

LEGO® Fußball Roboter Wettbewerb

Teilnehmende Gruppe des TG Rastatt:

„Little Big Football“

Schüler: Jan Issac, Stefan Maier (TGIJ1); Maik Fox, Thomas Kristof (TGIJ2)

Inhaltsverzeichnis

Inhaltsverzeichnis	1
1. Einführung	2
1.1. Um was geht es?.....	2
1.2. Wettbewerbsablauf.....	2
1.3. Wettbewerbsregeln.....	2
2. Hardware.....	4
2.1. Möglichkeiten	4
2.2. Vorüberlegungen.....	4
2.2.1. Antrieb.....	4
2.2.2. Ballführung und Schussmechanismus	4
2.2.3. Sensorik	6
2.3. Umsetzung und Probleme	6
V0.0 Ballführung	6
V1.0 Erste funktionsfähige Version.....	6
V1.1 Ein erster Sensor und die Fahne.....	7
V1.2 Komfortverbesserungen	7
V1.3 Verschleißverringerung	7
V1.4 Backspin-Geschwindigkeit	7
V1.5 Haltepodest für die Farbscheibe.....	8
V2.0 Navi-Verbesserungen.....	8
V3.0 Antriebsverbesserungen.....	8
V3.1 Backspinverbesserung	9
V3.2 Höhenverstellbare Stützvorrichtung	9
V3.3 Kleine Umbauten während dem Testen	9
V3.4 WETTBEWERBSVERSION	9
V3.5 Abgabe-Version	9
2.4. Was wir heute anders machen würden	10
2.4.1. Antrieb.....	10
2.4.2. Backspin	10
3. Software.....	11
3.1. Möglichkeiten	11
3.2. Vorüberlegungen.....	11
3.3. Das Modulprinzip.....	11
3.4. Die einzelnen Komponenten	12
3.4.1. Hauptsteuerung.....	12
3.4.2. Datenempfang und –Verarbeitung.....	12
3.4.3. Situationserkennung	13
3.4.4. Wegsteuerung.....	13
3.4.5. Motorsteuerung.....	14
3.4.6. Sensor(en).....	15
3.5. Probleme und Verbesserungen.....	15
3.5.1. Platz- und Kapazitätsprobleme	15
3.5.2. Datenempfangsprobleme	16
3.5.3. Reaktionszeitprobleme	16
3.5.4. Navigations- und Genauigkeitsprobleme	16
4. Der Wettbewerbstag.....	17
5. Presseecho.....	17
Literatur / Quellen	20

1. Einführung

1.1. Um was geht es?

Bei dem „LEGO® Fußball Roboter Wettbewerb“ im Jahr 2005 handelte es sich um eine Veranstaltung der FH Offenburg unter Leitung von Herrn Prof. Wülker. Ziel der Veranstaltung war das Design, der Bau und die Programmierung eines auf dem LEGO® Mindstorms™ 2 basierenden LEGO®-Roboters, der in der Lage ist, auf einem speziellen Spielfeld gegen einen Konkurrenten ein Fußballmatch zu spielen.

1.2. Wettbewerbsablauf

Jeder teilnehmenden Gruppe (zu Beginn waren es immerhin 24) wurde ein LEGO® Mindstorms™-Kasten zur Verfügung gestellt, der die erlaubten Bauteile zum Bau des Roboters enthielt. Andere Teile, abgesehen von Gummis und Akkus, durften nicht verwendet werden. Des Weiteren gab es eine CD mit diversen Beispielen, Entwicklungsumgebungen und sonstigen Hilfsprogrammen. Ein wichtiger Bestandteil war auch das auf der CD enthaltene LejOS, ein Open-Source-Betriebssystem für den Mindstorms™-Baustein. Somit hatten alle Gruppen von der Hardware her dieselben Grundvoraussetzungen.

Um die Teilnehmer auf die Programmierung des Bausteins in Java™ und das Verwenden des LejOS-Betriebssystems vorzubereiten, gab es zu Beginn des Wettbewerbs mehrere Einführungsvorlesungen, die den zuhörenden Schülern auch Einblicke in den Ablauf einer Lehrveranstaltung an einer Hochschule boten.

Prof. Wülker versuchte dabei, seinen von Sechstklässlern bis angehenden Abiturienten reichenden Zuhörern von den unterschiedlichsten Schularten und Landkreisen einen ersten Einblick in die Kunst der Java™-Programmierung zu ermöglichen und gleichzeitig Denkanstöße für die bevorstehende Programmierung des Roboters zu geben. Über die Java™-Inhalte hinausgehende Informationen betrafen die Konstruktion des Fußballroboters, den Ablauf des Wettbewerbs und der Empfang und die Verwendung der Positionsdaten des Systems, dass für die Positionserfassung der Roboter und des Balls zuständig ist.

Geplant waren insgesamt 4 dieser vorlesungsähnlichen Veranstaltungen innerhalb von 4 Wochen, beginnend am 17.01.2005. Nach der letzten Vorlesung, die auch mit amüsantem Videomaterial anderer Fußballroboterwettbewerbe angereichert war, blieben den Teams 4 Wochen bis zum Wettkampf, die letzten beiden dienten der praktischen Erprobung der Konstrukte auf dem speziellen Spielfeld vor Ort in der FH Offenburg. Der eigentliche Wettbewerb fand dann am Dienstag, den 15.03.2005, statt. An diesem Termin fanden sich dann nur noch 17 Teams in der FH ein, die in zwei verschiedenen Ligen angetreten sind: die eigentlich als ausschließlich stattfindende geplante autonome Liga mit sechs Teilnehmern und die Liga der ferngesteuerten Roboter mit insgesamt elf Teilnehmern.

1.3. Wettbewerbsregeln

Im Folgenden werden die Regeln grob und unter Verzicht der Feinheiten wiedergegeben.

- Es dürfen nur die Bauteile aus dem LEGO® Mindstorms™-Kasten verwendet werden, dem noch ein dritter Motor beigelegt war. Ausnahme war der Tausch von gealterten Gummis und Batterien.
- Als Ball dient ein orangener handelsüblicher Golfball
- Die Größe des Roboters durfte eine Höhe von 15 cm und einen Radius von 11 cm nicht überschreiten.
- Die Roboter dürfen den Ball nicht zu mehr als 20% in ihren Umriss aufnehmen und nicht festhalten, damit ein Gegner die Möglichkeit hat, den Ball an sich zu reißen.

2. Hardware

2.1. Möglichkeiten

Es stand ein um einen dritten Getriebemotor ergänzter LEGO® Mindstorms™-Kasten zur Verfügung, um den Roboter zu bauen.

Die Möglichkeiten mit LEGO® sind sehr vielfältig und daher wurde ein weites Spektrum an Lösungen erwartet.

2.2. Vorüberlegungen

2.2.1. Antrieb

Zu Beginn machten wir uns Überlegungen über den Antrieb des Roboters.

Anhand der Vorbereitungsveranstaltungen und diverser Vorlesungsfolien, die auf der Homepage des Wettbewerbs [3] zu finden sind, machten wir uns Gedanken über die Umsetzungsmöglichkeiten eines möglichst einfach zu bauenden und einfach ansteuerbaren Antrieb.

Unsere Entscheidung fiel relativ schnell auf einen Antrieb, der mittels zweier Motoren, die jeweils ein Rad antreiben. Die Motoren waren relativ zur Robotermitte symmetrisch angeordnet, prinzipbedingt waren beide Räder auf einer imaginären gleichen Achse, damit man durch eine gleiche Umdrehungsgeschwindigkeit der Motoren gleichmäßig nach vorne fahren kann, jedenfalls in der Theorie. Diese Konstellation mit zwei Antriebsrädern erfordert außerdem eine Stützvorrichtung, damit ein stabiler Zustand erreicht werden kann. Der Roboter musste daher auch so konstruiert werden, dass der Schwerpunkt möglichst zwischen der imaginären Antriebsachse und dieser Stütze liegt.

Damit bleibt auch der dritte Motor für eine benötigte Schuss- oder Ballführungseinrichtung übrig.

2.2.2. Ballführung und Schussmechanismus

Ziele der Entwicklung der Ballführung waren vor allem eine sichere Ballführung und auch eine möglichst kraftvolle Schussvorrichtung.

Am Anfang unserer Überlegungen stand eine Art Rammbock, der mittels eines Spanngummis und einer motorgetriebenen Aufzieh- und Einrastvorrichtung den Ball beim Auslösen ein Stück „schießen“ sollte. Wegen der schwierigen Umsetzung, die weder viel Platz noch Bauteile verschlingen hätte dürfen und außerdem zuverlässig hätte funktionieren müssen, haben wir diese Idee jedoch wieder fallengelassen.

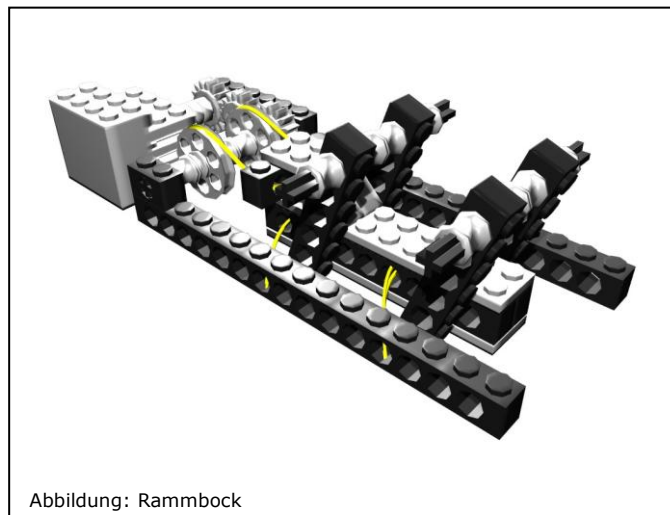


Abbildung: Rammbock

Eine andere Idee könnte man mit einem Uhrpendel vergleichen, das mittels eines Motors nach oben gezogen wird und dann mit Wucht gegen den Ball fallengelassen wird. Nachteil dieser Lösung wären mangelnde Präzision, großer Platzbedarf und, ebenso wie bei der ersten Idee, eine relativ lange „Aufladezeit“ nach einem Schussversuch gewesen.

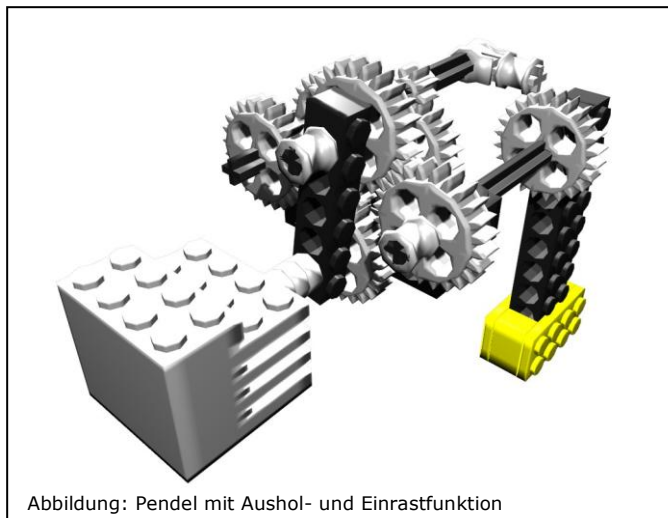


Abbildung: Pendel mit Aushol- und Einrastfunktion

Vor dem Ausbrüten der nächsten Idee haben wir die Voraussetzungen priorisiert und der sicheren Ballführung ein höheres Gewicht zugestanden. Daher fiel die Entscheidung auf eine so genannte Backspin-Vorrichtung, welchen dem Ball einen der Roboterbewegungsrichtung entgegengesetzten Drehimpuls versetzt und den Ball damit an Roboter und Backspin-Vorrichtung „kleben“ lässt. Um den Ball sicherer behalten zu können, wurden die Backspin-Walzen leicht V-förmig angeordnet, so dass sie gerade der 20%-Regel genügen (siehe 1.3) und den Ball damit auch nach dem Prinzip eines Schneeräumfahrzeuges immer in der Mitte halten können sollte. Des Weiteren wäre somit eine präzise Manövrierung des Balles möglich, da er sich mit gewisser Wahrscheinlichkeit immer in der Mitte der Walzen aufhalten sollte.

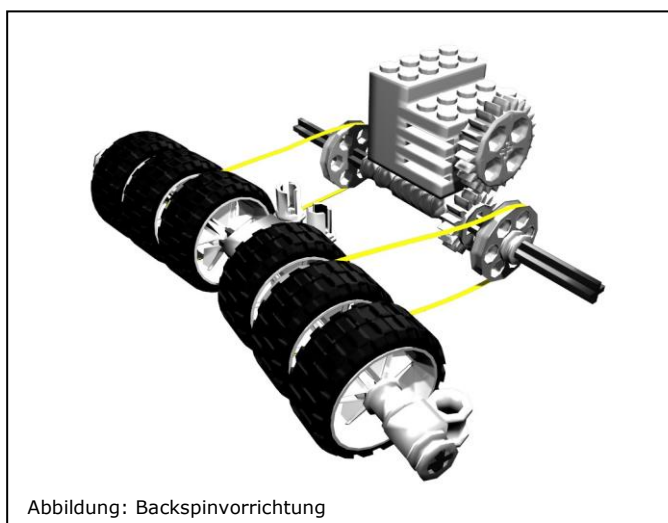


Abbildung: Backspinvorrichtung

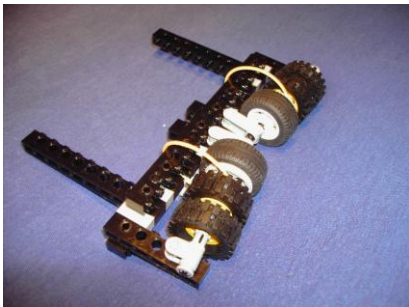
2.2.3. Sensorik

Bei unseren Vorüberlegungen hatten wir erstmal keine Anwendung für die dem Baukasten beigelegten Licht- und Tastsensoren, da die Positionsbestimmung der Gegenstände auf dem Spielfeld auf 0,75 cm genau sein sollte und wir uns daher zu Beginn keine weiteren Gedanken über eine Onboard-Navigation machten.

2.3. Umsetzung und Probleme

Die Entwicklungsgeschichte des Roboters ist chronologisch aufgebaut und zur besseren Übersicht mit Versionsnummern versehen. Diese finden sich immer vor der Kapitelüberschrift. Größere Entwicklungsschritte bekamen meistens eine neue Hauptversionsnummer, während kleine Änderungen nur eine neue Unterversionsnummer erhielten.

V0.0 Ballführung



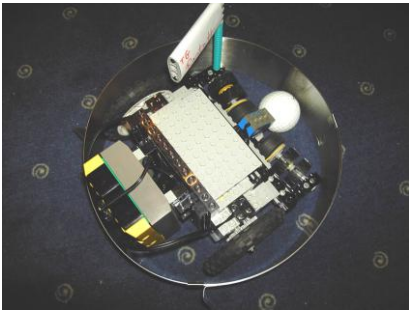
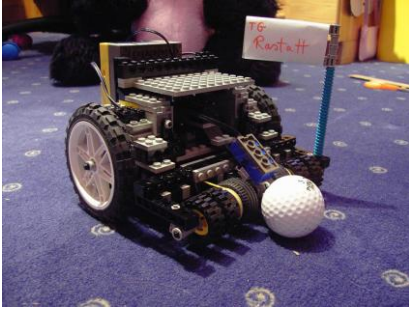
Die Ballführung bestand nur aus dem Nachbau der Backspin-Idee. Durch die trickreiche Konstruktion hätte man sogar die Neigung der Walzen nachträglich verändern können. Die auf dem Bild sichtbare Oberseite war jedoch am Roboter dann die Unterseite. Hauptkonstruktionsproblem dieses Moduls war, einen Übergang von stehenden zu liegenden Steinen zu schaffen.

V1.0 Erste funktionsfähige Version



Der erste Prototyp verdeutlicht schon grob die finale Form des Roboters. Die Antriebe sind angebracht, wie wir es geplant hatten und man sieht, dass die Ballführungseinrichtung angebaut wurde. Bei der Konstruktion wurde das Augenmerk auch auf Stabilität gelegt, einen (unfreiwilligen) Falltest aus 30 cm Höhe hat der Roboter ohne Schaden überstanden. Auf dem unteren Bild sieht man den LEGO® Mindstorms™ RCX-Baustein, der hinten stehend montiert wurde. Damit war der Infrarot-Empfang gegeben, da sich der Empfänger auf der schwarzen Oberseite des Bausteins befindet. Die rückwärtige Montage des Bausteins diente auch der Verlagerung des Schwerpunkts hinter die Antriebsachse, damit die Ballführung nicht auf dem Boden schleift. Es wurde dann aber die unter dem RCX-Baustein platzierte Stütze notwendig. Das Männchen auf dem Roboter diente nur der Dekoration, da es leider trotz vieler Tests nicht in der Lage war, den Roboter zu steuern.

V1.1 Ein erster Sensor und die Fahne



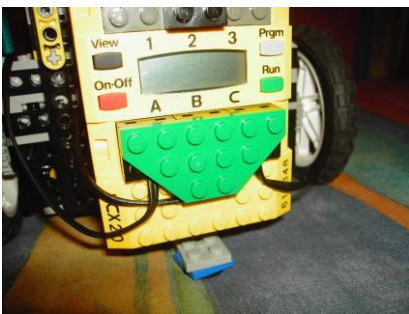
Aus Platz- und Gewichtsgründen wurde der Sitz samt Fahrer entfernt, da er leider über seine Inkompetenz nicht hinauswuchs. Außerdem wurde der Platz für einen Lichtsensor benötigt, der so positioniert ist, dass er den Ball erfassen kann, wenn er in der Ballführung in der optimalen Position liegt. Man kann so mit diesem Sensor die Bewegungsmuster und –Geschwindigkeiten einfacher und schneller an die Situation anpassen, als dies mit dem Bilderfassungssystem möglich gewesen wäre.

Neu war außerdem die Fahne, mit zugegebenermaßen provisorischer Flagge.

Auf dem unteren Bild sieht man einen ersten Größentest: Eine Kuchenbackform diente als Zylinder, um die Größe des Roboters auf Regelkonformität zu überprüfen.

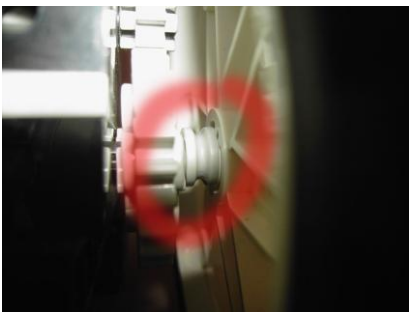
Auch von der Höhe her war unsere Konstruktion schon in diesem Stadium zulässig und es musste nichts verändert werden. Auch die Fahne ragte nicht über die zulässigen 15 cm hinaus.

V1.2 Komfortverbesserungen



Diese Version sah neben einer ausgearbeiteten Fahne, die das Geschmier aus der vorherigen Version ablöste, hauptsächlich Verbesserungen des Komforts vor: Ein Haltegriff, um den Roboter einfach anheben zu können, das im nebenstehenden Bild sichtbare „Quick-Clip“, um die Kabel schnell an- und abmontieren zu können, eine Art Kabelrolle für das zu lange Kabel des Lichtsensors und eine Art Sendegerüst für das Infrarot-Programmier-Interface von LEGO®.

V1.3 Verschleißverringern



Da sich das Zahnrad am Motor immer von selbst von der Achse schob und dann am Antriebsrad rieb, haben wir mittels eines kleinen Abstandshalters die Räder weiter von dem Zahnradgetriebe entfernt. Dadurch konnte das Zahnrad nicht mehr an den Rädern schleifen.

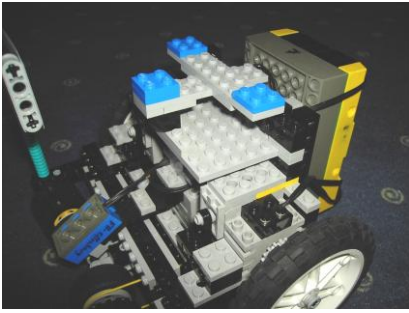
V1.4 Backspin-Geschwindigkeit



Um die Zuverlässigkeit und Haltewirksamkeit des Backspins zu erhöhen, stand eine Änderung der Übersetzung des Walzantriebs an: Die beiden gleichgroßen Zahnräder wurden gegen zwei unterschiedlich große getauscht, um die Drehgeschwindigkeit zu erhöhen. Durch eine veränderte Position des Antriebsmotors war dieser Umbau relativ aufwendig.

Mit diesem Umbau ging auch eine Veränderung des Kabelhalters des Lichtsensorkabels einher.

V1.5 Haltepodest für die Farbscheibe



Für das Bilderfassungssystem war eine Farbscheibe notwendig, die auf dem Roboter angebracht werden musste. Diese Halterung ist auf dem Bild links zu sehen.

Aus Platzgründen musste dann auch der Haltegriff wieder weichen.

V2.0 Navi-Verbesserungen



Der erste Hauptversionssprung kam nach dem großen Umbau nach den ersten paar Testläufen in Offenburg zustande.

Offensichtlichste Änderung ist der links zu sehende Tastsensor, der mittels 6 im Rad montierten Noppen die Radumdrehungsgeschwindigkeit erfassen konnte. Damit war ein weiterer Schritt in Richtung unabhängige Navigation getan, da wir mit der Geschwindigkeit des Bilderfassungssystems und der Datenübertragungszuverlässigkeit nicht besonders zufrieden waren.

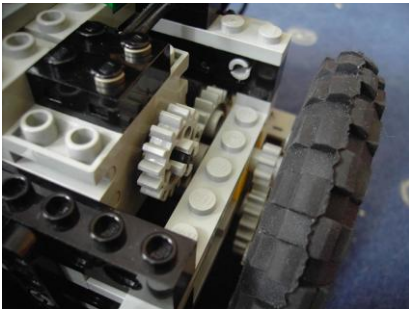
Weiterhin wurde das Übersetzungsverhältnis des Antriebs auf die Hälfte verlangsamt, womit ebenfalls eine genauere Navigation auf dem Spielfeld möglich wurde.

Durch diese markanten Umbauten wurde praktisch der halbe Body des Roboters umgebaut.

Für die 3 Sensoren wurde jetzt auch das „Quick-Clip“ eingeführt.

V3.0

Antriebsverbesserungen



Nach weiteren Testläufen kamen wir zu dem Schluss, dass wir den Antrieb weiter verlangsamen müssen. Dem folgte ein komplizierter Umbau, um die niedrigere Übersetzung über nun 2 Achsen erreichen zu können. Wegen der komplett unterschiedlichen Konstruktion wurde unter Modul-Recycling fast der gesamte Body des Roboters neu gebaut, auch der Aufbau des gerade in der Vorversion eingeführten Radumdrehungszahlmessers musste überarbeitet werden, damit er mit der neuen Konstruktion kompatibel war.

Positiver Nebeneffekt dieses Radikalumbaus war eine noch stabilere und flachere Konstruktion als vorher.

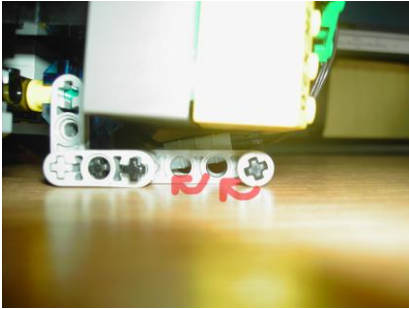
Des Weiteren war eine andere Stützvorrichtung unter dem RCX-Baustein notwendig, da die alte manchmal an den weißen Klett-Spielfeld-Markierungen hängen geblieben ist. Diese Vorrichtung sieht man im unteren Bild.

V3.1 Backspinverbesserung



Hier gab es einen Umbau der Stützvorrichtung, um einen besseren Berührungspunkt der Backspinwalze am Ball zu erreichen. Die Backspinrollen auf der Walze wurden neu angeordnet, um den Grip und die Führung zu verbessern.

V3.2 Höhenverstellbare Stützvorrichtung



Da sich die umgebaute Stützvorrichtung aus der Vorversion nicht behaupten konnte, wurde sie komplett umgebaut und durch eine höhenverstellbare Variante ersetzt. Im Bild links ist zu sehen, dass man mittels eines Stiftes drei verschiedene Höhen einstellen kann, um auch während den Testläufen mit der Einstellung experimentieren zu können.

V3.3 Kleine Umbauten während dem Testen

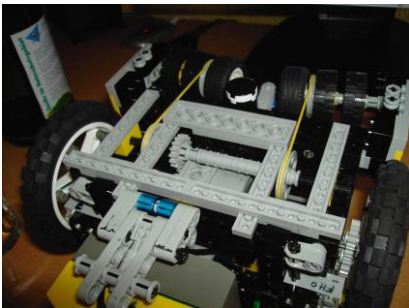
Einige minimale Änderungen während den Testläufen in Offenburg, u. a. Änderung der Höhen-Einstellung der Stützvorrichtung.

V3.4 WETTBEWERBSVERSION



Am Tag vor dem Wettbewerb haben wir noch ein paar Ballführungskomponenten am Roboter hinzugefügt, damit der Ball sich nicht in diesen Einbuchtungen verfangen konnte. Im Bild sieht man die abgedeckten Freiräume unter dem Backspingerüst. Außerdem wurde zum wiederholten Male die Neigung der Backspinwalze justiert, um den Ball noch besser führen zu können.

V3.5 Abgabe-Version



Durch die vielen Kollisionen mit den Gegnern beim Wettbewerb bemerkten wir noch eine Schwachstelle bei der Backspinantriebsachsenhaltevorrichtung, die sich bei Blockade der Backspinachse löste und dann unschöne Geräusche von sich gab. Diese Version wurde dann bei der FH Offenburg abgegeben und wird dort wahrscheinlich mit der CAD-Lösung SolidEdge nachgebaut.

2.4. Was wir heute anders machen würden...

2.4.1. Antrieb

Während unserer Tests stellten wir fest, dass das von uns gewählte Antriebskonzept nicht optimal ansteuerbar war. Auch mit den Drehzahlsensoren für die beiden Räder konnten wir nicht ausgleichen, dass die Motoren, die die Räder antrieben, mit unterschiedlichen Geschwindigkeiten liefen. Daher war nicht einmal ein Geradeausfahren möglich.

Eine Alternative wäre ein Gabelstapler ähnlicher Antrieb gewesen, mit einer durchgängigen Antriebsachse und daher gleichen Umdrehungsgeschwindigkeiten beider Antriebsräder, was ein Geradeausfahren ermöglichen würde. Die Lenkung würde dann mit einem Stützrad erfolgen, das sich per Motor in die gewünschte Lenkrichtung bewegen lassen würde. Um den gewünschten Lenkeinschlag einigermaßen genau erreichen zu können, wäre ein Tastsensor für die Nullstellung „Geradeaus“ benötigt worden und ein zweiter zum Erfassen des Lenkeinschlags in eine von der Übersetzung der Noppeneinrichtung abhängigen Auflösung.

Diese Idee kam uns aber zu spät, um eine Umsetzung noch riskieren zu können.

2.4.2. Backspin

Unser Backspin bot zwar die Möglichkeit, einen Ball geradeaus einigermaßen sicher zu führen, Rückwärtsfahren war jedoch damit gar nicht möglich.

Eine mögliche Verbesserung wäre hier gewesen, größere Räder mit noch höherer Umdrehungsgeschwindigkeit zu verwenden. Dass das funktioniert, konnten wir bei einem Roboter einer anderen teilnehmenden Gruppe beobachten. Mit deren Roboter war es sogar möglich, mit Ball rückwärts zu fahren.

3. Software

3.1. Möglichkeiten

Theoretisch hätten wir den RCX-Baustein programmieren können, wie es uns beliebt. Es gab keine Restriktionen in Bezug auf Programmiersprache, Betriebssystem oder Entwicklungsumgebung.

Da aber der Professor eine Einführung in das Open Source Betriebssystem „LeJos“ [4] bot, benutzten wir es auch. Dieses Betriebssystem führt speziell kompilierte Java-Programme aus und ist daher relativ einfach zu programmieren.

Als Entwicklungsumgebung benutzen wir „eclipse“ [5] mit einem Plugin für LeJos, um die Vorteile der umfassenden Programmierhilfen von eclipse nutzen zu können.

3.2. Vorüberlegungen

Wir informierten uns im Vorfeld über das Betriebssystem, bevor wir dem Planen und Programmieren angingen. Uns fiel auf, dass das LeJOS-Betriebssystem sich anders als vom PC gewohnte Java-Virtual-Machines verhält. Hauptproblem liegt bei dem Mini-Java-Betriebssystem in der Speicherverwaltung: Der Speicher, der von einem einmal instanziierten Objekt verwendet wurde, wird nie mehr freigegeben, um ihn von anderen Objekten nutzen zu können. Daher entschieden wir uns für eine weitgehend objektfreie Implementierung unseres Programms, da wir so eine bessere Kontrolle über den Speicher hatten und auch nicht fürchten mussten, dass das Programm wegen ausgegangenem Speicher hängen bleibt und nicht mehr weiterläuft.

Auch entschieden wir uns gegen andere, PC-übliche Vorgehensweisen wie Threads, da wir die Kontrolle behalten wollten, wann der Baustein was macht. Ein paar Threads ließen sich jedoch nicht vermeiden, da es z.B. von LeJOS so vorgesehen ist, Threads für den Datenempfang und Listener für die Sensoren zu verwenden. Da diese aber nur einen kleinen Teil des Programms ausmachen, war das jedoch kein Problem.

Somit war unser Programm nach dem klassischen EVA-Prinzip aufgebaut: Eingabe, Verarbeitung, Ausgabe. Zuerst wollten wir auf neue Daten warten, dann ermitteln, wie wir uns zu verhalten haben, um dann zum Schluss die Motorenansteuerung aktualisieren zu können.

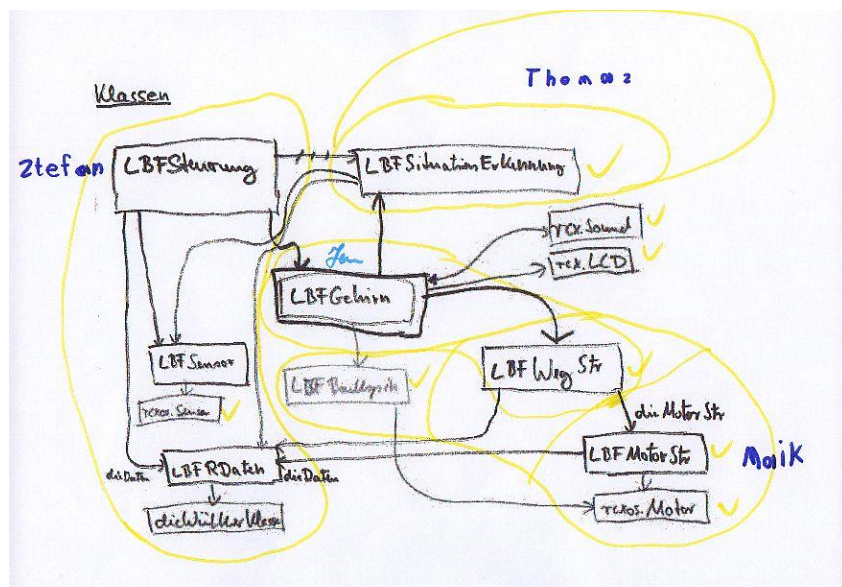
Außerdem entschieden wir uns für einen Aufbau nach einem Modulprinzip, was Vorteile bei der Wartbarkeit des Quellcodes bietet und auch ermöglicht, Aufgaben im Team zu verteilen.

3.3. Das Modulprinzip

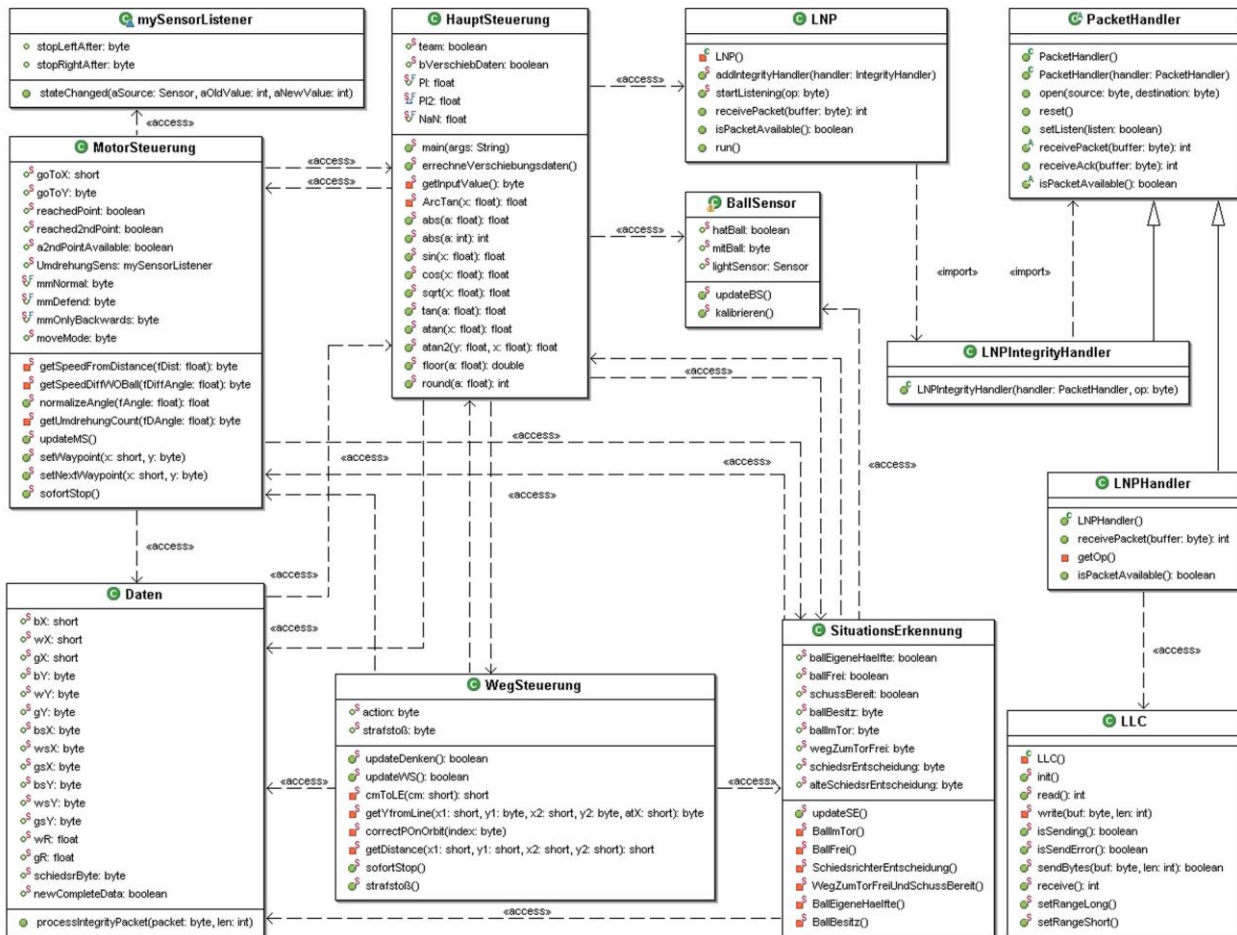
An einem Treffen des Teams entschieden wir uns, wie schon im vorherigen Abschnitt gesagt, für einen modularen Aufbau des Programms.

An diesem Treffen ermittelten wir die einzelnen benötigten Module anhand einer Analyse der Funktionen, die die Software erfüllen muss.

Außerdem diente es der Aufgabenverteilung, wie man an der Grafik rechts erkennen kann, die eine erste Skizze darstellt und nicht den finalen Stand der Software.



3.4. Die einzelnen Komponenten



Obiges UML-Klassendiagramm stellt den Wettbewerbs-Stand der Software dar, nur etwas um diverse Variablen abgespeckt, damit es einigermaßen übersichtlich bleibt.

3.4.1. Hauptsteuerung

Hauptaufgabe der Hauptsteuerung ist einerseits die Initialisierung aller anderen Klassen, das Starten des Datenempfangs und der Listener, das Justieren des Ballsensors und der zyklische Aufruf der Aktualisierungsoperationen der einzelnen Klassen, andererseits die Bereitstellung von allgemeinen optimierten Funktionen wie den mathematischen Operationen.

Auch eine einen Tag vor dem Wettbewerb eingebaute Positionsschätzung, die versucht, die Verzögerung des Bilderfassungssystems und der Übertragung zu kompensieren, findet hier statt.

3.4.2. Datenempfang und –Verarbeitung

Der Datenempfang wird mittels eines Threads realisiert und basiert zu Teilen auf einer Vorlage des Professors. Diese Vorlage konnten wir leider nicht so weiterverwenden, wie sie uns vorlag, da sie nicht in unser Modell passte. Außerdem waren wir mit der Effizienz und der Codegröße nicht zufrieden.

Effektiv funktionierte der Datenempfang dann folgendermaßen: Wenn ein neues, gültiges Datenpaket empfangen worden ist, wurde ein Bit gesetzt. Die Hauptsteuerung wartet auf dieses Bit und wenn es dann gesetzt ist, veranlasst sie eine Verarbeitung der empfangenen Rohdaten. Die Verarbeitung der Rohdaten findet in der Datenklasse direkt statt, die auch dann die eigentlichen Positionsdaten allen anderen Klassen zur Verfügung stellt.

Während der Verarbeitung der Rohdaten dürfen keine neuen Daten empfangen werden, da sonst inkonsistente Daten mit vermischten Positionsdaten das restliche Programm durcheinander bringen würden und je nach dem sogar eine falsche taktische Entscheidung hervorrufen würden.

3.4.3. Situationserkennung

Kein Fußballspiel, ohne dass man erkennt, in welcher Situation man gerade ist.

Anhand der empfangenen und verarbeiteten Positionsdaten und den Sensordaten wird die derzeitige Situation auf dem Spielfeld in einfach zu handhabende Statusinformationen übersetzt, mit denen weiter gearbeitet werden kann. Die Situationserkennung liefert somit wichtige Daten für alle folgenden taktischen Entscheidungen und auch für das Motorsteuerungsverhalten.

Wenn die update-Operation durch die Hauptsteuerung aufgerufen wird, wird folgendes ermittelt:

- Auf welcher Spielfeldhälfte sich der Ball befindet
- Ob der Gegner im Ballbesitz ist
- Ob wir selbst den Ball haben, und wenn ja, wo am Roboter (in der Mitte der Walze oder leicht daneben vor dem Roboter)
- Ob der Weg zum gegnerischen Tor frei ist
- Ob wir uns in Schussposition befinden und eine Chance haben, das gegnerische Tor zu treffen
- Ob der Ball frei ist oder ob er sich in einem Tor befindet

3.4.4. Wegsteuerung

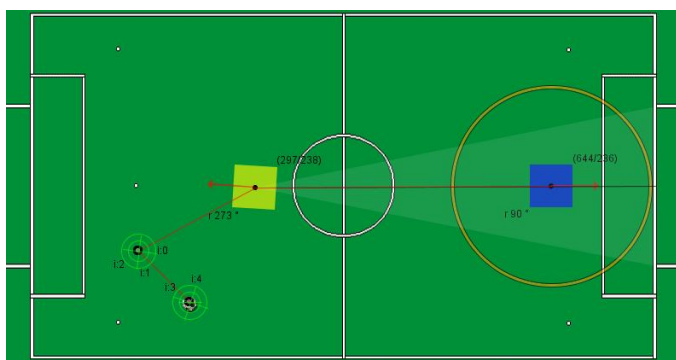
Die Wegsteuerung hatte als Aufgabe, taktische Entscheidungen zu treffen und der Motorsteuerung die Wegpunkte zu übergeben, die der Roboter anfahren soll.

Grundlegend gibt es drei Situationen: Ball holen, Angriff und Verteidigung. In welcher sich der Roboter derzeit befindet, wird anhand der Daten der Situationserkennung entschieden.

Allen Situationen ist gemeinsam, dass versucht wird, um den Gegner herumzufahren, um kein Foul zu riskieren.

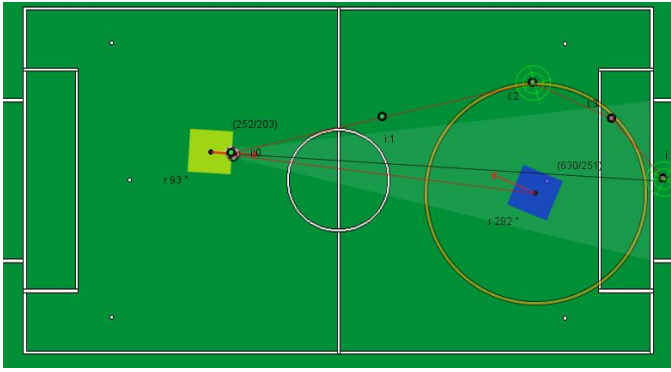
Vorgehensweise in der Situation „Ball holen“:

- Wegpunkte werden so um den Ball herum gesetzt, dass der Roboter, sobald er den Ball hat, immer in Richtung Tor steht. Wichtiger Faktor ist hier der sog. Umfahrradius, der angibt, in welchem Abstand der Ball umfahren wird.



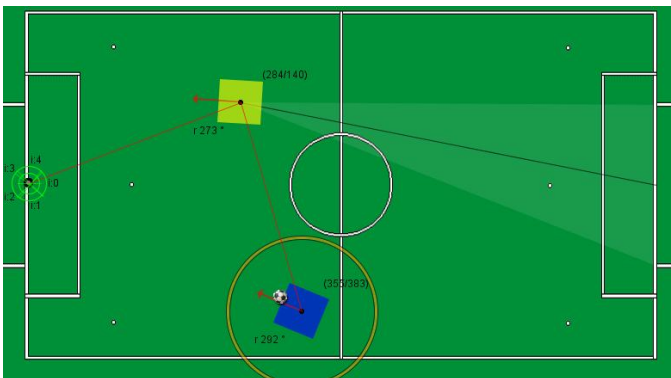
Vorgehensweise in der Situation „Angriff“:

- Generelles Ziel ist, den Ball ins Tor zu bringen.
- Gegner umfahren unter Verwendung der Ausweichrichtung auf dem Spielfeld, die mehr Platz und damit mehr Sicherheit bietet.
- Wenn der Gegner im Tor steht, werden die Wegpunkte um den Gegner herumgesetzt und ein letzter Wegpunkt in die Mitte des Tores gesetzt. Damit soll erreicht werden, dass der Roboter um den Gegner herum den Ball ins Tor bringt.



Vorgehensweise in der Situation „Verteidigen“:

- Schnellstmöglich zum eigenen Tor fahren
- Auf Kollisionskurs zum Gegner gehen, um ihn zum Ausweichen zu zwingen. Falls jedoch der Abstand zum Gegner zu gering wird, bleiben wir stehen, um kein Foul zu provozieren.



3.4.5. Motorsteuerung

Grundlegende Funktion dieses Moduls ist die Ansteuerung der beiden Antriebsmotoren und die Auswertung der Daten von den Radumdrehungszahlsensoren.

Die Motorsteuerung bekommt von der Wegsteuerung bis zu zwei Wegpunkte übergeben, die dann der Reihe nach angefahren werden sollen.

Auch wenn mehr geplant war, unterstütze die Endversion nur das Vorwärtsfahren zuverlässig. Rückwärtsfahren und Spezialmanöver wie das Drehen auf der Stelle waren zwar programmiert, aber wegen Zeitmangel nicht getestet.

Funktionsweise der Motorsteuerung:

- Grundlegend wird der Winkel zum Ziel ermittelt und dann berechnet, mit welcher Drehrichtung sich der Roboter am schnellsten auf das Ziel ausgerichtet hat.
- Desto näher wir an einen Wegpunkt herankommen, desto langsamer wird die Geschwindigkeit, mit der wir uns bewegen. Auf großer Distanz, wird versucht, mit der größtmöglichen Geschwindigkeit zu fahren. Da die Wegpunkte effektiv kaum erreicht werden, da sich ständig ändernde Situationen dauernd neue Wegpunkte hervorrufen, sieht man jedoch trotzdem eine durchgehende Bewegung und nur in Situationen wie dem Ballholen diesen Effekt.
- Um die Trägheit des Bilderverarbeitungssystems auszugleichen, werden für größere Drehungen die Radumdrehungszahlsensoren ausgewertet und gestoppt, wenn ein bestimmter Drehwinkel erreicht wurde. Das funktioniert jedoch nur bei der Drehung mit einem aktiven Antriebsrad, während das andere stillsteht.

3.4.6. Sensor(en)

Das Sensormodul war eigentlich nur für den Lichtsensor für die Ballerkennung zuständig, da die beiden Tastsensoren schon von der Motorsteuerung ausgewertet werden.

Das sog. Ballsensormodul hatte die Aufgabe, den Sensor bei Start des Roboters auf den Ball einzukalibrieren und dann anhand dieser Kalibrierungsdaten zu ermitteln, ob der Ball in der Mitte der Backspinwalze liegt oder nicht.

3.5. Probleme und Verbesserungen

3.5.1. Platz- und Kapazitätsprobleme

Hauptanfangsproblem war, unsere erste Prototypensoftware auf dem Baustein zum Laufen zu bekommen.

Der RCX-Baustein hat ein statisches RAM von 32 KB Größe. Da das Betriebssystem 20 KB davon in Anspruch nimmt und das verbleibende RAM sowohl als Programmspeicher als auch als Arbeitsspeicher dient, durfte das Programm nicht größer als ungefähr 10 KB werden.

Unsere erste Programmversion war jedoch 18 KB groß und konnte somit nicht einmal auf den Baustein übertragen werden. Nachdem wir also viel Code entfernt und optimiert haben, konnten wir das Programm auf den Baustein übertragen, jedoch nicht ausführen. Unser Programm nahm soviel Platz in Anspruch, dass das verbleibende RAM nicht mehr als Arbeitsspeicher ausreichte. Bei dem Start des Programms kam nur der Fehler „5“, der sich nach einiger Recherche als Speicher-Voll-Fehler entpuppte.

Nach der nächsten radikalen Verschlinkungskur hatten wir das Programm auf eine Größe von 8 KB reduziert, jedoch wurden dabei viele Funktionen entfernt, um den Rumpf des Programms samt Datenempfang zum Laufen zu bekommen. Mit dieser Größe ließ sich das Programm dann starten und der Roboter erwachte das erste Mal zu einem zugegebenermaßen chaotischen Leben.

Weitere Recherchen brachten noch eine große Schwachstelle des Betriebssystems LejOS zum Vorschein: Die eigentlich von der JavaVM übernommene Aufgabe der Speicherverwaltung erfüllt LejOS nur in absoluten Grundzügen. Einmal erzeugte Objekte belegen bis zum Neustart des Bausteins Arbeitsspeicher, den man dann nicht mehr verwenden kann. Typische Java-Programmiertechniken wie übertriebene Objektorientierung und dynamisch erzeugte Objekte waren daher nicht sehr zuverlässig zu verwenden, da der Baustein einfach stehen blieb, wenn er keinen Arbeitsspeicher mehr hatte.

Um alle Funktionen einbauen zu können, die wir geplant hatten, mussten wir jedoch noch sehr viel mehr am Quellcode und auch an den Klassen des Betriebssystems, die uns zur Verfügung standen, umbauen. Zusammengefasst haben wir folgende Änderungen vorgenommen:

- Die vom Professor vorgegebene Initialisierungsweise des Empfangssystems hatte den Effekt, dass Speicher doppelt verschwendet wurde, da die benötigten Objekte zweimal erzeugt wurden. Wir haben einfach eine Zeile Quellcode entfernt und es funktionierte trotzdem noch alles.
Ersparnis: 2 KB
- Alle Verwendungen des Datentyps Integer wurden auf Typen mit weniger Speicherplatzbedarf umgestellt, wo es möglich war.
- Kompletter Umbau der Datenempfangsklassen, sowohl unserer eigenen als auch von Teilen der LejOS-Klassen. *Ersparnis: 2-3 KB*
- Die Math-Klasse, die für mathematische Operationen nötig ist, verwendet durchgängig den Datentyp Double, der einen enormen Speicherplatzbedarf aufweist und wegen der 8 Bit-Natur des Microcontrollers im RCX-Baustein auch sehr viel Code erzeugt, um mit ihm rechnen zu können. Die benötigten Funktionen aus dieser Math-Klasse haben wir daher in unsere Hauptsteuerungs-Klasse übernommen und auf den für uns ausreichenden Datentyp Float umgebaut. *Ersparnis: 1,5 KB*

3.5.2. Datenempfangsprobleme

Einerseits gab es Probleme, die den Transfer des Programms auf den RCX-Baustein betreffen. Es war teilweise nicht festzustellen, warum ein Transfer wieder abgebrochen ist. Auch wenn die Programme nur um die 10 KB groß waren, dauerte eine komplette Übertragung knapp zwei Minuten. Mit der Zeit stellten wir fest, dass sowohl andere sich im Raum befindliche Infrarotsender und auch Leuchtstoffröhren potentielle Störquellen waren. Der Rückzug in dunkle Ecken und das Programmieren in einem Schuhkarton waren daher die besten Lösungen, die wir hatten.

Andererseits gab es Probleme beim Empfang der Positionsdaten auf dem Spielfeld. Ohne feststellen zu können, warum, hörten die Baustein manchmal einfach auf, Daten zu empfangen. Das kurze Abdecken des Infrarotempfängers am RCX-Baustein konnte das Problem meistens beheben. Beim Wettkampf selbst waren wohl die vielen Infrarot-Abstandsmesseinrichtungen von Digitalkameras und Videokameras eine zusätzliche Störquelle, die zu einem deutlich verminderten Datenempfang führte.

3.5.3. Reaktionszeitprobleme

Hauptproblem war hier die Zeit, die verging, bis eine Positionsänderung vom Bildverarbeitungssystem erfasst war, und dann vom Server per Infrarot versendet war und dann noch von unserem RCX-Baustein empfangen und verarbeitet war. Diese Zeit betrug ungefähr eine dreiviertel Sekunde.

Dies versuchten wir durch eine Positionsabschätzung aus unserer eigenen Geschwindigkeit, die wir mit Hilfe der Radumdrehungszahlmessung ermitteln konnten, und unserer eigenen Rotations-Geschwindigkeit, die wir anhand des Bildverarbeitungssystems ermittelten, auszugleichen.

Der Lichtsensor, der für die Erkennung des Balls zuständig war, lieferte manchmal kurzzeitig seltsame Werte. Normalerweise haben wir sofort auf einen „Ballverlust“ reagiert, und sind dann in einen anderen Bewegungsmodus gegangen. Da wir aber in Wirklichkeit den Ball noch hatten, haben wir ihn dadurch meistens verloren. Daher haben wir eine kleine Verzögerung eingebaut, die erst bei dreimaliger Meldung, dass kein Ball mehr in Reichweite des Sensors ist, auch ausgibt, dass kein Ball mehr vorhanden ist. Damit konnte das eben geschilderte Problem beseitigt werden, wenn der Ball jedoch wirklich verloren ging, dauerte es länger, bis wir darauf reagieren konnten.

Durch die begrenzte Geschwindigkeit des RCX-Bausteins dauerte es auch immer eine gewisse Zeit, bis unser Programm abgearbeitet war und auf neue Situationen reagieren konnte.

3.5.4. Navigations- und Genauigkeitsprobleme

Problemquelle war hier hauptsächlich das Bilderfassungs- und Verarbeitungssystem, das von Genauigkeit und Zuverlässigkeit her zu wünschen übrig ließ.

Zu Beginn der Testläufe stand nur eine Hälfte des Spielfelds zu Verfügung, da die Software des Professors noch nicht für das komplette Spielfeld ausgelegt war. Als dann auch die zweite Kamera eingebunden wurde, die den restlichen Teil abdeckte, tauchten Probleme im Bereich der Spielfeldmitte auf, den beide Kameras beobachteten und daher für die dort befindlichen Objekte zwei verschiedene Positions- und Rotationsdaten zur Verfügung standen. Diese Tatsache machte dem Übertragungsprogramm des Professors zu schaffen und führte zu Daten, die teilweise jeglichen Bezug zur Realität vermissen ließen.

Auch sonst hat schon eine Hand, die sich knapp im Spielfeld befand, das Bildverarbeitungssystem glauben lassen, dass diese Hand wegen der ähnlichen Farbe den Ball darstellt. Die Ballposition sprang damit teilweise unkontrolliert in der Gegend umher und die Roboter verhielten sich entsprechend chaotisch.

Ebenso gab es Probleme mit der Genauigkeit, die laut Professor optimistische 0,75 cm betragen sollte. Jedoch war es trotzdem schwer festzustellen, ob der Ball nun wirklich beim eigenen Roboter oder ein paar cm daneben oder sogar, ob der Ball vor oder hinter dem eigenen Roboter lag.

Auch schaffte ein Programmierfehler in einem Programm des Professors Verwirrung, da die Farbscheiben, die die Teams identifizierten, durcheinander gebracht worden sind.

4. Der Wettbewerbstag

Der eigentliche Wettbewerb fand dann am Dienstag, den 15. März, in der FH Offenburg statt.

Wie schon am Anfang erwähnt, fanden sich nur noch 17 Teams ein, die in zwei verschiedenen Ligen angetreten sind: die eigentlich als ausschließlich stattfindende geplante autonome Liga mit sechs Teilnehmern und die Liga der ferngesteuerten Roboter mit insgesamt elf Teilnehmern.

Eifel-Storms 1	Eifel-Storms 2	3
TGI Botham 0	TG Spitzkicker 1	
Lörrach 3 0	Eifel-Storms 1	
TG Spitzkicker 2	EKG-Clan 2	
Lörrach 2 0	EKG-Clan 1	
EKG-Clan 5	Lörrach 1 0	
Lörrach 4 Verletzt		
Lörrach 1 0	EKG-Clan 1 2	1
	KIT-Robotics 1	
Käfer	Käfer 1	2
TG-Robosports	KIT-Robotics 2	2
KIT-Robotics Los	KIT-Robotics	

Turnierplan mit Ergebnissen der Fernsteuerliga

Banane flanken 4	Banane flanken 3 0	
Willibald 0	LittleBigFootball 2	
gregor 0	LittleBigFootball 1	
LittleBigFootball 1	LittleBigFootball 1	1
Dragonbot	DaBallmuesnue 0	
DaBallmuesnue Los	DaBallmuesnue 2	

Turnierplan mit Ergebnissen der autonomen Liga

Wie man am Turnierplan der autonomen Liga, an der wir teilgenommen haben, sieht, haben wir in jedem Spiel mindestens ein Tor geschossen und somit den Wettbewerb gewonnen. Daher kam es nie zu einem Elf-Meter-Schießen, welches unserer Roboter überhaupt nicht beherrscht hätte...

5. Presseecho



Noch stolpert der Lego-Roboter über das Grün

Prof. Dr. Michael Wülker gesteht zu, dass die am Montag gemachten leidlichen Erfahrungen dazu gehören. Die Schüler sind motiviert und wollen aus den Erfahrungen lernen. Es läuft die Endphase im Lego-Roboter-Wettbewerb der Hochschule Offenburg. 20 Gruppen nehmen daran teil – von Lörrach bis zur Eifel. Image und eventuell den einen oder anderen Schüler später als Studenten, das ist das Ziel der Hochschule. Am 15. März müssen die Schüler der Oberstufe (unser Foto: Technisches Gymnasium Rastatt) zeigen, dass ihre Lego-Roboter in der Lage sind, allein durch die Programmierung den Ball zu umdribbeln und dann über die Linie zu befördern. Foto: rek

Stadtanzeiger Offenburg 2. März 2005



Der Ball ist rund, das Spiel dauert acht Minuten

Nicht die regulären 90 Minuten, sondern lediglich zweimal vier Minuten spielten die Teilnehmer des „Lego-Roboter-Fußball-Wettbewerbs“, der am gestrigen Abend in der Offenburger Fachhochschule ausgetragen wurde. Ansonsten war aber – von den Ballkünstlern aus bunten Legosteinen, Platinen, Kabeln und Mikrochips sowie dem fehlenden Torwart einmal abgesehen – alles genau so wie bei den Kickern aus Fleisch und Blut: Es gab Eck- und Schiedsrichterbälle, Strafstoße, Großbildleinwände, skandierende Fans mit Spruchbändern, aufgebrauchte Trainer... Die Sieger standen bis Redaktionsschluss nicht fest, da die Programme immer wieder abstürzten. Technik und Fußball passen eben doch nicht zusammen. Text / Foto: pl

Stadtanzeiger Offenburg 16. März 2005



Roboter »Lörrach 2« ist gerade am Ball, doch der EKG-Clan mit Steuermann Benedikt Seifert (rechts) greift an: Unter den 20 Schülerteams, die mit ihren selbst gebauten Fußballrobotern gegeneinander antreten, waren zehn aus der Ortenau. Foto: Stephan Hund

Kampf der Lego-Fußball-Roboter

IT-Wettbewerb der Hochschule Offenburg / Originellster Roboter aus Hausacher Gymnasium

VON SIMONE DENZLER

Käfer« ist am Ball. Die rote Kugel scheint wie am Roboter festgeklebt, folgt dem Lego-Fahrzeug mit den hübschen Kulleraugen in alle Richtungen, dank des »Backspin«-Mechanismus sogar rückwärts. »Steuermann« Justus Braach lässt »Käfer« am Roboter der »KIT-Robotics« vorbeidröbeln, spielt ihn aus. Doch da greift der Gegner an. Der Roboter der Kaufmännischen Schulen Offenburg ist schneller, schlagkräftiger als »Käfer«, schießt ein Tor – und gewinnt das Match. Der erste Lego-Roboter-Fußballwettbewerb der Hochschule Offenburg ist für »Käfer« hier beendet. Doch sein Team der Klasse 7a des Robert-Gerwig-Gymnasiums in Hausach gewinnt den Preis für den originellsten Roboter.

Im Publikum sitzen die Bastler und verfolgen gespannt die Spiele der Liga der ferngesteuerten Roboter. Manche legen noch hektisch letzte Hand an ihre Roboter an. Wie Fußballspieler sehen die Legobauten nicht aus. Sie erinnern mit ihren Rädern eher an Rasenmäher, an Gabelstapler. Manche haben vorne lange Schaufeln zum Schießen des Balls.

Sebastian Klaas vom Offenburg Oken-Gymnasium zuckt die Achseln: »Ob unser Roboter den Ball trifft, wissen wir

nicht«, erzählt der 18-Jährige und wirft einen schmunzelnden Blick auf Teamkollegin Delia Braun. Die 15-Jährige ist eines der wenigen Mädchen im Wettbewerb. Stolz erzählt sie, dass ihr »Dragonbot« in der Liga der »autonomen« Roboter spielt. Das bedeutet, dass die Maschine völlig selbstständig Fußball spielen kann. Nur sechs Teams der 20 haben es geschafft, ihren Lego-Roboter so zu programmieren.

Finale der Ferngesteuerten

Die Liga der ferngesteuerten Roboter geht ins Finale. Die »KIT-Robotics« treten gegen den »EKG-Clan« des Erich-Klausener-Gymnasiums aus Adenau an. Und während sich der »Spieler« der »KIT-Robotics« durch eine breite Schiebefläche auszeichnet, besticht der Roboter des EKG-Clans durch seine Schlagvorrichtung. Ein spannendes Match, das nach den für den Wettbewerb üblichen sechs Minuten mit einem drei zu zwei für die Adenauer endet. Die Spiele der autonomen Liga beginnen. »Die Roboter werden mit Hilfe von zwei Kameras über dem Fußballfeld gesteuert«, erklärt Projektleiter Michael Wülker, Professor an der Hochschule, die Funktionsweise. Anhand der Farbscheiben auf den Robotern ermittelt die Kameras, wo sich der Roboter und der Fußball befindet. Die Daten werden per In-

frarot an die »Spieler« gesendet, die sie auswerten und sich nach vorprogrammierten Mustern bewegen.

Schiedsrichter Daniel Fuchs, Student an der Hochschule, pfeift das erste Spiel an. »Banane flanken« und »Willibald« fahren los. »Banane flanken« kreist um den Ball, ohne ihn zu finden. »Willibald« dreht sich im Kreis und bleibt an der Glaswand hängen. Da kommt »Banane flanken« an den Ball und schießt das erste Tor. Die Zuschauer jubeln. Es geht weiter. »Willibald« verliert einen Baustein und will das Feld verlassen. Halb hängt er schon über der Glasabsperrung, als »Banane flanken« das zweite Tor schießt. Fünf zu Null endet das Spiel zwischen den Siemens-Lehrlingen aus Freiburg und der Heimschule Lender in Sasbach.

Mitte Januar trafen sich die Teams zum ersten Mal in der Fachhochschule. Dort wurde ihnen die Aufgabe vorgestellt. »Wir haben die Einladung an sämtliche Schulen der Ortenau geschickt«, erzählt Wülker. Gemeldet hätten sich Teams von Lörrach bis Adenau in der Eifel. Jedes Team erhielt einen Lego-Mindstorms-Baukasten und Unterricht in der Programmiersprache »Java«. Der Wettbewerb richtete sich nach den Regeln der internationalen »Robocup-Eleague«. Betreut wurden die Schüler von Wülker

und zwei Ingenieuren. Der Wettbewerb steht unter der Schirmherrschaft der IHK Südlicher Oberrhein.

»Die Schüler sollten sehen, wie wichtig der Bereich Informationstechnik ist«, so Wülker. Außerdem sei der Wettbewerb für die jungen Leute eine Möglichkeit, die Offenburg Hochschule kennen zu lernen. Dass so wenige Mädchen am Wettbewerb teilnehmen, bedauert der Hochschullehrer. Um künftig auch die weibliche Welt anzusprechen, plant die Hochschule deshalb einen Roboter-Wettbewerb für Tanz, Musik und Theater.

Die Gewinner

■ Liga der ferngesteuerten Roboter:

1. »EKG-Clan« des Erich-Klausener-Gymnasiums Adenau, 2. »KIT-Robotics« der Kaufmännischen Schulen Offenburg, 3. »Eifel-Storms« des Erich-Klausener-Gymnasiums Adenau.

■ **Autonome Liga:** 1. »LittleBigFootball« des Technischen Gymnasiums Rastatt, 2. »Ballmuesnütze« der GHSE Emmendingen, 3. »Banane flanken« der Lehrlinge der Siemens Professional Education Freiburg.

■ **Originellster Roboter:** Roboter »Käfer« des Teams der 7a des Robert-Gerwig-Gymnasiums in Hausach.

Offenburger Tageszeitung 17. März 2005



So sehen Gewinner aus: Technik-Cracks vom Rastatter TG.

Foto: pr

Schüler des Technischen Gymnasiums holen ersten Preis bei FH-Wettbewerb

Fußball spielende Roboter

Rastatt (red) – Schüler des Technischen Gymnasiums Rastatt haben den ersten Preis in einem Roboterwettbewerb gewonnen. Jan Issac und Stefan Maier (TGI 1), sowie Maik Fox und Thomas Kristof (TGI 2) haben dafür zunächst an der Fachhochschule Offenburg vier Einführungsvorlesungen über die Java-Technologie gehört, wie es in einer Mitteilung der Schule heißt. Im Anschluss daran wur-

den acht Testläufe durchgeführt. Ziel dieses von der FH Offenburg initiierten Wettbewerbs war es, einen autonomen Fußballroboter aufzubauen und zu programmieren. Dieser erhält über Infrarot-Verbindung die Koordinaten des Gegners und des Balls sowie die Schiedsrichterentscheidungen. Das Fußballfeld ist mit einem Bilderfassungssystem ausgestattet. Die Roboter der sechs zum

Wettbewerb angetretenen Gruppen mussten in Spielen von jeweils zwei mal vier Minuten gegeneinander antreten. Insgesamt hatten sich 24 Gruppen angemeldet, von denen sich jedoch nur 14 tatsächlich dem Wettbewerb stellten. Diese wurden in zwei Ligen eingeteilt, wobei die Liga „ferngesteuerter Roboter“ aus acht Gruppen, die Liga „autonome Roboter“ aus sechs Gruppen bestand.

Badisches Tagblatt (Ausgabe Rastatt) 26. März 2005

Literatur / Quellen

- [1] **Regeln der E-Liga für die Hochschule Offenburg 2005**
<http://mv-sirius.m.fh-offenburg.de/robotik/SW05Material/ELeagueRegelnFHO.htm>
→ auch als Anlage enthalten
- [2] **Wettbewerbsregeln und Umfeld der Hochschule Offenburg 2005**
<http://mv-sirius.m.fh-offenburg.de/robotik/SW05Material/UmfeldUndRegeln-2004-ws.pdf>
- [3] **Homepage des Wettbewerbs**
<http://mv-sirius.m.fh-offenburg.de/robotik/>
- [4] **Homepage des LejOS-Projekts**
<http://www.lejos.org/>
- [5] **Homepage des eclipse-Projekts**
<http://www.eclipse.org/>