Jugend forscht

NXT Android Control

Internetbasierte Steuerung eines mit einem Smartphone ausgestatteten Legoroboters

Alexander Puck Neuwirth
05.03.2015

Kurzfassung

Der Legoroboter verfügt standardmäßig über Bluetooth- und USB-Kommunikation, deshalb ist eine Steuerung nur innerhalb begrenzter Reichweiten (ca. 5 m) möglich. Diese Beschränkung der Reichweite soll mithilfe einer Internetanbindung aufgehoben werden.

Eine Steuerung über große Reichweiten erfordert eine optische Kontrolle. Diese soll durch den Einsatz einer Kamera realisiert werden.

Ziel des Projektes ist die Ansteuerung eines LEGO® MINDSTORMS® NXT Roboters mittels Bluetooth und Internet sowie die Erweiterung der NXT Funktionalitäten um Kameraübertragung, Soundübertragung und GPS-Sensoren unter Verwendung eines Smartphones.

1. Ansteuerung des Roboters über Bluetooth

Die Programmierung erfolgt in Java. Hierzu muss die Standardfirmware des NXT durch LeJOS ersetzt werden. Zur Ansteuerung wird ein Android Smartphone verwendet.

2. Erweiterung der Funktionalitäten, WLAN-Anbindung, Steuerung über Computer

Die Kommunikation zwischen Mensch Computer–Smartphone–NXT–NXTUmfeld erfolgt in einem LAN-Netzwerk.

3. Erweiterung der Funktionalität durch einen zweiten NXT-Block

Ein NXT-Roboter ist mit drei Motoren ausgestattet. Durch die Erweiterung der Ansteuerung um einen zweiten NXT-Block, also um drei weitere Motoren, lässt sich das Smartphone um eine weitere Achse schwenken (oben/unten) und ein Greifarm könnte angesteuert werden.

4. Steuerung über mobile Endgeräte

In der Arbeit wurden die Punkte 1 und 2 implementiert. Die Punkte 3 und 4 sind grundsätzlich aufgrund der Überlegungen in der Ausarbeitung realisierbar. Die erfolgreiche Umsetzung hat gezeigt, dass durch Einbindung einer neuen Komponente (Smartphone) und die Standardisierung durch Programmierung in Java die Möglichkeiten des NXT-Roboters erheblich erweitert werden können.

5. Internetanbindung

Die bisherige Steuerung des Roboters mittels WLAN (begrenzte Reichweite)soll durch WWAN ersetzt werden (weltweite Reichweite). Sowohl der Computer als auch das Smartphone benötigen hierfür einen Internetzugang.

1 Inhalt

1 Einleitung 1
1.1 Erste Phase: Bluetooth und WLAN Kommunikation 1
1.2 Zweite Phase: Erweiterung der Funktionalitäten 2
1.3 Dritte Phase: Erweiterung der Funktionalitäten durch einen zweiten NXT-Block 3
1.4 Vierte Phase: Steuerung über mobile Endgeräte 3
1.5 Fünfte Phase: Internetanbindung 3
2 Vorüberlegungen 4
2.1 Kommunikation 4
2.2 Graphische Benutzeroberfläche 4
2.3 Kameraübertragung 4
2.4 Soundübertragung 4
3 Lego-Roboter 5
4 Modellierung 7
4.1 Verbindungsaufbau 7
4.2 Kommunikation 8
5 Implementierung 10
5.1 Kommunikation 10
5.2 Steuerung des Roboters 10
5.3 GPS und Rotation 12
5.4 Kameraübertragung 12
5.5 Sound und Sprachübertragung 13
5.6 Erweiterung der Funktionalität durch einen zweiten NXT-Block 13
6 Probleme und Lösungen 14
6.1 Kamerazugriffseinschränkung 14
6.2 Steuerungsschwierigkeiten 14
6.3 Verschiedene Grundlagen 15
6.4 Zufällige Verbindungsunterbrechungen 15

7 Ausblick 15
7.1 Mobile Endgeräte 15
7.2 WWAN 15
8 Zusammenfassung 16
9 Literaturverzeichnis 17

1 Einleitung

Ziel des Projektes ist die Ablösung der reichweitenbegrenzten Steuerung eines LEGO® MINDSTORMS® NXT 2.0 durch eine internetbasierte Steuerung (weltweite Reichweite) sowie die Erweiterung der NXT Funktionalitäten um Kameraübertragung, Soundübertragung und GPS-Sensoren unter Verwendung eines Smartphones.

Da der Legoroboter standardmäßig über Bluetooth- und USB-Kommunikation verfügt, ist eine Steuerung nur innerhalb begrenzter Reichweiten (ca. 5 m) möglich. Diese Beschränkung der Reichweite soll mithilfe einer Internetanbindung aufgehoben werden. Hierfür bietet sich ein Smartphone mit Bluetooth und WWAN (Wireless Wide Area Network) Schnittstelle an, da es nahezu weltweit erreichbar ist.

Verwendete Materialien:

- Laptop mit Windows 7
- Smartphone mit Android 4.2.1
- LEGO® MINDSTORMS® NXT 2.0 mit leJOS NXJ 0.9.1 beta

1.1 Erste Phase: Bluetooth und WLAN Kommunikation

In der ersten Phase erfolgt die Kommunikation zwischen Computer – Smartphone – NXT in einem LAN-Netzwerk. Hierbei wird durch das Smartphone ein kabelloses Netzwerk erstellt (Tether) und ein WLAN-fähiger Computer verbindet sich mit diesem. Außerdem verbindet sich das Smartphone über Bluetooth mit dem NXT.

Befehle und Daten können nun interaktiv vom Computer an das Smartphone gesendet werden, welche dieses, entweder selbst verarbeitet oder falls nötig, zur weiteren Verarbeitung an den Legoroboter weiterleitet. Dieser kann auf umgekehrtem Wege über das Smartphone mit dem Computer kommunizieren.

Um betriebssystembasierte Fehler weitestgehend zu vermeiden, wird auf allen Geräten (Computer, Smartphone, NXT) in Java programmiert. Dafür muss die Standardfirmware des NXT durch leJOS ersetzt werden.

1.2 Zweite Phase: Erweiterung der Funktionalitäten

Zur Steuerung des NXT werden mehrere Funktionen implementiert. Auch die Kommunikation wird erweitert, sodass sie zwischen Mensch (Controller) – Computer – Smartphone – NXT – NXT-Umfeld stattfindet. Dabei gibt es sowohl Funktionen, die Befehle an den NXT als auch Funktionen, die interaktiv Daten vom NXT an den Computer liefern:

1. **Motorsteuerung:** Controller→ Computer→ Smartphone→ NXT
   Ermöglicht sowohl vorwärts als auch rückwärts geradlinige und kurvenförmige Bewegungen des Roboters durch zwei seiner Motoren. Der dritte Motor wird zum Drehen des Smartphones verwendet und ermöglicht so einen 360°-Rundblick für die Kamera.
2. **Lichtsteuerung:** Controller→ Computer→ Smartphone→ NXT
   Um auch bei Dunkelheit gesteuert zu werden, kann sowohl ein LED-Licht am NXT als auch das Kamerablitzlicht des Smartphones an- und ausgeschaltet werden.
3. **Sonar:** NXT-Umfeld→ NXT→ Smartphone→ Computer→ Controller
   Der im LEGO MINDSTORMS Set enthaltene Sonarsensor liefert Daten zur Distanzmessung.
4. **Orientierungssystem:** NXT→ Smartphone→ Computer → Controller
   Der Roboter gibt die Rotation einzelner Motoren mit einer Präzision von 1° aus. Zusammen mit den im Smartphone enthaltenen Sensoren (GPS, (Erd)-Magnetfeld- und Gravitationssensor) lässt sich die Position sowie die Orientierung des Roboters bestimmen und auf einer OpenStreetMap anzeigen.
5. **Kameraübertragung:** NXT-Umfeld→ Smartphone → Computer→ Controller
   Das Smartphone schickt kontinuierlich Kamerabilder, die von dem Computer dann ausgegeben werden sollen. Dies sollte möglichst in Echtzeit erfolgen, um eine direkte Steuerung des Roboters zu gewährleisten.
6. **Soundübertragung:** NXT-Umfeld→ Smartphone → Computer→ Controller
   Das Smartphone ermöglicht es Geräusche aus der Umgebung zu übertragen.
7. **Sprachausgabe:** Controller→ Computer → Smartphone → NXT-Umfeld
   Das Smartphone verfügt über ein TTS-System(Text-to-Speech). So kann vom Computer ein Text geschickt werden, der dann vom Smartphone als Sprachnachricht ausgegeben wird.

1.3 Dritte Phase: Erweiterung der Funktionalitäten durch einen zweiten NXT-Block

Ein NXT-Roboter ist mit drei Motoren ausgestattet (siehe 1.2 Zweite Phase: Erweiterung der Funktionalitäten). D.h. der Roboter kann das Smartphone lediglich um die eine Achse drehen (links/rechts). Mit einem zweiten NXT-Block, also drei weiteren Motoren, lässt sich das Smartphone um eine weitere Achse schwenken (oben/unten). Das Smartphone muss sich hierzu mit zwei NXT-Blöcken über Bluetooth verbinden. Das Smartphone muss dann die Pakete immer an den entsprechenden NXT über Bluetooth weiter leiten.

1.4 Vierte Phase: Steuerung über mobile Endgeräte

In der vierten Phase soll die Steuerung des Roboters durch einen Computer auf ein Tablet oder Smartphone erweitert werden.

1.5 Fünfte Phase: Internetanbindung

In der dritten Phase soll die bisherige Steuerung des Roboters mittels WLAN (begrenzte Reichweite) durch WWAN ersetzt werden (weltweite Reichweite). Sowohl der Computer als auch das Smartphone benötigen hierfür einen Internetzugang. Geprüft werden muss hierfür, ob die Bandbreite hoch genug ist, um die Bilder in nahezu Echtzeit zu überliefern. Gegebenenfalls ist hier der Bildfrequenz entsprechend anzupassen.
Auch gibt es mehrere Ansätze, damit Smartphone und Computer miteinander kommunizieren können:

- VPN
- Dynamic DNS
- http-Server/web-dav/php
- peer-to-peer

2 Vorüberlegungen

2.1 Kommunikation

Damit das Programm schnell und einfach auch auf anderen Geräten oder Betriebssystemen funktioniert, habe ich mich entschieden auf allen Geräten in Java zu programmieren. Dies ermöglicht es einfacher Bluetooth- sowie WLAN-Kommunikation zu standardisieren.
Die Steuerung des Roboters durch das Smartphone kann grundsätzlich auch durch das LEGO® MINDSTORMS® NXT Communication Protocol (LCP) erfolgen. In Tests war dies einerseits langsamer als ein eigenes Protokoll (siehe 4.2 Kommunikation), andererseits ist durch das LCP eine Standardisierung der Kommunikation nicht möglich, da es speziell für Bluetooth- bzw. USB-Kommunikation entwickelt wurde. Folglich hätte für WLAN/WWAN die Kommunikation separat entwickelt werden müssen. Auch bietet ein eigenes Protokoll auf dem NXT mehr Kontrolle und Möglichkeiten, da das Programm auf dem Roboter ausgeführt wird und nicht nur Befehle über Bluetooth entgegen nimmt.

2.2 Graphische Benutzeroberfläche

Die Benutzeroberfläche steht nicht im Mittelpunkt der Arbeit und wird deshalb auf die notwendigen Funktionalitäten zur Steuerung reduziert.

2.3 Kameraübertragung

Die Kameraübertragung soll für eine optische Kontrolle möglichst aktuelle Bilder der Kamera liefern. Dieses Kriterium wird höher eingestuft als die Qualität des Einzelbildes oder die Übertragungsrate, da die Steuerung des Roboters so am besten unterstützt wird.

2.4 Soundübertragung

Da die Übertragung an Daten bereits bei WLAN und noch mehr bei WWAN limitiert ist, soll es eine Option zum Ausschalten der Soundübertragung geben.

3 Lego-Roboter

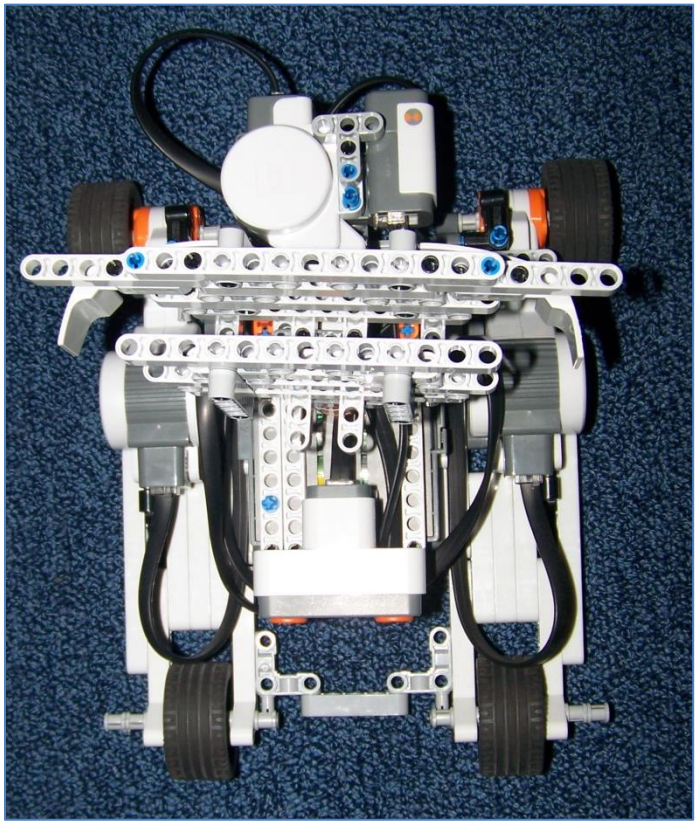

Abbildung 1: Lego Roboter ohne Gehäuse

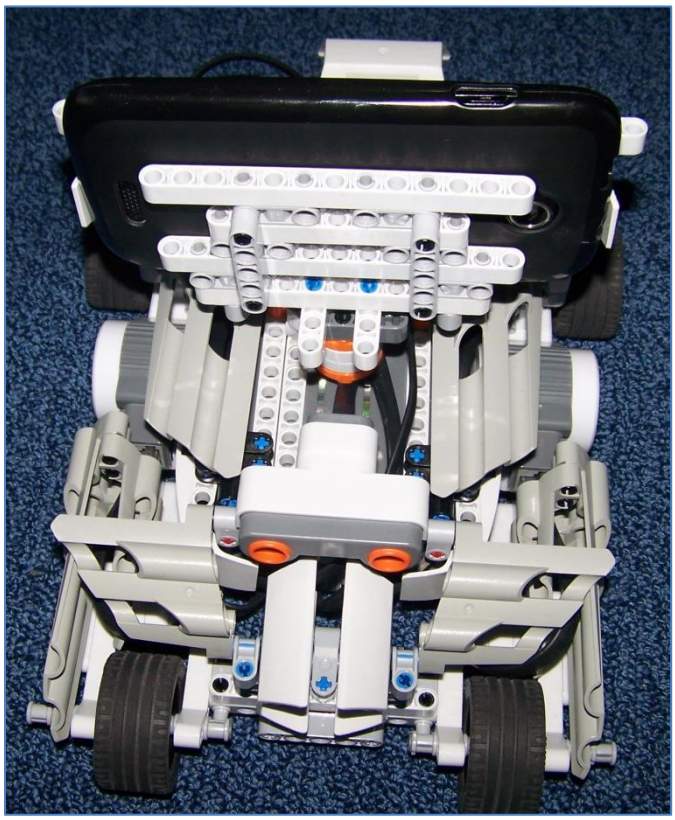

Abbildung 2: Lego Roboter mit Gehäuse und Smartphone

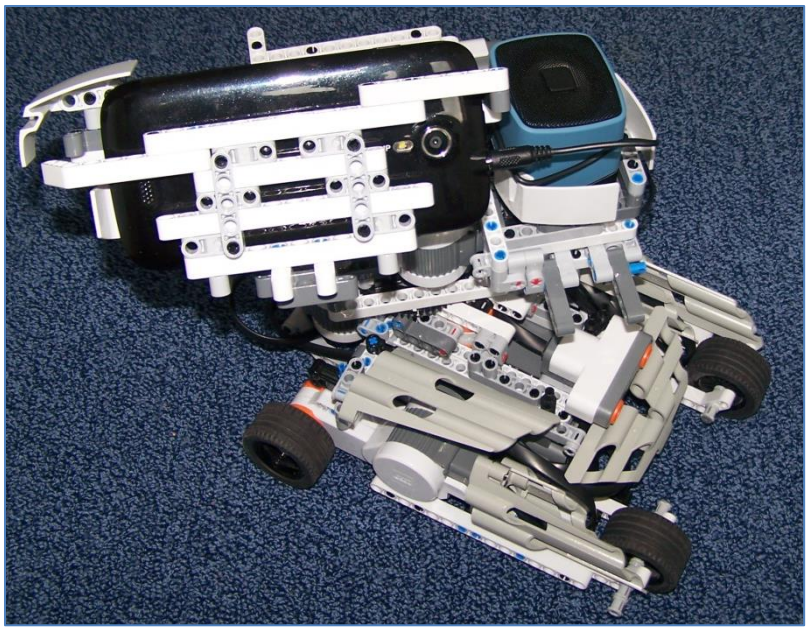

Abbildung 3: Lego Roboter mit zwei NXT-Blöcken (Ansicht vorne)

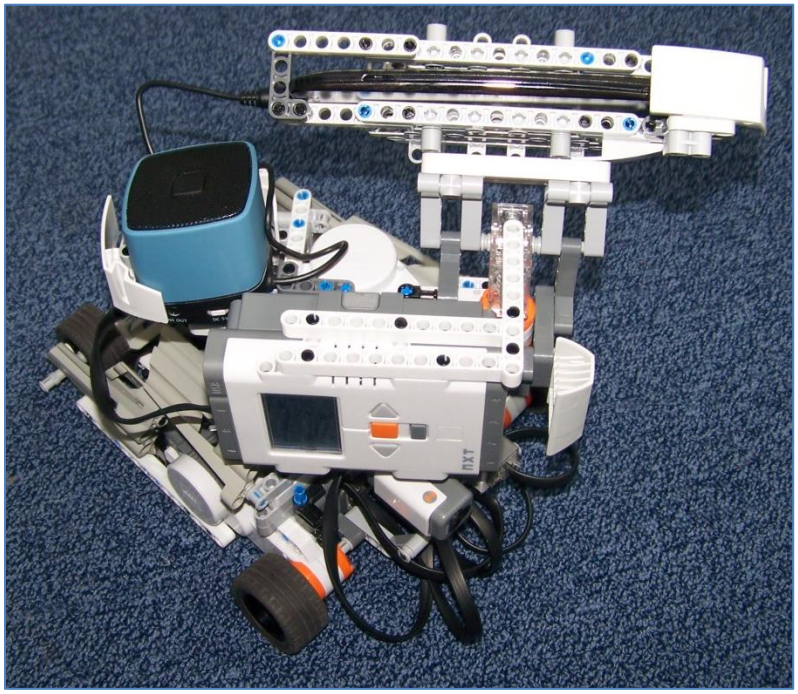

Abbildung 4: Lego Roboter mit zwei NXT-Blöcken (Ansicht hinten)

4 Modellierung

4.1 Verbindungsaufbau

Für die Datenübertragung zur Robotersteuerung ist der Aufbau einer Verbindung erforderlich. Sie erfolgt wie in Abbildung 5: Verbindungsaufbau dargestellt.

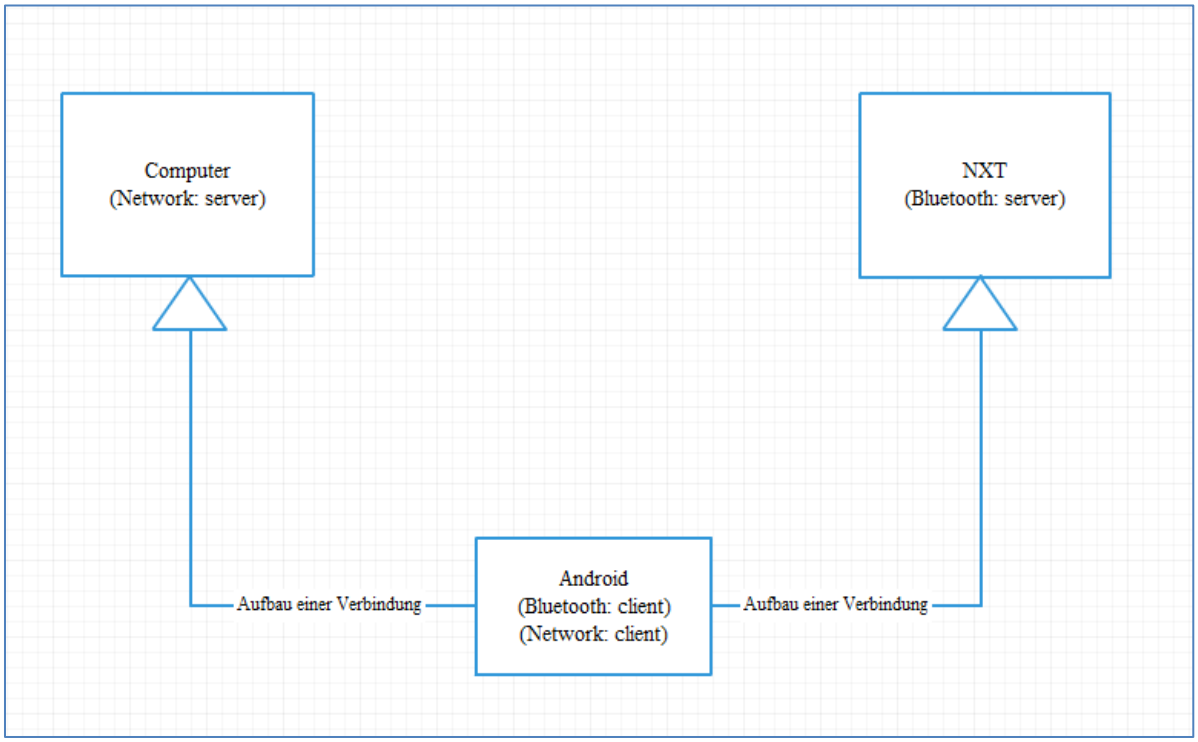

Abbildung 5: Verbindungsaufbau

Sowohl NXT als auch der Computer warten auf eine eingehende Verbindung. Das Smartphone verbindet sich mit dem NXT über Bluetooth mittels der MAC-Adresse und mit dem Computer durch die IP-Adresse. Beide Adressen werden im Smartphone eingegeben.

4.2 Kommunikation

Die Kommunikation soll nur aus einzelnen Kommandopaketen, die Befehle oder Daten übermitteln, bestehen. Um die Kommandopakete voneinander unterscheiden zu können, brauchen sie eine ID. Um möglichst wenig Daten übertragen zu müssen, aber eine Vielzahl von Paketen definieren zu können, wurde für die ID der Datentyp Short (2 Bytes) gewählt. Des Weiteren variiert die Größe von Paketen, deshalb muss auch für jedes Paket die Größe des Inhalts mitgeschickt werden. Hierfür bot sich ein Integer (4 Bytes) an, da die Größen sehr stark variieren.

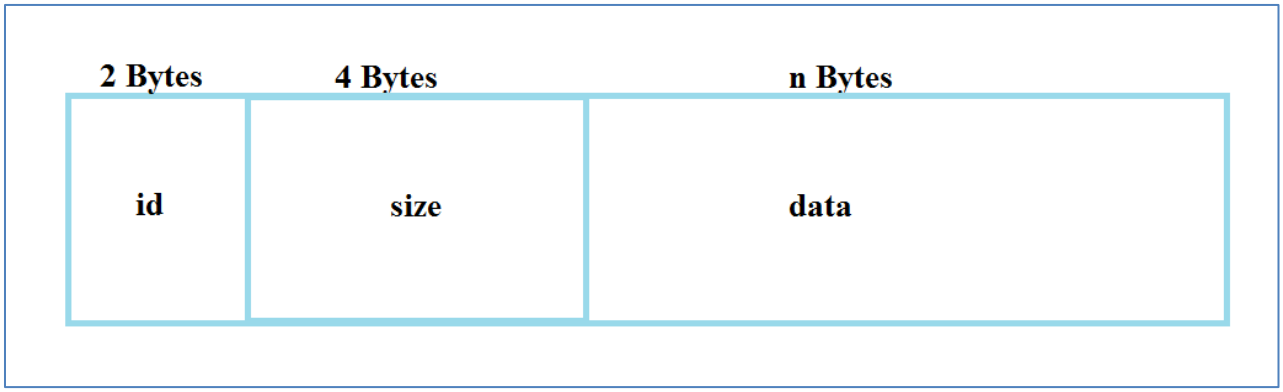

Abbildung 6: Aufbau eines Kontrollpakets

Das Gerät, das eine entsprechende Folge an Bytes empfängt, kann daraus das ursprünglich beim Sender erstellte Paket-Objekt erzeugen, da ID und Daten bekannt sind. Auch kann ein so wiederhergestelltes Paket durch das Smartphone vom Computer zum NXT oder umgekehrt weitergeleitet werden.

Folgende Abbildung zeigt die Kommunikation zwischen Controler und Roboter:

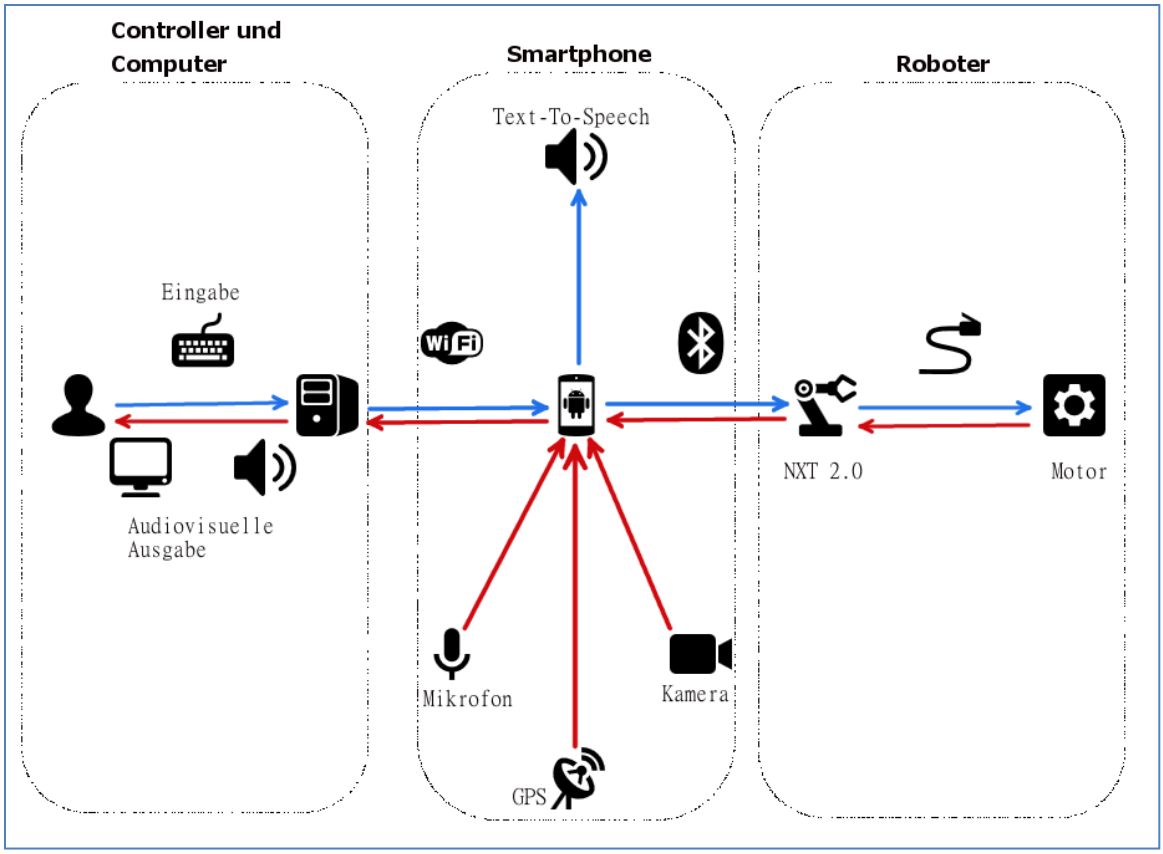

Abbildung 7: Kommunikationsverlauf

Blaue Pfeile stellen die Übertragung von Befehlen, rote von (Sensor-)Daten dar.

5 Implementierung

Der gesamte Sourcecode ist unter der Adresse: https://www.github.com/APN-Pucky/NXT-Android-Control hinterlegt.

5.1 Kommunikation

Die Kommunikation wird auf allen Geräten gleich gehandhabt. Die abstrakte Klasse *Packet* verfügt über 3 Attribute: einen statischen Short (*id*), ein Integer (*size*) und ein Byte-Array (*data*). Diese repräsentieren die einzelnen Elemente eines Pakets (siehe: 4.2 Kommunikation). Die Klasse besitzt zwei abstrakte Methoden. Die Methode *fillValuesData()* erstellt aus dem Wert des Packets das Byte-Array, die Methode *fillValues()* wandelt das Byte-Array in den Wert des Packets zurück.
Die abstrakte Klasse *Packet* hat für diverse Unterklassen (z.B. *PacketImage*, *PacketAudio*), jede Unterklasse hat eigene Methoden zum Umwandeln des Byte-Arrays.
Ein Objekt der Klasse *PacketHandler* wird mit einem *OutputStream* und einem *InputStream* als Parameter erstellt. Die zwei grundlegenden Funktionen *sendPacket(Packet p)* und *readPacket(Packet p)* greifen auf die *Streams* synchronisiert zu und versenden bzw. empfangen ein Byte-Array.

5.2 Steuerung des Roboters

Die Steuerung des Roboters erfolgt durch die Übertragung von *Packets*. Für die Bewegungen werden nur Kontrollpakete *PacketNxtCommand* verwendet. Jedes *PacketNxtCommand* beinhaltet ein Objekt der Klasse *NxtCommand*, welches aus einem Short zur Typangabe (vorwärts, rückwärts, rechtsgekurvt, halten, …) und einem Integer für weitere Informationen (stark gekrümmt, Geschwindigkeit, …) zusammengesetzt ist. Sowohl Typangabe als auch weitere Informationen werden aus der Eingabe am Computer (Steuerbefehl) errechnet. Daneben gibt es noch weitere Kontrollpakete zur Steuerung (siehe Tabelle).

Steuerbefehl	ID	Packet	Roboter
„1“-„9“	6	*PacketNxtCommand*	Stellt die Geschwindigkeit des Roboters ein.
„w“	6	*PacketNxtCommand*	Gerade Bewegung nach vorne
„s“	6	*PacketNxtCommand*	Gerade Bewegung nach hinten
„a“	6	*PacketNxtCommand*	Linkskurve nach vorne

„d“	6	*PacketNxtCommand*	Rechtkurve nach vorne
„y“	6	*PacketNxtCommand*	Linkskurve nach hinten (großer Radius)
„c“	6	*PacketNxtCommand*	Rechtskurve nach hinten(großer Radius)
„q“	6	*PacketNxtCommand*	Linkskurve nach vorne(großer Radius)
„e“	6	*PacketNxtCommand*	Rechtskurve nach vorne(großer Radius)
„l“	6	*PacketNxtCommand*	Rechts Drehung des Smartphones (45°)
„j“	6	*PacketNxtCommand*	Links Drehung des Smartphones (45°)
„b“	6	*PacketNxtCommand*	Dreht das Smartphone wieder auf 0°
„ “	6	*PacketNxtCommand*	Stoppt die Bewegung des Roboters
„f“	11	*PacketFlash*	Schaltet Licht des NXTs und des Smartphones an bzw. aus
„g“	12	*PacketSwapCamera*	Wechselt die Kamera (Front und Back)
„v“	14	*PacketMute*	Schaltet Soundübertragung des Smartphones ein bzw. aus.
„\n“	10	*PacketSpeakString*	Schickt einen Text an das Smartphone, den dieses als Sprachnachricht ausgeben soll
„t“	15	*PacketLang*	Ändert die Sprache die das Smartphone spricht
Erweiterung mit zweitem NXT-Block: „Schwenken der Smartphone Kamera“ (siehe Kapitel 5.6)			
„i“	6	*PacketNxtCommand*	Schwenkt die Smartphone Kamera nach oben
„k“	6	*PacketNxtCommand*	Schwenkt die Smartphone Kamera nach unten

5.3 GPS und Rotation

Um die Steuerung des Roboters (weltweit) zu erleichtern, wurde in die Benutzeroberfläche eine OpenStreetMap integriert (JMapViewer). Der NXT sendet alle 2 Sekunden ein Paket der Klasse *PacketMapData* an das Smartphone. In dieses fügt er den Tachowert des Motors, der die Drehung des Smartphones kontrolliert, und die Distanz zum nächsten Objekt in Fahrtrichtung mittels des Sonarsensors ein. Das Smartphone erhält das Paket und fügt diesem Kompass- sowie GPS-Daten hinzu. Das so erweiterte Paket wird dann an den Computer geschickt.

Der Computer kann nun auf der Karte anzeigen, wo sich der Roboter befindet. Mit dem Rotations- und dem Kompasswert lassen sich nun auch der Blickwinkel des Smartphones und die Ausrichtung des Roboters in der Karte anzeigen.

Abbildung 8: OpenStreetMap

5.4 Kameraübertragung

Das Umsetzten der Kameraübertragung auf der Computerseite war einfach, da lediglich das aktuellste empfangene Bild angezeigt werden muss. Android bietet die Möglichkeit einzelne Bilder oder gesamte Videos aufzunehmen. Beide Funktionen sind auf der Android Plattform nativ, d.h. in C geschrieben. Die Aufnahme eines einzelnen Bildes braucht allerdings etwa 1-2 Sekunden und eignet sich daher nicht für eine Kameraübertragung. Die Aufnahme eines Videos ist in einen *Stream* umleitbar, jedoch ist die Übertragung nicht flüssig, da das Video

mit Sound verschickt wird (hohe Datenmenge). Auch ist ein Video nicht als Kontrollpaket zu behandeln.

Dieses Problem wurde durch die Funktion *onPreviewFrame(byte[] data, Camera camera)* gelöst. Sie dient eigentlich der Voranzeige, was man fotografieren oder filmen möchte, und liefert kontinuierlich Bilder ohne Sound. Die Bilder werden in *PacketImage* Objekten versendet.

5.5 Sound und Sprachübertragung

Die Soundübertragung (vom Smartphone zum Computer) wird auf dem Smartphone mit der Klasse *AudioRecord* umgesetzt. Ein Objekt dieser Klasse nimmt im Hintergrund des Programms Sound nach einem bestimmten *AudioFormat* (Hier: sample rate: 44100, Channel: mono, Encoding: PCM 16bit) auf. Die letzten aufgenommenen Bytes lassen sich mit der Funktion *read(byte[] buffer, int start, int end)* in ein Byte-Array übertragen. Dieses wird in einem *PacketAudio* an den Computer geschickt.

Bei Sprachübertragung (vom Computer zum Smartphone) wird ein Text am Computer (Sprachnachricht) eingegeben und als *PacketSpeakString* dem Smartphone übermittelt. Dieses gibt den Text mittels der bei Android mitgelieferten Text-to-Speech Engine wieder. Die verwendete Sprache kann durch ein *PacketLang*-Paket geändert werden (bisher: Englisch, Deutsch und Spanisch).

5.6 Erweiterung der Funktionalität durch einen zweiten NXT-Block

Ein NXT-Roboter ist mit drei Motoren ausgestattet (siehe 1.2 Zweite Phase: Erweiterung der Funktionalitäten). D.h. der Roboter kann das Smartphone lediglich um die eine Achse drehen(links/rechts). Mit einem zweiten NXT-Block, also drei weiteren Motoren, lässt sich das Smartphone um eine weitere Achse schwenken (oben/unten). Das Smartphone muss sich hierzu mit zwei NXT-Blöcken über Bluetooth verbinden (2 MAC-Adressen, siehe Abbildung 9: Verbindungsaufbau mit zwei NXT-Blöcken). Aufgrund der Standardisierung der Kommunikation ließ sich auch schnell ein Kontrollpaket für die Steuerung des zweiten NXT-Blocks erstellen. Das Smartphone muss dann die Pakete (siehe Kapitel 5.2 Steuerung des Roboters, Tabelle) immer an den entsprechenden NXT über Bluetooth weiter leiten. Welcher NXT-Block angesprochen werden soll wird in

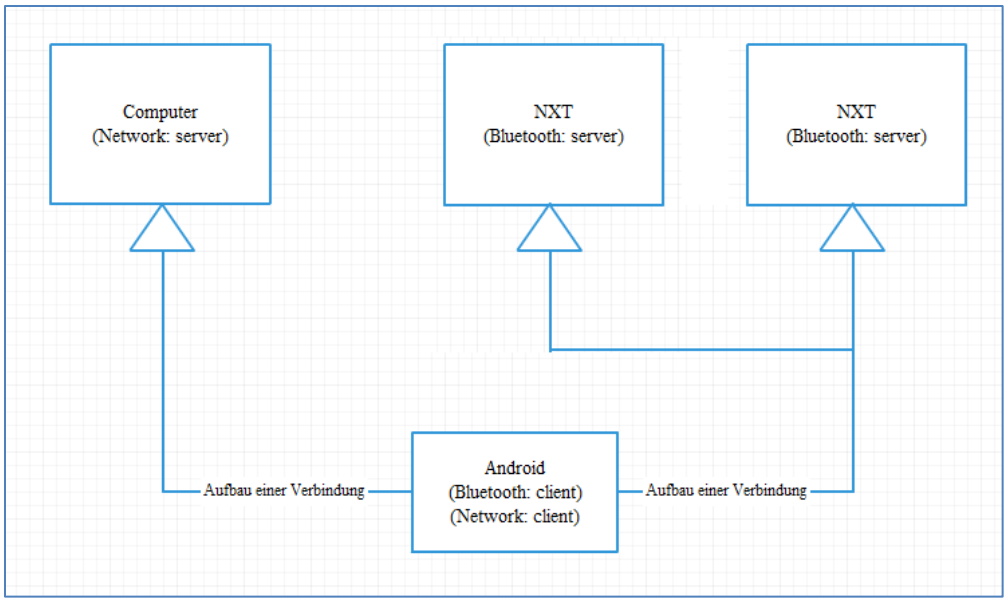

Abbildung 9: Verbindungsaufbau mit zwei NXT-Blöcken

6 Probleme und Lösungen

In der Implementierungs- und Testphase traten einige Probleme auf. Einige davon werden hier beispielhaft erläutert. Sofern beschrieben, wurden die Lösungen umgesetzt.

6.1 Kamerazugriffseinschränkung

Problem: Da die meisten Smartphones eine Front- und eine Back-Kamera haben, wollte ich dies aus nutzen und eine Art Rückspiegel mit in die Benutzeroberfläche am Computer einbauen. Es zeigte sich jedoch, dass bei Android aus Sicherheitsgründen immer nur auf eine Kamera zugegriffen werden kann.

Lösung: Durch einen Befehl kann von der Front- zur Back-Kamera und umgekehrt gewechselt werden.

6.2 Steuerungsschwierigkeiten

Problem: Durch die symmetrische Konstruktion des Roboters, ist das Steuern des Roboters ziemlich kompliziert, da das Objektiv der Smartphone Kamera nicht in der Mitte des Roboters sitzt. Aufgrund der perspektivischen Aufnahme der Kamera fährt der Roboter nicht parallel zum aufgenommenen Bild.

Lösung: 1. 1. Die Benutzeroberfläche wurde um eine Klasse, die Richtungslinien in das übertragene Bild zeichnet, erweitert.

2. Der Roboter mit zwei NXT-Blöcken wurde so konstruiert, dass das Objektiv des Smartphones mittig auf der Drehachse des Roboters sitzt (siehe Abbildung 3: Lego Roboter mit zwei NXT-Blöcken (Ansicht vorne)).

2. Der zweite Roboter wurde so konstruiert, dass das Objektiv des Smartphones in der Mitte des Roboters ist.

6.3 Verschiedene Grundlagen

Problem: Die Java VM des Lego Roboters mit dem Betriebssystem leJOS ist durch begrenzten Speicher und begrenzte Rechenleistung eingeschränkt.

Lösung: Zum Umwandeln von Werten (z.B. integer, string) in Bytes wurde eine eigene Klasse geschrieben. Andere Klassen wurden als Dummy erstellt um Unterschiede der Geräte (z.B. Roboter hat nur eine Dummy-Image-Klasse) auszugleichen und die betriebssystemübergreifende Kommunikation zu standardisieren.

6.4 Zufällige Verbindungsunterbrechungen

Problem: Die Bluetooth-Verbindung des Roboters mit dem Smartphone wird zufällig unterbrochen und ein neuer Verbindungsaufbau ist erst nach Neustart des Smartphones möglich.

Lösung: Das Problem wurde durch ein Verbindungssignal behoben werden. Dieses wird alle 1,5 Sekunden vom Smartphone an den Roboter gesendet werden. Wenn der Roboter nun 2 Sekunden lang kein Verbindungssignal bekommt, bleibt er stehen und wartet auf eine Wiederherstellung der Verbindung, bzw auf ein neues Paket.
annwird1,52, bzw. ein neues Paket.

7 Ausblick

7.1 Mobile Endgeräte

7.2 WWAN

Die weltweite Steuerung des Roboters wird schwer realisierbar, da die bereits genannten Lösungsansätze:

- VPN
- Dynamic DNS
- http-Server/web-dav/php
- peer-to-peer

nicht kostenlos zur Verfügung stehen oder die Übertragung von Daten extrem einschränken.

8 Zusammenfassung

In der Arbeit wurden die „Erste Phase: Bluetooth und WLAN Kommunikation“ und die „Zweite Phase: Erweiterung der Funktionalitäten“ implementiert. Die „Dritte Phase: Erweiterung der Funktionalitäten durch einen zweiten NXT-Block“ wurde nach dem Regionalwettbewerb am 27.02.2015 implementiert. Die „Vierte Phase: Steuerung über mobile Endgeräte“ und die „Fünfte Phase: Internetanbindung“ sind grundsätzlich aufgrund der Überlegungen in der Ausarbeitung realisierbar.

Die erfolgreiche Umsetzung der Ansteuerung eines LEGO® MINDSTORMS® NXT Roboters mittels Bluetooth und Internet unter Verwendung eines Smartphones zeigt, dass die Möglichkeiten des NXT-Roboters erheblich erweitert werden können. Die Reichweite der Steuerung des Roboters von ca. 5 Metern bei Bluetooth konnte im Test auf mehr als 15 Meter erhöht wurden (je nach Smartphone WLAN-Hotspot oder verwendeten WLAN Router).

Die Standardisierung durch Programmierung in Java in Verbindung mit dem Konzept der Kommunikation mittels einzelner Kommandopakete ermöglicht es, die Steuerung des Roboters schnell und effizient zu erweitern. So konnte zum Beispiel ein zweiter NXT-Block zum Schwenken der Smartphone-Kamera zusätzlich angesteuert werden. Ebenso könnte die Ansteuerung eines Greifarms realisiert werden.

9 Literaturverzeichnis

Android. 2015. Android Developer. [Online] 14. Januar 2015. http://developer.android.com/training/index.html.

Fraunhofer IAIS. 2010. *Roberta® Programmieren mit Java.* Stuttgart : Fraunhofer IRB Verlag, 2010. ISBN 978-3-8167-8401-2.

leJOS. 2015. leJOS Tutorial. [Online] 14. Januar 2015. http://www.lejos.org/nxt/nxj/tutorial/Communications/Communications.htm.

OpenStreetMap. 2015. OpenStreetMap Wiki. [Online] 14. Januar 2015. http://wiki.openstreetmap.org/wiki/JMapViewer.

stackoverflow. 2015. stackoverflow. [Online] 14. Januar 2015. http://stackoverflow.com/questions/15349987/stream-live-android-audio-to-server; http://stackoverflow.com/questions/3058919/text-to-speechtts-android.

Ullenboom, Christian. 2005. *Java ist auch eine Insel.* Bonn : Galileo Press GmbH, 2005. ISBN 3-89842-526-6.

Valk, Laurens. 2011. *LEGO®-ROBOTER.* Heidelberg : dpunkt.verlag, 2011. ISBN 978-3-89864.