"If you don't want a generation of robots, fund the arts!" — Cath Crowley, Graffiti Moon

Roboter mit LeapMotion

Dokumentation

Mattias Abfalterer, Simon Bodner, Alexander Hoppichler

Inhaltsverzeichnis

Allgemeines

Unser Projekt „LeapMotionRobot“

Erklärung

Ein von uns entworfener Roboter soll ferngesteuert werden können. Als Antrieb dienen zwei Gleichstrommotoren, welche über einen Arduino angesteuert werden können. Ebenfalls zum Ansprechen der Motoren wird ein Motorshield benötigt, welches als Steckbrett verwendet wird und auf den Arduino hinaufgesteckt wird. Für die Kommunikation wird Aurduinoseitig ein Bluetoothshield verwendet, welches die Bewegungsdaten in Empfang nimmt.
Die Fernkommunikation erfolgt clientseitig mittels einer LeapMotion (wird noch erklärt), die über Bluetooth Bewegungsdaten der Hand weiter an das Bluetoothshield sendet.

Welche Anforderungen sollen erfüllt werden?

Der Roboter soll nur mithilfe einer Hand gesteuert werden. Je nach Neigung der Hand (nach links oder rechts bzw. nach vorne oder hinten) soll der Roboter entweder nach links oder rechts bzw. nach vorne oder zurück fahren. Ist die Hand nicht geneigt, soll der Roboter stehen bleiben.
In weiterer Folge soll noch realisiert werden, dass der Roboter mit verschiedenen Geschwindigkeiten fahren kann.
Des Weiteren soll der Roboter gegen kein Hindernis (Mauer, Stuhl, Tisch, etc.) fahren.

Vorgehensweise

Zu Beginn musste recherchiert werden, was alles notwendig war, um das geplante Projekt starten zu können. Dazu wurde auch ein Pflichtenheft erstellt. Dann wurden die Arbeiten aufgeteilt in Arduino – Programmierung, C# - Programmierung (für LeapMotion) und zusammenfügen der Hardware.
Anfangs wurden Testprogramme geschrieben und getestet, diese wurden dann schrittweise zusammengefügt. Fehler wurden solange ausgebessert, bis die Muss-Kriterien erfüllt waren.
Danach wurden noch die Kann-Kriterien ausgearbeitet und in die bereits existierenden Programme implementiert.

Die Hardware

LeapMotion

Die LeapMotion ist ein Sensor, entwickelt von der gleichnamigen Firma, welcher die Neigung, Position, Konturen und Bewegungen einer menschlichen Hand erkennen kann. Über eine spezielle Software werden die angeführten Aspekte am Desktop angezeigt.
Die LeapMotion kann man mit einer Kinect – Steuerung (für die Spielekonsole Xbox) verglichen werden.

Der Arduino

Für unser Projekt wird ein Arduino Uno verwendet.
Dieser Mikrocontroller bekommt Bewegungsdaten der Hand über das Bluetoothshield, welche dann der Arduino in Steuersignale für die Motoren umwandelt. Diese Steuersignale gibt der Arduino danach weiter an das Motorshield.

Das Motorshield
Dies ist eine spezielle Hardware, welche die Steuersignale, die vom Arduino weitergegeben wurden, verarbeitet und an die Motoren weiterleitet. Je nach Signal werden beide Motoren eingeschaltet, nur der rechte Motor oder nur der linke Motor.

Das Bluetoothshield
Ist die Schnittstelle zwischen dem Mikrocontroller (Arduino) und dem PC. Das Bluetoothshield nimmt die Daten vom PC entgegen und gibt diese an den Arduino weiter.

Der Distanzmesser
Der Distanzmesser ist ein Infrarotsensor, welcher ein Hindernis erkennen soll. Durch einen Infrarotstrahl wird die Distanz zu einem Hindernis gemessen. Je nach Abstand zum Hindernis wird durch den Distanzmesser eine LED zum Leuchten gebracht, welche signalisiert, dass das Hindernis nicht mehr weit entfernt ist. Wird der Abstand noch kleiner, dann wird der Roboter zum Stillstand gebracht und dieser kehrt dann automatisch um.

Der „Roboter“
Alle Teile zusammengesetzt, ausgenommen von der LeapMotion, inklusive anderer benötigter Hardware (Batterien, Kabel, Räder, etc.) ergeben den Roboter.

Die Software

C#

Erklärung
Auf Basis von C# wurde eine Software entwickelt, welche die Gesten der Hand über dem LeapMotion-Controller einliest, interpretiert und über Bluetooth an den Arduino sendet.

Anforderungen an das Programm
Am Anfang soll das Programm eine Bluetooth-Verbindung mit dem Roboter (Arduino) aufbauen, wobei der ausgehende Port des PCs vom Benutzer eingegeben werden kann. Das Programm ist ein Konsolenprogramm.
Die Position der Hand wird mit ihren konkreten Positionswerten laufend eingelesen. Aufgrund dieser Werte soll in Folge ein Code generiert werden, welcher an den Arduino gesendet wird.
Natürlich ist auch Exception-Handling einzubauen, welches absichert, dass das Programm im Falle von nicht vorgesehenen Benutzerinteraktionen nicht abstürzt.

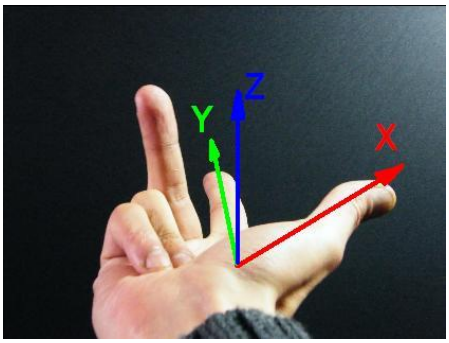
Code
Dass eine Kommunikation zwischen zwei oder mehreren Geräten über einen speziellen Code funktionieren muss, ist wahrscheinlich jedem klar.

Von uns wurde für die Kommunikation LeapMotion (bzw. PC) und Arduino dafür ein eigener Code entworfen, der im C#-Programm zusammengestellt und über Bluetooth gesendet wird.

Erklärung/Interpretation des Codes

Der im C#-Programm erstellte und im Arduino-Programm interpretierte Code sieht wie folgt aus:
AxxxBxxx
An erster Stelle, also zwischen A und B, steht die Zahl für die Geschwindigkeit, welche abhängig von der Neigung der Hand ist.

Zur Erklärung des Codes werden folgende Achsen der Hand herangezogen:
Ist die Hand mit den Fingerspitzen maximal nach unten und mit dem Arm maximal nach oben geneigt, d.h. Fingerspitzen in Richtung negativer Z- und Arm in Richtung positiver Z-Achse, so ist der erste Wert zwischen A und B 0. Umgekehrt, wenn die Fingerspitzen maximal in Richtung positiver Z- und der Arm in Richtung negativer Z-Achse geneigt ist, so ist der Wert 200. Befindet sich die Hand genau in Position der Y-Achse, also ist die Hand in Richtung Y-Achse gerade, so ist der Wert 100.



Der zweite Wert nach B ist analog dazu aufgebaut. Dieser gibt an, ob, in welche Richtung und wie stark gelenkt werden soll. Ist die Hand maximal nach links geneigt, d.h. die X-Achse maximal in Richtung positiver Z-Achse, so ist der Wert 0. Umgekehrt ist der Wert, wenn die Hand maximal nach rechts, d.h. die X-Achse maximal in Richtung der negativen Z-Achse geneigt ist, 200. Ist die Hand gerade, also in Richtung der X-Achse waagrecht, so ist dieser Wert 100.
Die Zahl 100 beschreibt also für beide Fälle das Neutrum.

Um eine gute Steuerung gewährleisten zu können, muss um den Neutralbereich (100) eine Toleranzzone eingebaut werden, da es menschlich kaum möglich ist, die Hand exakt waagrecht zu halten. Ebenso muss ab einer gewissen Neigung alles darüber bzw. darunter als Maximum betrachtet werden, da sonst ein einwandfreies Funktionieren des LeapMotion-Controllers nicht gewährleistet ist. Diese Toleranz- bzw. Maximumsgrenze wurde ausgemessen.
Sinnvolle Werte:
Toleranzbereich Pitch (Vor/Zurück): 10
Toleranzbereich Roll (Lenken): 12
Maximalwert Pitch: 50
Maximalwert Roll: 50
Der Code ist ein Prozentwert, da jeder Teil auf einer Spanne von 100 beruht. Das heißt, in der Berechnung des Codes müssen für Pitch die Spanne von 40 auf 100, für Roll die Spanne von 38 auf 100 ausgeweitet werden.

Arduino

Erklärung

Mithilfe der Arduino-Entwicklungsumgebung wurde ein Programm entwickelt, welches die Bewegungsdaten vom PC entgegennimmt, interpretiert und in Bewegungen umsetzt.

Anforderungen an das Programm

Am Beginn soll das Programm eine Bluetooth-Verbindung zum PC aufbauen. Das Programm liest laufend die Daten ein und verarbeitet diese. Direkt nach dem Einlesen werden die 2 wesentlichen Informationen in 2 separaten Zwischenvariablen gespeichert. Die 2 Informationen sind die Geschwindigkeit und die Lenkung. Anhand der Variablen wird bestimmt wie die Motoren angesteuert werden. Falls die Geschwindigkeitsvariable über 100 ist fährt der Roboter vorwärts. Je nachdem wie weit der Wert über 100 ist fährt der Roboter eine schnellere Geschwindigkeit.

Analog dazu fährt der Roboter rückwärts wenn der Wert unter 100 ist. Umso näher bei 0 der Wert ist umso schneller fährt der Roboter rückwärts.

Die 2. Variable ist die der Lenkung. Bei einem Wert von 100 fährt der Roboter geradeaus (neutrale Position). Wenn der Wert kleiner als 100 ist macht der Roboter eine Linkskurve. Proportional zur Neigung wird auch der Kurvenradius bestimmt. Dasselbe auf der rechten Seite bei Werten über 100.

Zusätzlich zum Einlesen, Interpretieren und Verwerten des Codes wird überprüft ob sich der Roboter vor einem Hindernis befindet. Falls er nahe einem Hindernis ist, beginnt eine LED zu leuchten bis sich der Roboter weit genug vom Hindernis entfernt. Falls er sich zu nahe an einem Hindernis befindet und kurz vor einer Kollision steht beginnt die LED zu blinken und eine automatische Routine zum Umdrehen wird durchgeführt. Dabei bleibt der Roboter kurz stehen, fährt dann zurück und macht eine 180°-Drehung.

Testläufe

Testlauf 1

Beim 1. Testlauf wurde getestet, ob das 1. Prototypprogramm am Arduino läuft. Dieses soll die Motoren links, rechts, vor und zurück fahren lassen. Dazu wurden im Programm Testdaten erzeugt und verwendet. Über den Serial Monitor wurde anschaulich gemacht, welcher Motor bei welchen Daten welches Signal bekommen würde. Sobald diese Routine funktionstüchtig war, wurde der Code für weitere Zwecke verwendet.

Testlauf 2

Im zweiten Testlauf wurde die Routine des Arduinos bereits mit den Motoren getestet. Dazu wurden Testdaten erzeugt, die schon den fertigen Code, wie er von der LeapMotion kommen soll, simulieren. Anfangs kamen einige Schwierigkeiten zum Vorschein, da die Verarbeitung nicht richtig funktionierte. Nach Fehlersuche und Programmänderungen kamen wir schlussendlich zum gewünschten Ergebnis und die Motoren taten das, was sie laut Routine machen sollten.
Der Roboter konnte nun:

- Sich vor bewegen
- Nach links und rechts fahren

Testlauf 3

In diesem Test wurde die Bluetooth – Connectivity getestet. Auduinoseitig und auch am PC wurden dazu jeweils ein Programm geschrieben, welche mit den Hauptprogrammen zusammengefügt wurden. Das C# - Programm soll Testdaten erzeugen und über Bluetooth an den Arduino (genauer: zum Bluetoothshield) senden. Dieser nimmt die Daten entgegen und verarbeitet diese so, dass die Motoren das machen, was sie machen sollen.
Das Senden und Empfangen der Daten funktionierte bereits beim ersten Versuch einwandfrei, nur die Verarbeitung am Arduino schlug noch fehl.
Der Fehler konnte jedoch rasch behoben werden. Somit konnten die Motoren des Roboters über den PC angesprochen werden.

Testlauf 4

Nun wurde versucht, über die LeapMotion den Roboter bzw. dessen Motoren anzusprechen. Ein C# - Programm soll die Daten der LeapMotion übernehmen, diese so verarbeiten, dass sie dem von uns entworfenen Code entsprechen (in der Form AxxxBxxxC) und anschließend an den Arduino übermitteln.
Anfangs ergaben sich noch einige Probleme, da der Roboter überhaupt nicht das machte, was er machen sollte.

Nachdem das Hauptprogramm des Arduinos umgeschrieben wurde, funktionierte die Routine einwandfrei. Der Roboter konnte nun über die LeapMotion ferngesteuert werden, somit war unser SOLL-Kriterium erfüllt.

Testlauf 5

Vor diesem Testlauf wurden diverse Programmverbesserungen vorgenommen. Nun war es möglich, je nach Neigung der Hand schneller bzw. langsamer zu fahren. Je weiter die Hand nach vorne geneigt wird, desto schneller fährt der Roboter nach vorne.
Ebenfalls in die Routine mit eingebaut wurde die „Lenkintensität“. Je weiter die Hand zur Seite geneigt ist (in beide Richtungen), desto enger wird der Kurvenradius gewählt.
Der Test lieferte uns ein positives Ergebnis, somit war schon ein erstes KANN-Kriterium erfüllt worden.

Testlauf 6

Beim 6. und letzten Testlauf wurde der Infrarotdistanzmesser am Roboter angebracht. Fährt der Roboter auf ein Hindernis (Wand, Tür, etc.) zu, so erkennt dies der Sensor und hält den Roboter an. Des Weiteren dreht sich der Roboter selbstständig um, wenn er vor einem Hindernis steht. Aufgrund des erfolgreichen Testlaufes wurde ein weiteres KANN-Kriterium erfüllt.

Am Roboter wurde ebenfalls eine LED angebracht. Diese fängt zu blinken an, sobald sich der Roboter einem Hindernis nähert. Sobald der Roboter kurz vor dem Hindernis ist, bleibt dieser automatisch stehen, wie oben beschrieben, und gleichzeitig leuchtet die LED durchgehend. Dieser Test verlief ebenfalls nach Wunsch und somit wurde das letzte KANN-Kriterium erfüllt.

Probleme während der Projektentwicklung / Projektentstehung

Während wir an unserem Projekt arbeiteten, kristallisierten sich einige Probleme heraus.

- **Hardware**
  Der Transport unserer Hardware brachte oft Schwierigkeiten mit sich, denn die Kabel lösten sich bzw. die Lötstellen hielten nicht. Deshalb mussten wir des Öfteren die zerstörten Stellen neu verlöten.
- **Software – Arduino**
  Sehr große Probleme bereitete uns die Implementierung des Programmes für die Kollisionssicherheit in das Hauptprogramm. Durch eine Doppelbelegung eines Pins vom Arduino (PIN 13) funktionierte das Hauptprogramm nicht mehr. Dieser Fehler konnte erst nach einigen Überlegungen entdeckt werden.
- **Kommunikation zwischen den Geräten**
  Das Senden der richtigen Daten von der LeapMotion zum Arduino über das Bluetooth bereitete uns Schwierigkeiten, da ein gewisser Takt sowohl softwareseitig im C# - Programm, als auch am Arduino vorgegeben werden müssen. Diese Taktfrequenz musste optimal angepasst werden.
- **Fahrgeschwindigkeit beim Rückwärtsfahren**
  Aufgrund der Fahrgeschwindigkeit beim Rückwärtsfahren fängt der Roboter an zu „ruckeln“. Dieses Problem liegt immer noch vor, wie dem entgegengewirkt werden könnte → siehe Erweiterungsmöglichkeiten

Erweiterungsmöglichkeiten

Für unseren Roboter steht eine Reihe von Erweiterungsmöglichkeiten offen. Mögliche Punkte wären:

- Verschönerung des Aussehens
- Hinzufügen eines Stützrades im hinteren Bereich → ruhigere Fahrweise, v.a. beim Rückwärtsfahren
- Einbau von mehreren Infrarotsensoren bzw. Servomotor
  Momentan schaut unser Distanzmesser nur gerade nach vorne, d.h. er bemerkt schräge Hindernisse nicht. Es könnten entweder mehrere Infrarotsensoren eingebaut werden, oder dieser eine Sensor mithilfe eines Servomotors um die Kurve schauen. Der Sensor soll dann laufend von links nach rechts und retour gedreht werden, um auch seitliche Hindernisse beim Anfahren im spitzen Winkel bemerken zu können.
- Einbau eines Lautsprechers, um Alarmtöne bei geringer Distanz zusätzlich zur leuchtenden bzw. blinkenden Led auch akustisch ausgeben zu können.
- Wird die Steuerhand aus dem Blickfeld der LeapMotion entfernt, so soll der Roboter stehen bleiben
- Mittels einer speziellen Handgeste soll der Roboter zum Stillstand gebracht werden können. Eine mögliche Handgeste dafür wäre eine Faust.