

## USBomat-XR

USBomat wurde entwickelt, um einen graphischen und zustandsorientierten Ansatz zur Ansteuerung von einfachen Kreuzungsmodellen aus LEDs und Schaltern über eine USB-Schnittstellenkarte zu verwirklichen. Die Benutzung einer Schnittstellenkarte (mit mehreren digitalen und analogen Ein- und Ausgängen) ließ darüber hinaus noch andere Anwendungen zu.

USBomat wurde im Rahmen einer Arbeit zur Binnendifferenzierung in heterogenen Gruppen entwickelt; als Alternative stand eine Entwicklungsumgebung mit einer formalen Sprache zur Verfügung, um die gleichen Aufgaben zu erledigen.

**USBomat-XR** ist die Anpassung von USBomat an die neuen Kreuzungs- und Ampelmodelle von Knobloch Elektronik.

## CrossRoads, TrafficLights und Signal

Die Modelle CrossRoads, TrafficLights und Signal von Knobloch Elektronik sind die modernen Nachfolger alter HIBS-Modelle. Sie können über einen USB-Anschluss direkt mit dem Computer verbunden werden. Nötige Treiber sind beim Hersteller erhältlich. USBomat-XR benötigt außerdem noch eine DLL (umFish40.DLL), um die Karten anzusprechen.

- **Signal** besitzt nur 3 LEDs (rot, gelb, grün), um eine Ampel zu simulieren.
- **TrafficLights** besitzt 7 LEDs in Form zweier Ampeln mit einer weiteren LED dazwischen, so dass sich auch ein elektronischer Würfel realisieren lässt, zudem einen Taster (genauer eine kapazitive Sensorfläche) und einen Helligkeitssensor (lichtempfindlicher Widerstand).
- **CrossRoads** ist ein Modell einer Kreuzung mit Ampeln für Autos und Fußgänger (insgesamt 30 LEDs), sowie Taster (für Fußgänger) und Magnetsensoren (als Modell für Induktionsschleifen) auf den Straßen und einen Helligkeitssensor.

Vor allem mit CrossRoads kann eine Ampelsteuerung samt Bedarfsschaltung und anderer Spezialitäten in großer Tiefe behandelt werden; die anderen Modelle sind eher zum Einstieg geeignet.

USBomat-XR kann zwischen *TrafficLights* und *CrossRoads* umgeschaltet werden; *Signal* wird am besten im *TrafficLights*-Modus angesteuert (Menü → USB-Karte)

## Bedienung

USBomat realisiert einen Moore-Automat mit Zuständen, denen jeweils eine Ausgabe (ein Muster von an- und ausgeschalteten Lämpchen) zugeordnet ist. Es ist immer ein Zustand gerade aktiv; dieser aktuelle Zustand wechselt je nach Eingabe (Tasten oder Zeit).

Einen neuen **Zustand** erzeugt man, indem man ihn aus dem 'Neu'-Feld herauszieht (Drag&Drop); man kann ihn benennen und seine Ausgabe (Lämpchen) ändern.

**Übergänge** erzeugt man, indem man zwei Zustände mit der rechten Maustaste verbindet.

Man muss angeben, wann der Übergang stattfinden soll:

- beim Drücken eines Tasters (schließen)
- beim Loslassen eines Tasters (öffnen, eher selten nützlich)
- oder automatisch nach einer einstellbaren Zeit (z.B. 5s oder 0,3s)

Ein Übergang kann auf mehrere Sensoren reagieren.

Mehrere Übergänge zwischen zwei Zuständen sind möglich.

Mehrdeutige Übergänge werden als Fehler markiert.

Oben neben dem Menü befinden sich Knöpfe

- zum Laden und Speichern der Automaten-Programme,
- zum Starten und Stoppen des Programms
- und zum Verbinden und Trennen der Karte.

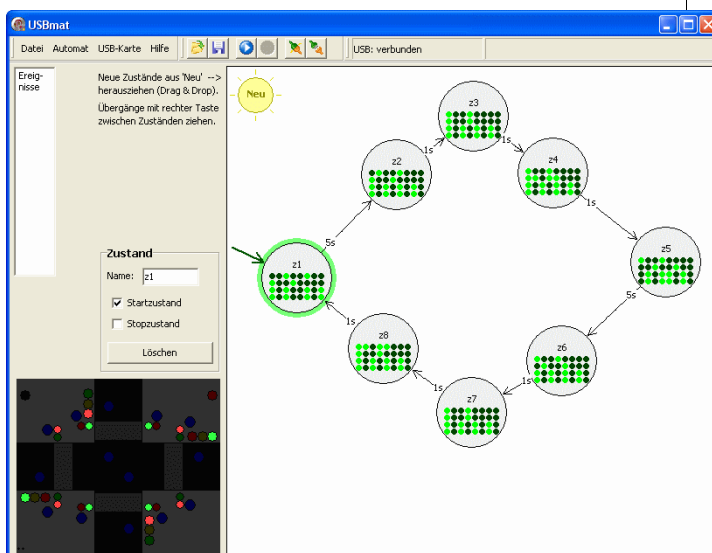
Im Menü unter USB-Karte kann außerdem zwischen *TrafficLights*- und *CrossRoads*-Modus umgeschaltet werden.

Hinweise:

Die kapazitiven Tastsensoren reagieren recht träge (etwa eine halbe Sekunde); das liegt am verwendeten Sensorprinzip der Modelle (die dafür keine beweglichen und fehleranfälligen Teile besitzen) und nicht am Programm. Die Magnetschalter reagieren schnell. Der Lichtsensor wird zur Zeit noch nicht unterstützt.

URL: [hbecker.sytes.net/usbomat](http://hbecker.sytes.net/usbomat)

mail: [hbecker@xmail.net](mailto:hbecker@xmail.net)



USBomat-XR im CrossRoads-Modus mit einem einfachen, zeitgesteuerten Programm, das Auto- und Fußgängerampeln in 8 Phasen schaltet. Das Programm liegt als 'zeitgesteuert.usbomat' bei.

Der folgende Text gibt in Auszügen einige didaktische Betrachtungen zu Automaten, Ampeln und USBomat wieder.

### Automatenmodelle

Innerhalb der zustandsorientierten Modellierung gibt es verschiedene Automaten-Modelle wie Endlicher Automat, Keller-Automat, Registermaschine oder Turing-Maschine, von denen es meisten in einer deterministischen und einer nicht-deterministischen Variante gibt. Das einfachste Automatenmodell ist dabei der Deterministische Endliche Automat (DEA). Für praktische Zwecke gibt es davon wiederum zwei Varianten, die in der technischen Informatik von großer Bedeutung sind<sup>1</sup>.

### Moore/Mealy

Die beiden DEA-Varianten unterscheiden sich in ihrem Ausgabe-Verhalten: Beim einfacheren Moore-Automaten wird die Ausgabe nur vom aktuellen Zustand bestimmt, während sie beim Mealy-Automaten eine Funktion von Zustand und aktuellem Eingabe-Symbol ist, also letztlich vom Zustandsübergang abhängt. Beide Automaten sind im Prinzip gleichwertig und ineinander umformbar, ein Mealy-Automat ist aber aufgrund seiner größeren Vielfalt oft kompakter als ein Moore-Automat, der die gleiche Aufgabe löst. Der Moore-Automat hat demgegenüber jedoch den Vorzug, leichter verständlich zu sein, denn im gleichen Zustand erfolgt immer die gleiche Ausgabe. Im Hinblick auf Ampelschaltungen ist der Moore-Automat das angemessene Modell, denn dann entspricht jeder Ampelphase (Zustand) eindeutig eine Festlegung, welche Lichter leuchten.

### Ampelschaltung

In den Empfehlungen des Didaktischen Forums sind Technikmodelle als Additum empfohlen, die auch im Schulcurriculum als Themenkreis 'Steuern und Regeln' gerne behandelt werden – sofern Technikmodelle samt brauchbarer Steuerungssoftware zur Verfügung stehen. Der Themenkreis Steuern und Regeln wird darüber hinaus für einige der Schüler auch beruflich relevant werden. Die Ampelschaltungen sollen daher nicht simuliert, sondern zumindest als Modell verwendet und vom Computer über die USB-Schnittstelle angesteuert werden. Einige Schülerinnen und Schüler erleben so vielleicht zum ersten Mal direkt, dass der Computer auch außerhalb des gewohnten Wirkungskreises (Monitor, Drucker, andere vernetzte Rechner) physisch etwas bewirken kann. Dadurch ist auch eine höhere Motivation bei den Schülerinnen und Schülern zu erwarten. Das mögliche Aufbauen von Ampel- und Kreuzungsmodellen und deren Steuerung über eine allgemeine Schnittstellenkarte erleichtert den Modellbildungsprozess und bietet einen enaktiven Zugang zum Thema; die fertigen Kreuzungsmodelle beschleunigen die Einarbeitung und lenken den Fokus auf die Programmierung.

### Graphisches Programmieren

Um die Probleme mit der Syntax formaler Sprachen ganz zu vermeiden, wurden Konzepte der graphischen Programmierung entwickelt. Meist werden in diesen

Programmierungsumgebungen Anweisungen und Kontrollstrukturen als graphische Elemente dargestellt und können mit der Maus aneinander plaziert oder mit Linien verbunden werden, die als 'roter Faden' den Ablauf der Befehls-Elemente festlegen. Grafischen Programmierungsumgebungen ist gemeinsam, dass sie die Benutzerin oder den Benutzer an die Hand nehmen und ihm eine begrenzte Anzahl von Möglichkeiten zur Auswahl bieten, die in das graphische Programm eingefügt werden können. Im Gegensatz dazu darf der Benutzer bei formalen Programmiersprachen im Prinzip alles eintippen, also auch beliebigen 'Unsinn', und welche Worte (Befehle) in welcher Anordnung einen Sinn ergeben, muss man im Prinzip auswendig wissen<sup>2</sup>.

### USBomat

Die Entwicklung von USBomat ist zum einen geleitet von dem Gedanken, das Programmieren noch einfacher zu gestalten und sprachliche bzw. syntaktische Probleme auszublenden. Es widmet sich außerdem dem Problem, dass die meisten Einstiege ins Programmieren (ob graphisch oder sprachlich) den modellbildenden Charakter nur wenig betonen – mit Ausnahme von Kara<sup>3</sup>, das als Anregung für USBomat diente. Oft neigen die Schülerinnen und Schüler dazu, einfach 'drauf los' zu programmieren und herumzuprobieren. Um aber sinnstiftend Konzepte der Informatik zu erlernen, soll es sich beim Programmieren "um die Implementierung eines vorher entwickelten Modells handeln, also keinesfalls um Stegreifprogrammierung"<sup>4</sup>. USBomat fordert das Entwickeln eines Zustandsmodells dadurch ein, dass das Zustandsmodell bereits dem graphischen Programm entspricht. USBomat kann auch ansatzweise als Werkzeug zur Entwicklung eines Zustandsmodells genutzt werden. Die Zustände lassen sich benennen, und die Ausgabe besteht aus den Schaltzuständen der einzelnen Ausgänge der USB-Karte. Übergänge werden mit der Maus gezogen. Als Eingabe lassen sich die Eingänge der USB-Karte nutzen, aber auch zeitgesteuerte Übergänge sind möglich<sup>5</sup>. USBomat zeigt die aktuelle Ein- und Ausgabe graphisch an, und die Eingänge können durch Maus oder Tasten simuliert werden, so dass man auch ohne angeschlossene Karte schon vieles ausprobieren kann; die Schüler können somit auch zuhause ohne Karte arbeiten. Widersprüchliche Übergänge werden als Fehler rot hervorgehoben, damit sie sofort erkannt werden können. Ein Unterschied zum Moore-Automaten besteht darin, dass nicht von jedem Zustand aus ein Übergang für jeden Tastendruck angegeben werden muss. Nicht spezifizierte Eingaben bewirken dann einfach keine Zustandsänderung. Aus diesem Grund sind auch Übergänge von einem Zustand zu sich selbst weder nötig noch implementiert.

2 Oder jedes Mal in der Hilfe nachlesen, aber das bremst ungemein. Moderne IDEs unterstützen ihre Benutzer/innen durch Codevervollständigung von Befehlen samt ihrer Parameter, so dass man auch hier je nach Kontext aus einer Liste passender Befehle auswählen kann.

3 <http://www.swisseduc.ch/informatik/karatojava/kara/>

4 P. Hubwieser: "Didaktik der Informatik", Springer Verlag 2000, S. 86

5 Genau betrachtet wird der Automat durch die Zeitabhängigkeit eigentlich sehr kompliziert, denn die verstrichene Zeit ist im Prinzip Teil des Zustands des Automaten. Diese Betrachtung ist allerdings weder handhabbar noch hilfreich, deshalb wird hier einfach das Verstreichen einer eingestellten Zeit als eine Eingabe modelliert.

1 Automaten vernetzen also im Sinne kumulativen Lernens verschiedene Bereiche innerhalb der Informatik. USBomat kann daher beispielsweise auch noch kurz in der Oberstufe zur Einführung in Automatenmodelle genutzt werden.