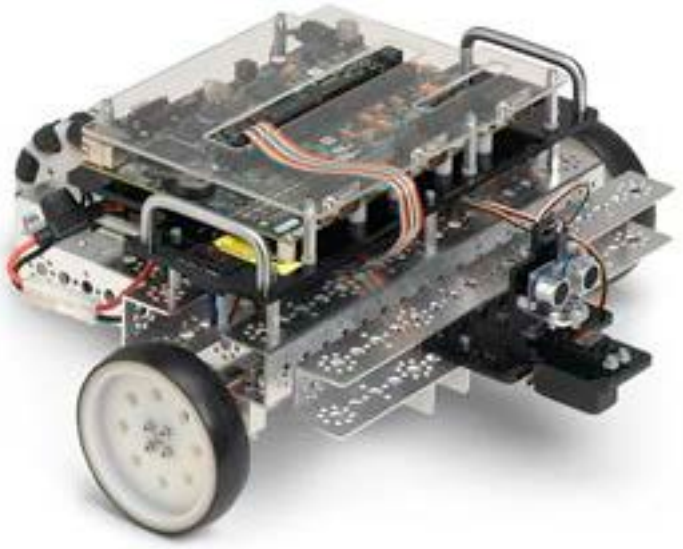
Berner Fachhochschule
Technik und Informatik

Projektarbeit 1 2012

NI LabVIEW Robotics Starter Kit 2.0



Fachbereich Maschinenbau
Studierender Martin Pletscher
Professor Prof. Daniel Lanz

Zusammenfassung

Vor einiger Zeit hat Prof. Daniel Lanz ein sogenanntes NI Robotics Starter Kit erworben. Die Idee war, dieses als Eurobot Plattform einzusetzen. Doch leider kam es aus Mangel an Zeit und der neuesten LabVIEW Robotics Version nie dazu diesen Plan in die Realität umzusetzen. Dann wurde die Idee geboren, die ersten Schritte in der Entwicklung mit dem Robotics Starter Kit als PA1 anzubieten.

Das Ziel war es, dass Robotics Starter Kit in Betrieb zu nehmen und die Grundlagen der Programmierung zu erarbeiten. Weiter sollten einige einfache Aufgabenstellungen mit dem Starter Kit umgesetzt werden, um zu zeigen welche Möglichkeiten die Technologie bietet.

Ich habe mich nun dieser Aufgabe angenommen. Innerhalb von 6 Wochen habe ich mich in LabVIEW Robotics eingearbeitet und einige Ideen umgesetzt. Darunter die manuelle Steuerung des Roboters über ein LabVIEW GUI, das fahren nach eingegebenen Pattern oder die Kommunikation mit einem iPad. Zum Schluss des Projektes ist es mir gelungen die Teile zu einem grossen Ganzen zu vereinen und den Roboter drahtlos mit einem iPad zu steuern. Auch ist es nun möglich zwischen automatischer und manueller Navigation umzuschalten.

Während des Projektes habe ich immer wieder festgestellt, dass ich eigentlich nur an der Oberfläche gekratzt habe und dass die LabVIEW Robotics Technologie noch unvorstellbar viel zu bieten hat. In weiteren Projektarbeiten sollte man unbedingt versuchen, noch mehr zu lernen und ein tieferes Know How aufzubauen.

Inhaltsverzeichnis

1 Einleitung

Um programmierbare logische Bausteine wie FPGAs programmieren zu können sind gute Kenntnisse der Programmiersprachen VHDL oder Verilog zwingend erforderlich. Dieser Umstand stellt den Konstrukteur konkreter Anwendungen oft vor unüberwindbare Hürden. National Instruments bietet mit LabVIEW als graphische Programmiersprache eine Alternative zu textuellen Programmiersprachen, zur Programmierung von FPGAs und Echtzeitprozessoren. Das LabVIEW Robotics Starter Kit, zur Prototyperstellung autonomer Systeme gedacht, umfasst nebst einer erweiterbaren mechanischen Grundkonstruktion einen Echtzeitprozessor, ein rekonfigurierbares FPGA sowie analoge- und digitale Ein- und Ausgänge. [pa1_aufgabenstellung, Prof. Daniel Lanz, 26.03.2012]

Das Ziel dieser PA1 ist es, dass Robotics Starter Kit 2.0 in Betrieb zu nehmen und mit LabVIEW Robotics erste einfach Programme dafür zu entwickeln. Im späteren Verlauf der Arbeit sollen dann Ideen für weitere Aufgabenstellungen entwickelt werden. Diese sollen wenn möglich umgesetzt und getestet werden.

Diese Arbeit soll auch dazu dienen, einem nachfolgenden Studenten einen einfacheren Einstieg in die Roboter Programmierung mit LabVIEW und dem Robotics Starter Kit 2.0 (Single Board RIO) zu ermöglichen.

2 Inbetriebnahme Robotics Kit

2.1 Inbetriebnahme

Zur Inbetriebnahme des Roboters sind einige wichtige Schritte in der aufgeführten Reihenfolge auszuführen.

1. Der Roboter muss vor dem Einschalten mit dem BFH Netzwerk verbunden werden. Dies geschieht über ein LAN Kabel das beim Roboter am Switch eingesteckt wird.
2. Der Motor Schalter sollte in die OFF Stellung gebracht werden, da man nie weiss welches Programm der letzte Benutzer im Speicher hiterlassen hat.
3. Nun kann die Steckverbindung vom Akku zum Roboter verbunden und der Master Schalter in die ON Stellung gebracht werden.

Der Roboter wird dann zuerst einen Selbsttest durchführen, was an der leuchtenden Status LED erkannt werden kann. Dieser dauert ca. 3 Sekunden. Informationen zu den LED und allgemein zum Single Board RIO können der Kurzanleitung (Ressource/Manuals/NI_sbRIO-961x_963x_964x_Quick_Reference.pdf) oder dem User-Manual (Ressource/Manuals/NI_sbRIO-961x_963x_964x_and_NI_sbRIO-9612XT_9632XT_9642XT_User_Guide.pdf) entnommen werden.

Danach wird es einige Zeit (bis zu 5 Minuten) dauern bis der WLAN AP voll funktionsbereit und das robinet WLAN aufgebaut ist. Wenn auf dem Roboter ein Programm als Startup definiert ist, kann es noch weitere Zeit beanspruchen dieses zu laden.

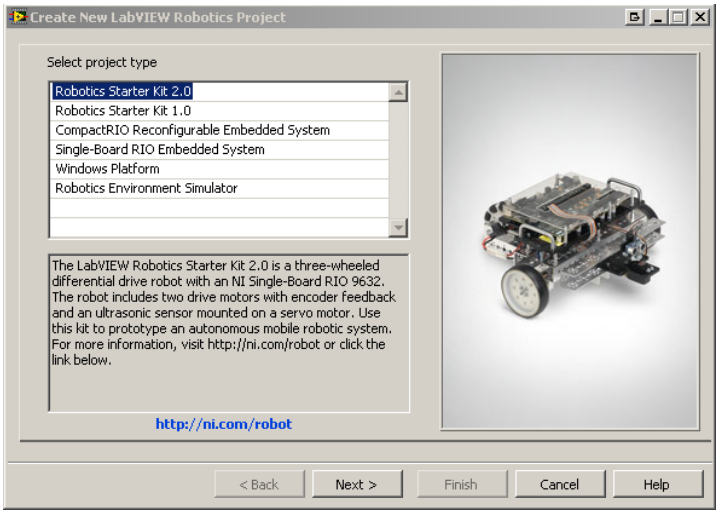
Der Roboter hat von der BFH folgende Netzwerkdaten zugewiesen bekommen:

IP : 147.87.172.82
Subnetmask : 255.255.254.0

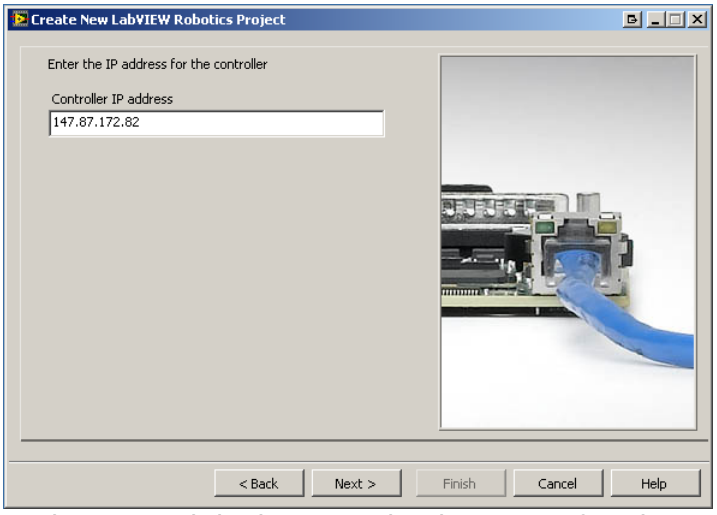
Zur Kontrolle kann der Roboter einmal angepingt werden. Wenn er reagiert, sollte alles funktionsbereit sein.

2.2 Demo Projekt ausführen

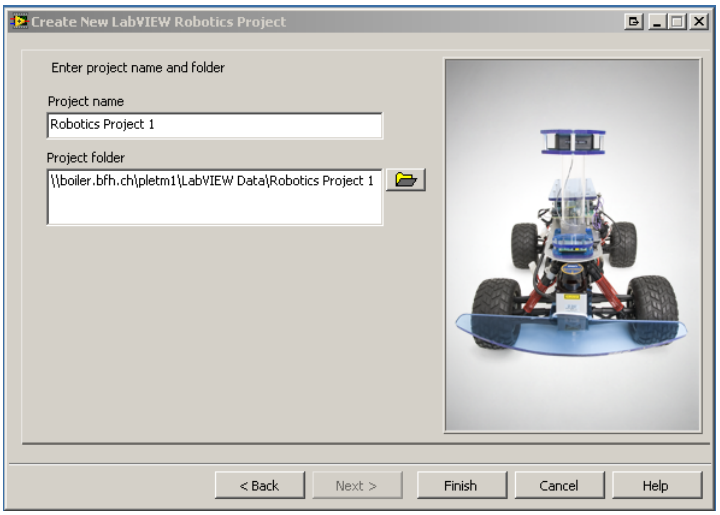
Um den Roboter ein wenig kennen zu lernen, kann man ein NI Robotics Demo Projekt erstellen und auf dem Roboter ausführen. Dazu klickt man im LabVIEW Robotics Startfenster auf Robotics Project.



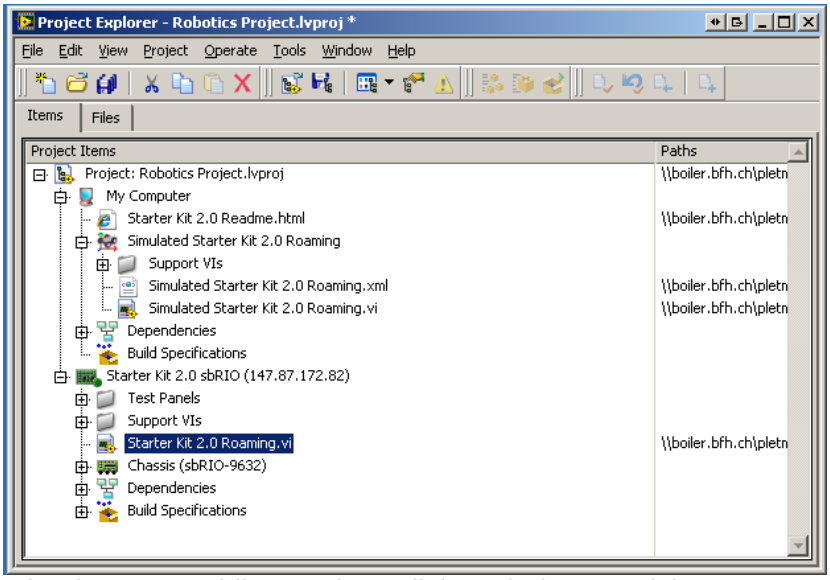
Dadurch öffnet sich der Robotics Project Assistent. Im ersten Schritt kann man die Art des Projektes auswählen. Hier sollte man das Robotics Starter Kit 2.0 auswählen.
Im nächsten Schritt muss man die IP Adresse des Roboters angeben.



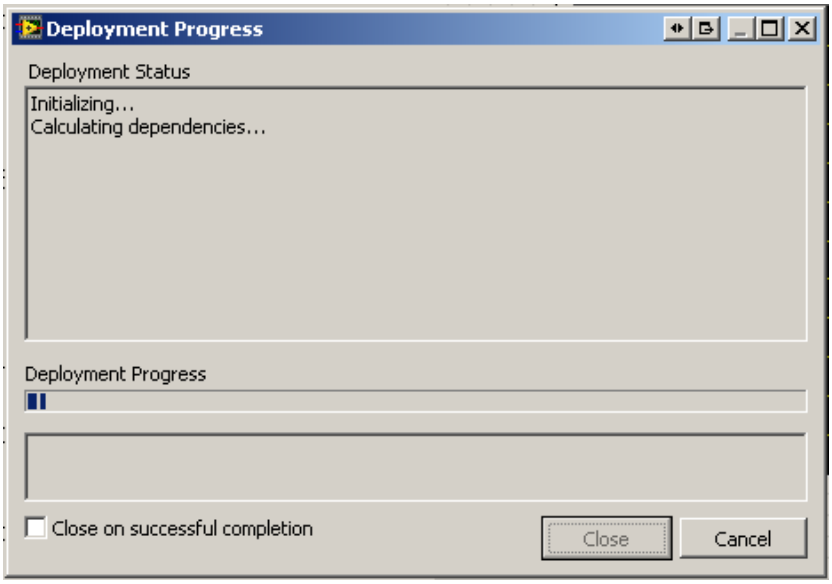
Im letzten Schritt kann noch ein Name für das Projekt vergeben und der Speicherpfad



angepasst werden. Zum Schluss klickt man auf Finish und das Projekt wird erstellt.
Mit einem Doppelklick auf die Starter Kit 2.0 Roaming.vi kann die Demo VI geöffnet werden.



Mit einem anschliessenden Klick auf den Ausführen Button wird die VI auf den Roboter heruntergeladen und ausgeführt.



2.3 Neues Projekt erstellen

Um ein eigenes Robotics Projekt zu erstellen, klicken sie im LabVIEW Robotics Startbildschirm (Abb. 2.3.1) auf Blank Project.

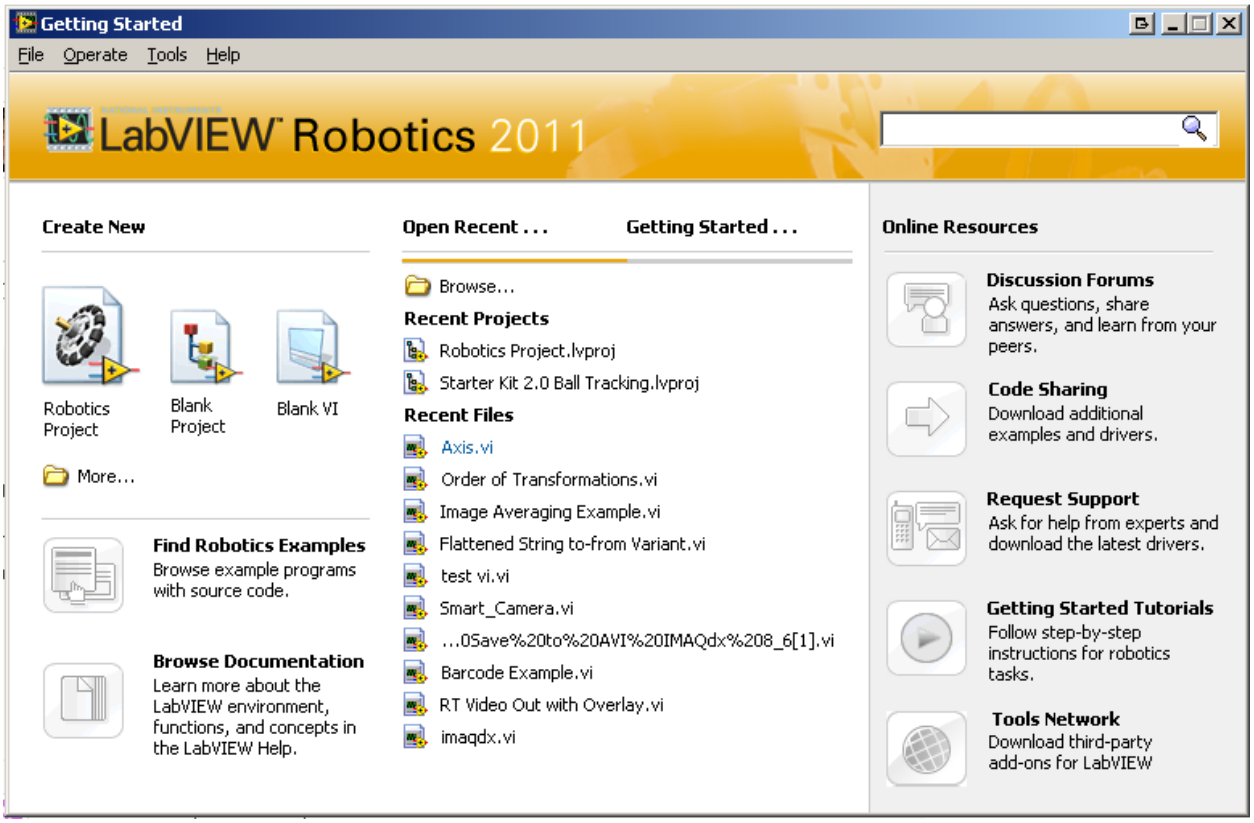


Abb. 2.3.1

Abb. 2.3.2

Wenn sich der Projekt Explorer geöffnet hat, kann man auf der obersten Ebene im Baum rechtsklicken (Abb. 2.3.2) und dann „New->Targets and Devices“ anklicken. Dadurch öffnet sich nun ein Dialog, (Abb. 2.3.3) der es ermöglicht auszuwählen, was für ein Target man hinzufügen

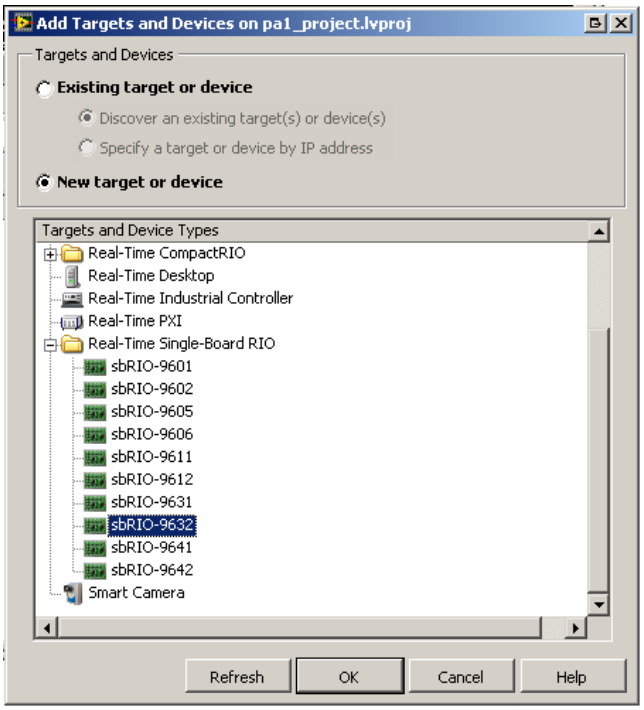


Abb. 2.3.3

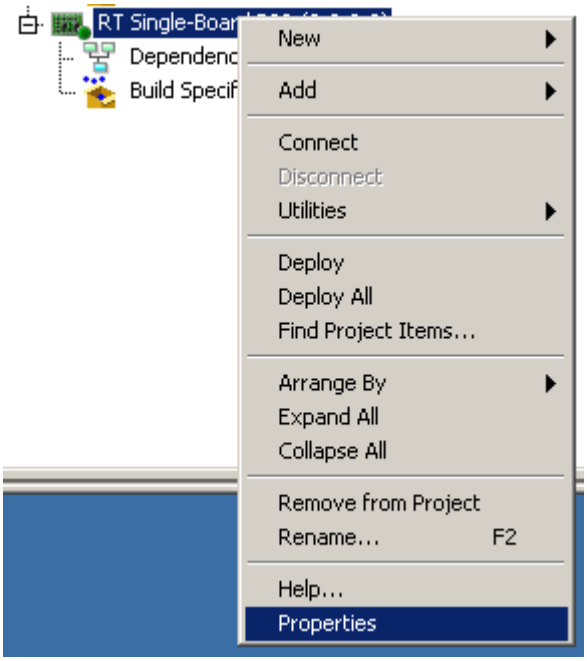


Abb. 2.3.4

möchte.
In diesem Dialog wählt man dann „New target or device“ und dann unter „Real-Time Single-Boar RIO“ das sbRIO-9632 aus. Zur Bestätigung muss nun der OK Button gewählt werden. Dieses Target muss nun noch konfiguriert werden, indem man darauf rechtsklickt und Properties (Abb. 2.3.4) auswählt. Danach erscheint ein Dialog (Abb. 2.3.5) mit einer Vielzahl an

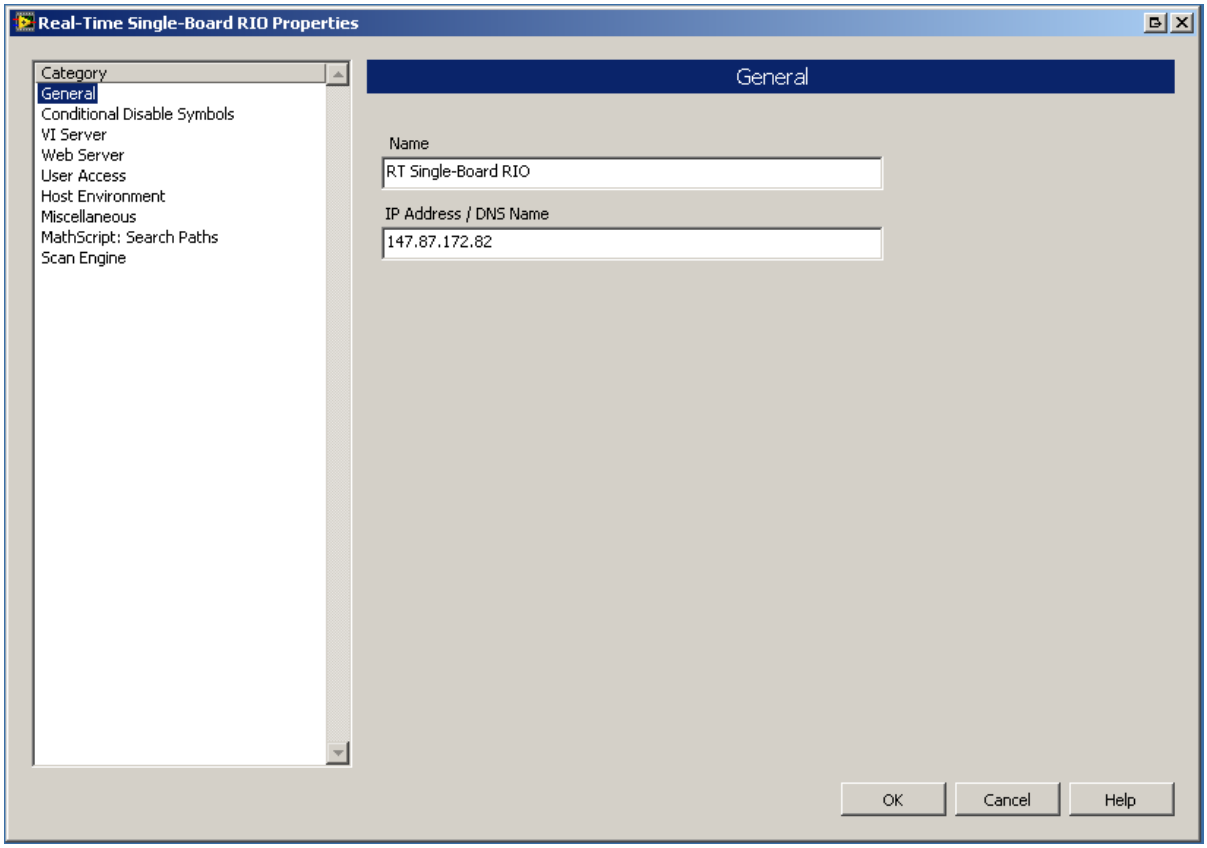


Abb. 2.3.5

Einstellmöglichkeiten. Die wichtigste ist zu diesem Zeitpunkt die IP Adresse. Diese muss in das entsprechende Feld eingetragen werden. Danach kann man den Dialog mit OK verlassen.

Nun kann man auf dem Target eine neue VI erstellen. (Abb. 2.3.6)

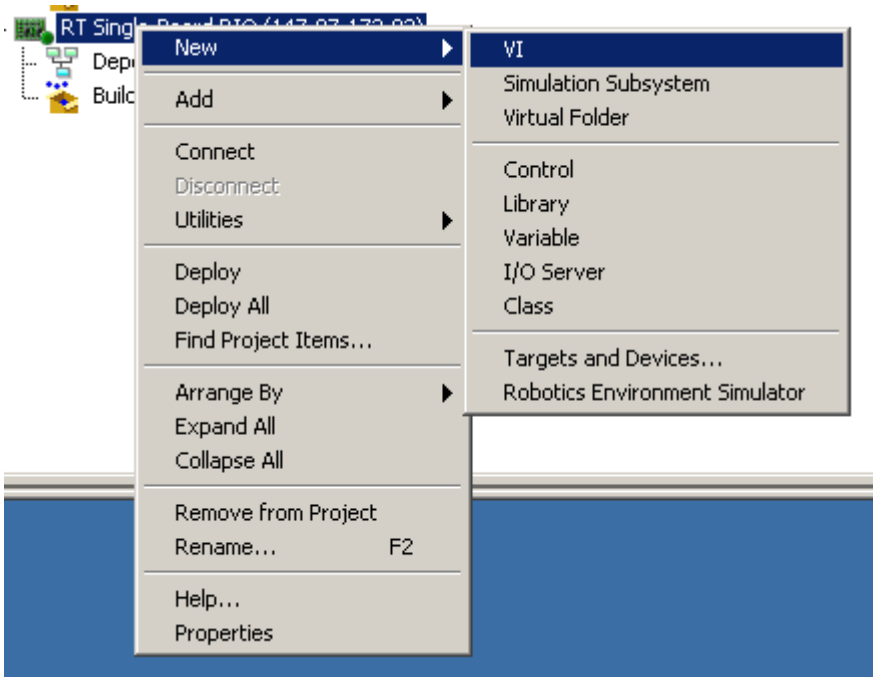


Abb. 2.3.6

2.4 VIs auf dem Roboter ausführen

Um eine VI direkt auszuführen, z.B. zum Testen der Funktionalität, reicht es diese ganz normal auszuführen. Dadurch wird das Programm auf den Roboter heruntergeladen und dort ausgeführt.

2.5 VI als Startup definieren

Um eine VI als Startup zu definieren muss mann die Applikation zuerst Kompilieren und dann als Startup ausführen.

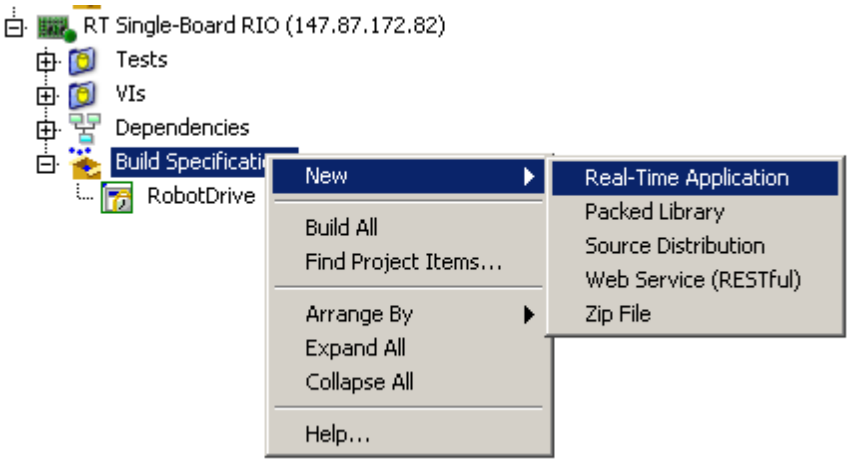


Abb. 2.5.1

Dazu klickt man mit der rechten Maustaste auf den Eintrag „Build Specifications“ und wählt „New->Real-Time Application“ (Abb. 2.5.1). Dadurch wird eine neue Applikation erzeugt. Im erscheinenden Dialog (Abb. 2.5.2) kann man nun einen Namen für die Applikation festlegen.

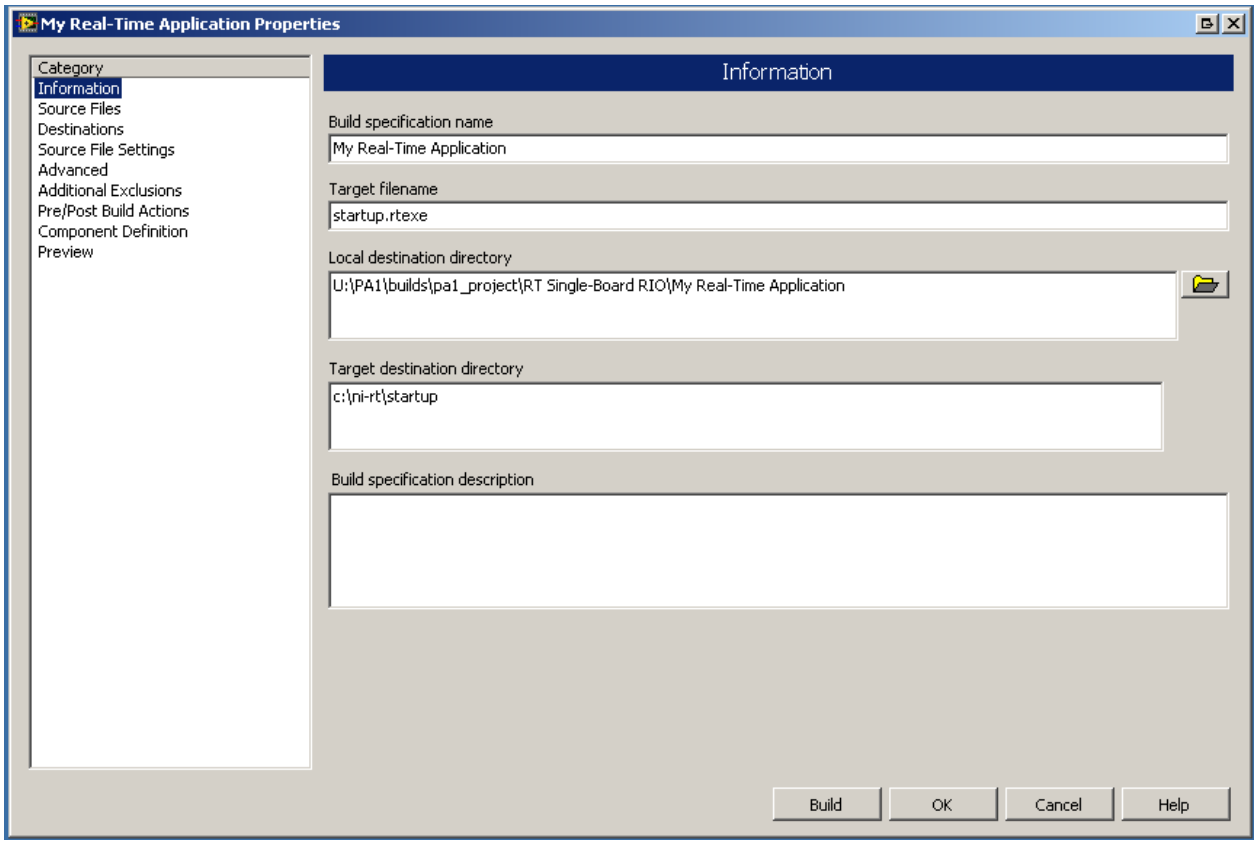


Abb. 2.5.2

In der Kategorie Source Files (Abb. 2.5.3) kann nun die VI ausgewählt werden, die bei Starten des Roboters ausgeführt werden soll. Zusätzlich können im unteren Feld VIs hinzugefügt

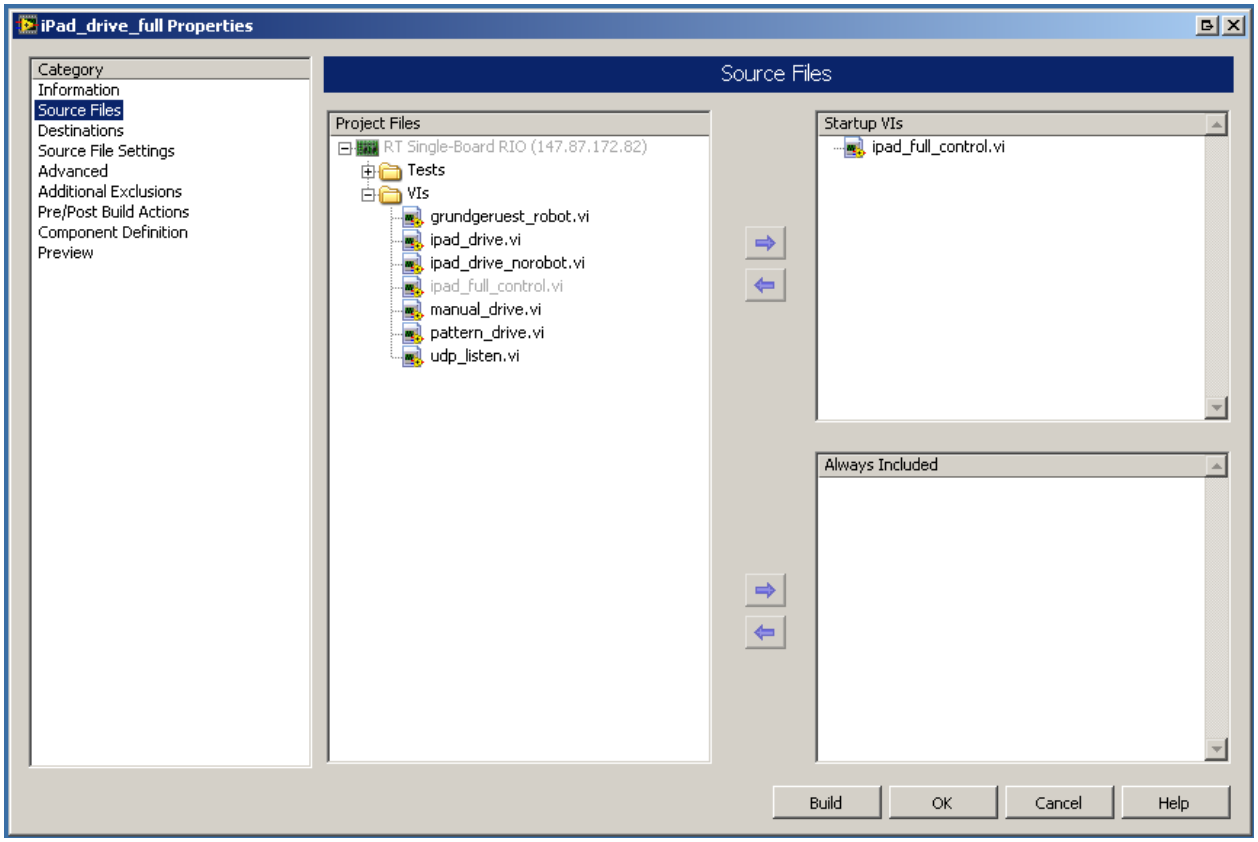


Abb. 2.5.3

werden die zum Ausführen der Applikation erforderlich sind.
Mit einem Klick auf Build wird die Konfiguration abgeschlossen und die Applikation Kompiliert.

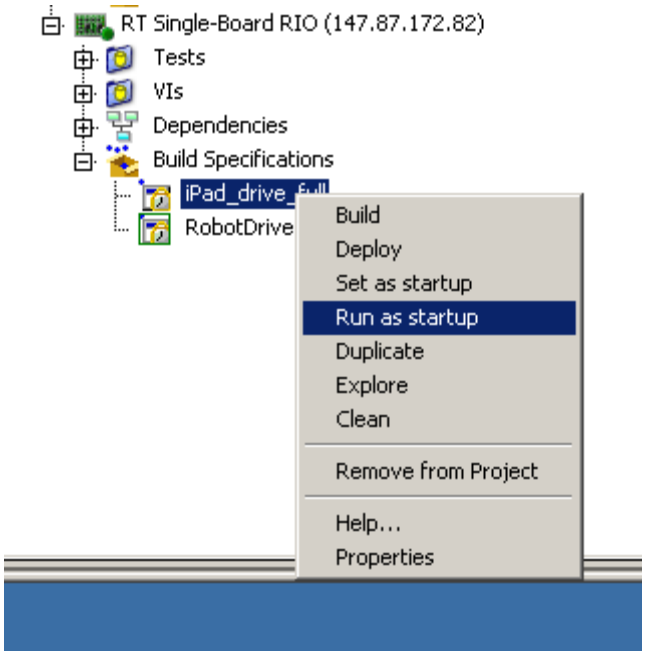


Abb. 2.5.4

Um nun die Applikation als Start zu setzen, klickt man mit der rechten Maustaste auf die soeben erstellte Applikation (Abb. 2.5.4) und wählt „Run as Startup“. Dadurch wird die Applikation auf den Roboter heruntergeladen und als Start Applikation verankert. Danach muss der Roboter einmal neu Gebootet werden.

2.6 Tipps und Tricks

2.6.1 Keine Netzwerkverbindung zum Roboter

Beim Einschalten des Roboters kann es vorkommen, dass der Netzwerkstack nicht richtig initialisiert wird. Dies kann man daran erkennen, dass der Roboter nicht angepingt werden kann. Die einzige Lösung die ich für dieses Problem gefunden habe, ist den Roboter noch einmal neu zu starten.

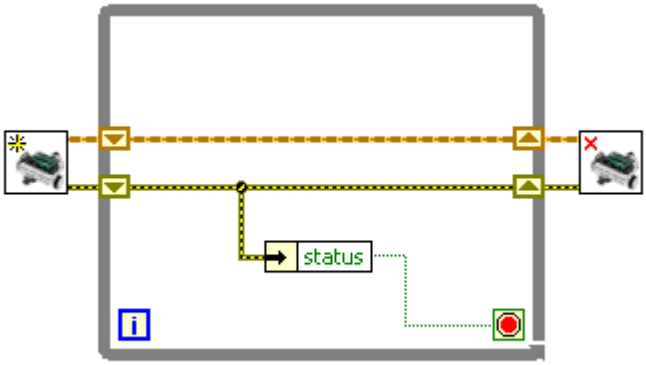
2.6.2 Tablet erhält keine IP Adresse im robinet WLAN

Da das robinet WLAN so konfiguriert wurde, dass es nur eine Erweiterung des BFH Netzwerkes darstellt, besitzt es keinen eigenen DHCP Server. Man muss daher beim ersten einwählen des Tablets / Phones den Roboter per Kabel mit dem BFH Netzwerk verbinden. Die Konfiguration wurde so gewählt, damit man mit allen Computern im Schulgebäude, ohne neue Konfiguration des Netzwerkes, mit dem Roboter kommunizieren kann.

3 Robotics Kit Grundlagen

3.1 Grundgerüst Robotics Kit

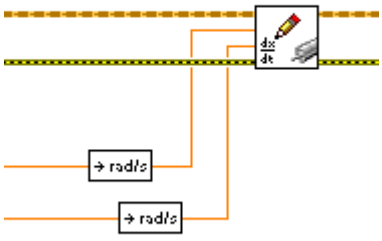
Das Grundgerüst um das Starterkit anzusteuern besteht im Wesentlichen aus nur 3 Elementen. Dem **„Initialize Starter Kit 2.0“** Block, einer while Schleife und dem **„Close Starter Kit“** Block.



Zusätzlich sollte die while Schleife noch abgebrochen werden sobald ein Fehler auftritt. Dazu wird der Error Cluster unbundled und der Status abgefragt. Solange dieser false ist läuft die Schleife weiter.

Dieses Grundgerüst kann nun kompiliert und auf den Roboter geladen werden. Auch wenn es nicht sehr sinnvoll ist, ist es doch vollständig und fehlerfrei. Darauf können nun alle weiteren Entwicklungen aufbauen.

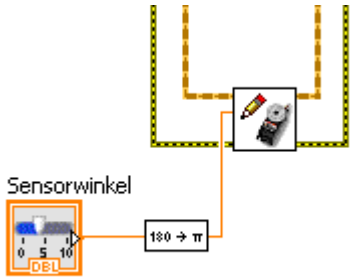
3.2 Ansteuerung Motoren



Um die Motoren des Starterkits anzusteuern kann man einfach einen **„Write DC Motor Velocity Setpoints“** Block in das oben vorgestellte Grundgerüst einfügen. Dieser erwartet die Session und den Error und zusätzlich zwei Geschwindigkeiten für die jeweiligen Motoren. Diese Geschwindigkeiten müssen in rad/s übergeben werden. Allerdings gibt es unzählige Umrechnugsblöcke, so dass mit fast allen Arten von Geschwindigkeiten gerechnet werden kann.

Wenn man nun erreichen möchte, dass der Roboter sich vorwärts bewegt, muss mann die eine Geschwindigkeit negativ angeben. Das rührt daher, dass beide Motoren in die gleiche Richtung drehen, aber spiegelverkehrt im Roboter eingebaut sind.

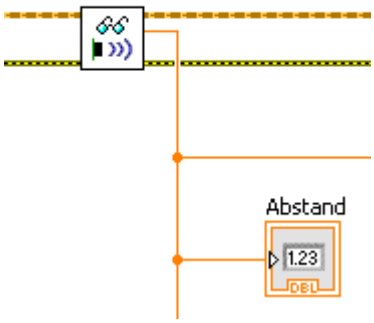
3.3 Ansteuerung Servo



Für die Ansteuerung des Servos gibt es den Block **„Write Sensor Servo Angle“** und ebenfalls das Pendant **„Read Sensor Servo Angle“,** auf das ich jedoch nicht weiter eingehen werde.

Der Block hat die üblichen Session und Error Ein- und Ausgänge. Dazu hat er einen Angle Eingang. Über diesen Eingang kann der Winkel des Servos absolut von -90 bis +90 Grad gesetzt werden. Da der Block den Winkel allerdings in rad erwartet, muss der Winkel umgerechnet werden.

3.4 Ansteuerung Ultraschall Sensor Ping)))



Der Ultraschall Sensor Ping))) von Parallax kann mit dem Block **„Read PING))) Sensor Distance“** des Starterkits ausgelesen werden. Er hat die üblichen Session und Error Ein- und Ausgänge und zusätzlich noch einen Distanz Ausgang. Auf diesem gibt der Block einen double Wert aus, der den Abstand in Meter darstellt.

3.5 Steering Frame

Das Steering dient dazu, den Roboter in in Echtzeit zu steuern. Dazu wird zuerst das Steering Frame des Roboter definiert. Danach kann man mit vielen verschiedenen Berechnungs- und

Steuermethoden die Bewegung des Roboters berechnen. Ich habe zwei davon genauer angeschaut und in meinen Handling Aufgaben verwendet.

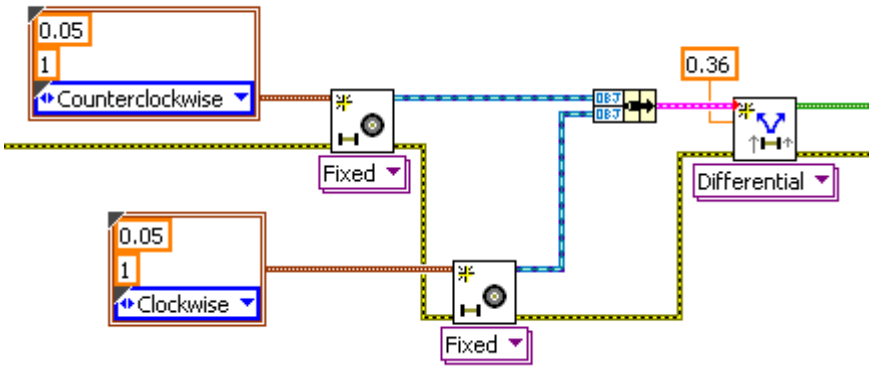


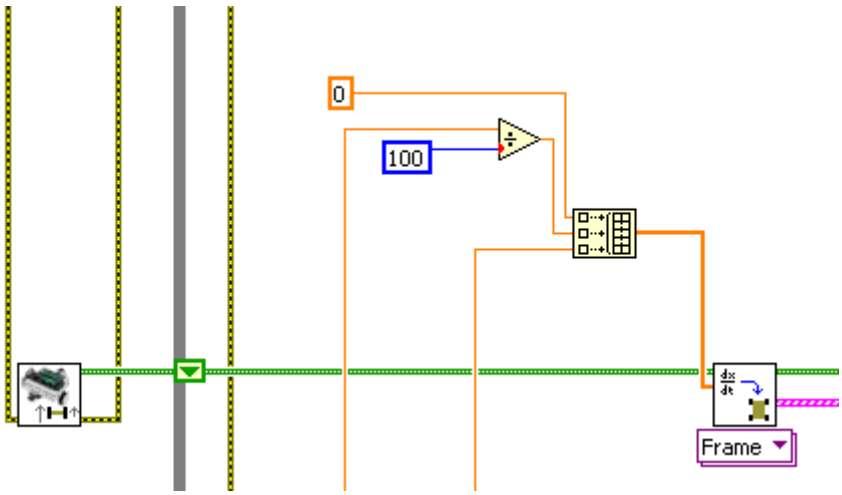
Abb. 3.5.1

Um ein Steering Frame für den Starter Kit Roboter zu definieren, kann man den Vordefinierten Block **„Starter Kit Steering Frame“** verwenden. Wenn man diesen genauer betrachtet (Abb. 3.5.1) wird ersichtlich, dass man eigentlich die Geometrie des Roboters darin definiert. Man fügt alle angetriebenen Räder hinzu und definiert den Abstand zwischen Ihnen. Diese Informationen verwendet der **„Vehicle Steering"** Block dann, um die Geschwindigkeiten für die einzelnen Antriebe zu berechnen.

3.5.1 Vehicle Steering

Wenn man nun das Steering Frame initialisiert hat, kann man den Block **„Vehicle Steering"** einfügen und das Steering Frame einhängen. Dieser Block hat 2 verschiedene Implementierungen aus denen ausgewählt werden kann. Frame und Arc.

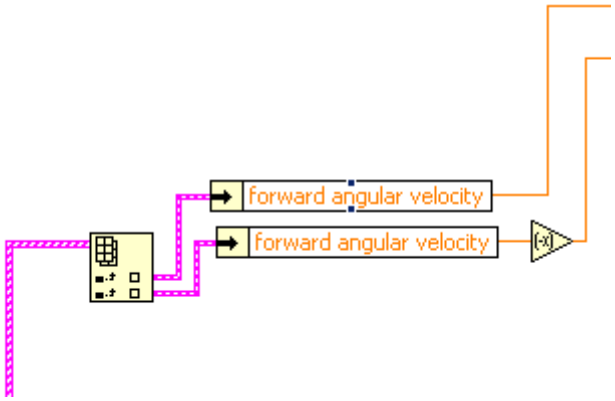
3.5.1.1 Frame



Die Frame Implementation kann aus 3 Eingabewerten die Geschwindigkeiten für beide Antriebsachsen berechnen. Folgende Werte werden vom Block als Array erwartet:

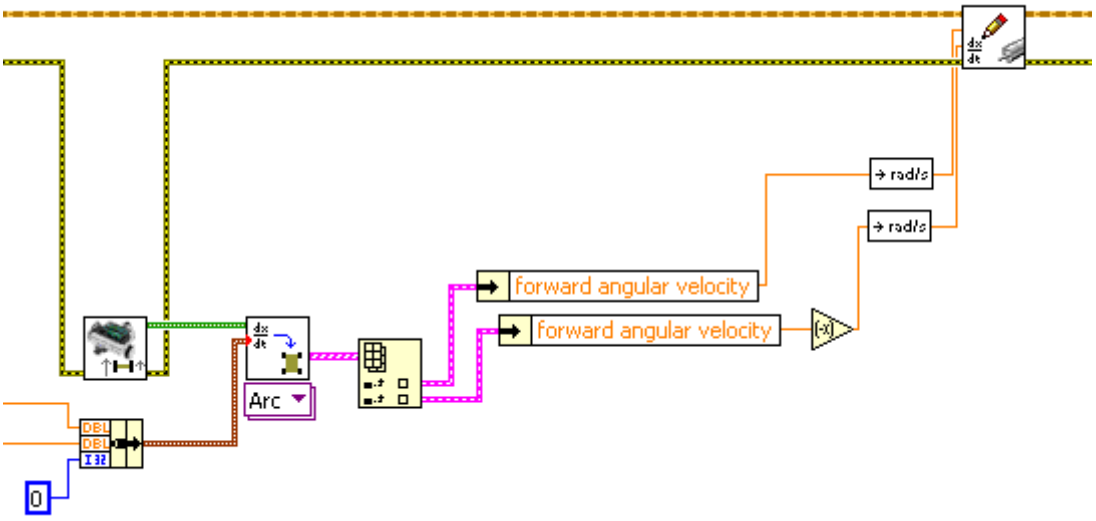
Y Geschwindigkeit
X Geschwindigkeit
Rotationsgeschwindigkeit

Der Block gibt dann eine Array von Clustern mit den berechneten werten aus. Diese müssen dann zuerst indexiert und unbundled werden und können anschliessen in Form zweier double Werte auf den **„Write DC Motor Velocity Setpoints“** Block gegeben werden.



3.5.1.2 Arc

Genau wie die Frame Implementation benötigt auch die Arc Implementation 3 Eingabewerte, erwartet diese allerdings als Cluster von double Werten.



Die Folgenden 3 Werte werden erwartet:

Radius der gefahren werden soll
Tangentiale Geschwindigkeit
Dritter wert = 0

Leider ist es mir nicht gelungen den dritten Wert zu verstehen. Darum habe ich für diesen immer 0 angenommen. Und es hat so funktioniert, wie ich mir das vorgestellt habe.

Die Ausgabe der berechneten Werte erfolgt exakt gleich wie bei der Frame Implementation.

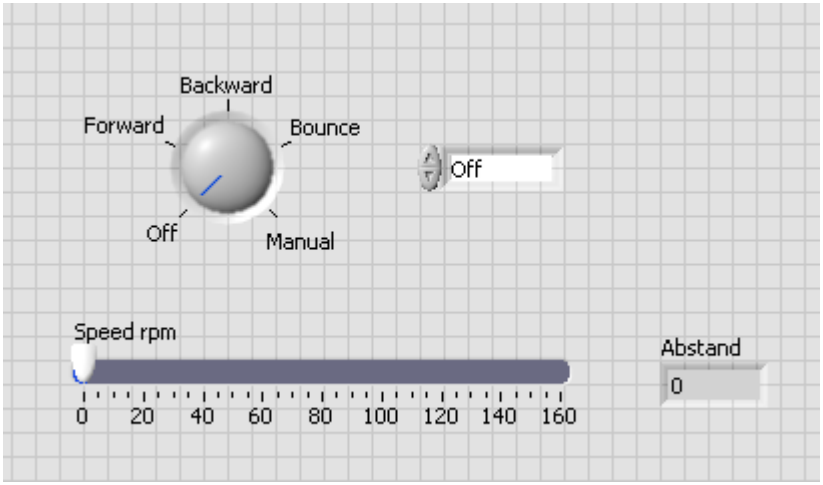
4 Aufgaben

4.1 Manuelles Fahren

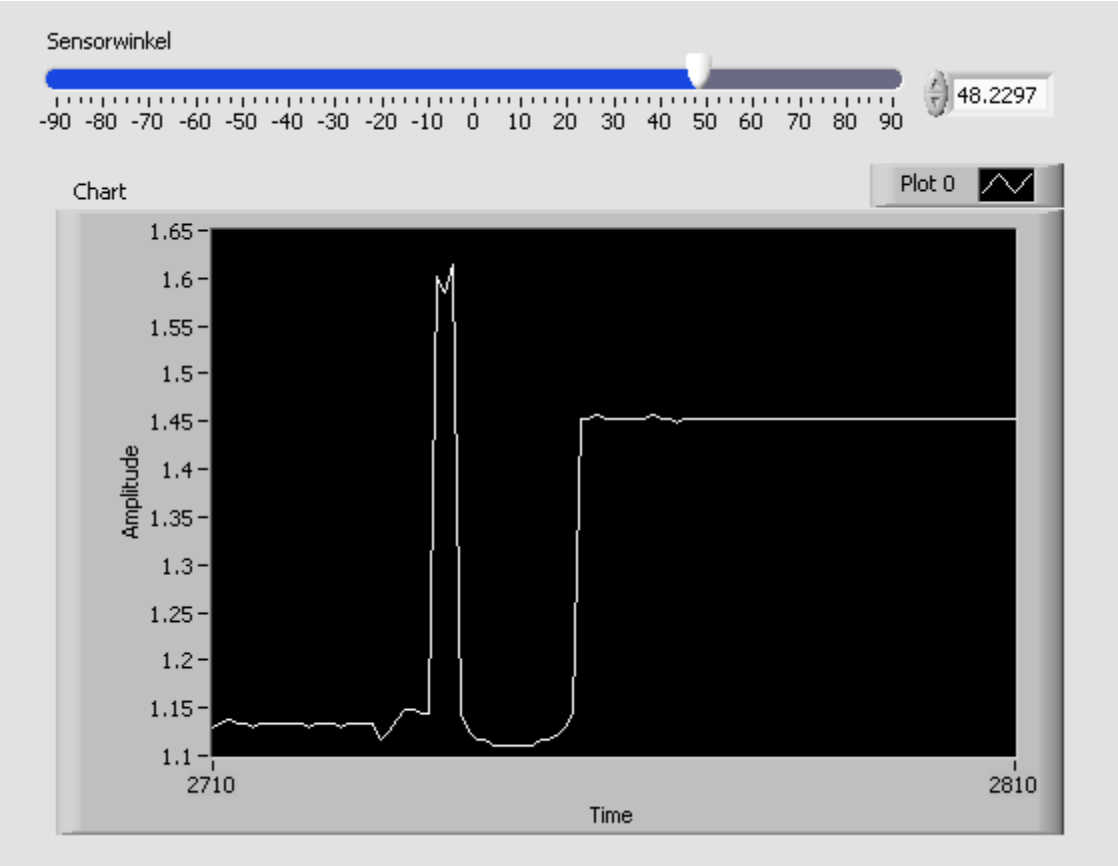
Die entsprechende VI befindet sich in pa1_project/VIs/manual_drive.vi

Das Ziel war es, den Roboter mit einer einfachen LabVIEW GUI direkt manuell steuern zu können. Dabei gab es verschiedene Möglichkeiten.

Als erstes habe ich mich mit der Ansteuerung der Motoren beschäftigt. Dazu habe ich einen Auswahlschalter hinzugefügt mit dem man vorwärts oder rückwärts fahrt auswählen kann. Dazu gibt es einen Slider mit dem die Geschwindigkeit bestimmt werden kann.



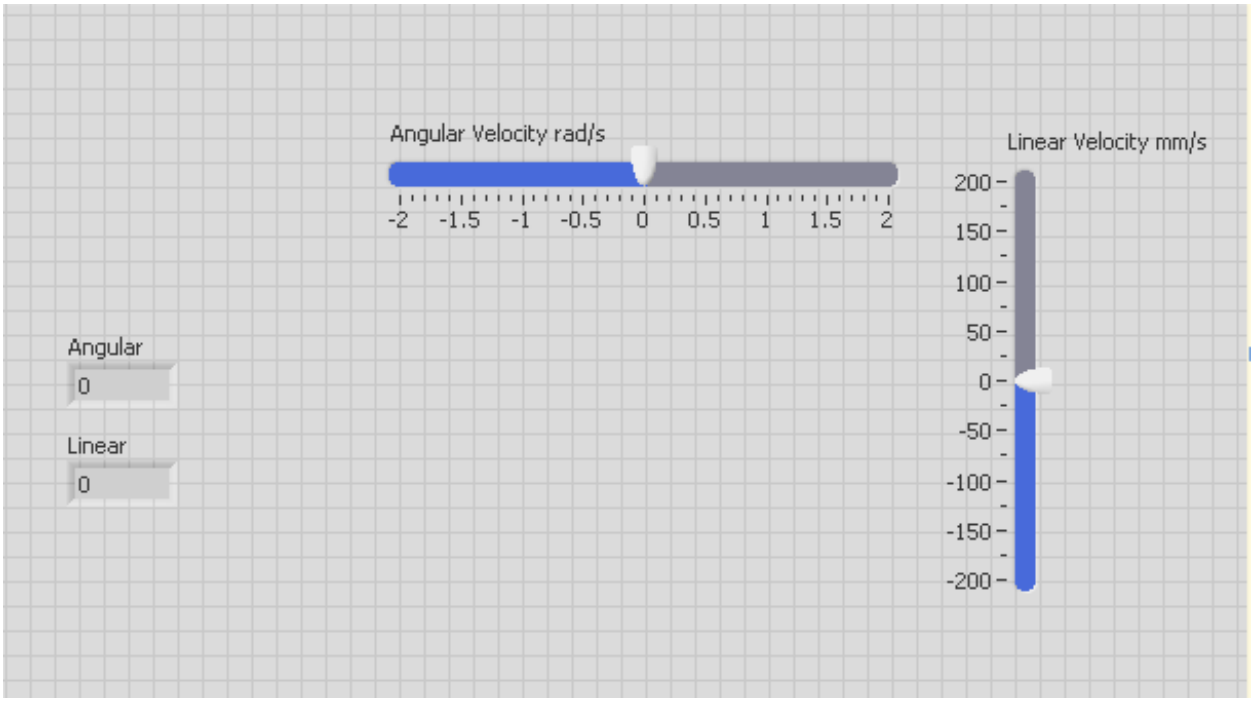
Nach der Vor- und Rückwärtsbewegung kam der Ultraschallsensor an die Reihe. Ein Waveform Chart stellt die gemessenen Distanzen grafisch dar. Die Ausgabe des Sensors alleine war jedoch nicht sehr interessant. Wichtiger war das man den Sensor mittels Servo Motor von -90 bis 90 Grad schwenken kann. Die Einstellung geschieht wieder über einen Slider. Beim bewegen des Sliders kann man nun erkennen, wie sich die gemessene Distanz ändert.



Um beide Elemente (Motor Steuerung und Ultraschallsensor) zu kombinieren dient der bounce Modus. In diesem Modus fährt der Roboter so lange vor oder rückwärts bis der Abstand entweder grösser als 2 Meter oder kleiner als 1 Meter ist. Danach wird die Bewegungsrichtung umgekehrt.

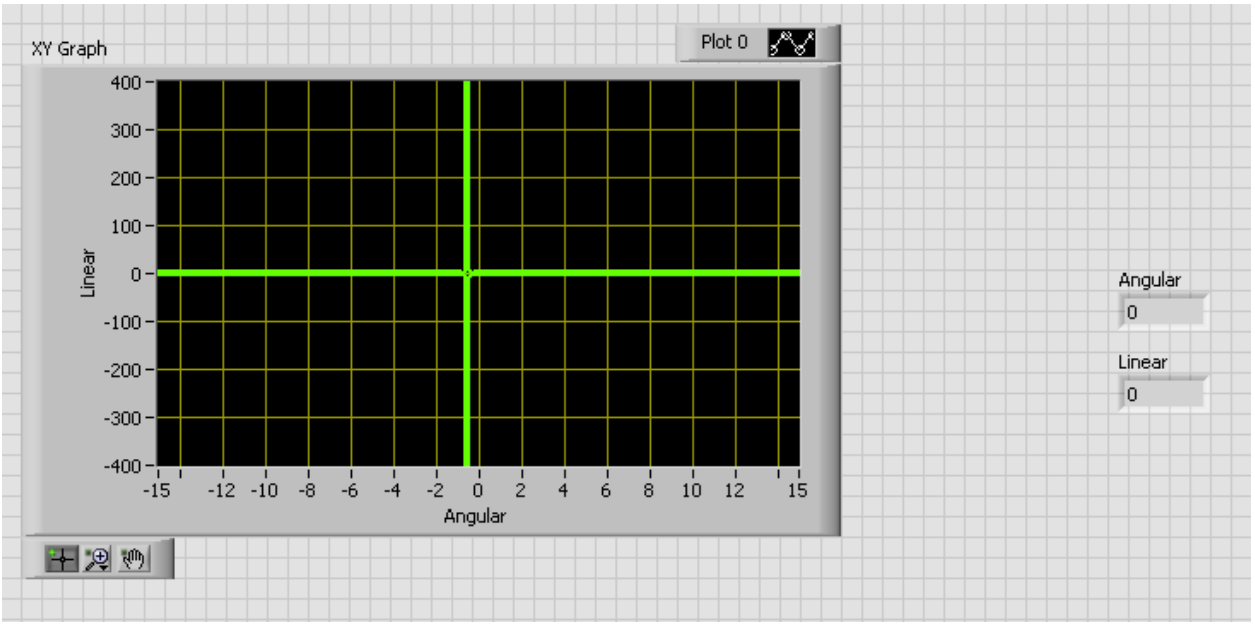
Die letzte Möglichkeit ist die manuelle Steuerung die als differentiale Steuerung umgesetzt wurde. Dabei kann eine lineare Geschwindigkeit und eine Rotationsgeschwindikeit vorgegeben werden. Der Roboter berechnet dann daraus die Geschwindigkeit für beide Räder.
In der LabVIEW GUI gibt es zwei Gruppen von Steuerelementen mit denen die beiden Variablen beeinflusst werden können.

Zum einen sind das zwei einfache Slider, einer für die Lineare und einer für die Rotationsgeschwindigkeit. Damit kann der Roboter schon einigermassen präzise gesteuert



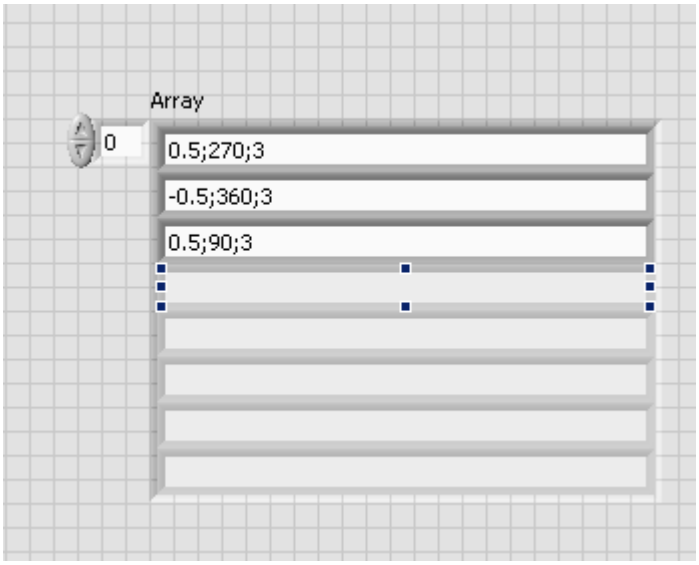
werden. Es ist jedoch nicht möglich beide Geschwindigkeiten gleichzeitig zu verändern.

Diese Einschränkung kann überwunden werden, indem eine XYGraph Anzeige in ein Steuerelement umgewandelt wird. Dadurch kann der Benutzer zwei Werte gleichzeitig beeinflussen wenn er das Achsenkreuz im Steuerelement bewegt. Gleichzeitig ähnelt diese Eingabemethode auch einem analogen Joystick.



4.2 Pattern Fahren

Das Ziel dieser Aufgabe war es mit dem Roboter vorgegebene Pattern abzufahren. Dazu habe ich ein einfaches GUI aufgebaut, in dem eine Reihe von Strings eingegeben, von LabVIEW geparst und dann vom Roboter ausgeführt werden können.



Diese Strings müssen folgende Form haben:

{Radius [m]};{Winkel [°]};{Geschwindigkeit[cm/s]}

Diese String wird dann mit Split Blöcken in seine Einzelteile zerlegt und mit einem ARC Steering Block in eine Geschwindigkeit umgerechnet.

Das Vorzeichen beim Radius entscheidet dabei über die Drehrichtung.
Auf der Abbildung sieht man ein Pattern, dass die Zahl Acht ergibt.

4.3 Fernsteuerung mit iPad/iPhone

Das Ziel dieser Aufgabe war es den Roboter mittels eines iPad/iPhones (ebenfalls mit anderen Tablets oder Mobiltelefonen möglich) manuell zu steuern und nach Bedarf auch den automatischen Navigationsmodus aktivieren zu können.

4.3.1 UDP Kommunikation

Um eine nichtblockierende Kommunikation herzustellen eignet sich das UDP Protokoll hervorragend. Beim UDP Protokoll wird nicht geprüft ob die Pakete beim Empfänger ankommen oder nicht. Dadurch wir der Client nicht Blockiert, wenn er keine Pakete erhält.

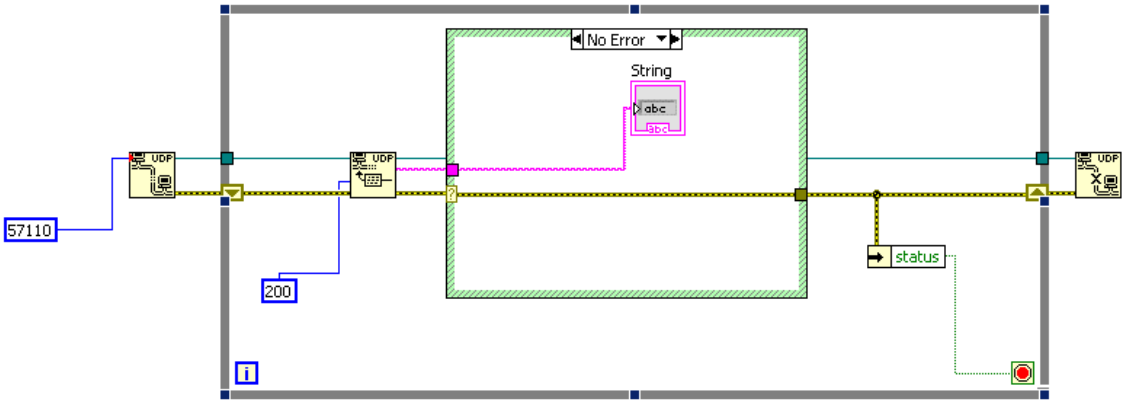


Abb. 4.3.1.1

Zuerst habe ich mit LabVIEW einen einfachen UDP Receiver entwickelt, um zu sehen wie das genau Funktioniert. Diesem habe ich dann mit der OSC App Nachrichten gesendet und den empfangenen String angeschaut.

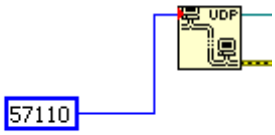


Abb. 4.3.1.2

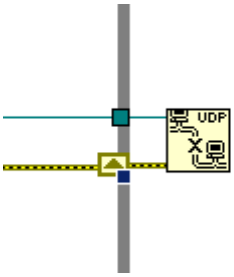


Abb. 4.3.1.3

Um ein UDP Packet zu empfangen, muss zuerst ein UDP Socket mit einer Portnummer Initialisiert werden. Dies geschieht mit dem **„UDP Open“** Block (Abb. 4.3.1.2). Am Ausgang des Blocks kann man dann die UDP Session und den Error abgreifen. Beides muss am Ende des Programmes wieder mittels **„UDP Close“** Block (Abb. 4.3.1.3) geschlossen werden.

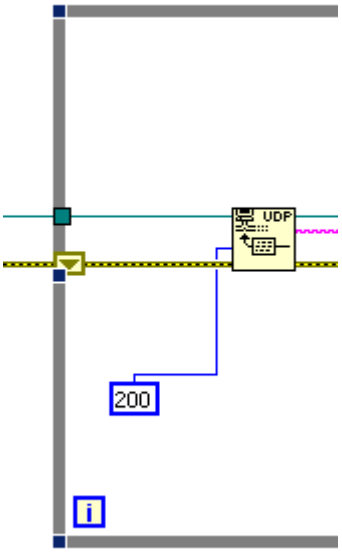


Abb. 4.3.1.4

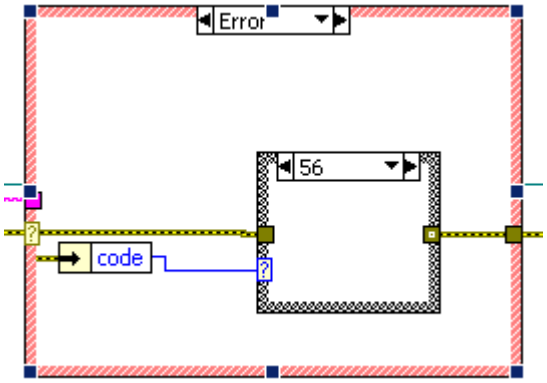


Abb. 4.3.1.5

Zwischen den Blöcken fürs öffnen und schliessen kann dann der UDP Buffer ausgelesen werden. Dies geschieht am besten in einer endlos Schleife. Der **„UDP Read“** Block (Abb. 4.3.1.4) zum lesen des Buffers benötigt eine Timeout Zeit. Da sich der ganze Vorgang in einer Schlaufe befindet, kann diese Zeit sehr kurz gewählt werden.

Wichtig ist allerdings das nach dem Lesen des Buffers, der Fehler 56 (Abb. 4.3.1.5) abgefangen wird. Dieser bedeutet ein Timeout beim Lesen. D.H. innerhalb der Timeout Zeit sind keine Daten im UDP Buffer eingegangen. Dies ist aber kein Problem. Die Schleife soll einfach noch einmal von vorne beginnen, solange bis Daten ankommen. Darum muss der Fehler zwingend abgefangen und behandelt werden.

4.3.2 OSC Protokoll

Open Sound Control (OSC) ist ein nachrichtenbasiertes Kommunikationsprotokoll, welches hauptsächlich für die Echtzeitverarbeitung von Sound über Netze und Multimedia-Installationen verwendet wird.

Steuernachrichten können von Hardware (z. B. MIDI-Keyboard) oder Software (z. B. Processing, Vvvv, Csound, Max/MSP, Pure Data, SuperCollider, ChucK, EyesWeb) erzeugt und dann via OSC in Form von sog. Nachrichten (OSC-Messages), welche wiederum in Bündel (OSC-Bundles) verpackt werden, an eine Schnittstelle weitergegeben werden und so eine Ausgabe steuern. Dieses können z. B. weitere Soundausgaben sein, etwa eine Soundanwendung auf einem anderen Computer.
[Wikipedia, http://de.wikipedia.org/wiki/Open_Sound_Control]

Eine OSC Message ist folgendermassen aufgebaut:

/Address/of/Control „f“ 157.23f

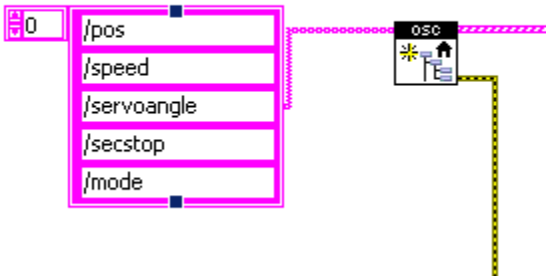
Zuerst kommt die Adresse des Controls. Diese kann auch mehrere mit / getrennte Teile enthalten, wenn das Control z.B. mehrere Elemente besitzt. Danach kommt der Datentyp und zum Schluss der Wert der übertragen werden soll.

Es sind folgende Datentypen erlaubt:

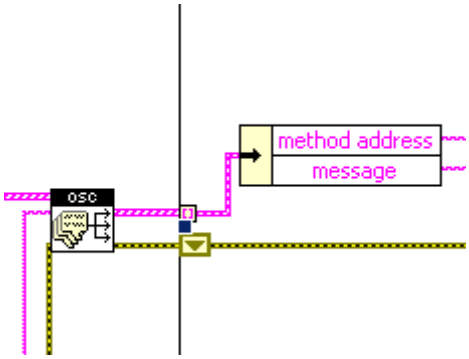
Int32, float32, OSCString, OSCTimetag, OSCBlob

Um das OSC Protokoll in LabVIEW verwenden zu können, muss man die OSC Library installieren. (Ressourcen/Software/national_instruments_lib_ni_osc-1.0.1.5.vip) Danach stehen einem die OSC Blöcke zur Verfügung.

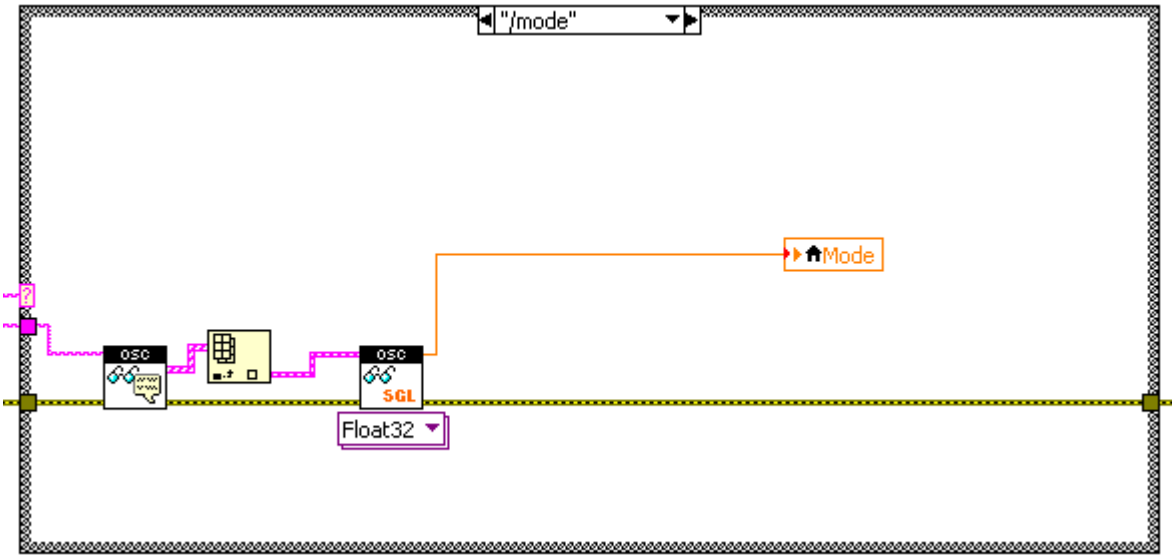
4.3.2.1 OSC Messages Empfangen



Als erstes wird mit dem **„Create Address Space“** Block der OSC Address Stack initialisiert. Aufgrund dieser Adressen werden dann die Messages mit dem **„Dispatch All Messages“**

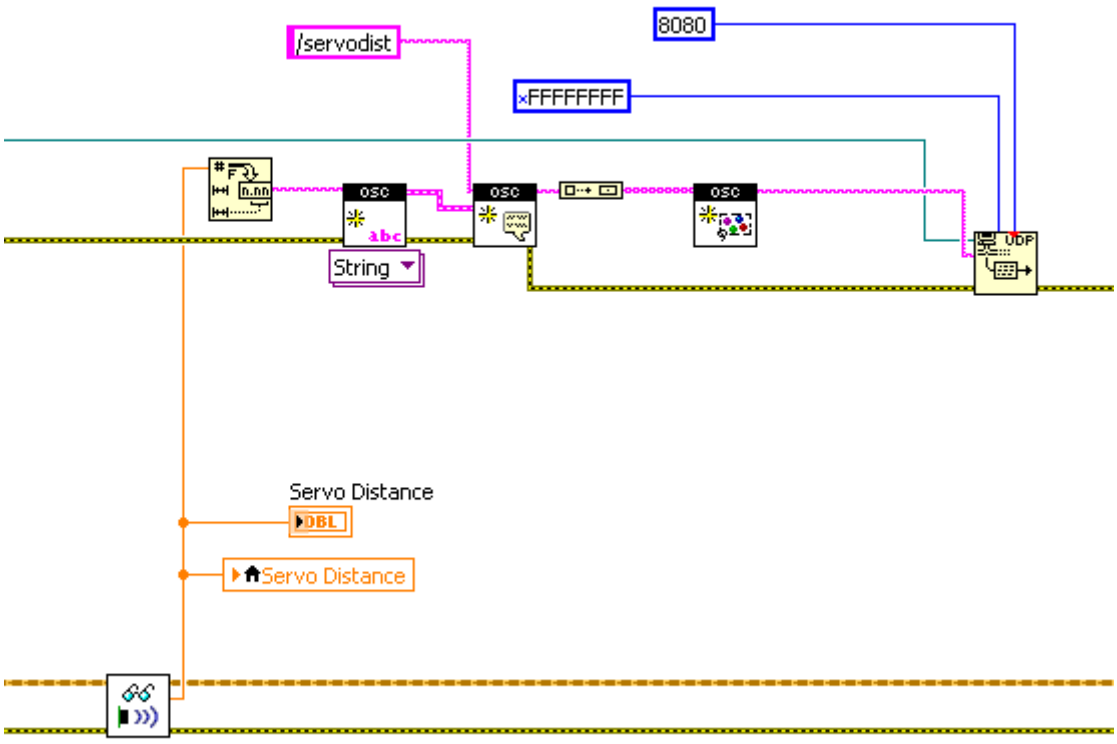


Block im Message Cluster identifiziert und ausgelesen.
Um die Message zu verarbeiten, braucht man eine Case Struktur die jede bekannte Control Adresse behandelt. Im jeweiligen Case kann Anschliessend die einzelne Message mit dem **„Parse Message“** Block geparst und die übermittelten Werte in ihren zugehörigen Datentypen konvertiert werden. (**„Argument to [Datatype] Data“** Block)



4.3.2.2 OSC Messages Senden

Das Senden von OSC Messages funktioniert genau umgekehrt zum empfangen. Dabei werden zuerst die Daten in einen OSC String umgewandelt und dieser wird dann mit der Adresse des Controls zusammen in eine OSC Message verpackt. Diese Message wird zum Schluss in einen Message Cluster gehüllt und an den **„UDP Send“** Block übergeben.



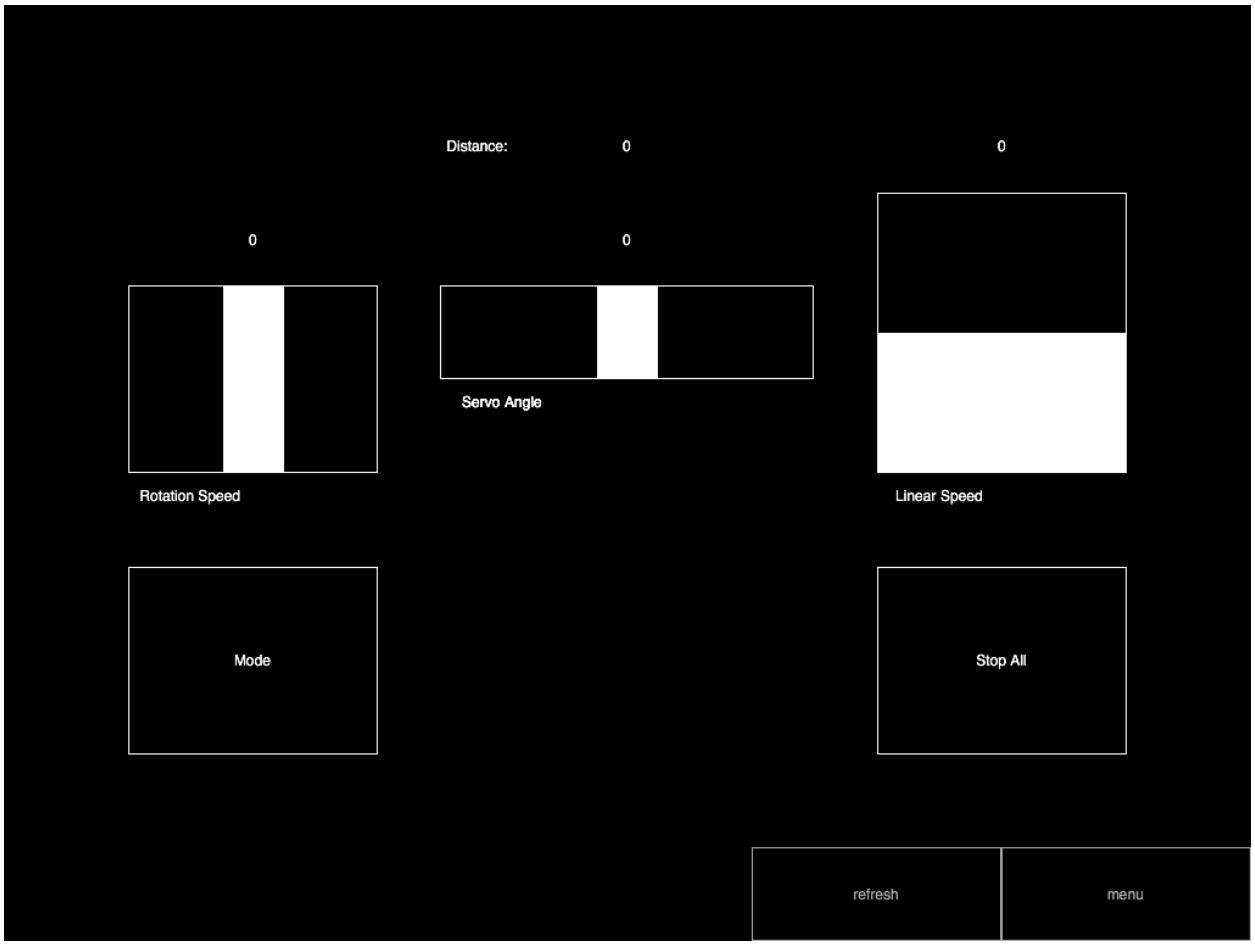
Hier verwende ich die UDP Broadcast Adresse, damit die Sensor Daten an alle verbundenen Geräte gesendet werden.

4.3.3 iPad/iPhone App

Um den Roboter mittels OSC vom iPad aus zu steuern, verwendete ich die App „Control (OSC + MIDI)“ die gratis im Appstore von Apple sowie auch im Google Play Store zum Herunterladen verfügbar ist.

Mit dieser App ist es möglich selber GUIs für OSC Applikationen zu entwickeln.

Das von mir entwickelte GUI für die VI ipad_drive_full.vi findet man unter Ressourcen/OSC/oscGUIfull.js auf der CD.



Dieses GUI File muss man dann auf einen Webserver hochladen und in der App unter „Interfaces“ über den Plus (+) Button die Web Adresse eingeben. Danach wird das GUI in der App installiert und erscheint in der Liste der verfügbaren Interfaces.

Zusätzlich müssen in der App noch unter „Destinations“, über den Plus (+) Button die Netzwerkdaten des Roboters eingeben werden. Erst dann können die Messages auch an den richtigen Empfänger geschickt werden.

Folgende Daten sind einzugeben:

Destination URL: 147.87.172.82
Destination Port: 57110

4.3.4 GUI selbst entwickeln

Das GUI kann auch angepasst oder ein neues GUI für andere Zwecke entwickelt werden. Die GUI Definitionen sind im JSON Format geschrieben und werden dann von der Control (OSC + MIDI) App grafisch dargestellt.

Beispiel für einen einfachen Button (Mode Auswahl Button, oscGUIfull.js)

```
{
    "name" : "mode",
    "type" : "Button",
    "x" : 0.1, "y" : 0.6,
    "width" : 0.2, "height" : 0.2,
    "mode" : "toggle",
    "min":1, "max":2,
}
```

Alle weiteren Informationen zur GUI Entwicklung finden sich auf der Webseite des Herstellers.
http://charlie-roberts.com/Control/

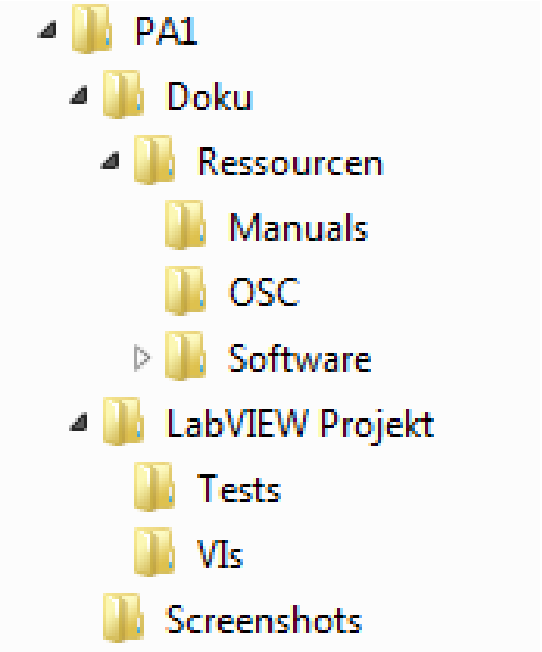
5 Abschluss

In dieser Arbeit ist es mir gelungen, die Grundlegenden LabVIEW Kenntnisse die ich während meines Studiums erlangte, einzusetzen und auszubauen. Ich habe während 6 Wochen die Gelegenheit gehabt, dass NI Robotics Developer Kit 2.0 kennenzulernen und mit dem Roboter einige einfache Handling Aufgaben zu entwickeln. Einiges davon ist mir gelungen wie z.B. das Fahren von Kurven oder die manuelle Steuerung des Roboters. Auch habe ich ein wenig mit der mitgelieferten Demo Software experimentiert und verstehe nun ziemlich gut wie diese Funktioniert. Andere Ideen die ich hatte konnte ich leider innerhalb dieser kurzen Zeit nicht umsetzen. Z.B. hätte ich gern noch ein Vision System implementiert, mit dem der Roboter etwa ein Objekt hätte verfolgen können. Zum Schluss der Arbeit ist es mir gelungen eine Verbindung zum einem iPad herzustellen und den Roboter damit vollständig sowohl manuell als auch im automatischen Modus fernzusteuern.

Zum weiteren Vorgehen gibt es natürlich unzählige Möglichkeiten. Beispielsweise wäre es denkbar die Single Board RIO Plattform für den Eurobot Wettbewerb einzusetzen. Dazu müsste man aber eine andere Hardware Plattform entwickeln oder z.B. die M-Bot Plattform übernehmen. Um tiefer in LabVIEW Robotics eintauchen zu können, sind sicherlich auch weitere Sensoren nötig die Umgebungsparameter bestimmen können. Auch denkbar wäre die Entwicklung eines absoluten Positionierungssystems um auf dem Eurobot Tisch navigieren zu können.

Zum Abschluss kann ich sagen das ich mit dieser Projektarbeit viel Spass hatte und einen sehr guten Einblick in die LabVIEW Robotics Technologie bekommen haben.

Anhang A Übersicht Projektverzeichniss (auf CD)



Mauals

NI Robotics Development Kit
Getting Started.pdf
NI_sbRIO-961x_963x_964x_Quick_Reference.pdf
NI_sbRIO-961x_963x_964x_and_NI_sbRIO-9612XT_9632XT_9642XT_User_Guide.pdf

Vision System Manuals
NI-IMAQdx_User_Manual.pdf
205_qig_en_21121.pdf

OSC

oscGUI8.js
oscGUIfull.js

Software

vipm-windows.exe
national_instruments_lib_ni_osc-1.0.1.5.vip

uEye_LabVIEW_40000.exe
AXISMediaControlSDK_6_03.zip

Anhang B Weblinks

Technische Daten des Roboters
http://sine.ni.com/ds/app/doc/p/id/ds-217/lang/de

Robotics Development Kit 2.0
http://sine.ni.com/nips/cds/view/p/lang/de/nid/208010

LabVIEW Community Zone mit Code Beispielen und Diskussionsforen
https://decibel.ni.com/content/community/zone/labviewrobotics

Anhang C Arbeitsjournal

Arbeitsjournal PA 1 Robotics Starter Kit

Datum	Aktion / Entscheidung / Information
W1	
	Einrichten des Arbeitsplatzes
	Lesen der Projektbeschreibung
07.05.2012	Kennenlernen des Robotics Startekits
08.05.2012	Career Day Biel
	Herausfinden der IP des Roboters
	Inbetriebnahme mit Starterkit Demo VI
09.05.2012	Herumfahren im Gang
	Aufbau einer realen Roboter Umgebung mit Kabelnachführung
10.05.2012	Experimentieren mit Motorsteuerung und Ultraschall Sensor
	Automatisches Vor und Rückwärtsfahren nach Distanz
11.05.2012	Suche nach weiteren Handling Aufgaben für PA1
W2	
	Demonstration des Erreichten mit Herrn Lanz
	Besprechen der weiteren Ideen
	Entscheidung für Vision System mit IP Kamera
14.05.2012	und ev. iPad Steuerung
15.05.2012	Inbetriebnahme und Experimentieren mit D-Link DCS-2121
	Aufgabe nach erfolglosem Versuch D-Link mit LabView anzusteuern
16.05.2012	Neuer versuch mit uEye USB Kamera
	uEye Kamera erfolgreich mit LabVIEW ausgelesen
	uEye verworfen weil Roboter kein USB port besitzt
	Experimentieren mit NI Vision Kamera
	Image Aquisition mit Vision Assistant
17.05.2012	Erkennen von Kanten und Partikeln
	Steering Frame kennengelernt
	Kurvenfahren mit Arc Steering Frame implementiert
	Weitere Experimente mit NI Vision Starter Kit
18.05.2012	Leider kein Erfolg mit eigener VI
W3	

	Demonstration des Erreichten
	Kurvenfahren mit dem Roboter und Vision Builder Assistant
21.05.2012	iPad Steuerung angefangen
22.05.2012	Geschäftlich abwesend
	iPad LabVIEW Verbindung über OSC und UDP geschafft
23.05.2012	Übertragung von Werten und XY Koordinaten
	Steuerung des Roboters mit iPad gelungen
	Verbesserung der Steuer GUI für iPad
24.05.2012	Experimente mit Rückführung von werten von Roboter zum iPad
	Wireless AP bestellt
	Weitere Experimente mit NI Vision Starter Kit
	Eigene VI für Vision Starter Kit geschafft
25.05.2012	Vision Starter Kit ist eigenes RT Target

W4

28.05.2012	Pfingstmontag
29.05.2012	Dokumentation
	Axis Kamera mittels ActiveX Komponente zum Laufen gebracht
30.05.2012	Erste Experimente mit Bild Verarbeitung
31.05.2012	Abwesend aus geschäftlichen Gründen
	Auspacken, aufbauen und einrichten des neuen WLAN APs
	Erste WLAN Tests
	Ausmessen und Löten eins power Adapters für den WLAN AP
01.06.2012	Erste Testfahrt mit WLAN funktioniert!

W5

04.06.2012	Abwesend geschäftlich
05-08.06.2012	Abwesend durch Krankheit

W6

	Beginn Zusammenbau aller entwickelten Funktionen
11.06.2012	zu einer grossen Applikation
12.06.2012	Zusammenbau fortgefahren
	Zusammenbau abgeschlossen
	Programm Test
13.06.2012	GUI Verbesserungen
	Lezte Tests
14.06.2012	Dokumentation
15.06.2012	Dokumentation

W7

	Vorführung und Besprechung mit Hern Schib von NI
18.06.2012	Dokumentation
19.06.2012	Dokumentation Zuhause
20.06.2012	Dokumentation
21.06.2012	Dokumentation Zuhause
	Dokumentation Fertigstellen
	Dokumentation Drucken und CD Brennen
22.06.2012	Abgabe der Projektarbeit 1

Erklärung des Studenten

Rechte an der Projektarbeit

Ich bin damit einverstanden, dass die BFH-TI alle Güter inklusive der Immaterialgüter der durch die BFH-TI geleiteten Arbeiten grundsätzlich nutzen darf. Besondere Abmachungen zwischen Dozenten und beteiligten Unternehmungen und Institutionen bleiben vorbehalten.

Selbständige Arbeit

Ich bestätige mit meiner Unterschrift, dass ich meine Projektarbeit 1 selbständig durchgeführt habe. Alle Informationsquellen (Fachliteratur, Besprechungen mit Fachleuten, usw.), die wesentlich zu meiner Arbeit beigetragen haben, sind in meinem Arbeitsbericht im Anhang vollständig aufgeführt.

Datum	Freitag, 22. Juni 2012
Name Vorname	Pletscher Martin
Unterschrift	.....