kinect Kamera benötigt eine 12 V Versorgung, die sonst über eine Steckdose erfolgt. Da aber unser Gerät mobil unterwegs ist, ist es nötig dafür einen Akku zu benutzen. Im Labor sind nur 9 V Akkus vorhanden, weshalb eine Schaltung benötigt wird, die die Spannung auf 12 V erhöhen kann. Dafür wird ein Hochsetzsteller verwendet. Der Schaltplan dazu verdeutlicht die benötigte Schaltung:

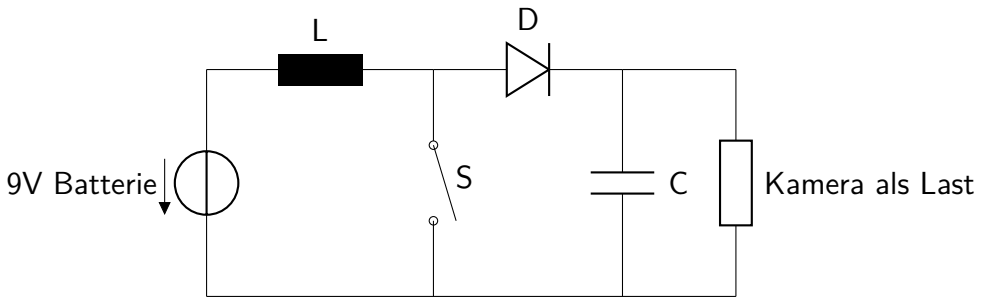



Abbildung 2.10: Hochsetzsteller für die Kinect Kamera

Der Hochsetzer erhöht die 9 V, indem die Spule, die ein Stromspeicher ist, den Kondensator auflädt, bis die 12 V erreicht sind. Die Diode sperrt den Strom in Richtung Spule, sodass die Entladung nur am Widerstand erfolgt. Der Schalter, der einen Transistor darstellt, schließt sich, sobald der Spannungswert am Kondensator sinkt, damit dieser wieder seine 12 V Spannung hat. Wenn der Transistor geschlossen ist, kann sich der Kondensator entladen, da der Strom von der Spule zur Masse fließt. So entsteht ein Spannungs-Rippel am Kondensator, der so gering ist, dass man von einer Gleichspannung ausgehen kann.

Das Kabel wird durchgeschnitten und anschließend an einem männlichen Stecker angelötet.

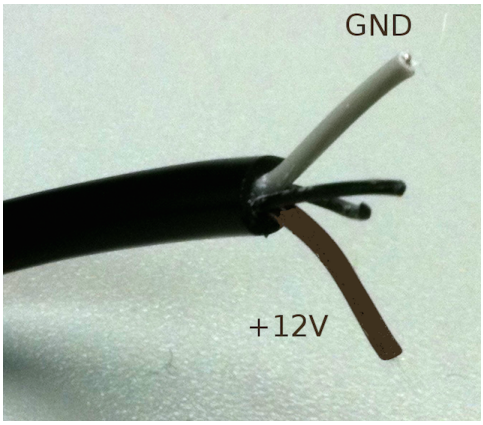



Abbildung 2.11: Kabeldurchtrennung

Das weiße Ende wird an den Schirm vom Stecker angelötet, das rote wird mit der Mitte des Steckers verbunden.

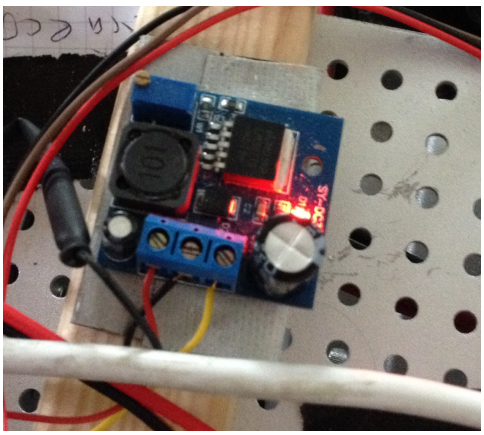


Abbildung 2.12: Hochsetzsteller

Unser Hochsetzsteller in Abb. 2.12 hat 3 Anschlüsse. Die schwarze Verbindung ist für Masse, rot für die Eingangsspannung und gelb für die Ausgangsspannung. Bevor die Verbindungen gemacht werden, muss der Hochsetzsteller mit einem Potentiometer auf 12 V Ausgangsspannung eingestellt werden. Hierbei wird die Eingangsspannung mithilfe einer Spannungsquelle auf 3V gesetzt und anschließend mit einem Multimeter die Ausgangsspannung gemessen. Am Potentiometer wird so lange gedreht, bis man die 12 V erreicht hat.

Das schwarze Ende vom weiblichen Steckerkabel wird mit der Masse vom Hochsetzsteller verbunden, ebenfalls der Minus Anschluss vom Akku. Der Pluspol von der Batterie kommt an den Eingang des Hochsetzstellers. Die Ausgangsspannung wird wieder mit der Mitte des weiblichen Steckers verbunden. Somit können beide Stecker miteinander verbunden werden und es liegen 12 V an der Kinect Kamera an.

Abbildung 2.13: Stecker für die Kinect

So kann dann das andere Ende des weiblichen Stecker mit dem Akku verbunden werden.

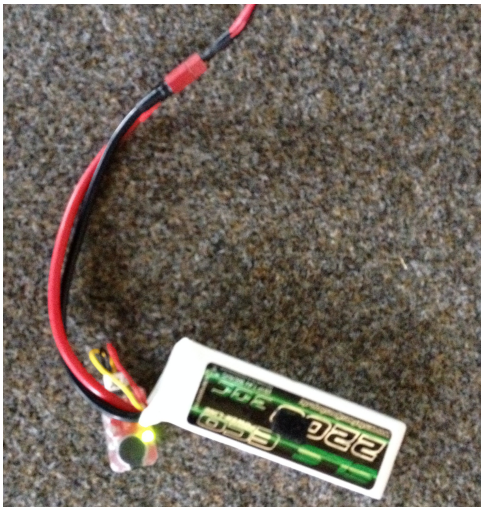


Abbildung 2.14: Akku

Mit diesem Aufbau kann sich unser Roboter mobil im Raum fortbewegen.

## 2.6 Pong

Um einen Indikator dafür zu haben wer die Belohnung bekommt oder ob überhaupt eine Belohnung ausgegeben werden darf, entschieden wir uns das Spiel Pong in Processing zu implementieren. Bevor das eigentliche Spiel startet, ist es möglich aus zwei verschiedenen Darstellungsarten zu wählen. Im nächsten Bild wird kurz die Steuerung erklärt dann wird gespielt. Im Game Over Bildschirm wird letztendlich der Gewinner festgelegt. Um ein Spiel bzw. um Animation darstellen zu können ist es nötig aufeinanderfolgende Frames zu berechnen. Einen Rahmen dafür gibt die Draw-Funktion von Processing. Für jedes neue Bild müssen die Parameter des Balls bzw. der Pongschläger neu berechnet und dargestellt werden. Damit die Kollision zwischen Ball und Schläger stimmt müssen Außerdem Hitboxen festgelegt werden. Um das Spiel noch etwas interessanter zu gestalten wird die Geschwindigkeit des Balls von Zeit zu Zeit erhöht und je nach Spieldarstellung auch die Größe des Balls.

# 3 Ergebnis

Wir haben es geschafft einen Roboter zu bauen, der ein stabiles Gehäuse hat und in der Lage ist, Menschen zu erkennen und zu diesen hinzufahren. Nachdem dies ausgeführt wurde, startet ein Menü, wo man Pong auswählen kann. Nachdem man gespielt hat, wird angezeigt, ob man gewonnen oder verloren hat.

Uns ist es jedoch nicht gelungen danach ein Belohnungssystem zu bauen, dass z. B. ein Getränk bereitstellt. Zudem müsste noch ein System entwickelt werden, dass Objekte erkennt und diese umfährt. Beides ist machbar, leider haben wir es in der vorgegebenen Zeit nicht geschafft, aufgrund von Problemen mit dem fahrbahren Untersatz, da die erste Vorgehensweise der Bewegungssteuerung zwar funktionierte, jedoch mit erheblichen Komplikationen. Außerdem nahm es viel Zeit in Ansprch einen neu geschriebenen Code zu testen, da jedesmal compiliert und hochgeladen werden musste.

# 4 Code

In diesem Abschnitt befindet sich unser Code.

## 4.1 main Kinect Code

```
1  /*
     ------------------------------------------------------------------
2   * SimpleOpenNI User Test
3   *
       ------------------------------------------------------------------
4   * Processing Wrapper for the OpenNI/Kinect 2 library
5   * http://code.google.com/p/simple-openni
6   *
       ------------------------------------------------------------------
7   * prog:  Max Rheiner / Interaction Design / Zhdk /
       http://iad.zhdk.ch/
8   * date:  12/12/2012 (m/d/y)
9   *
       ------------------------------------------------------------------
10  */
11
12
13 ////////////////////////////////////////////////////////// global
     //////////////////////////////////////////////////////////
14
15 import SimpleOpenNI.*;
16 /**
```

```
17  * SendSomeInt
18  *
19  * Send command for setting an ArduPar parameter to a
       Serial Port
20  * This example is in the public domain.
21  */
22 import processing.serial.*;
23
24
25
26 Serial myPort;  // Create object from Serial class
27
28 int curNumber=1; // The Number we will send to the Arduino
29
30 float x , z = 0;
31
32 SimpleOpenNI context;
33
34 color[]       userClr = new color[]{ color(255,0,0),
      color(0,255,0), color(0,0,255), color(255,255,0),
      color(255,0,255), color(0,255,255)};
35
36 PVector com = new PVector();
37
38 PVector com2d = new PVector();
39
40
41 int previousMillis = 0;
42 boolean millisVergeben = false;
43 ///////PONG Varibalen////////////
44 boolean pongActive = false;
45 int brightness = 250;// Farbwert für die Helligkeit
46 int brightnessSpeed = 5;// geschwindigkit mit der sich
      brightness größer bzw. niedriger wird
47 int ballSize= 1;
```

```
48 int ballSizeMax = 20;
49 int ballSizeSpeed = 1;
50 int ballX,ballY = 0;// Koordinaten d. Balls
51 int ScoreL, ScoreR = 0; // Punkte d. linken u. rechten
     Spielers
52 int changeX = -5;// Geschwindigkeit der Ballbewegung
53 int changeY = -5;
54 int gameCount = 0;
55 int screenCount = 0;//zum switchen zwischen den verschiedenen
     Bildschirmen
56 int tapCount = 0;// zählt die Berührungen von Schlägern und
     Ball
57 boolean firstTap = false;
58 int rectPosL, rectPosR = 0;// variable Y-Position der
     Schläger
59 boolean qPress,aPress,oPress,lPress = false;// kontrolle
     ob eine Taste gedrückt ist
60 boolean choiceL, choiceR, enter = false;
61 int IncDec = 250; //Inkrement bzw. Dekrement für brightDark
     funktion
62 int speed = 5; // geschwindigkeit für IncDec
63  ////////////////////////////////////////////////////// global
      end ////////////////////////////////////////////////////////
64
65
66 ////////////////////////////////////////////////////// setup
     ////////////////////////////////////////////////////////
67
68 void setup()
69 {
70
71 initialization();
72
73 }
```

```
74 ////////////////////////////////////////////////////// setup
     end //////////////////////////////////////////////////////
75
76
77
78 ////////////////////////////////////////////////////// draw
     //////////////////////////////////////////////////////
79
80 void draw()
81 {
82 if(pongActive == true){
83
84 //sketchFullScreen();
85  playPong();
86 }else{
87 tracking();
88 }
89
90  //---------------UART---------------start
91 /*myPort.write("someInt "+curNumber+"\n"); //send a
     line containing the name and value separated by a
     space to the Serial.
92 curNumber++;   // count on...
93 if (curNumber>255)curNumber=0; // start over at zero
     once we have reached 255...*/
94
95
96  //---------------UART---------------end
97
98
99
100
101 }
```

```
102 ////////////////////////////////////////////////////// draw end
    //////////////////////////////////////////////////////
```

Listing 4.1: main Kinect Code

## 4.2 initialisiere Kinect

```
1
2
3  void initialization()
4  {
5
6  //-------------------UART-----------start
7  // I know that the last port in the serial list on my PC
8  // is always my FTDI adaptor, so I open
      Serial.list()[Serial.list().length-1].
9  // If the whole thing does not work, try a different port...
10 // you can also just directly use the port name as a parameter:
11 // (i.e. type '"COM18" instead of
      'Serial.list()[Serial.list().length-1]')
12
13 // Open whatever port is the one you're using.
14 String portName =
      Serial.list()[Serial.list().length-1];
15
16 myPort = new Serial(this, portName, 57600);
17 //open a connection with 115200Baud
18 //- this has to match the Baudrate in your Arduino sketch!
19
20 frameRate(60);
21 //---------------------UART----------end
22
23 //size(1600,900);
24 size(640,480);
25 context = new SimpleOpenNI(this);
26
27 if(context.isInit() == false)
28 {
29
```

```
30  println("Can't␣init␣SimpleOpenNI,␣maybe␣the␣camera␣is␣
      not␣connected!");
31   exit();
32   return;
33
34  }
35
36  // enable depthMap generation
37  context.enableDepth();
38
39  // enable skeleton generation for all joints
40  context.enableUser();
41
42  background(200,0,0);
43
44  stroke(0,0,255);
45  strokeWeight(3);
46  smooth();
47  ////////////////////PONG intialisation
48
49  }
```

Listing 4.2: initialisiere Kinect

## 4.3 Tracking

```
1
2  void tracking()
3  {
4
5 /* float[] data=readLineFromSerial(myPort);
6
7 }*/
8
9  /*if(context.isTrackingSkeleton == false){
10 if(data.length>0 && data[0] == 1){
11 myPort.write("someX "+0+"\n");
12 println("x = 0" + 0);
13 }
14 if(data.length>0 && data[0] == 2){
15 myPort.write("someZ "+0+"\n");
16 println("z = 0 " + 0);
17 }
18 }*/
19
20
21 // update the cam
22 context.update();
23
24 // draw depthImageMap
25 //image(context.depthImage(),0,0);
26 image(context.userImage(), 0, 0);
27
28 // draw the skeleton if it's available
29 int[] userList = context.getUsers();
30
31
32 for(int i = 0; i < userList.length; i++)
33 {
```

```
34 println(userList[i]);
35
36  if(context.isTrackingSkeleton(userList[0]))
37  {
38
39  stroke(userClr[ (userList[i] - 1) % userClr.length ]
       );
40 // drawSkeleton(userList[i]);
41
42 // circleForAHead(userList[i]);
43 //send a line containing the name and value separated by a space
      to the Serial.
44
45
46 }
47
48 ////////////////////////////////////////////////////////// draw the
      center of mass
      //////////////////////////////////////////////////////////
49
50 if(context.getCoM(userList[0], com))
51 {
52
53  context.convertRealWorldToProjective(com, com2d);
54  stroke(0, 0, 255);
55  strokeWeight(2);
56
57  beginShape(LINES);
58 vertex(com2d.x, com2d.y - 5);
59 vertex(com2d.x, com2d.y + 5);
60 vertex(com2d.x - 5, com2d.y);
61 vertex(com2d.x + 5, com2d.y);
62 endShape();
63
64 fill(0,255,100);
```

```
65
66 text(Integer.toString(userList[0]), com2d.x, com2d.y);
67  z = com.z;
68  x = com.x;
69  if(z < 700 && z > 500 ){
70  if(millisVergeben == false){
71  millisVergeben = true;
72 previousMillis = millis();
73  }
74  int currentMillis = millis();
75  if(currentMillis - previousMillis >= 3000){
76 pongActive = true;
77  }
78
79  }else{
80 millisVergeben = false;
81  }
82
83
84  ////////////////////////////////////////
85
86  float[] data=readLineFromSerial(myPort);
87 if(data.length>0 && data[0] == 1){
88  println("x " + x);
89  myPort.write("someX "+x+"\n");
90 }
91  if(data.length>0 && data[0] == 2){
92  myPort.write("someZ "+z+"\n");
93 println("z " + z);
94 }
95 /*if(data.length>0 && data[0] == 3){
96 pongActive = true;
97 }*/
98
99  // check if data was received...
```

```
100
101  ////////////////////////////////////////
102
103
104
105 }
106
107 }
108 ////////////////////////////////////////////////////////// draw the
       center of mass end
       //////////////////////////////////////////////////////
109
110  }
111
112
113
114  //////////////////////////////////////////////////////
        drawSkeleton
        //////////////////////////////////////////////////////
115 // draw the skeleton with the selected joints
116
117 void drawSkeleton(int userId)
118 {
119
120 // to get the 3d joint data
121 /*
122 PVector jointPos = new PVector();
123 context.getJointPositionSkeleton(userId,SimpleOpenNI.SKEL_NECK,joi
124 println(jointPos);
125 */
126
127 context.drawLimb(userId, SimpleOpenNI.SKEL_HEAD,
       SimpleOpenNI.SKEL_NECK);
128
```

```
129 context.drawLimb(userId, SimpleOpenNI.SKEL_NECK,
      SimpleOpenNI.SKEL_LEFT_SHOULDER);
130 context.drawLimb(userId,
      SimpleOpenNI.SKEL_LEFT_SHOULDER,
      SimpleOpenNI.SKEL_LEFT_ELBOW);
131 context.drawLimb(userId, SimpleOpenNI.SKEL_LEFT_ELBOW,
      SimpleOpenNI.SKEL_LEFT_HAND);
132
133 context.drawLimb(userId, SimpleOpenNI.SKEL_NECK,
      SimpleOpenNI.SKEL_RIGHT_SHOULDER);
134 context.drawLimb(userId,
      SimpleOpenNI.SKEL_RIGHT_SHOULDER,
      SimpleOpenNI.SKEL_RIGHT_ELBOW);
135 context.drawLimb(userId,
      SimpleOpenNI.SKEL_RIGHT_ELBOW,
      SimpleOpenNI.SKEL_RIGHT_HAND);
136
137 context.drawLimb(userId,
      SimpleOpenNI.SKEL_LEFT_SHOULDER,
      SimpleOpenNI.SKEL_TORSO);
138 context.drawLimb(userId,
      SimpleOpenNI.SKEL_RIGHT_SHOULDER,
      SimpleOpenNI.SKEL_TORSO);
139
140 context.drawLimb(userId, SimpleOpenNI.SKEL_TORSO,
      SimpleOpenNI.SKEL_LEFT_HIP);
141 context.drawLimb(userId, SimpleOpenNI.SKEL_LEFT_HIP,
      SimpleOpenNI.SKEL_LEFT_KNEE);
142 context.drawLimb(userId, SimpleOpenNI.SKEL_LEFT_KNEE,
      SimpleOpenNI.SKEL_LEFT_FOOT);
143
144 context.drawLimb(userId, SimpleOpenNI.SKEL_TORSO,
      SimpleOpenNI.SKEL_RIGHT_HIP);
145 context.drawLimb(userId, SimpleOpenNI.SKEL_RIGHT_HIP,
      SimpleOpenNI.SKEL_RIGHT_KNEE);
```

```
146 context.drawLimb(userId, SimpleOpenNI.SKEL_RIGHT_KNEE,
      SimpleOpenNI.SKEL_RIGHT_FOOT);
147
148 }
149 ////////////////////////////////////////////////////////
      drawSkeleton end
      ////////////////////////////////////////////////////////
150
151
152
153 ////////////////////////////////////////////////////////
      circleForAHead
      ////////////////////////////////////////////////////////
154 // draws a circle at the position of the head
155
156 void circleForAHead(int userId)
157 {
158
159 // get 3D position of a joint
160 PVector jointPos = new PVector();
161 context.getJointPositionSkeleton(userId,
      SimpleOpenNI.SKEL_HEAD, jointPos);
162  /*println("xWert"+jointPos.x);
163  println("yWert"+jointPos.y);
164  println("zWert"+jointPos.z);
165
166 c=sqrt((jointPos.x)+(jointPos.y)+(jointPos.z));
167 println("c " + c);
168
169  myPort.write("someInt "+c+"\n");*/
170 // convert real world point to projective space
171 PVector jointPos_Proj = new PVector();
172 context.convertRealWorldToProjective(jointPos,
      jointPos_Proj);
173
```

```
174 // a 200 pixel diameter head
175 float headsize = 200;
176
177 // create a distance scalar related to the depth (z dimension)
178 float distanceScalar = (525/jointPos_Proj.z);
179
180 // set the fill colour to make the circle green
181 fill(0, 255, 0);
182
183 // draw the circle at the position of the head with the head size
      scaled by the distance scalar
184 ellipse(jointPos_Proj.x, jointPos_Proj.y,
      distanceScalar*headsize, distanceScalar*headsize);
185
186 }
187 ////////////////////////////////////////////////////////////
      circleForAHead end
      ////////////////////////////////////////////////////////////
188
189
190 // --------------------------------
191 // SimpleOpenNI events
192
193
194 ////////////////////////////////////////////////////////////
      SimpleOpenNI events
      ////////////////////////////////////////////////////////////
195
196 void onNewUser(SimpleOpenNI curContext, int userId)
197 {
198
199 println("onNewUser - userId: " + userId);
200 println("\tstart tracking skeleton");
201
202 curContext.startTrackingSkeleton(userId);
```

```
203
204 }
205
206 void onLostUser(SimpleOpenNI curContext, int userId)
207 {
208
209 println("onLostUser - userId: " + userId);
210
211 }
212
213 void onVisibleUser(SimpleOpenNI curContext, int userId)
214 {
215
216 //println(önVisibleUser - userId: "+ userId);
217
218 }
219
220
221  /* void keyPressed()
222 {
223
224 switch(key)
225 {
226
227 case ' ':
228 context.setMirror(!context.mirror());
229 break;
230
231 }
232
233 }  */
234
235 float[] readLineFromSerial(Serial  port) {
236 byte temp[] = port.readBytesUntil('\n'); //read data from
      buffer until a new line is finished
```

```
237
238 // check if any data is available
239 if (temp == null) {
240 return new float[0]; //got nothing - return an empty array
241 }
242 else {
243 String inBuffer= new String(temp); //convert raw data to a
      string
244 inBuffer=trim(inBuffer); //cut off whitespace
245 float[] numbers=float(split(inBuffer, "\t")); //cut into
      pieces at tab-stops and convert to an array of floats
246 return  numbers; //return the numbers we got
247 }
248 }
249
250 ////////////////////////////////////////////////////////////
      SimpleOpenNI events end
      ////////////////////////////////////////////////////////////
251
252 //--------------------------------
```

Listing 4.3: Tracking

## 4.4 Start Pong

```
1
2
3  void playPong(){
4
5  if(ballY<0) //Spielfeldbegrenzungen für den Ball
6  {
7  changeY= -changeY;
8  }
9  if(ballY>height)
10 {
11 changeY= -changeY;
12 }
13
14
15 if(rectPosL<0 ){ //Pongschläger: Bewegungsgrenzen
16 qPress = false;
17 }
18 if (rectPosL>height-200){
19 aPress = false;
20 }
21
22 if(rectPosR<0 ){
23 oPress = false;
24 }
25 if (rectPosR>height-200){
26 lPress = false;
27 }
28
29 if(ballX <= 54 && ballY >= rectPosL && ballY <=
      rectPosL+200 && ballX <= 34){  //Berührung und abprallen
      des Balls vom Schläger
30 changeX =-changeX;
31 firstTap = true;
```

```
32 tapCount++;
33 }
34
35 if(ballX >= width-54 && ballY >= rectPosR && ballY <=
     rectPosR+200 && ballX <= width-34){
36 changeX =-changeX;
37 firstTap = true;
38 tapCount++;
39 }
40
41
42  if(ballX <=0){
       //Punktevergabe, Reset des Balls auf
       Startkoordinaten
43  ScoreR++;
44  gameCount++;
45  ballY = height/2;
46  ballX = width/2;
47  }
48
49  if(ballX >=width)
50  {
51 ScoreL++;
52 gameCount++;
53 ballY = height/2;
54 ballX = width/2;
55  }
56
57 rectMovement();                                    //Bewegung
     der Schläger
58
59
60 switch (screenCount){
61 case 0:{
     //Startbildschirm: classic oder
```

```
      Fancy?
62  textSize(50);
63  background(0);
64  fill(255);
65  textSize(40);
66  text("Press␣LEFT␣or␣RIGHT.␣Press␣ENTER␣to␣
      accept.",380,800);
67  //text("Let us play PONG!",700,100);
68  textSize(50);
69  text("CLASSIC", width/4,height/2);
70  text("FANCY", width/4*3-150,height/2);
71  //text(Ënter"+enter,400,400);
72  choiceFeedback();
73  if(enter == true && choiceL == true){
74  screenCount = 1;
75  enter = false;
76  }
77  if(enter == true && choiceR == true){
78  screenCount = 1;
79  enter = false;
80
81  }
82  break;
83  }
84
85  case 1:{                                    // Erklärung der
      Steuerung
86  textSize(60);
87  background(0);
88  brightDark();
89  triangle(50, 300, 100, 250, 150, 300 ); //Pfeile
90  triangle(displayWidth-50, displayHeight-300,
      displayWidth-100,
      displayHeight-250,displayWidth-150,
      displayHeight-300 );
```

```
91  triangle(displayWidth-50, 300, displayWidth-100,
       250,displayWidth-150, 300 );
92  triangle(50, displayHeight-300, 100,
       displayHeight-250, 150, displayHeight-300 );
93  text("Q",80,355);                          // Beschriftung d.
       Pfeile
94  text("A",80,590);
95  text("O",displayWidth-120,355);
96  text("L",displayWidth-120,590);
97  text("Press␣ENTER␣to␣proceed.",450,800);
98
99
100 if(enter == true && choiceL == true){
101 screenCount = 2;                          //Weiterleitung zu
       Classic durch bestätigung mit ENTER
102 }
103 if(enter == true && choiceR == true){
104 screenCount = 3;                          //Weiterleitung zu
       Fancy durch bestätigung mit ENTER
105 background(255);
106 }
107
108
109 break;
110 }
111
112
113 case 2: {                                      //Entscheidung:
       classic
114 switch(gameCount){
       //Hintergrundfarben in abhängigkeit vom
       Punktestand
115 case 0:
116 background(0,0,0);
117 break;
```

```
118 case 1:
119 background(100,0,0);
120 break;
121 case 2:
122 background(150,0,0);
123 break;
124 case 3:
125 background(200,0,0);
126 break;
127 case 4:
128 background(250,0,0);
129 break;
130 }
131  textSize(32);
132  fill(255);
133 text(ballX+" "+ballY,700,700);
134 text("Game "+(gameCount+1)+" of 5",width/2-100,100); //
      Spielstand
135 text("SCORE:"+ScoreL,50,50);                          //
      Punktestand
136 text("SCORE:"+ScoreR,width-150,50);
137 rect(20,rectPosL,20,200);
      //Rechteck
      Links
138 rect(width-40,rectPosR,20,200);
      //Rechteck
      rechts
139 ellipse(ballX,ballY,20,20);
      //Ball
140 ballX += changeX;
      //Ballanimation
141 ballY += changeY;
142
143
```

```
144 if(tapCount %5 ==0 && firstTap == true){// alle x
      Schlägerberührungen geschwindigkeit erhöhen
145 if(changeX<0){
146 changeX -= 2;
147 }else{
148 changeX += 2;
149 }
150 if(changeY<0){
151 changeY -= 2;
152 }else{
153 changeY += 2;
154 }
155 tapCount++;
156 }
157 if(gameCount == 5){
158 screenCount = 4;
159 }
160 break;
161 }
162
163
164
165 case 3:{                                    //Entscheidung:
      FANCY
166 textSize(32);
167 text("Game␣"+(gameCount+1)+"␣of␣5",width/2-100,100);
168 text("SCORE:"+ScoreL,50,50);
169 text("SCORE:"+ScoreR,width-150,50);
170 rect(20,rectPosL,20,200);                   // Rechteck Links
171 rect(width-40,rectPosR,20,200);             //Rechteck rechts
172 fill(random(255),random(255),random(255));
173 ellipse(ballX,ballY,ballSize,ballSize); //Ball
174 ballX += changeX;                           //Ballanimation
175 ballY += changeY;
176 ballSize = ballSize + ballSizeSpeed;
```

```
177 if (ballSize == ballSizeMax||ballSize == 0){
178 ballSizeSpeed = -ballSizeSpeed;
179 ballSizeMax+=10;
180 }
181
182 if(tapCount %5 ==0 && firstTap == true){ // alle 5
      Schlägerberührungen Geschwindigkeit erhöhen
183 if(changeX<0){
184 changeX -= 2;
185 }else{
186 changeX += 2;
187 }
188 if(changeY<0){
189 changeY -= 2;
190 }else{
191 changeY += 2;
192 }
193 tapCount++;
194 }
195
196 if(gameCount == 5){
197 screenCount = 4;
198 }
199 break;
200 }
201
202
203 case 4:{                                    // Game Over
      Bildschirm
204 background(100,100,200);
205 fill(255);
206 if(ScoreL<ScoreR){
207 text("Right␣Player␣Wins!",width/2+200,height/2);
208 }else{
209 text("Left␣Player␣Wins!",width/4,height/2);
```

```
210 }
211 break;
212 }
213 }
214 }
```

Listing 4.4: Code, um Pong zu starten

## 4.5 Pong Methoden

```
1  ////////////////////////////////////////////Methoden
2  void keyPressed (){      // Steuerung ob eine Tast gedrückt
      ist bzw. losgelassen
3  if (key == 'q') {
4  qPress=true;
5  }
6  if (key == 'a') {
7  aPress=true;
8  }
9  if (key == 'o') {
10 oPress=true;
11 }
12 if (key == 'l') {
13 lPress=true;
14 }
15 if(keyCode == ENTER){
16 enter=true;
17 }
18 }
19 void keyReleased() {
20 if (key == 'q') {
21 qPress=false;
22 }
23 if (key == 'a') {
24 aPress=false;
25 }
26 if (key == 'o') {
27 oPress=false;
28 }
29 if (key == 'l') {
30 lPress=false;
31 }
32 if(keyCode == ENTER){
```

```
33 enter=false;
34 }
35 }
36
37
38 void rectMovement (){          //Pongschläger: Bewegung mit
      q,a,o,l
39 if (qPress == true){
40 rectPosL -= 15;
41 }
42 if (aPress == true){
43 rectPosL += 15;
44 }
45 if (oPress == true){
46 rectPosR -= 15;
47 }
48 if (lPress == true){
49 rectPosR += 15;
50 }
51 }
52
53 void choiceFeedback(){
54 if (keyPressed){
55 if(keyCode == RIGHT){         // Fokusänderung auf Classic
      oder Fancy im Startbildschirm
56 choiceR = true;
57 choiceL = false;
58 }
59 if(keyCode == LEFT){
60 choiceL = true;
61 choiceR = false;
62 }
63
64 }
65 if(choiceR == true){          //Fokus auf Fancy mit animation
```

```
66 fill((int)random(255),(int)random(255),(int)random(255));
67 text("FANCY", width/4*3-150,height/2);
68 delay(100);
69 fill(255);
70 text("CLASSIC", width/4,height/2);
71 }
72
73 if (choiceL == true){          //Fokus auf Classic mit
      animation
74 fill(255);
75
76 text("FANCY", width/4*3-150,height/2);
77 brightDark();
78 text("CLASSIC", width/4,height/2);
79 }
80 }
81
82 /*boolean sketchFullScreen() {
83 return true;
84 }*/
85
86 void brightDark(){    // lässt nachfolgende füllungen heller u.
      wieder dunkeler werden
87 fill(IncDec);
88 IncDec = IncDec+speed;
89 if (IncDec >= 255 || IncDec <= 0){
90 speed =-speed;
91 }
92 }
```

Listing 4.5: Headerfile mit den Methoden für Pong

## 4.6 main Arduino Code

```
1  #include "ArduPar.h"
2  #include "avr/eeprom.h"
3  #include "SoftwareSerial.h"
4  #include "Timer.h"
5  #include "initialization.h"
6  #include "zustandsdiagramm.h"
7  #include "motor.h"
8
9
10 Timer t1;
11
12
13 void setup()
14 {
15
16 initialization(); // siehe initialization.h
17
18
19 }
20 void loop()
21 {
22 Serial.println(1);                          // Anfordern der
      x- und z-Werte von der Kinect Kamera/Processing
23 updateParametersFromStream(&Serial,10);
24 Serial.println(2);
25 updateParametersFromStream(&Serial,10);
26
27 zustandsdiagramm(); // siehe zustandsdiagramm.h
28 //continue to check the Serial input for ßomeIntänd set the value
      if it is received...
29 //Enter i.e. ßomeInt 1100ïnto the serial monitor to set the
      parameter to a new value.
30
```

```
31 }
```

Listing 4.6: main Arduino Code

## 4.7 Initialisation vom Arduino Uno

```
1  #ifndef _INITIALIZATION_H_
2  #define _INITIALIZATION_H_
3
4  #include "ArduPar.h"
5  #include "avr/eeprom.h"
6  #include "SoftwareSerial.h"
7
8
9
10
11 SoftwareSerial mySerial(13, 12);
12
13 IntArduPar someZ;
14 IntArduPar someX;
15
16 void initialization()
17 {
18
19 Serial.begin(57600); //start Serial Communication
20 pinMode(13,OUTPUT);   //some output.
21 //we need to set up the someIntSetting to make it useful:
22 someX.setup(F("someX"), -1000, 1000); //empfängt alle Z
      werte der Kinect(Z ist der Abstand der Kinect zu dem erechneten
      mittelpunkt einer getrackten Person)
23
24
25
26 // The command used to change the parameter. The F("foo") syntax
      saves memory by putting the command into flash-memory. (look up
      "progmem stringsïf you care)
27 // The lowest value the parameter can have. Values received via
      Serial will be clipped to this range.
```

```
28 // The highest value the parameter can have. Values received via
      Serial will be clipped to this range.
29
30
31 someZ.setup(F("someZ"), 0, 5000); // empfängt alle X werte
      der Kinect (X ist Indikator dafür wie weit nach rechts oder
      links eine getrackte Person zur Kinect steht)
32
33
34 digitalWrite(13,LOW);
35
36 // set the data rate for the SoftwareSerial port
37 mySerial.begin(19200);
38 // further Data-Query Commands:
39 mySerial.write(0x83);
40
41 delay(100);
42
43 if(mySerial.read() == 1)
44 {
45
46 Serial.println("Serial is in control!");
47
48 }
49
50 else
51 {
52
53 Serial.println("Serial ist not in control!");
54
55 }
56
57
58 }
59
```

```
60 #endif
```

Listing 4.7: initialisiere Arduino Uno

## 4.8 Motorsteuerung

```
1  #ifndef __MOTOR_H_
2  #define __MOTOR_H_
3
4  #include <ArduPar.h>
5  #include <avr/eeprom.h>
6  #include <SoftwareSerial.h>
7  #include <Timer.h>
8
9  int beschleunigung = 10;
10 int led = 13;
11 boolean schonAngefahren = false;
12 boolean angekommen = false;
13 int geschwL = 0;  //aktueller Geschwindigkeit der Motoren auf
      der Linken Seite
14 int geschwR = 0;  //aktueller Geschwindigkeit der Motoren auf
      der Rechten Seite
15 ////////////////////////////////////////////////////// motor
      //////////////////////////////////////////////////////
16
17
18 void blinken(int wiederholungen, int geschwindigkeit)
      // z.B. für standby o. Kontrolle von werten da zur Laufzeit der
      Port von kinect besetzt ist und damit der Serial Monitor nicht
      benutzbar ist
19 {
20
21 for(int i=0; i < wiederholungen;i++)
22 {
23
24     digitalWrite(led, HIGH); // turn the LED on (HIGH is the
          voltage level)
25     delay(geschwindigkeit);
```

```
26      digitalWrite(led, LOW); // turn the LED off by making the
           voltage LOW
27      delay(geschwindigkeit);
28
29  }
30
31  }
32
33  void vorwaerts(int zielGeschw){ // läst den Roboter vorwärts
       fahren abhängig von der aktuellen Geschwindigkeit.
34  int wdh = abs((zielGeschw - aktGeschw)/10); // Berechnung
       der schleifenwiederholugen
35  for(int i=0; i<wdh; i++){
36  if(aktGeschw < zielGeschw)
37  aktGeschw += 10;
38  else
39  aktGeschw -= 10;
40  mySerial.write(0xC2);
41  mySerial.write(aktGeschw);
42  mySerial.write(0xCA);
43  mySerial.write(aktGeschw);
44  delay(200);                                    // das Delay
       sorgt für eine langsame Beschleunigung wodurch das Wackeln
       veringert wird
45  }
46  }
47
48  void rueckwaerts(int zielGeschw){ // wie vorwaerts ...nur
       rueckwaerts
49  geschwL = 0;
50  geschwR = 0;
51  int wdh = abs((zielGeschw - aktGeschw)/10);
52  for(int i=0; i<wdh; i++){
53  if(aktGeschw < zielGeschw)
54  aktGeschw += 10;
```

```
55 else
56 aktGeschw -= 10;
57 mySerial.write(0xC1);
58 mySerial.write(aktGeschw);
59 mySerial.write(0xC9);
60 mySerial.write(aktGeschw);
61 delay(200);
62 }
63 }
64
65
66 void links(int geschwindigkeit) // fahren einer Linkskurve
67 {
68
69 mySerial.write(0xC2); //forward motor 1
70 mySerial.write(geschwindigkeit); //Speedvalue from 0 to 127
71 mySerial.write(0xCA); //forward motor 2
72 mySerial.write(geschwindigkeit/2); //Speedvalue from 0 to
      127
73
74 }
75
76 void rechts(int geschwindigkeit) / fahren einer
      Rechtsskurve
77 {
78
79 mySerial.write(0xC2); //forward motor 1
80 mySerial.write(geschwindigkeit/2); //Speedvalue from 0 to
      127
81 mySerial.write(0xCA); //forward motor 2
82 mySerial.write(geschwindigkeit); //Speedvalue from 0 to 127
83
84 }
85
86 void drehenL(int geschwindigkeit)
```

```
87 // dreht mit übergebener Geschwindigkeit auf der Stelle nach links
88 {
89
90 mySerial.write(0xC2); //forward motor 1
91 mySerial.write(geschwindigkeit); //Speedvalue from 0 to 127
92 mySerial.write(0xC9); //forward motor 2
93 mySerial.write(geschwindigkeit); //Speedvalue from 0 to 127
94
95 }
96
97 void drehenR(int geschwindigkeit) //dreht mit übergebener
      Geschwindigkeit auf der Stelle nach rechts
98 {
99
100 mySerial.write(0xC1); //forward motor 1
101 mySerial.write(geschwindigkeit); //Speedvalue from 0 to 127
102 mySerial.write(0xCA); //forward motor 2
103 mySerial.write(geschwindigkeit); //Speedvalue from 0 to 127
104
105 }
106
107
108 void drehenLa(int maxi) // dreht mit übergebener
      Geschwindigkeit auf der Stelle nach links...
109 {
110 delay(10);
111 aktGeschw += beschleunigung; // ...allerdings mit
      Beschleunigung...
112 mySerial.write(0xC2);
113 mySerial.write(aktGeschw);
114 mySerial.write(0xC9);
115 mySerial.write(aktGeschw);
116 if(aktGeschw== maxi )
117 beschleunigung= -beschleunigung; // ...und Abbremsen um das
      Wackeln zu veringern
```

```
118 if( aktGeschw == 0){
119 beschleunigung= -beschleunigung;
120 return;
121 }
122
123 }
124
125 void drehenRa(int maxi) // dreht mit übergebener
      Geschwindigkeit auf der Stelle nach rechts...
126 {
127 //delay(10);
128 aktGeschw += beschleunigung; ...allerdings mit
      Beschleunigung...
129 mySerial.write(0xC1);
130 mySerial.write(aktGeschw);
131 mySerial.write(0xCA);
132 mySerial.write(aktGeschw);
133 if(aktGeschw== maxi )
134 beschleunigung= -beschleunigung; ...und Abbremsen um
      das Wackeln zu veringern
135 if( aktGeschw == 0){
136 beschleunigung= -beschleunigung;
137 return;
138 }
139 }
140
141
142 void stoppen()// häöt den Roboter an. 1 da 0 eine Fehlermeldung
      ergibt.
143 {
144 mySerial.write(0xC1);
145 mySerial.write(1);
146 mySerial.write(0xCA);
147 mySerial.write(1);
148 aktGeschw = 0;
```

```
149
150 }
151
152
153 void ausrichten(){   // dreht abhägig vom x-Wert Stückweise
      nach links oder rechts
154 if(someX.value<0)
155 drehenR (65);
156 else drehenL (65);
157 delay(150);
158 stoppen();
159 delay(2000);
160 }
161
162 void bremsen(){ // lässt die Geschwindigkeit des Roboters
      langsam auf 0 sinken
163
164 while(geschwL >= 0 || geschwR >= 0){
165 mySerial.write(0xCA);
166 mySerial.write(geschwL);
167 mySerial.write(0xC2);
168 mySerial.write(geschwR);
169 delay(100);
170 geschwL -= 10;
171 geschwR -= 10;
172 }
173 schonAngefahren = false;
174
175 }
176
177 int normieren(int grundGeschw); // Prototypes
178 void geschwKorrigieren(int &zielGeschwL, int
      &zielGeschwR);
179 void anfahren(int &zielGeschwL,int &zielGeschwR);
180
```

```
181 void zielVerfolgen(float geschwMulti, int drehenMulti){
182     // lässt den Roboter eine von der Kinect erkannte Person mit
        Varibler geschwindigkeit verfolgen
183 someX.valueReceived = false; // Anfordern der x- und z-Werte
184 someZ.valueReceived = false;
185 Serial.println(1);
186 while(someX.valueReceived == false)
187 updateParametersFromStream(&Serial,10);
188 Serial.println(2);
189 while(someZ.valueReceived == false)
190 updateParametersFromStream(&Serial,10);
191 int grundGeschw =
    (int)(someZ.value-1000)/50*geschwMulti; // eine
    Grundgeschwindigkeit wird aus der zu Überwindenden Entfernung
    Berechnet
192 grundGeschw = normieren(grundGeschw);
193 int zielGeschwR = grundGeschw;
194 int zielGeschwL = grundGeschw;
195
196 float winkel = atan((float)
    someX.value/(float)someZ.value); // der zu Überwindende
    Winkel wird berechnet...
197 winkel =(int) (winkel*180 / PI+0.5);
198 int drehung = winkel*drehenMulti;
199 if(drehung > 0)
200 zielGeschwR += drehung;//...und auf der richtigen Seite wird
    die Gescwndigkeit erhöht
201 else
202 zielGeschwL -= drehung;
203
204 if(schonAngefahren == false){
205 schonAngefahren = true;
206 anfahren(zielGeschwL, zielGeschwR);
207 }else{
208 geschwKorrigieren(zielGeschwL, zielGeschwR);
```

```
209 Serial.println(1);
210 while(someX.valueReceived == false)
211 updateParametersFromStream(&Serial,10);
212 Serial.println(2);
213 while(someZ.valueReceived == false)
214 updateParametersFromStream(&Serial,10);
215
216 }
217 }
218
219
220
221 void geschwKorrigieren(int &zielGeschwL, int
      &zielGeschwR){ // Die berechneten Geschwindigkeiten werden
      ohne Verzögerzng and die Motoren übergeben
222 geschwL = zielGeschwL;
223 geschwR = zielGeschwR;
224 mySerial.write(0xCA);
225 mySerial.write(zielGeschwL);
226 mySerial.write(0xC2);
227 mySerial.write(zielGeschwR);
228 }
229 //
230 void anfahren(int &zielGeschwL, int &zielGeschwR)
                  //Die berechneten Geschwindigkeiten werden langsam
      ansteigend an die Motoren übergeben
231 while(geschwL <= zielGeschwL && geschwR <=
      zielGeschwR){
232 if(geschwL <= zielGeschwL){
233 geschwL += beschleunigung;
234 mySerial.write(0xCA);
235 mySerial.write(zielGeschwL);
236 }
237 if(geschwR <= zielGeschwR){
238 geschwR += beschleunigung;
```

```
239 mySerial.write(0xC2);
240 mySerial.write(zielGeschwR);
241 }
242 delay(50);
243 }
244 }
245
246 int normieren(int grundGeschw){
247 if (grundGeschw < 30)       // sorgt dafür, dass der Roboter
       überhaupt fährt...
248 grundGeschw = 30;
249 if (grundGeschw > 70)       //...bzw. einen bestimmten Wert
       nich überschreitet, damit noch Richtungsgeschwindigkeit
       aufgerechnet werden kann
250 grundGeschw = 70;
251 return grundGeschw;
252 }
253
254
255 #endif
256
257 ////////////////////////////////////////////////////// motor
       end ////////////////////////////////////////////////////////
```

Listing 4.8: Motorsteuerung

## 4.9 Code für unser Zustandsdiagramm

```
1  #ifndef __ZUSTANDSDIAGRAMM_H_ // modifiziertes
       zustandsdiagramm.h. erstellt zum seperaten Testen der methode
       drehenPhi()u. drehenb().
2  #define __ZUSTANDSDIAGRAMM_H_
3
4
5  #include <ArduPar.h>
6  #include <avr/eeprom.h>
7  #include "SoftwareSerial.h"
8  #include <Timer.h>
9  #include "motor.h"
10
11 enum zustandsTyp
12 {
13
14 standby,
15 suche,
16 bewegung,
17 aufmerksamkeit,
18 interaktion,
19 pong,
20 quiz,
21 verloren,
22 gewonnen,
23 belohnung
24
25 };
26
27 zustandsTyp aktuellerZustand = standby;
28
29 void zustandsdiagramm()
30 {
31
```

```
32 switch(aktuellerZustand)
33 {
34
35 case standby:
36 blinken(5, 200);
37 aktuellerZustand = suche;
38 break;
39
40 case suche:
41 aktuellerZustand = bewegung;
42 break;
43
44
45
46 case bewegung:
47 if(someZ.value < 900 && someZ.value > 600 ){ //
     Begrenzung für den Bereich in dem der Roboter kurz vor der
     Zielperson ist und Bremsen soll
48  bremsen();
49
50 }else{
51    zielVerfolgen(2.0,3);                          //
        ansonsten Zielperdon verfolgen
52  }
53  if(someZ.value == 0.0 || someX.value == 0.0){// für den
     Fall das keine Person erkannt und deswegen nur null-Werte
     geschickt werden soll der Roboter anhalten
54
55   bremsen();
56  }
57  if(someZ.value < 600){                          // falls
     die Person dem Roboter zu hane kommt soll er langsam rückwärts
     fahren
58  rueckwaerts(30);
59  }
```

```
60                          aktuellerZustand = suche;
61                  }
62
63
64  break;
65
66
67  case aufmerksamkeit:
68
69  break;
70
71
72  case interaktion:
73
74  break;
75
76
77  case pong:
78
79  break;
80
81
82  case quiz:
83
84  break;
85
86
87  case verloren:
88
89  break;
90
91
92  case gewonnen:
93
94  break;
```

```
95
96
97  case belohnung:
98
99  break;
100
101 }
102 }
103
104
105
106
107
108 #endif
```

Listing 4.9: Code für unser Zustandsdiagramm

# Abbildungsverzeichnis

# Quellenverzeichnis


Abbildung 2.1: http://www.mintgruen.tu-berlin.de/robotikWiki/lib/exe/fetch.php?media=projektews2013:moperro:projektdokumentation.pdf

Abbildung 2.2: http://www.pololu.com/product/777

Abbildung 2.11: http://mediabox.grasp.upenn.edu/roswiki/kinect(2f)Tutorials(2f)Adding(20)a(20)Kinect(20)to(20)an(20)iRobot(20)Create.html

Code Abschnitt 4.1 und 4.3: http://www.mintgruen.tu-berlin.de/robotikWiki/doku.php?id=techniken:datenaustausch:ardupar

Code Abschnitt 4.2 und 4.7: http://www.mintgruen.tu-berlin.de/robotikWiki/doku.php?id=techniken:kinect#die_kinect_3d-kamera_in_processing_betreiben

Code Abschnitt 4.8: http://www.mintgruen.tu-berlin.de/robotikWiki/doku.php?id=bauteile:trexdmc