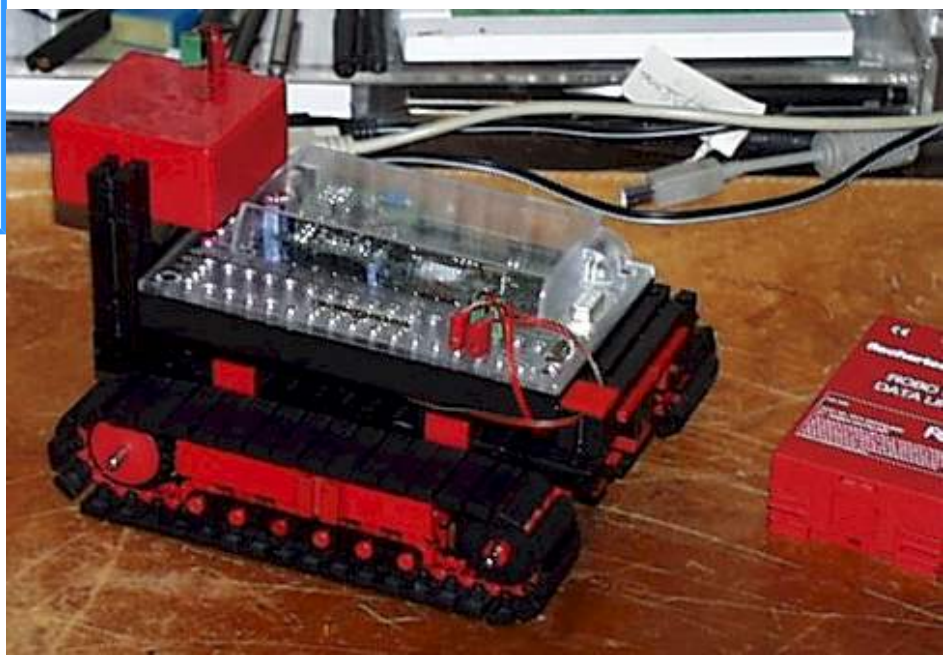
Raupen

Raupe pur : Steuerung über das ROBO RF DataLink



Chassis der Power Bulldozer 16 552 (Version mit 2 Motoren), ROBO Interface drauf, Akku darüber, fertig.

M1 : linker Motor, M2 rechter Motor. Sonst nichts, aber nicht vergessen : ROBO RF Datalink an USB anschließen.

Ziel der Übung : Fernsteuerung des stattlichen Fahrzeuges über den PC. Hier eine C# .NET Lösung

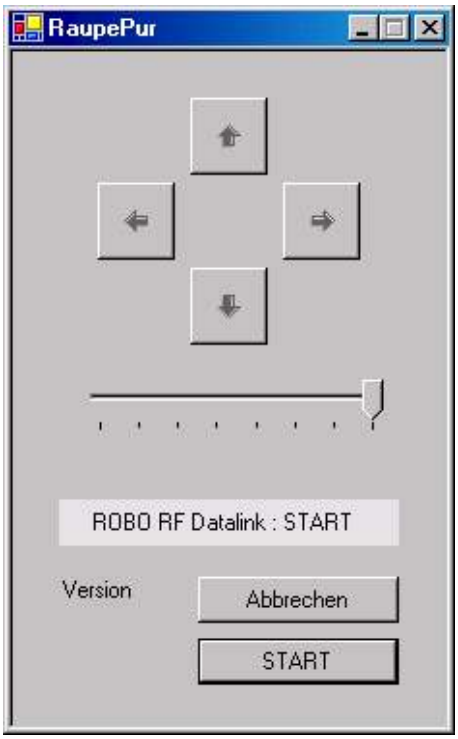
Anmerkungen zum C# Betriebsprogramm

Eingesetzt wurde FishFace40.DLL mit der aktuellen umFish40.DLL v4.0.3, die getrennt zu laden ist. ROBO Interface Firmware ab v1.53.0.03 und 0.34.0.02 für das RF DataLink und die Platine im Interface.

Schwerpunkt des Programms ist die Bedienoberfläche, der Betrieb der Raupe selber beschränkt sich auf Kombinationen von zwei SetMotor-Methoden :

```
ft.SetMotor(cMotLinks, cVor, trbSpeed.Value);
ft.SetMotor(cMotRechts, cRuck, trbSpeed.Value);
```

Die Raupe kann wahlweise mit der Maus über die Richtungsbuttons und über den Pfeiltastenblock der Tastatur bedient werden. Für die Tastaturbedienung muß die



Eigenschaft KeyPreview der Form auf true gesetzt werden.

Die Steuerung geschieht über die Routinen TurnLeft, TurnRight, GoForward, GoBack und StopAll. Die Fahrgeschwindigkeit kann über den Slider eingestellt werden, was gerade läuft wird über das Status-Feld angezeigt.

Bei der Bedienung über die Buttons werden deren Ereignisse
MouseDown und MouseUp (nur Aufruf der Steuerungsroutinen). Die Buttons sind auf einem Panel angeordnet um sie gemeinsam zu aktivieren nachdem ein internes ft.OpenInterface passiert ist.

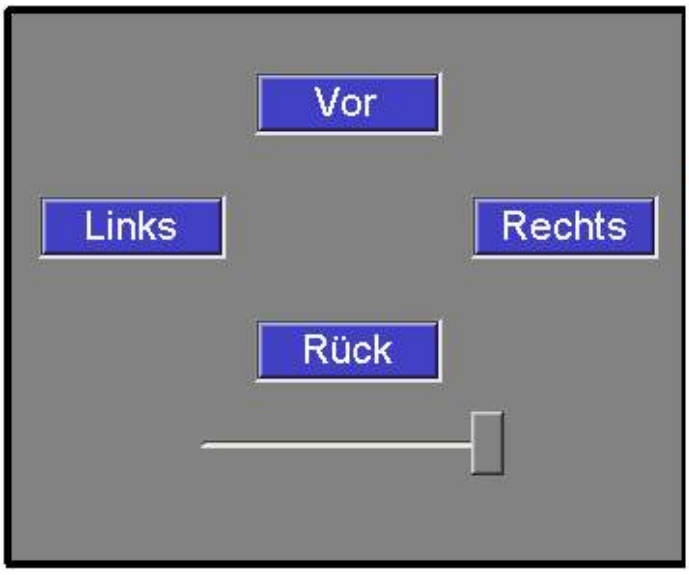
Bei der Bedienung über die Tastatur werden deren Ereignisroutinen
KeyDown und KeyUp genutzt. Sie enthalten einen Spurngverteiler zum Aufruf der tastenspezifischen Steuerungsroutine :

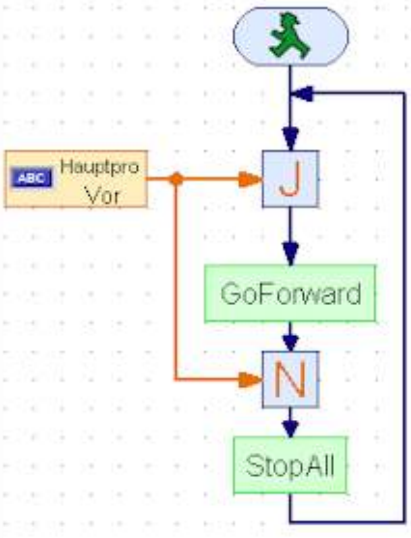
```
private void frmMain_KeyDown(object sender, System.Windows.Forms.KeyEventArgs e){ if(cmdAction.Enabled) return; switch (e.KeyValue) { case 37: // --- Left TurnLeft(); break; case 38: // --- Up GoForward(); break; case 39: // --- Right TurnRight(); break; case 40: // --- Down GoBack(); break; default: break; } e.Handled = true; }
```

Der Parameter e liefert die aktuell gedrückte Taste.

Die Beschriftung der Tasten cmdAction (hier "START") und cmdEnde (hier "Abbrechen") wird zur Steuerung des Programmendes genutzt. cmdAction wird dazu nach dem OpenInterfae auf Enabled = false gesetzt um eine wiederholte Bedienung zu unterdrücken. cmdEnde schließt bei Beschriftung "HALT" die Interface Verbindung und bei "ENDE" das Programm.

Anmerkungen zum ROBO Pro Programm



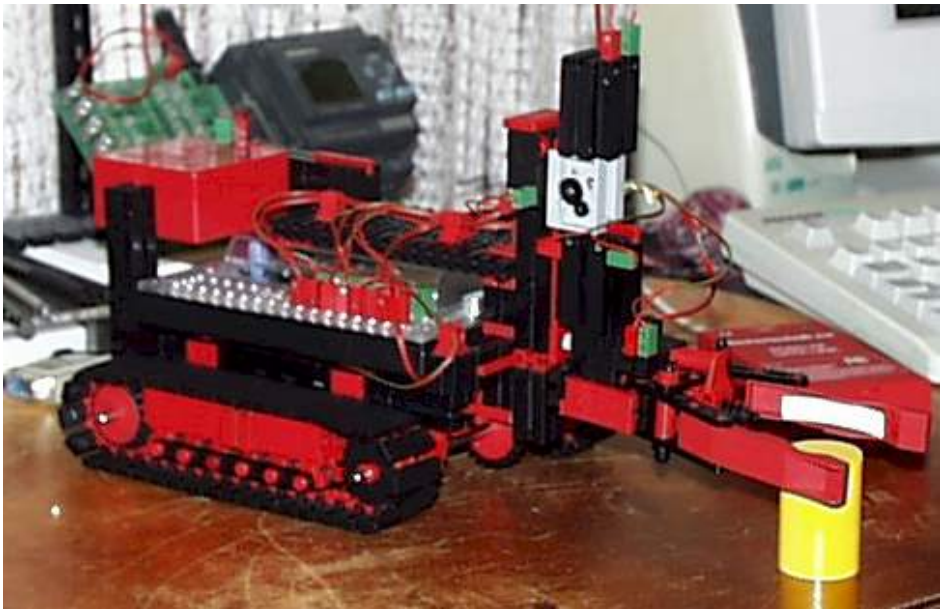


Eingesetzt wurde Level 3 mit 4 Threads in denen jeweils ein Steuerungstaste abgefragt - für eine Richtung - abgefragt wird. Zu Eingang des Threads auf Ja, dann wird das zugehörende UP aufgerufen und danach

wird gewartet, bis der Bediener den Zeigefinger wieder losläßt. Entspricht in der C#-Lösung den Ereignisroutinen MouseDown und MouseUP.

Ich bin meiner Marotte treu geblieben und habe das Bedienpanel auf der Seite des Hauptprogramms plaziert, ist mir immer noch bequemer.

Raupe mit Greifer



Fahrgestell wie oben. Das Interface wurde mit einer kleinen schwarzen Bauplatte überdacht, vorn wurde ein Hubgetrieb angebaut (siehe FTS Mobile Robots II) und am Getriebe hängt ein Greifer, der nach dem des Säulenrobots II gebaut wurde.
M1 : linker Motor, M2 : rechter Motor, M3 : Hubmotor (Ende durch I5 und I6), M4 Greifermotor mit I7 Ende- und I8 Impulstaster.

Steuerung wahlweise über Maus oder Tastatur (Pfeile : Fahren, Bild : Hub, Pos1/Ende : Greifer). Bequem vom Schreibtisch aus kann man jetzt gelbe Tonnen einsammeln und neben dem linken Fuß abstellen, wenn man das Ding über ROBO RF DataLink - ROBO Interface betreibt.

Anmerkungen zum C# Betriebsprogramm

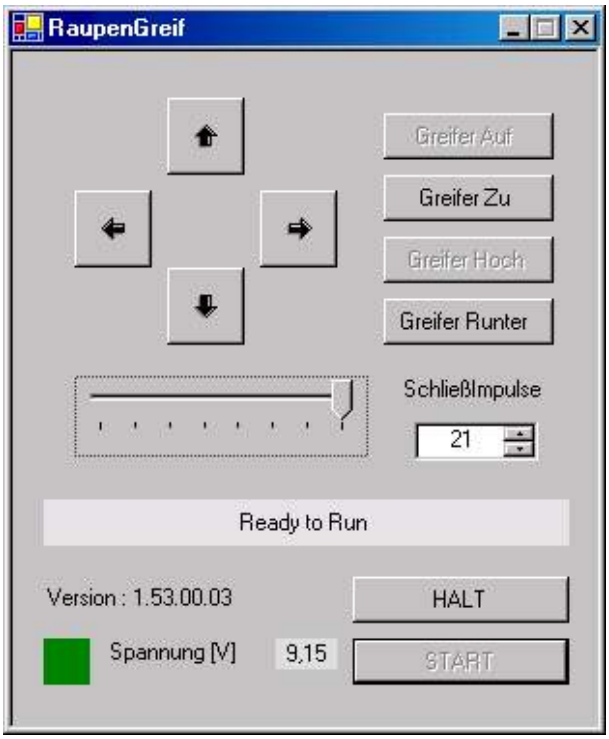
Das Programm ist eine Weiterentwicklung des Programms für Raupe pur. Es entspricht daher weitgehend dem obigen. 1 1/2 Seiten Source-Text sind hinzugekommen.

Hinzugekommen sind Routinen wie diese :

```
private void GreiferAuf() { GreiferStatus = cBusy; lblStatus.Text = "Greifer öffnet"; cmdGrAuf.Enabled = false; ft.SetMotor(mGreifer, cGrAuf); ft.WaitForInput(eGrOffen); ft.SetMotor(mGreifer, cAus); cmdGrZu.Enabled = true; GreiferStatus = eGrOffen; lblStatus.Text = "Greifer offen"; }
```

Es wird einiger Aufwand mit der Gesamtsteuerung des Programms getrieben. Der Greifer selber wird durch SetMotor - WaitForInput - SetMotor geöffnet.

Durch GreiferStatus wird eine erneute Betätigung des Button bzw. der Taste gesperrt.
Das Enabled steuert die jeweils möglichen Commands.



Und dann wird in lblStatus noch fleißig der Status angezeigt.

Anmerkungen zur ROBO Pro Lösung

Die Angelegenheit läuft auf eine Verdoppelung der Threads (grüne Männchen) hinaus (RaupenGreif). Achtung bei Eigenschaften die Anzahl der Prozesse anpassen, eingestellt sind nur fünf.

In den neuen Threads gibts die Befehle für Greifer Auf/Zu und Greifer Hoch/Runter, die wieder über Buttons des Bedienfeldes gesteuert werden. Das liegt jetzt tatsächlich auf dem Blatt Bedienfeld, da es auf dem Funktionen-Blatt doch recht voll geworden ist.

Man kann natürlich auch sein Zeichentalent voll entfalten und eine Version mit nur einem Thread erstellen, das Programm hat dann deutlich mehr Linien, aber alles wird in einem Thread (Prozess) zusammengehalten.

Wie mans macht ist hier wohl Geschmacksache, bei der Performance macht es sich nicht bemerkbar.

Download Raupen.ZIP

Stand : 19.11.2005