Diplom Informatik - Intelligente Systeme
Fachhochschule Brandenburg
Betreuer: Dipl.-Inform. Ingo Boersch

Projekt “Künstliche Intelligenz” Teilprojekt Autonome Mobile Systeme Hasenjagd 2007

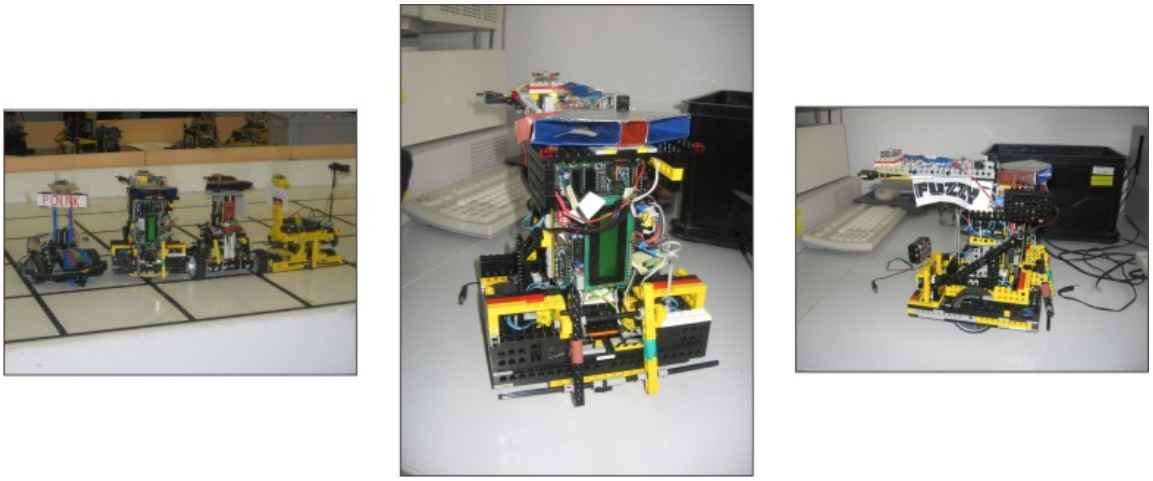
22. Juni 2007

von Brandolf Gumz, Sascha Krieg

Vorstellung

Gestatten FUZZY: Ich bin ein FUTURISTISCHER UNIVERSELLER ZOOLOGISCHER ZIELORIENTIERTER YUPPIE. Meine Aufgabe ist es andere meiner Art die sich als Hasen ausgeben zu fangen oder vor anderen Füchsen zu fliehen wenn ich im Hasenmodus über das Spielfeld fege. Zur Ortung meiner Kontrahenten besitze ich einer 3-Zonen Trichter, meine “Augen”, sobald ich Jemanden ins Visier genommen habe lasse ich ihn nicht mehr so schnell außer Sicht. Des weiteren haben mich meine Erbauer mit einem Getriebe ausgestattet welches mir eine rasante Beschleunigung und eine hohe Geschwindigkeit ermöglicht. Meine Überlebensstrategie ist recht einfach aufgebaut: Umgebung erfassen, Gegner visieren und los geht es.

Abbildung 0.1: Fotos von FUZZY



Hardware

Antriebsart

FUZZY besitzt zwei 12 Volt Motoren. Die jeweils auf eine Achse übertragen werden und die Räder antreiben. Beide können einzeln angesteuert werden.

Wendigkeit und Steuerbarkeit

- kann gleichermaßen vorwärts und rückwärts fahren
- Kann sich auf der Stelle drehen oder Kurven fahren
- wird über das AKSEN-Board programmiert und gesteuert
- per DIP-Schalter läßt sich zwischen den beiden Modi Hase und Fuchs wechseln
- per DIP-Schalter kann man die Dauer der Spielzeit wählen (60s oder 120s)
- über ein Signal an einem Digital-Port wird der Startcountdown ausgelöst

besondere Elemente

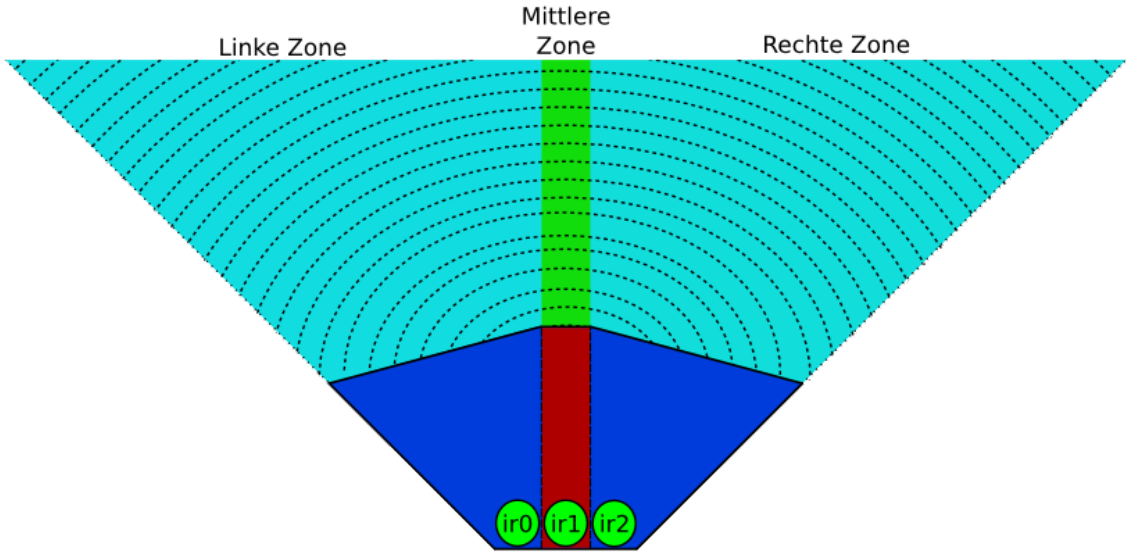
- selbst gebauter Trichter mit 3 Zonen, der aus Pappe und Isolierband besteht, und zur Ortung des Gegners dient
- ein komplett drehbarer Turm, der es ermöglicht den 3-Zonen-Trichter in die gewünscht Richtung zu drehen
- Stoßstange, realisiert über 2 Taster die das Detektieren einer Wand / Roboter beim anfahren ermöglicht

Sensoren

Typen

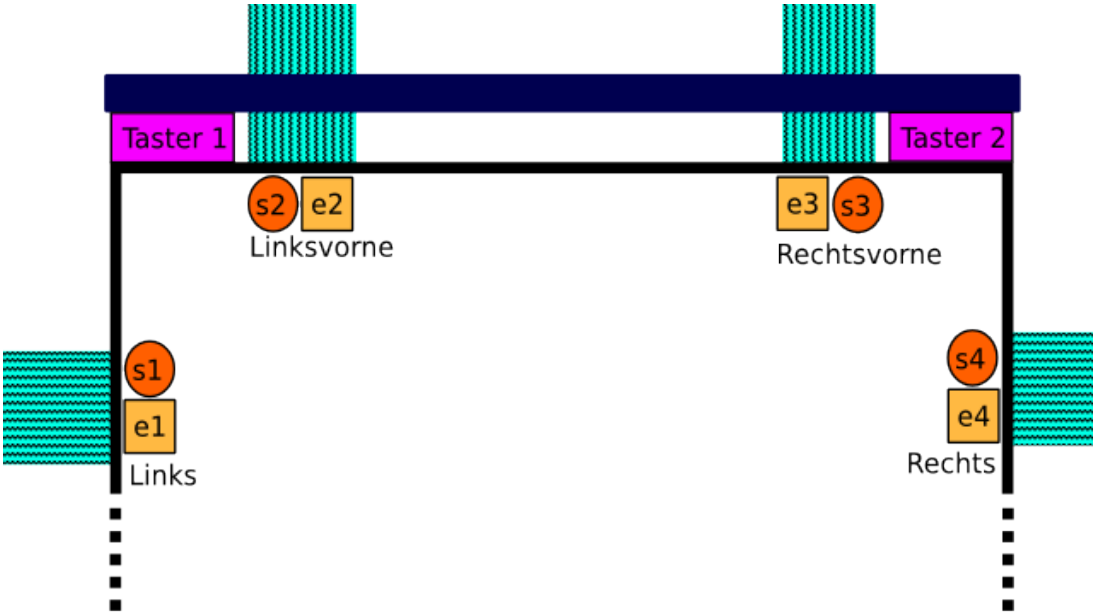
- 4 IR-Empfänger für die Wanddetektion
- 4 IR-Sender ebenfalls für die Wanddetektion
- 2 Taster für die Stoßstange
- 1 Beacon um das Hasen- / Fuchssignal zu senden
- 3 IR-Empfänger für die Erfassung des Gegners
- 2 LEDs um den aktiven Modus zu signalisieren

Abbildung 0.2: Anordnung der Sensoren im 3-Zonen-Trichter



- Die 3 IR-Empfänger (ir0, ir1, ir2) sind im 3-Zonen-Trichter montiert. Diese empfangen IR-Signale aus den jeweiligen Zonen(Linke Zone, Mittlere Zone, Rechte Zone)

Abbildung 0.3: Anordnung der Sensoren für die Wanderkennung



- Die 2 Taster (Taster 1, Taster 2) sind zwischen Stoßstange und Gehäuse montiert
- Die 4 normalen IR-Empfänger (e1, e2, e3, e4) sind mit den IR-Sendern (s1, s2, s3, s4) jeweils Paarweise übereinander montiert, so dass sie in die Richtungen Links, Links vorne, Rechts vorne und Rechts IR-Signale abstrahlen und empfangen können.
- Der Beacon befindet sich in 25cm Höhe am hinteren Teil des Turmes.

Erfassung und Vorverarbeitung der Sensordaten

Die Sensordaten werden alle auf einmal erfasst und in globalen Variablen zwischengespeichert. Für die Detektion der Wanderkennung verwenden wir eine Differenzmessung. D.h. wir ermitteln einmal das reine IR-Signal welches der Empfänger erhält und in einer zweiten Messung schalten wir zusätzlich den IR-Sender an dessen Signal wenn eine Wand vor dem Empfänger ist reflektiert wird. Durch Differenzbildung dieser 2 Messwerte können die Wände relativ sicher detektiert werden. Die Taster der Stoßstange sowie die Empfänger des “Trichters” können direkt ausgelesen werden.
Wenn wir alle Sensordaten erfasst haben werden diese in einer separaten Programmroutine für die Steuerung der Motoren ausgewertet.

Lösungsweg

Arbeitsaufteilung innerhalb des Projektes

Brandolf Gumz: Chefdesigner, Co-Programmierer & Logiker
Sasch Krieg: Chefprogrammierer, Co-Designer & Stratege

Zeitlicher Ablauf

Tabelle 0.1: zeitlicher Ablauf (chronologisch)

Dauer	Tätigkeit
2 Wochen	Prototyp des Roboters gebaut, erste Probefahrt
1 Woche	geradeaus fahren verbessert, Drehbewegungen (links, rechts, 180°) implementiert
1 Woche	Einbau und Test der Sensoren für die Wanderkennung,
3 Woche	Beacon getestet, Trichter für die IR-Empfänger gebaut, erster Testlauf ob Signal des Gegners empfangen wird
1 Woche	IR-Empfänger abschirmen so dass er nicht von unserem eigenen Sender gestört wird
1 Woche	Roboter zentriert sich auf das Gegnerische IR-Signal
1 Woche	Prototyp-Gegner zum testen gebaut, Problem wenn der Roboter herumfährt fängt der Motor nach einiger Zeit an zu stottern, Fehlersuche begonnen
1 Woche	Fehlersuche Motorstottern: Board getauscht, Motor getauscht, Strom gemessen, Überhitzung eines Bausteines festgestellt, Programmcode optimiert so dass die Motoren nicht mehr ständig an und abgeschaltet werden
1 Woche	Roboter neu gebaut: neues Getriebe und einen drehbaren Turm
1 Woche	Stoßstange gebaut und getestet, Programm neu strukturiert so dass alle Sensorsignale als globale Variablen verfügbar sind
1 Woche	Logik für den Fuchs programmiert
1 Woche	Logik für den Hasen programmiert

Ideen und Strategien

Der Fuchs fährt über das Spielfeld durch den relativ großen Einfallswinkel des Trichters wird der Gegner in den meisten Fällen schnell geortet.Wenn der Hase geortet wird versucht der Fuchs in zu zentrieren. Dazu wird während der Fahrt zu dem Hasen ständig die Richtung so angepasst das dass IR-Signal des Hasen's von dem IR-Empfänger in der Mitte unseres Trichters empfangen wird. Wenn das Signal zentriert ist wird direkt auf den Hasen zugesteuert dabei werden die Wände ignoriert.
Die ursprüngliche Idee des Hasen war es den Turm um 180° zu drehen um den Fuchs hinter uns zu orten und daraufhin in die richtige Richtung zu fliehen. Diese Idee haben wir jedoch verworfen da wir zu oft an die Bande gedrängt wurden und außerdem die Programmierung etwas komplizierter wurde. Unsere letztendliche Lösung bestand darin das empfangene Signal des Fuchses als Wand zu interpretieren sodass der Hase einfach durch die Gegend fährt und allen Wänden ausweicht.
Verworfen hatten wir zudem relativ Frühzeitig die Idee das wir die Drehung des Roboters mithilfe eines Optokopplers an einem Rad mit einer Kreisskala ermitteln. Die Drehungen wurden einfach über die zeitliche Ansteuerung der Motoren realisiert. Im späteren Verlauf haben wir festgestellt das wir nur in seltenen Fällen eine fest implementierte Drehbewegung benötigen da wir einfach die Geschwindigkeit oder Drehrichtung der Motoren ändern und wiederum die neu ermittelten Sensordaten verarbeiten und auf diese reagieren. Das bedeutet zum Beispiel das wenn wir vor uns eine Wand detektieren wir die Motoren so ansteuern das wir uns nach links wegdrehen wenn die Sensoren Vorne keine Wand mehr detektieren werden die Motoren wieder so angesteuert das man geradeaus fährt.

Probleme

- Getriebe: Stangen verkanten sich, Zahnräder lockern sich, Legosteine lockern sich es muss immer mal wieder nachjustiert werden
- Sensoren: Kabel reißen schnell ab, eigener Beacon stört die Empfänger auch die Abschirmung und Verlegung des Beacons nach hinten löst das Problem nicht vollständig
- Aufgrund der zu hohen Stromaufnahme der beiden Motoren wurde ein Bauteil zu heiß Motor fing an zu stottern, Behebung durch Verteilung der Motoren auf jeweils einen Baustein dadurch höhere Stromaufnahme ohne Überhitzung möglich
- Testen des Codes recht aufwändig, kein Debugger und Fehlermeldungen in cygwin schwer lesbar, teilweise Abhilfe dadurch das wir alle empfangenen Signale ständig ausgeben

Software

Programm-Code

Das gesamte Programm befindet sich in einer einzigen Datei (hasenjagdMain.c) und umfasst um die 500 Zeilen Code. Die Ports für die Taster, IR-Empfänger, Motoren etc. haben wir als defines deklariert und alle zu interpretierenden Sensordaten als globale Variablen festgelegt. In der Methode “detektiereWand()” ermitteln wir die IR-Sensordaten für die Wanderkennung und die Signale der Stoßstange. Die Methoden “fuchs_init()” und “hase_init()” dienen zum initialisieren der beiden Modi. Dabei wird die richtige Frequenz für die IR-Aktoren / Sensoren gesetzt sowie die LED die den Modus signalisiert. Daneben gibt es noch jeweils eine Methode für den Start-Countdown, die Sensorabfrage der IR-Empfänger für den Trichter, eine Ausgabe Methode sowie unterschiedliche Möglichkeiten sich zu drehen. Den Hauptbestandteil bilden die beiden Methoden “fuchs()” und “hase()” die das ansteuern der Motoren regeln um Wänden auszuweichen und den gegnerischen Hasen zu verfolgen.

Schwierigkeiten im Wettbewerb

Eine Stunde vor Wettbewerbsbeginn ist uns ein Transistor am AkSen-Board durchgeschmort weshalb wir das Board austauschen mussten was bei uns unglücklicher weise nicht ganz einfach vonstatten geht. Nach dem Wechsel des Boards hatten wir zudem vergessen die Stoßstange zu überprüfen, welche wir, wie sich später herausstellte, an einem falschen Port angeschlossen haben. Die erste Hälfte des Rennes verlief für uns gut dann hat sich aber leider das ausgetauschte Board verabschiedet sodass wir alle folgenden Wettbewerbe verloren haben und uns mit einem dritten Platz abfinden mussten.