Lego Mindstorms Tutorial

Lego Firmware

Als allerestes muss eine “Firmware“ auf den Mindstorms Computer (fortan kurz RCX) geladen werden. Dieser Programmcode hat die Funktion, Befehle per Infrarot (IR) entgegenzunehmen und sie auszuführen. Ohne laufende Firmware ist es gar nicht möglich, selber erstellte Programme laufen zu lassen. Die Firmware steuert auch das LCD Display des RCX. Die Firmware bleibt auch ohne Batterien erhalten und muss also nicht jedesmal neu geladen werden. Falls es doch mal nötig ist, kannst du das in einer DOS Box mit dem Kommando:

```
nqc –firmware c:\mindstorms\ws0102\firm\firm0309.lgo
```

Dieser Vorgang kann mehrere Minuten dauern.

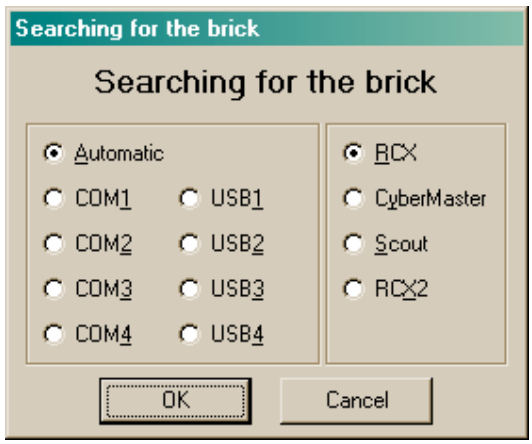
Die Programmiersprache Not Quite C

Es existieren sehr viele verschieden Möglichkeiten eigene Programme für den RCX zu schreiben. LEGO selber stellt eine grafische Arbeitsumgebung zur Verfügung mit der einfache Programme durch Zusammensetzen von Funktionsblöcken erstellt werden können. Diese Methode ist aber mit vielen Einschränkungen verbunden, zB. können keine eigenen Variablen definiert werden. Viel mehr Möglichkeiten bietet zB. eine Java-Variante für den RCX (http://lejos.sourceforge.net), oder eine reduzierte Variante von C (eben „Not Quite C“, kurz nqc, http://www.enteract.com/~dbaum/nqc/). Nqc wirst du nun etwas genauer kennen lernen.

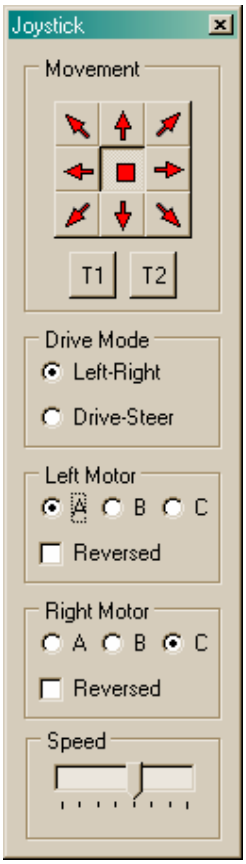
BricxCC Entwicklungsumgebung

Das Programm BricxCC stellt eine grafische Umgebung für die Entwicklung von NQC Programmen zur Verfügung. Zusätzlich enthält es eine Reihe von kleinen Hilfsprogrammen, mit denen man den RCX direkt steuern oder überwachen kann.
Klicke auf den Shortcut BricxCC.exe um das Programm zu starten.

ACHTUNG: Vor dem Programmstart muss der RCX eingeschaltet sein, der IR Sender muss mit dem Rechner verbunden. Der Sender muss direkt vor dem RCX plaziert werden, sodass die IR-Kommunikation möglich ist. Beim Starten des Programms erscheint der unten abgbildete Dialog: Wähle die Optionen `Automatic` *und* `RCX`*.*



Direkte Steuerung des RCX



In diesem Modus wird auf dem RCX kein eigentliches Programm ausgefuehrt sondern nur einzelne Kommandos die per IR uebermittelt werden. Die Buchstaben `A,B` und `C` stehen für die 3 Ausgänge des RCX, über die zB Motoren, Lämpchen, usw. angesteuert werden können.

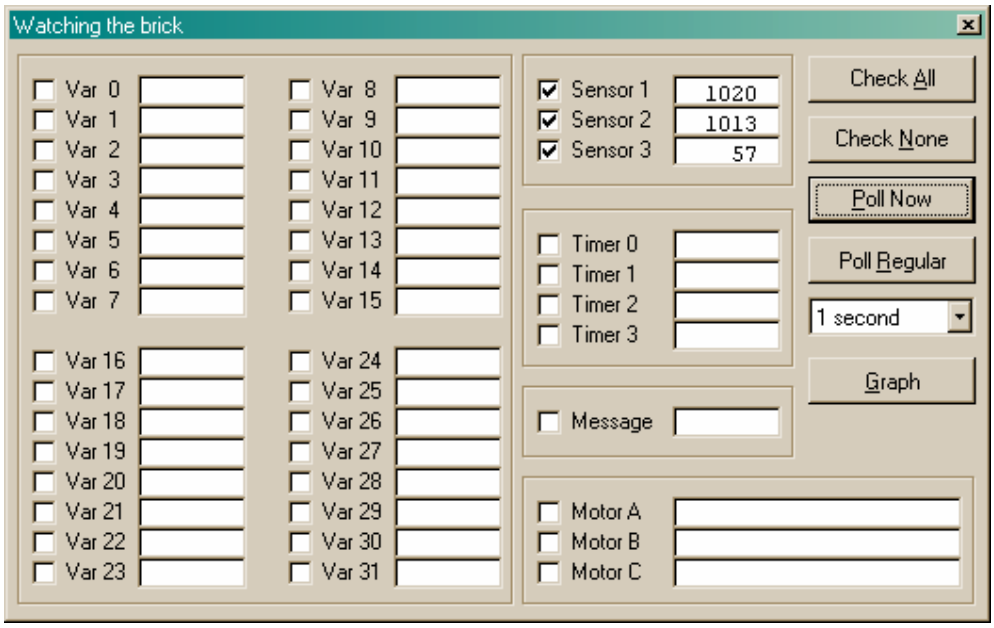
- Starte das Utility `Brick Joystick` aus dem Menue `Tools`.
- Falls nötig, ändere die Einstellungen (wie im Bild rechts):
  `Drive Mode: Left-Right`
  `Left Motor: A`
  `Right Motor C`.
- Mit den Pfeiltasten kann der RCX nun direkt gesteuert werden. (Tastatur: Zehnerblock)

? Wie funktioniert der "Drive-Steer" Modus? Wie müsste ein dem entsprechender Roboter gebaut sein ?

Überwachen der Sensoreingänge

Der RCX besitzt auch 3 Eingänge: Die mit `1,2` und `3` beschrifteten grauen Felder.

- Starte das Utility `Tools/Watching the Brick`.
- `Sensor 1, Sensor 2, Sensor 3` anwählen
- `Poll Now` drücken



Der angezeigte Wert verändert sich, wenn die Taster des RCX gedrückt sind während dem `Poll Now` ausgelöst wird.

- Probiere auch den Lichtsensor aus. Er scheint nicht recht zu funktionieren. Lichtsensoren sind aktive Sensoren, d.h. sie müssen vom RCX gespiesen werden. Mit `tools/direct control` kann der entsprechende Sensormodus gesetzt werden.

Das erste NQC-Programm

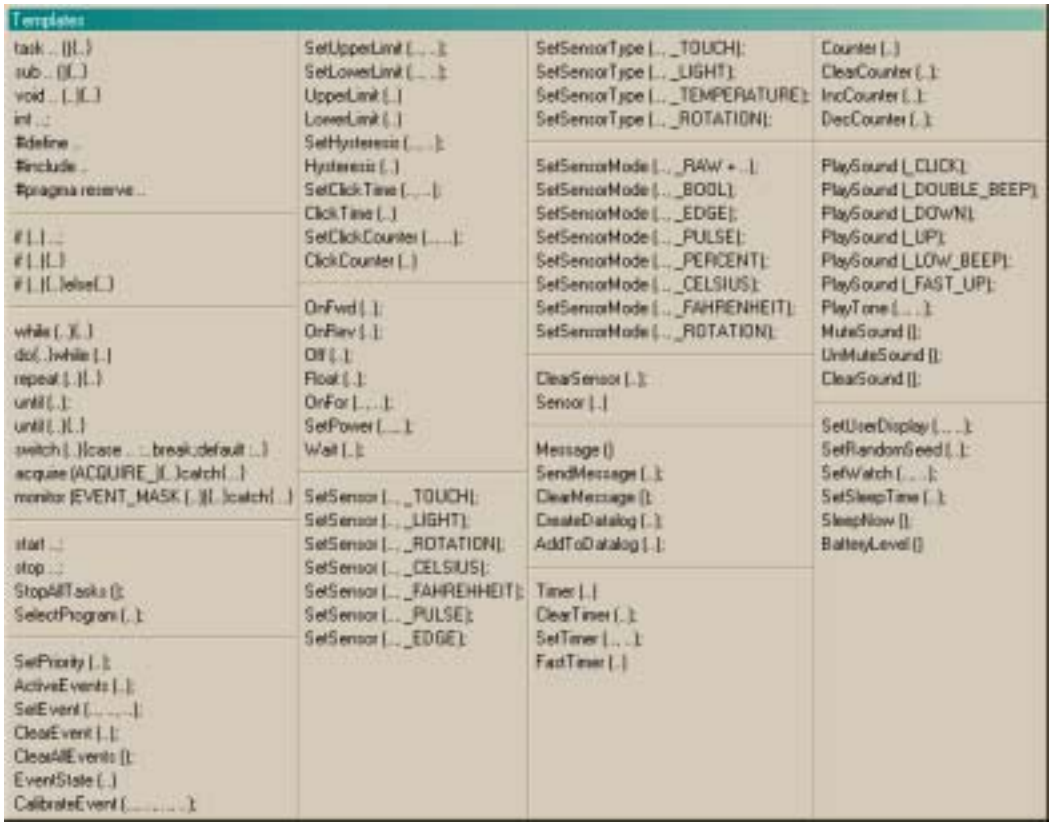
Der RCX kann auch autonom Programme ausfuehren, ohne permanent per IR-Link mit dem Host verbunden zu sein. Dazu muss das Programm zuerst auf den RCX geladen und dann mit der Taste <Run> auf dem RCX gestartet werden. Der RCX fuehrt dann den `Task main` aus.

- Oeffne ein neues Editierfenster mit `file/new`
- Tippe folgendes Programm ein:

```
#define FORWARD_TIME 500 // definiert eine Konstante

task main() {
    OnFwd(OUT_A+OUT_C); // schalte Ausgaenge A und C ein
    Wait(FORWARD_TIME); // warte 500 * 10ms
    Off(OUT_A+OUT_C);   // schalte A und C aus
}
```

Durch einfaches klicken auf eine Funktion im “Templates“ Fenster (siehe unten) wird diese in dein Editierfenster kopiert. Die Kommentare hinter „//“ brauchst du nicht abzutippen.



- Speichere das Programm mit `file/save` unter einem beliebigen Namen.
- Compiliere das Programm mit `F5`.

- Lade das Programm auf den RCX mit `F6`
- Starte das Programm mit `F7` oder der <Run> Taste auf dem RCX.
- Dieses Programm faehrt fuer 5 Sekunden geradeaus und bleibt dann stehen.

.. und nun etwas schlauer

Der RCX soll natuerlich nicht einfach stur geradeaus fahren, sondern auf Hindernisse reagieren. Im naechsten Schritt sollen also waehrend der Fahrt die Sensorern ueberwacht werden und wenn ein Hinderniss auftaucht, soll der RCX stehen bleiben. Studiere das folgende nqc Codefragment:

```
task main() {
   SetSensor (SENSOR_1, SENSOR_TOUCH);
   SetSensor (SENSOR_3, SENSOR_TOUCH);

    OnFwd(OUT_A+OUT_C);
    while((SENSOR_1 == 0) && (SENSOR_3 == 0)){
       // do nothing = wait
    }
    PlayTone (440,50);
    Off(OUT_A+OUT_C);
}
```

Zuerst werden die Sensoreingaenge in den richtigen Modus gesetzt. Danach werden die Motoren gestartet und die Sensoreingaenge ueberwacht. Dazu wird das folgende Konstrukt gebraucht:

```
while(Bedingung){Anweisung}
```

Der RCX wird überprüfen, ob `Bedingung` erfüllt ist. Falls ja, führt er `Anweisung` aus und überprüft danach ereneut die `Bedingung`. Er tut dies solange, bis die `Bedingung` falsch wird. Erst dann fährt der RCX mit dem Programm fort, diesmal ohne `Anweisung` auszuführen. In unserem Fall wird also überprüft, ob beide Sensoren *nicht* gedrückt sind. Falls das erfüllt ist, tut der RCX gar nichts, sondern prüft sofort wieder die Sensoren. Erst wenn mindestens ein Sensor gedrückt wurde - also ein Hinderniss detektiert wurde - pipst der RCX kurz und die Motoren werden abgeschaltet.

Vervollstaendige nun das folgende Fragment so, dass

- der RCX vorwaerts faehrt bis er auf ein Hinderniss stoesst.
- Wenn der linke Sensor betaetigt wurde, soll der RCX nun etwas zureckfahren und sich nach rechts drehen
- Wenn der rechte Sensor betaetigt wurde, soll der RCX ebenfalls zurueckfahren und sich nach links drehen
- Danach beginnt der RCX wieder vorwaerts zu fahren bis ein neues Hinderniss auftaucht.

```
#define BACK_TIME 50
#define TURN_TIME 80

task main() {
     SetSensor (SENSOR_1, SENSOR_TOUCH);
     SetSensor (SENSOR_3, SENSOR_TOUCH);
     OnFwd (OUT_A+OUT_C);
     while (true)
     {
       if (SENSOR_1 == 1) //falls Hinderniss links
```

```
        {
          PlayTone (440,50);
          // HIER VERVOLLSTAENDIGEN
          // wechsle die Richtung beider Motoren
          // warte BACK_TIME
          // drehe nach rechts
          // warte TURN_TIME
          // und wieder vorwaerts..

        }
        if (SENSOR_3 == 1) // falls Hindernis links
        {
           PlayTone (880,50);

            // HIER VERVOLLSTAENDIGEN (wie oben)

        }
      } // while

}   // main()
```

Kommunikation

Als nächsten Schritt sollen die Robos nun nach Artgenossen ausschau halten. Immer wenn der RCX auf einen anderen RCX trifft, soll er eine kurze Melodie (na ja, sowas ähnliches zumindest) spielen, sich um 180 Grad drehen und wieder weiterfahren.
Dies kann mit Hilfe der Infrarot Schnittstelle realisiert werden. Jeder RCX schickt kontinuierlich Nachrichten und empfängt diese Nachrichten von anderen RCXs. Da dies WAEHREND dem normalen Programmablauf geschehen soll, wird ein neuer Task definiert:

```
#define MESSAGE 3
#define MESSAGE_INTERVAL 100

task meet() {
     while (true) {
           SendMessage (MESSAGE);
           Wait (MESSAGE_INTERVAL);
           PlaySound (SOUND_CLICK);
           if (MESSAGE == Message())
           {
             ClearMessage ();
             PlaySound (SOUND_UP);
           }
     }
}
```

Damit dieser Task nun auch wirklich parallel zum Task main() ablaeuft, muss er von main() gestartet werden:

```
task main() {
     SetSensor (SENSOR_1, SENSOR_TOUCH);
     SetSensor (SENSOR_3, SENSOR_TOUCH);
     start meet;

     OnFwd (OUT_A+OUT_C);
     …..
```

Programmiere nun eine subroutine, welche aufgerufen wird, wenn der RCX einen Artgenossen trifft, und in der der RCX zuerst etwas zurueck faehrt, sich dann um 180 Grad dreht, und schliesslich die normale Ausfuehrung von main() wieder aufnimmt.
Der Task meet() muss dazu folgendermassen modifiziert werden:

```
task meet() {
     while (true) {
           SendMessage (MESSAGE);
           Wait (MESSAGE_INTERVAL);
           PlaySound (SOUND_CLICK);
           if (MESSAGE == Message())
           {
             ClearMessage ();
             greet();
           }
     }
```

Programmiere die subroutine greet()

```
sub greet() {
...
}
```

Links zu Mindstorms

.. sind bei Google/Altavista/Yahoo und Consorten leicht zu finden. Als Ausgangspunkt kann auch unsere eigene Mindstorms Page mit Links zu aktuellen und abgeschlossenen Projekten dienen: http://www.tik.ee.ethz.ch/mindstorms/