LBlocks:

Programmieren ohne Tastatur

Benutzerhandbuch

Version 3 / Rev. A

Inhaltsverzeichnis

Abbildungsverzeichnis

Kapitel 1

Bevor es los geht

1.1 Vor- Vorwort

Da ich ein begeisterter Anhänger freier Software bin, unterliegen die Programme der GPL (GNU General Public Licence) Version 3 oder höher. Letztendlich bedeutet das, dass Ihnen die vier Grundfreiheiten freier Software gewährt werden.

- Sie können das Programm zu jedem beliebigen Zweck verwenden (sofern es sich dazu eignet)
- Sie können die Funktionsweise des Programmes studieren, dazu steht Ihnen der Quelltext zur Verfügung
- Sie können das Programm ohne Einschränkungen weitergeben
- Sie können das Programm modifizieren und Ihre modifizierte Variante unter Beibehaltung der Lizenz weitergeben

Die Dokumentation unterliegt der FDL (Free Document License).

Die Veröffentlichung dieses Projekts erfolgt in der Hoffnung, daß es Ihnen von Nutzen sein wird, aber OHNE IRGENDEINE GARANTIE, auch ohne die implizite Garantie der MARKTREIFE oder der VERWENDBARKEIT FÜR EINEN BESTIMMTEN ZWECK.

Alle im Text genannten Marken sind Eigentum des entsprechenden Inhabers.

1.2 Vorwort

Nachdem unser Sohn jetzt 7 Jahre alt ist und sich für Papas Basteleien interessiert, war es einfach Zeit dafür. Ausschlaggebend war dann aber Lego Mindstorms auf einer Messe. So etwas ähnliches müsste sich doch auch selbst bauen lassen. Pläne dazu gab es schon länger, über mehr als Konzeptionen ging es aber nie hinaus.
Herausgekommen ist eine Art „Grafische Programmiersprache", mit der sich einfache Programme mittels verschiedener Elemente konstruieren lassen. Dabei habe ich viel Wert auf kindgerechte Bedienung gelegt. Ob das gelungen ist, muss wohl jeder für sich selbst entscheiden.
Das Programm ist konsequent auf Maus-Bedienung ausgelegt und bereits auf Touch-Bedienung vorbereitet. Deshalb sind einige Dinge anders als man sie bei anderen Programmen vorfindet. Es gibt kein Menü und alle Dialoge, Abfragen etc. finden im Hauptfenster statt und haben relativ große Bedienelemente.

Dieses Büchlein erkärt Schritt für Schritt, wie man selbst Programme erstellt, für den Aufbau des Controllers und die Installation und Einrichtung der Software gibt es das Technik-Manual. Beginnend mit einfachen Aufbauten werden die Projekte im Verlauf immer ausgefeilter und komplexer.

Um nicht für jede neue Version die Dokumentation ändern zu müssen, kann es durchaus sein, dass einige Screenshots in der Dokumentation vom aktuellen Programm abweichen. Das allerdings nur in nicht relevanten Bereichen.

1.3 Aufbau der Experimente

Zu jedem Experiment oder Projekt oder wie man es auch nennen möchte, gibt es eine Hardware- und einen Softwareteil. Die Hardwarebeschreibung gibt an, welcher Peripheriebaustein an welchen Anschluss des Controllers angeschlossen wird und sieht in der Regel so aus:

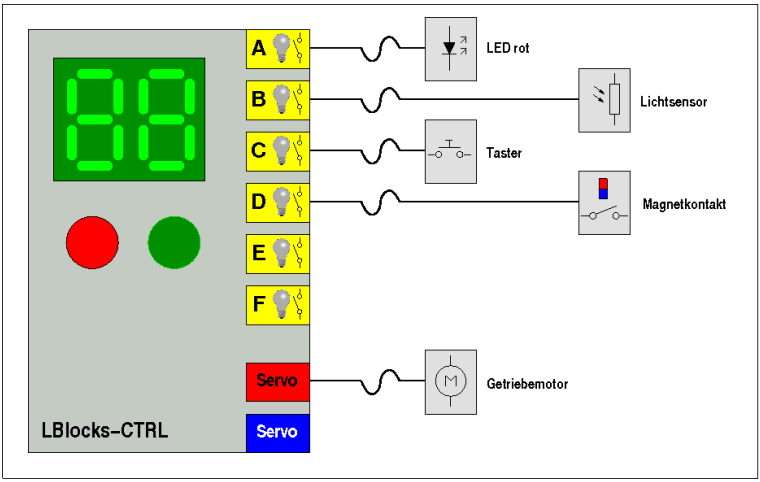


Abbildung 1.1: Eine Beispielschaltung

Links ist symbolisch der LBlocks Controller dargestellt, rechts daneben die verbundenen Peripheriebausteine. Da es keine Vorgaben gibt wie die Peripheriebausteine aussehen müssen, können Ihre Aufbauten natürlich stark von meinen abweichen. Aus diesem Grund werden alle Bausteine nur symbolisch dargestellt.
Um die Peripheriebausteine möglichst flexibel nutzen zu können, habe ich diese in LEGO Bausteine eingebaut.
Wie das aussehen und wie man sich selbst einen LBlocks-Controller bauen kann ist nicht Bestandteil dieses Büchleins, dafür gibt es die technische Dokumentation.

Parallel dazu gibt es einen oder mehrere Logikpläne, da ein Programm aus mindestens einem Hauptprogramm und bis zu vier Unterprogrammen (Boxen) bestehen kann.

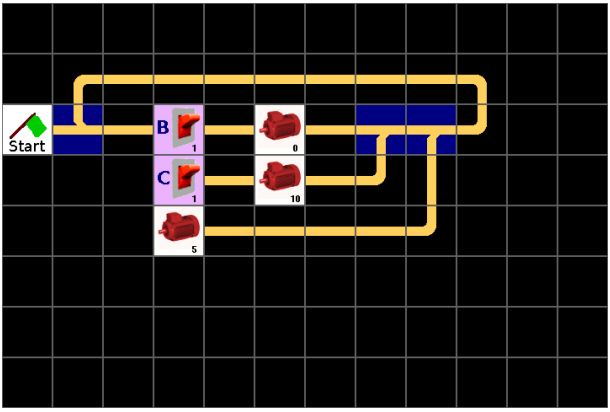


Abbildung 1.2: Ein Beispielprogramm

Um die eigenen Experimente zu dokumentieren ist übrigens die Snapshot-Funktion gedacht. Diese wird über die Schaltfläche mit dem Fotapparat ausgelöst und speichert das Bild im PNG-Format als „screenshot.png" im **lblocks** Ordner im Benutzerverzeichnis.

Kapitel 2

Erste Schritte

2.1 Programmstart

Eigentlich gibt es zwei Programme. Das „normale" **lblocks** für Auflösungen ab 1024x768 und ein **lblocks-small** welches auch für kleinere Auflösungen ab 640x480 geeignet ist. Die X86-Binärdateien sind für PCs mit 32-Bit und 64-Bit (mit zusätzlichen 32-Bit Bibliotheken) geeignet, die ARM-Binärdateien wurden auf einem Emulator für das Raspberry Pi getestet.
Das Programm kann direkt vom Desktop oder auch aus einer Konsole heraus gestartet werden. Im letzteren Fall lässt sich Fehlverhalten des Programmes leichter diagnostizieren, da z.B. auch die Meldungen des Programmers hier angezeigt werden.
Auch gibt es

- **–m88** erzeugt Code für einen ATMega88 (zur Zeit noch nicht getestet
- **–en** die Programmoberfläche wird in englischer Sprache dargestellt (weitestgehend komplett)
- **–ro** die Programmoberfläche wird in rumänischer Sprache dargestellt (noch unvollständig)

2.2 Bedienelemente

Im folgenden Beispiel wird gezeigt, wie mit wenigen Schritten ein Programm erstellt wird. Nachdem das Programm gestartet ist, sieht das Programmfenster so aus:

Abbildung 2.1: Der Startbildschirm

Ganz oben im Fenster befinden sich verschiedene Aktionsknöpfe, die wir im weiteren Verlauf noch kennenlernen werden. Mit dem Info-Knopf ganz rechts gelangt man immer wieder zum Startbild wie wir es gerade sehen.

Darunter ist das Eigenschaften-Feld, in dem die Eigenschaften der Elemente geändert werden können und in dem auch Frage-Dialoge beantwortet werden. Da wollen wir auch gleich tun und klicken auf die gelbe Schaltfläche „Beispiele installieren". Damit lassen sich die Beispiel-Programme aus der Anleitung einfach in den Daten-Ordner kopieren. Diese erkennt man an einem **xs-** am Anfang des Programmnamens. Aber Vorsicht! Wenn man die Beispiele bearbeitet, wird durch diese Funktion der originale Zustand wiederhergestellt und alle Änderungen sind somit verloren. Deshalb sollte vor dem Bearbeiten den Beispielen als erstes ein neuer Programmname gegeben werden. Mit der Schaltfläche „Controller einrichten" werden die Fuses eines angeschlossenen Controllers auf die erforderlichen Werte gesetzt.

Nun sollte das Programmfenster wie folgt aussehen:

Abbildung 2.2: Beispiele installieren

Links im Eigenschaften-Feld steht nun ein Fragezeichen-Symbol, die Frage, ob die Beispiele installiert werden sollen und zwei Schaltflächen mit **Ja** und **Nein**. Solche Fragen werden immer dann gestellt, wenn durch Aktionen Daten gelöscht werden oder verlorengehen können. An der Stelle vom Fragezeichen-Symbol können in bestimmten Situationen auch eines der folgenden Symbole zu sehen sein:

Wenn dieses Symbol angezeigt wird. dann wurde eine Aktion erfolgreich abgeschlossen. Zum Beispiel wenn das aktuelle Programm erfolgreich gespeichert oder auf den LBlocks-Controller übertragen wurde.

Dieses Symbol zeigt an, wenn etwas nicht geklappt hat. Zum Beispiel wenn man Programme vom LBlocks-Controller lesen oder darauf schreiben will und kein Controller oder Programmer angeschlossen ist.

Jetzt klicken wir auf das grüne **Ja** - Feld und danach sollte das Programmfenster wie unten auf dieser Seite dargestellt aussehen.

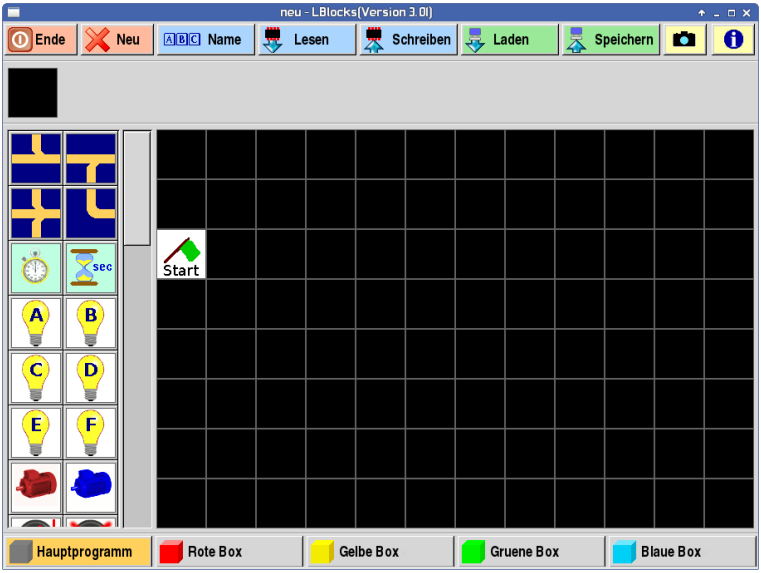


Abbildung 2.3: Die Aufteilung des Fensters

Unter dem Eigenschaften-Feld, welches jetzt nur noch ein schwarzes Quadrat als Symbol enthält folgen folgen der Listenbereich und die Arbeitsfläche. Darunter sind noch 5 Felder, wovon eins einen gelben Hintergrund hat und die anderen nur einen grauen. Damit kann zwischen 5 verschiedenen Arbeitsflächen umgeschaltet werden. Am Anfang werden wir uns jedoch nur auf der Hauptprogramm-Arbeitsfläche bewegen. Klickt man auf ein Element im Listenbereich, so wird im Eigenschaften-Feld ein Hinweis auf die Funktion dieses Elements angezeigt.
Wenn keine Beispiele intalliert und auch kein Controller eingerichtet werden soll, kommt man zu diesem Pnkt auch durch klicken auf das große Bild auf der rechten Seite des Fensters. Das Bild verschwindet und es kommt ein Raster von 8 mal 12 Feldern zum Vorschein. Das ist sozusagen unsere Arbeitsfläche. Links daneben gibt es einen Bereich in dem wie in einer Liste alle verfügbaren Elemente angeordnet sind. Im weiteren Verlauf werde ich diesen als Listenbereich bezeichnen.
Klickt man auf ein Element im Listenbereich, so wird im Eigenschaften-Feld ein Hinweis auf die Funktion dieses Elements angezeigt.

2.3 Ein Programm entsteht

Die meisten Felder auf der rechten Seite sind schwarz, nur auf der linken Seite sehen wir win weißes Feld mit einer grünen Flagge. Das ist das Startfeld, bei dem alle Programme beginnen. Jetzt werden wir ein erstes Element in unser Programm einfügen. Und zwar wollen wir eine Lampe einschalten. Dafür gibt es das Listenfeld mit einer Glühlampe und dem Buchstaben **A** als Symbol. Klicken wir auf dieses Feld, wird „Helligkeit Anschluss A" angezeigt. Damit ist der Anschluss A am Controller gemeint, wir können eine Lampe dort anschließen und deren Helligkeit mit diesem Feld festlegen.
Also ziehen wir jetzt das Feld mit der Lampe vom Listenfeld auf die Arbeitsfläche. Das geht indem man auf das Feld im Listenbereich mit der linken Maustaste klickt und bei gedrückter Maustaste auf die Arbeitsfläche zieht. Dabei bleibt das Symbol sozusagen am Mauszeiger „kleben". Das gerade aktive Feld auf der Arbeitsfläche wird mit einem hellgrauen Rechteck markiert, wird jetzt die Maustaste losgelassen befindet sich nun an der Stelle das Feld mit der Lampe.
Ziel ist es, die Lampe rechts neben dem Startfeld zu platzieren und dabei ein Feld Zwischenraum zu lassen.

Das Ergebnis sollte nun so aussehen:

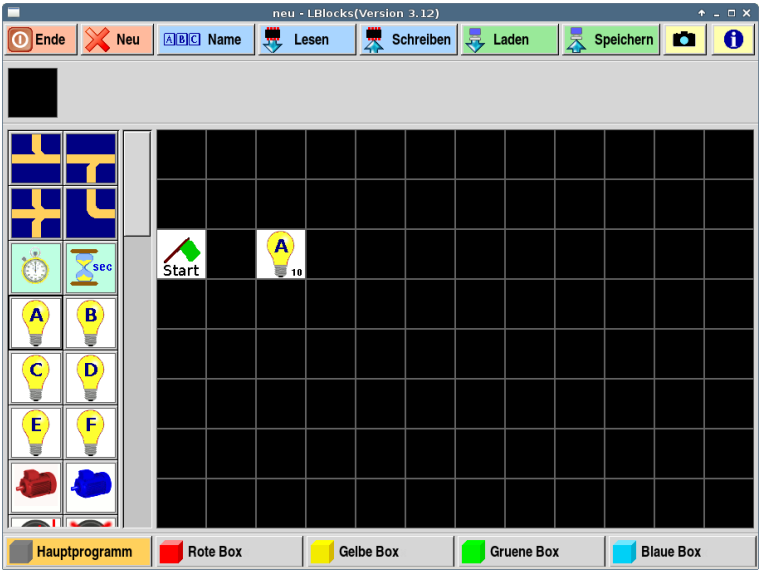


Abbildung 2.4: Ein Element platzieren

Wenn es nicht gleich gelingt, kann man auf die gleiche Weise Elemente innerhalb der Arbeitsfläche verschieben. Was nicht geht, Elemente auf bereits belegte Felder zu verschieben. Das kann man ausprobieren, indem man das Feld mit der Lampe auf das Startfeld zu ziehen versucht.
Um Elemente aus der Arbeitsfläche zu löschen, muss man diese nur mit der linken Maustaste fassen und auf das Listenfeld ziehen, sozusagen wieder zurück in die Vorratskiste legen.
Rechts unten im Feld mit der Lampe steht eine kleine „10”. Bei einigen Element-Typen gibt es einen einstellbaren Eigenschaftswert, und dieser wird rechts unten im Feld angezeigt.

Wenn man jetzt auf das Feld mit der roten Lampe klickt, bekommt es einen farbigen Rand und das Eigenschaften-Feld sieht jetzt anders aus:

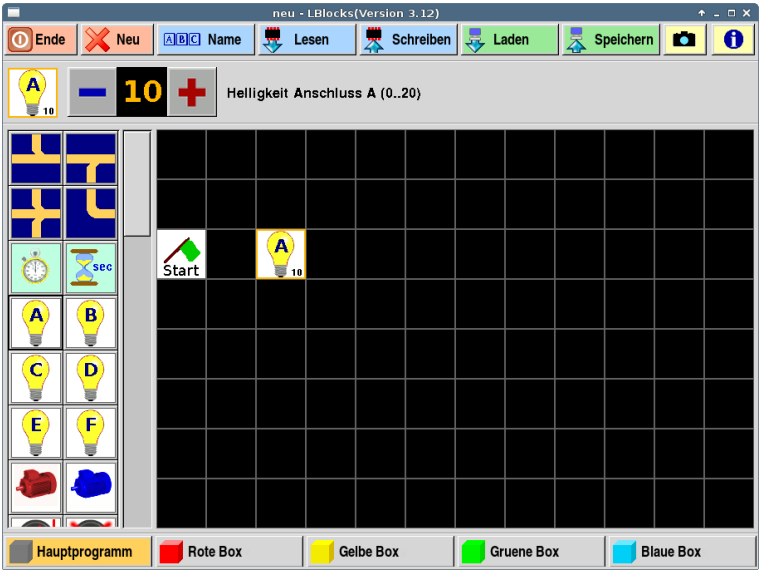


Abbildung 2.5: Elementeigenschaften ändern

Im Eigenschaften-Feld ist jetzt das ausgewählte Element zu sehen, daneben gibt es eine „+" und eine „-" Schaltfläche mit einer Zahl dazwischen. Diese entspricht dem Eigenschafts-Wert unseres Lampen-Elements. Dabei entspricht der Wert 0 einer ausgeschalteten Lampe und der Wert 20 dass die Lampe mit voller Helligkeit leuchtet. Indem wir auf die Bildschirmtaste „+" klicken, können wir jetzt die Helligkeit auf den Wert 20 verändern. Zusätzlich zur Zahl wischen der „+" und „-" Schaltfläche ändert sich bei jedem Klick auch der Eigenschaftswert im ausgewählten Feld.
Rechts neben den Schaltflächen wird auch die zu ändernde Eigenschaft sowie der mögliche Wertebereich angezeigt. Und der Wert, in unserem Beispiel die Helligkeit am Anschluss **A** des Controllers lässt sich auch nur in diesem Bereich einstellen.

Nun haben wir den Helligkeitswert 20 eingestellt und müssen noch irgendwie festlegen, dass der Controller als Aktion genau die Helligkeit am Anschluss **A** einstellen soll. Dazu müssen wir das Startfeld mit dem Feld unseres Lampen-Elements verbinden. Zuerst klicken wir zweimal auf das Startfeld. Der Hintergrund des Feldes wird jetzt gelb und im Eigenschaften-Feld steht jetzt: **Jetzt auf Zielfeld der Verbindung klicken**. Das tun wir auch und klicken auf das Feld mit unserem Lampen Element. Nun sind das Startfeld und das Feld mit dem Lampen-Element durch eine dicke orange Linie verbunden:

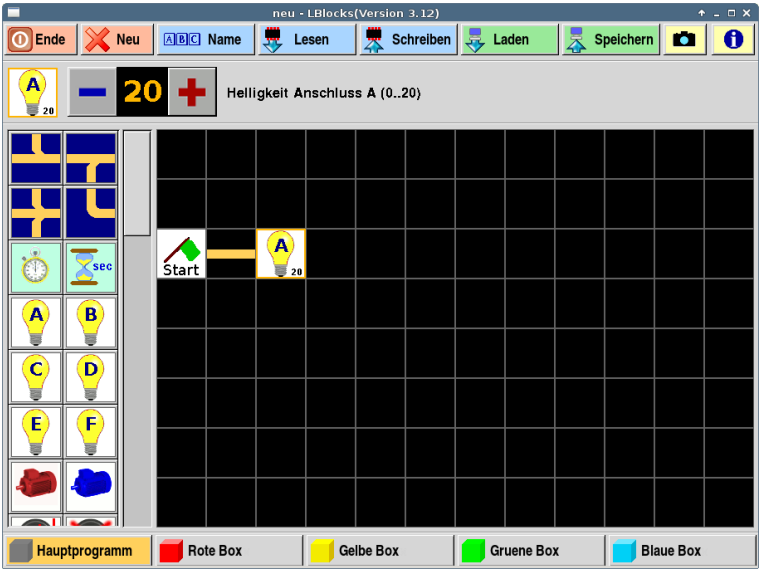


Abbildung 2.6: Elemente verbinden

Wir hätten auch das Lampen-Element direkt rechts neben dem Startfeld positionieren können, das Ergebnis wäre das gleiche. Die Anordnung der Elemente neben- und untereinander bestimmt die Reihenfolge, wie der Controller die Elemente abarbeitet. In diesem Fall besteht das Programm aus dem Startfeld, welches immer vorhanden ist und unserem Lampen-Element. Nachdem beide Elemente abgearbeitet sind, wird das Programm anhalten. Später werden wir noch sehen, dass Elemente auch untereinander angeordnet werden können, vorerst genügt uns dieses einfache Programm.

2.4 Das Ding bekommt einen Namen

Ganz oben in der Titelleiste des Fensters steht der Programmname, zu Beginn ist er immer **Neu**. Da das kein besonders aussagekräftiger Name für unser Programm / Experiment ist, wollen wir ihn jetzt ändern. Dazu klicken wir in der Aktionsleiste auf die blaue Schaltfläche mit der Aufschrift **Name** und statt unserer Arbeitsfläche sehen wir jetzt ein verkleinertes Bild von unserer Arbeitsfläche und daneben den aktuellen Name des Programmes. Darunter ist eine Bildschirmtastatur zu sehen.

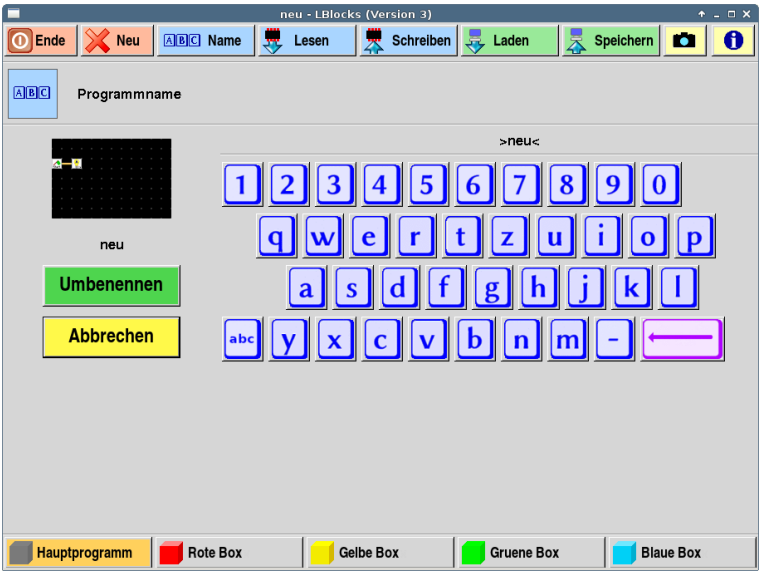


Abbildung 2.7: Programmnamen ändern

Nun können wir unserem Programm einen neuen Namen geben. Als Beispiel wollen wir den Namen „TEST" festlegen. Neben den Buchstaben- und Zahlentasten gibt es noch ein paar Sonderfunktionen:

- Die **Minus** Taste schreibt ein Minus in den Namen. Das ist das einzig mögliche Sonderzeichen.
- Mit der **Pfeil zurück** Taste in der oberen Reihe wird der letzte Buchstabe/Zahl gelöscht
- Mit der **abc** Taste wird zwischen Klein- und Großbuchstaben umgeschaltet

- Mit Klick auf das **Umbenennen**-Feld im linken Bereich wird der neue Name übernommen
- Ein Klick auf das **Abbrechen**-Feld bricht die Namensänderung ab

Also löschen wir jetzt mit der **Pfeil-zurück** Taste das Wort „Neu". Ein Klick auf die **abc** Taste ändert deren Aussehen. Die Farbe hat von blau nach rot gewechselt und anstelle von „abc" steht jetzt „ABC". Dies bedeutet dass jetzt Großbuchstaben eingegeben werden können. ein erneuter Klick schaltet wieder auf Kleinbuchstaben um und so weiter. Jetzt können wir aber den neuen Namen „TEST" eingaben, indem wir nacheinander auf die Buchstabentasten **T**, **E**, **S** und **T** klicken. Dabei ändert sich der Name neben dem Vorschaubild entsprechend. Die „>" und „<" Zeichen dienen nur zur optischen Begrenzung des Namens. Die Anzeige sollte jetzt so aussehen:

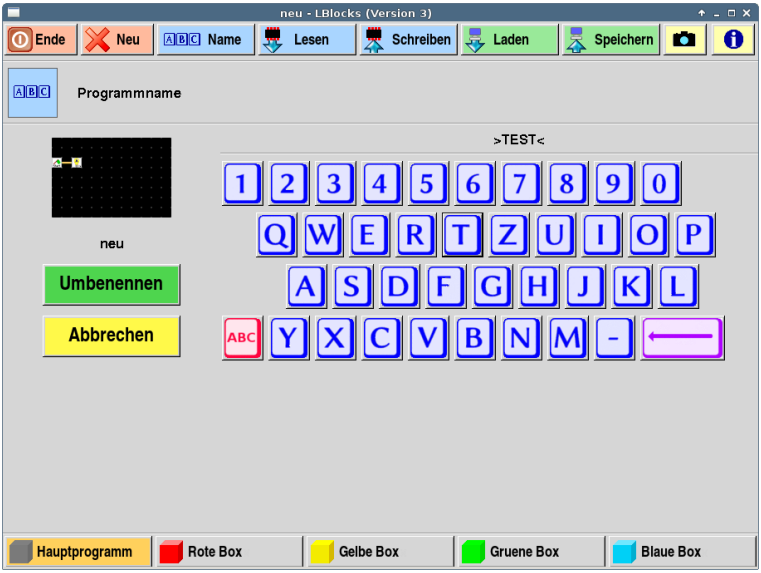


Abbildung 2.8: Der neue Programmname

Mit einem Klick auf das **Umbenennen**-Feld wird wieder zurück zur Arbeitsfläche gewechselt und der Name in der Titelleiste des Fensters hat sich in „TEST" geändert.

2.5 Speichern im Computer

Würden wir jetzt das Programm beenden wäre unser Programm „weg", also wollen wir es im Computer dauerhaft speichern. Das geht ganz einfach, ein Klick auf die Schaltfläche „Speichern" in der Aktionsleiste genügt. Ein grünes Symbol im Eigenschaften-Feld zeigt an, dass die Aktion erfolgreich war:

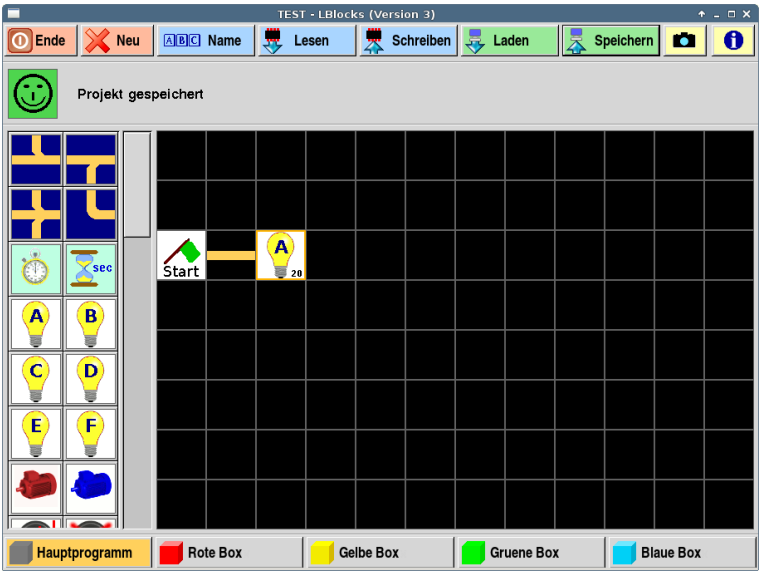


Abbildung 2.9: Das Programm ist gespeichert

Wir werden später das Programm unter seinem Namen „TEST" später wieder laden. Jetzt stellt sich noch die Frage was passiert, wenn „TEST" schon vorhanden wäre. Einerseits wollen wir vielleicht das Programm nach einer Korrektur neu abspeichern und andererseits könnten wir auch vergessen haben den Namen zu ändern oder haben uns verschrieben. Wenn das Programm schon im Computer gespeichert ist, wird darauf hingewiesen und die Frage gestellt, ob das bestehende Programm überschrieben werden soll.

2.6 Ein gespeichertes Programm laden

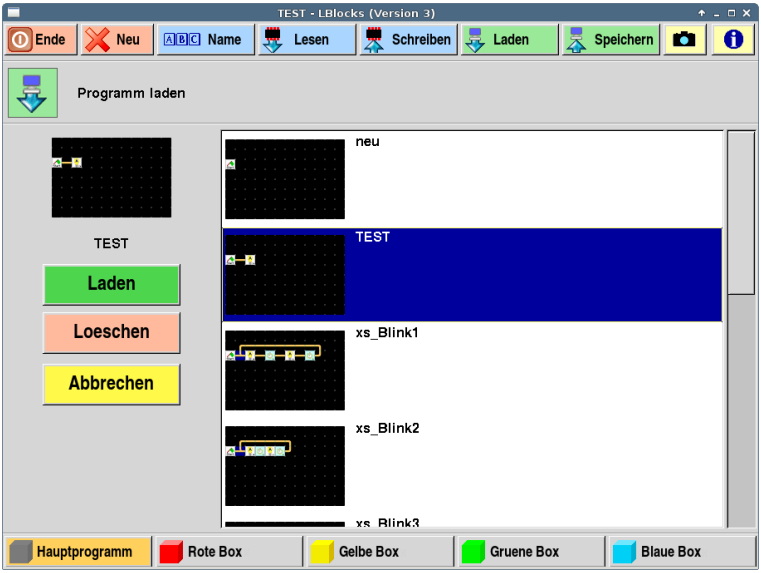


Abbildung 2.10: Ein Programm laden

Durch Klick auf die Schaltfläche „Laden” in der Aktionsleiste wird eine Auswahlliste der auf dem Computer gespeicherten Programme angezeigt. Zuerst wird das zu ladende Programm im rechten Bereich ausgewählt und dann die Auswahl mit dem **Laden**-Feld im linken Bereich bestätigt. Das Vorschaubild im linken Bereich zeigt das gerade bearbeitete Programm an, beim Laden eines anderen Programmes werden diese Daten im Speicher des Computers überschrieben.
Gleichzeitig ist es mit der Schaltfläche **Löschen** möglich, einzelne Programme aus der Liste zu löschen. Damit dies nicht unabsichtlich geschieht, gibt es vorher eine Sicherheitsabfrage. Und Schließlich kann die Aktion mit dem **Abbrechen**-Feld auch ohne Laden eines Programmes beendet werden.

2.7 Ab in den Controller...

Jetzt wird es aber Zeit, dass wir unser Programm in den LBlocks-Controller übertragen. Dazu dient die Schaltfläche „Schreiben” in der Aktionsleiste. Für den Fall, dass kein passender Controller angeschlossen ist, erhält man nur eine Fehlermeldung:

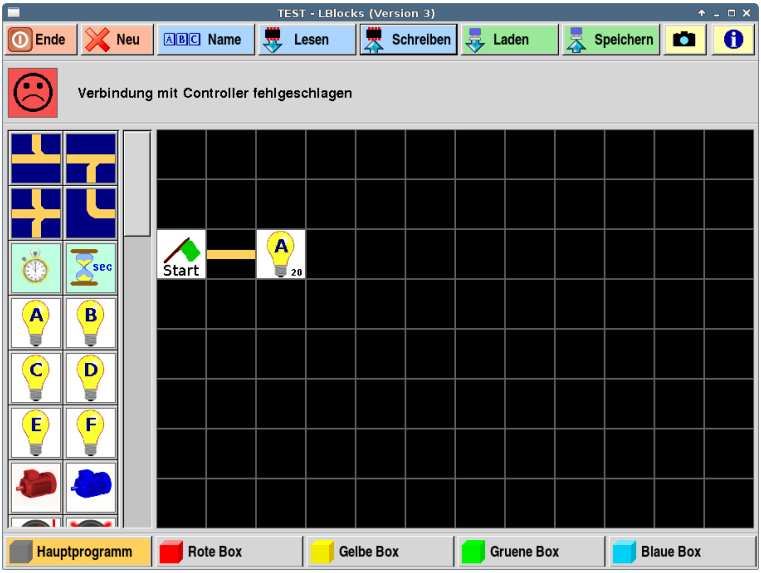


Abbildung 2.11: Das Schreiben ist fehlgeschlagen

Sollte ein LBlocks-Controller via Programmer angeschlossen sein und trotzdem nur die Fehlermeldung erscheint, ist es meistens hilfreich, das Programm zu beenden und von einem Terminalfenster aus zu starten. In diesem Fenster werden dann die Meldungen des Programmers angezeigt, was dann meist schnell zur Ursache führt. Fehlerursache könne z.B. sein:

- Der Controller ist ausgeschaltet und wird nicht vom Programmer versorgt
- Der Programmer ist nicht richtig angesteckt oder es fehlen Schreibrechte
- Es wird ein anderer Programmer als avrdude/USBASP verwendet und die Scripte wurden nicht angepasst

Wenn das Verbinden erfolgreich war und die im Controller gespeicherten Programme ausgelesen werden konnten, wird jetzt im Fenster ein ähnliches Bild wie beim Laden von Programmen angezeigt.

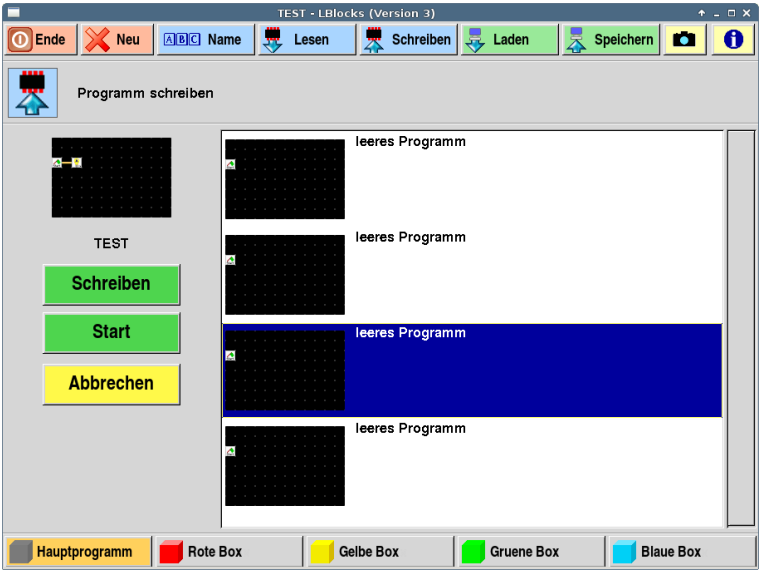


Abbildung 2.12: Das Schreiben hat geklappt

In jedem Controller können 4 unterschiedliche Programme gespeichert werden. Dazu gibt es im rechten Bereich eine Auswahlliste, mit den im Controller gespeicherten Programmen. Hier könne wir auswählen, auf welchen Programmplatz wir unser Programm speichern wollen. Als Beispiel verwende ich hier das 3.Programm. Links gibt es neben der obligatorischen **Abbrechen**-Schaltfläche eine mit **Schreiben** und eine mit **Start**. Die Funktion **Abbrechen** sollte eigentlich klar sein, mit **Schreiben** wird das Programm in den LBlocks-Controller geschrieben. Wird **Start** ausgewählt, wird das Programm in den Controller geschrieben und gleich gestartet. Das besondere daran ist, dass jetzt dieses Programm bei jedem Einschalten des LBlocks-Controllers gestartet wird.
Aber jetzt reicht es aus, das Programm nur in den Controller zu schreiben. Nun kann das Programm am Controller gestartet werden. Dazu wird mit der Stop-Taste das Programm 3 gewählt (Anzeige **P3**) und mit der Starttaste gestartet. Die „RUN"-LED leuchtet und auf der Anzeige steht **- -**. Das bedeutet, dass das Programm zu Ende ist. Naja, viel gab es da wohl nicht zu sehen...

Aber halt, was war mit der Lampe an Anschluss **A**? Die war ja gar nicht angeschlossen. Also drücken wir die Stop-Taste am Controller und schließen eine Lampe (LED) an den Anschluß **A** an.

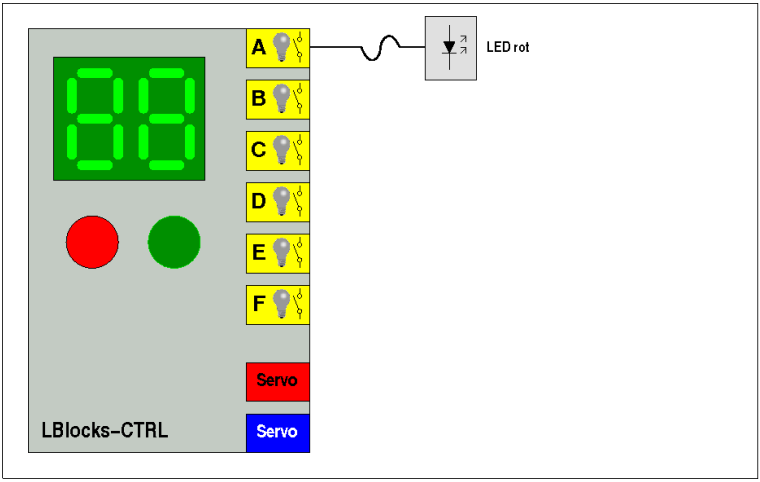


Abbildung 2.13: Die Schaltung für unseren Test

Wenn wir jetzt wieder sie Start-Taste drücken, kommt zwar wieder die Anzeige **- -**, die LED am Anschluss **A** sollte aber jetzt hell leuchten. Nach Betätigen der Stop-Taste wird wieder **P3** angezeigt und die angeschlossene LED ist wieder aus.

2.8 ...Und wieder zurück

Natürlich können auch Programme vom LBlocks-Controller in den Computer geladen werden um sie beispielsweise zu ändern. Und, wer hätte das gedacht, dazu dient die Schaltfläche „Lesen” in der Aktionsleiste.

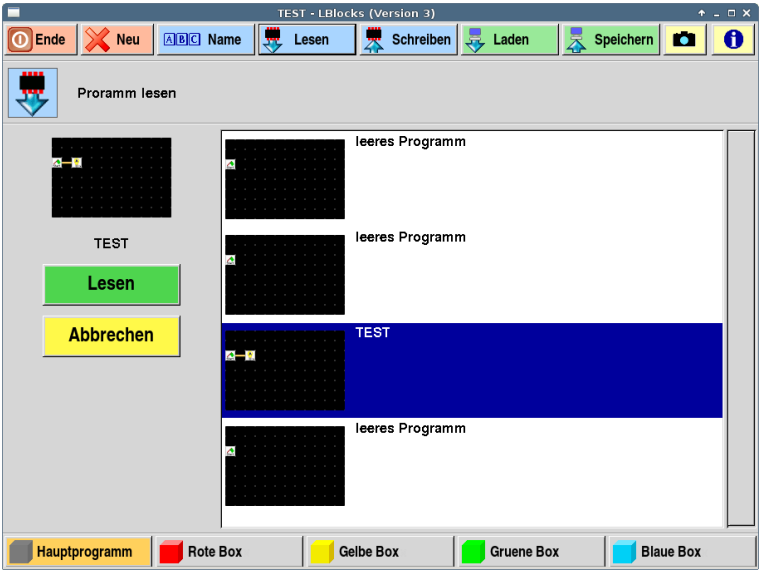


Abbildung 2.14: Zurücklesen eines Programmes

Das 3.Programm, welches wir beim Speichern benutzt haben, ist jetzt schon vorgewählt. Mit der Schaltfläche Das Vorschaubild im linken Bereich zeigt das gerade bearbeitete Programm an, beim Lesen eines Programmes aus dem Controller werden diese Daten im Speicher des Computers überschrieben.

Kapitel 3

Die Elemente

3.1 Allgemeines

Jedes Programm beginnt beim Startfeld, das Ende ist dann erreicht wenn kein Element mehr abgearbeitet werden kann. Man kann sich das so vorstellen, dass eine Art „Staffelstab" existiert, der von Element zu Element weitergegeben wird. Die Abarbeitung erfolgt von links nach rechts und es dort nicht weitergeht von oben nach unten. Die Entscheidung ob und wie es weitergeht, hängt vom jeweiligen Element ab. Der „Eingang" jedes Elements ist links, der „Ausgang" auf der rechten Seite. Verbindungen können nur vom „Ausgang" eines Bausteins zum „Eingang" eines anderen Bausteins gezogen werden. Zusätzlich können Elemente untereinander angeordnet werden. Jedes Element hat eine interne Bedingung, die entweder WAHR oder FALSCH ist. Der „Staffelstab" wird nur zum Ausgang eines Elements herausgegeben, wenn die interne Bedingung WAHR ist. Ist die Interne Bedingung FALSCH oder fehlt am Ausgang ein nachfolgender Eingang, so kann unser „Staffelstab" an ein darunter liegendes Element weitergegeben werden.
Liegen mehrere Elemente untereinander, so wird der Stab immer wieder von oben nach unten weitergegeben, bis bei einem der Elemente die interne Bedingung WAHR ist und sein Ausgang mit dem Eingang eines anderen Elements verbunden ist. Das klingt zwar jetzt recht kompliziert, wird aber im Laufe der Beispiele noch weiter und in kleinen Schritten erklärt.

3.2 Bschreibung der Elemente

3.2.1 Ausgabefunktionen

Bei sämtlichen Ausgabefunktionen ist die interne Bedingung immer WAHR. Das heißt, sobald der Ausgang mit dem Eingang eines anderen Elements verbunden ist, werden eventuell darunterliegende Elemente ignoriert.

Mit den Lampen-Elementen lässt sich die Helligkeit von angeschlossnenen Leuchtdioden einstellen. Das Symbol gibt es 6 mal mit den Buchstaben **A** bis **F**. Dabei bestimmt der Buchstabe, welcher Anschluss gemeint ist. Die Helligkeit lässt sich im Bereich 0. . . 20 einstellen, wobei der Wert 0 für Lampe aus und der Wert 20 für maximale Helligkeit steht.

Mit den Motor-Elementen lässt sich die Geschwindigkeit von angeschlossnenen Getriebemotoren oder die Position von angeschlossenen Servos einstellen. Das Symbol gibt es 2 mal und zwar in den Farben Rot und Blau. Das entspricht dem roten bzw. blauen Servoanschluss. Der Wertebereich geht von 0 bis 20. Der Wert 10 bedeutet die Mittelstellung. Das heisst, daß bei diesem Wert Getriebemotoren stillstehen und sich Servos in der Mittelstellung befinden sollten. Ist dies nicht der Fall, müssen die Servoausgänge noch kalibriert werden. Wie, steht im ersten Kapitel dieses Büchleins. Ausgehend von der Mittelstellung drehen oder bewegen sich die Motoren oder Servos nach links oder rechts.

Dieses Element steuert den eingebauten Lautsprecher an. Der Wert entspricht der Tonhöhe, wobei 0 den tiefsten und 20 den höchsten Ton ergibt. Zwischen den Tonhöhen entspricht eine Sufe einem Halbtonschritt, die genaue Tonhöhe ist jedoch vom RC-Oszillator im Controller abhängig und damit nicht besonders genau. Ein einmal eingestellter Ton wird bis zum Abschalten oder Ändern der Tonhöhe kontinuierlich ausgegeben, die Lautstärke kann nicht eingestellt werden.

Dieses Element schaltet den eingebauten Lautsprecher ab. Außer dem Programmende ist das die einzige Möglichkeit, die Ausgabe eines Tones zu beenden. Ist der lautsprecher nicht aktiv, hat das Element keine Wirkung.

3.2.2 Eingabefunktionen

Eigentlich gibt es nur eine Eingabefunktion, und das ist der Schalter. Später im Abschnitt über die Variable werden wir sehen, dass das nicht ganz stimmt…

Wir können an alle 6 Anschlüsse Schalter anschließen, natürlich nur so lange wir den Anschluss nicht für Ausgabeelemente benötigen. Beim Schalter gibt es die Zustaände „AUS” **(0)** und „EIN” **(0)** die auch als Eigenschaftswert einem Schalter-Element zugewiesen werden können. Hier kommen jetzt wieder WAHR und FALSCH ins Spiel. Die interne Bedingung ist immer WAHR, wenn der Zustand des Schalters dem eingestellten Wert entspricht.Ein Schalter-Element mit dem Wert **1** ist dann WAHR, wenn der Schalter am zugehörigen Anschluss betätigt wird, bei einem Schalter-Element mit dem Wert **0** dann, wenn der Schalter nicht betätigt wird.
Benötigt man eine Programmverweigung abhängig davon ob ein Schalter betätigt ist oder nicht, reicht es aus zwei Schalterelemente mit den Werten **0** und **1** untereinander zu platzieren.

3.2.3 Zeit

Bei den „normalen" Zeitfunktionen ist die interne Bedingung immer WAHR. Das heißt, wenn der Ausgang mit dem Eingang eines anderen Elements verbunden ist, werden eventuell darunterliegende Elemente ignoriert. Allerdings blockieren die „normalen" Zeit-Elemente den Programmablauf so lange, bis die Zeit abgelaufen ist.

Das Stoppuhr-Element fügt in den Programmablauf eine Wartezeit ein. Einstellbar ist ein Bereich von 1 bis 20 Zehntelsekunden. Mittels der Stop-Taste kann das laufende Programm auch während einer aktiven Wartezeit abgebrochen werden.

Reicht die Wartezeit von maximal 1 Sekunde nicht aus, so kann mit dem Sanduhr-Element eine Wartezeit von 1 bis 20 Sekunden eingestellt werden. Mittels der Stop-Taste kann das laufende Programm auch während einer aktiven Wartezeit abgebrochen werden. Noch längere Wartezeiten können nur mittels Schleifen und Unterprogrammen realisiert werden.

Bei den unterbrechbaren Zeitfunktionen ist die interne Bedingung erst dann WAHR, wenn die Zeit abgelaufen ist. Damit kann z.B. mit einem darunterliegenden Tasterelement die laufende Zeit unterbrochen werden.

Das unterbrechbare Stoppuhr-Element entspricht dem normalen, lediglich die interne Bedingung funktioniert anders. Die interne Bedingung ist solange FALSCH, bis die Zeit abgelaufen ist. Einstellbar ist ein Bereich von 1 bis 20 Zehntelsekunden. Genau wie bei der normalen Variante ist auch ein Abbruch über die Stop-Taste möglich.

Das unterbrechbare Sanduhr-Element entspricht dem normalen, lediglich die interne Bedingung funktioniert anders. Die interne Bedingung ist solange FALSCH, bis die Zeit abgelaufen ist. Einstellbar ist ein Bereich von 1 bis 20 Sekunden. Wie bei der normalen Variante ist auch ein Abbruch über die Stop-Taste während des Programmlaufes möglich.

HINWEIS: Die unterbrechbaren Zeitelemente sollten immer als oberstes Element eines Vertikalen Stapels verwendet werden, da ansonsten beim Aufruf die Zeit nicht neu gestartet wird.

3.2.4 Die Variable

Zuerst stellt sich natürlich die Frage „Was ist eine Variable?”. Nun gut, im Namen steht etwas von variabel, ein anderes Wort für „veränderlich”. Und genau das ist es, eine Variable ist eine Art Behälter für Werte, die sich ändern können. In LBlocks wird eine Variable durch ein Saftglas mit lecker Orangensaft dargestellt. Wie im richtigen Leben kann dieses Glas leer, voll oder teilweise gefüllt sein. Unsere Variable kann dabei die Werte von 0 (leer) bis 20 (voll) annehmen.

Mit diesem Element lässt sich der momentane Wert der Variable auf der Anzeige des LBlocks-Controllers anzeigen. Das ist besonders hilfreich für Experimente, bei denen zum Beispiel eine LED auf einen Fotowiderstand ausgerichtet werden muss. Aber auch wenn notwendig ist Programmzustände anzuzeigen, kann dieses Element sinvoll genutzt werden. Neben diesem Element befindet sich noch ein weiteres mit durchgestichener Anzeige. Mit diesem kann die Segmentanzeige auch wieder abgeschaltet werden.

Das zweite Element dient dazu, der Variable einen festen Wert zuzuweisen. Das **X** solldabei den Wert der Variable darstellen. Einstellbar sind Werte von 0 bis 20. Das mag jetzt erstmal wenig sinnvoll erscheinen, denn dazu bräuchte man ja keine Variable. Aber im Zusammenspiel mit anderen Elementen ist es damit beispielsweise möglich, bestimmte Dinge in einer festgelegten Anzahl zu tun. Also zum Beispiel eine Lampe 9 mal blinken zu lassen ohne dafür 9 x Lampe Ein, 9 x Lampe aus und 17 x Warten benutzen zu müssen. Da die Anzahl der Elemente pro Programm auf 32 begrenzt ist ließe sich dieses Beispiel ohne Schleifen überhaupt nicht realisieren.

Will man den Wert der Variablen um einen festen Wert erhöhen oder erniedrigen, kann dieses Element verwendet werden. Der einstellbare Bereich geht von -5 bis +5. Dabei kann der Wert nicht kleiner als 0 und nicht größer als 20 werden. Dieses Verhalten nennt man auch Sättigung. Ein Beispiel für dieses Element sind zum Beispiel Programmteile, die in einer bestimmten Anzahl (Schleifen) abgearbeitet werden. Zu Beginn wird die Variable auf einen festen Wert gesetzt und nach jedem Programmdurchlauf um 1 verringert. Ist sie dann immer noch größer als 0, dann wird zum Anfang des Programmteils zurückgesprungen.

Das nächste Element enthält eine Ausgabefunktion, Genaugenommen gibt es sechs davon, für jeden Anschluss eins. Damit lässt sich die Helligkeit von angeschlossnenen Leuchtdioden abhängig vom Variablenwert einstellen.

Auch dieses Element ist eine Ausgabefunktion. Damit lässt sich einer der beiden Motoren/Servos in Abhängigkeit vom Inhalt der Variable steuern. Für jeden der beiden Motoren-Anschlüsse (rot und blau) gibt es ein eigenes Element.

Mit diesem Element lässt sich der Inhalt der Variable invertieren. Genaugenommen wird der Inhalt der Variable von 20 abgezogen. Das Heisst auch, wenn die Variable den Wert 5 hat, ändert sich dieser duch das Element nicht. Ein Beispiel dafür ist die Steuerung eines Fahrzeuges mit zwei Motoren. Wenn sich jetzt einer links und einer rechts befindet, müssen sie entgegengesetzt angesteuert werden, damit das Fahrzeug geradeausfährt. Hier kommt jetzt das Invertierungselement in Zusammenarbeit mit dem gerade vorher beschriebenen Element zum Einsatz.
Die Reihenfolge wäre dann:

1. Setzen der Variable für die Geschwindigkeit
2. Motor 1 (z.B. rot) durch die Variable steuern
3. Variable invertieren
4. Motor 2 durch die Variable steuern
5. Variable invertieren

Das zweite Invertieren der Variable dient dazu, dass der alte Wert wiederhergestellt wird. Das Element kann weggelassen werden, wenn der Wert der Variable danach nicht mehr benötigt wird.

Wie es die Würfel im Element-Symbol schon andeuten, mit diesem Element wird die Variable auf einen zufälligen Wert gesetzt, sozusagen ausgewürfelt. Dabei kann eingestellt werden, was der maximale Wert ist, der dabei eingestellt werden kann. Der kleinste Wert ist immer Null. Um jetzt einen Würfel mit den Zahlen 1 bis 6 nachzubilden, muss man als größten Wert **5** wählen und dann 1 dazuzählen.

Mit diesem „Messgerät”-Symbol wird der Variable der am entsprechenden Eingang anliegende Spannungswert zugewiesen. Da die Anschlüsse den Minuspol als zweiten Anschlusspunkt haben, wird der gemessene Spannungswert invertiert.

Dieses Element führt einen Vergleich durch. Ist der Wert der Variable gleich dem eingestellten Wert am Element, dann ist das Ergebnis des Elements **WAHR**, in allen anderen Fällen ist es **FALSCH**. So lassen sich Programmverzweigungen abhängig vom Wert der Variable realisieren.

Dieses Element führt einen Vergleich durch. Ist der Wert der Variable **un**gleich dem eingestellten Wert am Element, dann ist das Ergebnis des Elements **WAHR**, in allen anderen Fällen ist es **FALSCH**. Setzt man z.B. das Ungleich- und das zuletzt beschriebene Gleich-Element untereinander, lassen sich einfache Programmverzweigungen realisieren

Dieses Element führt einen Vergleich durch. Ist der Wert der Variable kleiner als der eingestellten Wert am Element, dann ist das Ergebnis des Elements **WAHR**, in allen anderen Fällen ist es **FALSCH**. So lassen sich Programmverzweigungen abhängig vom Wert der Variable realisieren.

Dieses Element führt einen Vergleich durch. Ist der Wert der Variable größer als der eingestellten Wert am Element, dann ist das Ergebnis des Elements **WAHR**, in allen anderen Fällen ist es **FALSCH**. So lassen sich Programmverzweigungen abhängig vom Wert der Variable realisieren.

3.2.5 Hilfsvariablen

Manchmal kann es sein, dass eine Variable nicht ausreicht. Dafür gibt es zwei Hilfsvariablen. Diese sind in den Symbolen mit einem rot gefüllten und mit einem grün gefüllten Glas dargestellt.

Dieses Element tauscht den Inhalt (Wert) der Variable mit einer der beiden Hilfsvariablen. Damit kann der Variablenwert einer lokalen Variablen auch im Hauptprogramm verfügbar gemacht werden.

Addiert den Wert einer der beiden Hilfsvariablen zum Wert der Hauptvariablen. Dabei wird das Ergebnis auf den Wert 20 begrenzt, 17 + 8 ergeben demnach 20, was eigentlich mathematisch nicht korrekt ist. Da aber der Wertebereich aller Elemente und Variablen nur von 0 bis 20 geht, kommt es zu dieser Einschränkung.

Wie schon beim Tausch-Element angedeutet, kann dieses benutzt werden um den Variablenwert am Ende eines Unterprogrammes zu „retten". Dazu tauscht man die Werte von Variable und Hilfsvariable vor dem Ausgang ins Hauptprogramm. Zu diesem Zeitpunkt erhält dann die Hilfsvariable den Wert der lokalen Hauptvariable. Im Hauptprogramm wird dann nach dem Unterprogrammaufruf wieder das Tausch-Element aufgerufen. Danach enthält die Hauptvariable den Wert der lokalen Variable im Unterprogramm und die Hilfvariable den Wert der Hauptvariablen vor dem Unterprogrammaufruf.

Benötigt man anstelle der Addition eine Subtraktion, so kann das folgendermaßen erreicht werden:

1. Variable invertieren
2. Hilfsvariable addieren
3. Variable invertieren

3.2.6 Unterprogramme

Sicherlich ist schon die Leiste am unteren Bildrand aufgefallen, in der neben dem Hauptprogramm auch noch vier verschiedene Boxen ausgewählt werden können. Damit ist es möglich, Programme aufzuteilen wenn sie z.B. zu groß für ein „Arbeitsblatt“ werden, also nicht mehr in das Hauptprogramm passen.

Mit den Box-Elementen lassen sich die vier verschiedenen Unterprogrammboxen in den Programmablauf integrieren. Auch wenn im Symbol nur links und rechts Verbindungen dargestellt sind, lassen sich Unterprogrammboxen genauso wie andere Elemente untereinander stapeln. Das Ergbnis eines Unterprogramms ist immer **WAHR**, der Inhalt der Variable wird durch Unterprogramme nicht beeinflusst.

Dieses Element existiert in allen Unterprogrammen und markiert den Eingang vom Hauptprogrogramm her. Das Element kann verschoben, aber nicht gelöscht werden. Unterprogramme haben ihre eigenen Variablen, diese nennt man auch **lokal**, da sie nur im Bereich des Unterprogramms gelten. Beim Eintritt in das Unterprogramm wird der Inhalt der Hauptvariablen in die lokale Variable kopiert.

Dieses Element existiert in allen Unterprogrammen und markiert den Ausgang zurück zum Hauptprogrogramm. Das Element kann verschoben, aber nicht gelöscht werden. Der Inhalt der lokalen Variable geht dabei verloren. Soll dieser in die Hauptvariable übernommen werden, so ist das über eine Hilfsvariable möglich.

Kapitel 4

Experimente

4.1 Vorwort

Dieses Kapitel soll nach und nach Anleitungen zu verschiedenen Experimenten erhalten. Allerdings wird es einige Zeit brauchen, bis alle bereits realisierten Experimente auch dokumentiert worden sind. Anregungen nehme ich natürlich gerne entgegen. Das Gleiche gilt für Hinweise auf Fehler, die sich doch hier und da einschleichen können.

4.2 Licht an, Licht aus

4.2.1 Wir lassen eine Lampe blinken

Das einfachste aller Programme ist meistens ein „Hallo Welt", bei dem das Programm sich mit dem genannten Text meldet. Bei Steuerungen die keine Textausgabe besitzen lässt man meist einfach eine LED blinken. Das wollen wir auch hier machen und können dazu das Aufbaubeipiel aus der Einleitung benutzen:

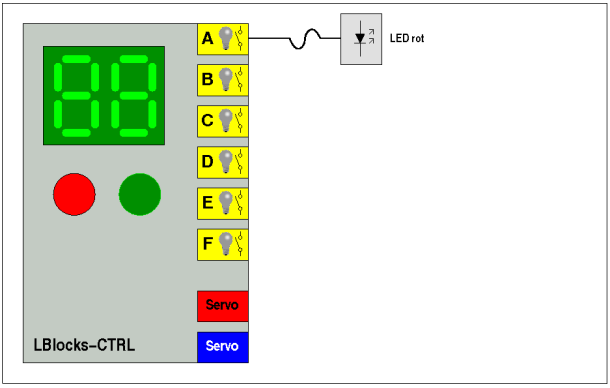


Abbildung 4.1: Versuchsaufbau für die Blinkschaltung

Der Ablauf des Programmes ist folgender:

1. LED einschalten
2. etwas warten
3. LED ausschalten
4. etwas warten
5. wieder bei 1. beginnen

Das Programm sieht schon etwas komplizierter aus, aber das ist nur auf den ersten Blick so.

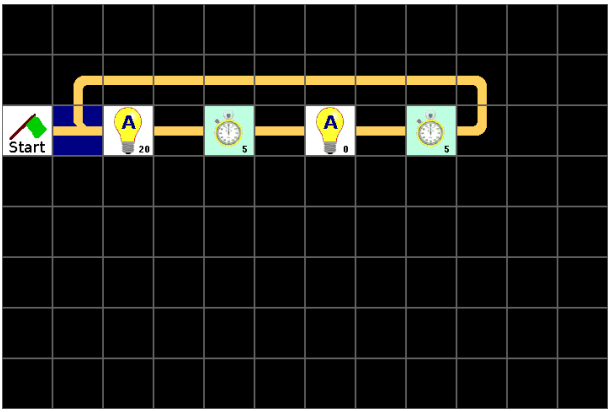


Abbildung 4.2: Beispielprogramm **xs-blink1**

Das oben dargestellte Programm ist aber nur eine Möglichkeit, wir können zum Beispiel die Verbindungen auch zum Teil weglassen und die Elemente weiter zusammenrücken:

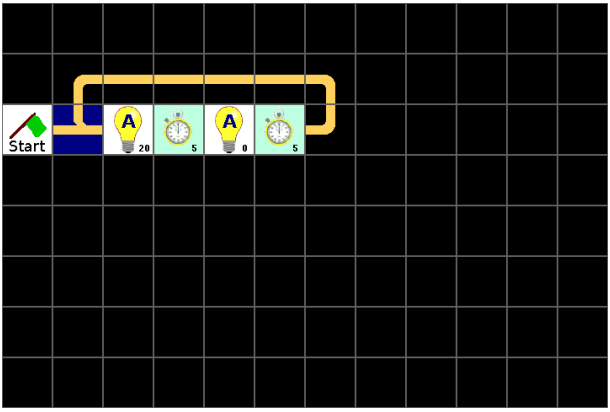


Abbildung 4.3: Beispielprogramm **xs-blink2**

Und es gibt noch eine weitere Möglichkeit. Im Kapitel bei der Auflistung der Elemente stand, dass untereinander angeordnete Elemente zyklisch ausgeführt werden, wenn es keine Möglichkeit gibt, den „Staffelstab” nach rechts weiterzugeben.

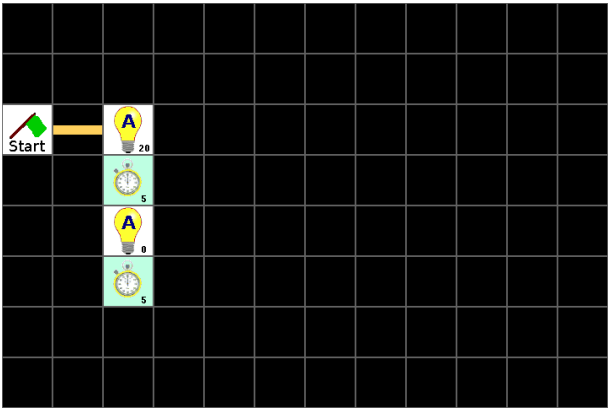


Abbildung 4.4: Beispielprogramm **xs-blink3**

Jetzt könnte man als Aufgabe die Zeiten so verstellen, dass die Lampe immer nur kurz aufblitzt.

4.2.2 Geht es auch von Hand?

Jetzt wollen wir aber nicht nur zusehen, sondern die Lampe (LED) selbst steuern. Dafür brauchen wir einen Taster, den wir an den Anschluß **B** anschließen.

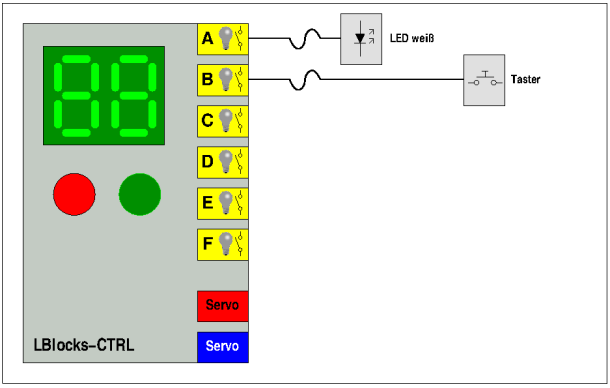


Abbildung 4.5: Die Anschlüsse für die Handsteuerung (1)

Nur mit elektrischen oder elektronischen Bauelementen aufgebaut, wäre wohl eine der einfachsten Schaltungen der Welt. Vielleicht geht es ja so:

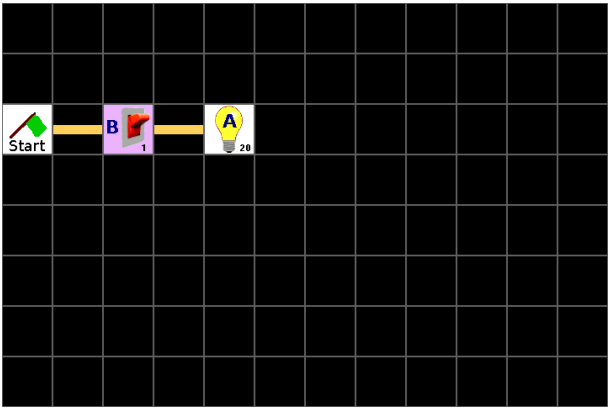


Abbildung 4.6: Beispielprogramm **xs-taster1**

Hmmm. Irgendwie geht das nicht. Wenn wir den Taster drücken zeigt die Anzeige am Controller an, dass das Programm beendet ist und die LED ist aus. Vielleich kann man sehen, dass sie ganz kurz geleuchtet hat. Also müssen wir die Sache anders angehen und uns aber überlegen, was denn eigentlich passieren soll:

1. Wenn der Taster aus ist, dann die LED ausschalten
2. Wenn der Taster an ist, dann die LED einschalten
3. wieder bei 1. beginnen

Und genau das muss jetzt als Programm für den LBlocks-Controller umgesetzt werden. Als Programm könnte das Ganze dann so aussehen:

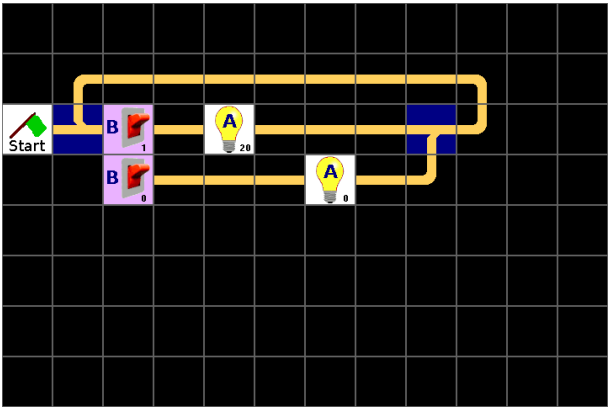


Abbildung 4.7: Beispielprogramm **xs-taster2**

Jetzt können wir die LED mit dem Taster ein- und ausschalten. Und wenn die Werte der beiden LED-Elemente vertauscht werden, funktioniert alles umgekehrt: Solange der Taster gedrückt wird, geht die LED aus.

4.2.3 Und jetzt mit 2 Tastern

Als nächstes wollen wir 2 Taster anschließen, und zwar einen zum Einschalten und einen zum Ausschalten. Den zweiten taster schließen wir dazu an Anschluß **C** an.

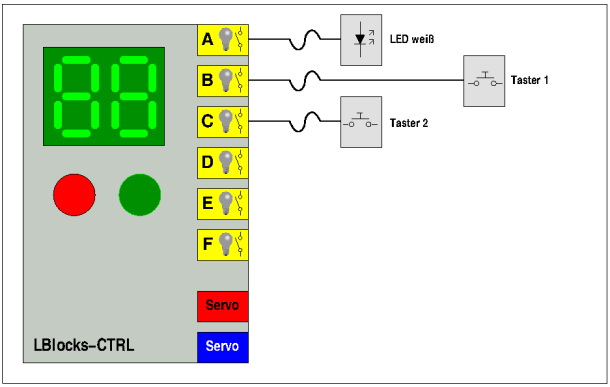


Abbildung 4.8: Die Anschlüsse für die Handsteuerung (1)

Das Passende Programm sieht dann so aus:

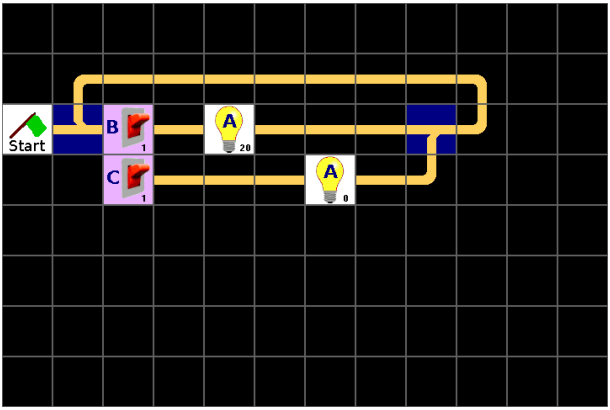


Abbildung 4.9: Beispielprogramm **xs-taster3**

4.2.4 Ein Treppenlicht-Automat

Wer kannt das nicht, man drückt auf den Taster im Treppenaufgang und das Licht bleibt für eine Zeitlang an. Auch das können wir mit unserem LBlocks-Controller realisieren. Dazu brauchen wir wieder eine LED und einen Taster, die wie folgt angeschlossen werden:

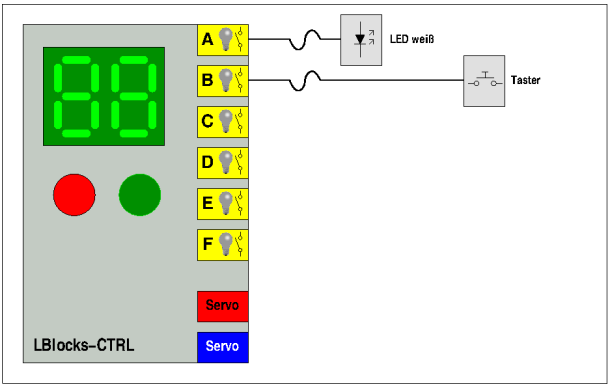


Abbildung 4.10: Der Treppenlichtautomat

Das passende Programm sieht dann so aus:

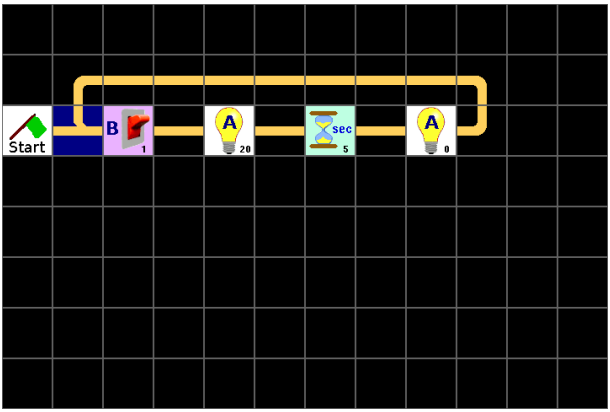


Abbildung 4.11: Beispielprogramm **xs-treppe1**

Besonders gut funktioniert das leider nicht, denn man kann die Zeit nicht verlängern, indem man während der Ein-Zeit den Taster erneut drückt. Hier kommen jetzt die unterbrechbaren Zeiten ins Spiel. Wird nämlich jetzt der Taster gedrückt während die Zeit läuft, wird das Programm an der Stelle fortgesetzt, an der die Lampe eingeschaltet wird und die Wartezeit beginnt.

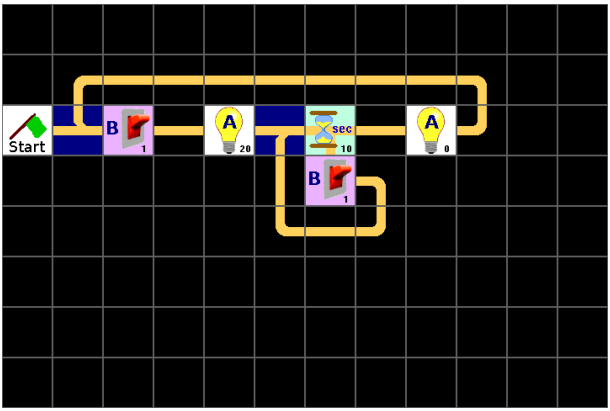


Abbildung 4.12: Beispielprogramm **xs-treppe2**

Das erneuete Starten der Zeit wird über den unteren „Zweig" des Programmes realisiert. Solange die Zeit noch nicht abgelaufen und der Taster nicht gedückt ist, wird der vertikale Stapel zyklisch abgearbeitet. Erst wenn eine der inneren Bedingungen WAHR ist (Zeit abgelaufen oder Taster gedrückt) wird der Stapel verlassen.
Ist zuerst die Zeit abgelaufen, wird die LED ausgeschaltet und das Programm wartet wieder, dass der Taster betätigt wird. Wird zuerst der Taster betätigt, wird das Programm als nächstes die Zeit neu starten.

Zum Abschluss noch ein bisschen Komfort. Es wäre doch schön, wenn das Licht nicht abrupt abschaltet sondern erstmal etwas dunkler wird. Damit hätte man dann noch Gelegenheit, nach dem Taster zu suchen ohne gleich im Dunklen zu stehen. Auch das ist kein Problem für unseren LBlocks-Controller:

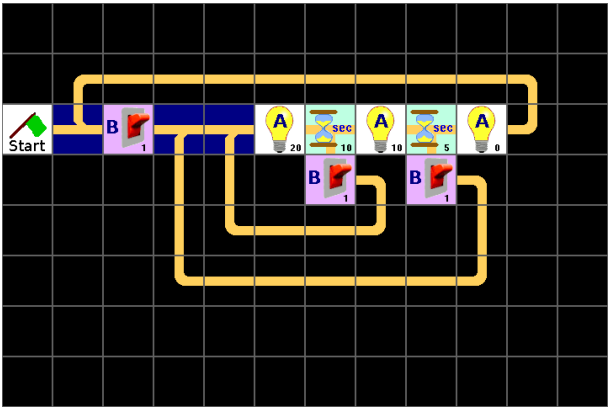


Abbildung 4.13: Beispielprogramm **xs-treppe3**

4.2.5 Eine Fußgängerampel

Jetzt noch ein anderes Experiment. Neben den Ampeln an Kreuzungen gibt ja auch solchen, bei denen Fußgänger sicher auf die andere Seite kommen sollen. Beobachtet man eine solche Ampel, wird man feststellen, dass der Ablauf immer der gleiche ist:

1. Die Fußgängerampel steht auf rot und die Autofahrer haben grün. Das ist sozusagen die Grundstellung.
2. Wenn ein Fußgänger auf den Taster drückt, passeriert erstmal nichts, manchmal leuchtet dann eine Lampe mit „Signal kommt".
3. Nach einer Weile schaltet die Autofahrerampel auf gelb und danach auf rot.
4. Jetzt bekommen die Fußgänger für eine Weile Grün.
5. Nachdem die Fußgängerampel wieder auf rot geschaltet hat, bekommen die Autofahrer wieder Grün und dasGanze geht wieder von vorne los.

Also, was brauchen wie jetzt?

- Rote, gelbe und grüne LED für die Autofahrer
- Rote und grüne LED für die Fußgänger
- Einen Taster für die Fußgänger

Insgesamt sind das 6 Dinge zum anschließen, passt also gut mit den 6 Anschlüssen an unserem LBlocks-Controller zusammen. Also schließe wir die LEDs und den Taster wie folgt an:

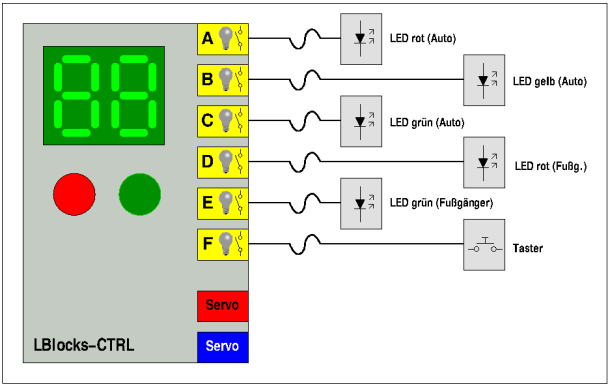


Abbildung 4.14: Die Schaltung für die Fußgängerampel

Im Programm versuchen wir, die oben beschriebenen Zustände als vertikale Stapel nachzubilden. Also die LEDs **A** bis **E** und darunter dann die Wartezeit bis zum nächsten Stapel oder auch den Taster.

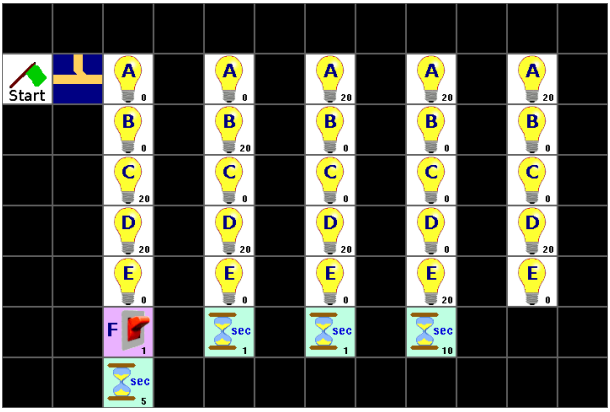


Abbildung 4.15: Beispielprogramm **xs-ampel1**

Jetzt fehlt noch ein Element und zwar eine Zeit unter dem Stapel ganz rechts, aber die lässt sich irgendwie nicht aus der Liste dorthin ziehen. Leider gibt es eine Einschränkung. Und die legt fest, dass nur maximal 32 Elemente auf einer Seite sein dürfen. Das Gleiche gilt für maximal 32 Verbindungen, aber das ist nur ein theoretischer Wert, da 32 Elemente nie mehr als 32 Verbindungen haben können. Diese Einschränkungen haben sich bei der Entwicklung des LBlocks-Programmes ergeben und können auch nicht so einfach aufgehoben oder erweitert werden. Aber schauen wir uns unser Programm mal genau an, ob sich nicht ein paar Elemente einsparen lassen. Die Werte mancher Lampen ändern sich gar nicht, wenn zum nächsten Stapel gewechselt wird.

Und jetzt? Die erste Lösung ist, nur die Lampen-Elemente zu benutzen, bei denen sich auch wirklich etwas ändert. Das dazugehörige Programm bleibt noch recht übersichtlich, wenn man gleiche Elemente nebeneinander setzt:

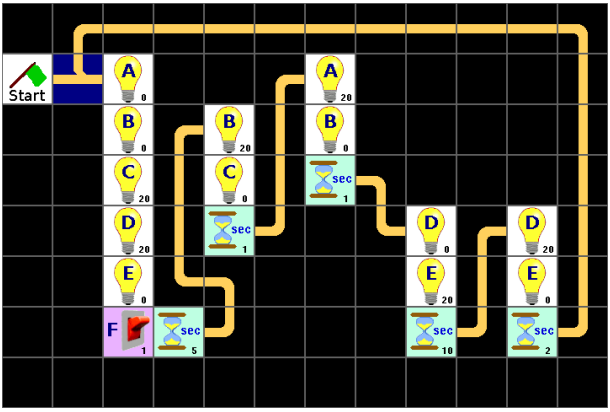


Abbildung 4.16: Beispielprogramm **xs-ampel2**

Eine weitere Möglichkeit wäre, ein Unterprogramm zu benutzen und zum Beispiel die letzten beiden Stapel dort auszuführen. Auf der folgenden Seite sind das dazugehörige Hauptprogramm und die rote Box dargestellt.

Zuerst kommt das Hauptprogramm. Anstelle der beiden rechten Stapel wird jetzt das Unterprogramm in der roten Box ausgeführt.

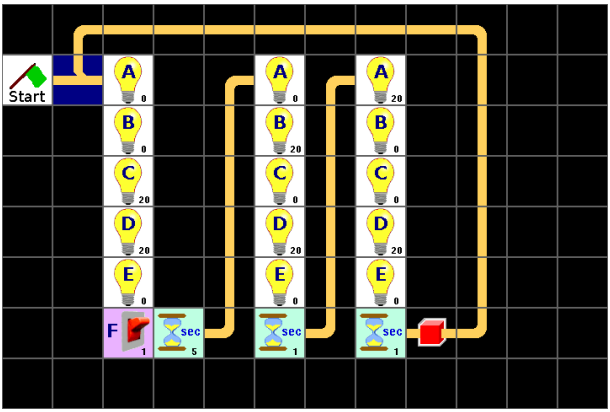


Abbildung 4.17: Beispielprogramm **xs-ampel3** (Hauptprogramm)

Und hier das Unterprogramm in der roten Box. Bei den Unterprogrammen ist es wichtig, daß es eine Verbindung (auch über andere Elemente) vom Eingangselement zum Ausgangselemet gibt.

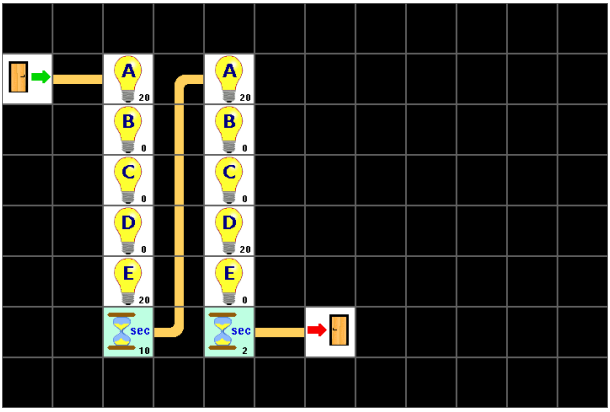


Abbildung 4.18: Beispielprogramm **xs-ampel3** (rote Box)

4.3 Der Fotowiderstand

4.3.1 Wir testen den Fotowiderstand

Das Prinzip einer sogenannten Lichtschranke ist, dass eine Aktion bei Unterbrechung eines Lichtstrahls ausgelöst wird. Meist wird dazu unsichtbares Infrarotlicht verwendet, wir nutzen aber das Licht einer weißen LED. Als Lichtempfänger dient ein Fotowiderstand. Das ist ein elektronisches Bauelement welches seinen Widerstand abhängig vom einfallenden Licht ändert. Zunächst wollen wir mal sehen, was der Fotowiderstand an unserem Controller bewirkt. Dazu schließen wir an Anschluß **A** eine weiße LED und an Anschluß **B** den Fotowiderstand an.

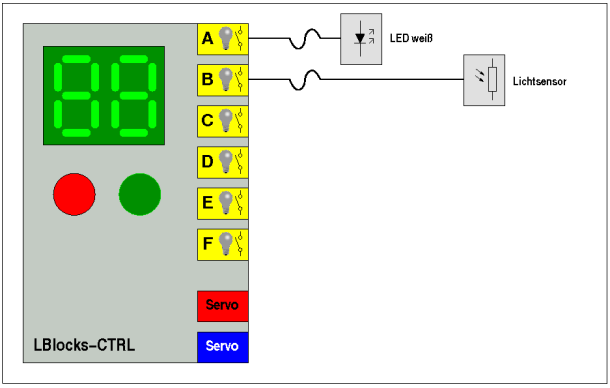


Abbildung 4.19: Messschaltung für den Fotowiderstand

Um zu sehen, was passiert, wollen wir den Messwert auf der Segment-Anzeige des Controllers anzeigen lassen. Das Programm dazu sieht dann so aus:

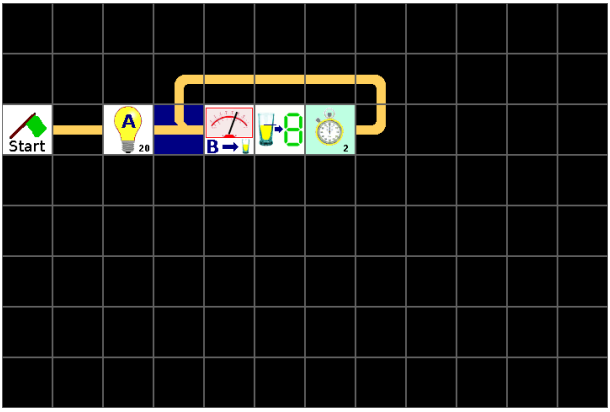
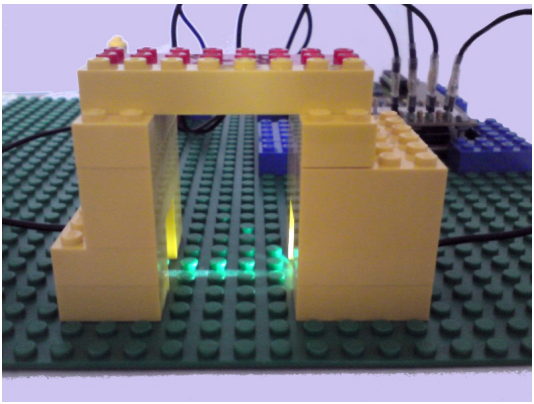


Abbildung 4.20: Beispielprogramm **xs_FOTOMESS**

Zuerst wird die LED am Anschluss **A** eingeschaltet, dann geht es in eine Schleife die bis zum Ausschalten oder Drücken der Stop-Taste abgearbeitet wird. In jedem Durchlauf wird der Analogwert von Anschluss **B** in die Variable eingelesen (das Elemet mit dem Messgerät-Symbol), der Wert der Variable auf der Anzeige des Controllers angezeigt und danach 0,2 Sekunden gewartet.
Vielleicht stellt sich jetzt die Frage, wozu die Wartezeit gut ist. Aber während dier Widerstand unseres Fotowiderstands eine Größe ist, die beliebig viele Zuwischenwerte annehmen kann, zeigt unsere Anzeige nur Stufen an. Und wenn jetzt der Wert genau zwischen zwei Stufen liegt, kann man kaum etwas erkennen. Das kann man auch einfach ausprobieren, indem das Warte-Element (Stoppuhr) entfernt wird.
Um das Programm zu testen, stellen wir LED und Fotowiderstand gegenüber, starten das Programm und schauen was passiert wenn Gegenstände zwischen die Leuchtdiode und den Fotowiderstand gebracht werden. Wenn alles funktioniert leuchtet die LED und die Anzeige auf dem Controller-Mudul ändert sich hin zu kleineren Werten, wenn wir den Fotowiderstand abdecken. In hellen Umgebungen kann es sinnvoll sein, vor den Fotowiderstand eine Art kleinen Tunnel zur Abschirmung von Fremdlicht zu bauen. Jetzt können wir das Ganze noch in eine Tür oder einen Durchgang einbauen, vielleicht so wie in der Abbildung auf der nächsten Seite.

4.4 Achtung Einbrecher!

4.4.1 Eine einfache Lichtschranke



Wenn wir jetzt das Programm starten, sehen wir auf der Anzeige des LBlocks-Controllers, wenn der Lichtstrahl von der LED zum Fotowiderstand unterbrochen wird. Da man dazu ständig auf die Anzeige schauen muss, könnte man auch gleich den Durchgang bewachen. Also brauchen wir ein Signal welches und die Unterbrechung des Lichtstrahls meldet und welches danach wieder von Hand zurückgestellt werden muss. Dazu erweitern wir unsere Schaltung auf die folgende Weise:

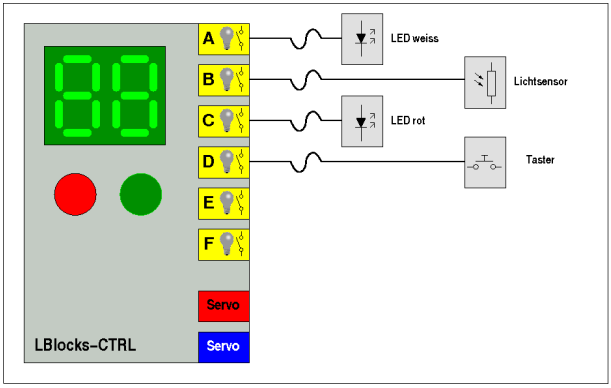


Abbildung 4.21: Die Schaltung der Lichtschranke

Das neue Programm kann dann folgendermaßen aussehen, wobei hier noch eine Tonausgabe mit hinzugekommen ist.

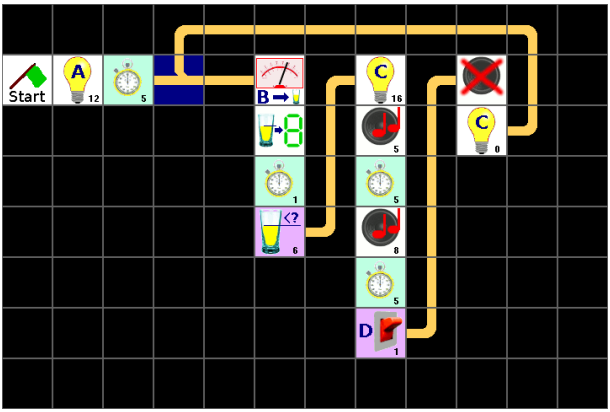
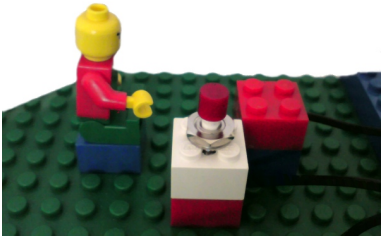


Abbildung 4.22: Beispielprogramm **xs_lschranke1**

Das gesamte Programm ist in 3 vertikale Stapel aufgeteilt. Der erste Stapel enthält wieder unsere Lichtmessung, wobei jetzt noch eine Abfrage hinzugekommen ist, ob der Wert der Variable unter 6 gesunken ist. Wenn ja, wird der zweite Stapel ausgeführt. Hier werden zuerst die rote LED eingeschaltet und danach zwei unterschiedlich hohe Töne im Wechsel ausgegeben (Sirene). Erst mit der Betätigung des Tasters an Anschluss **D** wird der Alarm wieder abgeschaltet (3.Stapel) und das Programm beginnt wieder mit dem ersten Stapel. Dabei muss der Taster für minestens eine Sekunde gedrückt bleiben.



Jetzt kann der Posten ruhig in seiner Zentrale sitzen und wenn der Lichtstrahl unterbrochen wird, dann geht die rote Lampe an und die Sirene ertönt. Erst wenn der Abstellknopf lange genug gedrückt wurde, ist der Spuk wieder vorbei.

4.4.2 Wir verbessern die Lichtschranke

Unsere Lichtschranke funktioniert ja schon ganz gut, **ABER** man kann sie mit jeder Taschenlampe austricksen. Wenn damit der Fotowiderstand beleuchtet wird ist es kein Problem den Lichtstrahl zur LED zu unterbrechen ohne dass Alarm ausgelöst wird. Eine Möglichkeit wäre, mit einem zusätzlichen Vergleich auf einen Maximalwert die erlaubte Lichtstärke am Fotowiderstand einzugrenzen, aber das ist auch nicht besonders sicher und kann Probleme bei unterschiedlicher Umgebungshelligkeit bringen. Eine andere Möglichkeit ist, das Licht abwechselnd ein- und auszuschalten und dabei jeweils einen Vergleich auf zu geringen oder zu hohe Helligkeit zu machen. Um diese Lichtschranke auszutricksen, müsste man mit der Taschenlampe in exakt dem gleichen Rhythmus blinken. Und das wird ganz schön schwierig, besonders wenn das Blinken sehr schnell ist. Das Ganze hat auch einen Namen und nennt sich „Wechsellichtschranke".
Das neue Programm ist schon recht komliziert und lässt kaum noch Platz für Erweiterungen. Abhilfe könnte aber eine Verlagerung in Unterprogramme bringen.

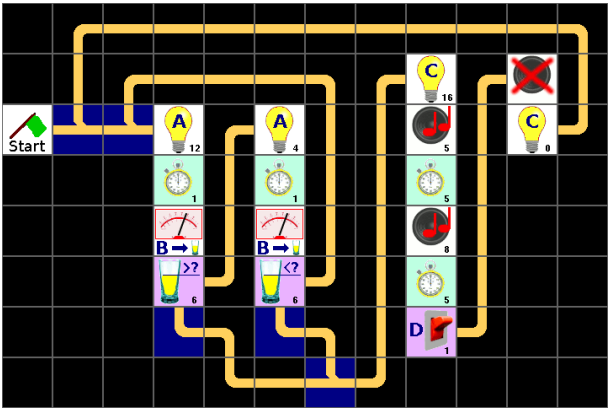


Abbildung 4.23: Beispielprogramm **xs_lschranke2**

4.4.3 Ein Dämmerungsschalter

Wie der Name schon sagt, schaltet ein Dämmerungsschalter in der Dämmerung. Zum Beispiel kann er die Staßenbeleuchtung abends ein- und morgens wieder schalten. Gut, das könnte man auch mit einer Schaltuhr (einer Uhr mit Schaltkontakten, die zu festgelegten Zeiten ein- oder ausschalten) relisieren. Aber im Sommer kann die Straßenbeleuchtung früher ausgeschaltet und abends auch später eingeschaltet werden, da es dann länger hell ist als im Winter. Ein Dämmerungsschalter erledigt diese Anpassung automatisch.
Wieder schließen wir an Anschluß **A** eine weiße LED und an Anschluß **B** den Fotowiderstand an.

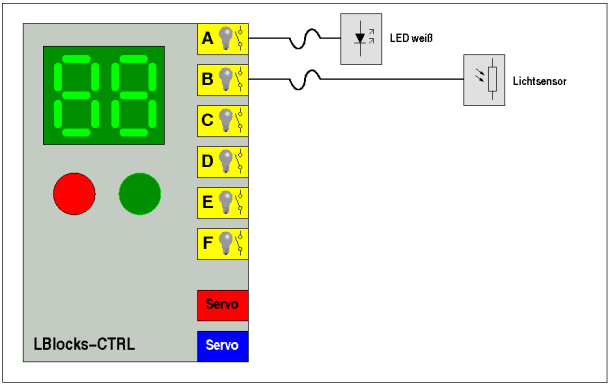


Abbildung 4.24: Schaltung für den Dämmerungsschalter

Und hier noch dazu das passende Programm:

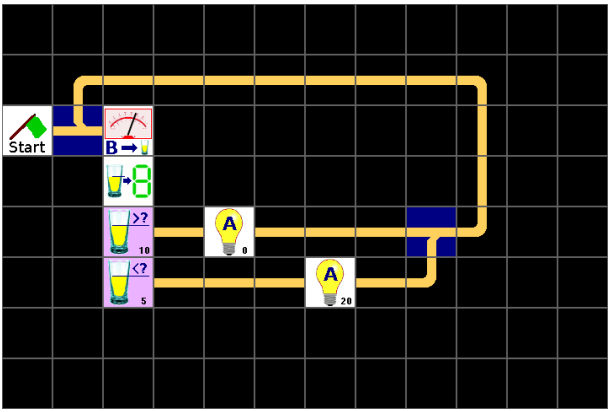


Abbildung 4.25: Beispielprogramm **xs_daemm1**

In jedem Zyklus wird der Wert des Fotowiderstandes gemessen, in die Variable eingelesen und mit 2 Werten verglichen. Ist der Wert kleiner als 6 (also dunkel), wird die LED eingeschaltet. Ist der Wert größer als 10 (hell) wird sie wieder ausgeschaltet. Aber warum macht man die beiden Werte soweit auseinander? Damit verhindert man ganz einfach, dass schon kleinste Helligkeitsschwankungen zum Umschalten führen. Wenn sich die Helligkeit im Bereich der eingestellten Schaltschwelle befindet, könnte sonst schon ein vorbeifliegender Vogel dafür sorgen dass das Licht kurz angeht.

4.4.4 Ein Autoblitz für die Kamera

Moderne Kameras schalten bei Dunkelheit automatisch den Blitz ein. Da wir jetzt wissen wie man den Fotowiderstand einsetzen kann, sollte das eigentlich einfach zu realisieren sein. Da wir jetzt keine Kamera haben, benutzen wir eine LED als „Verschluss". Das Wort Verschluss stammt noch aus der Zeit, als es noch keine Digitalkameras gab. Da musste nämlich die Objektivöffnung die meiste Zeit verschlossen sein, damit der Film nicht schon vorher belichtet wurde. Nur für einen ganz kurzen Moment, die Belichtungszeit, wurde dieser Verschluss dann geöffnet. Bei Digitalkameras ist heute der Verschluss oft elektronisch realisiert.
Aber zurück zum Thema. Also wir werden eine grüne LED an Anschluss **D** als Verschluss-Anzeige nutzen und eine weiße LED an Anschluss **C** als Blitz. Den Fotowderstand schließen wir an Anschluss **A** an und einen Taster als Auslöser an Anschluss **B**.

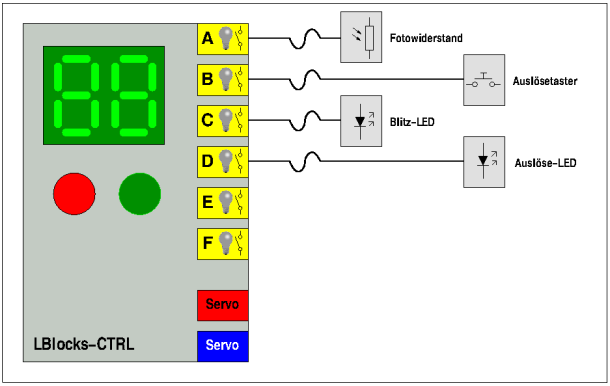


Abbildung 4.26: Die Autoblitz-Schaltung

Hier das Programm für die Blitzschaltung. Im Vergleichs-Element (in der Mitte) wird entschieden, ob zusätzlich zur Auslöser-LED auch die Blitz-LED eingeschaltet werden soll. Bevor die Programmschleife von Neuem beginnt, wird noch geaertet, bis der Auslösetaster nicht mehr gedrückt ist. Dadurch wird verhindert, dass der Auslöser periodisch ausgelöst wird.

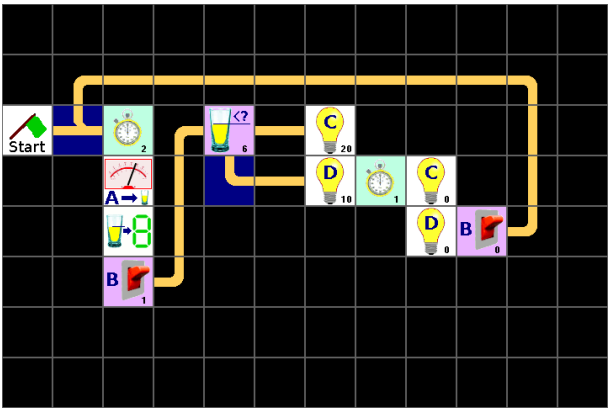


Abbildung 4.27: Beispielprogramm **xs_autoblitz1**