Weird Science Club Darmstadt – Jugend Forscht 2007 – Lichtenbergschule Darmstadt

Bau eines Kugellabyrinth-Roboters

Oliver Lott, Thomas Hiemstra, Roland Wirth

Inhaltsverzeichnis

Abbildungsverzeichnis

Bau eines Kugellabyrinth-Roboters

Einleitung

Als wir hörten, dass es an unserer Schule den Weird Science Club gibt, der an Wettbewerben teilnimmt entschieden wir uns, auch ein Projekt zu machen. Nachdem wir uns über die Richtung unseres Projektes einig waren, trafen wir uns auf Empfehlung unseres Lehrers Dr. Dlabal mit Professor Adamy vom Institut für Regelungstheorie und Robotik der TU-Darmstadt.

Hier fingen wir an, uns mit möglichen Projekten auseinander zu setzen. Unser erster Entschluss war, einen Tippkick-Roboter zu bauen. Dieser Roboter sollte den Torwart und den Feldspieler steuern. Zur Umsetzung hatten wir mehrere Ideen. Zuerst überlegten wir, einen kleinen Industrieroboter zu verwenden, für den wir einen passenden Aufsatz bauen und ihn mit einem Programm steuern könnten. Allerdings war diese Idee erstens zu teuer und zweitens ist die Programmierung eines solchen Roboterarmes, der die nötige Präzision besitzt, sehr schwierig. So war uns klar, dass wir selbst einen Roboter bauen mussten.

Als einfachste und billigste Lösung erschien uns hier ein Brückenkran. So fingen wir an, für diesen Entwurf technische Zeichnungen anzufertigen, um eine verbindliche Vorlage zu schaffen und mögliche Probleme schon im Voraus zu erkennen. Nach einiger Zeit merkten wir, dass unsere Idee sehr aufwändig und zeitintensiv war. So fingen wir an, das Projekt herunterzubrechen und konzentrierten uns nur noch auf den Feldspieler. Nach dem wir hier noch immer auf sehr viele Probleme stießen, wie zum Beispiel beim Schussmechanismus und ob der Feldspieler fest mit dem Kran verbunden sein sollte oder nicht, entschieden wir uns gegen dieses Projekt und suchten ein neues, das in unserem zeitlichen Rahmen zu lösen war.

Nach einigen Suchen im Internet interessierte uns am meisten ein Projekt der IESE (Fraunhofer Institut für Experimentelles Software Engineering), die mit einem Kugellabyrinthroboter, der den Spieler unterstützt, eine Echtzeitvariante des Betriebssystems Linux vorstellte. Da dieser Roboter allerdings das Spiel nicht alleine lösen konnte und selbst für die bloße Unterstützung des Spielers sehr aufwändige Hardware benötigte – eine Kamera aus dem Forschungsbereich, die einen kleinen Bereich ihres Blickfeldes bis zu 300 Mal pro Sekunde abtastet –, setzten wir uns das Ziel, einen

Roboter zu bauen und zu programmieren, der ein solches Kugellabyrinth völlig selbstständig löst und dabei nicht mehr als eine hochwertige Webcam benötigt.

Dieser Roboter muss das Kugelspiel steuern können und gleichzeitig in der Lage sein, zu sehen wie die Veränderung des Spielfeldes die Kugel beeinflusst. Er besteht aus zwei Motoren, die die zwei Achsen steuern, einer Webcam zur Kontrolle und einem Computer, der die notwendigen Reaktionen der Motoren aus den Informationen der Webcam mithilfe eines Modells berechnet. Zusätzlich stellt sich das Problem der Reaktionszeit des Roboters. Der Roboter befindet sich zwischen zwei Bildern der Webcam in einer blinden Phase. Die hier auftretenden Fehler, z.B. Ungenauigkeiten in der Regelung der Kugelbewegung, die dazu führen können, dass die Kugel in ein Loch des Spielfeldes fällt, müssen vermieden werden. Um die Bildrate und damit die Zeit ohne Positionsdaten nicht noch weiter zu verlängern, muss unser Programm die Daten so schnell wie möglich in Steuerbefehle umsetzen, was aber bei den Geschwindigkeiten heutiger Computer kein allzu großes Problem mehr darstellt. Zusätzlich erstellen wir ein Modell der Kugel, das die Reaktionen der Kugel aus den Kameradaten und den Steuerungsbefehlen der Regelung vorhersagt. Um die benötigten Reaktionen zu veranlassen, verarbeiten wir die von der Webcam aufgenommenen Bilder so, dass wir die Position der Kugel und die Lage des Spielfelds bestimmen können. Weil wir die Bahn der Kugel nicht direkt steuern können, verwenden wir einen PID-Regler (**P**roportional, **I**ntegrierend, **D**ifferenzierend), um die Kugel den gewünschten Weg entlangrollen zu lassen.. Aus der Lage der Kugel im vorherigen und jetzigen Bild kann die Geschwindigkeit, Beschleunigung und Bewegungsrichtung berechnet werden werden. Zusätzlich kann man aus dem Kamerabild auf die Grenzen des Spielfelds schließen. Aus diesen Werten bestimmt unsere Regelung nun die neue Auslenkung des Labyrinths und gibt die Befehle hierzu an die Motoren weiter.

Ideenfindung

Im Dezember 2005 entschieden wir uns dazu den Tippkickroboter zu bauen und fertigten Zeichnungen ab Januar 2006 an. Mitarbeiter des Instituts halfen uns dabei, die Richtigkeit der Zeichnungen sicher zu stellen und sorgten dafür, dass alle Details berücksichtigt werden. Ende Mai waren wir dann so weit eine Bestellliste der benötigten Teile aufzustellen. Ende Juni merkten wir, nachdem wir schon bei den Zeichnungen das Projekt vereinfacht hatten, dass das Projekt noch immer zu aufwändig war und wir machten uns Gedanken über ein anderes Projekt.

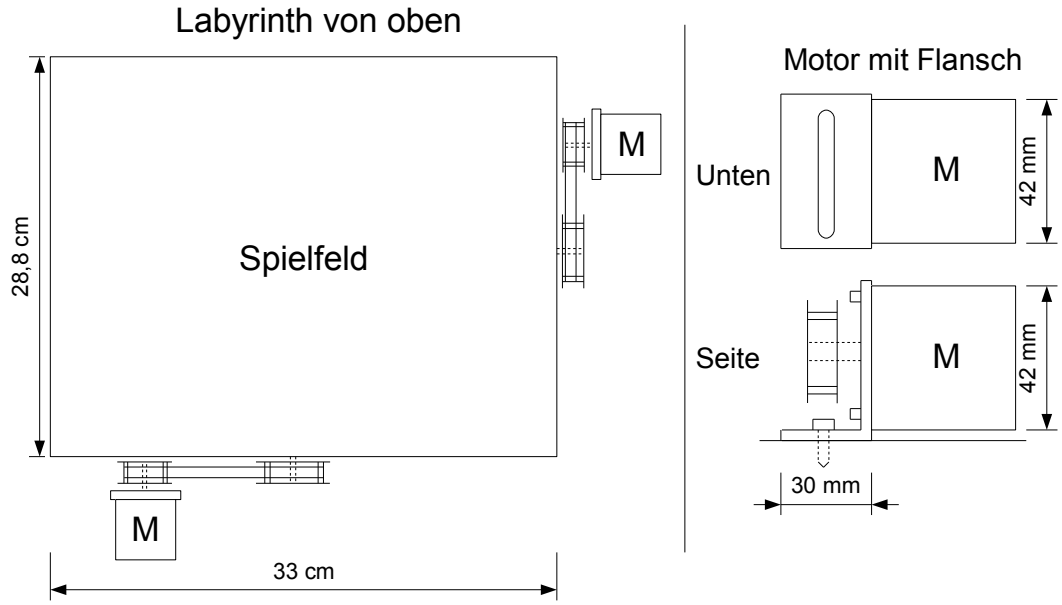

Abbildung 1 - Skizze des Aufbaus

So entstand die Idee, ein Kugellabyrinth durch einen Roboter lösen zu lassen. Wir fingen sofort an Zeichnungen zu erstellen und überlegten uns, dass das Spiel durch zwei Schrittmotoren an den Achsen bewegt werden soll. Zusätzlich sollte eine Kamera über dem Spielfeld die nötigen Positionsdaten an einen Computer liefern. Eine Bildverarbeitung und eine PID-Regelung auf dem Computer berechnen aus den Kameradaten die zur Regelung notwendigen Informationen und senden die entsprechenden Befehle an die Motoren.

Im Oktober war unsere Planung dann soweit, dass wir Teile bestellen konnten:

- 4 Zahnriemenscheiben
- 2 Zahnriemen
- 2 PANdrive PD3-110-42-232

Die Webcam – eine Apple iSight –, eine Firewirekarte, sowie eine Holzplatte als „Baugrund“ konnten wir stellen. Das Stativmaterial zur Befestigung der Kamera bekamen wir von der Lichtenbergschule. Ein passendes Labornetzgerät und einen Scheinwerfer zur Beleuchtung stellte uns das Institut zur Verfügung.

Umsetzung

Der Versuchsaufbau

Mit der Hilfe der Werkstatt des Institutes, die uns bei der Verarbeitung des Metalls half, montierten wir das Spiel auf eine Pressspanplatte und befestigten die Motoren und die Zahnscheiben. Hierzu mussten wir mehrere Winkel für das Spielfeld so wie die Motorenflansche herstellen. Zusätzlich befestigten wir die Zahnriemenscheiben an den Spielachsen sowie den Motoren. Wir entschieden uns für einen Zahnriemen, da dieser ein sehr geringes Spiel zwischen Motor und Achse lässt. Zahnräder würden durch das Spiel zwischen ihren Zähnen nur eine ungenaue Ansteuerung zulassen. Außerdem hätten die Motoren direkt unter oder neben den Achsen montiert werden müssen, was größere mechanische Probleme beim Aufbau verursacht hätte. Die Kamera konnten wir mit zwei Stativen und einer Aluminiumplatte, die wir uns zurechtfräsen ließen, über das Labyrinth hängen. Ein Vorteil der Apple iSight ist, dass sie mit einem Klemmfuß geliefert wird, damit sie an einem MacBook befestigt werden kann. Diese Klemme konnten wir nutzen, um die Kamera an der Aluminiumplatte zu befestigen. Die Kamera ist – anders als die meisten Webcams – nicht über USB 1.1 sondern über FireWire angeschlossen, was durch die höhere Datenrate erlaubt, Bilder unkomprimiert zum Computer zu übertragen. Außerdem können FireWire-Kameras über ein Standardprofil angesprochen werden; man benötigt also im Gegensatz zu USB-Webcams keine zusätzlichen Treiber.

Die Bewegung des Spielfeldes

Hiernach konnten wir uns an die Programmierung des Computers machen. Die Verbindung zu den Motoren wird über die serielle Schnittstelle des PCs hergestellt und ein Programm gibt den Motoren über ein spezielles Protokoll, die Trinamic Motion Control Language (TMCL), Anweisungen, wie z.B. Bewegungsbefehle oder Änderungen der Parameter der Motorenregelung. Die Motoren, die wir ausgewählt haben, besitzen einen eigenen Mikrocontroller, der alle Größen, die nötig sind, um den Motor eine bestimmte Aktion ausführen zu lassen, selbstständig berechnet. Daher können wir ihm direkte Befehle geben, wie zum

Abbildung 2 - Motor mit Zahnriemen an einer Achse des Spielfeldes

Beispiel eine Position an die der Motor fahren soll, und müssen nicht selbst noch eine aufwändige Motorenregelung entwickeln.

Die Erkennung von Spielfeld und Ball

Die Kamerabilder verarbeiten wir mit Hilfe der OpenCV-Bibliothek, die viele nützliche Funktionen zur Bildverarbeitung bereitstellt, z.B. Farbraumtransformationen, Filterfunktionen oder Entzerrer. Diese von Intel entwickelte Bibliothek ist in C geschrieben und quellcodeoffen, wodurch wir in der Lage sind, unsere C++ Programme mit ihr zu entwickeln.

Zunächst versuchten wir den Ball über die Veränderung von einem auf das nächste Bild zu ermitteln (der Ball sollte das einzige Objekt sein, das sich merklich bewegt). Das scheiterte daran, dass durch Bildrauschen immer Bewegung im Bild war und auch das Neigen des Spielfeldes große Veränderungen in das Bild brachte. Ein anderer Versuch, den metallisch glänzenden Ball über die Farbe zu erkennen, scheiterte an Reflexionen von Lichtquellen und daran, dass er sich als graues Objekt bei nicht so guter Beleuchtung nicht sehr stark von seinem Untergrund abhob. Um diese Fehler zu minimieren, installierten wir über dem Spielfeld eine starke Lichtquelle, die andere Quellen überstrahlen soll, was das Bildrauschen merklich verringerte, aber dennoch nicht den gewünschten Effekt lieferte. Auch waren weiterhin Spiegelungen auf dem Ball zu verzeichnen. Würde man eine qualitativ höherwertige Kamera verwenden, die weniger Rauschen und mehr Farbechtheit liefern würde, dann wären diese Lösungsansätze vielleicht erfolgreich gewesen. Auch eine höhere Bildrate und ein vorheriges Aussortieren vieler Bildpunkte durch die Kamera (eine Region-of-Interest-Funktion) würde die Aufgabe des Regelns erheblich vereinfachen, aber diese Wünsche lassen sich mit keiner Webcam erfüllen – man bräuchte dafür eine Kamera aus dem wissenschaftlichen Bereich. Da wir uns aber auf eine Lösung

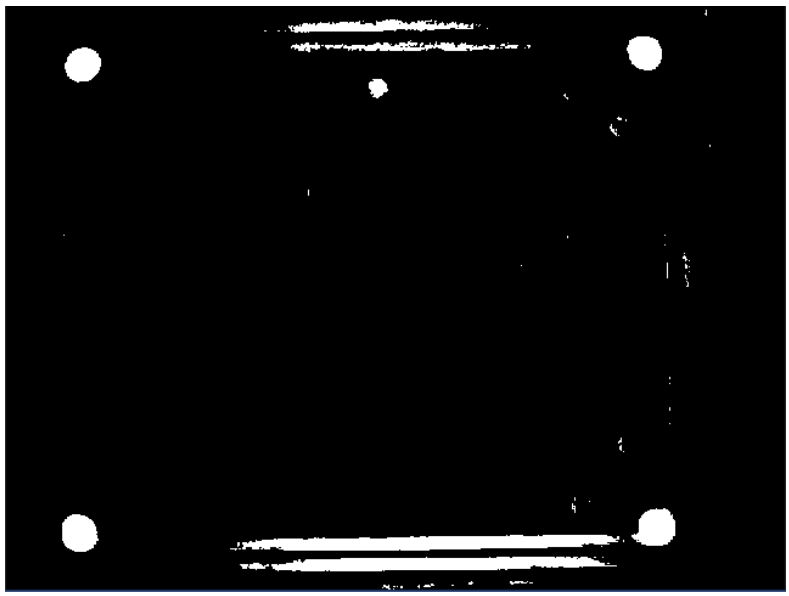

Abbildung 3 - Sättigungskanal des Spielfeldes nach der HSL-Transformation (mit Schwellwert)

konzentrierten, die mit einer Webcam funktionieren sollte, versuchten wir eine andere Methode zu finden.

Wir färbten die Kugel mit der Signalfarbe Orange ein und brachten zur Besseren Erkennung orangefarbene Klebepads auf den Ecken des Spielfeldes an. Nun unterschieden sich die Spielfeldecken und der Ball deutlich vom Rest des von der Kamera erfassten Bildes. So waren wir in der Lage die Koordinaten der Eckpunkte des Spielfelds im Bild zu ermitteln. Diese benötigten wir, um die absolut gemessene Position des Balles in das Bezugssystem des Spielfeldes einordnen zu können. Wenn das Spielfeld geneigt ist, wird das rechteckige Feld zu einem von der Neigung abhängigen Trapez verzerrt. Deswegen müssen die gemessenen Koordinaten der Eckpunkte über eine Transformation wieder zu einem Rechteck entzerrt werden, weil sonst die Position des Balles nicht mehr akkurat ermittelt werden kann. Diese Entzerrung nimmt eine Funktion der OpenCV-Bibliothek vor. Die errechnete Position des Balles im Bezug auf die Ecken des Spielfeldes wird nun an den PID-Regler übergeben.

Um das Bild für uns leichter handhabbar zu machen, transformierten wir dessen Farbinformationen zuerst in den HSL-Farbraum. Der HSL-Farbraum wird als Doppelkegel (siehe Abbildung 4) dargestellt – im Gegensatz zum geläufigen RGB-Farbraum, der eine Art Würfel ist –, in dessen Mittelpunkt ein neutrales Grau liegt. Über die drei Parameter „Farbwinkel“ (**H**ue), „**S**ättigung“ und „**L**uminanz“ kann man nun jeden Punkt innerhalb dieses Farbraums erreichen. Diese Methode der Farbbestimmung ist deutlich intuitiver als die über den RGB-Raum, weil sie mehr der menschlichen Wahrnehmung entspricht.

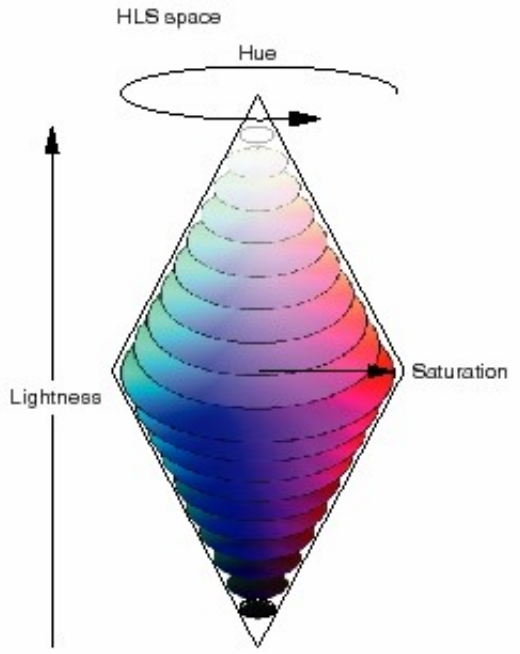

Abbildung 4 - Der HSL-Farbraum im Modell als Doppelkegel

Aus dem RGB- kann man über relativ einfache mathematische Operationen in den HSL-Farbraum konvertieren. So können wir das Bild nach Farben und Sättigung filtern, um andersfarbige Objekte vor Hintergrund zu erkennen.

Nun sticht das Orange aus dem Bild heraus, da der Sättigungskanal hier ein deutliches Maximum zeigt (siehe Abbildung 3). Eine

darauf folgende Schwarz-Weiß Konvertierung liefert dann eine einfacher weiter zu verarbeitende Grafik, in der nun die verschiedenen Objekte (zusammehängende weiße Flächen) markiert werden. Von jedem dieser markierten Objekte wird nun eine Struktur mit dessen Eigenschaften „Schwerpunkt“, „Fläche“ und „einrahmendes Rechteck“ angelegt. Nun werden die fünf Objekte heraus gesucht, von denen vier ein Trapez bilden (Eckpunkte) und der Fünfte im Trapez liegt (Ball), und deren Koordinaten entzerrt und an den Regelmechanismus weitergeleitet.

Die Regelung der Kugelbewegung

Ein PID-Regler besteht generell aus drei unterschiedlich gewichteten Komponenten: einer proportionalen, einer differenzierenden und einer integrierenden. Als Basis dient die Differenz des Ist- mit einem festgelegten Sollwert. Der proportionale Anteil steigt mit steigender Ist-Soll-Differenz, der differenzierende Anteil ist abhängig von der Änderung der Ist-Soll-Differenz und der integrierende Anteil ändert sich mit der Gesamtänderung der Ist-Soll-Differenz. Die gewichtete Addition der drei Komponenten bestimmt den neuen Ausschlag der Motoren, wenn man die Ist-Position der Kugel und den nächsten Wegpunkt, den sie anfahren soll, als Eingangsgrößen angibt.

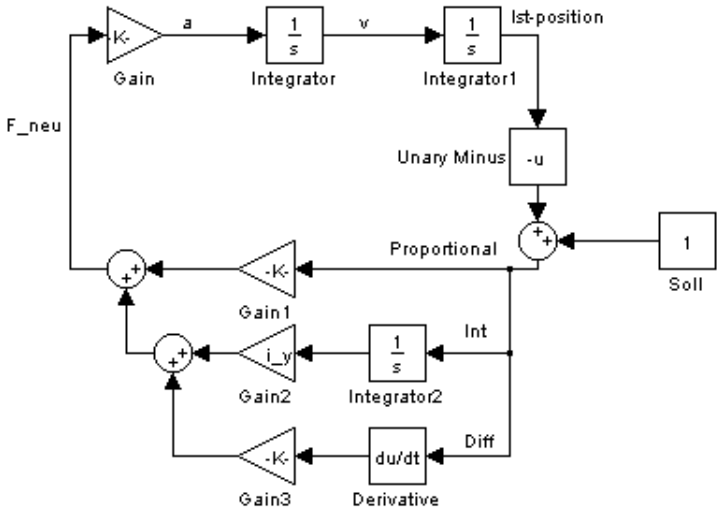

Abbildung 5 - PID-Regler mit Modell einer beschleunigten Masse in Simulink

Um erste Erfahrungen mit dem PID-Regler zu sammeln, konstruierten wir ein Modell der Kugel mit der zu Matlab gehörenden Software Simulink und regelten dieses Kugelmodell mithilfe zweier PID-Regler (einer für jede Achse), die wir selbst parametrisierten, so, dass es verschiedenen Bahnen folgte oder sich zu bestimmten Punkten bewegte. Diese Regler funktionierten sogar noch, nachdem wir in den Regelkreis ein Verzögerungsglied zwischengeschaltet hatten. Dabei ist uns aufgefallen, dass die PID-Regler teils sehr große Regelausschläge produzieren. Da wir unser Spielfeld nicht unbegrenzt weit auslenken können, mussten wir den Regelausschlag begrenzen. Dies hatte glücklicherweise in der Simulink-Simulation keine nennenswerten Auswirkungen, solange die Gewichtung der integrativen Komponente

nicht zu klein war; diese glich den durch die Begrenzung erzeugten Fehler wieder aus.

In Abbildung 5 sieht man den Beispielaufbau eines PID-Reglers, der die Kraft regelt, die auf das Modell einer beschleunigten Masse wirkt. Dieses Modell befindet sich im oberen Teil. Man kann rechts gut erkennen, wie die Differenz zwischen Ist- und Sollposition gebildet und an das Porportional-, Integral- und Differentialglied weitergeleitet wird. Deren Ausgaben werden mit einem Gewichtungsfaktior multipliziert und Aufaddiert. Diese Summe ist die neue auf das Modell einwirkende Kraft.

Weitere Probleme

Ein großes Problem bei unserer Steuerung ist die geringe Bildwiederholrate der Kamera, durch die lange Zeitspannen (so genannte Totzeiten) entstehen, während derer uns keine Informationen über die Position des Balles zur Verfügung stehen. Diese Zeiten versuchen wir mit einem physikalischen Modell der Kugel auszugleichen, indem wir aus den gegebenen Daten die aktuelle Position des Balles extrapolieren. Das Modell soll sich in einer späteren Ausbaustufe selbst durch Erkennen des gemachten Fehlers korrigieren können. Aus den theoretischen 15 Bildern pro Sekunde wurden in der Praxis nach der HSL-Transformation ca. 8, obwohl der Kameraprozess gekapselt auf der zweiten CPU ausgeführt wird. Dieser Umstand zwingt uns dazu, dass unser PID-Regler nur alle 125ms neue Positionsdaten zum Vergleich mit den bisherigen Daten bekommt, welche auch noch hinter dem wirklichen Zustand hinterherhinken.

Ein Gamepad samt Software kann zur Lösung eines weiteren Problems genutzt werden: Zur Einstellung des Nullpunktes der Motoren. Dieser bezeichnet die Position, in der das Spielfeld absolut waagrecht ist. Manuell wird nun das Spielfeld in die Waagrechte gefahren und durch einen Knopfdruck der Nullpunkt gesetzt. Nur durch die korrekte Kalibrierung des Spielfeldes kann der Regler gut arbeiten.

Test des Systems

Als ersten Test unserer Motorensteuerung programmierten wir ein Programm, das die Positionsdaten eines Joysticks – in unserem Falle die Positionen der Analog-Sticks eines Gamepads – auswertet und diese in Steuerdaten für die Motoren umsetzt. So kann man das Kugellabyrinth über den Computer aber trotzdem manuell steuern.

Auch Bilder konnten wir bereits von der Kamera bekommen und in einem Testprogramm für die ersten Schritte der Bildverarbeitung und Objekterkennung einsetzen.

Danksagungen

Danken möchten wir Herrn Professor Adamy, der uns tatkräftig unterstützte und ohne dessen finanzielle Unterstützung dieses Projekt nicht möglich gewesen wäre. Ebenfalls danken wir Herrn Dipl-Ing. Christian Voigt und Frau Dipl-Ing. Anna Flemming vom Institut für Regelungstheorie und Robotik für ihre für uns geopferte Zeit, die hilfreichen Ratschläge, die Treffen, die uns immer einen Schritt näher zum Ziel gebracht haben, und dafür, dass sie uns jederzeit für Fragen zur Verfügung standen.

Dank auch an Herrn Alexander Stark und sein Team für die große Hilfe beim Aufbau des Roboters und für die vielen Erklärungen und Anleitungen zu den Maschinen in der Werkstatt und zur Metallverarbeitung.

Zuletzt möchten wir auch unserem Physiklehrer Herrn Dr. Milan Dlabal danken, der uns die nötigen Anstöße und Kontakte verschafft hat und ohne den das Projekt nicht so gut hätte laufen können.