Programmierung der education Line Komponenten

eduLine.DLL
CrossRoads.DLL

VC++ 6.0 / Delphi / Python
VB 2005 / 2008 und C# 2005 / 2008
Java / BlueJ

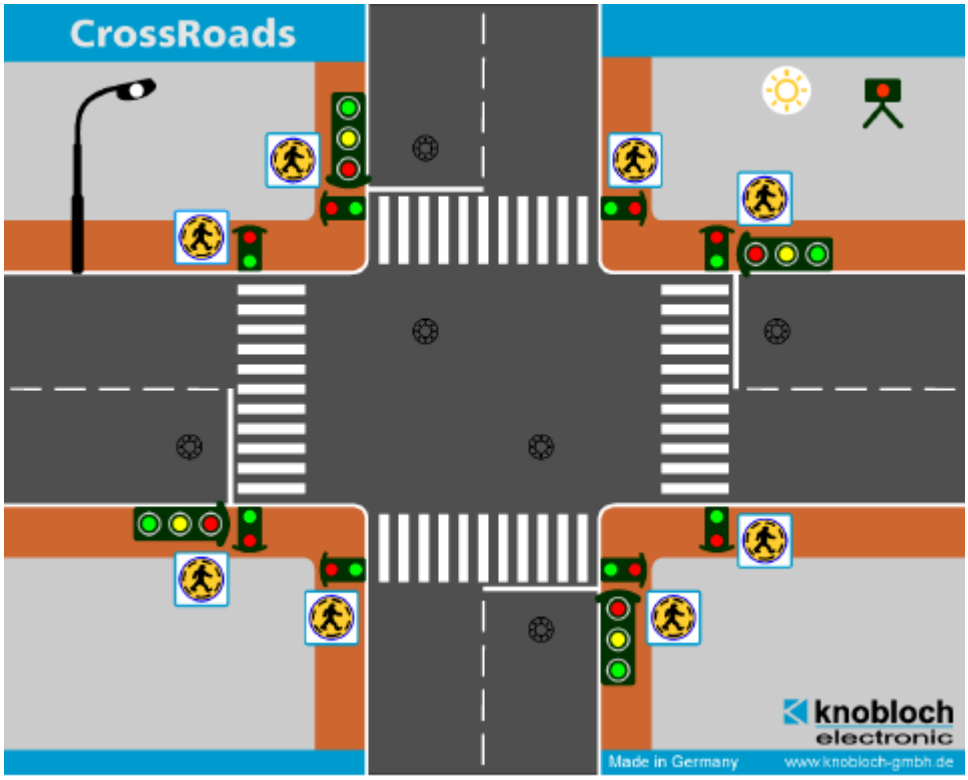
Ulrich Müller

Inhaltsverzeichnis

Übersicht

education Line : CrossRoads

Während die kleineren Devices der education Line sicher vorzugsweise mit ROBO Pro programmiert werden, ist für die CrossRoads aufgrund der Menge der Einzelkomponenten eine Programmierung mit einer textuellen Sprache sinnvoller. Möglich ist das mit Sprachen, die mit der eduLine.DLL bzw. einer CrossRoads.DLL umgehen können. Alternativ kann aber auch mit umFish40.DLL / FishFacexx.DLL programmiert werden. Es stehen dort die gleichen Funktionen zur Verfügung. Allerdings ist das Angebot etwas unübersichtlicher, da auch die ROBO-Interface Funktionen angeboten werden. Das eduLine-Device muß dann die Typ-Kennung 60 (ROBO-Interface) haben. Man kann die dazugehörenden Sources auch als Vorlage für eigenen Lösungen verwenden.



CrossRoads stellt eine Ampelkreuzung dar. CrossRoads bietet 30 (LED) Ausgänge und 14 Sensor (Druck, Magnet) Eingänge. Es wird über USB an einen PC (ab Windows 2000) angeschlossen. Die Stromversorgung erfolgt über die USB-Schnittstelle. Siehe auch die Betriebsanleitung. Dort wird auch die Installation des erforderlichen Treibers beschrieben.

Hersteller : Knobloch GmbH (www.knobloch-gmbh.de).

Anschlußbelegung

Intern werden ein "Robo Interface" (IF1) und drei "Extensions" (EM1, EM2, EM3) verwendet. Sie entsprechen dem normalen Robo Interface mit 3 Extensions.

Ausgänge

IF1O1(1)	Ampel : Hauptstrasse links, rot
IF1O2(2)	- gelb
IF1O3(3)	- grün
……..	
EM1O7(15)	Blitzer
EM1O8(16)	Straßenlampe
…….	
EM3O8(32)	Ampel : Nebenstrasse oben rechts, grün

Eingänge

IF1I1(1)	Fußgängertaster Hauptstraße links unten
….	
IF1I8(8)	Fußgängertaster Nebenstraße oben rechts
…	
EM1I6(14)	Magnetsensor (Kanaldeckel) Mitte oben

Vollständige Aufstellung : Siehe Bedienungsanleitung Seite 14.
Zahlen in Klammern : num. Werte

eduLine.DLL

Ist eine in VC++ 6.0 geschriebene DLL, die die Funktionen der eduLine-Devices an die Anwendung weiterreicht. Zusätzlich gibt es für einige Programmiersprachen die erforderlichen Deklarationen. Für Java existiert eine besonderde eduJava.DLL, siehe dort (eduJava.DLL). Bei Python wird die normale umFish40.DLL v4.77 verwendet.

Parameterbezeichnungen

iHandle	Handle zur Identifizierung des aktuellen Interfaces im Programm (1 - 8)
LampNr	Nummer eines Ausganges im Bereich 1 - 32
InputNr	Nummer eines Einganges im Bereich 1 - 14
VoltNr	Nummer des Helligkeitseinganges (1)
OnOff	Schaltzustand Ein / Aus (1 / 0)
ifTyp	Typnummer des eduLine-Devices (siehe enum weiter unten)
SerialNr	Standardseriennummer des CrossRoad-Devices
Fehlercode	Verlauf der Funktion 0 = OK sonst rbFehler alle Funktionen können alternativ zu ihrem korrekten Rückgabewert den Fehlercode rbFehler liefern.
int	Allgemeiner 32bit Integerwert mit Vorzeichen
uint	Allgemeiner 32bit Integerwert ohne Vorzeichen

Befehlsliste

iHandle	**edOpenUSB**(ifTyp, SerialNr) Herstellen einer Verbindung zu einem CrossRoads-Device an USB. Parameter 0, 0 : Das erste Device an USB wird genommen. Parameter 60, n : Das erste CrossRoad Device mit der angegebenen Seriennummer wird genommen.
Fehlercode	**edCloseUSB**(iHandle) Beenden der Verbindung zu CrossRoads
OnOff	**edGetInput**(iHandle, InputNr) Auslesen des Zustandes des angegebenen Einganges
int	**edGetInputs**(iHandle) Status des Zustandes aller Eingänge
int	**edGetVoltage**(iHandle, VoltNr) Auslesen des Spannungswertes der Helligkeitsanzeige in mV
uint	**edGetOutputs**(iHandle) Einschaltstatus aller Augänge
Fehlercode	**edSetOutputs**(iHandle, MotorStatus) Setzen des Einschaltstatus aller Ausgänge
Fehlercode	**edSetOutput**(iHandle, LampNr, OnOff) Setzen eines einzelnen (LED) Ausganges
int	**edGetActDeviceType**(iHandle) Auslesen des aktiven Devicetyps. erfolgreiches rbOpenInterface erforderlich, sonst Fehlercode.
int	**edGetActDeviceSerialNr**(iHandle) Auslesen der SerialNr des aktiven Devices.
int	**edGetActDeviceFirmwareNr**(iHandle) Auslesen der FirmwareNr des aktiven Devices. Sollte byteweise als String formatiert werden.
uint	**edFehler** = 0xE0000001 allgemeiner Fehlercode
-	**Sleep**(mSek) Systemfunktion zum Parken des aktuellen Threads für die angegeben Anzahl von Millisekunden
uint	**GetTickCount**() Systemfunktion : Anzahl Millisekunden seit Mitternacht
uint	**GetAsyncKeyState**(vkey) Systemfunktion : Abfrage, ob eine bestimmte Taste gedrückt wurde uint ESCAPE = 1B (Hexawert)

CrossRoads.DLL

Beschrieben wird hier die .NET Version, sie ist die umfangreichste. Bei den anderen Sprachen gibt es ähnliche, meist mit kleinerem Befehlsumfang.

Die Ausgänge werden auf ein 32bit unsigned integer abgebildet. Die Eingänge auf ein 32bit signed integer. Die Eingänge können einzeln oder auch als Ganzes abgefragt werden, ebenso können die Ausgänge einzeln oder als Ganzes gesetzt werden.

Die Methoden sind unterbrechbar (DoEvents) und abbrechbar (ESC-Taste, NotHalt = true).

Alle Methoden und Eigenschaften können Exceptions vom Typ CrossRoadsException auslösen. Der Messagetext enthält dann den Namen der auslösenden Methode und den Grund.

Enums

IFTypen
first_USB = 0, ROBO_Interface_USB = 60, EDU_LINE_Module = 300, EDU_LINE_CrossRoads = 301, EDU_LINE_TrafficLights = 302, EDU_LINE_Signal = 303

Dir
Schaltstatus der Ausgänge
On/Ein, Off/Aus

Inp
Benennung der Eingänge nach Bild auf Seite14 der Betriebsanleitung
A1 = 1, IFiI1 = 1 …. EM1I8 = 16

Out
Benennung der Ausgänge nach Bild auf Seite 14 der Betriebsanleitung
IF1O1 = 1 … EM3O8 = 32

OutM
Masken für die einzelnen Ausgänge im (UINT) OutputStatus
IF1O1 = 0x1 … EM3O8 = 0x80000000

Exceptions

CrossRoadsException(Message)
Allgemeiner Fehler am CrossRoads-Device. Mit Aufrufvarianten

Konstruktor

CrossRoads()
Parameterloser Konstruktor

Eigenschaften

DeviceData **ActDevice**
Struktur mit den Eigenschaften des CrossRoads-Devices

bool **NotHalt**
Anforderung eines Abbruchs aller länger laufenden Methoden bei NotHalt = true

bool **OpenStatus**
Gibt den Openstatus zurück

uint **Outputs**
Auslesen des Status aller Ausgänge

string **Version**
Ausgabe der Version von CrossLights.DLL

Methoden

ClearOutputs()
Löschen aller Ausgänge

CloseUSB()
Schließen der Verbindung zu CrossRoads

bool **Finish**()
Endewunsch, ausgelöst durch ESC-Taste oder Eigenschaft NotHalt = true

bool **GetInput**(InputNr)
Statusabfrage eines einzelnen Einganges

int **GetInputs**()
Auslesen des Status aller Eingänge

int **GetVoltage**(Inp.A1)
Auslesen des Spannungswertes am Lichtsensor in mV

OpenUSB([ifTyp, SerialNr])
Herstellen einer Verbindung zu einem CrossRoads-Device.
Ohne Parameter : erstes CrossRoads an USB
Parameter CrossRoads_USB, n : CrossRoads-Device mit der SerialNr n

Pause(mSek)
Anhalten des Programmablaufs um mSek MilliSekunden (unterbrechbar, abbrechbar)

SetOutput(OutputNr, OnOff)
Setzen eines Ausganges

SetOutputs(uint Status)
Setzen aller Ausgänge

WaitForInput(InputNr [, OnOff])
Warten auf den angegebenen Zustand eines Einganges

Nutzung

Allgemeines

eduLine.DLL liegt in compilierter Form vor (z.Zt. auf Basis von FtLib v1.77). Zusätzlich gibt es für VC++ 6.0, Delphi, VB2005 und C# 2005 die erforderlichen Deklarationen und dazu ein kleines HelloLights Beispielprogramm.

eduLine.DLL ist in erster Linie als Basis für die Entwicklung von sprachspezifischen (eigenen) Bibliotheken gedacht. Eine direkte Nutzung kann problematisch sein, da z.B. die Unterbrechbarkeit(DoEvents) und die Abbrechbarkeit (ESC-Taste) nicht gegeben ist.

CrossRoads ist eine kleine Klassenbibliothek zum direkten Einsatz in Anwendungsprogrammen.

Sowohl eduLine.DLL und CrossRoads können problemlos auch im Zusammenhang mit VB2008 bzw. C# 2008 eingesetzt werden.

VC++ 6.0

eduLine.DLL

```
#include "eduLineVC.H"

void main() {
   char Ende;
   cout << "--- Hallo CrossLine : es geht los ---" << endl;
   int iHandle = edOpenUSB(0, 0);
   if (iHandle == edFehler) {
      cout << "Da stimmt etwas nicht : ENDE (Enter-Taste)" << endl;
      cin.get(Ende);
      return;
   }
   cout << "Interface  : " << edGetActDeviceName(iHandle)
   << ", Typ          : " << edGetActDeviceType(iHandle)
   << ", SerialNr     : " << edGetActDeviceSerialNr(iHandle) << endl;
   cout << "Firmware   : " << edGetActDeviceFirmware(iHandle) << endl;
   cout << endl << "Start : I1 druecken" << endl;
   while(!edGetInput(iHandle, 1)) {Sleep(123);}
   for (int i = 0; i < 5; i++) {
      edSetOutput(iHandle, 1, 1);
      Sleep(333);
      edSetOutput(iHandle, 1, 0);
      Sleep(333);
   }
   rbCloseUSB(iHandle);
   cout << endl << "--- FINIS : Enter-Taste ---" << endl;
   cin.get(Ende);
}
```

Console Projekt : Einfacher Blinker an Ausgang O1. Start durch Drücken des Fußgängerknopfes I1 links unten. Vorher werden noch Daten des angeschlossenen Devices ausgegeben.

Delphi 4

eduLine.DLL

```
uses … eduLine;
……
var
  eduMain: TeduMain;
  edu: LongInt;

implementation
{$R *.DFM}
procedure TeduMain.cmdActionClick(Sender: TObject);
var i: LongInt;
begin
  edu := edOpenUSB(0, 0);
  if edu = edFehler then begin
    lstAusgabe.Items.Add('Hier stimmt was nicht');
  end
  else begin
    lstAusgabe.Items.Add('Hallo eduLine');
    lstAusgabe.Items.Add('Device : ' +
                              IntToStr(edGetActDeviceType(edu)));
    lstAusgabe.Items.Add('Start : I1 drücken');
    Application.ProcessMessages;
    while edGetInput(edu, 1) = 0 do Sleep(123);
    for i := 1 to 5 do begin
      lstAusgabe.Items.Add('Runde : ' + IntToStr(i));
      Application.ProcessMessages;
      edSetOutput(edu, 1, 1);
      Sleep(444);
      edSetOutput(edu, 1, 0);
      Sleep(333);
    end;
    edCloseUSB(edu);
    lstAusgabe.Items.Add('--- FINITO ---');
  end;
end;
```

Windows Projekt : Einfacher Blinker an Ausgang O1. Start durch Drücken des Fußgängerknopfes I1 links unten. Vorher werden noch Daten des angeschlossenen Devices ausgegeben.

Python

eduCrossRoads.py

Zu Python siehe auch www.ftcomputing.de/pythonecke.htm. Insbesondere dort das Handbuch. eduCrossRoads basiert im Gegensatz zu den anderen Bibliotheken auf der allgemeinen umFish40.DLL

```
from eduCrossRoads import *

print "------ Blinker wird gestartet ------"
cr = CrossRoads()
try:
  cr.OpenUSB()
  print "Zum Starten des Blinkers I1 Druecken"
  print "PollInterval : ", cr.PollInterval

  cr.WaitForInput(1)
  Runde = 1

  while not cr.Finish():
    print "Runde : ", Runde
    print "GetInputs : ", cr.GetInputs()
    cr.SetOutput(1, cr.Ein)
    cr.Pause(555)
    cr.SetOutput(1, cr.Aus)
    cr.Pause(333)
    Runde += 1
except CrossRoadsException:
  print "CrossRoads Fehler"
print "------- Blinker wird beendet -------"
cr.CloseUSB()
```

Console Projekt : Einfacher Blinker an Ausgang O1. Start durch Drücken des Fußgängerknopfes I1 links unten.

C# 2005 / 2008

Hinweis : Die Sources sind C# 2005, sie werden bei Start eines Projektes mit C# 2008 IDE automatisch konvertiert. Hinweis 2 : Bei Ablauf auf Win7/64bit Systemen ist eine Einstellung auf Platform x86 erforderlich, das kann bei der jeweiligen Express Version recht umständlich sein.

eduLine.DLL

Beispiel : Einfacher Blinker

```
using System;
using cs = System.Console;
using eduLine;
using edu = eduLine.eduLine;

namespace eduTest {
  class Program {
    uint iHandle;
    [STAThread]
    static void Main(string[] args) {
      Program cr = new Program();
      cs.WriteLine("--- Hello CrossLine gestartet ---");
      cr.Action();
      cs.WriteLine("--- Hello ROBO beendet (Return-Taste) ---");
      cs.Read();
    }
    private void Action() {
      iHandle = edu.rbOpenUSB((int)IFTypen.first_USB, 0);
      if (iHandle == edu.edFehler) {
        cs.WriteLine("Da stimmt was nicht : ENDE (Return-Taste)");
        return;
      }
      cs.WriteLine("Device Typ  : " +
                   edu.edGetActDeviceType(iHandle));
      cs.WriteLine("Serial Nr   : " +
                   edu.edGetActDeviceSerialNr(iHandle));
      cs.WriteLine("Firmware Nr : " +
                   edu.edGetActDeviceFirmwareNr(iHandle));
      cs.WriteLine("IN ACTION");
      for (int i = 0; i < 10; i++) {
        edu.edSetOutput(iHandle, 1, 1);
        edu.Sleep(444);
        edu.edSetOutput(iHandle, 1, 0);
        edu.Sleep(333);
      }
      edu.edCloseUSB(iHandle);
    }
  }
}
```

Console Projekt. Die Elemente eduLine sind statisch, deswegen keine Instanziierung.

CrossRoads.DLL

Hier ein etwas umfangreicheres Programm mit mehreren Proceduren, die Einzellösungen aufzeigen. Die CrossLights.DLL selber wurde ebenfalls in C# 2005 erstellt.

Programmrahmen

```
using CrossRoads40;

namespace CrossLightsBeispieleCS {
  public partial class BeispieleMain : Form {
    CrossRoads cr = new CrossRoads();
    public BeispieleMain() {InitializeComponent();}

    private void cmdAction_Click(object sender, EventArgs e) {
      try {
        lstAusgabe.Items.Clear();
        cr.OpenUSB();
        lstAusgabe.Items.Add(cr.ActDevice);
        lstAusgabe.Items.Add("Schleifenende : ESC-Taste");
        switch (cboBeispiele.Text) {
          case "MultiBlinker":
            MultiBlinker(); break;
          case "Ampel":
            Ampel();        break;
          case "AmpelListe":
            AmpelListe();   break;
          case "AmpelDuo":
            AmpelDuo();     break;
          case "FußÜber":
            FussUeber();    break;
          case "RotBlitzer":
            RotBlitzer();   break;
          default:
            break;
        }
        lstAusgabe.Items.Add("--- FINITO ---");
        cr.CloseUSB();
      }
      catch (CrossRoadsException ecr) {
        lstAusgabe.Items.Add(ecr.Message);
      }
    }
```

using : Zugriff auf den Namensraum von CrossRoads.DLL erlauben (Verweis einrichten)

CrossRoads cr : CrossRoads instanzieren

void cmdAction_Click des Buttons Action klammert alle Zugriffe auf CrossRoads durch ein try - catch Konstrukt (gilt auch für die aufgerufenen Methoden). Evtl. Fehler werden in der Liste lstAusgabe angezeigt.

cr.OpenUSB : Verbindung zum ersten eduLine-Device an USB herstellen

cr.ActDevice : Anzeige der Device Daten

switch case : Aufruf der eigentlichen Beispiele

cr.CloseUSB : Schließen der Verbindung zum eduLine-Device.

MultiBlinker

```
private void MultiBlinker() {
  uint Lampe = (uint)OutM.EM1O8;
  uint Blitzer = 0;
  lstAusgabe.Items.Add("Spielereien mit den Möglichkeiten");
  do {
    cr.SetOutputs(Blitzer + Lampe + (uint)OutM.IF1O1 +
                                    (uint)OutM.EM3O6);
    cr.Pause(444);
    cr.SetOutputs(Lampe + (uint)OutM.IF1O4 + (uint)OutM.EM3O8);
    cr.Pause(333);
    if(cr.GetInput(Inp.IF1I1))
          lstAusgabe.Items.Add("Läuft - Helligkeitswert : "
          + cr.GetVoltage(Inp.A1)).ToString();
     Blitzer = cr.GetInput(Inp.EM1I6) ? (uint)OutM.EM1O7 : 0;
  } while (!cr.Finish());
}
```

do … while !cr.Finish() : Endlosschleife, Ende durch ESC-Taste
cr.SetOutputs (Blitzer .. : Anzeige akt. Blitzer, akt. Lampe, Ampel rot Hauptstrasse links, Fußgängerampel grün Nebenstraße unten links für 444 MilliSekunden

cr.SetOutputs(Lampe .. Anzeige Lampe, Ampel rot Hauptstraße rechts, Fußgängerampel grün, Nebenstraße oben rechts für 333 MilliSekunden

cr.GetInput : Wenn gelbes Männchen unten rechts gedrückt ist, wird der aktuelle Helligkeitswert angezeigt

cr.GetInput(Inp.EM1I6) : Wenn Auto auf dem Kanaldeckel Mitte oben steht, wird geblitzt

Ampel

```
private void Ampel() {
  lstAusgabe.Items.Add("Einfache Ampel");
  do {
    cr.SetOutput(Out.IF1O3, Dir.On);
    cr.Pause(3000);
    cr.SetOutput(Out.IF1O3, Dir.Off);
    cr.SetOutput(Out.IF1O2, Dir.On);
    cr.Pause(1000);
    cr.SetOutput(Out.IF1O2, Dir.Off);
    cr.SetOutput(Out.IF1O1,Dir.On);
    cr.Pause(3500);
    cr.SetOutput(Out.IF1O2,Dir.On);
    cr.Pause(1000);
    cr.SetOutput(Out.IF1O1,Dir.Off);
    cr.SetOutput(Out.IF1O2,Dir.Off);
  } while (!cr.Finish());
}
```

Die Ampel an der Hauptstraße links unten wird schön brav geschaltet

AmpelDuo

```
private enum Duos : uint {
  Rot = OutM.IF1O1 + OutM.IF1O4,
  Gelb = OutM.IF1O2 + OutM.IF1O5,
  Gruen = OutM.IF1O3 + OutM.IF1O6,
  RotQ = OutM.EM1O1 + OutM.EM1O4,
  GelbQ = OutM.EM1O2 + OutM.EM1O5,
  GruenQ = OutM.EM1O3 + OutM.EM1O6
}
private void AmpelDuo() {
  lstAusgabe.Items.Add("Ampelkreuzung im Duett");
  do {
    cr.SetOutputs((uint)Duos.Rot + (uint)Duos.GruenQ);
    cr.Pause(3500);
    cr.SetOutputs((uint)Duos.Gelb + (uint)Duos.GelbQ);
    cr.Pause(1000);
    cr.SetOutputs((uint)Duos.Gruen + (uint)Duos.RotQ);
    cr.Pause(3500);
    cr.SetOutputs((uint)Duos.Gelb + (uint)Duos.GelbQ);
    cr.Pause(3500);
  } while (!cr.Finish());
}
```

Hier werden die Ampeln an der Haupt- und Nebenstraße geschaltet. Dazu wird eine enum Duos angelegt, die die Masken für die betroffenen Ampeln der Straßen enthält. Die Ampeln werden dann in einem Loop gleichzeitig geschaltet.

AmpelListe

```
private void AmpelListe() {
  uint[] SchaltListe = {
                (uint)Duos.Rot   + (uint)Duos.GruenQ,
                (uint)Duos.Rot   + (uint)Duos.GruenQ,
                (uint)Duos.Gelb  + (uint)Duos.GelbQ,
                (uint)Duos.Gruen + (uint)Duos.RotQ,
                (uint)Duos.Gruen + (uint)Duos.RotQ,
                (uint)Duos.Gruen + (uint)Duos.RotQ,
                (uint)Duos.Gelb  + (uint)Duos.GelbQ};
  uint Lampe = 0;
  lstAusgabe.Items.Add("Ampelkreuzung nach Liste");
  do {
    Lampe = cr.GetVoltage(Inp.A1) < 500
          ? (uint)OutM.EM1O8 : Lampe = 0;
    for (int i = 0; i < SchaltListe.Length; i++) {
      cr.SetOutputs(Lampe + SchaltListe[i]);
      cr.Pause(1000);
    }
  } while (!cr.Finish());
}
```

Die Ampeln an der Hauptstraße und die an der Nebenstraße werden nach Liste geschaltet. Dazu werden die entsprechenden Masken in eine Liste gestellt. Geschaltet wird einmal pro Sekunde, wenn eine Anzeige länger dauern soll, so ist sie entsprechend zu wiederholen.

Zusätzlich wird die Helligkeit der Umgebung gemessen. Bei Dunkelheit (< 500) soll die Lampe angeschaltet werden (Lampe). Die Schaltung selber wird beim Abarbeiten der Liste erledigt.

Fußgängerampel

```
bool FussWunsch = false;
private void tmrFuss_Tick(object sender, EventArgs e) {
  FussWunsch = cr.GetInput(Inp.IF1I1) || cr.GetInput(Inp.IF1I2)
             ? true : false;
}
private enum DuoF : uint {
  Rot = OutM.EM2O1 + OutM.EM3O1,
  Gruen = OutM.EM2O2 + OutM.EM3O2
}
private void FussUeber() {
  tmrFuss.Enabled = true;
  lstAusgabe.Items.Add("Fußgängerüberweg");
  do {
    if (FussWunsch) {
      FussWunsch = false;
      cr.SetOutputs((uint)Duos.Gelb + (uint)DuoF.Rot);
      cr.Pause(1000);
      cr.SetOutputs((uint)Duos.Rot + (uint)DuoF.Gruen);
      cr.Pause(3500);
      cr.SetOutputs((uint)Duos.Gelb + (uint)DuoF.Rot);
      cr.Pause(1000);
    }
    cr.SetOutputs((uint)Duos.Gruen + (uint)DuoF.Rot);
    cr.Pause(5000);
  } while (!cr.Finish());
  tmrFuss.Enabled = false;
}
```

Die Straßenampel steht normal auf Grün. Im Timer tmrFuss werden alle 100 MilliSekunden die Fußgängertasten abgefragt. Wenn eine gedrückt ist, wird FussWunsch auf true gesetzt.

In der Methode FussUeber läuft eine Endlosschleife, die FussWunsch abfragt. Im true-Fall wird ein Fußzyklus abgearbeitet.

RotBlitzer

```
private bool Anforderung = false;
private void tmrBlitzer_Tick(object sender, EventArgs e) {
  if (Anforderung) {
    if (cr.GetInput(Inp.EM1I5) && !cr.GetInput(Inp.EM1I3)) {
      cr.SetOutput(Out.EM1O7, Dir.On);
      cr.Pause(333);
      cr.SetOutput(Out.EM1O7, Dir.Off);
      Anforderung = false;
    }
  }
  else Anforderung = cr.GetInput(Inp.EM1I3);
}
private void RotBlitzer() {
  lstAusgabe.Items.Add("RotBlitzer ist aktiv");
  do {
    cr.SetOutputs((uint)Duos.Rot + (uint)Duos.GruenQ);
    cr.Pause(3500);
    cr.SetOutputs((uint)Duos.Gelb + (uint)Duos.GelbQ);
    cr.Pause(1000);
    cr.SetOutputs((uint)Duos.Gruen + (uint)Duos.RotQ);
    tmrBlitzer.Enabled = true;
    cr.Pause(3500);
    tmrBlitzer.Enabled = false;
    cr.SetOutputs((uint)Duos.Gelb + (uint)Duos.GelbQ);
    cr.Pause(1000);
  } while (!cr.Finish());
}
```

Eigentlich eine ganz normale Kreuzung (wie AmpelDuo) nur wird hier während der Rotphase der tmrBlitzer zugeschaltet.

Dort wird zunächst geprüft, ob der Anforderungskanaldeckel der Querstraße geschaltet ist. Im True-Fall wird der Kanaldeckel Mitte in Verbindung mit dem verlassenen Anforderungsdeckel geprüft und dann kann es blitzen.

VB 2005 / 2008

Hinweis : Die Sources sind VB 2005, sie werden bei Start eines Projektes mit VB 2008 IDE automatisch konvertiert. Hinweis 2 : Bei Ablauf auf Win7/64bit Systemen ist eine Einstellung auf Platform x86 erforderlich, das kann bei der jeweiligen Express Version recht umständlich sein.

eduLine.DLL

```
Private Sub cmdAction_Click(ByVal sender As System.Object, _
              ByVal e As System.EventArgs) Handles cmdAction.Click
  Dim iHandle As UInteger
  lstAusgabe.Items.Clear()
  lstAusgabe.Items.Add("eduLine gestartet")
  iHandle = eduLine.edOpenUSB(IFTypen.first_USB, 0)
  If iHandle = eduLine.edFehler Then
    lstAusgabe.Items.Add("Da stimmt was nicht")
    Return
  End If
  lstAusgabe.Items.Add("Device Typ : " & _
                              eduLights.rbGetActDeviceType(iHandle))
  lstAusgabe.Items.Add("SerialNr   : " & _
                           eduLights.rbGetActDeviceSerialNr(iHandle))
  lstAusgabe.Items.Add("FirmwareNr : " & _
                         eduLine.edGetActDeviceFirmwareNr(iHandle))
  lstAusgabe.Items.Add("IN ACTION")
  Application.DoEvents()
  For i As Integer = 1 To 10
    eduLine.edSetMotor(iHandle, 1, 1)
    eduLine.Sleep(444)
    eduLine.edSetMotor(iHandle, 1, 0)
    eduLine.Sleep(333)
  Next
  eduLine.rbCloseUSB(iHandle)
  lstAusgabe.Items.Add("--- FINITO ---")
End Sub
```

Einfacher Blinker. Die Elemente von eduLine sind statisch, deswegen keine Instantiierung.

CrossRoads.DLL

Hier ein etwas umfangreicheres Programm mit mehreren Subs, die Einzellösungen aufzeigen. Die CrossRoads.DLL selber wurde in C# 2005 erstellt.

Programmrahmen

```
Imports CrossRoads40
Public Class frmMain
   Dim cr As New CrossRoads()

Private Sub cmdAction_Click(ByVal sender As System.Object, _
                ByVal e As System.EventArgs) Handles cmdAction.Click
     Try
        lstAusgabe.Items.Clear()
        cr.OpenInterface(IFTypen.first_USB, 0)
        lstAusgabe.Items.Add(cr.ActDevice)
        lstAusgabe.Items.Add("Schleifenende : ESC-Taste")

        Select Case cboBeispiel.Text
          Case "MultiBlinker"
               MultiBlinker()
          Case "Ampel"
               Ampel()
          Case "Ampelliste()"
……
        End Select

        cr.CloseInterface()
     Catch ecr As CrossRoadsException
        lstAusgabe.Items.Add(ecr.Message)
     End Try
End Sub
```

Imports : Zugriff auf den Namensraum von CrossRoads.DLL erlauben (Verweis einrichten)

Dim cr : CrossRoads instanzieren

Sub cmdAction_Click des Buttons Action klammert alle Zugriffe auf CrossRoads durch ein Try - Catch Konstrukt (gilt auch für die aufgerufenen Subs). Evtl. Fehler werden in der Liste lstAusgabe angezeigt.

cr.OpenUSB : Verbindung zum ersten eduLine-Device an USB herstellen

cr.ActDevice : Anzeige der Device Daten

Select Case : Aufruf der eigentlichen Beispiele

cr.CloseUSB : Schließen der Verbindung zum Device.

MultiBlinker

```
Private Sub MultiBlinker()
     Dim Lampe As UInteger = OutM.EM1O8
     Dim Blitzer As UInteger = 0
     Do
       cr.SetOutputs(Blitzer + Lampe + OutM.IF1O1 + OutM.EM3O6)
       cr.Pause(444)
       cr.SetOutputs(Lampe + OutM.IF1O4 + OutM.EM3O8)
       cr.Pause(333)
       If cr.GetInput(Inp.IF1I1) Then
          lstAusgabe.Items.Add("Läuft - Helligkeitswert : " + _
                               cr.GetVoltage(Inp.A1).ToString())
       End If
       If cr.GetInput(Inp.EM1I6) Then Blitzer = OutM.EM1O7 _
       Else Blitzer = 0
     Loop Until cr.Finish()
End Sub
```

Do … Loop Until cr.Finish() : Endlosschleife, Ende durch ESC-Taste
cr.SetOutputs (Blitzer .. : Anzeige akt. Blitzer, akt. Lampe, Ampel rot Hauptstrasse links, Fußgängerampel grün Nebenstraße unten links für 444 MilliSekunden

cr.SetOutputs(Lampe .. Anzeige Lampe, Ampel rot Hauptstraße rechts, Fußgängerampel grün, Nebenstraße oben rechts für 333 MilliSekunden

cr.GetInput : Wenn gelbes Männchen unten rechts gedrückt ist, wird der aktuelle Helligkeitswert angezeigt

cr.GetInput(Inp.EM1I6) : Wenn Auto auf dem Kanaldeckel Mitte oben steht, wird geblitzt

Ampel

```
Private Sub Ampel()
     Do
       cr.SetLamp(Out.IF1O3, Dir.On)
       cr.Pause(3000)
       cr.SetLamp(Out.IF1O3, Dir.Off)
       cr.SetLamp(Out.IF1O2, Dir.On)
       cr.Pause(1000)
       cr.SetLamp(Out.IF1O2, Dir.Off)
       cr.SetLamp(Out.IF1O1, Dir.On)
       cr.Pause(3500)
       cr.SetLamp(Out.IF1O2, Dir.On)
       cr.Pause(1000)
       cr.SetLamp(Out.IF1O1, Dir.Off)
       cr.SetLamp(Out.IF1O2, Dir.Off)
     Loop Until (cr.Finish())
  End Sub
```

Die Ampel an der Hauptstraße links unten wird schön brav geschaltet

AmpelDuo

```
Private Enum Duos
     Rot = OutM.IF1O1 + OutM.IF1O4
     Gelb = OutM.IF1O2 + OutM.IF1O5
     Gruen = OutM.IF1O3 + OutM.IF1O6
     RotQ = OutM.EM1O1 + OutM.EM1O4
     GelbQ = OutM.EM1O2 + OutM.EM1O5
     GruenQ = OutM.EM1O3 + OutM.EM1O6
End Enum

Private Sub AmpelDuo()
     Do
       cr.SetOutputs(Duos.Rot + Duos.GruenQ)
       cr.Pause(3500)
       cr.SetOutputs(Duos.Gelb + Duos.GelbQ)
       cr.Pause(1000)
       cr.SetOutputs(Duos.Gruen + Duos.RotQ)
       cr.Pause(3500)
       cr.SetOutputs(Duos.Gelb + Duos.GelbQ)
       cr.Pause(1000)
     Loop Until cr.Finish()
End Sub
```

Hier werden die Ampeln an der Haupt- und Nebenstraße geschaltet. Dazu wird eine Enum Duos angelegt, die die Masken für die betroffenen Ampeln der Straßen enthält. Die Ampeln werden dann in einem Loop gleichzeitig geschaltet.

AmpelListe

```
Private Sub AmpelListe()
     Dim SchaltListe() As UInteger = { _
     Duos.Rot + Duos.GruenQ, _
     Duos.Rot + Duos.GruenQ, _
     Duos.Rot + Duos.GruenQ, _
     Duos.Gelb + Duos.GelbQ, _
     Duos.Gruen + Duos.RotQ, _
     Duos.Gruen + Duos.RotQ, _
     Duos.Gruen + Duos.RotQ, _
     Duos.Gelb + Duos.GelbQ}
     Dim Lampe As UInteger = 0
     Do
       If cr.GetVoltage(Inp.A1) < 500 Then Lampe = OutM.EM1O8 _
       Else Lampe = 0
       For i As Integer = 0 To SchaltListe.Length - 1
         cr.SetOutputs(Lampe + SchaltListe(i))
         cr.Pause(1000)
       Next
     Loop Until cr.Finish()
  End Sub
```

Die Ampeln an der Hauptstraße und die an der Nebenstraße werden nach Liste geschaltet. Dazu werden die entsprechenden Masken in eine Liste gestellt. Geschaltet wird einmal pro Sekunde, wenn eine Anzeige länger dauern soll, so ist sie entsprechend zu wiederholen.

Zusätzlich wird die Helligkeit der Umgebung gemessen. Bei Dunkelheit (< 500) soll die Lampe angeschaltet werden (Lampe). Die Schaltung selber wird beim Abarbeiten der Liste erledigt.

Fußgängerampel

```
Private FussWunsch As Boolean = False

Private Sub tmrFuss_Tick(ByVal sender As System.Object, _
               ByVal e As System.EventArgs) Handles tmrFuss.Tick
     If cr.GetInput(Inp.IF1I1) Or cr.GetInput(Inp.IF1I2) _
     Then FussWunsch = True _
     Else FussWunsch = False
End Sub

Private Enum DuoF
     Rot = OutM.EM2O1 + OutM.EM3O1
     Gruen = OutM.EM2O2 + OutM.EM3O2
End Enum

Private Sub FussUeber()
     tmrFuss.Enabled = True
     lstAusgabe.Items.Add("Fußgängerüberweg")
     Do
       If FussWunsch Then
         FussWunsch = False
         cr.SetOutputs(Duos.Gelb + DuoF.Rot)
         cr.Pause(1000)
         cr.SetOutputs(Duos.Rot + DuoF.Gruen)
         cr.Pause(3500)
         cr.SetOutputs(Duos.Gelb + DuoF.Rot)
         cr.Pause(1000)
       End If
       cr.SetOutputs(Duos.Gruen + DuoF.Rot)
       cr.Pause(5000)
     Loop Until cr.Finish()
     tmrFuss.Enabled = False
End Sub
```

Die Straßenampel steht normal auf Grün. Im Timer tmrFuss werden alle 100 MilliSekunden die Fußgängertasten abgefragt. Wenn eine gedrückt ist, wird FussWunsch auf True gesetzt.

In der Sub FussUeber läuft eine Endlosschleife, die FussWunsch abfragt. Im True-Fall wird ein Fußzyklus abgearbeitet.

RotBlitzer

```
Private RotPhase As Boolean
Private Anforderung As Boolean = False

Private Sub tmrBlitzer_Tick(ByVal sender As System.Object, _
                ByVal e As System.EventArgs) Handles tmrBlitzer.Tick
  If Anforderung Then
    If cr.GetInput(Inp.EM1I5) And Not cr.GetInput(Inp.EM1I3) Then
      cr.SetLamp(Out.EM1O7, Dir.On)
      cr.Pause(333)
      cr.SetLamp(Out.EM1O7, Dir.Off)
      Anforderung = False
    End If
  Else
    Anforderung = cr.GetInput(Inp.EM1I3)
  End If
End Sub

Private Sub RotBlitzer()
  lstAusgabe.Items.Add("RotBlitzer ist aktiv")
  Do
    cr.SetOutputs(Duos.Rot + Duos.GruenQ)
    cr.Pause(3500)
    cr.SetOutputs(Duos.Gelb + Duos.GelbQ)
    cr.Pause(1000)
    cr.SetOutputs(Duos.Gruen + Duos.RotQ)
    tmrBlitzer.Enabled = True
    cr.Pause(3500)
    tmrBlitzer.Enabled = False
    cr.SetOutputs(Duos.Gelb + Duos.GelbQ)
    cr.Pause(1000)
  Loop Until cr.Finish()
End Sub
```

Eigentlich eine ganz normale Kreuzung (wie AmpelDuo) nur wird hier während der Rotphase der tmrBlitzer zugeschaltet.

Dort wird zunächst geprüft, ob der Anforderungskanaldeckel der Querstraße geschaltet ist. Im True-Fall wird der Kanaldeckel Mitte in Verbindung mit dem verlassenen Anforderungsdeckel geprüft und dann kann es blitzen.

Java und BlueJ

Allgemeines zu

Erforderliche Software

JDK Java Development Kit

Ist bei den moderneren Rechner bereits installiert sonst
J2SE SDK http://java.sun.com downladen. Bei Einsatz auf 64bit-Systemen ist eine 32bit Version zu installieren.

BlueJ

Erforderliche ist eine Java Entwicklungsumgebung. Hier wird von BlueJ ausgegegangen. BlueJ ist für Programmieranfänger besonders geeignet. Wenn man mit Eclipse 3.4 vertraut ist sollte man auch dabei bleiben.
BlueJ kann von http://www.bluej.org kostenlos bezogen werden.

USB Treiber

Siehe Betriebsanleitung CrossRoads

DLLs

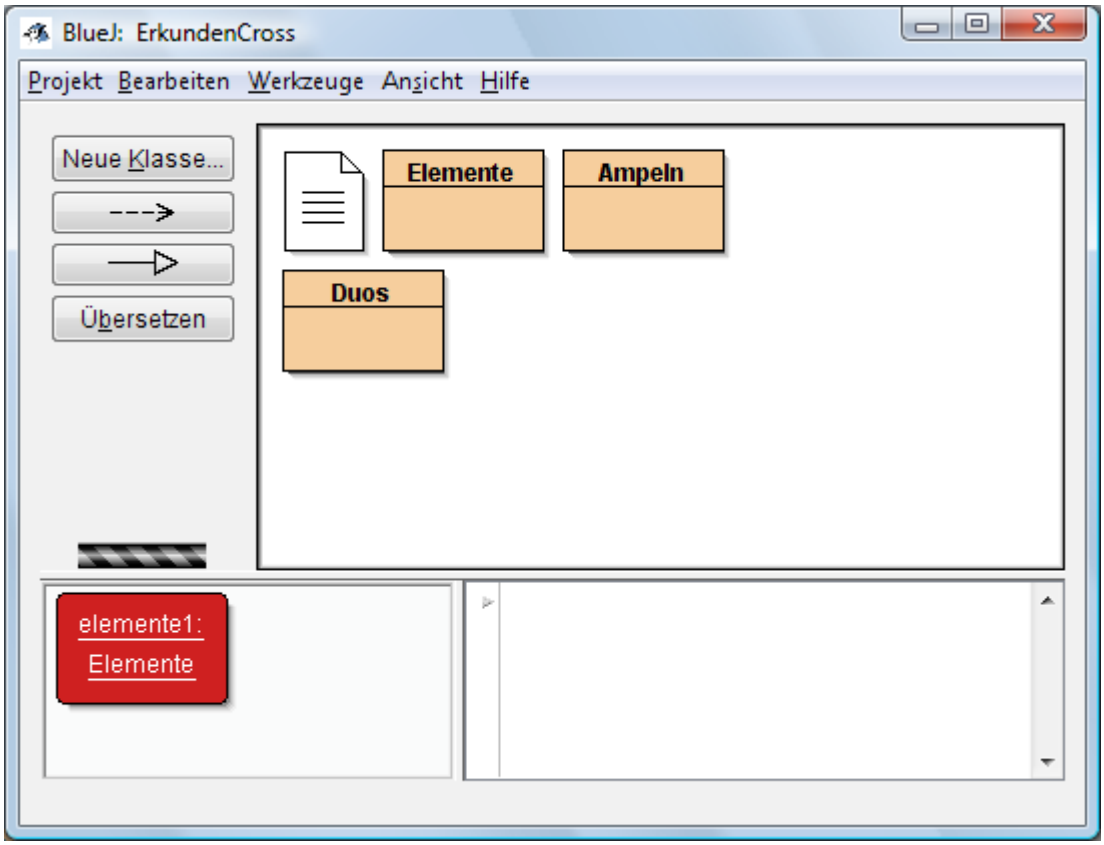
eduLine.DLL und eduJava.DLL. Muß an einem zentralen Ort abgelegt werden (am einfachsten \Windows\System32) in eduline.zip

package

package eduLine.jar aus eduline.zip

In der nutzenden Quelle ist ein import eduLine.*; erforderlich.

Hinweise zu BlueJ



BlueJ ist eine freie Entwicklungsumgebung mit dem Schwerpunkt : Erlernen der der Programmiersprache Java. Sie erlaubt das einfache Erstellen und Testen kleinerer Anwendungen. In der Test-Umgebung können auch einzelne Klasse instanziert (hier rotes Symbol elemente1, Instanz der Klasse Elemente) werden. Deren Methoden können dann einzeln ausgeführt werden.

Zusätzlich zum BlueJ Download sollte man das Tutorial zu BlueJ downloaden.

Programmieranfänger sollten vielleicht auch noch das Buch
"Java lernen mit BlueJ" (ISBN 978-3-86894- 001-5) anschaffen.

BlueJ wird standardmäßig in englisch installiert. Eine Änderung auf deutsch ist im File \<BlueJ>\lib\bluej.defs möglich.

eduLine.jar an zentralen Ort ablegen (passendes BlueJ Verzeichnis) und über
BlueJ Menü | Werkzeuge | Einstellungen | Bibliotheken hinzufügen.

eduJava.DLL / eduLine.DLL

Da Java die systemkonforme eduLine.DLL nicht direkt ansteuern kann, wurde die Wrapper.DLL eduJava.DLL zwischengeschaltet. Die Deklarationen zur Ansteuerung der education Line Devices sind in der Klasse eduJava untergebracht (package eduLine.*;). Einsatz eher zur Entwicklung eines eigenen packages.

Klasse eduJava

```
int jeOpenUSB(int ifTyp, int SerialNr);
```
Herstellen der Verbindung zum eduLine-Device an USB
```
int jeCloseUSB(int iHandle);
```
Beenden der Verbindung zum Device

```
int jeGetInput(int iHandle, int InputNr);
```
Auslesen eine Einganges

```
int jeGetInputs(int iHandle);
```
Auslesen aller Eingänge (bit 0 = Eingang 1)

```
int jeGetVoltage(int iHandle, int VoltNr);
```
Auslesen des Helligkeitssensors.

```
int jeGetOutputs(int iHandle);
```
Auslesen aller Ausgänge (bit 0 = Ausgang 1)

```
int jeSetOutputs(int iHandle, int Status);
```
Setzen aller Ausgänge (bit 0 = Ausgang 1)

```
int jeSetOutput(int iHandle, int OutNr, int OnOff);
```
Setzen eines Ausganges

```
int jeGetActDeviceType(int iHandle);
```
Auslesen der Typnummer des aktiven Devices

```
int jeGetActDeviceSerialNr(int iHandle);
```
Auslesen der Seriennummer des aktuellen Devices

```
int jeGetActDeviceFirmwareNr(int iHandle);
```
Auslesen der Firmwarnummer des aktuellen Devices

```
static int jeError = 0xE0000001;
```
Fehlerkennung

```
static int escape();
```
ESC-Taste getätigt = 1

```
static int getTickCount();
```
Anzahl TickCounts seit Mitternacht.

```
static void Sleep(int MilliSek);
```
Pause in Millisekunden.

Beispiel

```
import eduLine.*;

public class eduMain {
  public static void main(String[] args) {
     eduJava ej = new eduJava();
     int iHandle = ej.jeOpenUSB(0, 0);
     if(iHandle == eduJava.jeError) {
        System.out.println("Open.Problem : ENDE");
        return;
     }
     System.out.println(
                  "Action : IF1-I1 (Fußgänger links unten) drücken");
     while(ej.jeGetInput(iHandle, 1) == 0) Thread.yield();
     System.out.println(
            "Ende : ESC-Taste oder IF1-I5 (Fußgänger rechts unten");
     int Runde = 0;
     do {
        try{
          System.out.println("Runde   : " + ++Runde);
          ej.jeSetOutput(iHandle, 1, 1);
          Thread.sleep(333);
          ej.jeSetOutput(iHandle, 1, 0);
          Thread.sleep(333);
        } catch(InterruptedException e){}
     } while((ej.jeGetInput(iHandle, 5) == 0)
                                          && (eduJava.escape() == 0));
     ej.jeCloseUSB(iHandle);
     System.out.println("--- FINITO ---");
  }
}
```

Console Projekt : Einfacher Blinker an Ausgang IF1-O1. Start durch Drücken des Fußgängerknopfes IF1-I1 links unten. Vorher werden noch Daten des angeschlossenen Devices ausgegeben. Ende durch IF1-I5 oder ESC-Taste.

Klasse CrossRoads und zugehörige

Gedacht für eine komfortable Anwendungsprogrammierung

Allgemeines

```
public CrossRoadsException(String message)
```
Durch CrossRoads ausgelöste Exception

```
IFTypen first_USB = 0,                  ROBO_Interface_USB  = 60,
        EDU_LINE_Module = 300,          EDU_LINE_CrossRoads = 301,
        EDU_LINE_TrafficLights = 302,   EDU_LINE_Signal     = 303
```
Unterstütze Devices

```
public enum Dir {Off, On}
```
Schaltzustände der Ausgänge

```
public enum Inp {IF1I1, IF1I2, IF1I3, IF1I4, IF1I5, IF1I6, IF1I7,
                 IF1I8, EM1I1, EM1I2, EM1I3, EM1I4, EM1I5, EM1I6}
```
Liste der Eingänge (siehe auch CrossRoads Betriebsanleitung Seite 14)

```
public enum Out {IF1O1, IF1O2, IF1O3, IF1O4, IF1O5, IF1O6, IF1O7,
                 IF1O8, EM1O1, EM1O2, EM1O3, EM1O4, EM1O5, EM1O6,
                 EM1O7, EM1O8, EM2O1, EM2O2, EM2O3, EM2O4, EM2O5,
                 EM2O6, EM2O7, EM2O8, EM3O1, EM3O2, EM3O3, EM3O4,
                 EM3O5, EM3O6, EM3O7, EM3O8}
```
Liste der Ausgänge (siehe auch CrossRoads Betriebsanleitung Seite 14)

```
public class OutM {int IF1O1 = 0x1,IF1O2 = 0x2, …
```
Liste der Ausgänge : Position im 32bit Status-Wort

```
DeviceData getActDevice()
```
Auslesen der Daten des aktuellen Devices

Methoden

`void clearOutputs()`
Zurücksetzten aller Ausgänge

`void closeUSB()`
Schließen der USB-Verbindung zum Device

`boolean finish()`
Feststellen eines Abbruchwunsches (ESC-Taste, NotHalt)

`boolean getInput(Inp inputNr)`
Auslesen des Zustandes eines Einganges

`int getInputs()`
Auslesen des Zustandes aller Eingänge

`int getVoltage()`
Auslesen des aktuellen Helligkeitswertes

`boolean getNotHalt()`
Festellen, ob eine Abbruchwunsch angemeldet wurde

`void  setNotHalt(boolean OnOff)`
Anmelden eines Abbruchwunsches

`void openUSB()`
Herstellen einer Verbindung zum ersten Device an USB

`void openUSB(int ifTyp, int serialNr)`
Herstellen einer Verbindung zum beschriebenen Device

`int getOutputs()`
Auslesen aller Ausgänge

`void pause(int mSek)`
Anhalten des Programmablaufes für die angegebenen Millisekunden

`void setOutput(Out outputNr, Dir onOff)`
Setzen eines Ausganges

`void setOutputs(int status)`
Setzen aller Ausgänge (bit 0 = Ausgang 1)

`void waitForInput(Inp inputNr)`
Warten auf Ausgang = true

`void waitForInput(Inp inputNr, boolean onOff)`
Warten auf Ausgang = onOff

`void waitForLow(Inp inputNr)`
Warten auf einen true-false Durchgang am angegebenen Eingang

`void waitForHigh(Inp inputNr)`
Warten auf eine false-true Durchgang am angegebenen Eingang

`static String Version()`
Auslesen der package-Version

Beispiele

FirstCross

```
import eduLine.*;
public class FirstCross {
    public static void main() {
        CrossRoads cr = new CrossRoads();
        System.out.println("FirstCross gestartet, " +
                           "Ende : ESC-Taste");
        cr.openUSB(0,0);
        boolean flipFlop = false;
        while(!cr.finish()) {
            if(flipFlop) cr.setOutput(Out.EM1O2, Dir.Off);
            else         cr.setOutput(Out.EM1O2, Dir.On);
            cr.pause(333);
            flipFlop = !flipFlop;
        }
        cr.setOutput(Out.EM1O2, Dir.Off);
        cr.closeUSB();
    }
}
```

Einfacher Blinker : Gelbe Lampe EM1-O1 rechts unten. Ablauf als Console main
`import` : Einbeziehen des package eduLine.jar.
`CrossRoads cr` : Instanzieren der Klasse CrossRoads
`cr.openUSB` : Herstellen einer Verbindung zum ersten eduLine Device an USB
Die Lampe wird über ein flipFlop geschaltet das in einer Endlos-Schleife (while ..) läuft, die durch die ESC-Taste beendet werden kann (cr.finish())
`cr.setOutput` : Der ordnunghalber abschalten aller Ausgänge und kappen der USB-Verbindung
`Out.EM1O1` Name aus der enum Out, der hier als symbolischer Parameter für `setOutput` genutzt wird.

SecondCross

```
import eduLine.*;
public class SecondCross {
    CrossRoads cr;
    public static void main() {
        System.out.println("SecondCross gestartet, " +
                           "Ende : ESC-Taste");
        SecondCross fc = new SecondCross();
        fc.cr = new CrossRoads();
        try {
            fc.cr.openUSB(0,0);
            fc.Action();
            fc.cr.closeUSB();
        }
        catch(CrossRoadsException eft) {System.out.println(eft);}
        finally {System.out.println("FINITO");}
    }
```

Console main wie FirstCross. Hier wird aber eine Instanz von SecondCross erstellt. Sinnvoll, wenn man mit mehreren Methoden arbeiten will, die sich auch noch globale Variable teilen. Die Instanz von SecondCross wird jetzt in main erstellt. Die Methode Action wird aus einem try - catch Konstrukt aufgerufen, um Device-Fehler so abzufangen. Außerdem findet hier noch openUSB / closeUSB statt.

```
    private void Action() {
        boolean flipFlop = false;
        while(!cr.finish()) {
            if(flipFlop) cr.setOutputs(OutM.IF1O2 + OutM.IF1O5);
            else         cr.setOutputs(OutM.EM1O2 + OutM.EM1O5);
            cr.pause(333);
            flipFlop = !flipFlop;
        }
        cr.clearOutputs();
    }
}
```

Die Methode Action läßt die gelben Lichter aller Ampeln im Wechsel blinken. Dazu werden mit `setOutputs` alle Ausgänge geschaltet. Als Parameter werden die bitWerte der einzuschaltenen Lampen mitgegeben. Alle anderen werden ausgeschaltet.
IF1O2 = Ox10 = 1 0000binär und EM1O2 = Ox200 = 10 0000 0000, bit 5 und 10 werden geschaltet.

Elemente

Eine Klasse ohne main, aber mit mehreren Methoden. Für einen Testlauf wird hier die Möglichkeit von BlueJ genutzt, einzelne Methoden ausführen zu können.

```
import eduLine.*;
public class Elemente {
    private CrossRoads cr;
    public Elemente() {
        cr = new CrossRoads();
        cr.openUSB(0, 0);
    }
```

Im Konstruktor wird CrossRoads instanziert und die Verbindung zum Device hergestellt. Anschließend wird mit BlueJ ein Object von Elemente angelegt, man kann dann über eine Auswahlliste (RechtsClick) die gewünschte Methode aufrufen.

```
    public void outListe() {
        System.out.println("outListe gestartet");
        for(int i = 1; i <= 32; i++) {
            Out O = Out.values()[i];
            System.out.println("Out." + O);
            cr.setOutput(O, Dir.On);
            cr.waitForHigh(Inp.IF1I6);
            cr.setOutput(O, Dir.Off);
        }
        System.out.println("--- FINITO ---");
    }
```

Da die Bezeichnung der Ein- und Ausgänge nicht direkt auf das zugehörende Symbol des CrossRoads-Panels führt (sie sind an Interface und 3 Extension Module angelehnt), kann man sich hier - einen nach dem anderen - die Ausgänge anzeigen. Dazu muß man dann nur den FußTaster IF1I6 finden (links unten). Das Programm besitzt eine for-Schleife über alle Ausgänge. In der Schleife wird als erstes der Name des zugehörenden enum-Elementes bestimmt und angezeigt, anschließend wird der Ausgang eingeschaltet und darauf gewartet (waitForHigh : Taster loslassen und dann wieder drücken, bei waitForInput rauscht die Liste nur so durch), bis man durch Betätigen des FußTaster anzeigt, dass man den Ausgang gefunden hat. Dann wird er wieder abgeschaltet.

```
public void fussListe() {
    System.out.println("fussListe gestartet");
    do {
        int iWert = cr.getInputs();
        int maske = 1;
        for(int i = 1; i <= 8; i++) {
            if((iWert & maske) > 0)
                   System.out.println("Inp." + Inp.values()[i]);
            maske *= 2;
        }
        cr.pause(333);
    } while(!cr.finish());
    System.out.println("--- FINITO ---");
}
```

Das gleiche für die Eingänge, erstmal die FußTasten. Beim Drücken auf eine beliebige Taste wird der zugehörende enum-Name angezeigt. Dazu enthält die Methode zwei ineinander geschachtelte Schleifen (Außen do - innen for). In der do-Schleife wird der Status aller Eingänge nach iWert eingelesen. In der for-Schleife wird dieser Status dann bitweise abgefragt. bit 0 = IF1-I1 … Ist das entsprechende bit gesetzt, wird der Name angezeigt. Begonnen wird mit bit 0 (maske = 1) und es geht bis bit 7 (maske = 128)

```
public void deckelListe() {
    System.out.println("deckelListe gestartet");
    do {
        int iWert = cr.getInputs();
        int maske = 256;
        for(int i = 1; i <= 6; i++) {
            if((iWert & maske) > 0)
               System.out.println("Inp." + Inp.values()[i + 8]);
            maske *= 2;
        }
        cr.pause(333);
    } while(!cr.finish());
    System.out.println("--- FINITO ---");
}
```

Entspricht der Methode zum Anzeigen der FußTasten, nur muß man hier mit dem Auto über die Kanaldeckel fahren. Die Runde beginnt mit dem nächsten bit (8, maske 256). Da es nur 6 Deckel gibt, hört sie etwas früher auf.

```
public void radarBlitz() {
    System.out.println("radarBlitz gestartet");
    do {
        cr.waitForLow(Inp.EM1I3);
        cr.setOutput(Out.EM1O7, Dir.On);
        System.out.println("hat ihn");
        cr.pause(333);
        cr.setOutput(Out.EM1O7, Dir.Off);
    } while(!cr.finish());
    System.out.println("--- FINITO ---");
}
```

Und dann gibt es da noch den radarBlitz EM1-O7 (rechts oben), eigentlich war er schon bei der outListe dran, aber da es so schön ist hier ein Solo : Warten bis Deckel EM1-I3 vom Auto überfahren wird und dann mit EM1-O7 blitzen.

```
    public void helligkeit() {
        System.out.println("helligkeit gestartet");
        do {
            int hell = cr.getVoltage();
            if(hell < 555) cr.setOutput(Out.EM1O8, Dir.On);
            else if(hell > 631) cr.setOutput(Out.EM1O8, Dir.Off);
            System.out.println("Helligkeitswert : " + hell);
            cr.pause(1234);
        } while(!cr.finish());
        System.out.println("--- FINITO ---");
    }
}
```

Der Helligkeitssensor (IF1-A1, rechts oben) muß auch noch gewürdigt werden. Hier wird in einer Schleife die aktuelle Helligkeit angezeigt. Wenns zu dunkel ist wird die Straßenlaterne EM1-O8 (links oben) eingeschaltet und wenn es wieder hell genug ist, wieder ausgeschaltet. Was hell und dunkel ist sollte über die Werte der abgefragten Konstanten festgelegt werden. Anmerkung : man kann int hell und int dunkel auch als Parameter beim Aufruf der Methode festlegen (BlueJ unterstützt das)

Ampeln

```
public class Duos
{
    public final static int Rot      = OutM.EM1O1 + OutM.EM1O4;
    public final static int RotGelb  = OutM.EM1O1 + OutM.EM1O4 +
                                       OutM.EM1O2 + OutM.EM1O5;
    public final static int Gelb     = OutM.EM1O2 + OutM.EM1O5;
    public final static int Gruen    = OutM.EM1O3 + OutM.EM1O6;
    public final static int RotQ     = OutM.IF1O1 + OutM.IF1O4;
    public final static int RotGelbQ = OutM.IF1O1 + OutM.IF1O4 +
                                       OutM.IF1O2 + OutM.IF1O5;
    public final static int GelbQ    = OutM.IF1O2 + OutM.IF1O5;
    public final static int GruenQ   = OutM.IF1O3 + OutM.IF1O6;
    public final static int RotF     = OutM.EM2O1 + OutM.EM3O1;
    public final static int GruenF   = OutM.EM2O2 + OutM.EM3O2;
}
```

Konstanten in Form einer enum zum Schalten von jeweils zwei zusammengehörenden Ampellichtern.

```
import eduLine.*;
public class Ampeln {
    private CrossRoads cr;
    private boolean fussWunsch;
    private boolean anforderung;
    private boolean rotActive;

    public Ampeln() {
        cr = new CrossRoads();
        cr.openUSB(0, 0);
    }
```

Eine Klasse in der sich nun endlich um die Ampeln als Ganzes gekümmert wird. Die Klasse enthält noch ein paar globale Variable und weiter unten auch noch private Klassen. Dazu gehört dann auch noch die Klasse (eigentlich ein enum) Duos, die Konstanten zu paarweisen Schalten von gegenüberliegenden Ampeln enthält.

```
public void dreierAmpel()
{
    System.out.println("dreierAmpel gestartet");
    do {
        cr.setOutput(Out.EM1O3, Dir.On);
        cr.pause(3000);
        cr.setOutput(Out.EM1O3, Dir.Off);
        cr.setOutput(Out.EM1O2, Dir.On);
        cr.pause(1000);
        cr.setOutput(Out.EM1O2, Dir.Off);
        cr.setOutput(Out.EM1O1, Dir.On);
        cr.pause(3500);
        cr.setOutput(Out.EM1O2, Dir.On);
        cr.pause(1000);
        cr.setOutput(Out.EM1O1, Dir.Off);
        cr.setOutput(Out.EM1O2, Dir.Off);
    } while(!cr.finish());
    System.out.println("--- Finito ---");
}
```

Zunächst mal die Ampel unten links an der Hochstraße schaltet endlos ihren Standard-Zyklus : grün - gelb - rot - rotgelb.

```
public void paarAmpeln() {
    System.out.println("paarAmpeln gestartet");
    do {
        cr.setOutputs(Duos.Gruen);
        cr.pause(3000);
        cr.setOutputs(Duos.Gelb);
        cr.pause(1000);
        cr.setOutputs(Duos.Rot);
        cr.pause(3500);
        cr.setOutputs(Duos.RotGelb);
        cr.pause(1000);
    } while(!cr.finish());
    System.out.println("--- Finito ---");
}
```

Nun werden beide Ampel der Hochstraße synchron geschaltet. Dafür werden setOutputs und jeweils ein Element von Duos genutzt. Ausschalten ist hier nicht mehr erforderlich, das wird gleich mit dem Einschalten erledigt.

```
public void listenAmpeln() {
    int lampe;
    int[] takte = new int[] {
        Duos.Rot + Duos.GruenQ, Duos.Rot + Duos.GruenQ,
        Duos.Gelb + Duos.GelbQ,
        Duos.Gruen + Duos.RotQ, Duos.Gruen + Duos.RotQ,
        Duos.Gruen + Duos.RotQ,
        Duos.Gelb + Duos.GelbQ
    };
    System.out.println("listenAmpeln gestartet");
    do {
        lampe = cr.getVoltage() < 500 ? OutM.EM1O8 : 0;
        for(int i = 0; i < takte.length; i++) {
            cr.setOutputs(lampe + takte[i]);
            cr.pause(1000);
        }
    } while(!cr.finish());
    cr.clearOutputs();
```

```
        System.out.println("--- Finito ---");
    }
```

Nun ist es Zeit die Ampeln von Hoch- und Querstraße, auf einander abgestimmt, zu schalten. Das geschieht in festen Taktzeiten nach der Tabelle takte. Die Einträge für die Grün- bzw. Rotphasen sind mehrfach vorhanden, da sie üblicherweise länger sind als die Gelbphasen. Damit die Sache spannender wird, wird die Schaltung der Straßenbeleuchtung integriert : Zu Beginn eines Ampelzyklus wird die Helligkeit bestimmt und daraus abgeleitet ein bitWert zum Schalten der Straßenbeleuchtung abgeleitet (lampe). Dieser Wert wird dem Schaltwert aus takte einfach zuaddiert.

```
    private class FussTaste implements Runnable {
        @Override public void run() {
            do {
                fussWunsch = cr.getInput(Inp.IF1I1) ||
                             cr.getInput(Inp.IF1I2);
                cr.pause(123);
            } while (!cr.finish());
        }
    }
```

Die `run` - Methode für den Thread fuss. In `run` werden die FußTasten eines einfachen Fußgängerüberweges ständig abgefragt

```
    public void fussAmpel() {
        Thread fuss = new Thread (new FussTaste());
        fuss.start();
        System.out.println("fussAmpeln gestartet");
        do {
            if(fussWunsch) {
                fussWunsch = false;
                cr.setOutputs(Duos.GelbQ + Duos.RotF);
                cr.pause(1000);
                cr.setOutputs(Duos.RotQ + Duos.GruenF);
                cr.pause(3500);
                cr.setOutputs(Duos.GelbQ + Duos.RotF);
                cr.pause(1000);
            }
            cr.setOutputs(Duos.GruenQ + Duos.RotF);
            cr.pause(1000);
        } while(!cr.finish());
        cr.clearOutputs();
        System.out.println("--- Finito ---");
    }
```

Hier wird fusswunsch in einer Endlosschleife abgefragt und im true-Fall eine Fußgängerphase durch geführt. Zu Beginn der Methode wird ein Thread `fuss` zur Abfragen der Anforderungstasten angelegt und gestartet.

```
    private class Blitzen  implements Runnable {
        @Override public void run() {
            do {
                if(anforderung && rotActive) {
                    if(cr.getInput(Inp.EM1I5) &&
                      !cr.getInput(Inp.EM1I3)) {
                       cr.setOutput(Out.EM1O7, Dir.On);
                       cr.pause(444);
                       cr.setOutput(Out.EM1O7, Dir.Off);
                       anforderung = false;
                    }
                }
                else anforderung = cr.getInput(Inp.EM1I3);
                cr.pause(123);
            } while(!cr.finish());
        }
    }
```

Blitzen implementiert wieder die run-Methode eines Threads. Hier werden die Magnetsensoren in den Kanaldeckel überwacht wenn die Rotphase der Hochstraße aktiv ist und das nur dann, wenn EM1I3 (der untere Deckel) von einem Auto eingeschaltet wurde. In diesem Fall wird das Überfahren von EM1I5 (Deckel mitte rechts) kontrolliert und ggf. der RotBlitz ausgelöst.

```
    public void rotBlitzer() {
        anforderung = false;
        rotActive = false;
        Thread blitz = new Thread (new Blitzen());
        blitz.start();
        System.out.println("rotBlitzer ist aktiv");
        do {
            cr.setOutputs(Duos.Rot + Duos.GruenQ);
            rotActive = true;
            cr.pause(3500);
            rotActive = false;
            cr.setOutputs(Duos.Gelb + Duos.GelbQ);
            cr.pause(1000);
            cr.setOutputs(Duos.Gruen + Duos.RotQ);
            cr.pause(3500);
            cr.setOutputs(Duos.Gelb + Duos.GelbQ);
            cr.pause(1000);
        } while(!cr.finish());
        cr.clearOutputs();
        System.out.println("--- Finito ---");
    }
}
```

Eigentlich eine ganz normale Ampelschaltung. Einleitend wird allerdings der blitz-Thread mit drumRum aufgesetzt. Vergiftet wird die Angelegenheit durch die Kontrollmitteilung rotActive an den blitz-Thread für die Dauer der Rotphase.

eduLine.ZIP : Sources / DLLs

DLLs

eduLine.DLL Systemkonforme DLL zum Zugriff auf die Funktionen der education Line Devices

eduJava.DLL JNI-konforme Wrapper.DLL für den Zugriff aus Java

CrossRoads.DLL .NET Assembly mit Komfortfunktionen zum Zugriff auf die Funktionen der education Line Devices

VC++ 6.0

eduMain.CPP Testrahmen

eduLineVC.H Deklarationen für eduLine.DLL

eduLine.DEF Entrypoints von eduLine.DLL

eduLine.H Deklarationen in eduLine.DLL

eduLine.CPP Source der eduLine.DLL

Java

eduMain.java Testrahmen

eduLineVC.H (VC++ 6.0) : Deklarationen für eduLine.DLL

eduLine_eduJava.h (VC++ 6.0) : JNI Deklarationen für eduLine.DLL

eduLine.CPP (VC++ 6.0) : JNI-Methoden für eduLine.DLL

Verzeichnis **eduLine** :

- **eduLine.jar** Package für das Arbeiten mit der Klasse CrossRoads und die Sources dazu
- **eduJava**.java Klasse für den Zugriff auf eduJava.DLL
- **CrossRoads**.java Klasse für CrossRoads-Anwendungen
- **CrossRoadsException**.java
- **IFTypen**.java enum Klasse mit den unterstützten eduLine Devices
- **Dir**.java enum On / Off
- **Inp**.java enum der CrossRoads-Eingänge
- **Out**.java enum der CrossRoads-Ausgänge
- **OutM**.java enum Klasse der CrossRoads-Ausgänge, bitDarstellung

Delphi

HelloMain.pas Testrahmen

eduLine.pas Deklarationen für eduLine.DLL

Phyton

eduBlinker.py Testrahmen

eduCrossRoads.py Klasse CrossRoads mit Methoden zur Ansteuerung der eduLine.DLL

VB 2005

eduMain.vb Testrahmen

eduLinevb Deklarationen für eduLine.DLL

CrossMain.vb Umfangreichere Demo der Möglichkeiten der Klasse CrossRoads (siehe C# 2005)

C# 2005

eduMain.cs Testrahmen für eduLine.DLL

eduLine.cs Deklarationen für eduLine.DLL

CrossMain.cs Testrahmen für die Klasse CrossRoads

CrossSamplesMain.cs Umfangreichere Demo der Möglichkeiten der Klasse CrossRoads

CrossRoads.cs Klasse CrossRoads und Zubehör für eine komfortable Nutzung der education Line Devices auf Basis der eduLine.DLL