Robo4girls

Design Spezifikation



V2.0

2019-03-20

# Einführung und Ziele

## Aufgabenstellung

## Qualitätsziele

## Stakeholder

|  |  |  |
| --- | --- | --- |
| Rolle | Kontakt | Erwartungshaltung |
| *<Rolle-1>* | *<Kontakt-1>* | *<Erwartung-1>* |
| *<Rolle-2>* | *<Kontakt-2>* | *<Erwartung-2>* |

# Randbedingungen

# Kontextabgrenzung

R4G wird für die Verwendung in Zimd „Roberta“ Workshops entwickelt. Das entwickelte Material wird für die Open Source Community zur Verfügung gestellt.

## Fachlicher Kontext

R4G System wird in einem pädagogischen Kontext eingesetzt.

**<Diagramm und/oder Tabelle>**

**<optional: Erläuterung der externen fachlichen Schnittstellen>**

## Technischer Kontext

**<Diagramm oder Tabelle>**

**<optional: Erläuterung der externen technischen Schnittstellen>**

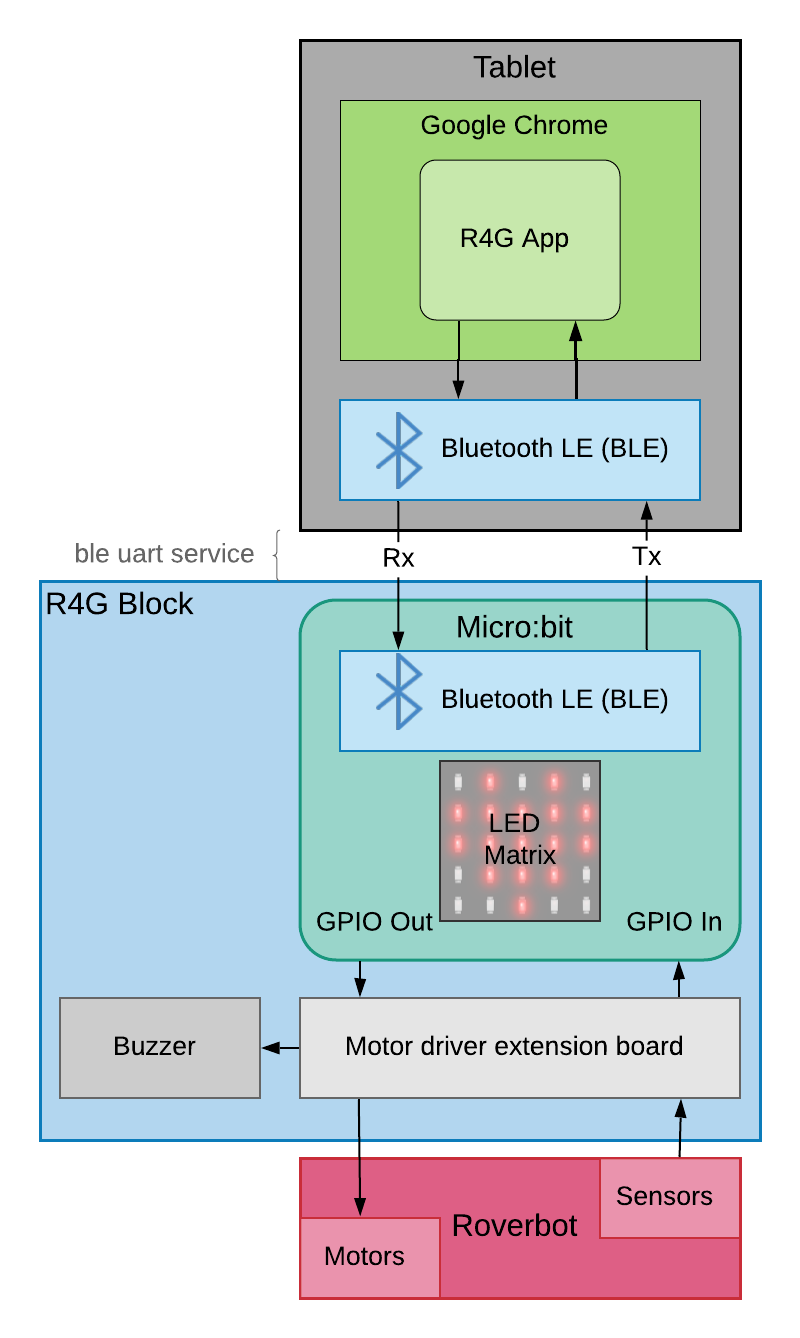
**<Mapping fachliche auf technische Schnittstellen>**

# Lösungsstrategie

Die Komponenten des geplanten Robo4girls Systems sind:

* **“R4G App“:** Eine browserbasierte Anwendung
* **“R4G Block“:** Eine kompakte Steuereinheit mit internen Aktoren und Sensoren.
* **Roverbot:** LEGO Mindstorms Robotics Interventions System (RIS) Roverbot Gestell, Sensoren und Motoren.

Begründung

*Die von Zimd bereits erworbenen Tablets (Huawei MediaPad M3 Lite 10) sollen in den Workshops verwendet werden. Um Plattformunabhängigkeit zu gewähren und laufende Software-Updates zu vermeiden wurde entschieden, dass die R4G App als Browserbasierte Anwendung umgesetzt wird. Die Steuerung der Hardware wird in einem kompakten* ***R4G Block*** *zusammengefasst. Dieser besteht aus einer Steuereinheit (Micro:bit Prozessor) und Schnittstellen zur Hardware (Micro:bit Aktoren und Sensoren, GPIOs, Erweiterungsmodule). Die einzelnen Teile werden in einem Gehäuse verbaut, das Anbaumöglichkeit und Schnittstellen (Stecker, LEGO Noppen) zur LEGO RIS Hardware bietet, die in den Workshops verwendet wird.   
Der R4G Block kann über Bluetooth 4.0 gesteuert werden. Hierfür wird ein uart-über-Bluetooth Profil genutzt. Dieses emuliert eine serielle UART (Universal Asynchronous Receiver Transmitter) Kommunikation über Bluetooth.*

# Bausteinsicht:

## Whitebox Gesamtsystem

Enthaltene Bausteine

### R4G App

R4G App UI

Blockly, Programmiertool (Blockly Workspace)

R4G App Backend

Bluetooth Web Api

Programmablauf

### Micro:bit Programm zur Steuerung der Hardware

Der Micro:bit ist ein eingebettetes System (embedded System), das 2012 von der BBC entwickelt wurde. Er wurde speziell für Bildungszwecke im Bereich der Informationstechnik und für die Anwendung in Schulen entwickelt. Das Herzstück des Micro:bit ist ein Microcontroller vom Typ ARM Cortex-M0 (nRF51822) mit 256KB Flash-Speicher und 16KB statischem RAM.  
Der Micro:bit hat eine 5 x 5 (roten) LED Matrix, 2 Taster (A,B), einen Reset Taster auf der Rückseite und weitere 19 General Purpose Inputs/Outputs (GPIOs)([Micro:bit Pinout](https://microbit.org/de/guide/hardware/pins/)). Weiters hat der Micro:bit bereits Sensoren auf dem Board integriert (Kompass, Accelerometer) und kann über Radio oder Bluetooth 4.0 kabellos angesprochen werden.

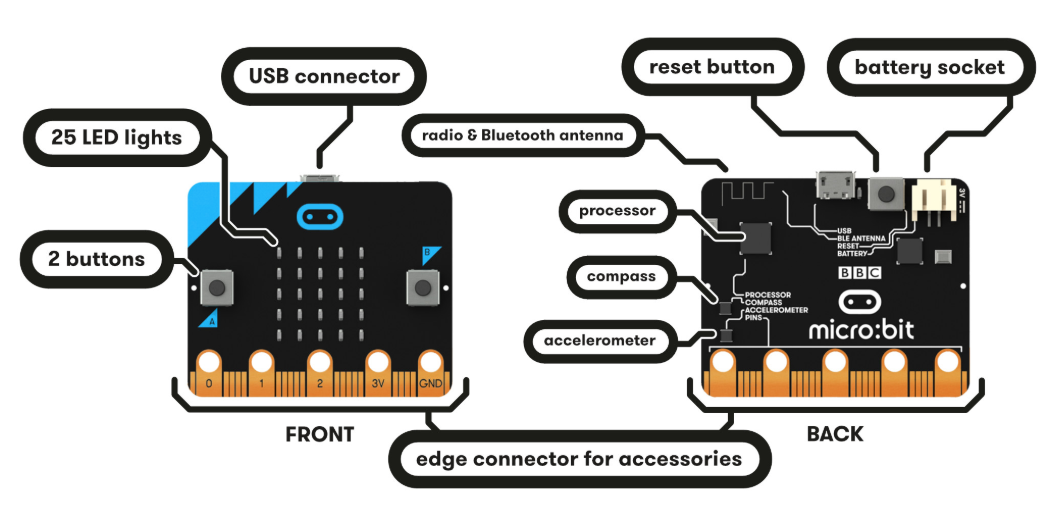


Abbildung 1: Microbit Features (Quelle:https://microbit.org/de/guide/features/)

Der Micro:bit kann mit Blocks, Python, JavaScript oder auch C/C++ programmiert werden. Für das R4G Micro:bit Programm wurde C/C++ verwendet, da nur diese Variante die volle Kontrolle über alle Komponenten des Micro:bits ermöglicht (Anm: besonders die genaue Kontrolle des Bluetooth Low Energy Profils).  
Zur Einstellung und Programmierung des Micro:bits wurde ein C++ Programm in der Online Programmierumgebung Mbed erstellt. Dieses kann mit Mbed zu einem hex file kompiliert werden. Das erstellte hex File kann über eine USB Schnittstelle auf den Micro:bit *geflasht* werden.

Wichtige Schnittstellen

*Bluetooth 4.0 (Bluetooth Low Energy - BLE)*

GPIOs

### Tablet

*<Zweck/Verantwortung>*

*<Schnittstelle(n)>*

*<(Optional) Qualitäts-/Leistungsmerkmale>*

*<(Optional) Ablageort/Datei(en)>*

*<(Optional) Erfüllte Anforderungen>*

*<(optional) Offene Punkte/Probleme/Risiken>*

### Browser (Google Chrome)

*<Blackbox-Template>*

### Bluetooth LE Web API

## Design R4G App

### Nutzeroberfläche (User Interface (UI))

Die Nutzerinnen erstellen ein Programm zur Steuerung der R4G Roberta. Dieses kann in einem Blockly Workspace aus vorgefertigten R4G Programmblöcken zusammengesetzt werden und über einen Start–Button gestartet werden. Die zur Programmierung verwendbaren Blöcke sind in der sogenannten Toolbox zusammengefasst und sind dort der Übersichtlichkeit wegen in Kategorien eingeteilt. Für die R4G Programmierung wurden sechs verschiedene Kategorien, nämlich Bewegungsblöcke, Kombinationsblöcke und Melodien, Einstellungen, LED Anzeige und Wiederholungen geplant. Tabelle 1 zeigt das Design der jeweiligen, für die R4G Toolbox erstellten Programmierblöcke.

# R4G Blockly Blöcke:

Tabelle 1: R4G Blockly Blöcke

|  |  |  |
| --- | --- | --- |
| Design | Beschreibung | Link |
| Bewegungsblöcke | | |
|  | Bewegungsblock mit Eingabe der Dauer | [Bewegungsblock (Vorwärts)](https://blockly-demo.appspot.com/static/demos/blockfactory/index.html#xv2zbt) |
|  | Bewegungsblock mit Eingabe der Dauer | [Bewegungsblock (Rückwärts)](https://blockly-demo.appspot.com/static/demos/blockfactory/index.html#xf4grh) |
|  | Bewegungsblock mit Eingabe der Dauer | [Bewegungsblock (Links drehen)](https://blockly-demo.appspot.com/static/demos/blockfactory/index.html#2g3h6f) |
|  | Bewegungsblock mit Eingabe der Dauer | [Bewegungsblock (Rechts drehen)](https://blockly-demo.appspot.com/static/demos/blockfactory/index.html#qgqs5i) |
|  | Bewegungsblock mit Eingabe der Dauer | [Bewegungsblock (Linkskehre)](https://blockly-demo.appspot.com/static/demos/blockfactory/index.html#ofugw8) |
|  | Bewegungsblock mit Eingabe der Dauer | [Bewegungsblock (Rechtskehre)](https://blockly-demo.appspot.com/static/demos/blockfactory/index.html#ttnvhb) |
| Kombinationsblöcke | | |
|  | Tanzen mit Auswahl der Intensität und Eingabe der Wiederholungen | [Kombinationsblock (Tanzen)](https://blockly-demo.appspot.com/static/demos/blockfactory/index.html#tvm4by) |
|  | Zickzack mit Auswahl der Intensität und Eingabe der Wiederholungen | [Kombinationsblock (Zickzack)](https://blockly-demo.appspot.com/static/demos/blockfactory/index.html#q7rnrx) |
|  | Schütteln mit Auswahl der Intensität und Eingabe der Wiederholungen | [Kombinationsblock (Schütteln)](https://blockly-demo.appspot.com/static/demos/blockfactory/index.html#77ndmh) |
|  | Pirouette mit Auswahl der Drehrichtung und Eingabe der Wiederholungen | [Kombinationsblock (Pirouette)](https://blockly-demo.appspot.com/static/demos/blockfactory/index.html#2imkdz) |
| Melodien | | |
|  | Melodieblock mit Melodie-Auswahl | [Melodie](https://blockly-demo.appspot.com/static/demos/blockfactory/index.html#5zmrse) |
| Einstellungen | | |
|  | Motor Block mit Motor- Auswahl und Eingabe der Leitung | [Motorleistung anpassen](https://blockly-demo.appspot.com/static/demos/blockfactory/index.html#6rdmxe) |
| LED Blöcke | | |
|  | Laufschrift für LED Matrix mit Texteingabe | [Schreibe Text](https://blockly-demo.appspot.com/static/demos/blockfactory/index.html#oo36sw) |
|  | Anzeige eines Bildes mit Auswahlmenu und Nummerneingabe | [Zeige Bild](https://blockly-demo.appspot.com/static/demos/blockfactory/index.html#kbbzas) |
| Wiederholungen | | |
|  | Wiederholungs Block mit Eingabe der Wiederholungen | [Wiederholung](https://blockly-demo.appspot.com/static/demos/blockfactory/index.html#z3ttaw) |

Alle R4G Blöcke sind zusammengefasst in diesem [workspace](https://blockly-demo.appspot.com/static/demos/blockfactory/index.html).

# Komandos aus der App an den Micro:bit

In der App werden mit Blöcken aus der Kategorien Programme erstellt, die als Strings (Zeichenketten) an den Micro:bit über Bluetooth gesendet werden. Diese Zeichenketten können von Micro:bit als ausfühbare Kommandos verstanden werden.

Ein Komando wird in folgender Form abgesetzt:

**Bv2.0:**

Jedes Kommando besteht aus Kategorie (B für Bewegung), Befehl (v für Vorwärts) und den Parametern (Dauer: 2.0 Sek.). Jedes Kommando muss mit einen Doppelpunkt abgeschlossen werden, da im Micro:bit die Kommandos bis zum Doppelpunkt gelesen werden.

Wenn der Micro:bit den Befehl ausgeführt hat, schickt er ein “OK“ zurück.

## Kategorie: Bewegung (B)

Hier sind einfache Strings gemeint, wie Vorwärts, Rückwärts, Links bzw. Rechts drehen und Links- bzw. Rechtskehre. Wie vorher gesagt, werden um den Micro:bit zu programmieren Buchstaben verwendet, **B** wird für die Kategorie **Bewegung** verwendet. Die Bewegungsbefehle sind **Vorwärts (v), Rückwärts (z), Linkskurve (l), Rechtskurve (r), Linkskehre (L), Rechtskehre (R) und Stopp (s)**. Als **Parameter** muss **die Dauer** in Sekunden und Zehntel-Sekunden angegeben werden. Möglich sind Eingaben von **0.0 Sek bis 9.9 Sek.**

## Kategorie: Kombinierte Bewegung (K)

Kombinierte Bewegungen werden aus Bewegungsbefehlen (B) zusammengesetzt. Wie die vorherige Kategorie, wird auch diese Kategorie mit einem Buchstaben gekennzeichnet: **K** fürKombinationen**.** Die Parameter hier sind **die Wiederholung (Wh) (**1-9 mal) und **die Intensität (I) (sanft, mittel oder stark).** Bei höherer Intesität werden die Bewegungen, vor allem die Geradeaus-Befehle, zeitlich länger ausgeführt.

Die Befehle dieser Kategorie sind: **Tanzen (T), Zickzack (Z), Schütteln (S)** und **Pirouette (P)**. Diese Befehle werden z.B. wie folgt zusammengesetzt:

* Zickzack (Parameter: Wh, I)
  + Linkskehre für halbe Sekunde (**BL0.5:**)
  + Vorwärts für eine Sekunde (**Bv1.0:**)
  + Rechtskehre für ganze Sekunde (**BR1.0:**)
  + Vorwärts für eine Sekunde (**Bv1.0:**)
  + Linkskehre für halbe Sekunde (**BL0.5:**)
* Schütteln (Parameter: Wh, I)
  + Linkskehre für halbe Sekunde (**BL0.5:**)
  + Rechtskehre für ganze Sekunde (**BR1.0:**)
  + Linkskehre für ganze Sekunde (**BL1.0:**)
  + Rechtskehre für ganze Sekunde (**BR1.0:**)
  + Linkskehre für halbe Sekunde (**BL0.5:**)
* Pirouette (Parameter: Wh, Drehrichtung,)
  + Leistung erhöhen (**Gb1024:)**
  + ca. 360° Drehung (**BL4.5:/BR4.5:)**
  + Leistung verringern auf normal (**Gb800:)**
* Tanzen: (Parameter: Wh, I)
  + Vorwärts für halbe Sekunde (**Bv0.5:**)
  + Rückwärts für halbe Sekunde (**Bz0.5:**)
  + Schütteln, (**KS2.0:**)

## Kategorie: Melodie (M)

Die dritte Kategorie ist Melodie (M). Man kann eine Melodie-ID aus dem Songbook auswählen. Melodie-ID wäre eine ganze Zahl von 1 bis X, bei X Liedern im Songbook. (z.B. **M3:** Lied 3 abspielen)

## Kategorie: LED Anzeige (A)

Über diese Kategorie kann man Bilder auf der 5x5 LED Matrix des Micro:bit anzeigen lassen oder Text über die Matrix scrollen lassen.

## Kategorie: Motorleistung (G)

Hiermit kann man die Leistung der zwei Motoren anpassen. Man kann den Motor und die Leistung, als eine ganze Zahl, auswählen (Motor: 1, 2 oder b (beide) und Leistung zwischen 0 und 1024).

Die Befehle, die der Microbit versteht, sind in Tabelle 2 zusammengefasst.

Tabelle 2: Übersicht der Micro:bit Kommandos

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Kommandos zur Kommunikation mit dem Micro:bit** | | | | | | |
| **Kategorie** | **Befehl** | | **Parameter** | | | |
| Bewegung |  | Richtungen | Dauer min | Dauer max | Einheit | |
| B | v  z  l  r  L  R | Vor  Zurück  Links  Rechts  Linkskehre  Rechtskehre | 0.0 | 9.9 | Sekunden | |
| Kombinationen |  | Befehl | Intensität | | Wiederholungen | |
| K | T  Z  S  P | Tanz  Zickzack  Schütteln  Pirouette | 1  2  3 | sanft  mittel  stark | Min  1 | Max  9 |
| Richtung | |
|  | | L  R | Links  Rechts |
| Melodien | Melodie-Auswahl | | Keine weiteren Parameter.  Melodie Id entspricht Liednummer in Songbook: 1..X bei X Liedern im Songbook | | | |
| M | Melodie Id | |
| LED Anzeige | Befehl | | Dauer min | Dauer max | Einheit | |
| A | B | Zeige Bild | 1 | 9 | Sekunden | |
|  | T | Scrolltext | User Eingabe: “Textstring“ | | | |
| Einstellungen |  |  | Motor | | Frequenz (PWM) | |
| G |  | Motorleistung | 1  2  b | Motor 1  Motor 2  beide | Zahleneingabe zwischen 0 und 1024 | |
| **Beispiel für ein Bewegungskommando:  Bv2.0:  Jedes Kommando wird mit einem Doppelpunkt abgeschlossen!** | | | | | | |

### Whitebox *<Baustein 1>*

*<Whitebox-Template>*

### Whitebox *<Baustein 2>*

*<Whitebox-Template>*

…

### Whitebox *<Baustein m>*

*<Whitebox-Template>*

## Micro:bit

### Whitebox <\_Baustein x.1\_>

*<Whitebox-Template>*

### Whitebox <\_Baustein x.2\_>

*<Whitebox-Template>*

### Whitebox <\_Baustein y.1\_>

*<Whitebox-Template>*

# 

# Laufzeitsicht

## *<Bezeichnung Laufzeitszenario 1>*

<hier Laufzeitdiagramm oder Ablaufbeschreibung einfügen>

<hier Besonderheiten bei dem Zusammenspiel der Bausteine in diesem Szenario erläutern>

## *<Bezeichnung Laufzeitszenario 2>*

…

## *<Bezeichnung Laufzeitszenario n>*

…

# Verteilungssicht

## Infrastruktur Ebene 1

***<Übersichtsdiagramm>***

Begründung

*<Erläuternder Text>*

Qualitäts- und/oder Leistungsmerkmale

*<Erläuternder Text>*

Zuordnung von Bausteinen zu Infrastruktur

*<Beschreibung der Zuordnung>*

## Infrastruktur Ebene 2

### *<Infrastrukturelement 1>*

*<Diagramm + Erläuterungen>*

### *<Infrastrukturelement 2>*

*<Diagramm + Erläuterungen>*

…

### *<Infrastrukturelement n>*

*<Diagramm + Erläuterungen>*

# Querschnittliche Konzepte

## *<Konzept 1>*

*<Erklärung>*

## *<Konzept 2>*

*<Erklärung>*

…

## *<Konzept n>*

*<Erklärung>*

# Entwurfsentscheidungen

# Qualitätsanforderungen

## Qualitätsbaum

## Qualitätsszenarien

# Risiken und technische Schulden

# Glossar

|  |  |
| --- | --- |
| Begriff | Definition |
| *<Begriff-1>* | *<Definition-1>* |
| *<Begriff-2* | *<Definition-2>* |

**Über arc42**

arc42, das Template zur Dokumentation von Software- und Systemarchitekturen.

Erstellt von Dr. Gernot Starke, Dr. Peter Hruschka und Mitwirkenden.

Template Revision: 7.0 DE (asciidoc-based), January 2017

© We acknowledge that this document uses material from the arc42 architecture template, <http://www.arc42.de>. Created by Dr. Peter Hruschka & Dr. Gernot Starke.