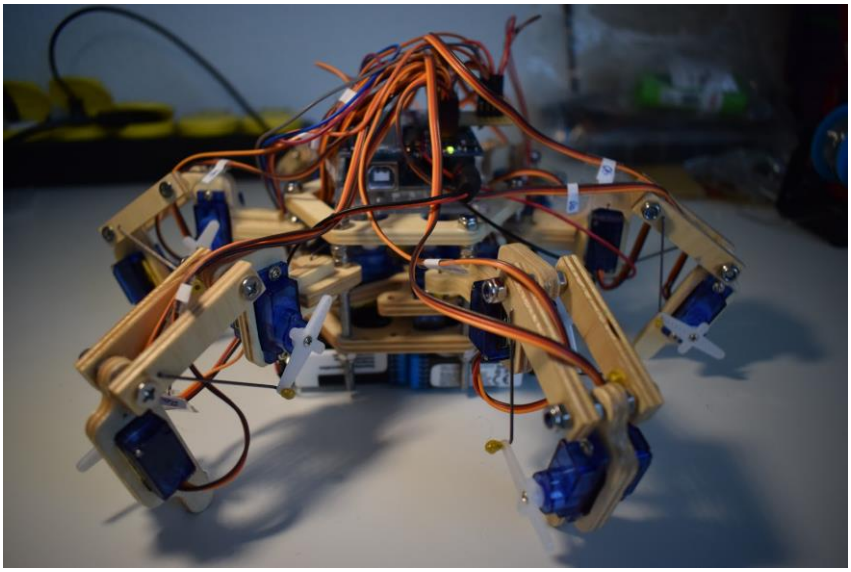


PROJEKT B - HEXAPOD

Dokumentation

Bettina Ganter
Kemal Amet

Prof. Dr. Andreas Plaß
Hochschule für angewandte Wissenschaften
Hamburg, DMI

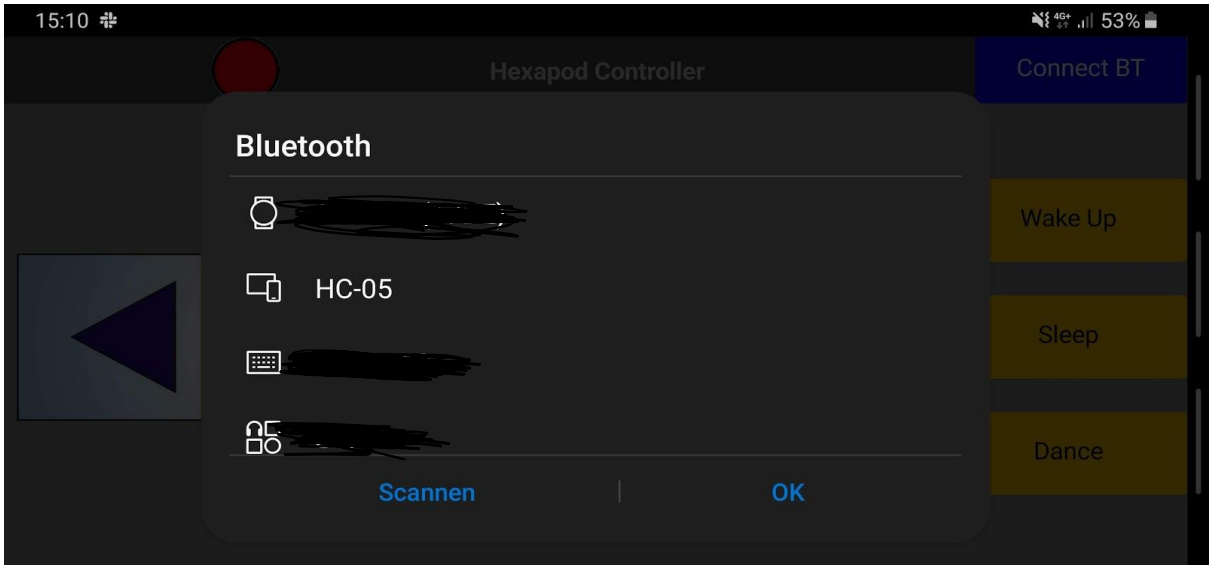
Dokumentation – Hexapod Projekt B

Bettina Ganter – bettina.ganter@haw-hamburg.de – 2257641 (Projektleitung)
Kemal Amet – kemal.amet@haw-hamburg.de – 2269511

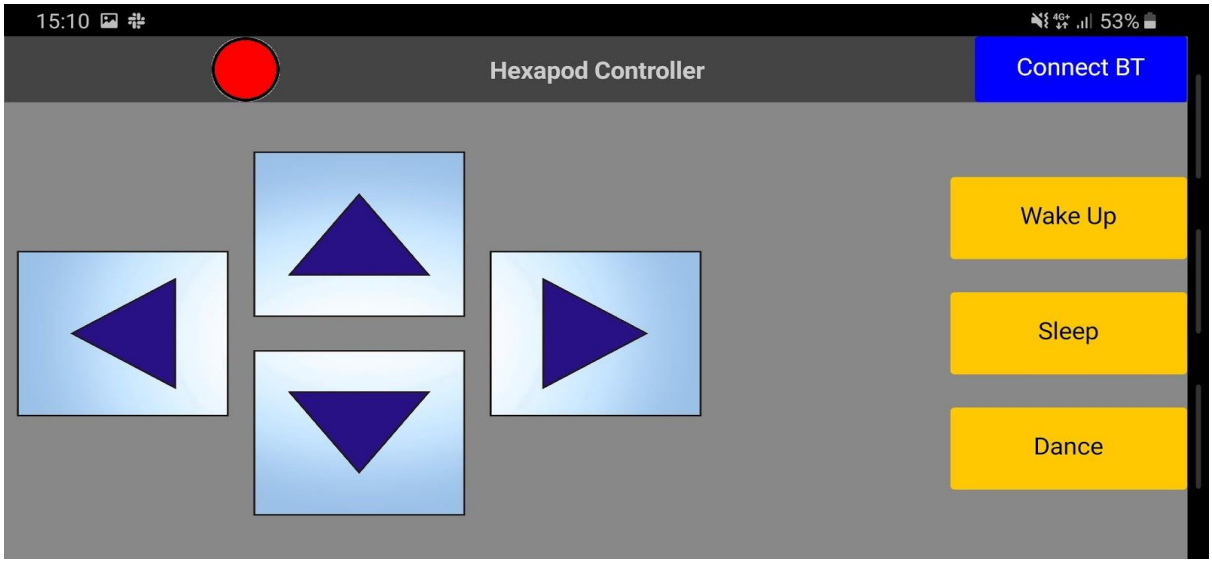
Inhalt

Bedienungsanleitung

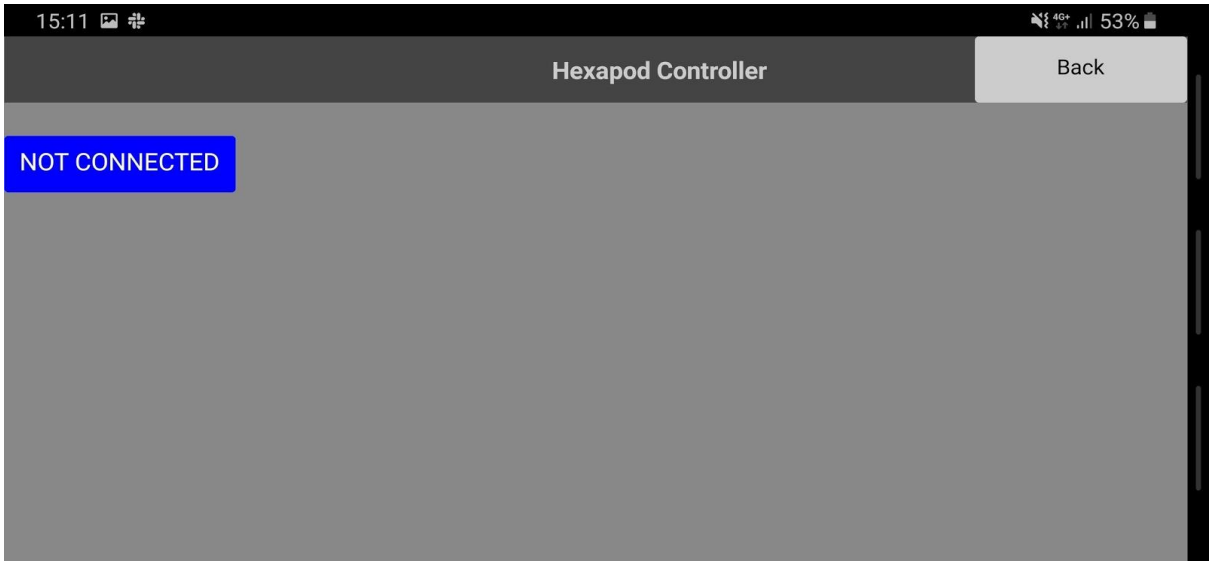
Um den Roboter steuern zu können, benötigt man ein Bluetooth fähiges Android Gerät. Bevor die Controller App geöffnet wird, muss das Bluetooth Modul mit dem Gerät gekoppelt werden. Dafür geht man im Gerät in die Bluetooth Einstellungen und sucht nach dem Bluetooth Modul (HC-05). Die Kopplung ist auch mit einem anderen Bluetooth Modul möglich. Wenn nach einem Passwort gefragt wird ist es Standardmäßig “1234” oder “0000”. Ist die Kopplung erfolgreich, kann man nun die App starten.



Es öffnet sich die Steueroberfläche. Hier sieht man die Richtungen, in die man den Roboter bewegen kann. Auf der rechten Seite sind ein paar vorprogrammierte Bewegungen. Je nachdem in welcher Position sich der Roboter befindet, kann man diesen aufstehen oder hinlegen lassen. Die untere Taste lässt den Roboter für ein paar Sekunden tanzen.



Doch bevor der Roboter gesteuert werden kann, müssen wir uns mit dem Modul verbinden. Dafür sehen wir oben zum einen eine Farbkennung ob wir verbunden sind oder nicht und rechts ist der Button „Connect BT“. Wenn dieser angeklickt wird, gelangen wir in ein anderes Fenster.



Wenn man dann auf „NOT CONNECTED“ klickt, öffnet sich eine Liste mit allen gekoppelten Bluetooth Geräten. Hier wählt man das HC-05 Modul aus. Sollte der Vorgang erfolgreich sein, dann steht auf dem Button „CONNECTED“. Mit „BACK“ kehrt man zurück in das Steuerfenster.

Der zuvor rote Kreis sollte sich nun Blau gefärbt haben. Wenn alles funktioniert hat, dann kann man nun den Roboter steuern!

Entwicklerhandbuch

Einführung - *Ein kurzer Überblick über das Hexapod-Entwicklerhandbuch*

Mit Hilfe dieses Entwicklerhandbuchs soll die Verwendung der von uns erstellten Anwendung möglichst einfach gemacht werden. Es enthält eine Übersicht über die Hardware, eine Installationsanleitung, sowie eine Beschreibung des Codes, der die Laufbewegung des Hexapods kontrolliert.

Hardware Spezifikation

1. **Verwendete Elektronik-Bauteile**

   1x Arduino Uno Rev3

   1x Adafruit 16-Channel 12-bit PWM/Servo Shield

   18x Micro SG90 Servo Motor 9g

   1x HC-05 Bluetooth Module

   2x 9V Batterie

2. **Pin Verteilung**

   Der Adafruit Servo Shield bietet Platz für bis zu 16 Servomotoren. Die Motoren werden wie folgt auf dem Shield platziert:

Steckplatz 0 – 5:	Die fünf Motoren, die jeweils am oberen Ende jedes Beins angebracht sind
Steckplatz 6 – 11:	Die fünf Motoren, die jeweils am unteren Ende jedes Beins angebracht sind
Steckplatz 12 – 15:	Die ersten vier der sechs innenliegenden Motoren

   Die übrigen zwei Motoren müssen direkt auf dem Arduino platziert werden:

Pin 10:	Der fünfte der innenliegenden Motoren
Pin 11:	Der sechste der innenliegenden Motoren

   Diese beiden Motoren müssen zusätzlich noch mit den entsprechenden Kabeln mit den 5V und den GND Anschlüssen des Arduino verbunden werden.

Diese Belegung ist in der Headerdatei „Setup.h“ bzw. in der Haupt-Arduinodatei „Static_Walking_onOrder“ festgelegt.

Außerdem wird noch das HC-05 Bluetooth Modul über die Steckplätze 0 (RX) und 1 (TX) auf dem Arduino angesteckt. Es muss ebenfalls noch an den 5V Pin und GND Pin angeschlossen werden.

Installation

Bevor der Hexapod verwendet werden kann, sind zunächst einige Schritte erforderlich.

1. **Installation der Arduino IDE**

   Zunächst muss die Arduino IDE heruntergeladen und installiert werden. Man findet sie auf der Homepage von Arduino, bzw. hier (https://www.arduino.cc/en/main/software).

   1.1. Herunterladen und Installieren der „Adafruit PWM Servo Driver Library”

   Nach der erfolgreichen Installation muss die „Adafruit PWM Servo Driver Library“ heruntergeladen werden. Diese wird benötigt, damit das auf dem Arduino verbaute Servo Shield betrieben werden kann. Die Installation erfolgt innerhalb der Arduino IDE. Nach öffnen der IDE findet man unter dem Menüpunkt „Sketch“ den Punkt „Bibliothek einbinden“ und hier wiederum den Unterpunkt „Bibliotheken verwalten“.

   Es öffnet sich ein neues Fenster, der „Bibliotheksverwalter“. Hier gibt man in dem Suchfenster „Adafruit PWM Servo Driver Library“ ein. Hat man die entsprechende Bibliothek gefunden, kann sie mit einem Klick auf „Installieren“ heruntergeladen und installiert werden.

2. **Installation der Hexapod.apk**

   Zur Steuerung des Hexapod wird ein Bluetooth fähiges Android-Gerät verwendet. Auf diesem muss die Hexapod.apk installiert sein, die hier (https://drive.google.com/file/d/1bM3yeHMS629l3KmiWP2TF-OLRuqMzT-L/view?usp=sharing) zu finden ist.

3. **Herunterladen des Quellcodes**

   Nun muss noch der Quellcode des Projekts heruntergeladen werden. Er ist in dem zum Projekt gehörenden Github Repository „Hexapod_ProjektB“ zu finden, bzw. hier (https://github.com/BettinaG/Hexapod_ProjektB/tree/master/Code).
   In dem Ordner „ArduinoFiles“ befinden sich die Dateien, die den Code enthalten, der den Hexapod laufen lässt. Diese können genauso wie sie heruntergeladen werden, auch belassen werden. In dem Ordner „HeaderFiles“ befindet sich die Datei „Setup.h“. Sie enthält alle Konstanten und Variablen des Projekts. Außerdem werden hier die benötigten Bibliotheken eingebunden.
   Diese Datei muss in den Ordner verschoben werden, in dem die Arduino IDE nach Bibliotheken sucht. Standardmäßig ist das unter C:\Users\USER\Documents\Arduino\libraries.

4. **Hochladen des Codes auf den Hexapod**

   Nun kann der Code mit einem Klick auf den „Hochladen“-Button innerhalb der Arduino IDE auf den Roboter geladen werden. Dieser ist nun startklar.

Codebeschreibung

Anmerkung: Im Folgenden wird öfters die Rede sein von „dem entsprechenden Motor“ und Ähnlichen. Dabei handelt es sich jeweils um die Nummerierung von 0-5, die die sechs Beine des Roboters kennzeichnen. Welches Bein dabei welcher Nummer entspricht ist den Beschriftungen direkt auf dem Hexapod zu entnehmen.

1. **Headerdatei „Setup.h“**

   In dieser Headerdatei werden alle benötigten Bibliotheken importiert und Konstanten festgelegt. Außerdem werden alle benötigten, globalen, Variablen initialisiert.

   a. Bibliotheken

      Wire.h – Diese Bibliothek ermöglicht einem Arduino, mit Geräten zu kommunizieren, die das I^2C-Protokoll verwenden und wird benötigt, um mit dem verwendeten Adafruit Servo Shield zu kommunizieren.

      Adafruit_PWMServoDriver.h – Hier sind die Servo Shield spezifischen Funktionen enthalten, mit denen in unserem Code die Servomotoren angesprochen und betrieben werden.

      Servo.h – Dies ist eine Arduinoeigene Bibliothek. Sie wird benötigt, um Servomotoren zu steuern, die direkt an dem Arduinoboard angeschlossen werden. Wir benötigen sie trotz des verwendeten Shields, da 2 der 18 benötigten Servos nicht auf das Shield passen und wir sie deshalb direkt über den Arduino ansteuern.

a. Konstanten

„SERVOMIN“, „SERVOMAX“ – sind Werte, die von dem Adafruit Servo Shield benötigt werden. Das Shield arbeitet nicht mit einer Gradzahl, wie es beim direkt an den Arduino angeschlossenen Servomotoren der Fall ist, sondern mit einem sogenannten „Pulse“. Bei der Umwandlung von Grad zu diesem Pulse werden die beiden Werte benötigt. Sie geben die Mindest- bzw. Höchstgrenze für den Bewegungsgrad der Servomotoren an und sollten jeweils auf die verwendeten Servomotoren angepasst werden.

„ELEMENTS“ – Dieser Wert gibt die Anzahl der jeweils in ihrer Funktion gleichartigen Motoren an, sprich Die Anzahl der Motoren die jeweils die Hüfte, den Oberschenkel und den Unterschenkel steuern. Dieser Wert ist immer 6 und wird für viele Schleifen benötigt.

„TOP_MOTORS“, „BOTTOM_MOTORS“, “INNER_MOTORS” – Hier werden die Positionen der Motoren auf dem Adafruit Shield festgelegt, wobei der angegebene Wert jeweils die Nummer des ersten Motors seiner Art (Hüfte, Oberschenkel, Unterschenkel) angibt.

b. Variablen

„pwm“ – Adarfuit_PWMServoDriver – Dieser Typ stammt aus der Adafruit_PWMServoDriver Library. Die Variable wird jedes Mal dann benötigt, wenn einem Motor ein neuer Grad-Wert zugewiesen werden soll. Dafür wird diese Variable mit der Funktion setPWM(channel, 0, pulse) aufgerufen.

„innerServo4/5“ – Servo – Diese beiden Variablen vom arduinoeigenen Typ „Servo“ werden benötigt, um die beiden Motoren zu kontrollieren, die nicht mehr auf das Servo Shield passen. Ihnen werden im späteren Code die Pins zugewiesen, an denen die Motoren angeschlossen werden. Die Grad-Werte können dann jeweils mit der Funktion write(degree) gesetzt werden.

„lastDegBot/Top/In[ELEMENTS]“ – Hier wird der jeweils zuletzt verwendete Grad-Wert des entsprechenden Motors gespeichert.

„degsBot[ELEMENTS]“ – In diese Variablen werden die Grad-Werte geschrieben, die als nächstes erreicht werden sollen.

1. **Hauptdatei „Static_Walking_onOrder.ino“**

   Bei dieser Datei handelt es sich um die Hauptdatei, die die arduinoeigenen setup und loop Funktionen enthält. Hier wird auch die Header-Datei Setup.h eingebunden.

   void setup() – Diese, von Arduino vorgegebene Funktion, wird einmal zu Beginn des Programms aufgerufen. Hier wird als erstes die serielle Kommunikation gestartet. Diese wird in unserem Fall vor allem zum Debuggen verwendet. Danach wird noch die Kommunikation mit dem Adafruit Servo Shield gestartet und deren Frequenz festgelegt, sowie die beiden Variablen, der Motoren, die direkt auf dem Arduino angeschlossen sind, ihren Pins zugewiesen.
   Als letztes wird der Hexapod noch in seine Augsangsposition gebracht. Dafür werden in einer for-Schleife die Variablen lastDeg… und degs… mit Werten gefüllt und die Motoren mit ihren entsprechenden Funktionen in die angegebene Position bewegt.

   void loop() – Diese, ebenfalls von Arduino vorgegebene Funktion, wird nach Ablauf der Setup Funktion immer und immer wieder aufgerufen, bis das Programm beendet wird. Hier wird immer der aktuelle Input abgefragt, der von der mit dem Arduino verbundenen App gesendet wird. Entsprechend dieses Inputs wird dann die jeweilige Bewegungsfunktion aufgerufen.

   int setPulse(int degree) – In dieser kurzen Funktion wurde die von Adafruit stammende Funktion
   int map(degree, DEGREEMIN, DEGREEMAX, SERVOMIN, SERVOMAX)
   ausgelagert, sodass bei der Umwandlung von degree zu pulse nicht jedes Mal all diese Werte angegeben werden müssen, sondern nur noch die gewünschte Gradzahl.

2. **“a_ServoMovement”**

   Dieser Tab enthält ausschließlich die Funktion moveServos(). Sie wird jedes Mal dann aufgerufen, wenn ein neuer Grad-Wert festgelegt wurde, der nun erreicht werden soll. Um zu vermeiden, dass die Motoren ruckartig von einem Extremwert zum anderen springen wurde diese Funktion erstellt. Sie bewegt die Motoren in einer gleichmäßigen Bewegung, Grad für Grad.

   void moveServos() – Als erstes wird eine int8_t Variable für jeden Motor erstellt (diff…[ELEMENTS]). Diese werden dann mit dem Wert -1, 0, oder 1 gefüllt. Der Wert gibt an, ob die aktuelle Gradzahl des Motors kleiner ist als die zu erreichende Gradzahl (-1), größer (1), oder identisch (0).
   Nun wird eine while-Schleife gestartet. Sie iteriert durch alle Motoren durch und nährt dabei die aktuellen Grad-Werte den Zielwerten an. Entsprechend der vorher festgelegten Variablen wird dabei der aktuelle Wert um eins erhöht, um eins verringert, oder gleich gelassen.
   Am Ende dieser while-Schleife wird geprüft, ob einer der Motoren inzwischen seinen Zielwert erreicht hat. Für jeden richtig ausgerichteten Motor wird ein Counter um 1 erhöht.
   Nach der Schleife wird geprüft, ob der Counter der Summe aller Motoren entspricht. Das heißt es wird geprüft, ob alle Motoren ihre Zielausrichtung erreicht haben. Ist dies der Fall wird der Boolean, der für die while-Schleife zuständig ist auf true gesetzt.

3. **“b_DegreeSetting”**

   In diesem Tab werden alle Gradzahlen festgelegt, die die Motoren für die verschiedenen Bewegungen erreichen müssen. Es gibt hier zwei Funktionen für feste Positionen: resetPosition() und goToSleep(), sowie zehn Funktionen, die für die veschiedenen Bewegungsabläufe benötigt werden.
   Außerdem gibt es noch eine Funktion mit der die lastDeg… Variablen schnell gesetzt werden können.
   Jede der Bewegungs-Funktionen ist gleich aufgebaut: Die lastDeg… Variablen werden mit den aktuellen Grad-Werten gefüllt, die degs… Variablen werden mit den neuen Grad-Werten gefüllt und die moveServos()- Funktion wird aufgerufen.

void resetPosition() – Diese Funktion bewegt den Hexapod in seine Ausgangsposition. Er steht mit allen Beinen auf den Boden, die Winkel der Hüfte, bzw. der Beine ist für alle Beine gleich.

void goToSleep() – Mit dieser Funktion wird der Hexapod in seine Transportform bewegt. Der Hexapod zieht die Beine ein, sodass er auf seinem Boden aufliegt.

void moveLegsUp/Down(int odd_even) – Durch diese Funktion hebt der Hexapod drei seiner sechs Beine an/ setzt sie ab. Übermittelt man den int Wert 0 werden alle geraden Beine, sprich 0, 2 und 4 angehoben/ abgesetzt. Übermittelt man den int wert 1 werden alle ungeraden Beine, sprich 1, 3, 5 angehoben/ abgesetzt.

Unter den restlichen acht Bewegungs-Funktionen gibt es für jede Laufrichtung (vorwärts, rückwärts, rechts, links) sowie für die jeweils drei geraden, bzw. drei ungeraden Beine eine Funktion:

- void moveEvenLegsForward()
- void moveOddLegsForward()
- void moveEvenLegsBackward()
- void moveoddLegsBackward()
- void moveEvenLegsRight()
- void moveEvenLegsLeft()
- void moveOddLegsRight()
- void moveEvenLegsRight()

void setLastDegrees() – in dieser Funktion werden alle degs…-Variablen in die jeweiligen lastDeg…-Variablen übertragen. Sprich die gegenwärtigen Gradzahlen der Servos werden in die Variablen für die alten Grad-Werte überschrieben, sodass die Variablen für die aktuellen Gradzahlen wieder frei sind für neue Werte.

4. **“c_WalkingRoutines”**

   Dieser Tab enthält alle Laufroutinen, die dem Hexapod einen flüssigen Bewegungsablauf ermöglichen, sowie eine Routine um den Roboter tanzen zu lassen. Eine Laufroutine setzt sich jeweils aus vier verschiedenen Bewegungs-Funktionen und den beiden „Beinen anheb/absetz“- Funktionen aus 4. „b_DegreeSetting“ zusammen.

   void walkForwards() – Durch diese Routine bewegt sich der Hexapod vorwärts. Vorwärts bedeutet in diesem Fall in die Richtung die mittig zwischen den 0ten und dem 5ten Motoren liegt.

   void walkBackwards() – Durch diese Routine bewegt sich der Hexapod rückwärts. Rückwärts bedeutet in diesem Fall in die Richtung die mittig zwischen den 2ten und 3ten Motoren liegt.

   void walkRight() – Durch diese Routine bewegt sich der Hexapod nach rechts. Rechts liegt hier in Richtung der 1ten Motoren.

   void walkLeft() – Durch diese Routine bewegt sich der Hexapod nach links. Links liegt hier in Richtung der 4ten Motoren.

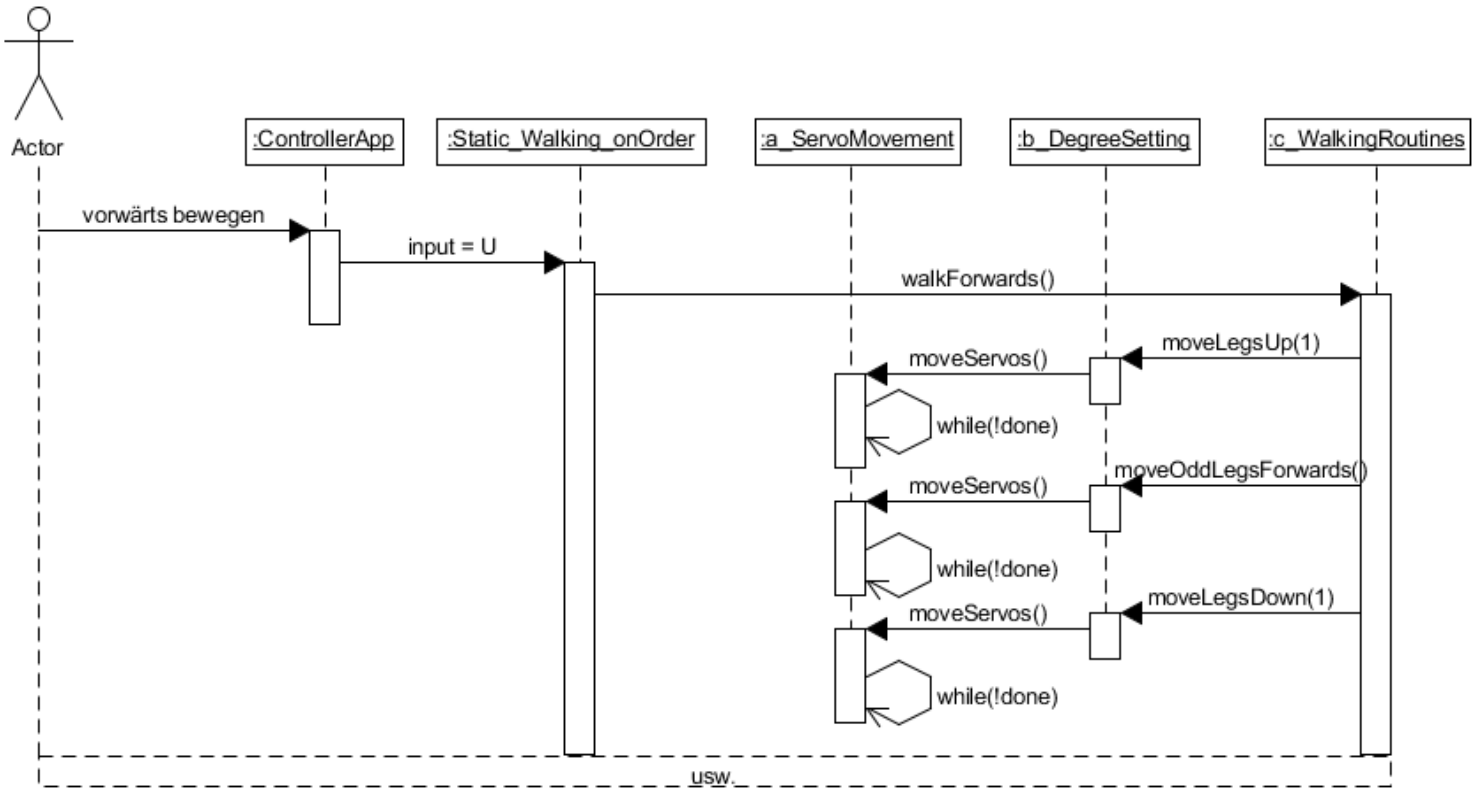
   void dance() – Mit dieser kleinen Routine bewegt sich der Hexapod in einer tanzähnlichen Bewegung hin und her.

Beschreibung der Softwarearchitektur

Arduino

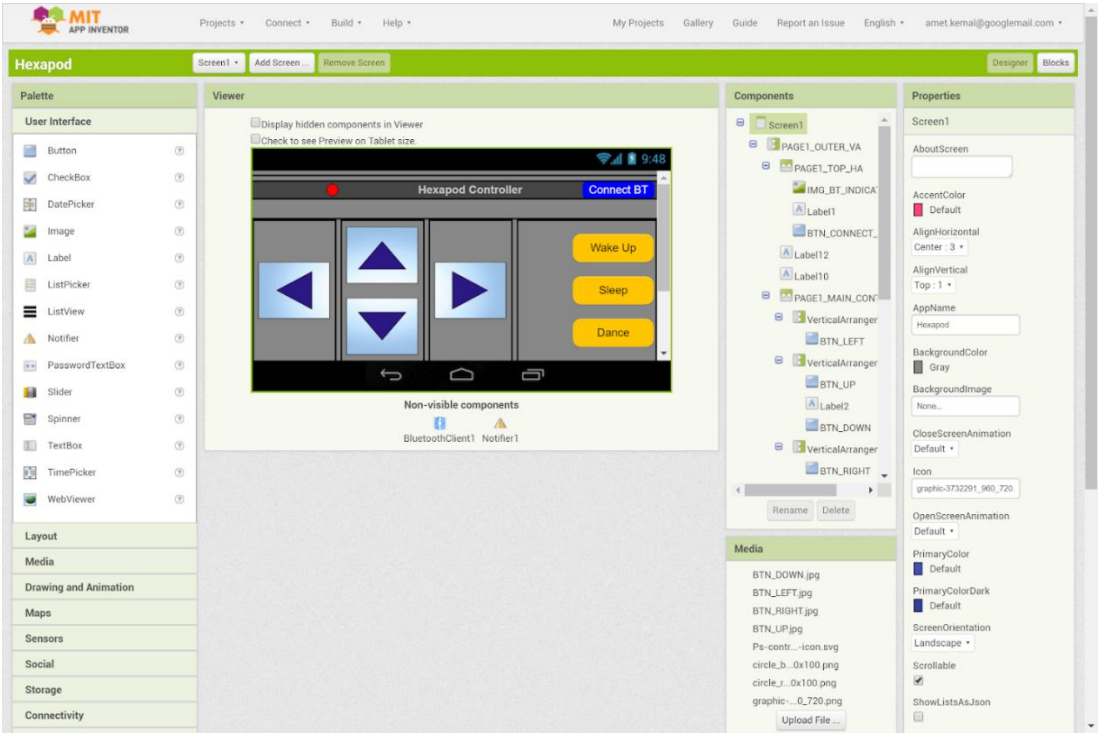
Die Software, die dem Hexapod das Laufen ermöglicht, wurde komplett mit Hilfe der Arduino IDE erstellt. Der Code wurde dabei in der Arduinoeigenen C++ Abwandlung geschrieben. Das bedeutet, dass kein reines C++ verwendet werden muss, sondern auf die vielen Komfortfunktionen von Arduino zurückgegriffen werden kann. Dadurch werden viele Dinge vereinfacht, wie z.B. das Zuweisen eines Pins.

Um den Code möglichst übersichtlich zu halten wurde mit sogenannten „Tabs“ gearbeitet. Damit wird ein Teil des Codes in eine externe Datei ausgelagert und dennoch beim Kompilieren automatisch wieder zu einem gesamten Skript zusammengefügt.



Controller Anwendung

Die App wurde mit dem App Inventor entwickelt, da der Fokus des Projektes beim Bau und Programmierung des Roboters liegt. Der „MIT App Inventor“ bietet eine grafische Entwicklungsumgebung. Mithilfe von Buttons, Layouts und Labels lässt sich die App zusammenbauen.



Wenn das Design fertig ist, muss man sich die Funktionen programmieren. Die Programmierung erfolgt in einem „Block“ Prinzip. Die nötigen Features steckt man sich wie in einem Puzzle zusammen.