Technische Dokumentation des Roboters „Zephyr“ - RoboCup 2016

Teammitglieder: David Bailey, Felix Diekmann

Steuergerät:
Das Steuergerät des Roboters ist ein auf dem ATMega328-P basierendes, selbst gebautes Controller-Board. Der MCU läuft auf 16 Mhz, und wird mithilfe eines STK500v2 über SPI programmiert.
Ebenfalls auf dem Board befindlich ist ein 5v Spannungsregler zur Stromversorgung, Anschlüsse für vier Sensoren (analog/digital), sowie drei Pololu A4988 Stepper Motor Treiber zur Steuerung der Schrittmotoren, wobei das Microstepping über einige Schalter angepasst werden kann. Zusätzlich können weitere Module mithilfe von TWI (bzw. I2C) angeschlossen werden.

Sensorik:
Die Sensorik des Roboters besteht aus drei DFRobot SEN0017 Digital Line Tracking Sensoren, sowie einem Endstop Switch.
Die Line Tracking Sensoren wurden am vordere Ende des Roboters angebracht und dienen dem Line-Following. Sie sind direkt am Controller-Board angeschlossen, und liefern digitale Daten über das (nicht) vorhanden sein der Linie an der jeweiligen Position.
Der Endstop Switch ist oberhalb der Line Tracking Sensoren angebracht, und dient der Erkennung von Hindernissen, um diese umgehen zu können.

Aktuatorik:
Die Aktuatorik des Roboters besteht aus drei in 120° voneinander gedrehten Positionen angebrachten Schrittmotoren (2x Pololu 35x36mm, 2.7V, 1A / Phase, 1x Pololu 35x26mm, 7.4V, 0.28A / Phase), an denen eigens erstellte und 3D gedruckte Omni-Wheels angebracht wurden. Sie werden über die auf dem Controller-Board befindlichen Stepper Motor Controller mit Strom versorgt und gesteuert.
Als Motor für den Greifarm wird ein HobbyKing HK15148B Digital Servo verwendet.

Software:
Die Software für den ATMega328-P wurde komplett in C++ geschrieben, und verwendet ausschließlich eigene, speziell für den Roboter geschriebene Bibliotheken. Als IDE dient Eclipse Luna mit AVR Plugin, um kompilieren mit dem AVR-GCC Compiler zu ermöglichen. Als VCS dient Github (https://www.github.com/Xasin/Hexa-Bot)
Die 3D-Modellierung aller Teile ist mit OpenSCAD geschehen. Auch hier wurde einzig selbst geschriebener Code verwendet.

Spezielle Hinweise:
Aktuatorik:
Die spezielle Aktuatorik des Roboters, bestehend aus den drei in 120°-Winkeln angeordneten Schrittmotoren, erlaubt für einen zentimetergenauen omnidirektionalen Antrieb. Dieser erlaubt es dem Roboter, sich komplett frei auf einer zweidimensionalen Oberfläche zu bewegen und sich problemlos zu Drehen, vorwärts/rückwärts und seitwärts zu fahren.
Dies macht das gesamte System sehr mobil und wendig, was viele Aufgaben, welche Präzision benötigen, einfacher gestaltet.
Software:
Um eine solche Bewegungsfreiheit voll ausnutzen zu können wurde die Software auf eine Art und Weise geschrieben, welches die Steuerung um ein vielfaches vereinfacht. Hierbei werden die einzelnen Schrittmotoren jeweils von einem eigenen Objekt angesteuert, welches über einen 5kHz ISR eine simple Geschwindigkeitssteuerung bietet und für einen ruhigen Lauf sorgt. Diese simplen Schrittmotor-Objekte werden von einer Controllsoftware mit ca. 150Hz angesteuert. Diese komplexere Software übernimmt die Berechnung von X und Y Geschwindigkeit um eine konstante lineare Geschwindigkeit zu realisieren, und speichert dabei die momentane Position ab, welche sie mit der Soll-Position vergleichen kann.
Dies erlaubt es dem Benutzer die Motoren durch simple Angabe von X und Y Koordinaten in Millimeter, bzw. Rotation in Grad zu steuern, sowie Geschwindigkeiten in mm/s und Grad/s ein zu stellen.

Zudem wurde der Code so flexibel wie möglich strukturiert. Objekte wurden, wenn möglich, verwendet. So z.B. gibt es eine bestimmte Klasse, welche nur für die Verwertung der Line Tracking Sensoren verwendet wird. Sie basiert auf einer Base-Klasse, und wird lediglich durch Pointer in anderen Objekten und Funktionen verwendet. Dies erlaubt es, schnell und ohne viel Mühe einen anderen Sensortyp ein zu bauen. Lediglich ein anderes Objekt muss instanziert und übergeben werden.
Auch in der Main-Loop wird dort lediglich die Funktion eines Pointers zu einem bestimmtem Objektes ausgeführt. Dieser Pointer kann dann während einer ISR umgeändert werden, sodass nun ein anderer Code ausgeführt wird. Dies erlaubt eine sehr flexible und verständliche Struktur, da jede Funktion eine bestimmte Aufgabe übernehmen kann, ohne dabei andere Funktionen zu beeinflussen.