Erste Schritte mit Robotik

mit dem

Thymio Roboter

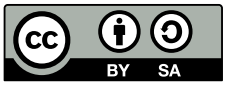
und der

Aseba/VPL Umgebung

Moti Ben-Ari und Andere

Details in der Datei `authors.txt`

Version 1.5.1 für Aseba 1.5.2



Inhaltsverzeichnis

Vorwort

Was ist ein Roboter?

Sie fahren mit ihrem Fahrrad und plötzlich sehen sie, dass die Strasse vor ihnen aufwärts geht. Sie treten kräftiger in die Pedale, um den Rädern mehr Energie zuzuführen. Dadurch werden sie auch aufwärts nicht langsamer. Nachdem sie oben angekommen sind, geht es wieder runter. Sie betätigen die Bremsen ihres Fahrrads wodurch sie nicht zu schnell werden. Wenn sie Fahrrad fahren, sind ihre Augen wie *Sensoren*, die ihre Umgebung wahrnehmen. Wenn diese Sensoren — ihre Augen — ein *Ereignis* erfassen (zum Beispiel eine Kurve), führen sie eine *Aktion* aus (zum Beispiel den Fahrradlenker nach links oder rechts zu bewegen).

In einem Auto sind Sensoren eingebaut, die *messen* was in der Umgebung passiert. Der Tachometer misst, wie schnell das Auto fährt. Wenn sie sehen, dass das Auto schneller als das Tempolimit fährt, sagen sie dem Fahrer, er fahre zu schnell. Darauf kann der Fahrer eine Aktion ausführen, nämlich die Bremse betätigen, damit das Auto langsamer wird. Die Tankanzeige misst, wie viel Benzin noch im Tank des Autos ist. Wenn sie sehen, dass die Anzeige zu tief ist, können sie der Fahrerin sagen, dass sie eine Tankstelle suchen muss. Sie kann dann eine Aktion ausführen: Sie kann den Blinker einschalten und das Steuerrad nach rechts drehen, um zur Tankstelle zu fahren.

Jeder Fahrradfahrer und Autofahrer erhält Informationen von den Sensoren, entscheidet dann welche Aktion nötig ist und führt diese Aktionen aus. Ein *Roboter* ist ein System, in welchem dieser Prozess — Informationen erhalten, Entscheidungen treffen, Aktionen ausführen — von einem Computer durchgeführt wird. Normalerweise macht der Roboter das ohne die Hilfe von Menschen.

Thymio Roboter und Aseba VPL Umgebung

Thymio ist ein kleiner Roboter, der für pädagogische Zwecke entwickelt wurde (Abbildung 1.1). Die Sensoren des Roboters messen Licht, Töne und Distanzen und detektieren, ob Knöpfe gedrückt werden oder ob an den Roboter geklopft wird. Die wichtigste Aktionen, die durchgeführt werden kann, ist die Bewegung durch zwei Räder, die mit je einem Motor angetrieben werden. Weitere Aktionen sind das Erklingen lassen von Tönen und das Ein- und Ausschalten von Lichtern in verschiedenen Farben.

In diesem Dokument sprechen wir jeweils von Thymio, obwohl natürlich die aktuelle Version Thymio II gemeint ist.

Aseba ist eine Programmierumgebung für kleine Roboter, wie Thymio. VPL ist ein Teil von Aseba und dient der visuellen Programmierung (visual programming). VPL wurde entwickelt, um Thymio auf einfache Art und Weise mit Ereignis- und Aktionsblöcken programmieren zu können.

Tutorialübersicht

Jedes Kapitel befasst sich mit einem besonderen Thema, welches aus der kurzen Einleitung hervor geht, gefolgt von der Darstellung der Ereignisse und Aktionsblöcke, die jeweils verwendet werden. Es wird empfohlen, zunächst die Kapitel im Standard-Modus zu bearbeiten, später den Fortgeschrittenen-Modus zu untersuchen oder einige Projekte zu versuchen. Das Parson-Rätsel soll versuchen, wer sein Wissen über VPL testen will. Lesen Sie Kapitel 21 sobald Sie VPL verlassen möchten, um sich der erweiterten Umgebung von Aseba Studio zu widmen. Die Anhänge umfassen Referenzmaterialien, die bei Bedarf gelesen werden sollen.

Teil I: Tutorial

Kapitel 1 und 2 geben eine grundlegende Einführung in den Roboter, die VPL-Umgebung und ihr wichtigstes Programmierkonstrukt: das Ereignis-Aktionen-Paar.

Ereignis: Tasten **Aktionsblöcke**: Farbe der oberen und unteren Lichter



Kapitel 3 bis 5 erklären die Aktionen und Algorithmen für autonome mobile Roboter. Sie stellen sicher den Kern jeder Beschäftigung mit Thymio und VPL dar.

Ereignis: Tasten, Distanzsensoren **Aktionsblöcke**: Motoren




Kapitel 6 beschreibt Funktionen des Roboters, deren Verwendung Spass macht, die aber nicht wesentlich sind: Geräusche und Erschütterungen.

Ereignis: Berührung, Klatschen **Aktionsblöcke**: Melodie, Farbe der Lichter





> **Fortgeschrittener Modus**
>
> VPL hat einen Standard-Modus, der elementare Ereignisse und Aktionen unterstützt, die für Anfänger einfach zu meistern sind. Der erweiterte Modus von VPL unterstützt weitere Ereignisse und Aktionen, die einen Anfänger überfordern würden, aber für den Fortgeschrittenen hilfreich sind. Er beginnt ab Kapitel 7.

Kapitel 7 erklärt zeit-bezogene Ereignisse. Es gibt einen Aktionsblock um einen Timer (Wecker) zu stellen; sobald der Timer abgelaufen ist, tritt das Ereignis ein.

Ereignis: Zeit abgelaufen **Aktionsblöcke**: Timer setzen


Kapitel 8 und 9 erklären Zustands-Maschinen, welche es dem Roboter erlauben, bestimmte Aktionen zu unterschiedlichen Zeitpunkten durchzuführen. Zustände können auch verwendet werden, um elementare arithmetische Operationen auszuführen wie das Zählen.

Ereignis: Zustand (bezogen auf Ereignis) **Aktionsblöcke**: ändern des Zustands


Kapitel 10 beschreibt, wie man die Beschleunigungsmesser von Thymio verwendet.

Ereignis: Beschleunigungsmesser


Teil II: Parson-Rätsel

Kapitel 11 stellt die Parson-Rätsel vor, mit denen man das Wissen über VPL testen kann.

Teil III: Projekte

Kapitel 12 bis 20 spezifizieren Projekte, die man selbständig designen und implementieren kann. Der passende Quellcode findet man im Archiv, es wird aber empfohlen, zunächst eine selbständige Lösung zu erarbeiten!

Teil IV: Übergang von visueller zu textbasierter Programmierung

Kapitel 21 verweist auf den nächsten Schritt: die Verwendung der textbasierten Programmierung mit Aseba-Studio. Diese bietet deutlich mehr Funktionen für die Entwicklung von Robotern als VPL.

Teil V: Anhänge

Anhang A enthält eine Beschreibung der Benutzerschnittstelle—die Knöpfe in der Toolbar.

Anhang B zählt die Ereignisse auf und die Aktionsblöcke des Standard- und der Fortgeschrittenen-Modus.

Anhang C enthält Leitlinien für Lehrer und Mentoren. Der erste Abschnitt zeigt, wie man das entdeckende Lernen und das Experimentieren fördern kann. Der nächste Abschnitt konzentriert sich auf gutes Programmieren. Der letzte Abschnitt listet einige Probleme auf, auf die man stossen kann und bietet Hinweise, wie diese zu überwinden sind.

Anhang D beschreibt Techniken im Umgang mit den Schiebereglern der Motoren und Sensoren.


Referenzkarten

Es mag hilfreich sein, eine oder beide VPL-Referenzkarte(n) auszudrucken. Sie sind in dem Zip-File enthalten, woher Sie diese Beschreibung haben; sie sind aber auch unter folgendem Link verfügbar: https://www.thymio.org/de:visualprogramming.

- Übersicht über die Ereignis und Aktions-Blöcke auf einer Seite.
- Ein zweiseitiges Dokument, das zu einer handlichen Karte gefaltet werden kann. Sie fasst die Benutzeroberfläche, die Ereignis- und Aktions-Blöcke zusammen und enthält einige Beispiele.

Aseba installieren

Um Aseba zu installieren (inkl. VPL), gehen sie auf folgende Seite https://www.thymio.org/de:start und klicken sie auf das Icon mit ihrem Betriebssystem (Windows, Mac OS, etc.). Folgen sie der Anleitung zum Download und zur Installation der Software. Die Aseba-Installation umfasst neben der VPL auch die Entwickler-Umgebung Aseba Studio (siehe Kapitel 21).

VPL Tutorial Versionsgeschichte

Version 1.5

- Events sind nun mit Hilfe des Aseba-Programmiersprachen-Konstrukts when implementiert anstelle von `if`. Dies bedeutet, dass Ereignisse nur dann stattfindet, wenn das Ereignis auftritt und nicht unabhängig von seinem Auftreten, wie auf der Seite 80 erklärt. Diese Änderung könnte zu unerwartetem Verhalten führen in einigen Programmen, die in diesem Tutorial beschrieben sind.
- Das dynamische Feedback während der Programm-Ausführung wurde implementiert (Seite 18).

Version 1.4

- Das graphische Design der Knöpfe wurde geändert, insbesondere um zusätzliche Funktionalität hinzuzufügen.
- In Version 1.3, hat ein *roter* Kasten im Ereignisbereich bedeutet, dass der Sensor ein Objekt wahr genommen hat, wohingegen ein *weisser* Kasten bedeutete, dass kein Objekt wahrgenommen wurde. In der Version 1.4, bedeutet ein *weisser* Kasten, dass Licht von einem Objekt reflektiert wurde, wohingegen ein *schwarzer* Kasten bedeutet dass das Objekt zu wenig Licht reflektiert hat oder dass gar kein Objekt vor dem Sensor steht (siehe Seite 22). Auch die Bodendistanzsensoren verwenden weiss und schwarz, anstelle von rot und weiss, aber das Verhalten unterscheidet sich in den beiden Versionen nicht.
- Im Fortgeschrittenen-Modus können die Schwellenwerte der Sensoren eingestellt werden (Seite 99).
- Im Fortgeschrittenen-Modus kann ein Ereignis mit einem Bereich von Werten für den Beschleunigungssensor für vorwärts/rückwärts und links/rechts eingestellt werden (Seite 51).
- Einem Ereignis können mehrere Aktionsblöcke zugewiesen werden (Seite 18).
- Blöcke und Ereignis-Aktions-Paare können kopiert werden (Seite 13).
- Screenshots aus VPL Programmen können exportiert werden (Seite 90).
- Knöpfe für das rückgängig-machen ”Zurücksetzen” bzw. erneut-ausführen ”Vorwärtsmachen” (undo/redo) wurden hinzugefügt (Seite 89).
- Der Ausführen-Knopf blinkt grün, wenn das Programm verändert wurde (Seite 14).
- Es ist nicht mehr möglich, das Farbschema von VPL anzupassen.

Teil I

Tutorial

Kapitel 1

Ihr erstes Roboter-Projekt

Thymio kennen lernen

Abbildung 1.1 zeigt den Roboter Thymio von vorne und von oben. Oben sieht man einen runden Knopf in der Mitte (A) und die vier Richtungsknöpfe (B). Hinter den Knöpfen wird der Ladezustand der Batterie in grün (C) angezeigt. Dahinter sieht man die zwei oberen Lichter (D), die auf rot eingestellt sind. Der Roboter hat unten weitere solche Lichter (siehe Abbildung 3.1). Die schmalen schwarzen Rechtecke (E) vorne sind Sensoren, die Sie im Kapitel 4 kennenlernen werden.

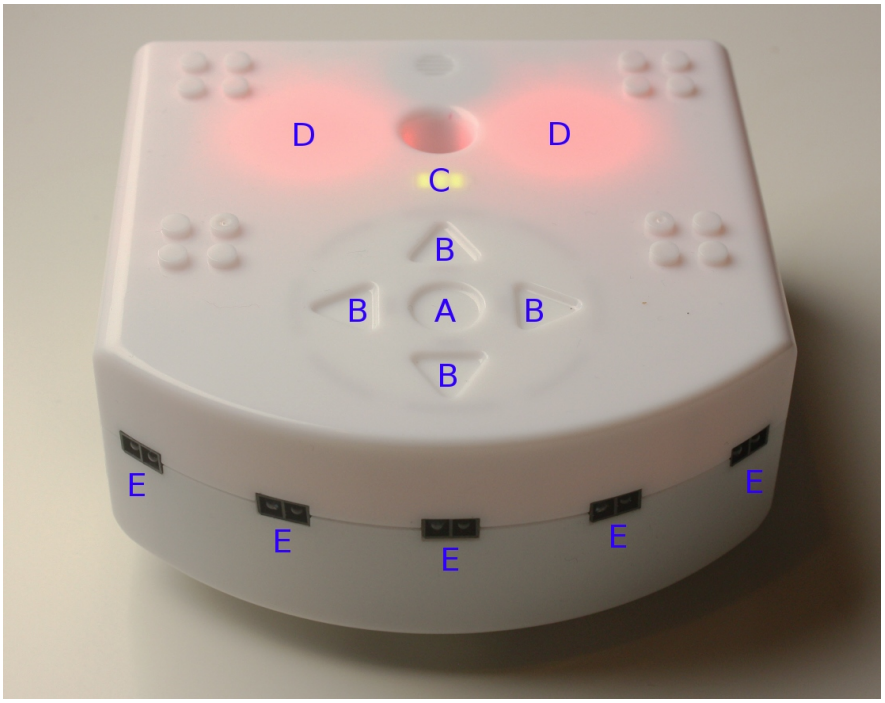


Abbildung 1.1: Thymio Roboter von vorne oben

Den Roboter verbinden und VPL starten

Verbinden Sie Ihren Thymio Roboter mit dem USB Kabel mit dem Computer. Der Roboter spielt eine kleine Tonabfolge ab und die Oberfläche leuchtet grün. Sollte der Roboter ausgeschaltet sein oder bei der Verbindung sich nicht einschalten, schalten sie ihn ein, indem sie während

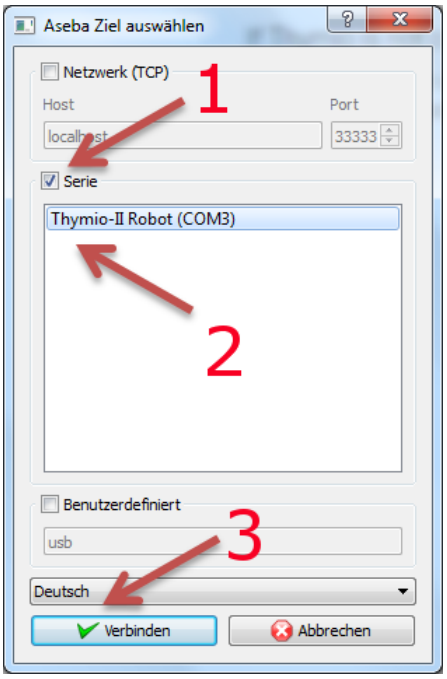


Abbildung 1.2: Thymio mittels USB-Kabel verbinden

fünf Sekunden den mittleren Knopf berühren. Starten sie nun VPL indem sie das VPL-Icon auf ihrem Computer mit Doppelklick anwählen.

Kleine Bilder

Wenn ein kleines Bild im Text erscheint, wird eine grosse Version davon am Rand dargestellt.

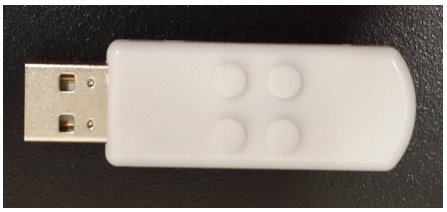
Möglicherweise verbindet sich VPL automatisch mit ihrem Roboter. Falls dies nicht der Fall sein sollte, wird das Bild wie in Abbildung 1.2 angezeigt. Wählen Sie das Kästchen **Serie** an, klicken Sie auf **Thymio-II Robot ...** darunter, wählen sie die Sprache und klicken sie auf **Verbinden**. Abhängig von der Konfiguration ihres Computers und des Betriebssystems, das sie einsetzen, kann es sein, dass die Einträge im Fenster und die folgenden Daten des Thymio etwas anders ausschauen, als in der Abbildung angezeigt.

Trick

Es ist auch möglich VPL aus dem Aseba Studio (textbasierte Programmierumgebung) heraus zu öffnen. Das VPL Plug-in befindet sich unten links im Fenster unter *Tool*.

Wireless Thymio

Es gibt eine Thymio-Version, die der Fähigkeit zur drahtlosen Kommunikation besitzt. Diese müssen nicht via Kabel verbunden werden. Der Wireless-Roboter wird mit einem kleinen Objekt ausgeliefert, dem sogenannten *Dongle*:



Stecken Sie den Dongle in einen USB-Steckplatz Ihres Computers, schalten Sie den Roboter ein und starten Sie VPL, wie weiter oben beschrieben. Wenn Sie das Verbindungsfenster sehen (Abbildung 1.2) wählen Sie die Zeile mit der Bezeichnung **Thymio-II Wireless (COMnn)**.

> **Wichtiger Hinweis**
>
> Sie benötigen immer noch ein USB-Kabel um den Roboter aufzuladen (Abbildung 1.4).

Bedienungsoberfläche von VPL

Die Bedienungsoberfläche von VPL wird unten angezeigt. Diese ist in sechs Bereiche aufgeteilt:

1. Oben hat sie eine Symbolleiste mit Symbolen, um Programme zu öffnen, zu speichern, auszuführen, etc.
2. Unterhalb der Symbolleiste befindet sich der Bereich, in dem man Programme für den Roboter erstellt.
3. In einem Nachrichten-Bereich wird angezeigt, ob das Programm funktioniert oder es werden Fehlermeldungen angezeigt.
4. Links eine Spalte mit den Ereignisblöcken.
5. Weiter rechts eine Spalte mit den Aktionsblöcken zur Konstruktion des Programms.
6. Auf der rechten Seite der Code, wie das Programm in die textbasierte Sprache von Aseba übersetzt wird.

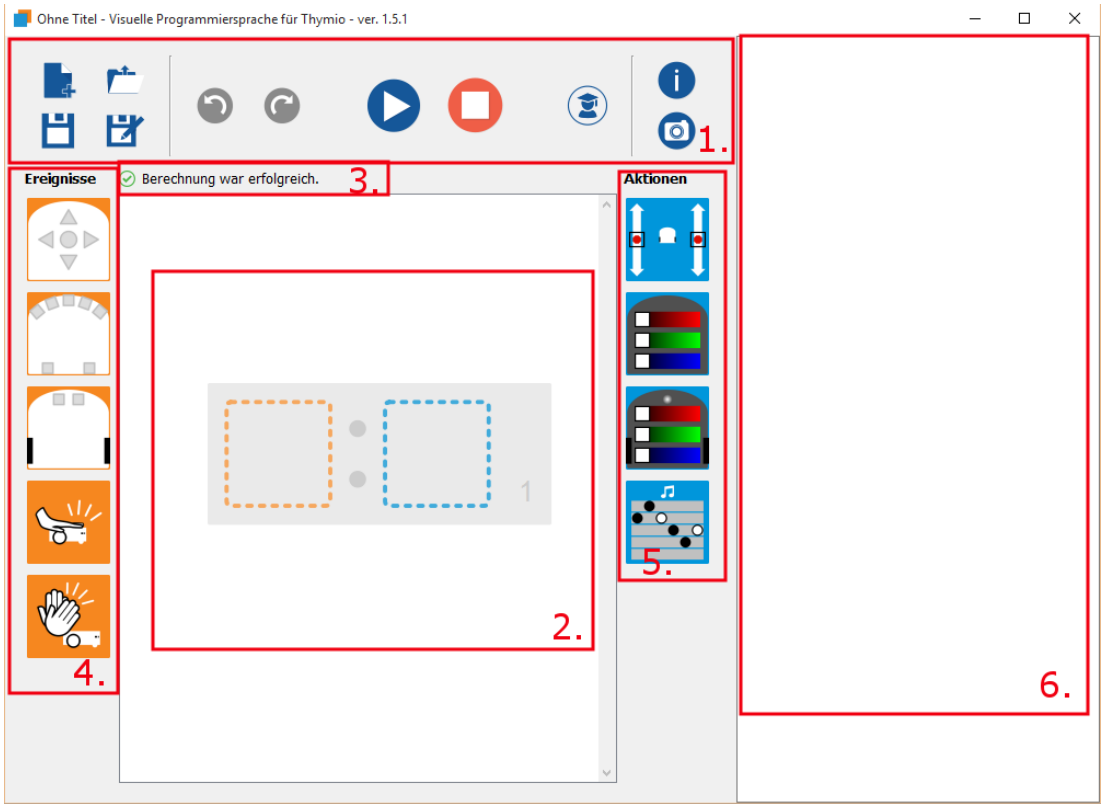


Abbildung 1.3: Die Bedienungsoberfläche von VPL

> **Weiterführende Informationen**
>
> Wenn Sie mit VPL ein Programm erstellen, wird eine Übersetzung in die textbasierte Programmiersprache AESL vorgenommen. Diese erscheint im Fenster rechts (Bereich 6. in der Abbildung). Dabei handelt es sich um das AESL Programm, welches gerade auf dem Roboter ausgeführt wird. Falls Sie mehr über AESL wissen wollen, werden Sie beim Kapitel 21 fündig, wo diese Übersetzung erklärt wird. Weitere Informationen finden Sie unter https://www.thymio.org/de:asebausermanual (Lern- und Referenzmaterial zu AESL und Aseba Studio).

Ein Programm erstellen

Beim Start von VPL erscheint ein leerer Programmierbereich.

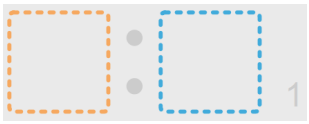
Falls Sie diesen Programmierbereich zurücksetzen wollen, nachdem Sie programmiert haben und neu beginnen wollen, klicken Sie auf (**Neu**).

Ein VPL-Programm besteht aus einem Paar oder mehreren Paaren von *Ereignis- und Aktions-Blöcken*. Ein Beispiel; das Paar:

die oberen Lichter werden auf rot gewechselt, wenn der vordere Knopf berührt wird.

> **Wichtiger Hinweis**
>
> Die Bedeutung eines Ereignis-Aktions-Paares ist:
> ***Wenn das Ereignis eintritt, wird die Aktion durchgeführt.***

Der Programmierbereich besteht zu Beginn aus einem leeren Rahmen für ein Ereignis-Aktions-Paar:



Um einen Block in den Programmierbereich zu bringen, wählt man ihn mit der Maus aus (Bereiche 4 und 5 in der Abbildung 1.3), und verschiebt ihn bei gedrückter Maustaste in das gestrichelte Quadrat. Sobald der Block über dem Quadrat ist, lässt man den Mausknopf wieder los und platziert damit den Block entsprechend.

> **Wichtiger Hinweis**
>
> Die beschriebene Technik nennt man *drag-and-drop* und wird häufig verwendet bei grafischen Benutzeroberflächen.

Beginnen Sie mit dem Ereignis-Knopf , indem Sie diesen von der linken Ereignis-Auswahl in das linke Quadrat ziehen. Wählen Sie nun den Aktionsblock für das obere Licht aus der rechten Auswahl und ziehen Sie ihn in das rechte Quadrat. Sie haben nun Ihr erstes Ereignis-Aktions-Paar erstellt.

Jetzt müssen wir das Ereignis und die Aktion so verändern, dass sie das machen was wir wollen. Beim Ereignis können wir zum Beispiel auf den Vorwärts-Knopf klicken; er wird dann rot:

Das bedeutet, dass **ein Ereignis stattfindet, wenn der Vorwärts-Knopf auf dem Thymio Roboter gedrückt wird**.

Der Farbaktionsblock (Aktions-Block für das Licht) hat drei Balken mit den Grundfarben Rot, Grün und Blau. Jeder dieser Balken hat ganz links ein weisses Quadrat. Die farbigen Balken mit dem weissen Quadrat werden Schieberegler (slider) genannt. Verschieben Sie das Quadrat von links nach rechts und Sie sehen, wie sich die Hintergrundfarbe des Blocks verändert - die

Farbe entspricht der späteren Farbe des Roboters. Alle Farben können durch Mischen der drei Grundfarben Rot, Grün und Blau erstellt werden. Schieben Sie das Quadrat im roten Balken nun ganz nach rechts und die Quadrate im grünen und blauen Balken ganz nach links. Die Farbe des Roboters leuchtet nun rot, ohne blau und ohne grün:

Speichern des Programms

Bevor Sie Ihr erstes Programm ausführen können, müssen Sie es speichern. Klicken Sie auf das Symbol in der Symbolleiste. Sie müssen dem Programm nun einen Namen geben; wählen Sie einen Namen, der Ihnen später hilft, sich daran zu erinnern was das Programm macht (zum Beispiel: **rot leuchten**). Wählen Sie einen Ort, wo Sie das Programm speichern möchten, z.B. auf dem Schreibtisch (Desktop) und drücken Sie **Speichern**.

> **Regelmässig speichern**
>
> Wenn Sie ein Programm ändern, vergessen Sie nicht, regelmässig zu **Speichern**, um die gemachten Änderungen nicht zu verlieren.

Ausführen des Programms

Um das Programm auszuführen, müssen Sie auf das Symbol **Laden und ausführen** klicken. Nun können Sie den Vorwärts-Knopf auf den Roboter berühren und dann sollte der Roboter rot leuchten.

> **Gratulation!**
>
> Sie haben Ihr erstes Programm erstellt und ausgeführt! Das Verhalten des Programms ist:
> **Wenn der Vorwärtsknopf gedrückt wird, wird der Roboter oben rot.**

Falls Sie das VPL Programm stoppen möchten, klicken Sie auf , die rote (**Halten**)-Taste. Dies ist wichtig, wenn ein Programm den Roboter bewegt, aber das Ereignis-Aktions-Paar für ein Anhalten nicht hinzugefügt wurde.

Den Roboter ausschalten

Um den Roboter auszuschalten, drücken Sie den mittleren Knopf für fünf Sekunden bis Sie eine Tonfolge hören. Der Akku des Roboters wird weiter aufgeladen, solange das Kabel an einen

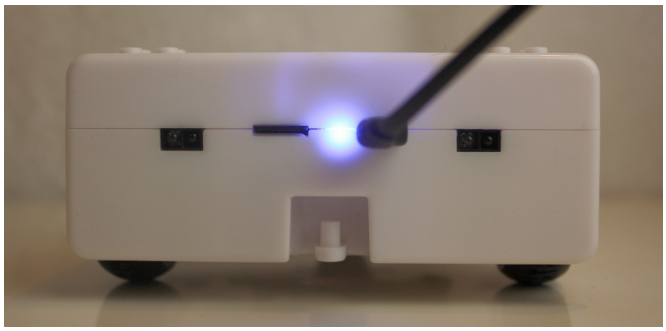


Abbildung 1.4: Die Rückseite von Thymio mit USB-Kabel und LED Ladeanzeige

eingeschalteten Computer angeschlossen ist. Der Akku lädt, wenn das kleine Licht neben dem USB Stecker rot leuchtet. Wenn es blau leuchtet, ist der Akku vollständig aufgeladen (Abbildung 1.4) - der Roboter kann nun vom USB-Kabel getrennt werden.

Trick

Der Akku wird schneller aufgeladen, wenn Sie ein Ladegerät eines Smartphones verwenden mit einem Micro-USB-Adapter.

Sollte die Verbindung des USB-Kabels während des Programmierens nicht funktionieren, blockiert VPL bis die Verbindung wieder hergestellt wird. Kontrollieren Sie beide Enden des Kabels, stellen Sie die Verbindung wieder her und schauen Sie, ob VPL wieder funktioniert. Falls ein Problem auftaucht, können Sie VPL schliessen, den Roboter neu anschliessen und VPL neu starten.

Ein Programm ändern

- Um ein Ereignis-Aktions-Paar zu löschen, klicken Sie auf ×, das oben rechts neben jedem Paar angezeigt wird.
- Um ein weiteres Ereignis-Aktions-Paar hinzuzufügen, klicken Sie auf +, das in der Mitte unter jedem Paar angezeigt wird.
- Um ein Ereignis-Aktions-Paar zu verschieben, können Sie es einfach an die neue Position ziehen.
- Um ein Ereignis-Aktions-Paar an eine andere Stelle im Programm zu kopieren, drücken Sie die Taste **Ctrl** oder **Strg** und ziehen Sie das Paar mit der Maus an die entsprechende Stelle.[1]

[1] Unter OSX (Mac) wird die Taste **Command** verwendet

Der blinkende Laden und Ausführen Knopf

Wenn Sie ein Programm verändern, beginnt der Knopf **Laden und Ausführen** in grüner und blauer Farbe zu blinken, was Sie daran erinnern soll, dass Sie diesen Knopf betätigen müssen, um das veränderte Programm auf den Roboter zu laden.

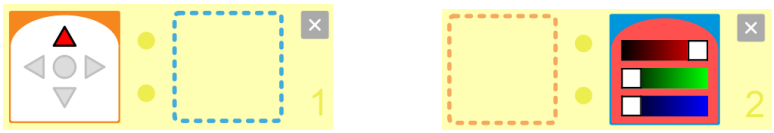
Falls Sie mit einem Programm experimentieren wollen, können Sie das bestehende Programm unter einem neuen Namen speichern, um so das ursprüngliche Programm nicht zu verlieren. Klicken Sie dazu einfach auf (**Speichern unter**) und geben Sie einen entsprechenden Namen ein.

Ein Programm öffnen

Angenommen Sie haben Ihr Programm gespeichert und den Roboter ausgeschaltet, möchten aber später wieder an Ihrem Programm weiter arbeiten. Verbinden Sie dazu den Roboter wie beschrieben und klicken Sie dann auf **Öffnen** und wählen Sie das Programm, das Sie öffnen möchten (zum Beispiel **rot-leuchten**). Das Programm wird jetzt im Programmierbereich angezeigt und Sie können es verändern.

Das aktuelle Ereignis-Aktions-Paar

Wenn man ein Ereignis-Aktions-Paar anklickt, wird sein Hintergrund gelb. Dies geschieht ebenso, wenn man ein Ereignis oder eine Aktion in einen leeren Block eingibt:



Das linke goldfarbene Quadrat ist der Bereich für das Ereignis; das rechte blaue Quadrat ist der Bereich für die erste oder einzige Aktion. Das Paar mit dem gelben Hintergrund nennen wir das *aktuelle* Paar.

Schnelle Eingabe

Mittels Doppelklick kann man einen Ereignis- oder Aktionsblock direkt in den Programmierbereich übernehmen. Dabei wird er an der aktuellen Stelle positioniert (gelber Hintergrund).

Die VPL Werkzeugleiste

Anhang A umfasst eine Beschreibung aller Elemente der VPL Werkzeugleiste. Schauen Sie bei Gelegenheit nach bis Sie die Handhabung gelernt haben.

Kapitel 2

Ändern der Farben

Farben anzeigen

Erstellen Sie ein VPL-Programm, das zwei verschiedene Farben oben auf dem Roboter anzeigt, wenn der Vorwärts- bzw. der Rückwärts-Knopf gedrückt wird. Wenn die Links- und Rechts-Knöpfe gedrückt werden, sollen unten am Roboter zwei andere Farben angezeigt werden.

Beispielprogramm **colors.aesl**


Wir brauchen vier Ereignis-Aktions-Paare, da es vier verschiedene Ereignisse gibt für das Drücken der vier Knöpfe. Mit jedem Ereignis ist eine Farbaktion verbunden. Beachten Sie den Unterschiede der Aktions-Blöcke für die obere Farben ( ) und die untere Farben) des Roboters. Der erste Block ändert die Farben der Lichter, die auf der Oberseite des Roboters liegen, während der zweite die Farben der Lichter der Unterseite des Roboters ändert. Im Block für die unteren Lichter sind die Räder als schwarze Balken abgebildet sowie mit einem weissen Punkt für die Ausbuchtung vorne unten (Abbildung 3.1).

Das fertige Programm ist im Abbildung 2.1 dargestellt.

Welche Farben werden angezeigt? In den ersten drei Ereignis-Aktions-Paaren ist jeweils eine Grundfarbe alleine gewählt. Die weissen Quadrate für die anderen beiden Farben sind jeweils ganz links. Diese drei Aktionen werden also reines Rot, Blau oder Grün anzeigen. Beim letzten Ereignis-Aktions-Paar werden aber zwei Farben gemischt: die weissen Quadrate sind bei Rot und Grün ganz rechts. Es wird also Rot und Grün gemischt, also Gelb dargestellt. Man erkennt, dass der Hintergrund des Kastens jeweils angepasst wird, wenn man den Regler bewegt.

Starten Sie das Programm ( ) und testen Sie die 4 Pfeiltasten aus. Abbildung 1.1 zeigt Thymio mit roter Farbe oben, Abbildung 3.1 zeigt ihn mit grüner Farbe unten.

Aufgabe 2.1

Experimentieren Sie mit den Farbreglern um herauszufinden, welche Farben möglich sind.

Hinweis

Durch Kombination der Farben Rot, Grün und Blau lassen sich alle Farben erzeugen (Abbildung 2.2)!

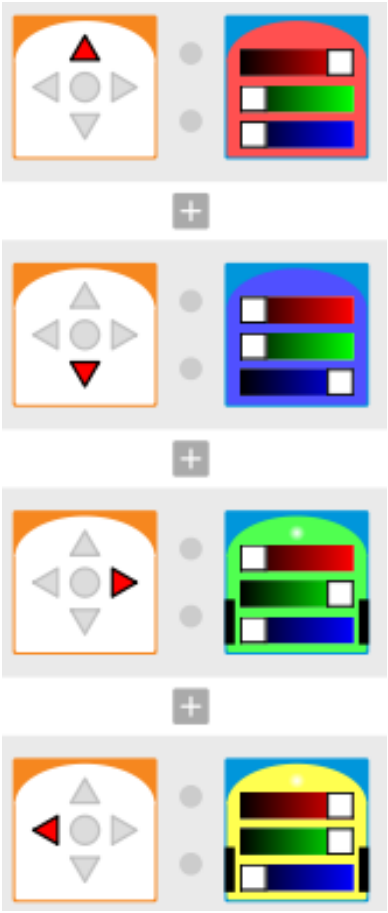


Abbildung 2.1: Ändern der Farben durch Betätigung der Knöpfe

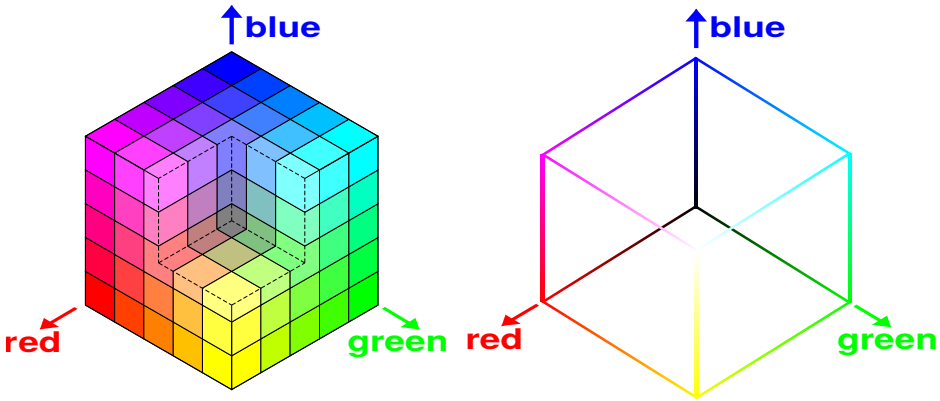
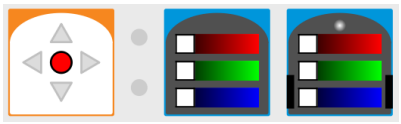


Abbildung 2.2: Der Rot-Grün-Blau-Farbwürfel (RGB)

Mehrere Aktionen nach einem Ereignis

Verändern wir das Programm so, dass alle Lichter ausgelöscht werden, wenn man den mittleren Knopf betätigt. Wir benötigen dazu *zwei* Aktionen für ein Ereignis in einem Ereignis-Aktions-Paar. Nachdem wir das Ereignis und die erste Aktion eingegeben haben, erscheint rechts von der Aktion ein mit einer grauen gestrichelten Linie umrahmtes Quadrat:

Sie können nun in dieses Quadrat die zweite Aktion hineinziehen, womit wir ein Paar mit einem Ereignis und zwei Aktionen erhalten:



Beispielprogramm **colors-multiple.aesl**

Vergessen Sie nicht, ▶ anzuklicken, um das Programm zu laden und zu starten. In Zukunft werden wir an diesen Schritt nicht mehr erinnern..

Regeln für Paare mit mehreren Aktionen

- Wenn ein Programm ausgeführt wird, werden *alle* Ereignis-Aktions-Paare ausgeführt.
- Man kann mehrere Ereignis-Aktions-Paare mit dem gleichen Ereignis konstruieren, solange sie verschiedene Parameter haben. Man kann mehrere Ereignisse mit Knöpfen haben, wenn dabei jeweils unterschiedliche Knöpfe betätigt werden müssen.
- Werden zwei oder mehrere Ereignis-Aktions-Paare eingegeben mit demselben Ereignis (und denselben Parametern), zeigt VPL eine Fehlermeldung an. (Feld 3 in Abbildung 1.3). Es ist nicht möglich ein Programm laufenzulassen solange Fehlermeldungen angezeigt werden.

Feedback zur Laufzeit

Jedes Mal, wenn eine Taste berührt wird, wird ein Ereignis erzeugt, und das Ereignis-Aktions-Paar, dem dieses Ereignis zugeordnet ist, wird ausgeführt. VPL bietet ein dynamisches Feedback zur Laufzeit, so dass Sie genau sehen können, welches Paar ausgeführt wird, nämlich dasjenige mit einem gelben Rahmen und einem gelben Pfeil auf der linken Seite:

Das Feedback wird nur kurz angezeigt, sobald das Ereignis eintritt. Wenn Sie zum Beispiel eine Taste berühren, wird ein Ereignis generiert, sobald Sie die Taste berühren; falls Sie die

Taste weiterhin berühren, werden keine zusätzlichen Ereignisse generiert — das Feedback wird entfernt. Es erscheint nur dann erneut, wenn Sie die Taste loslassen und dann wieder berühren.

Kapitel 3

Los, beweg dich

Vorwärts und rückwärts fahren

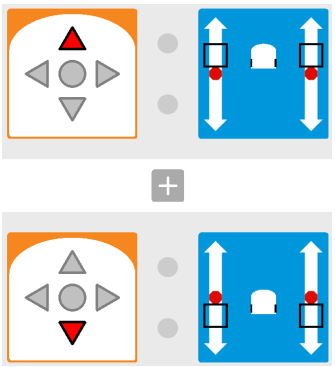
Thymio hat zwei Motoren mit denen er seine zwei Räder unabhängig voneinander antreiben kann. Beide Motoren können vorwärts und rückwärts drehen. Dadurch kann der Roboter vorwärts und rückwärts fahren. Wir wollen uns mit einem kleinen Projekt befassen, um mehr über die Motoren zu lernen.

Der Aktions-Block für die Motoren zeigt ein kleines Bild des Roboters in der Mitte mit zwei Schiebereglern links und rechts. Mit den beiden Balken können Sie die Geschwindigkeit der beiden Motoren einstellen, mit dem linken Balken die des linken Motors und mit dem rechten Balken die des rechten Motors. Wenn das weisse Quadrat in der Mitte ist (vertikal), dreht der Motor nicht. Sie können die Geschwindigkeit ändern, indem Sie das weisse Quadrat verschieben. Schiebt man das Quadrat nach oben, dreht der Motor vorwärts — je weiter oben, desto schneller; schiebt man es nach unten, dreht der Motor entsprechend rückwärts.

Erstellen Sie ein Programm, um den Roboter vorwärts fahren zu lassen, wenn der Vorwärts-Kopf gedrückt wird und rückwärts, wenn der Rückwärts-Knopf gedrückt wird.

Beispielprogramm **moving.aesl**

Wir brauchen zwei Ereignis-Aktions-Paare:



Ziehen Sie die Ereignis- und Aktions-Blöcke in die Programmierumgebung und stellen Sie für beiden Motoren die Schieberegler auf halbe Geschwindigkeit ein. Dies indem Sie die Quadrate für vorwärts fahren halb nach oben und für rückwärts fahren halb nach unten verschieben.

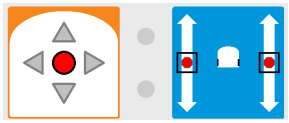
Führen Sie das Programm aus und drücken Sie die Knöpfe, um den Roboter vorwärts- und rückwärts fahren zu lassen.

Den Roboter anhalten

Hilfe! Ich kann den Roboter nicht mehr stoppen!

Klicken Sie auf das Symbol , um den Roboter zu stoppen.

Wir wollen diese Problem beheben, indem wir ein neues Ereignis-Aktions-Paar hinzufügen:



Dieses soll die Motoren stoppen, wenn der mittlere Knopf gedrückt wird. Wenn Sie den Motoraktionsblock in die Programmierumgebung ziehen, sind die Schieberegler in der Mitte, was die Motoren ausschaltet und den Roboter anhalten lässt.

Nicht vom Tisch fallen!

Wenn der Roboter auf dem Boden fährt, kann er im schlimmsten Fall in eine Wand fahren oder sein USB-Kabel herausziehen. Aber wenn der Roboter auf einem Tisch fährt, kann er auf den Boden fallen und kaputt gehen! Wir wollen den Roboter so programmieren, dass er anhält, sobald er die Tischkante erreicht.

> **Warnung!**
>
> Wenn der Roboter auf einem Tisch fährt, muss man bereit sein, um ihn aufzufangen, falls er hinunterfällt.

Drehen Sie Ihren Thymio auf den Rücken. Nun sehen Sie, dass er unten zwei kleine, schwarze Rechtecke mit optischen Elementen hat (Abbildung 3.1). Das sind *Bodensensoren*. Diese senden Infrarotlichtimpulse aus und messen wie viel Licht reflektiert wird. Auf einem hellen Tisch wird viel Licht reflektiert. Fährt der Roboter über die Tischkante wird wenig Licht reflektiert. Tritt dies ein, möchten wir, dass der Roboter stoppt.

> **Trick**
>
> Benutzen Sie einen hellen Tisch aber keinen aus Glas, da von diesem kein Licht reflektiert wird. Thymio kann dann nicht erkennen, ob er auf einem Tisch ist oder nicht.

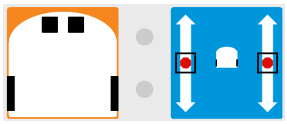
Ziehen Sie das Bodensensor-Ereignis in Ihr Programm. Oben hat dieser Block zwei kleine Quadrate. Wenn Sie sie anklicken, werden sie weiss mit rotem Rand, schwarz und dann wieder grau. Die Farben haben verschiedene Bedeutungen:

- **Grau**: Der Sensor wird nicht beachtet.

Abbildung 3.1: Thymios Unterseite mit den beiden Bodensensoren

- **Weiss**: Das Ereignis tritt ein, wenn viel Licht reflektiert wird. Links vom weissen Feld wird ein kleiner roter Punkt dargestellt; dies nimmt Bezug auf das rote Licht neben dem Sensor, das leuchtet, wenn der Sensor etwas entdeckt hat.[1]
- **Schwarz**: Das Ereignis tritt ein, wenn es wenig oder kein reflektierendes Licht gibt.

Damit der Roboter an der Tischkante anhält (wo es nur geringe Reflektion von Licht gibt) klicken Sie beide Quadrate bis sie schwarz werden. Erstellen Sie das folgende Ereignis-Aktions-Paar:



Stellen Sie den Roboter in der Nähe der Tischkante auf (auf die Tischkante ausgerichtet) und betätigen Sie den vorderen Knopf. Der Roboter sollte sich auf die Kante zu bewegen und anhalten, bevor er vom Tisch fällt.

Aufgabe 3.1

Experimentieren Sie mit der Geschwindigkeit. Gelingt es dem Roboter bei Höchstgeschwindigkeit doch noch rechtzeitig anzuhalten? Falls nicht: ab welcher Geschwindigkeit fällt er vom Tisch? Kann man den Roboter auch im Rückwärtsgang davon abhalten, vom Tisch zu fallen?

[1] Das weisse Quadrat hat einen roten Rand, der daran erinnert, dass das Ereignis eintrifft, wenn die Lichter neben dem Sensor aufleuchten.

Warnung!

Als wir dieses Programm getestet haben, *ist* der Roboter vom Tisch gefallen. Der Grund war, dass der Tisch eine abgerundete Kante hatte; sobald die fehlende Reflektion erkannt wurde, war es schon zu spät und der Roboter kippte nach vorne, verlor an Stabilität und viel vom Tisch.

Kapitel 4

Ein Roboter als Haustier

Einen Roboter mit unabhängigem Verhalten nennen wir einen *autonomen Roboter* und wir verbinden mit ihm Eigenschaften von lebenden Geschöpfen wir Katzen und Hunden. Das Verhalten wird erreicht durch *Rückkoppelung*: der Roboter nimmt die Umgebung über ein Ereignis wahr und passt sein Verhalten über Aktionen an.

Der Roboter gehorcht

Zuerst lehren wir den Roboter, uns zu gehorchen. Und zwar so, dass er sich normalerweise nicht bewegt; wenn er aber eine Hand vor sich wahrnimmt, soll er auf diese Hand zu fahren.

Beispielprogramm **obeys.aesl**

Der Roboter hat vorne fünf und hinten zwei horizontale Distanzsensoren. Diese sind ähnlich aufgebaut, wie die Bodensensoren auf der Unterseite des Roboter, die wir in Kapitel 3 benutzt haben. Wenn Sie Ihre Hand langsam näher an die Sensoren des Roboters heranführen, leuchtet ein kleines, rotes Licht neben dem Sensor auf und zeigt so an, dass die Hand erkannt wurde (Abbildung 4.1).

Der Block  wird gebraucht, um zu erfahren, ob etwas nahe am Sensor ist oder nicht. In beiden Fälle tritt ein Ereignis ein. Die schmalen, grauen Quadrate (fünf vorne und zwei hinten) können wahrnehmen, wann ein Ereignis stattfindet. Indem auf ein Quadrat gedrückt wird, ändert sich dieses von grau auf weiss, von weiss auf schwarz und zurück auf grau. Die Bedeutung der Farben ist für diesen Block sind:

- **Grau**: Der Sensor wird nicht beachtet.
- **Weiss**: Eine Aktion wird ausgelöst, falls ein Objekt im Bereich des Sensors erkannt wird.

Abbildung 4.1: Thymios Vorderseite. Zwei Sensoren erkennen die Finger.

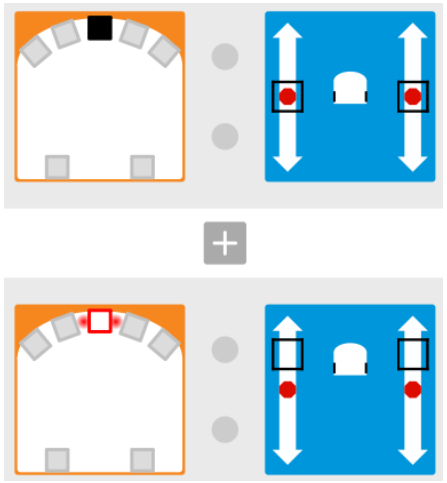


Abbildung 4.2: Bewegung auf Hand zu

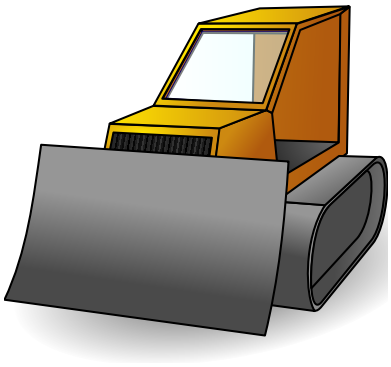


Abbildung 4.3: Ein Bulldozer mit Raupen

- **Schwarz**: Eine Aktion wird gestartet, falls *kein* Objekt im Bereich des Sensors erkannt wird.

Wenn Sie ein Ereignis auslösen wollen, wenn ein Objekt nahe beim Sensor ist, verwenden Sie ein weisses Quadrat, weil dann das Objekt viel Licht reflektieren wird. Wollen Sie andererseits ein Ereignis auslösen, wenn *kein* Objekt in der Nähe des Sensors ist, verwenden Sie ein schwarzes Quadrat, da dort nur wenig Licht reflektiert wird.

Um das gewünschte Verhalten zu implementieren, müssen wir zwei Ereignis-Aktions-Paare verwenden (Abbildung 4.2). Im ersten Paar ist der mittlere Frontsensor schwarz und die damit verbundene Aktion ist, dass die Motoren ausgeschaltet sind. Wenn also der Roboter kein Objekt erkennt, wird er sich nicht bewegen, bzw. anhalten, falls er in Bewegung war. Im zweiten Paar ist der mittlere Frontsensor weiss und die Schieberegler des Motorblocks sind beide vorne in Richtung Spitze; wenn also eine Hand vor dem mittleren Sensor erkannt wird, tritt das Ereignis ein und der Roboter setzt sich relativ rasch in Bewegung.

Steuerung des Roboter Thymio

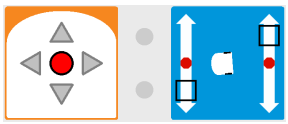
Der Roboter Thymio hat kein Steuerrad wie ein Auto und kein Lenker wie ein Fahrrad. Wie kann man ihn also lenken? Der Roboter benutzt ein *Differentialgetriebe*, welches ähnlich funktioniert wie bei einem Raupenfahrzeug, z.B. einem Bulldozer (Abbildung 4.3). Die gewünschte Richtung wird mit Hilfe von *unterschiedlichen* Geschwindigkeiten des linken und rechten Rades erreicht. Dreht das rechte Rad schneller als das linke, biegt das Fahrzeug nach links ab und dreht das linke Rad schneller als das rechte, biegt das Fahrzeug nach rechts ab.

In VPL wendet man das Differentialgetriebe an, indem man den linken und rechten Schieberegler des Motoraktionsblocks einzeln einstellt und dadurch die Geschwindigkeit der Räder verschieden einstellen kann. Um einen möglichst grossen Unterschied der Geschwindigkeiten zu erreichen, lässt man die Räder in die entgegengesetzten Richtungen drehen. Tatsächlich

dreht sich das Fahrzeug auf der Stelle, falls sich die Räder mit der genau *gleichen* Geschwindigkeit in die entgegengesetzte Richtung bewegen.

Zum Beispiel wird im Aktionsblock der linke Regler auf schnelle Geschwindigkeit rückwärts und der rechte Regler auf schnelle Geschwindigkeit vorwärts eingestellt. Als Resultat wird der Roboter eine enge Linkskurve vollführen, wie dies auf dem kleinen Bild des Roboter bzw. des Motoraktionsblocks dargestellt ist.

Experimentieren Sie mit einem Ereignis-Aktions-Paar wie diesem:



Stellen Sie den linken und rechten Schieber ein und lassen Sie das Programm laufen. Stoppen Sie dann den Roboter durch klicken auf . Ändern Sie nun die Schieberegler und versuchen Sie es erneut!

Trick

Das kleine Bild in der Mitte stellt den Thymio in seiner Bewegung dar. Wenn die Animation anhält, zeigt Thymio in die Richtung, in die sich der Roboter bewegen wird.

Der Roboter mag Sie

Ein echtes Haustier folgt seinem Meister. Damit der Roboter Ihrer Hand folgt, muss man zwei weitere Ereignis-Aktions-Paare hinzufügen. Falls der Roboter ein Objekt vor seinem ganz links platzierten Distanzsensor wahrnimmt, soll er nach links drehen und falls er ein Objekt vor seinem ganz rechts platzierten Distanzsensor wahrnimmt, soll er nach rechts abbiegen.

Beispielprogramm **likes.aesl**

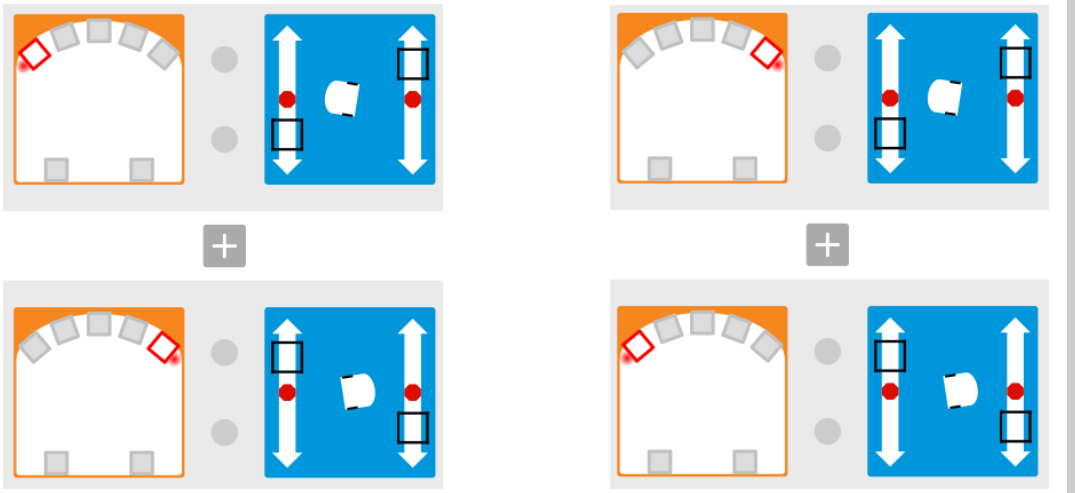
Das Programm für "Der Roboter mag Sie" besteht aus zwei Ereignis-Aktions-Paaren (Abbildung 4.4(a)). Probieren Sie die Regler an jedem Motoraktionsblock aus!

Aufgabe 4.1

Verändern Sie den Haustierroboter so, dass er vorwärts fährt, falls das Programm am laufen ist und stoppt, falls er das Ende des Tisches wahrnimmt.

Aufgabe 4.2

Was geschieht, wenn Sie die Reihenfolge der Ereignis-Aktions-Paare ändern?



(a) Der Roboter mag Sie

(b) Der Roboter mag Sie nicht

Abbildung 4.4: Programme für den Haustier-Roboter

Der Roboter mag Sie nicht

Manchmal mag dein Roboterhaustier in schlechter Laune sein und von deiner Hand zurückweichen. Erstellen Sie ein Programm, welches dieses Verhalten auslöst.

Beispielprogramm **does-not-like.aesl**

Öffnen Sie das Programm für den Haustierroboter, der Sie mag und ändern Sie die Zusammenhänge zwischen Ereignissen und Aktionen. Das Erkennen eines Hindernisses beim linken Sensor bewirkt, dass der Roboter nach rechts abbiegt, während das Erkennen eines Hindernisses beim rechten Sensor bewirkt, dass der Roboter nach links abbiegt (Abbildung 4.4(b)).

Aufgabe 4.3

Die vorderen horizontalen Sensoren sind von 0 bis 4 nummeriert (von links nach rechts). Die hinteren Sensoren tragen die Nummern 5 (links) und 6 (rechts). Ändern Sie das Programm Abbildung 4.4 so dass anstelle der Sensoren 0 und 4...

- die Sensoren 1 und 3 verwendet werden für die Drehung nach links bzw. rechts,
- die Sensoren 0 und 1 verwendet werden für die Linkskurve und die Sensoren 3 und 4 für die Rechtskurve.
- Fügen Sie Ereignis-Aktions-Paare hinzu für die Sensoren 5 und 6.

Trick

Anhang D erklärt, wie man die Schieber für die Motorengeschwindigkeit präzise steuern kann.

Sensoren im Fortgeschrittenen-Modus

Im Fortgeschrittenen-Modus (siehe Kapitel 7), gibt es eine zusätzliche Art, um die Ereignisse abhängig von den Sensoren auszulösen (nachzulesen in Anhang D).

Kapitel 5

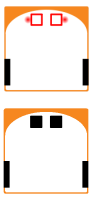
Der Roboter findet seinen Weg selbständig

Stellen Sie sich Lagerhaus mit Roboterwagen vor, welche Gegenstände von einer zentralen Verteilstelle herbringen. Dazu werden Linien auf den Boden des Lagerhauses gezeichnet und die Roboter erhalten die Anweisung bestimmten Linien zu folgen, bis diese den Lagerplatz für den gewünschten Gegenstand erreichen. Um dies zu tun, muss der Roboter diese Linien erkennen können. Schreiben Sie ein Programm, welches den Roboter einer Linie auf dem Boden folgen lässt.

Beispielprogramm **follow-line.aesl**

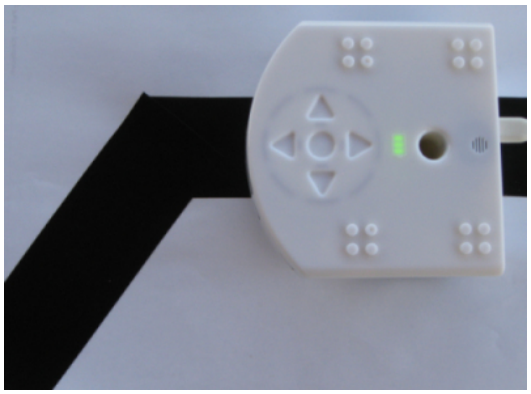
Die Aufgabe ”Einer-Linie-Folgen” zeigt die Herausforderungen bei der Konstruktion von Robotern in der realen Welt auf. So kann es sein, dass die Linie zum Beispiel nicht perfekt gerade ist, Staub kann auf der Linie liegen und diese abdecken, Dreck führt eventuell dazu, dass ein Rad weniger schnell dreht, als das andere. Damit der Roboter der Linie folgen kann, benötigt er eine *Steuereinheit*, die entscheidet, wie viel Leistung jeder Motor benötigt, abhängig von den Daten, die von den Sensoren geliefert werden.

Die Linie und der Roboter

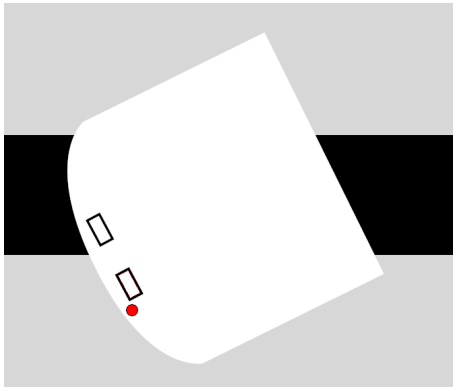


Um einer Linie zu folgen, benutzen wir die Bodensensoren (Kapitel 3). Zur Erinnerung: diese Sensoren arbeiten durch das Aussenden von Infrarotlicht (unsichtbar für das menschliche Auge) und dem Messen des reflektierten Lichts. Falls der Boden von heller Farbe ist, wird der Sensor viel reflektiertes Licht wahrnehmen und das Ereignis wird ausgelöst. Somit brauchen wir eine dunkle Linie, die wenig Licht reflektiert und das Ereignis auslöst. Das lässt sich einfach mit dem Aufmalen oder Ausdrucken einer schwarzen Linie oder mit dem Anbringen eines schwarzen Isolierklebebandes auf den Boden bewerkstelligen (Abbildung 5.1(a)). Das Isolierband muss breit genug sein, damit beide Bodensensoren die schwarze Fläche gleichzeitig erkennen können. Eine Breite von 5 Zentimetern ist auf jeden Fall ausreichend, um dem Isolierband folgen zu können, auch wenn die Breite ein wenig variiert.

Um ”Einer-Linie-folgen” zu implementieren, bringen wir den Roboter zuerst dazu, sich vorwärts zu bewegen, wenn *beide* Sensoren eine dunkle Oberfläche erkennen (das schwarze Isolierband) und stoppen, falls *beide* Sensoren die helle Oberfläche erkennen (nicht das Isolierband, sondern den hellen Boden bzw. Tisch). Die Ereignis-Aktions-Paare werden in der folgenden Abbildung 5.2(a) dargestellt.

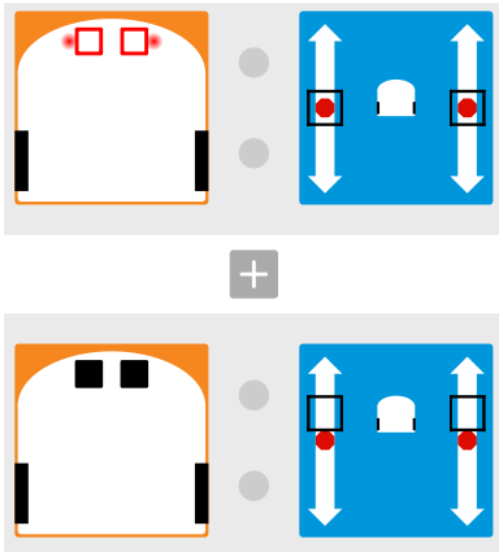


(a) Thymio folgt einer Linie aus Isolierband

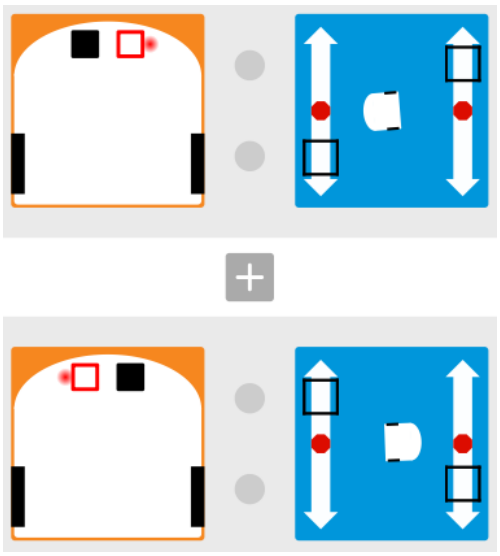


(b) Der linke Sensor ist neben und der rechte Sensor auf dem Isolierband Der rote Punkt zeigt an, dass der linke Sensor viel reflektiertes Licht erkennt

Abbildung 5.1: Thymio auf dem Isolierband



(a) Starten und stoppen des Roboters



(b) Korrektur bei Abweichungen

Abbildung 5.2: Programm ”Einer Linie Folgen”

Trick

Achten Sie darauf, ein USB-Kabel von ausreichender Länge zu verwenden (ca. 2m), damit Thymio trotz Bewegung mit dem Computer verbunden bleibt. Verlängerungskabel finden Sie in jedem Computershop.

Wireless Thymio

Falls Sie allerdings die neue Version von Thymio besitzen, haben Sie beinahe unbegrenzte Bewegungsfreiheit.

Ihre erste Steuerungseinheit

Der nächste Schritt besteht im Programmieren der Steuerungseinheit, die den Roboter der Linie folgen lässt. Wir benötigen wieder zwei Ereignis-Aktions-Paare (vgl. Abbildung 5.2(b)).

- Falls der Roboter nach der *linken* Seite vom Isolierband abkommt, wird der *linke* Sensor den hellen Boden erkennen, während der *rechte* Sensor immer noch das dunkle Isolierband erkennt. In diesem Fall soll der Roboter ein wenig nach *rechts* abbiegen.
- Falls der Roboter nach der *rechten* Seite vom Isolierband abkommt, wird der *rechte* Sensor den Boden erkennen, während der *linke* Sensor immer noch das dunkle Isolierband erkennt. In diesem Fall soll der Roboter ein wenig nach *links* abbiegen.

Einstellen der Parameter

Es ist einfach zu erkennen, dass der Roboter nach rechts drehen muss, falls er von der linken Seite des Isolierbands abkommt, wie in Abbildung 5.2(b) dargestellt. Aber wie weit nach rechts soll er drehen? Falls die Drehung zu gering ausfällt, wird der rechte Sensor eventuell *auch* vom Isolierband abkommen, bevor der Roboter auf die Spur zurückgelangt. Falls der Roboter jedoch zu stark nach rechts dreht, riskiert der Roboter die Isolierband-Spur auf der anderen Seite zu verlieren. In jedem Fall sind starke Drehungen gefährlich für den Roboter und jegliches Gut, das er transportiert.

Sie werden mit den Geschwindigkeiten des linken und rechten Motors experimentieren müssen, bis Sie *zuverlässige* Werte finden. Zuverlässig heisst hier, dass der Roboter mit denselben Programmeinstellungen mehrmals erfolgreich der Linie folgt. Platzieren Sie bei jedem Versuch den Roboter ein wenig anders in Bezug auf Richtung und Position zum Isolierband.

Die Geschwindigkeit des Roboters auf dem Isolierband ist auch eine wichtiger Parameter. Ist der Roboter zu schnell, ist dieser schon vom Isolierband weggefahren, bevor die Drehbewegung die Richtung des Roboters verändern konnte. Ist der Roboter jedoch zu langsam, wird niemand Ihren Roboter kaufen um in einem Lagerhaus einzusetzen.

Aufgabe 5.1

Der Roboter stoppt, falls beide Sensoren erkennen, dass sie vom Isolierband abgekommen sind. Verändern Sie das Programm so, dass der Roboter eine leichte Drehung nach links vollführt mit der Absicht das Isolierband wiederzufinden. Versuchen Sie es auf einem Isolierband mit einer Linkskurve, wie in der Abbildung 5.1(a) angezeigt. Versuchen Sie die Vorwärts-Geschwindigkeit des Roboters zu erhöhen. Was passiert, wenn der Roboter das Ende des Isolierbandes erreicht hat?

Aufgabe 5.2

Verändern Sie das Programm aus der vorangehenden Aufgabe so, dass der Roboter nach rechts dreht, falls er vom Isolierband abkommt. Was passiert? Es wäre wünschenswert, wenn der Roboter *sich erinnern* könnte, welchen Sensor als letzter den Kontakt zum Isolierband verloren hat, um den Roboter in die korrekte Richtung zu führen und das Isolierband wieder zu finden. In Kapitel 8 werden wir lernen, wie solche Informationen gespeichert werden.

Aufgabe 5.3

Experimentieren Sie mit unterschiedlichen Arten von Linien bzw. Anordnungen des Isolierbandes:

- weite Kurven
- enge Kurven
- Zickzack-Linien
- breitere Linien (benutzen Sie dazu die doppelte Isolierbandbreite)
- schmälere Linien (schneiden Sie dazu das Isolierband in zwei Teile)

Veranstalten Sie mit Freunden ein Roboterrennen: Welcher Roboter folgt erfolgreich den meisten Linien? Welcher Roboter fährt am schnellsten eine Linie ab?

Aufgabe 5.4

Diskutieren Sie den Effekt der folgenden Veränderungen auf den Thymio Roboter und dessen Fähigkeit der Linie zu folgen.

- Die Messereignisse der Bodensensoren erfolgen öfters bzw. weniger oft als 10 Mal pro Sekunde.
- Die Sensoren sind weiter auseinander bzw. näher beisammen.
- Es gibt mehr als zwei Bodensensoren an der Unterseite des Roboters.

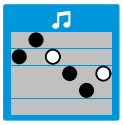
Kapitel 6

Schnickschnack

Wir wollen etwas Spass haben mt Thymio. Betrachten wir, wie Thymio Musik macht, auf Geräusche und auf Berührung reagiert.

Musik spielen

Der Roboter Thymio enthält einen Synthesizer den man programmieren kann, um einfache Melodien zu spielen. Dazu verwenden Sie den Aktionsblock für Musik:



Beispielprogramm **bells.aesl**

Sie werden wohl kaum zu einem zweiten Beethoven werden—es stehen nur sechs Noten, in fünf Tonlagen und zwei Tonlängen zur Verfügung—aber Sie können eine Melodie komponieren, die Ihren Roboter einzigartig macht. Die Abbildung 6.1 zeigt zwei Ereignis-Aktions-Paare, welche mit einer Melodie antworten, falls der vordere oder der hintere Knopf gedrückt wird. Jedem Ereignis ist eine andere Melodie zugeordnet.

Die kleinen Kreise stellen Noten dar. Ein weisser Kreis steht für einen langen Ton, ein schwarzer für einen kurzen. Die Tonlänge ändern Sie, indem Sie auf den Kreis klicken. Die fünf grauen, horizontalen Balken stehen für die Tonhöhe. Klicken Sie auf den *Balken* über oder unter dem Kreis, um den Kreis zu bewegen oder verschieben Sie ihn per drag and drop.

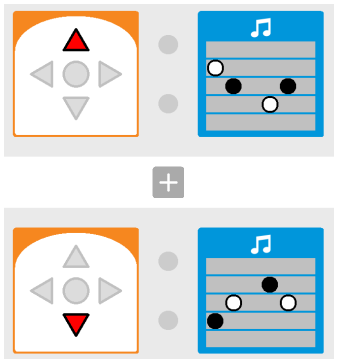


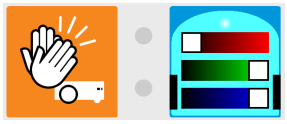
Abbildung 6.1: Melodie spielen

Aufgabe 6.1

Schreiben Sie ein Programm, das es Ihnen erlaubt eine Morsebotschaft zu verschicken. Eine Morsebotschaft wird mit langen und kurzen Tönen kodiert. (*Striche* für lange Töne und *Punkte* für kurze Töne). Zum Beispiel wird der Buchstabe *V* mit drei Punkten und einem Strich kodiert.

Roboter durch Töne steuern

Der Roboter Thymio hat ein Mikrofon. Das Ereignis tritt ein, wenn ein lautes Geräusch aufgenommen wird, wie z.B. in die Hände klatschen. Das Ereignis-Aktions-Paar:



wird die unteren Lichter in der Farbe Türkis einschalten, wenn man in die Hände klatscht.

Information

In einer lauten Umgebung kann ein Geräusch eventuell nicht als Ereignis verwendet werden, da durch den hohen Geräuschpegel dauernd Ereignisse ausgelöst werden.

Aufgabe 6.2

Schreiben Sie ein Programm, dass den Roboter losfahren lässt, wenn man in die Hände klatscht und den Roboter stoppt, wenn man einen Schalter betätigt.

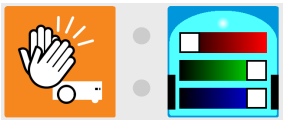
Schreiben Sie ein Programm, das umgekehrt funktioniert: Der Roboter soll losfahren, wenn man auf den Schalter drückt und stoppen, wenn man in die Hände klatschst.

Gute Arbeit Roboter!

Haustiere machen nicht immer das, was wir von ihnen verlangen. Manchmal brauchen sie einen freundschaftlichen Klaps um sie zu ermutigen. Genau gleich funktioniert das mit Ihrem Roboter. Thymio enthält einen Erschütterungssensor, welcher ein Ereignis auslöst, falls dem Roboter kurz auf seine Oberseite geklopft wird. So bewirkt zum Beispiel das Ereignis-Aktions-Paar:

dass die Lichter angehen, falls auf die Oberseite des Roboters geklopft wird.

Erstellen Sie ein Programm für dieses Ereignis-Aktions-Paar und eines für das folgende Paar:



welches die Bodenlichter in türkis angehen lässt, wenn Sie in die Hände klatschen.

Beispielprogramm **whistles.aesl**

Können Sie nur die oberen Lichter einschalten? Das ist schwierig, da ein Klaps immer auch ein Geräusch erzeugt, welches laut genug sein kann, um ebenfalls die Bodenlichter einzuschalten. Mit ein bisschen Übung wird es Ihnen möglich sein, dem Roboter einen so gefühlsvollen Klaps zu geben, dass das Geräusch kein Ereignis auslöst.

Aufgabe 6.3

Schreiben Sie ein Programm, das den Roboter vorwärts fahren lässt bis er die Wand berührt (Erschütterung).

Achten Sie darauf, dass der Roboter **langsam** fährt und durch den Aufprall nicht beschädigt wird.

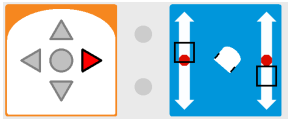
Kapitel 7

Angenehme Zeit (Fortgeschrittener Modus)

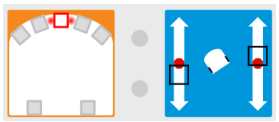
Im Kapitel 4 haben wir ein Roboterhaustier programmiert, das uns entweder mag oder nicht mag. Nun stellen wir uns ein etwas komplizierteres Verhalten vor: ein schüchternes Haustier, welches sich nicht entscheiden kann, ob es uns mag oder nicht. Anfänglich wird das Haustier sich unserer Hand zuwenden, dann aber zurückweichen und schlussendlich sich wieder zu unserer Hand hinbewegen.

Beispielprogramm **shy.aesl**

Das Verhalten des Roboters ist wie folgt: Falls der Rechte Schalter berührt wird, bewegt sich der Roboter nach rechts, falls er Ihre Hand wahrnimmt, bewegt er sich nach links, aber nach einer Weile bereut er die Entscheidung und bewegt sich wieder zurück. Wir wissen, wie wir die Ereignis-Aktions-Paare für die erste Bewegung erstellen:



und wie für das zurück-bewegen, falls Ihre Hand erkannt wird:



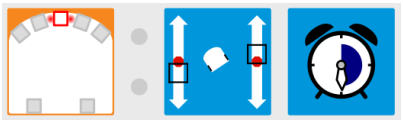
Das Verhalten des Zurückweichens "nach einer Weile" kann in zwei Ereignis-Aktions-Paare zerlegt werden:

- *Falls* der Roboter sich wegbewegt → *starte einen Timer* für zwei Sekunden.
- *Falls* der Timer null erreicht → *drehe* nach rechts ab.

Wir brauchen eine neue *Aktion* für das erste Verhalten und ein neues *Ereignis* für das zweite Verhalten.

Die Aktion ist, einen *Timer* einzustellen, welcher wie ein Wecker funktioniert . Normalerweise stellen wir den Wecker auf eine bestimmte Zeit ein, aber falls ich den Wecker auf meinem Smartphone auf eine bestimmte Zeit einstelle, wird mir diese als Zeitdauer angegeben: "Wecker läutet in 11 Stunden und 23 Minuten". Der Timerblock arbeitet auf die selbe Art und Weise. Der Timer wird auf eine bestimmte Anzahl Sekunden eingestellt, sobald das Ereignis stattfindet und die Aktion ausgelöst wird. Der Timer kann auf bis zu vier Sekunden eingestellt werden. Klicken Sie irgendwo innerhalb des schwarzen Kreises, wo die Oberfläche der Uhr gezeigt wird (aber nicht auf den schwarzen Kreis selber). Nach einer kleinen Animation wir die Zeitdauer des Timers (bis der Alarm ausgelöst wird) in blauer Farbe angezeigt.

Das Ereignis-Aktions Paar für das erste, oben beschriebene Verhalten ist:



Der Timer ist auf zwei Sekunden eingestellt. Falls das Ereignis Handerkennung stattfindet, werden zwei Aktionen ausgelöst: Drehung des Roboters nach links und Starten des Timers.

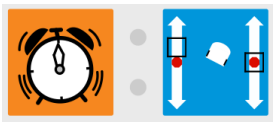
Der zweite Teil dieses Verhaltens benötigt ein Ereignis, das stattfindet, falls der Alarm ausgelöst wird. Dieses tritt ein, sobald die eingestellte Zeit abgelaufen ist. Der Ereignisblock  zeigt dann einen klingelnden Wecker.

Fortgeschrittener Modus

Timer werden nur im *Fortgeschrittenen Modus* unterstützt. Klicken Sie auf  um in den fortgeschrittenen Modus zu gelangen.

Das Icon ändert in  ; durch erneutes Klicken gelangt man zurück in den *Standard-Modus*.

Das Ereignis-Aktions-Paar ist:



welches den Roboter zurück nach rechts drehen lässt, sobald der Timer abgelaufen ist.

Aufgabe 7.1

Schreiben Sie ein Programm, welches den Roboter bei Höchstgeschwindigkeit für drei Sekunden vorwärts fahren und danach wieder zurückkehren lässt, sobald der Vorwärtsschalter gedrückt wird. Fügen Sie ein Ereignis-Aktions-Paar hinzu, das die Fahrt des Roboters stoppt, falls der mittlere Schalter gedrückt wird.

Kapitel 8

Zustand: Mach nicht immer dasselbe (Fortgeschritter Modus)

Ein Programm in VPL ist eine Liste von Ereignis-Aktions-Paaren. *Alle* Ereignisse werde eins nach dem anderen überprüft, ob sie stattfinden und falls ja, wird die dazugehörige Aktion ausgelöst. Danach beginnt die Überprüfung von vorne. Wir möchten nun, dass einige Ereignis-Aktions-Paare zu einem bestimmten Zeitpunkt aktiv sind und andere nicht.

Zum Beispiel in Kapitel 5, falls der Roboter vom Isolierband abkommt, möchten wir, dass er nach links oder nach rechts dreht, um das Isolierband zu suchen, je nach dem, von welcher Seite her er das Isolierband ”verloren” hat. Wir werden zwei Ereignis-Aktions-Paare benötigen: eines, um den Roboter nach links zu drehen, wenn er das Isolierband rechts verlassen hat und ein zweites um nach rechts zu drehen, wenn er das Isolierband links verlassen hat.

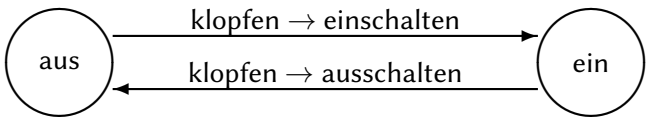
Klopf, klopf

In vielen Programmen haben wir einen Schalter benutzt, um ein Verhalten des Roboters auszulösen und einen anderen Schalter, um dieses Verhalten wieder zu stoppen. Wenn wir aber an den Startschalter eines Computers denken, haben wir denselben Schalter, um den Computer ein- oder auszuschalten. Der Schalter *weiss* in welchem Zustand er sich gerade befindet: **eingeschaltet** oder **ausgeschaltet**.

Schreiben Sie ein Programm, das die Lichter des Roboters einschaltet, falls er berührt wird und wieder ausschaltet, wenn er ein zweites Mal berührt wird.

Beispielprogramm **tap-on-off.aesl**

Das Verhalten kann auf einfache Art mit einem *Zustandsdiagramm* dargestellt werden:



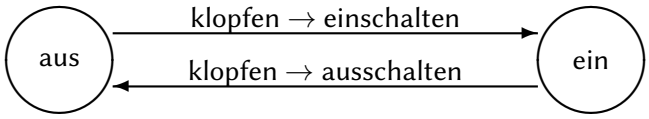
Im Diagramm sind zwei Zustände durch Kreise angezeigt, die mit den Namen des Zustandes **ein** und **aus** angeschrieben sind. Der Roboter kann von dem Zustand **ein** zum Zustand **aus** und zurück wechseln, aber nur durch das Befolgen der Instruktionen die auf den Pfeilen angegeben sind. Die Instruktionen beschreien, wann ein Übergang zu einem anderen Zustand geschehen kann und was dann genau geschieht.

- *Falls* der Zustand **aus** ist ***und*** ein *Klopfen* stattfindet, → schalten die Lichter *ein* ***und*** der Zustand wechselt auf **ein**.

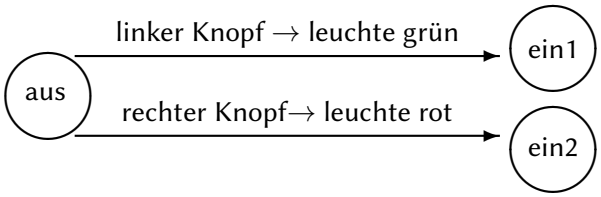
- *Falls* der Zustand **ein** ist ***und*** ein *Klopfen* stattfindet, → schalten die Lichter *aus* ***und*** der Zustand wechselt auf **aus**.

Das fett hervorgehobene Wort "***und***" vor dem Pfeil → bedeutet, dass es *zwei Bedingungen* gibt, die wahr sein müssen, damit der Umschaltvorgang ausgeführt wird. (a) Der Roboter muss in einem bestimmten Zustand sein und (b) das Ereignis muss stattfinden. Wenn beide Bedingungen wahr sind, wird das Umschalten durchgeführt, dies führt dazu, dass der Status geändert und die Aktion ausgelöst wird, die nach dem Pfeil → geschrieben steht.

Es ist wichtig zu verstehen, dass die zwei Teile der Bedingung unabhängig sind. In dem oben aufgeführten Diagramm (hier wiederholt):



erscheint das Ereignis *Klopfen* zweimal, aber die Aktion, ausgelöst durch das Stattfinden eines Ereignisses *hängt von* dem Zustand des Roboters ab, indem sich dieser gerade befindet.
Bei einem Einzelzustand können unterschiedliche Ereignisse zu unterschiedlichen Aktionen und Übergängen führen. Im nachfolgenden Diagramm wird dies erklärt:



Das Drücken des linken Knopfs während dem Zustand **aus** führt dazu, dass ein grünes Licht eingeschaltet wird und der Zustand wechselt in den Zustand **ein1**, während das Drücken des rechten Knopfes *im selben Zustand* dazu führt, dass eine andere Aktion ausgelöst wird, nämlich dass ein rotes Licht eingeschaltet wird und der Zustand in einen anderen Zustand wechselt, nämlich Zustand **ein2**.

Die Implementierung der Zustandsdiagramme mit Ereignis-Aktions-Paaren

Abbildung 8.1 zeigt die Implementierung des Verhaltens in der obigen Zustandsmaschine als Ereignis-Aktions-Paar. Der linke Kreis im Block ist ausgewählt (und erscheint in rot) um anzugeben, dass es sich um ein Erschütterungs-Ereignis handelt (klopfen).

Der Block Erschütterung/Beschleunigung im fortgeschrittenen Modus

Der Block für diese Ereignisse unterscheidet sich vom Standard-Modus, da er auch für die Ereignisse der Beschleunigung verwendet wird, die in Kapitel 10 beschrieben werden.

(a) Klopfen, um ein - und auszuschalten

(b) Klopfen, um den Zustand zu wechseln

Abbildung 8.1: Klopfen führt zu verschieden Resultaten, abhängig vom Zustand

Im ersten Ereignis-Aktions-Paar (Abbildung 8.1(a)), besteht das Ereignis aus einem Erschütterungsblock und einer Zustandsanzeige  . Eine Zustandsanzeige stellt vier Teile eines Kreises dar, welche entweder ein- oder ausgeschaltet sein können. Eingeschaltet wird durch orange Farbe angezeigt, ausgeschaltet durch weisse. Im vorliegenden Programm verwenden wir das Viertel oben links um anzugeben, ob die oberen Lichter des Roboters ein- oder ausgeschaltet sind.

Im Bild Abbildung 8.1(a) ist dieses Viertel weiss was bedeutet, dass das Licht ausgeschaltet ist. Zusammen mit dem Erschütterungs-Ereignis-Block bedeutet dies also, dass **wenn** der Roboter angeklopft wird **und** die oberen Lichter ausgeschaltet sind, **dann** schalte das Licht ein.

Im zweiten Ereignis-Aktions-Paar (Abbildung 8.1(b)) ist das Viertel orange eingefärbt, was bedeutet, dass das Licht eingeschaltet ist. Zusammen mit dem Erschütterungs-Ereignis-Block bedeutet dies, dass **wenn** der Roboter angeklopft wird **und** das Licht eingeschaltet ist, **dann** schalte das Licht aus.

Wenn Sie nochmals das Zustandsdiagramm betrachten, sehen Sie, dass erst die Hälfte der Arbeit erledigt ist. Wenn wir das Licht ein- und ausschalten müssen wir ebenfalls den Zustand ändern: von **aus** auf **ein** oder von **ein** auf **aus**. Um dies zu tun müssen wir bei jedem Paar einen *Zustands*-Aktionsblock hinzufügen (rechts):  . Mit diesem Block kann man den Zustand für alle vier Viertel auf weiss oder orange ändern.

Zusammenfassend bedeutet das Programm in Abbildung 8.1 also folgendes:

> *Wenn* der Roboter angeklopft wird *und* der Zustand **aus** ist,
> wechsle den Zustand auf **ein** *und* schalte das obere Licht **ein**.
>
> *Wenn* der Roboter angeklopft wird *und* der Zustand **ein** ist,
> wechsle den Zustand auf **aus** *und* schalte das obere Licht **aus**.

Jedes Ereignis führt also sowohl zu einer Aktion, als auch zu einer Änderung des Zustandes des Roboters. Die Aktion hängt dabei vom *aktuellen* Zustand des Roboters ab.

Wie viele Zustände stehen zur Verfügung?

Der Zustandsblock (sowohl als Ereignis als auch als Aktion), kann folgende Farben / Bedeutungen haben:

- **Weiss**: das Viertel ist *ausgeschaltet*;

- **Orange**: das Viertel ist *eingeschaltet*;
- **Grau**: das Viertel wird nicht berücksichtigt.

Ein Beispiel:  ; das obere linke und das untere rechte Viertel sind eingeschaltet, das obere rechte Viertel ist ausgeschaltet und das untere linke Viertel wird nicht berücksichtigt. Bringt man den Zustandsblock  mit einem Ereignis in Verbindung, bedeutet das, dass das Ereignis eintrifft wenn entweder

 oder 

gegeben sind.

Da jedes der vier Viertel ein- oder ausgeschaltet sein können, gibt es 16 Zustände: $2 \times 2 \times 2 \times 2 = 16$

(aus, aus, aus, aus), (aus, aus, aus, ein), (aus, aus, ein, aus),
…
(ein, ein, aus, ein), (ein, ein, ein, aus), (ein, ein, ein, ein).

Abbildung 8.2(a) zeigt alle 16 möglichen Zustände grafisch auf.

Wichtiger Hinweis

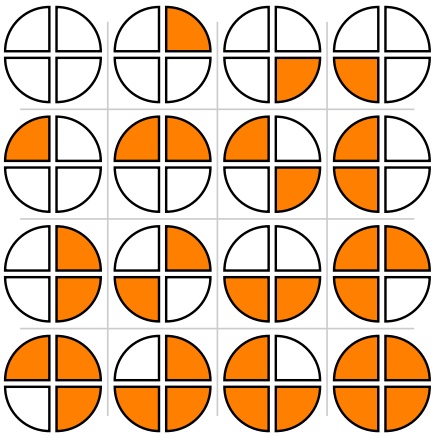
Der aktuellen Zustand des Roboters wird auch im vorderen, oberen Bereich des Roboters angezeigt. Dazu leuchten die diagonal angeordneten Segmente des Lichtkreises entsprechend auf. Abbildung 8.2(b) zeigt als Beispiel den Zustand **(ein, ein, ein, aus)**.

zur Information

Wenn ein Programm gestartet wird, sind alle Zustände ausgeschaltet. Der initiale Zustand ist also **(aus, aus, aus, aus)**:  .

Trick

Falls Sie nicht alle 16 möglichen Zustände verwenden, sondern nur 2 oder 4, dann ist es Ihnen freigestellt, welche Sie verwenden wollen. Wenn Sie andererseits 2 verschiedene Sachen untersuchen wollen, die beide nur 2 verschiedene Werte annehmen können, können Sie für die beiden Sachen zwei unterschiedliche Viertel verwenden. Hier erkennt man, wie nützlich die Möglichkeit ist, Zustände *nicht zu berücksichtigen*.



(a) Alle möglichen Zustände von Thymio

(b) Der Lichterkreis zeigt den Zustand auf

Abbildung 8.2: Thymios Zustände und ihre Darstellung

Fang die Maus

Schreiben Sie nun ein Programm welches das Verhalten einer Katze implementiert, die nach einer Maus sucht: Wenn der mittlere Knopf gedrückt wird, dreht sich der Roboter im Gegenuhrzeigersinn (von rechts nach links), um nach der Maus zu suchen. Sobald der Roboter eine Maus mit dem am weitesten rechts sitzenden Sensor entdeckt, dreht er sich im Uhrzeigersinn (von links nach rechts) so lange, bis der mittlere Sensor die Maus entdeckt; dann hält der Roboter an (siehe (Abbildung 8.3)).

Beispielprogramm **mouse.aesl**

Das folgende Zustandsdiagramm beschreibt das Verhalten des Roboters:

1. Wenn der mittlere Knopf gedrückt wird, kommt der Roboter in den Zustand **links suchen** und bewegt sich von rechts nach links.
2. Wenn der Roboter im Zustand **links suchen** ist und mit dem ganz rechts liegenden Sensor eine Maus entdeckt, wechselt er den Zustand in **rechts suchen** und dreht sich von links nach rechts.
3. Wenn der Roboter im Zustand **rechts suchen** ist und mit dem mittleren Sensor die Maus entdeckt, ändert er in den Zustand **gefunden** und hält an.

Wichtig zu erwähnen ist hierbei, dass wenn die Maus durch den mittleren Sensor entdeckt wird, der Roboter *nur* anhält, *falls* der Roboter sich im Zustand **rechts suchen** befindet. Sonst hält er nicht an (wenn die Maus durch den mittleren Sensor entdeckt wird, sich aber im Zustand **links suchen** befindet).

Abbildung 8.3: Die Roboter-Katze sucht nach der Maus

Wir wollen das beschriebene Verhalten nun Implementieren. Wir stellen den Zustand des Roboters im Viertel oben links dar. Wir wählen weiss für den Zustand **links suchen** und orange für den Zustand **rechts suchen**. Da das Programm beendet wird, wenn man im Zustand **rechts suchen** die Maus entdeckt, müssen wir den Zustand **gefunden** nicht explizit implementieren. Bei der Initialisierung sind alle 4 Viertel **aus** (weiss).

Das folgende Ereignis-Aktions-Paar implementiert das Verhalten in Schritt 1:

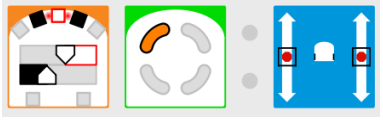
Wenn der mittlere Knopf gedrückt wird, wird der Zustand geändert auf **links suchen** *und* der Robotre dreht sich nach links.

Das Ereignis-Aktions-Paar, das den zweiten Schritt implementiert, sieht so aus:

Wenn die Maus durch den ganz rechts liegenden Sensor entdeckt wird, während man sich im Zustand **links suchen** befindet, dann ändert der Zustand auf **rechts suchen** *und* der Roboter dreht nach rechts.

Das kleine Quadrat neben dem Sensor ganz rechts wird auf schwarz gestellt, damit das Ereignis nur dann eintritt, wenn die Maus ausschliesslich durch den ganz rechts liegenden Sensor entdeckt wird.

Schritt 3 wird durch folgendes Ereignis-Aktions-Paar umgesetzt:



Wenn die Maus durch den mittleren Sensor entdeckt wurde, während der Roboter sich im Zustand **rechts suchen** befindet, dann hält der Roboter an.

Trick

Experimentieren Sie mit der Position der Maus. Ist diese zu nahe am Roboter, werden die Sensoren zu beiden Seiten die Maus ebenfalls erkennen. Das Ereignis findet jedoch nur statt, wenn die äusseren Sensoren die Maus *nicht* erkennen

Aufgabe 8.1

Schreiben Sie ein Programm, das den Roboter tanzen lässt: Der Roboter soll für zwei Sekunden nach links und dann für drei Sekunden nach rechts drehen. Diese Bewegung soll unendlich oft wiederholt werden.

Aufgabe 8.2 (schwierig)

Verändern Sie das Programm ”Einer-Linie-Folgen” aus dem Kapitel 5 so, dass der Roboter nach links dreht, falls dieser die Linie auf die rechte Seite verlässt und nach rechts dreht, falls dieser die Linie auf die linke Seite verlässt.

Kapitel 9

Zählen (Fortgeschrittener Modus)

In diesem Kapitel zeigen wir, wie die Zustände des Thymio Roboters benutzt werden können, um zu zählen und einfache Arithmetik zu betreiben.

Das Design und die Einführung des Projektes wird nicht im Detail präsentiert. Wir nehmen an, dass Sie inzwischen genügend Übung haben, um diese selbständig zu entwickeln. Der Quellcode eines funktionierenden Programms ist zwar im Archiv abgelegt, dies sollten Sie aber erst konsultieren, falls Sie wirklich auf ein Problem stossen, das Sie nicht lösen können. Dieses Projekt benutzt das Ereignis  , um den Zustand zu ändern und das voreingestellte Verhalten der LED Lichter, um die Zustände im Kreis anzuzeigen. Natürlich können Sie beide Verhaltensweisen anpassen.

Wichtiger Hinweis

Der Ist-Zustand des Roboter wird im LED Kreis auf der Oberseite des Roboters angezeigt. Abbildung 8.2(b) zeigt den Roboter im Zustand **(ein,ein,ein,ein)**.

Gerade und ungerade Zahlen

Programmieren
Wählen Sie ein Viertel der Zustandsanzeige. Es wird **aus** (weiss) anzeigen, wenn Sie eine gerade Anzahl Mal in die Hände klatschen und **ein** (orange) anzeigen, wenn Sie eine ungerade Anzahl Mal in die Hände klatschen. Wenn Sie den mittleren Knopf drücken, wird der Zustand auf gerade zurückgestellt (da 0 ja ebenfalls eine gerade Zahl ist).

Beispielprogramm **count-to-two.aesl**

Unsere Methode zu zählen (gerade und ungerade), zeigt das Konzept der *Modulo-2-Arithmetik* auf. Dabei zählen wir von 0 (gerade) auf 1 (ungerade) und dann zurück zu 0. Es handelt sich um die Ganzzahldivision (teilen mit Rest); die Operation modulo wird in der Regel mit ”mod” abgekürzt. Der Ausdruck *Modulo* entspricht dem Ausdruck *Rest*: Falls 7 Mal in die Hände geklatscht wurde, wird die 7 durch 2 geteilt, dies ergibt 3 Rest 1 (7 mod 2 = 3 Rest 1). Nun merken wir uns nur den Rest 1.

Ein anderer Ausdruck für das selbe Konzept ist die *zyklische Arithmetik*. Statt von 0 auf 1 und dann von 1 auf 2 zu zählen, wird im *Kreis* wieder zurück zum Anfang gezählt: 0, 1, 0, 1, ...

Diese Konzepte sind uns aus dem Alltag sehr vertraut, da sie auch bei der Zeitmessung verwendet werden. Minuten und Sekunden werden mod 60 gezählt und Stunden mod 12 oder mod 24. Deshalb bezeichnen wir die Sekunde nach der 59. Sekunde nicht mit 60 sondern mit 0. Wir zählen im Kreis und beginnen wieder bei 0. Ähnlich die Stunden: Nach 23 kommt nicht 24, sondern 0. Wenn es 23:00 Uhr ist und wir machen ab, uns in 3 Stunden zu treffen, dann ist die abgemachte Zeit für das Treffen 23+3 = 26 mod 24, was 02:00 Uhr in der Nacht ist.

Unäres Zählen

> Ändern Sie das Programm, um mit mod 4 zu zählen. Es gibt vier mögliche Reste: 0, 1, 2, 3. Wählen Sie drei der Zustands-Viertel, wovon jeder einen Rest-Wert repräsentiert, also 1, 2 oder 3; der Wert 0 soll durch den Zustand ”alle Viertel” auf **aus** repräsentiert werden.

Diese Methode der Repräsentation von Zahlen wird *unäre Repräsentation* genannt: die Zahlen werden durch verschiedenen Elemente dargestellt, denen verschiedene Zustände zugeordnet sind. Wir benützen die unäre Repräsentation zum Beispiel im Alltag, wenn wir etwas während eines Vorgangs immer weiter zählen (z.B. wie viele Drehungen hat der Roboter gemacht). Zum Beispiel:

 bedeutet 6.

Beispielprogramm **count-to-four.aesl**

Aufgabe 9.1

Wie hoch können wir mit Thymio zählen, wenn wir die unäre Repräsentation verwenden?

Binäres Zählen

Wir sind vertraut mit verschiedenen *Zahlensystemen* (oder Stellenwertsystemen), speziell mit dem Dezimalsystem (Basis 10). Die Symbole 256 im Dezimalsystem repräsentieren nicht drei verschiedene Objekte. Stattdessen repräsentiert die 6 die Einer, die 5 die Zehner und die 2 die Hunderter. Erst durch das Hinzufügen dieser Faktoren erhalten wir die Zahl ”Zweihundertsechsunfüngzig” ($2\times100 + 5\times10 + 6\times1 = 256$). Indem wir die Basis 10 verwenden, können wir sehr grosse Zahlen in einer kompakten Darstellung darstellen. Zudem ist die Arithmetik mit grossen Zahlen relativ einfach; man benutzt die Methoden aus der Schule.

Wir benutzen die 10er Repräsentation, weil wir 10 Finger haben. So ist es einfach diese Repräsentation zu lernen. Computer dagegen haben nur zwei “Finger” (**aus** und **ein**). Deshalb wird die binäre oder ”Basis-2”-Arithmetik beim Programmieren verwendet. Diese Arithmetik ist uns anfangs fremd, weil ähnliche Symbole 0 und 1 wie im Dezimalsystem zur Basis 10 benutzt werden. Die Regel beim zählen wird zyklisch bei der 2 und nicht erst bei der 10:

$$0, 1, 10, 11, 100, 101, 110, 111, 1000, \ldots$$

Wenn man die Binärzahl 1101 betrachtet, lesen wir sie von rechts nach links. Die erste Zahl ganz rechts repräsentiert die Zahl der Einer (1), die nächste Zahl Zweier (2 = 1×2), danach die Vierer, da 1×2×2 = 4 und die Zahl ganz links die 1×2×2×2=8 also Achter. 1101 repräsentiert deshalb je einen Einer, Vierer und Achter aber keinen Zweier oder die Zahl 1+0+4+8, was im Dezimalsystem Dreizehn.

Programmieren
Verändern Sie das Programm mit dem Sie mod 4 gezählt haben in eine binäre Repräsentation.

Beispielprogramm **count-to-four-binary.aesl**

Wir benötigen nur zwei der Zustands-Viertel, um die Zahlen 0–3 mit binären Zahlen auszudrücken. Legen wir fest, dass das Viertel rechts oben die Einer repräsentiert, **aus** (weiss) für Null und **ein** (orange) für Eins und das Viertel links oben repräsentiert die Zweier. So repräsentiert zum Beispiel der Zustand die Zahl 1 und der Zustand die Zahl 2. Falls beide Viertel weiss sind, repräsentiert der Zustand die Zahl 0 und wenn beide Viertel orange sind, repräsentieren die Zustände die Zahl 3.

Es gibt vier Übergänge (Zustandsdiagramm): $0 \to 1, 1 \to 2, 2 \to 3, 3 \to 0$, so werden vier Ereignis-Aktions-Paare benötigt und zusätzlich noch ein Paar, um das Programm in den Anfangszustand zurückzuversetzen, wenn der mittlere Knopf gedrückt wird.

Unbenutzte Vierter ignorieren

Die zwei unteren Viertel werden nicht benutzt, deshalb bleiben sie grau und werden durch das Programm ignoriert.

Aufgabe 9.2

Erweitere das Programm, dass es in mod 8 zählt. Das untere linke Viertel repräsentiert die Anzahl der 4er.

Aufgabe 9.3

Wie hoch können wir mit dem Thymio zählen, wenn wir binäre Repräsentation verwenden?

Addition und Subtraktion

Ein Programm zu schreiben, um bis 8 zählen zu können ist ziemlich mühsam, da wir 8 Ereignis-Aktions-Paare programmieren müssen, eines für jede Umschaltung von n nach $n+1$ (modulo 8). Natürlich zählen wir nicht so in einem Basiszahlensystem, stattdessen haben wir Methoden um Additionen auszuführen, indem Ziffern an gleicher Stelle addiert werden und allenfalls ein Übertrag eingerichtet wird. Im Zahlensystem mit der Basis 10 wird dies so dargestellt:

$$\begin{array}{r} 387 \\ +426 \\ \hline 813 \end{array}$$

und auf die gleiche Weise im Zahlensystem mit Basis 2:

$$\begin{array}{r} 0011 \\ +1011 \\ \hline 1110 \end{array}$$

Wenn wir 1 und 1 addieren, ergibt das nicht 2 sondern 10. Die 0 wird in der selben Spalte geschrieben und die 1 wird in die links davon liegende Spalte übertragen. Das Beispiel oben zeigt die Addition von 3 =(0011) und 11(=1011) was 14 ergibt (=1110).

> **Programmieren**
> Schreiben Sie ein Programm, das mit der Repräsentation von 0 beginnt. Jedes Mal, wenn man in die Hände klatscht, zählt das Programm 1 zur Zahl hinzu. Die Addition sei mod 16, deshalb ergibt 15 addiert um 1 gleich 16 mod 16 = 0.

Beispielprogramm **addition.aesl**

Anleitung:

- Beginnen Sie im Viertel oben rechts und gehen Sie im Gegenuhrzeigersinn weiter. Die Viertel repräsentieren die Anzahl der Einer, Zweier, Vierer und Achter in der gegebenen Zahl. Das Viertel unten rechts wird also gebraucht, um die 8er zu repräsentieren.
- Wenn das rechte obere Viertel die 1er repräsentiert und 0 (weiss) anzeigt, ändere dies einfach auf 1 (orange). Mach dies egal was die anderen Viertel anzeigen.
- Wenn das obere rechte Viertel die 1er repräsentiert und 1 (orange) anzeigt, ändere dies auf 0 (weiss) und behalte dann 1. Das gibt drei Ereignis-Aktions Paare, abhängig von der Anordnung des *nächsten* Viertels, das 0 (weiss) anzeigt.
- Wenn alle Viertel 1 anzeigen (orange) wird 15 repräsentiert, wird 1 zu 15 modulo 16 addiert, ergibt dies 0, repräsentiert indem alle Viertel 0 (weiss) anzeigen.

Aufgabe 9.4

Ändern Sie das Programm so, dass es mit 15 beginnt und mit jedem Mal Klatschen 1 subtrahiert bis 0 und dann wieder bei 15 beginnt.

Aufgabe 9.5

Platzieren Sie eine Sequenz mehrerer kurzer schwarzer Isolierband-Streifen auf eine helle Oberfläche (oder helle Streifen auf einer dunklen Oberfläche). Schreiben Sie ein Programm, welches den Roboter vorwärts bewegen und stoppt lässt, sobald er das vierte Klebeband erreicht hat.

Diese Aufgabe ist nicht einfach: Die Klebebandstreifen müssen genügend weit auseinander platziert werden, damit der Roboter sie entdeckt, aber nicht so weit, dass nicht mehr als ein Ereignis pro Streifen stattfindet. Sie müssen allenfalls mit der Geschwindigkeit des Roboters experimentieren.

Kapitel 10

Beschleunigungssensor (Fortgeschrittenen Modus)

Wir kenne alle das Prinzip der *Beschleunigung*, die Rate der Geschwindigkeitszunahme, wenn ein Auto beschleunigt oder abbremst. Ein *Beschleunigungssensor* ist ein Gerät, welches die Beschleunigung misst. Ein Airbag in einem Auto verwendet einen Beschleunigungssensor um festzustellen, ob die Fahrzeuggeschwindigkeit "zu schnell" abnimmt, weil das Auto einen Aufprall hatte. Falls dies festgestellt wird, wird der Airbag gezündet (aufgeblasen).

Thymio besitzt drei Beschleunigungssensoren, für jede Richtung einen: vorwärts / rückwärts, links / rechts, rauf / runter.

Abgesehen von der *Schwerkraft* (Gravitation) ist es schwierig, messbare Beschleunigungen zu erreichen. Die Schwerkraft ist eine Beschleunigung in Richtung des Erdmittelpunktes. In diesem Projekt verwenden wir die Beschleunigungsmesser um den den Winkel zu bestimmen, um den der Roboter gekippt wird.

Es gibt zwei Ereignisse um den Winkel des Roboters relativ zur Erde zu bestimmen:

- : Ein Ereignis tritt ein, wenn der links- / rechts-Winkel des Roboters sich innerhalb des weissen Winkelsegments im Halbkreis befindet; (der Fachausdruck für diese Bewegung um die Längsachse lautet *rollen* oder wanken, engl.: roll)

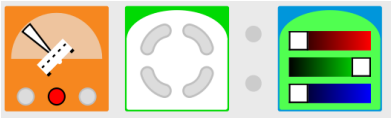
- : Ein Ereignis tritt ein, wenn der vorwärts / rückwärts Winkel des Roboter sich innerhalb des weissen Winkelsegments im Halbkreis befindet; (der Fachausdruck für diese Bewegung um die Querachse lautet *nicken* oder stampfen, engl.: pitch).

Zu Beginn ist das weisse Winkelsegment nach oben ausgerichtet (90°), d.h. das Ereignis tritt ein, wenn der Roboter auf einer Ebene aufrecht steht. Mit der Maus kann dieses Winkelsegment nun zwischen 0° und 90° (links) oder 90 und 180° (rechts) verschoben werden. Im untenstehenden Beispiel tritt das Ereignis ein, wenn der Roboter etwa auf halbem Weg nach links gekippt ist:

Programmierung
Halten Sie den Roboter so, dass er Ihnen zugewandt ist und kippen Sie ihn nach links und nach rechts. Das obere Licht des Roboters wird eine andere Farbe für 5 Bereiche des Neigungswinkels anzeigen.

Beispielprogramm **measure-angles.aesl**

Konstruieren Sie eine Reihe von Ereignis-Aktions-Paaren, wobei jedes Ereignis eine links- / rechts-Beschleunigung sein soll. Tritt das Ereignis ein, soll sich die Farbe des oberen Lichts ändern:



Machen Sie eine Liste für den Zusammenhang von Farben und Winkeln, so dass Sie jede Farbe in einen bestimmten Winkel übersetzen können. Alle Viertel des Ereignis-Zustand-Blocks sind grau, so dass das Ereignis unabhängig vom Zustand eintritt.

Aufgabe 10.1

Können zwei Ereignisse dasselbe Winkelsegment verwenden?
Wie viele verschiedene Winkel können festgestellt werden?

Aufgabe 10.2

Schreiben Sie ein Programm das den Roboter vorwärts fahren lässt, wenn ein Knopf betätigt wird und anhält, wenn er zu kippen beginnt.
Zur Vermeidung von Schäden am Roboter testen Sie das Programm beim Fallen von einer oder zwei Zeitschriften auf den Tisch!

Beispielprogramm **acc-stop.aesl**

Teil II

Parsons Rätsel

Kapitel 11

Parsons-Rätsel für VPL

Was sind Parsons Rätsel?

Parsons Rätsel sind eine spezielle Art von Übungen, die den Lernenden vermitteln, wie man programmiert.[1] Ein Parsons Rätsel besteht aus der Spezifikation eines Programmes, sowie einer Menge von Anweisungen in einer Programmiersprache. Die Aufgabe besteht darin, die Anweisungen in die richtige Reihenfolge zu bringen, so dass ein Programm entsteht das macht, was spezifiziert wurde. Ein Parsons Rätsel kann *Ablenker* (Distraktoren) beinhalten, bei denen es sich um falsche oder überflüssige Anweisungen handelt, die für die Lösung nicht benötigt werden. Der Vorteil eines Parsons Rätsels ist, dass alle zur Lösung nötigen Anweisungen vorliegen und dass ihre Syntax stimmt.

In VPL ist die Reihenfolge der Anweisungen (Ereignis-Aktions-Paare) nicht relevant. Daher werden die Rätsel hier aus Blöcken bestehen, bei denen der Ereignis- oder der Aktions-Block fehlen; ausserdem können auch ganze Blöcke fehlen. Auf der rechten Seite der Ereignis-Aktions-Blöcke stehen jeweils zwei oder mehrere Blöcke, aus denen der richtige Block ausgewählt und mit einem Pfeil mit dem Ereignis-Aktions-Block verbunden werden soll.

Beispiel Wenn der Vorwärts-Knopf gedrückt wird, leuchtet das obere Licht grün.

Die Rätsel

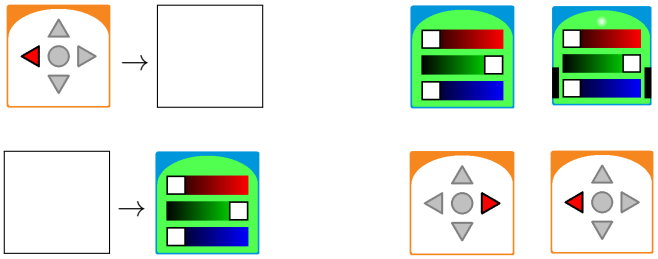
1. Wenn der rechte Knopf gedrückt wird, leuchten die unteren Lichter rot.

[1] Parsons, D. and Haden, "P. Parson's programming puzzles: A fun and effective learning tool for first programming courses." *Proceedings of the 8th Australian Conference on Computing Education*, Darlinghurst, Australia, 2006, Seiten 157–163.

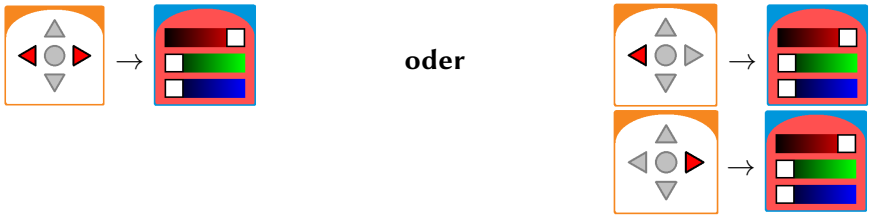
2. Wenn der rechte Knopf gedrückt wird, leuchtet das obere Licht rot.

3. Wenn der linke Knopf gedrückt wird, leuchten die unteren Lichter grün.

4. Wenn der linke **oder** der rechte Knopf gedrückt wird, leuchtet das obere Licht grün.



5. Wenn der linke und der rechte Knopf **gleichzeitig** gedrückt werden, leuchtet das obere Licht rot. Wählen Sie eines der folgenden Programme:

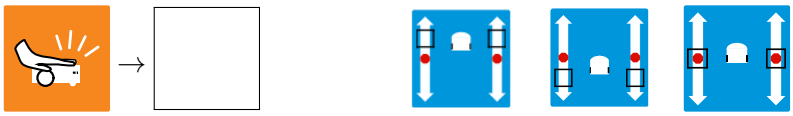


6. Wenn ein Hindernis **ausschliesslich** mit dem am weitesten links liegenden Sensor entdeckt wird, dreht der Roboter nach links.

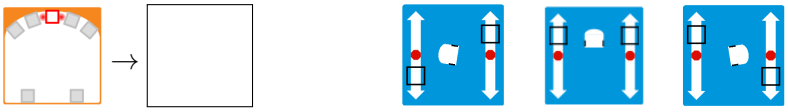
7. Halten Sie den Roboter an, wenn die Tischkante erreicht wurde.

8. Wenn der Roboter eine Wand entdeckt, leuchtet das obere Licht rot.

9. Wenn der Roboter in die Wand gefahren ist, hält er an.

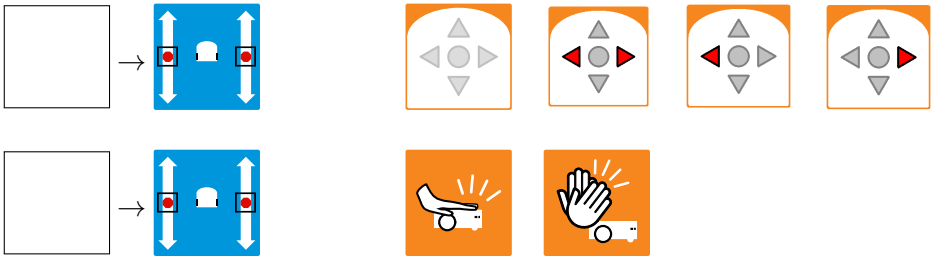


10. Der Roboter dreht nach links, wenn der mittlere vordere Sensor ein Hindernis entdeckt.

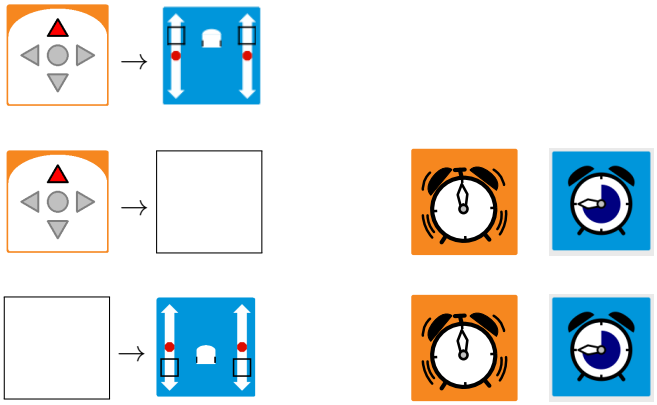


11. Der Roboter dreht nach rechts, wenn der mittlere vordere Sensor **kein** Hindernis entdeckt.

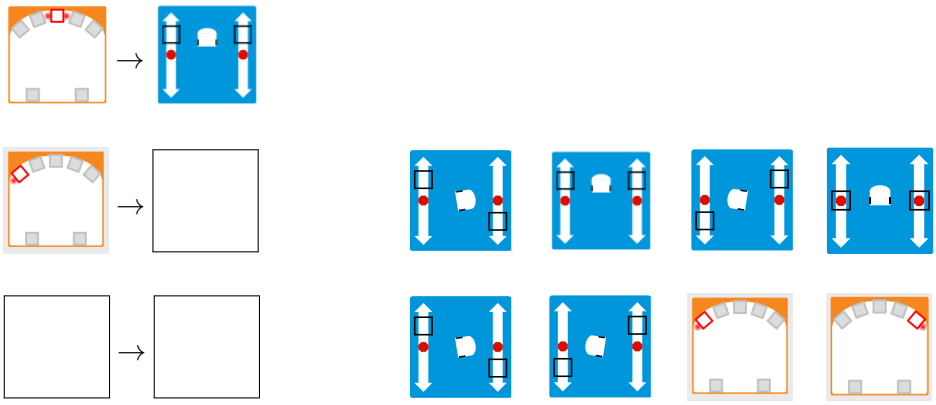
12. Die Motoren werden ausgeschaltet, wenn der linke Knopf gedrückt wird **oder** wenn der Roboter berührt (angetippt) wird.



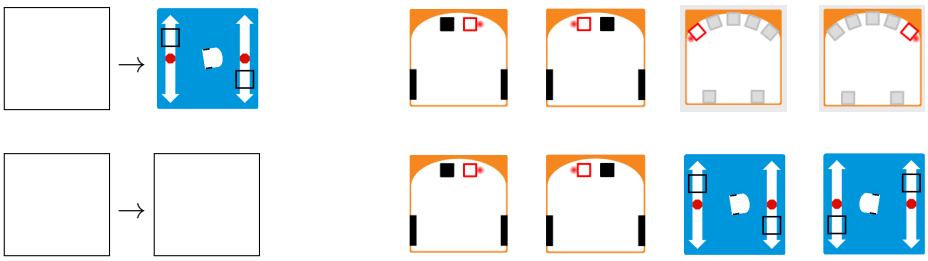
13. Wenn der Vorwärts-Knopf betätigt wird, fährt der Roboter während 3 Sekunden nach vorne und fährt anschliessend rückwärts.



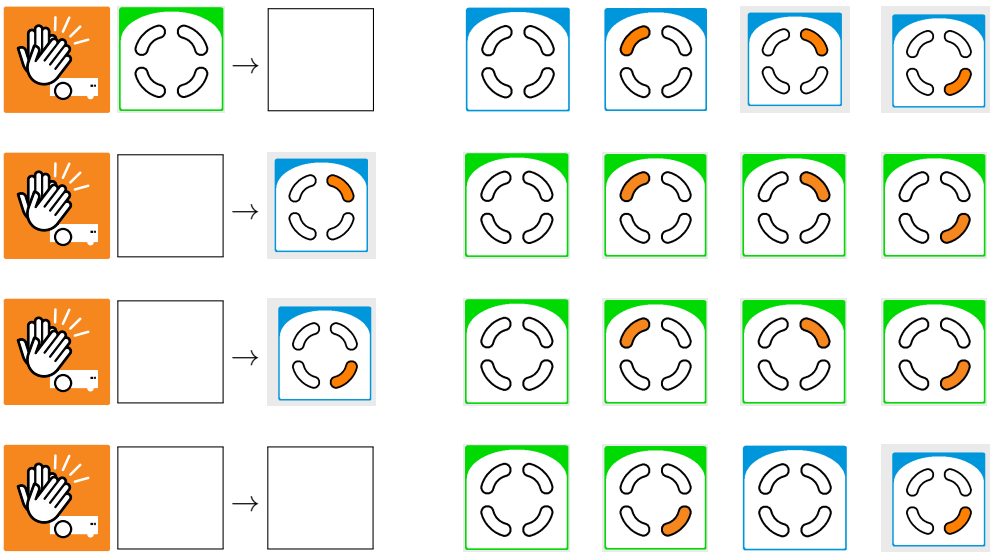
14. Der Roboter fährt auf ein Objekt zu, welches der rechte, mittlere oder linke Sensor entdeckt hat.



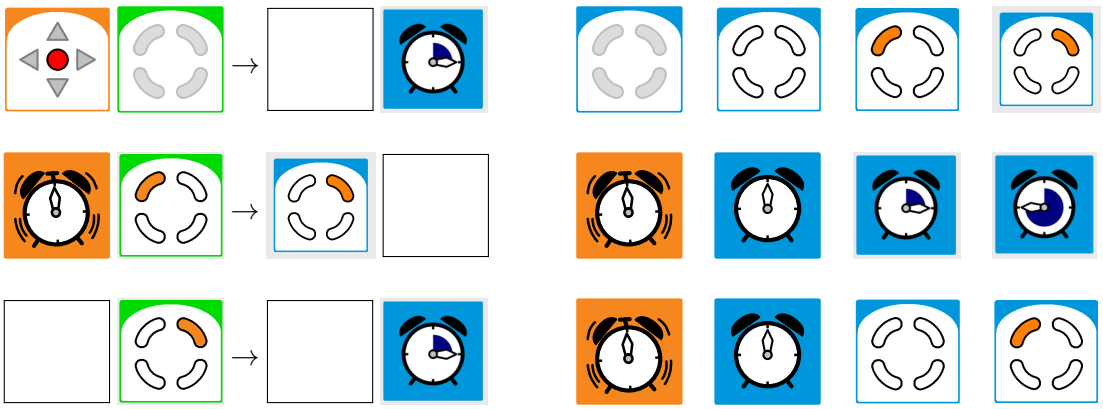
15. Der Roboter folgt einer Linie auf dem Boden. Wird die Linie vom rechten Sensor nicht mehr wahrgenommen, dreht er nach links; wird sie vom linken Sensor nicht mehr wahrgenommen, dreht er nach rechts.



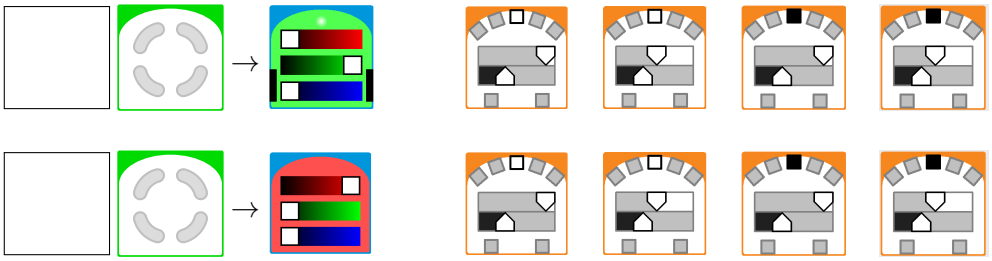
16. Der Roboter zählt 0,1,2,3,0,1,2,3, ..., wenn in die Hände geklatscht wird.



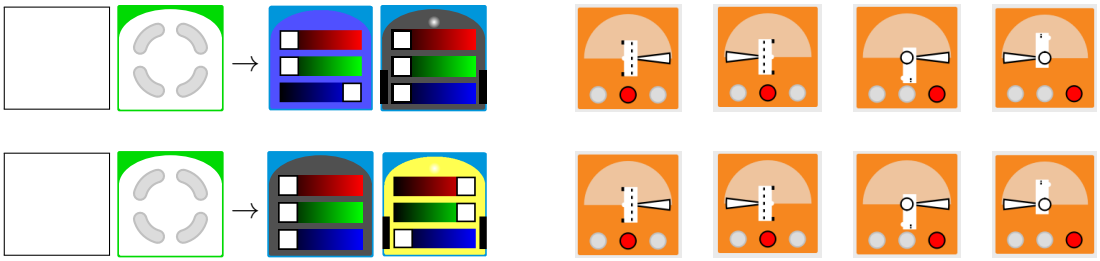
17. Wenn der mittlere Knopf betätigt wird, werden die Lichter vorne rechts und vorne links abwechselnd in einem Sekundenintervall ein- und ausgeschaltet.



18. Die unteren Lichter leuchten grün, wenn in der Ferne ein Objekt entdeckt wird und die oberen Lichter leuchten rot, wenn ein nahes Objekt entdeckt wird.



19. Liegt der Roboter auf seine linke Seite, leuchten die oberen Lichter blau und die unteren Lichter werden ausgeschaltet. Liegt der Roboter auf dem Rücken (Hinterseite) leuchten die unteren Lichter gelb und die oberen Lichter werden ausgeschaltet.



Teil III

Projekte

Kapitel 12

Braitenberg Vehikel

Was sind Braitenberg Vehikel?

Valentin Braitenberg war ein Südtiroler Hirnforscher und Kybernetiker, der ein Buch über das Verhalten von kybernetischen Vehikeln schrieb, welche ein überraschend komplexes Verhalten an den Tag legen.[1] Braitenbergs Vehikel werden oft verwendet wenn es um die Robotik in der Bildung geht. Wissenschaftlich am MIT Media Lab entwickelten Hardware-Implementierungen dieser Vehikel, die sie *Braitenberg Kreaturen* nannten.[2] Diese Vehikel werden unter Verwendung von (angepassten) *programmierbaren Bricks* hergestellt, die eine Vorversion der LEGO Mindstorms Robotik Kits sind.

Dieses Kapitel gibt eine Implementierung für den Thymio Roboter mit VPL; die meisten der Braitenberg Kreaturen aus dem erwähnten MIT-Bericht werden beschrieben. Die MIT-Hardware verwendete Licht- und Berührungs-Sensoren, während der Thymio Roboter sich in erster Linie auf Infrarot-Näherungssensoren stützt. Um die Kreaturen besser mit dem MIT-Bericht vergleichen zu können, wurden die dortigen Namen übernommen, auch wenn diese für die Implementierung für den Thymio manchmal unpassend sind. Auch die Reihenfolge der Darstellung der Vehikel wurde aus dem besagten Bericht übernommen, obwohl dies nicht der Schwierigkeit der Umsetzung in VPL entspricht.

In der Beschreibung meint “erkennt ein Objekt” jeweils, dass der mittlere vordere Sensor ein Objekt entdeckt hat. Am einfachsten simuliert man dies, indem man seine Hand vor den Roboter hinhält.

Der Quellcode in VPL ist im Archiv verfügbar. Die Dateinamen sind dieselben wie die Namen der Kreaturen mit der Endung `aesl`. Für manche Kreaturen wurde zusätzliche Verhaltensweisen vorgeschlagen, deren Implementierung sich ebenfalls im Archiv befindet.

Spezifikation der Kreaturen

Schüchtern Wenn der Roboter kein Objekt erkennen kann, fährt er vorwärts, wenn er ein Objekt entdeckt, hält er an.

Unentschlossen Wenn der Roboter kein Objekt erkennen kann, fährt er vorwärts. Wenn er ein Objekt entdekct, fährt er rückwärts. Beim korrekten Abstand wird der Roboter *oszilieren*, d.h. er wird in rascher Abfolge nach vorn und nach hinten fahren.

[1] V. Braitenberg. *Vehicles: Experiments in Synthetic Psychology* (MIT Press, 1984).

[2] David W. Hogg, Fred Martin, Mitchel Resnick. *Braitenberg Creatures*. MIT Media Laboratory, E&L Memo 13, 1991. http://cosmo.nyu.edu/hogg/lego/braitenberg_vehicles.pdf.

Paranoid Wenn der Roboter ein Objekt entdeckt, fährt er vorwärts. Wenn er kein Objekt entdeckt, dreht er nach links.

Zusatz (Paranoid1) Wenn ein Objekt durch den mittleren Sensor erkannt wird, fährt der Roboter vorwärts. Wenn ein Objekt durch den rechten Sensor erkannt wird (aber nicht durch den mittleren), dann dreht der Roboter nach rechts. Wenn ein Objekt durch den linken Sensor erkannt wird (aber nicht durch den mittleren), dann dreht der Roboter nach links.

Zusatz (Paranoid2, (fortgeschrittener Modus)) Wie in **Paranoid**, aber der Roboter alterniert die Richtung abwechselnd jede Sekunde. **Hinweis**: verwenden Sie Zustände um die Richtung zu bestimmen und einen Timer um die Zustände zu wechseln.

Verbissen Wenn der Roboter ein Objekt vor sich entdeckt, fährt er rückwärts. Wenn er eines hinter sich entdeckt, fährt er vorwärts.

Zusatz (Verbissen1) Wie bei **Verbissen**, aber wenn kein Objekt erkannt wird, hält der Roboter an.

Unsicher Wenn der Roboter mit seinem linken Sensor kein Objekt erkennen kann, schaltet er den rechten Motor ein und den linken aus. Wird mit dem linken Sensor ein Objekt entdeckt, schalte den rechten Motor aus und den linken ein. Der Roboter sollte einer Wand folgen die sich zu seiner Linken befindet. **Hinweis**: Beachten Sie den Hinweis über das drehen des Roboters in Anhang B.

Angetrieben Wenn ein Objekt durch den linken Sensor erfasst wird, schaltet der Roboter den rechten Motor ein und den linken Motor aus. Wenn ein Objekt durch den rechten Sensor erfasst wird, schaltet der Roboter den linken Motor ein und den rechts Motor aus. Der Roboter sollte sich dem Objekt in einem Zickzack-Kurs nähern.

Beharrlich (fortgeschrittener Modus) Der Roboter fährt nach vorne, bis er ein Objekt erkennt. Anschliessend fährt er für eine Sekunde nach hinten, dreht sich und bewegt sich wieder nach vorne.

Anziehung und Abstossung Wenn sich ein Objekt dem Roboter von hinten nähert, entfernt sich der Roboter, bis er ausser Reichweite ist.

Beständig (fortgeschrittener Modus) Der Roboter durchläuft zyklisch vier Zustände, wenn er berührt wird: bewegt sich vorwärts, dann nach links, dann nach rechts, dann rückwärts.

Wild (fortgeschrittener Modus) Das obere Licht blinkt rot. **Tipp**: Sie können die Sensor-Ereignis-Blöcke ohne aktivierte Sensoren verwenden (allen Sensoren ”grau”), wie in Anhang B erklärt wurde.

Zusatz (Wild1, (fortgeschrittener Modus)) Implementieren Sie das blinkende Licht unter Verwendung des Tasten-Ereignis-Blocks statt eines Sensor-Ereignis-Blocks. Gibt es einen Unterschied? Falls ja: was ist die Ursache?

Aufmerksam (fortgeschrittener Modus) Wenn der Roboter mit seinen rechten Sensor ein Objekt entdeckt, schaltet er das obere Licht auf grün. Wenn er das Objekt mit seinem linken Sensor entdeckt, schaltet er das obere Licht auf rot. Nachdem er das Licht eingeschaltet hat, wartet er 3 Sekunden bis er das Licht wieder ausschaltet — während dieser Zeit ändert das Licht nicht.

Kapitel 13

Der Hase und der Fuchs

Dieses Kapitel enthält die Spezifikation eines grösseren Projekts (mein Programm besteht aus 7 Ereignis-Aktions-Paaren mit jeweils 2 bis 3 Aktionen). Sie sollten inzwischen ausreichend Erfahrung haben mit der Gestaltung und Umsetzung von VPL-Programmen, um dieses Projekt selbständig lösen zu können. Nachfolgend wird die Spezifikation gegeben anhand einer Liste von Aufgaben und Verhaltensweisen. Wir schlagen vor, dass Sie die Implementierung Schritt für Schritt vornehmen.

Story[1] Der Roboter ist ein Hase, der durch den Wald läuft. Ein Fuchs jagt den Hasen und will ihn von hinten fangen. Der Hase bemerkt den Fuchs, dreht sich um und fängt den Fuchs.

Spezifikation

Für jedes Ereignis definieren wir eine Farbe für das obere Licht, welche aufleuchtet, wenn das Ereignis eintritt.

1. Berühren des Vorwärts-Knopfes: der Roboter fährt vorwärts (blau).
2. Berühren des Rückwärts-Knopfes: der Roboter hält an (aus).
3. Wenn der Roboter die Tischkante erkennt, hält er an (aus).
4. Wenn der linke hintere Sensor etwas entdeckt, dreht sich der Roboter schnell nach links (im Gegenuhrzeigersinn) bis er das Objekt mit seinem mittleren vorderen Sensor wahr nimmt (rot).
5. Wenn der rechte hintere Sensor etwas entdeckt, dreht sich der Roboter schnell nach rechts (im Uhrzeigersinn) bis er das Objekt mit seinem mittleren vorderen Sensor wahr nimmt (grün).
6. Wenn das Objekt vom mittleren vorderen Sensor entdeckt wird, fährt er für eine Sekunde schnell nach vorne (gelb) und hält dann an (aus).

Beispielprogramm **rabbit-fox.aesl**

[1] Die Geschichte wurde lose inspiriert von einem Witz der bei Doktoranden gut bekannt ist.

Kapitel 14

Barcode-Leser

Barcodes werden in Einkaufsläden und anderswo verwendet, um Objekte zu identifizieren. Die Identifizierung erfolgt über eine Zahl oder eine Folge von Symbolen, die für jeden Objekttyp unterschiedlich sind. Die Identifikation wird für den Zugriff auf eine Datenbank verwendet, die Informationen über das Objekt enthält, wie beispielsweise den Preis. Wir wollen nun mit dem Thymio-Roboter einen Barcode-Leser bauen.

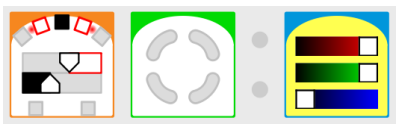
Spezifikation

1. Messen Sie sorgfältig den Abstand zwischen zwei vorderen, horizontalen Sensoren und die Breite dieses Sensors. Fabrizieren Sie dann mit einem dünnen Stück Karton, schwarzem Isolierband und Alufolie einen Barcode wie nachfolgend dargestellt:

2. Jede Anordnung der drei mittleren horizontalen Sensoren repräsentiert einen anderen Code. (Wie viele Codes sind möglich?) Für einige oder alle dieser Codes soll ein Ereignis-Aktions-Paar implementiert werden, welches die obere Farbe je nach Code unterschiedlich anzeigt!

Anleitung:

Wir werden nur die mittleren drei Sensoren verwenden, d.h. die beiden äusseren Sensoren bleiben unberücksichtigt (grau). Für die mittleren Sensoren müssen für die reflektierten Teile entsprechende weisse Sensor-Quadrate verwendet werden, für die schwarzen Stellen schwarze Sensor-Quadrate. Das nachfolgende Ereignis-Aktions-Paar beispielsweise schaltet das obere Licht auf gelb wenn es den `ein-aus-ein` entdeckt:



Das Beispielprogramm im Archiv behandelt alle Barcodes, die an zwei der drei Stellen Folie haben, sowie den Code für keine Folie. Beispielprogramm **barcode.aesl**

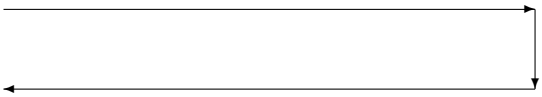
Kapitel 15

Bodenwischer

Sind Sie müde Ihr Haus zu reinigen? Nun gibt es *automatische Staubsauger*, die diese Arbeit für Sie erledigen können! Der Roboter bewegt sich systematisch über den Boden Ihrer Wohnung, weicht Möbeln und anderen Hindernisse aus, und saugt dabei den Staub.

Spezifikation

Wenn der Vorwärts-Knopf gedrückt wird, fährt Thymio von einer Seite des Raumes auf die gegenüberliegende Seite des Raumes, dreht dann, fährt ein wenig weiter und fährt dann weiter auf die ursprüngliche Seite:



Anleitung

Die Lösung sollte aus drei Teilschritten bestehen: (1) lange Fahrt durch den Raum (nach links oder nach rechts), (2) nach rechts drehen (3) kurze Fahrt (ein wenig weiter fahren). Die Teilschritte werden in der folgenden Reihenfolge ausgeführt:

Der Roboter muss Zustände verwenden, um die einzelnen Teilschritte identifizieren zu können. Die Richtung und die Dauer der Fahrt werden durch die Geschwindigkeit des linken und rechten Motors bestimmt, sowie durch die Dauer der jeweiligen Aktion. Daher wird jeder Teilschritt mit einem Ereignis-Aktions-Paar implementiert, wo das Ereignis der Ablauf der vorgängigen Timers ist und wo die Aktion darin besteht, die Parameter für den nächsten Teilschritt festzulegen: (1) Zustand; (2) Geschwindigkeit des linken und rechten Motors; (3) Dauer des Timers. Das Programm starte durch ein Knopf-Ereignis.

Sie werden mit der Dauer und Geschwindigkeit experimentieren müssen, um den gewünschten Pfad (Rechteck) zu erreichen.

Beispielprogramm **sweep.aesl**

Verwenden Sie spasseshalber folgende Farben für die oberen Lichter: grün für die lange Fahrt, gelb für die Drehung und rot für das Anhalten.

Beispielprogramm **sweep1.aesl**

Kapitel 16

Geschwindigkeitsmessung

Spezifikation

Wir wollen die Geschwindigkeit von Thymio messen bei unterschiedlichen Einstellungen für die beiden Motoren. Kleben Sie einen streifen schwarzes Isolierband auf eine weisse Fläche wie wir es bereits beim "Einer-Linie-Folgen" in Kapitel 5 gemacht haben. Stellen Sie den Roboter vor das eine Ende der schwarzen Linie und implementieren Sie folgendes Verhalten:

- Der Roboter fährt geradeaus, sobald der mittlere Knopf betätigt wird.
- Sobald die schwarze Linie durch die Bodensensoren entdeckt wird, starten Sie einen Eine-Sekunde-Timer.
- Wenn der Timer abgelaufen ist, ändern Sie die obere Farbe und starten Sie den Timer wieder für eine Sekunde.
- Wenn das Ende des Isolierbandes erreicht ist, schalten Sie den Motor aus.

Starten Sie das Programm und zählen Sie, wie oft die Farbe ändert. Dies ist die Dauer in Sekunden, die der Roboter für die Strecke benötigt hat. Teilen Sie die Länge der Strecke durch die Anzahl Sekunden und Sie erhalten die Geschwindigkeit. Wenn Sie zum Beispiel eine Isolierband-Strecke von 30cm auslegen und die Farbe 6 Mal wechselt, beträgt die Geschwindigkeit $30 \div 6 = 5$ Zentimeter pro Sekunde.

Versuchen Sie es mit unterschiedlichen Motoreneinstellungen und Isolierbandlöngen!

Anleitung

Erstellen Sie eine Farben-Liste, z.B. 1=rot, 2=blau, 3=grün, 4=gelb, usw. und verwenden Sie die Liste um die Anzahl Sekunden zu bestimmen.

Verwenden Sie Zustände um den Überblick über die aktuelle und die nächste Farben zu behalten. Beispielsweise in Zustand 3, ist die Farbe grün; wenn der Timer abläuft *und* der Zustand 3 ist, dann ändere den Zustand in 4, die Farbe auf gelb und setze den Timer zurück. Es gibt jeweils drei Aktionen für jedes Timer-Ereignis.

Beispielprogramm **measure-speed.aesl**

Kapitel 17

Fangen Sie die Temposünder

Spezifikation

Helfen Sie der Polizei, etwas gegen Raser zu unternehmen. Messen Sie die Geschwindigkeit indem sie feststellen, wie weit ein Fahrzeug während einer festen Zeitperiode fährt.

Der Roboter erkennt ein Objekt, das sich vor den Sensoren von seiner linken zu seiner rechten Seite bewegt. Schalten Sie das obere Licht in einer jeweils unterschiedlichen Farbe ein, je nach dem, wie weit sich das Objekt während einer von seinem am weitesten links liegenden Sensor nach rechts bewegt hat.

Anleitung

- Im initialen Zustand, sobald der linke vordere Sensor das Objekt entdeckt, starten Sei einen Timer für eine Sekunde.
- Wenn der Timer abgelaufen ist, ändern Sie den Zustand auf einen neuen Zustand; wir wollen ihn *Messzustand* nennen.
- Erstellen Sie vier Ereignis-Aktions-Paare, je eines für die 4 weiteren vorderen Sensoren. Das Ereignis tritt nur ein, wenn man sich im Messezustand befindet. Wenn ein Sensor ein Objekt entdeckt, wird das obere Licht eingeschaltet in der Farbe, die dem Sensor zugeordnet ist.
- Stellen Sie sicher, dass das Ereignis-Aktions-Paar nur eintritt, wenn der entsprechende Sensor das Objekt entdeckt, d.h. verhindern Sie, dass das Objekt auch durch die Nachbarsensoren wahrgenommen wird.

Beispielprogramm **speeders.aesl**

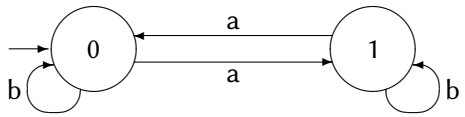
Kapitel 18

Endlicher Automat

Ein *endlicher Automat (EA)*[1] ist ein Modellrechnern oder eine abstrakte Maschine, die Berechnungen durchführen kann. EA sind sehr wichtig in verschiedenen Bereichen der Informatik.[2] Betrachten wir eine endliche Zeichenketten die aus zwei Symbolen bestehen: a und b:

aabbbababbaba

Die Aufgabe besteht darin, eine solche Zeichenfolgen zu lesen und zu entscheiden, ob die Anzahl der a's ungerade ist oder gerade. Ein EA, der dieses Problem löst, hat zwei Zustände: einen Zustand 0, wenn die bisher gelesene Anzahl a's gerade ist und einen Zustand 1, wenn das bisher gelesene Anzahl a's ungerade ist. Der EA wird wie folgt dargestellt (zwei Zustände 0 und 1 sowie vier Übergänge zwischen den Zuständen, die mit a und b beschriftet sind):



Wenn sich der EA in einem Zustand befindet und ein Symbol aus der Zeichenkette liest, wechselt er in einen anderen Zustand, je nach dem, was die Übergänge vorgeben. Wenn die Anzahl der bisher gelesenen Symbole a gerade ist (Zustand 0) und ein weiteres a gelesen wird, nimmt der EA den Übergang zu Zustand 1; umgekehrt falls die bisherige Anzahl ungerade ist und ein a gelesen wird, wird der Übergang von Zustand 1 zu Zustand 0 genommen. Wenn ein b gelesen wird, ändert der Zustand nicht, weil sich dadurch die Anzahl der gelesenen Symbole a nicht ändert.

Der EA ist zu Beginn im Zustand 0, weil die Anzahl der gelesenen Symbole a 0 ist, welches eine gerade Zahl ist. Dieser initiale Zustand wird durch einen kleinen Pfeil dargestellt.

Spezifikation[3]

Drucken Sie die Datei `fa-path-alternate.pdf` aus, welche das folgende Bild enthält [4]

[1] Die Mehrzahl ist *endliche Automaten*, die Abkürzung ist ebenfalls EA.

[2] EA werden formal beschreiben in Lehrbüchern wie demjenigen von J.E. Hopcroft, R. Motwani, J.D. Ullman. *Einführung in Automatentheorie, Formale Sprachen und Berechenbarkeit, Pearson, 2013.*

[3] Diese Spezifikation und ihre Implementierung sind inspiriert durch Lichtmalerei mit Barcodes.

[4] Der Pfad zur Datei befindet sich im Verzeichnis **images** in diesem Archiv.

Die schraffierte Linie wird verwendet, um sicherzustellen, dass sich der Roboter auf der Linie vorwärts bewegt und nicht nach lnks oder rechts abweicht. Die Zeichenfolge wird durch Felder (Quadrate oder Rechtecke) codiert, wobei a für ein schwarzes Feld steht und b für ein weisses. Dieses Bild stellt die Zeichenkette $bababababab$ dar.

Stellen Sie den Roboter an den linken Rand des Bildes vor das erste Feld nach rechts schauend mit dem rechten Bodensensor in der Mitte der schraffierten Linie. Das Verhalten ist wie folgt:

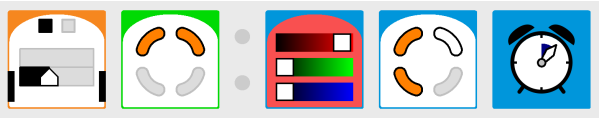
1. Betätigen Sie den vorderen Knopf um den Roboter zu starten. Das obere Licht ist ausgeschaltet, der Zustand wird initialisiert (siehe weiter unten) und der Timer wird gestartet.
2. Betätigen Sie den mittleren Knopf um den Roboter anzuhalten.
3. Der Roboter fährt nach rechts (in Bezug auf das Bild — eigentlich fährt er ja geradeaus). Er benutzt den *rechten* Bodensensor um festzustellen, ob er sich nach links oder rechts bewegt. Wenn der Roboter nach rechts dreht, erkennt der Sensor, dass die Werte kleiner (dunkler) werden und er dreht nach links; wenn er nach links dreht, erkennt der Sensor höhere (hellere) Werte und er dreht nach rechts.
4. Wenn der Timer abgelaufen ist, wird der Wert des *linken* Bodensensors untersucht. Wenn der Roboter über einem weissen Feld ist (was ein b bedeutet), wird das obere Licht auf rot geschaltet und der Timer wird zurückgesetzt. Wenn er über einem schwarzen Feld ist (was ein a bedeutet), wird das obere Licht auf grün geschaltet und der Timer wird zurückgesetzt und der Zustand wird umgeschaltet (von gerade auf ungerade oder von ungerade auf gerade).

Anleitung

Der Roboter benötigt drei der vier Viertel im Zustandsblock für Ereignisse und Aktionen:

- Das Viertel oben links gibt an, ob das Programm läuft oder nicht.
- Das Viertel oben rechts gibt an, dass die Farbe (schwarz oder weiss) gelesen werden soll.
- Das Viertel unten links gibt an, ob die Anzahl der gelesenen Symbole a gerade oder ungerade ist.

Hier ein Beispiel eines Ereignis-Aktions-Paares:



Wenn der linke Sensor wenig Licht erkennt (ein schwarzes Feld) *und*
das Programm läuft *und* die Farbe des Feldes erkannt werden soll*und*
bisher eine gerade Anzahl der Symbole a gelesen wurde, dann
schalte das obere Licht auf rot

wechsle den Zustand so, dass die Anzahl der Symbole a ungerade ist *und*
die Farbe der des Feldes nicht erkannt werden soll
Setze den Timer zurück.

Beispielprogramm **fa.aesl**

Sie werden mit der Fahrgeschwindigkeit und der Dauer des Timers experimentieren müssen, bis die schwarzen und weissen Felder verlässlich erkannt werden.

Übung Drucken Sie die Datei `fa-path-blank.pdf` aus, auf welcher alle Felder weiss sind. Benutzen Sie einen Filzstift um einige Felder schwarz einzufärben und überprüfen Sie Ihr Programm. Überprüfen Sie insbesondere den Fall, wo zwei oder mehr Felder hintereinander schwarz sind, was einer Zeichenkette mit einer Unterfolge von mehr als einem a entspricht.

Zusatz Ändern Sie das Programm so, dass es den Rest 0, 1 oder 2 darstellt, wenn man die Anzahl der Symbole a durch 3 dividiert (mod).

Verändern des Layout

> **Warnung!**
>
> Dieser Abschnitt erklärt, wie man den Pfad (die schraffierte Linie) verändert; Kenntnisse der Textverarbeitungs-Software LaTeX müssen aber vorausgesetzt werden.

Die Dateien `fa-path-*.tex` enthalten den LaTeX Quellcode.

Die folgenden Befehle zeichnen die schraffierte Linie mit einer Breite von 2 cm und einer länge von 23 cm.[5]

```
\shade[left color=black,right color=white] (0,0) rectangle +(2,23);
```

Die schwarzen und weissen Felder werden mit den folgenden Befehlen gezeichnet:

```
\foreach \a in {1, 3, 5, 7}
  \filldraw[color=black] (\offset,\height*\a) rectangle +(\width,\height);

\foreach \a in {0, 2, 4, 6, 8}
  \draw (\offset,\height*\a) rectangle +(\width,\height);
```

Sie können die Liste der Zahlen in der Schleife `foreach` ändern, um anzugeben, welche Felder weiss und welche schwarz sein sollen.

Die Befehle `filldraw` und `draw` verwenden Längenangaben, die man leicht anpassen kann:

```
\setlength{\height}{2.4cm}  % Höhe des Feldes
\setlength{\width}{3cm}     % Breite des Feldes
\setlength{\offset}{2.3cm}  % Abstand zwischen Feld und schraffierter Linie
```

Formatieren Sie die Datei mit ihrem Latex-Programm und drucken Sie die Seite mit einem PDF-Reader aus. Das Bild wird in Hochformat erstellt, man kann die Ausrichtung aber wechseln. Zwei oder mehrere Bilder können auf dem Tisch hintereinander geklebt werden, um eine längere Sequenz zu erhalten, was eine längere Zeichenkette repräsentiert.

[5] Diese Befehle entstammen nicht der Ti*k*Z Grafikbibliothek.

Kapitel 19

Mehrfache Sensorschwellen

Wie im Anhang D beschrieben können im fortgeschrittenen Modus Sensorereignisse auf drei verschiedene Arten abgefragt werden: ein Ereignis tritt ein, wenn das reflektierte Licht unter einem Schwellwert liegt (schwarz), ein Ereignis ritt ein, wenn das reflektierte Licht über einem SChwellwert liegt (weiss mit rotem Rand) und ein Ereignis tritt ein, wenn das reflektierte Licht zwischen zwei Schwellwerten liegt (dunkelgrau):

Spezifikation

Erstellen Sie ein Programm, welches den Roboter dazu bringt, sich einem Objekt zu nähern, wobei er zunächst mit hoher Geschwindigkeit fährt, dann langsamer und kurz vor dem Objekt schliesslich anhält.

Anleitung

- Verwenden Sie drei Ereignis-Aktions-Paare, je eines mit jeder der drei Arten von Sensor-Ereignissen.
- Passen Sie die Schieberegler sorgfältig an, so dass der Schwellwert des einen Sensors dem Schwellwert des nächsten Sensors entspricht (vergleichen Sie Anhang D).
- Fügen Sie eine Farbe für jedes Paar ein, so dass Sie die Geschwindigkeitseinstellungen erkennen können.
- Verwenden Sie reflektierende Streifen wie in Anhang B beschrieben, um die Reichweite zu erhöhen.

Beispielprogramm **slow.aesl**

Spezifikation

Das ”Einer-Linie-Folgen”–Programm, das wir in Kapitel 5 gesehen haben, benutzte zwei Sensoren um zu entscheiden, ob der Roboter sich links oder rechts von der Linie entfernt. Implementieren Sie den nachfolgenden Algorithmus, der nur einen Sensor verwendet!

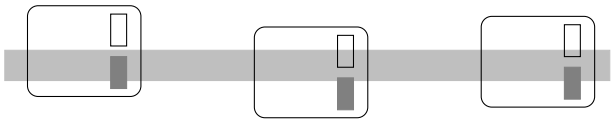
Anleitung

Der Roboter folgt der Linie dem *Rand* entlang und nicht in der Mitte. Die Bodensensoren erhalten eine Reflektion aus einem relativ breiten Bereich, daher kann die Entscheidung gefällt

werden mit einer Mehrfachen Sensorschwelle. Wenn der rechte Sensor verwendet wird, um der rechten Kante einer dunklen Linie zu folgen:

- Wenn der Sensor zu weit links vom Rand ist, wird wenig Licht erkannt.
- Wenn der Sensor zu weit rechts vom Rand ist, wird viel Licht erkannt.
- Wenn der Sensor über der Kante ist, wird Licht erkannt, das irgendwo zwischen den beiden Extremwerten liegt.

Die drei Fälle werden im folgenden Diagramm dargestellt:



Beispielprogramm **line-one.aesl**

Kapitel 20

Mehrere Thymios

Sie können zwei oder mehrere Thymio Roboter gleichzeitig steuern.

Spezifikation

Setzen Sie zwei Roboter T1 und T2 so, dass sie sich gegenüber stehen; T1 ist der Jäger, T2 ist die Beute. Wenn T1 erkennt, dass er nahe bei T2 ist, hält er an; wenn T2 bemerkt, dass er nahe bei T1 ist, zieht er sich zurück bis er T1 nicht mehr erkennt.

Anleitung

- Die Programme für T1 und T2 haben zwei Ereignis-Aktions-Paare: eines, dessen Ereignis das Erkennen eines Objekts mit dem mittleren horizontalen Sensor ist und ein anderes, des Ereignis das Nicht-Erkennen eines Objekts ist. Die Aktionen für T1 und T2 sind allerdings unterschiedlich.
- Verbinden Sie zwei Thymios T1 und T2 mit Ihrem Computer und schalten Sie sie ein. Starten Sie zwei Kopien von VPL. Im Ziel-Auswahl-Fenster (siehe Bild 1.2) sollten beide Roboter T1 und T2 erscheinen; wählen Sie T1 in der einen Kopie von VPL bzw. dem Ziel-Auswahl-Fenster und T2 im anderen. Öffnen und starten Sie in der ersten Kopie von VPL das Programm `chase` (Jäger) und im anderen das Programm `retreat` (Beute).

Experimente

- Was passiert, wenn Sie die Programme austauschen: auf T1 läuft `retreat` und auf T2 läuft `chase`? Erklären Sie!
- Experimentieren Sie im fortgeschrittenen Modus mit unterschiedlichen Werten für die Schwellwerte.

Beispielprogramme **chase.aesl**, **retreat.aesl**

> ⭐ **Kommunikation zwischen Robotern**
>
> Thymio Roboter können sich Meldungen schicken. Diese Funktionalität wird in der AESL-Sprache und -Umgebung unterstützt.

Teil IV

Von der visuellen zur textbasierten Programmierung

Kapitel 21

AESL lernen von VPL Programmen

Herzliche Gratulation! Sie sind ein Experte im Programmieren von Thymio mit der *Visuellen Programmierumgebung (VPL)*. Nun wollen Sie weiter gehen zur professionellen *Studio Programmierumgebung* (Abbildung 21.1) und seiner textbasierten Programmiersprache, der Aseba-Ereignis-Skriptsprache AESL *Aseba Event Scripting Language*.

VPL übersetzt graphische Programme (Ereignis-Aktions-Paare) in ein textbasiertes AESL Programm, welches auf der rechten Seite des VPL-Fensters dargestellt wird (Bereich 6 in der Grafik Abbildung 1.3 auf Seite 10). Im vorliegenden Lernprogramm werden VPL Programme aus den vorherigen Kapiteln verwendet, um die entsprechenden AESL-Programme zu erklären. Sie werden in der Lage sein, Ihre Kenntnis von VPL zu benutzen um die Konzepte von AESL zu erlernen.

Die Programmierung in Aseba Studio basiert wie gehabt auf Ereignissen und Aktionen. Da VPL-Programme in AESL-Programme übersetzt werden, stehen alle VPL-Konstrukte auch in AESL zur Verfügung. Zusätzlich haben Sie die Flexibilität einer normalen Programmiersprache mit Variablen, Ausdrücke und Anweisungen für die Ablaufsteuerung.

Wenn Sie mit Aseba Studio arbeiten, können Sie VPL öffnen, indem Sie den Knopf **Launch VPL** im Bereich *Tools* anwählen (im unteren Bereich des Fensters). Sie können ein VPL-Programm in Aseba Studio importieren, indem Sie einfach das VPL-Dokument öffnen.

Abschnitte die mit * markiert sind, beschreiben Programmierkonzepte, die über das hinausgehen, was in den VPL-Projekte zu finden ist. Sie können diese Abschnitte überspringen, wenn Sie das Lernprogramm zunächst nur durchlesen wollen.

Dokumentation

Um Aseba Studio und AESL zu lernen, gehen Sie auf die Seite *Programming Thymio* page at https://www.thymio.org/de:asebausermanual. Dort finden Sie Unterlagen zu folgenden Themen:

- Die Programmierumgebung.
- Die Programmiersprache AESL.
- Die Schnittstelle des Roboters Thymio. (Es hat auch eine Referenzkarte zu dieser Schnittstelle).
- Die Bibliotheksfunktionen, die von AESL unterstützt werden.

Des weiteren gibt es eine Sammlung von interessanten AESL-Projekten inkl. Quellcode von vorgeschlagenen Lösungen.

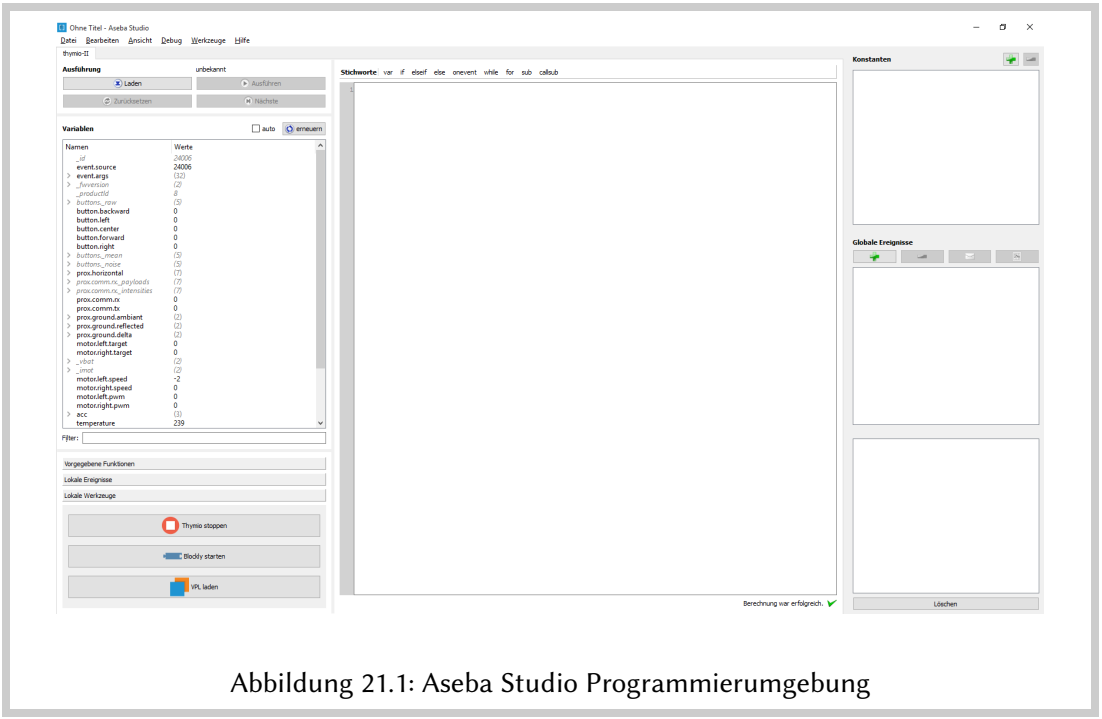
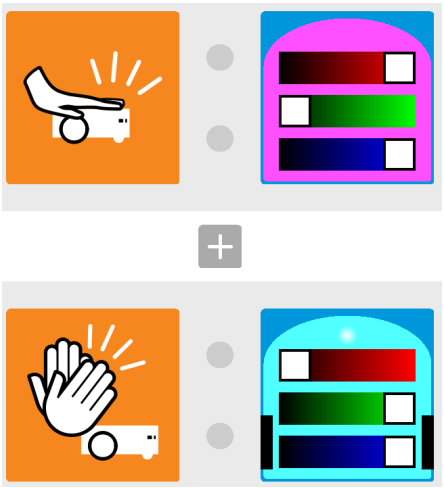


Abbildung 21.1: Aseba Studio Programmierumgebung

Die Schnittstelle von Thymio

Nachfolgend die Darstellung des Programms `whistles.aesl` aus Kapitel 6 mit einem Ausschnitt aus dem dazugehörenden AESL Programm:



```
onevent tap
  call leds.top(32,0,32)

onevent mic
  call leds.bottom.left(0,32,32)
  call leds.bottom.right(0,32,32)
```

Eventhandler (Ereignisbehandlungsroutinen)

Wenn ein Klopfen-Ereignis eintritt, wird das oberen LED-Licht eingeschaltet in der Farbe *Magenta* (Rotblau, Fuchsia, helles Purpur), und wenn ein Klatschen-Ereignis eintritt werden die unteren Lichter in der Farbe *Cyan* (Blaugrün, Türkis) eingeschaltet. Die Ereignis-Aktions-Paare in VPL sind sogenannte *Eventhandler*, die in AESL durch das Schlüsselwort onevent eingeführt werden - man liest das englische Schlüsselwort als zwei Wörter ”on event”. Sie finden eine

Liste aller Ereignisse in der Programmierumgebung Aseba Studio unten links unter "Lokale Ereignisse" aufgelistet (die in den Programmierbereich gezogen werden können).

Die Zeilen, die auf `onevent` folgen, bilden den Körper des Eventhandlers und entsprechen den Aktions-Blöcken im rechten Bereich in VPL.

Wenn ein Klopfen-Ereignis eintritt, wird die *Schnittstellen-Funktion* `leds.top` aufgerufen. Der Aufruf erfolgt mit drei *Parametern*, welche die Intensität der roten, grünen und blauen Anteile des LED-Lichts angeben. Ihr Wert liegt zwischen 0 (ausgeschaltet) und 32 (voll). Die Mischung aus rot und blau ergibt Magenta (Fuchsia).

Das Klatschen-Ereignis in VPL entspricht dem `mic` Ereignis in AESL (mic steht als Abkürzung für microphone). Wenn das Ereignis eintritt, werden die unteren LED-Lichter eingeschaltet. In VPL werden beide unteren LED-Lichter mit einem Aktions-Block eingeschaltet. In AESL können die beiden Lichter unterschiedlich und direkt angesteuert werden. Hier werden beide mit voller Stärke auf grün und blau gesetzt, was Cyan (Türkis) ergibt.

Einen Wert einer Variablen zuordnen

Schauen Sie sich das AESL-Programm nochmals an. Die ersten beiden Zeilen lauten wie folgt:

```
# setup threshold for detecting claps
mic.threshold = 250
```

Eine Zeile, die mit einem **#** beginnt, ist ein *Kommentar*. Kommentare haben keinen Einfluss auf das laufende Programm; sie werden benutzt um dem Leser Informationen weiterzugeben. Der angegebene Kommentar wird vom Übersetzungsprogramm vergeben und bedeutet auf Englisch, dass der Schwellwert für das Klatsch-Ereignis eingestellt werden wird: ein Geräusch, das grösser als *threshold* ist, führt dazu, dass das Ereignis erkannt wird. Die zweite Zeile des Programms spezifiziert nun den Schwellwert mit 250; ausgelöst wird das Ereignis ab einem Wert von 250 - der Wertebereich liegt zwischen 0–255.

In VPL ist der Schwellwert fix vorgegeben, wohingegen man im textbasierten Programm den Wert mit einer einfachen *Zuweisung* ändern kann:

```
mic.threshold = 180
```

Die Bedeutung der Zuweisung ist, dass der Wert der *Wert* auf der rechten Seite des **=** (Gleichheitszeichens) in die *Variable* auf der linken Seite kopiert wird. Die Variable `mic.threshold` ist vordefiniert (muss daher nicht deklariert werden).

Initialisierung des Roboters Thymio

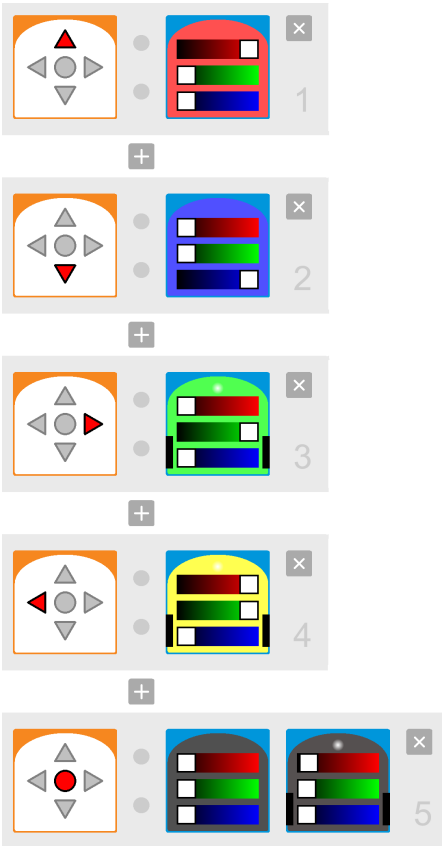
Am Anfang jedes Programms wird eine Reihe von Befehlen eingefügt, die den Sound und alle Lichter aus schalten:

```
# reset outputs
call sound.system(-1)
call leds.top(0,0,0)
call leds.bottom.left(0,0,0)
call leds.bottom.right(0,0,0)
call leds.circle(0,0,0,0,0,0,0,0)
```

Diese *Initialisierung* ist im VPL-Programm nicht zu erkennen. Es wird empfohlen, diese Initialisierung in einem textbasierten Programm zu verwenden, aber dies ist nicht zwingend vorgeschrieben.

Fallunterscheidung (Alternativen)

Das Programm `colors-multiple.aesl` aus Kapitel 2 wechselt die Farben der oberen und unteren Lichter, wenn die Knöpfe berührt werden:



```
onevent buttons
  when button.forward == 1 do
    call leds.top(32,0,0)
  end
  when button.backward == 1 do
    call leds.top(0,0,32)
  end
  when button.right == 1 do
    call leds.bottom.left(0,32,0)
    call leds.bottom.right(0,32,0)
  end
  when button.left == 1 do
    call leds.bottom.left(32,32,0)
    call leds.bottom.right(32,32,0)
  end
  when button.center == 1 do
    call leds.top(0,0,0)
    call leds.bottom.left(7,0,0)
    call leds.bottom.right(7,0,0)
  end
```

Im AESL-Programm tritt ein *einzelnes* Ereignis ein, wenn man einen beliebigen Knopf drückt. Die Aktion des Eventhandlers `onevent buttons` hängt davon ab, welcher Knopf gedrückt wurde. Wir müssen daher die *Variable* `button` untersuchen, um die richtige Aktion auswählen zu können:

```
when button.forward == 1 do
  call leds.top(32,0,0)
end
```

bedeutet *when* (falls) der Wert der Variablen `button.forward` von einem Wert (hier: 0) auf 1 ändert, *dann* wird die beschriebene Aktion ausgeführt, die sich zwischen den Schlüsselwörtern `do` und `end` befindet bzw. befinden. Es gibt fünf `button`-Variablen, eine für jeden Knopf. Der Wert der Variablen ist 1, wenn der Knopf gedrückt wurde und 0 wenn der Knopf losgelassen wurde. Eine oder zwei Aktionen werden ausgeführt, wenn ein `when`-Fall wahr *wird*.

Ein Ereignis oder mehrere Ereignisse*

Die Schnittstelle von Thymio umfasst eigene Ereignisse für jeden Knopf sowie zusätzlich das oben beschriebene `buttons` Ereignis, welches zutrifft, wenn irgend ein Knopf gedrückt oder losgelassen wurde. Wir könnten das Programm daher auch folgendermassen implementieren, unter Verwendung von mehreren Ereignissen und ohne die `when`-Anweisung und button-Variablen:

```
onevent button.forward
  call leds.top(32,0,0)

onevent button.backward
  call leds.top(0,0,32)

onevent button.right
  call leds.bottom.left(0,32,0)
  call leds.bottom.right(0,32,0)

onevent button.left
  call leds.bottom.left(32,32,0)
  call leds.bottom.right(32,32,0)

onevent button.center
  call leds.top(0,0,0)
  call leds.bottom.left(1,0,0)
  call leds.bottom.right(1,0,0)
```

Der Vorteil bei der Verwendung von mehreren Ereignissen ist, dass das Programm leichter lesbar ist, aber es gibt Fälle, wo man das Ereignis `buttons` nehmen muss: (a) um zu unterscheiden, ob der Knopf gedrückt oder losgelassen wird, und (b) um festzustellen, dass zwei Knöpfe gleichzeitig gedrückt wurden:

```
onevent buttons
  # Turn the top LEDs on when the forward button is released
  when button.forward == 0 do
    call leds.top(32,0,0)
  end

  # Turn the bottom LEDs on when
  #   both the left and the right buttons are  touched
  when button.left == 1 and button.right == 1 do
    call leds.bottom.left(0,32,0)
    call leds.bottom.right(0,32,0)
  end
```

Ein weiterer Unterschied ist, dass die individuellen Ereignisse immer eintreten, wenn ein Knopf gedrückt oder losgelassen wird, wohingegen das einfache Ereignis `buttons` in regelmässigen Abständen untersucht wird (mit einer Frequenz von 20 Hz; mehr zu diesem Konzept auf Seite 81).

`if`- und `when`-Anweisung

AESL unterstützt zwei alternative Anweisungen:

```
when v == 1 do ... statements ... end

if v == 1 then ... statements ... end
```

deren Bedeutung sich wie folgt unterscheidet:

> *when* der Wert von v 1 *wird*, führe die Befehle aus
>
> *if* der Wert von v 1 *ist*, führe die Befehle aus

when-Anweisungen werden üblicherweise verwendet bei Variablen, die ein Ereignis darstellen, denn üblicherweise wollen wir Befehle ausführen, wenn etwas ändert und nicht wenn eine Variable einen bestimmten Wert hat. Wir könnten ein `buttons`-Ereignis mit einem Eventhandler implementieren, der `if`-Anweisungen verwendet:

```
onevent buttons
  if button.forward == 1 then
    ... statements ...
  end
```

Allerdings werden in diesem Fall die Befehle mehrfach ausgeführt, wenn wir längere Zeit auf den Knopf drücken. Wenn die Befehle nur dir Farbe ändern, macht es zwar keinen grossen Unterschied, aber es gibt Fälle, wo es einen Unterschied macht und man eine when-Anweisung nehmen muss.

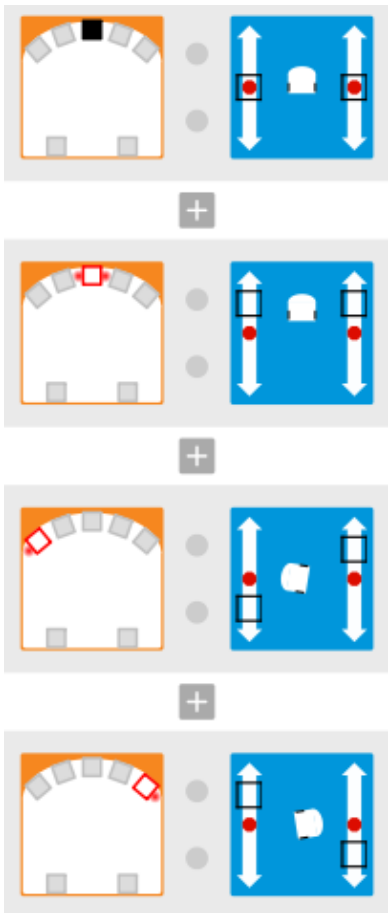
Eine `if`-Anweisung ist geeignet, wenn uns der Wert einer Variable interessiert und nicht ihre Änderung. Die Nachfolgenden Befehle setzen den Wert der Variable `max` auf das Maximum der erhaltenen beiden Sensorwerte der unteren Distanz-Sensoren:

```
if prox.horizontal[5] > prox.horizontal[6] then
  max = prox.horizontal[5]
else
  max = prox.horizontal[6]
end
```

Weitere Beispiele für die `if`-Anweisung finden sich im nächsten Abschnitt und in der Abbildung 21.3.

Arrays (Felder)

Das Programm `likes.aesl` im Kapitel 4 programmiert den Roboter so, dass er Ihrer Hand folgt, wenn man sie vor den horizontalen Distanzsensoren hin und her bewegt. Wenn kein Objekt erkannt wird, hält der Roboter an; wenn ein Objekt vom mittleren Sensor erkannt wird, fährt der Roboter vorwärts; wenn ein Objekt von der äusseren Sensoren links oder rechts erkannt wird, dann dreht er sich in die jeweilige Richtung.



```
onevent prox
  when prox.horizontal[2] < 1000 do
    motor.left.target = 0
    motor.right.target = 0
  end
  when prox.horizontal[2] > 2000 do
    motor.left.target = 300
    motor.right.target = 300
  end
  when prox.horizontal[0] > 2000 do
    motor.left.target = -300
    motor.right.target = 300
  end
  when prox.horizontal[4] > 2000 do
    motor.left.target = 300
    motor.right.target = -300
  end
```

Abbildung 21.2: Das Programm"`likes`" in VPL und AESL

Das AESL-Programm (Abbildung 21.2) besteht aus einem Eventhandler mit mehreren `when`-Anweisungen. Die Werte der Motoren-Variablen `motor.left.target` und `motor.right.target` wurden auf Werte gesetzt, die der Position der Schieberegler im Motoren-Block entspricht. In VPL ändern die Werte in 50er-Schritten, aber in AESL kann man sie auf einen beliebigen Wert zwischen -500 und 500 setzen.

Das Ereignis wird `prox` genannt (kurz für proximity = Nähe). Im Gegensatz zu den Knopf-Ereignissen, welche eintreten, wenn etwas "pasiert", tritt dieses Ereignis *10 Mal pro Sekunde* ein. Bevor das Ereignis eintritt, werden die von den Sensoren gemessenen Werte in die Variable `prox.horizontal` gespeichert. Details hierzu finden sich in der Dokumentation zur Thymio-

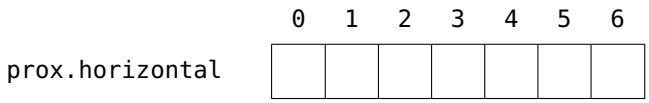
Programmierschnittstelle. [1]

Multiple Variablen in einem Array

Der Roboter Thymio besitzt 7 horizontale Distanz-Sensoren, 5 vorne und 2 hinten. Um die Werte zu lesen, die dieses Sensoren messen, könnte man 7 verschiedene Variablen definieren:

```
prox.horizontal.front.0
prox.horizontal.front.1
prox.horizontal.front.2
prox.horizontal.front.3
prox.horizontal.front.4
prox.horizontal.back.0
prox.horizontal.back.1
```

In AESL kann man stattdessen ein *array* definieren, welches eine Abfolge von Variablen darstellt, die alle denselben Namen haben. Die unterschiedlichen Variablen im Array werden durch eine Zahl identifiziert. Das Array für die horizontale Distanz-Sensoren ist vordefiniert und heisst `prox.horizontal`:



Die ersten 5 Elemente sind für die vorderen Sensoren von links bis recht und die letzten beiden Elemente für die hinteren beiden Sensoren wieder von links nach rechts. Sollten Sie diese Zuordnung vergessen, schauen Sie in der Dokumentation nach oder noch besser auf der Referenzkarte.

Um auf eine Variable in einem Array zugreifen zu können, schreibt man seine Zahl in eckigen Klammern nach dem Variablennamen. Diese Zahl nennt man *Index* des Arrays. Die nachfolgende Befehlsfolge legt fest, dass die Motoren-Variablen auf 300 festgelegt werden, wenn der *mittlere Distanzsensor* (index 2) grösser als 2000 wird:

```
when prox.horizontal[2] > 2000 do
  motor.left.target = 300
  motor.right.target = 300
end
```

Später werden wir sehen, dass Array-Variablen mit einer Zuweisung der folgenden Art festgelegt werden können:

```
timer.period[0] = 1979
```

`for`-Schleifen und Index-Variablen*

Man Verallgemeinert die Verwendung eines Arrays, indem man eine Variable für den Index verwendet, anstelle einer Konstante. [2] Das Programm `cats.aesl` enthält folgende Befehle:

[1] Die Einheit für die *Frequenz*, also die Angabe, wie oft etwas pro Sekunde passiert, nennt man *Hertz*, abgekürzt *Hz*. Das Dokument zur Schnittstelle spezifiziert, dass das `prox`-Ereignis mit einer Frequenz von *10 Hz* eintritt, also alle 100ms.

[2] Bei der Übersetzung des Programms aus VPL in AESL wird dies nicht gemacht, ausser bei einem fortgeschrittenen Konstrukt für Töne; daher wird hier das einfache Beispiel aus dem AESL-Projekt genommen.

```
var i

for i in 0:4 do
  if prox.horizontal[i] > DETECTION then
    state = 2
  end
end
```

Bisher hatten wir nur Variablen benutzt, die bereits in der Schnittstelle von Thymio eingabaut waren; hier nun *deklariert* die erste Zeile eine neue Variable mit Namen `i`. Die nächste Anweisung ist eine `for`-Anweisung mit folgender Bedeutung:

- weise den Wert 0, 1, 2, 3, 4 nachfolgend der Variable `i` zu;
- für jede Zuweisung führe die Befehle zwischen `do` und `end` aus.

Im vorliegenden Fall gibt es nur eine einzelne `if`-Anweisung zwischen `do` und `end`. Dabei wird der Wert der horizontalen Distanzsensoren untersucht und der Wert der Variable `state` wird auf 2 gesetzt, falls der gemessene Wert grösser ist als die Konstante `DETECTION`.[3]

Die Variable `i` bekommt nacheinander die Werte 0, 1, 2, 3 und 4, daher wird jedes Mal, wenn die `if`-Anweisung ausgeführt wird mit `prox.horizontal[i]` ein anderer Sensor untersucht. Das Ergebnis der `for`-Anweisung ist daher dass die Variable `state` auf 2 gesetzt wird, *falls irgendeiner* der vorderen Distanzsensoren ein Objekt erkennt.

Deklarieren eines Arrays

Ein Array wird deklariert, indem man seinen Namen angibt gefolgt von seine Grösse in eckigen Klammern. Bei der Deklaration kann man auch den Wert der Komponenten (Variablen an den einzelnen Indexpositionen) angeben:[4]

```
var state[4]             # Ein Array mit vier Komponenten
var state[] = [0,0,0,0]  # Ein Array mit vier Komponenten vom Wert 0
```

In der Übersetzung des VPL-Programms werden Grösse und initialer Wert der Komponenten festgelegt. Dies funktioniert, so lange die Grösse und die Anzahl Werte übereinstimmen:

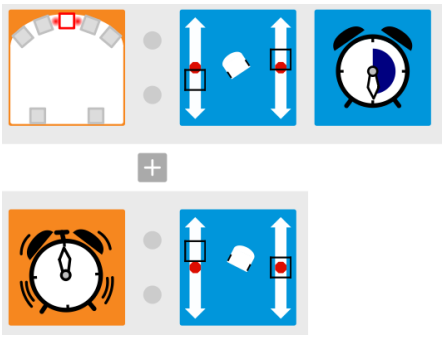
```
var state[4] = [0,0,0,0]  # OK, aber redundant
```

[3] Hinweise zur Definition von Konstanten finden Sie in der Aseba-Studio Dokumentation.

[4] Ein Kommentar muss nicht am Anfang einer Zeile stehen. Jedes Zeichen zwischen dem Zeichen # und dem Ende der Zeile wird durch den Computer ignoriert.

Timer

In Kapitel 7 wurden *Timer* eingeführt. Das Programm `shy.aesl` bringt den Roboter nach links auszuweichen, wenn der mittlere vordere Sensor Ihre Hand entdeckt; zwei Sekunden später weicht er nach rechts aus:



```
onevent prox
  when prox.horizontal[2] > 2000 do
    motor.left.target = -150
    motor.right.target = 100
    timer.period[0] = 2000
  end

onevent timer0
  timer.period[0] = 0
  motor.left.target = 200
  motor.right.target = 0
```

Im Roboter Thymio gibt es zwei Timer. Sie werden den beiden Werten des Arrays `timer.period` zugewiesen. Wenn Sie eine Dauer zuweisen, verwenden Sie entweder die Komponente 0 oder 1 aus dem Array. Der Wert der Dauer wird in *Millisekunden* angegeben, also in Tausendsteln einer Sekunde. Wenn wir eine Dauer von 2 Sekunden im Timer 0 festlegen wollen, schreiben wir daher `timer.period[0] = 2000`.

Es gibt 2 Ereignisse, `timer0` und `timer1`, ein Ereignis für beide Timer. Wenn die Zeit vorbei ist, d.h. wenn der Timer *abgelaufen* ist, tritt das Ereignis ein. Im Eventhandler für das Ereignis `timer0` legen wir die Dauer mit 0 fest, damit das Ereignis nicht nochmal eintritt und die Einstellungen des Motors verändert.

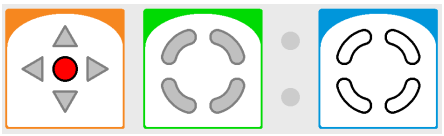
Bei der Initialisierung des Timers wird der Timerwert auf 0 gesetzt, damit die Aktion nicht aus Versehen startet:

```
# stop timer 0
timer.period[0] = 0
```

Zustände

In Kapitel 8 und 9 wurde die Verwendung von *Zuständen* erklärt. Der Roboter Thymio kann sich in einem von 16 Zuständen befinden und man kann spezifizieren, dass ein Ereignis nur dann eine Aktion auslöst, wenn sich der Roboter in einem bestimmten Zustand befindet. Im Programm `count-to-two.aesl` aus Kapitel 9 wird der Zustand auf 0 gesetzt, wenn der mittlere Knopf gedrückt wurde; danach zählt er die Anzahl Geräusche (Klatschen) und gibt en, ob es eine gerade oder ungerade Anzahl ist, indem abwechselnd der Zustand auf 0 und 1 gesetzt wird. Das VPL Programm und seine AESL-Übersetzung für die Berührung des mittleren Knopfes sind:[5]

[5] Das hier abgebildete AESL-Programm unterscheidet sich vom generierten Programm, was später in diesem Kapitel erklärt wird.



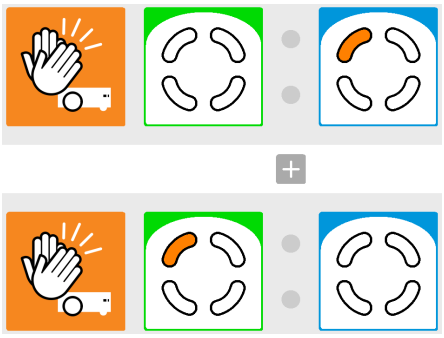
```
var state[] = [0,0,0,0]

onevent buttons
  when button.center == 1 do
    state[0] = 0
    state[1] = 0
    state[2] = 0
    state[3] = 0
  end
```

Der Zustand wird in einem Array `state[]` gespeichert, der 4 Komponenten besitzt. Jede Komponente kann den Wert 0 oder 1 haben, daher gibt es $2 \times 2 \times 2 \times 2 = 16$ mögliche Werte für die Werte des Arrays. Den Komponenten des Arrays werden bei der Initialisierung allen der Wert 0 zugewiesen: `[0,0,0,0]`. Mit der Initialisierung wird gleichzeitig die Grösse des Arrays festgelegt; da es hier 4 Werte sind in `[0,0,0,0]`, gibt es auch 4 Komponenten im Array.

Der Ereignis-Zustandsblock (grün, neben dem Knopf-Ereignisblock) ist dunkelgrau in allen 4 Vierteln; das bedeutet, dass das Ereignis eintreten kann, unabhängig vom Wert der Zustandsvariablen. Daher wird, wenn der mittlere Knopf betätigt wird, die Status-Aktion ausgeführt (blau) was dazu führt, dass alle Komponenten des Arrays `state` auf 0 gesetzt werden, was durch die weissen Viertel angezeigt wird.

Im entsprechenden AESL-Programm untersucht die when-Anweisung ob der mittlere Knopf gedrückt wurde, aber nicht, welche Werte die Komponenten des Arrays `state` hatten. Wenn der Knopf gedrückt wird, werden alle Komponenten auf 0 gesetzt.

Es gibt zwei Klatsch-Ereignis-Blöcke, jeder verbunden mit einem anderen Zustands-Block. Im textbasierten Programm wird ein einzelnen Eventhandler `mic` verwendet. Die Anweisungen, die im jeweiligen Fall zu befolgen sind, hängen von den abgefragten Werten in der `if`-Anweisung ab (Abbildung 21.3).



```
onevent mic
  if state[0] == 0 and
     state[1] == 0 and
     state[2] == 0 and
     state[3] == 0 then
    state[0] = 1
    state[1] = 0
    state[2] = 0
    state[3] = 0
  end
  if state[0] == 1 and
     state[1] == 0 and
     state[2] == 0 and
     state[3] == 0 then
    state[0] = 0
    state[1] = 0
    state[2] = 0
    state[3] = 0
  end
```

Abbildung 21.3: Reaktion auf Klatschen abhängig vom Zustand

Die Bedeutung des Schlüsselwortes and ist, dass *alle* Bedingungen in der `if`-Anweisung

erfüllt sein müssen, damit die Befehle zwischen `then` und `end` ausgeführt werden. Wenn alle Komponenten von `state` den Wert 0 haben, wird der Wert der Komponente `state[0]` auf 1 gesetzt, und die anderen werden auf 0 gesetzt. Dies entspricht dem Zustands-Ereignis-Block mit allen Vierteln auf weiss eingestellt und dem Aktions-Ereignis-Block mit dem Viertel oben links auf orange[6]. In gleicher weise wird der Wert von `state[0]` untersucht: ist er 1 und alle anderen Komponenten 0, dann werden alle Komponenten auf 0 gesetzt.

Unterprogramme

Es kommt relativ häufig vor, dass man dieselbe Sequenz von Anweisungen an vielen Stellen im Programm wieder verwenden muss. Wir können diese Anweisungen ein Mal schreiben und dann x Mal kopieren. Eine einfachere Lösung ist, ein *Unterprogramm* zu verwenden, welches aus einem Namen und einer Sequenz von Anweisungen besteht. In diesem Programm wird die Deklaration `sub display_state` verwendet, um dem Unterprogramm `display_state` eine Sequenz von Befehlen zuzuweisen (hier bestehend aus einem einzigen Befehl auf zwei Zeilen: `call leds.circle`):

```
# Unterprogramm zur Darstellung des aktuellen Zustands
sub display_state
  call leds.circle(
    0, state[1]*32, 0, state[3]*32, 0, state[2]*32, 0, state[0]*32)
```

Wenn das Unterprogramm *aufgerufen* wird, ruft es seinerseits die Anweisungen auf, die bei der Definition des Unterprogramms angegeben wurden:

```
callsub display_state
```

Die Schnittstellen-Funktion `leds.circle` legt die Lichtstärke der abgerundeten acht LED-Lichter fest, die um die Knöpfe herum angebracht sind. Sie müssen die Referenzkarte konsultieren um zu sehen, welcher Parameter welches LED-Licht festlegt!

Die Intensität der einzelnen LED-Lichter wird festgelegt, indem man dem entsprechenden Parameter einen Wert zwischen 0 (ausgeschaltet) und 32 (voll eingeschaltet) zuweist. Das vordere, rechte, hintere und linke LED-Licht wird ausgeschaltet, indem man ihnen den Wert 0 zuweist; den diagonal angeordneten LED-Lichtern wird der Wert des entsprechenden Zustands-Viertels zugewiesen. Dies geschieht, indem man den *arithmetischen Ausdruck* `state[...]*32` verwendet, der den Wert der Komponente im Array mit 32 multipliziert. Wenn der Wert der Zustands-Komponente 0 ist, wird der Parameter 0 übergeben, wenn er 1 ist, wird 32 übergeben.

[6] Man könnte sich auf die Behandlung des ersten Viertels beschränken oder die Zustandswerte, die nicht geändert werden sollen, so belassen, wie sie sind, doch hilft es, Fehler zu vermeiden, wenn man die Werte explizit angibt und ändert.

Eingebaute (mitgelieferte) Funktionen (native functions)

Das obige Programm hat ein Problem. Da den Komponenten des Arrays `state` einer nach dem anderen Werte zugewiesen werden, ist es möglich dass ein anderes Ereignis eintritt wenn einigen aber noch nicht allen Komponenten des Arrays Werte zugewiesen wurden. Um dies zu verhindern und alle Komponenten eines Arrays gleichzeitig Werte zuzuweisen, werden die Werte zunächst einem zweiten Array (`new_state`) zugewiesen und dann wird der zweite in den ersten (`state`) Array kopiert:

```
# variables for state
var state[4] = [0,0,0,0]
var new_state[4] = [0,0,0,0]

onevent buttons
  when button.center == 1 do
    new_state[0] = 0
    new_state[1] = 0
    new_state[2] = 0
    new_state[3] = 0
  end

  call math.copy(state, new_state)
  callsub display_state
```

Die *eingebaute Funktion* `math.copy` wird verwendet um Arrays zu kopieren. Eingebaute oder mitgelieferte Funktionen sind in den Roboter Thymio eingebaut und sind effizienter als Sequenzen von Anweisungen in AESL. Die eingebauten Funktionen sind in der Aseba-Dokumentation beschrieben.

Die aktuelle Version von AESL erlaubt die Zuweisung eines Arrays an einen anderen Array, es wäre also möglich die folgende Zuweisung zu verwenden:[7]

```
  state = new_state
```

[7] Die Zuweisung eines Arrays an einen anderen Array wird in eine Sequenz von Zuweisungen für jede Komponente übersetzt, so dass dies unser Problem nicht lösen würde.

Teil V

Anhänge

Anhang A

Die VPL Benutzer-Schnittstelle

Zuoberst im VPL-Fenster hat es eine Werkzeugleiste (Toolbar):

Neu : Löscht den bisher programmierten Code und stellt eine leere Programmierumgebung dar.

Öffnen : Klicken Sie, um ein bestehendes VPL-Programm zu öffnen. Ein Fenster wird erscheinen und Sie können durch die Ordner navigieren zum Verzeichnis, wo Sie Ihre Programmdateien gespeichert haben (Datei-Erweiterung `aesl`).

Speichern : Speichert das aktuelle Programm. Es ist eine gute Idee, diesen Knopf regelmässig zu betätigen, damit Sie Ihre Arbeit nicht verlieren, wenn ein Fehler auftritt.

Speichern als : Speicher das aktuelle Programm unter einem *anderen Namen*. Benutzen Sie "Speichern als" wenn Sie in einem Programm etwas ausprobieren wollen und das bisherige Programm nicht verlieren wollen.

Zurücksetzen (undo): Mit "Zurücksetzen" oder auch "rückgängig machen" werden getätigte Eingaben rückgängig gemacht (wie z.B. das Löschen eines Ereignis-Aktions-Paares)

Vorwärts machen (redo): Mit "Vorwärts machen" oder "Wiederholen" wird eine Eingabe, die mit "Zurücksetzen" rückgängig gemacht wurde, erneut ausgeführt.

Laden und ausführen (run): Kompiliert das Programm, lädt es auf den Roboter und führt es aus. Die Übertragung geschieht nur, das Programm korrekt kompiliert werden konnte. Wenn Sie das Programm geändert haben, blinkt der Knopf grün um Sie daran zu erinnern, dass Sie das geänderte Programm erneut kompilieren und übertragen müssen.

Halten (stopp): Stoppt das laufende Programm und lässt den Roboter anhalten. Verwenden Sie diesem Knopf um den Roboter anzuhalten, wenn es kein Ereignis-Aktions-Paar gibt, das den Roboter anhalten lässt.

Fortgeschrittener Modus: Der fortgeschrittene Modus aktiviert zusätzliche Elemente und Optionen wie Zustände, Timer, Beschleunigungssensoren und Sensor-Grenzwerte. Der Knopf ändert die Farbe in orange.

Standard-Modus: Der selbe Knopf wird betätigt, um wieder zurück in den Standard-Modus zu wechseln (die Farbe wechselt wieder zurück in blau).

Hilfe: Stellt in einem Internet-Browser-Fenster die VPL-Dokumentation unter folgendem Link dar: https://www.thymio.org/en:thymiovpl. Eine Internetverbindung ist nötig!

Bildschirmfoto: Exportiert ein Bild vom aktuellen VPL-Programm. Sie können dieses Bild dann in ein beliebiges Dokument integrieren. Verschiedene Grafikformate stehen zur Verfügung; SVG ergibt die höchste Qualität, aber PNG ist wohl am weitesten verbreitet.

Anhang B

Zusammenfassung der VPL-Blöcke

Ereignisblöcke

Buttons. Klickt man auf einen oder mehrere der Knöpfe im abgebildeten Block, werden diese rot. Ein Ereignis tritt ein, wenn die roten Knöpfe betätigt werden.

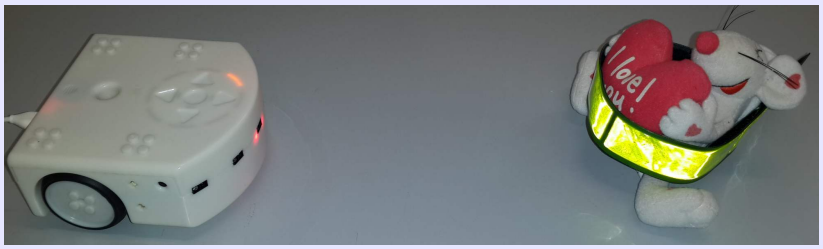
Horizontale Distanzsensoren (fünf vorne und zwei hinten). Klickt man auf eines oder mehrere der kleinen Quadrate im abgebildeten Block, ändern diese ihre Farbe. Zu Beginn sind alle Quadrate grau, was bedeutet, dass die Sensoren ignoriert werden.

Wenn das Quadrat weiss mit rotem Rand ist , tritt ein Ereignis ein, wenn dort viel Licht reflektiert wird.

Wenn das Quadrat schwarz ist , tritt ein Ereignis ein, wenn dort sehr wenig Licht reflektiert wird.

Trick

Gewöhnliche Objekte müssen sehr Nahe sein, um von Thymio erkannt zu werden. Sie können die Reichweite stark ausweiten, wenn Sie *Reflexstreifen* an dem Objekt befestigen, welche man bei Fahrrädern einsetzt.[a]
Vergleichen sie folgendes Bild mit der Abbildung 8.3:



[a] Danke an Francesco Mondada für diesen Tipp!

Bodendistanz-Sensoren (zwei auf der Unterseite von Thymio). Gleiche Verwendung wie die horizontalen Distanzsensoren.

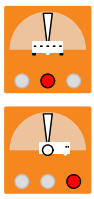
Sensoren im Fortgeschrittenen Modus. Gleiche Verwendung wie bei den vorigen Blöcken. Der obere Schieberegler legt für den rot umrahmten Sensor fest, ab wann das Ereignis eintritt (Grenzwert). Der untere Schieberegler legt für den schwarzen Sensor fest, bis zu welchem Grenzwert das Ereignis des Nicht-Erkennens eintritt.

Es gibt für das Sensorquadrat noch eine weitere Farbe im Fortgeschrittenen-Modus: dunkelgrau; hier sind beide Schieberegler aktiv und man legt den Bereich des das Ereignis auslösenden Bereichs mittels Obergrenze und Untergrenze des Grenzwerts fest.

Klopfen. Ein Ereignis tritt ein, wenn man auf den Thymio klopft.

Klopfen, Fortgeschrittener-Modus. Gleicher Einsatz wie im Standard-Modus. Um die Beschleunigungssensoren zu aktivieren, muss der mittlere oder rechte Kreis ausgewählt werden.

Beschleunigungssensor, Fortgeschrittener Modus. Ziehen Sie das weisse Winkel-Segment im Halbkreis nach links oder rechts. Ein Ereignis tritt ein, wenn der entsprechende Winkel links / rechts bzw. vorwärts / rückwärts mit der Lage des Roboters übereinstimmt.

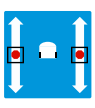


Timer, Fortgeschrittener Modus. Ein Ereignis tritt ein, wenn der Timer bei Null angelangt ist. Der Timer muss durch eine vorgängige Timer-*Aktion* gesetzt werden.

Zustands-Ereignis, Fortgeschrittener Modus. Das Ereignis tritt nur ein, wenn die Komponenten des aktuellen Zustands mit den entsprechenden weissen und orangen Vierteln dieses Blocks übereinstimmen. Die Komponenten, die den grauen Vierteln entsprechen, müssen nicht übereinstimmen.

Aktions-Blöcke

Motoren. Bewegen Sie den linken und rechten Schieberegler nach oben oder unten um die gewünschte Bewegung zu erreichen (links = linkes Rad, rechts = rechts Rad; oben = vorwärts, je höher desto schneller - analog unten für rückwärts).



Obere Lichter. Schieben Sie die drei Schieberegler nach rechts, um den Farbanteil von rot, grün und blau anzupassen.

Untere Lichter. Schaltet die unteren Lichter ein; Verwendung wie beim vorherigen Block.

Musik. Die sechs kleinen Kreise sind Noten. Ein schwarzer Kreis steht für eine kurze Note, ein weisser für eine lange, ein leerer (unsichtbarer) Kreis bedeutet eine Pause. Klicken Sie auf den Kreis, um die Dauer zu ändern, klicken (oder ziehen) Sie auf die Linien, um die Tonhöhe zu ändern.

Timer, Fortgeschrittener Modus Der Timer kann für eine Zeit von bis zu 4 Sekunden eingestellt werden. Klicken Sie innerhalb des weissen Kreises (Uhr), um den Timer zu setzen. Es erscheint dann eine kurze Animation und die Dauer des Timers wird blau dargestellt.

Zustände, Fortgeschrittener Modus Die vier Viertel im Block entsprechen den vier Komponenten des Zustands. Klicken Sie ein Viertel an, um die Farbe auf grau, orange bzw. weiss zu ändern.

Hinweise zu den VPL-Blöcken

Drehen an Ort oder sanfte Wende

Wenn die Motoren bei gleicher Geschwindigkeit in die Gegenrichtung drehen, dreht der Roboter an Ort. Wird nur ein Motor eingeschaltet, fährt der Roboter und wendet sich dabei in Richtung des still stehenden Motors. Um den Bodenwiderstand zu überwinden, müssen Sie möglicherweise eine höheren Leistung einstellen.

Wiederholungs-Geschwindigkeit Sensor-Ereignisse

In den meisten Projekten werden die Sensoren entsprechend eingestellt (weiss, schwarz, dunkelgrau), wodurch diese Sensoren das Ereignis bestimmen und die übrigen werden *ausgefiltert*. Wenn allerdings *alle* Sensorblöcke grau bleiben, werden alle Sensoren ausgefiltert. In diesem Fall tritt das Ereignis 10 Mal pro Sekunde ein, egal welche Werte die Sensoren wahrnehmen würden.

Wiederholungs-Geschwindigkeit Tasten-Ereignisse

Das Gleiche gilt für die Tasten-Ereignisse, nur dass hier das Ereignis 20 Mal pro Sekunde eintritt.

Anhang C

Tipps für die VPL-Programmierung

Erforschen und Experimentieren

Verständnis für jeden Ereignis- und Aktions-Block Für jeden Ereignis-Block und jeden Aktions-Block sollten Sie etwas Zeit einsetzen, um mit den Einstellungen herum zu experimentieren und um festzustellen, wie der Block genau funktioniert. Um das Verhalten eines Aktions-Blockes zu erforschen, erstellen Sie ein Ereignis-Aktions-Paar, wo das Ereignis ein einfaches Knopf-Ereignis ist. Durch das einfache Ereignis kann man leichter lernen, was der Aktions-Block genau macht. Wenn Sie umgekehrt ein Ereignis untersuchen wollen, nehmen Sie dazu einen Aktions-Block, der nur die Farbe des Roboters ändert.

Experimentieren mit den Sensor-Ereignis-Blöcken Die kleinen Lichter neben den Sensoren zeigen, dass der Sensor etwas wahrgenommen hat. Bewegen Sie Ihre Finger vor den Sensoren, um zu sehen, welche lichter an gehen. Konstruieren Sie ein Ereignis-Aktions-Paar bestehend aus dem Sensor-Ereignis und dem oberen Licht und experimentieren Sie mit den Einstellungen der kleinen Quadrate im Ereignis-Block (grau, weiss, schwarz und dunkelgrau im Fortgeschrittenen Modus).

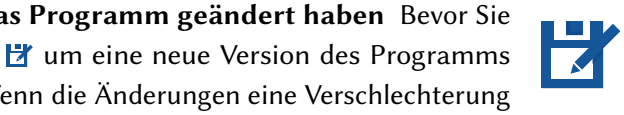
Experimentieren mit den Motoren Experimentieren Sie mit den Einstellungen der Motoren, um zu sehen, wie schnell der Roboter bei unterschiedlichen Einstellungen fährt. Untersuchen Sie das Drehverhalten, indem Sie die Geschwindigkeit (und Richtung) der beiden Motoren variieren.

Konstruieren eines Programms

Programm planen Bevor Sie mit dem Schreiben des Programms beginnen, fertigen Sie ein kurze Beschreibung an, wie das Programm später arbeiten soll: ein Satz pro Ereignis-Aktions-Paar reicht aus.

Konstruieren Sie ein Ereignis-Aktions-Paar nach dem anderen Wenn Sie verstehen, wie jedes Ereignis-Aktions-Paar funktioniert, können Sie diese in das Programm zusammenfügen.

Testen Sie jede Änderung des Programms aus Führen Sie einen Test durch, nach jeder Änderung oder Ergänzung eines Ereignis-Aktions-Paar um einen allfälligen Fehler besser erkennen zu können.

Benutzen Sie Speichen unter nachdem Sie das Programm geändert haben Bevor Sie das Programm verändern, klicken Sie auf  um eine neue Version des Programms unter einem neuen Namen zu speichern. Wenn die Änderungen eine Verschlechterung darstellen, können Sie so einfach auf die frühere Version zurückgehen.

Anzeigen, was passiert Benutzen Sie Farben, um aufzuzeigen, was das Programm gerade macht. Ein Beispiel: wenn ein Sensor in einem Paar den Roboter nach links fahren lässt und ein Sensor in einem anderen Paar den Roboter nach rechts fahren lässt, fügen Sie zu jedem Paar eine Aktion hinzu, die den Roboter jeweils in einer unterschiedlichen Farbe leuchten lässt. So können Sie feststellen, ob es ein Problem mit den Sensoren gibt oder ob die Motoren nicht richtig auf das Ereignis reagieren.

Fehlerbehebung

Verwenden Sie eine glatte Oberfläche Achten Sie darauf, dass die Oberfläche, auf der sich der Roboter bewegt — Tisch oder Boden — sehr sauber und glatt ist. Ansonsten können die Motoren den Roboter nicht richtig bewegen oder die Kurven werden ungenau.

Verwenden Sie ein langes Kabel Achten Sie darauf, ein Kabel zu nehmen, das lange genug ist, oder arbeiten Sie mit der Kabellosen Version. Wenn der Roboter zu weit fährt, kann das Kabel ihn abbremsen oder gar zum stehen bringen.

Sensor-Ereignisse können übersehen werden Sensor-Ereignisse werden pro Sekunde 10 Mal abgefragt. Wenn der Roboter zu schnell fährt, kann ein Ereignis übersehen werden.

Wenn beispielsweise der Roboter die Tischkante erkennen und anhalten soll, er aber zu schnell unterwegs ist, kann er vom Tisch Fallen, bevor er der Sensor ihn zum anhalten bringt. Wenn Sie ein Programm starten, beginnen sie mit tiefer Geschwindigkeit und erhöhen Sie sie nur allmählich.

Für ein weiteres Beispiel denken Sie an das Spur-Folgen-Programm in Kapitel 5. Sein Algorithmus hängt von der Fähigkeit ab, dass der eine Bodensensor die Linie wahrnimmt und der andere sie nicht wahr nimmt. Wenn der Roboter zu schnell unterwegs ist, kann das Ereignis nicht ausgelöst werden, weil die Position für die Sensoren zu schnell verschwindet.

Probleme mit Bodensensoren In Programmen wie dem Spur-Folgen-Programm müssen die Sensoren unterscheiden zwischen viel und wenig reflektiertem Licht. Dies funktioniert nur, wenn der Kontrast zwischen den beiden Farben hoch ist. Wenn Sie beispielsweise eine nicht allzu helle Tischplatte verwenden, sollten Sie weisses Papier als Unterlage verwenden.

Als Alternative können Sie natürlich den Grenzwert im Erweiterten Modus verändern.

Ereignis-Aktions-Paare werden nacheinander ausgeführt Theoretisch werden die Ereignis-Aktions-Paare *gleichzeitig* abgefragt — zur selben Zeit; in der Praxis werden sie aber oft nacheinander abgefragt und das in der Reihenfolge, wie sie programmiert wurden. Wie

in Aufgabe 4.2 gezeigt kann das zu Problemen führen, wenn die zweite Aktion mit der ersten Aktion in Konflikt steht.

Probleme mit dem Klatschen-Ereignis Verwenden Sie das Klatschen-Ereignis *nicht*, wenn die Motoren laufen. Das Geräusch der Motoren kann unerwarteter Weise das Klatschen-Ereignis auslösen.

Des weiteren sollte das Klopfen-Ereignis *nicht* gemeinsam mit dem Klatschen-Ereignis verwendet werden. Das Klopfen erzeugt ein Geräusch, das wieder das Klatschenereignis auslösen kann.

Anhang D

Techniken für den Einsatz der Schieberegler

Einstellen der Schieberegler in den Motorblöcken

Es ist schwierig, die Schieberegler präzise einzustellen, so dass zum Beispiel beide Motoren mit der gleichen Geschwindigkeit laufen. Wir können Präzision erhöhen, wenn wir die Übersetzung der VPL-Ereignis-Aktions-Paare in ein textbasiertes AESL-Programm verwenden.

Trick

Wenn man die Schieberegler im Motorblock verschiebt, stellt man eine Sprunghafte Veränderung der Geschwindigkeit fest (`motor.X.target`); es handelt sich um 50er Schritte im Bereich von -500 to 500. Wenn man vorsichtig schiebt (oder die Cursor-Tasten verwendet) kann man jenden dieser Werte eingeben.

Die nachfolgende Abbildung D.1 zeigt das Programm aus Abbildung 4.4 (wo das Haustier Sie mag und Ihnen folgt), sowie rechts die textbasierte Übersetzung des Programms. Dieser Text wird automatisch verändert, wenn man die Schieberegler betätigt.

`onevent prox` bedeutet: wann immer die Distanzsensoren ausgewertet werden (Distanzsensoren englisch *proximity* sensors, abgekürzt *prox*), werden die nachfolgenden Befehle bis zur Zeile `end` ausgeführt. Die Auswertung erfolgt 10 Mal pro Sekunde.

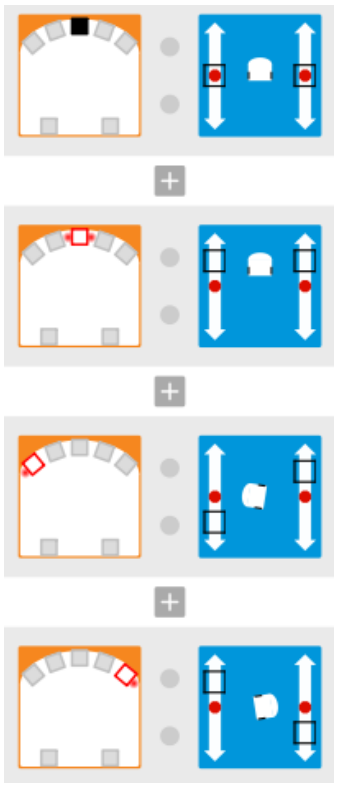
Wenn das Ereignis der Auswertung eintritt, werden die Werte der Sensoren abgefragt und mit einer Selektion ausgewertet. Der Sensor mit der Nummer 2 (vorne in der Mitte) wird als erster untersucht: `prox.horizontal[2]`. Falls dieser Wert unter 1000 liegt, wird die Geschwindigkeit auf 0 gesetzt mit folgenden Befehlen:

```
motor.left.target  = 0
motor.right.target = 0
```

Jeder Block `if ... then ... end` testet einen Sensor und startet die dazu passende Befehlsfolge, falls der Test zutreffend ist. Das Programm folgt dem folgenden Algorithmus:

0. Teste, ob nichts vor dem Roboter ist; falls dies zutrifft, hält er an.
1. Teste, ob etwas vor dem Roboter ist; falls dies zutrifft, fährt er voraus.
2. Teste, ob etwas links vor dem Roboter ist; falls dies zutrifft, dreht er sich nach links.
3. Teste, ob etwas rechts vor dem Roboter ist; falls dies zutrifft, dreht er sich nach rechts.

Sobald die Sensoren gelesen wurden und die passende Befehlsfolge gestartet wurde, wartet das Programm auf das nächste `prox`-Ereignis, um die Tests erneut zu starten. Dies wiederholt sich so lange, bis das Programm angehalten wird.



```
onevent prox
    if prox.horizontal[2] < 1000 then
        motor.left.target = 0
        motor.right.target = 0
    end
    if prox.horizontal[2] > 2000 then
        motor.left.target = 300
        motor.right.target = 300
    end
    if prox.horizontal[0] > 2000 then
        motor.left.target = -300
        motor.right.target = 300
    end
    if prox.horizontal[4] > 2000 then
        motor.left.target = 300
        motor.right.target = -300
    end
```

Abbildung D.1: Ein VPL-Programm mit dem dazugehörenden Text-Programm.

Einstellen der Dauer des Timers

Trick

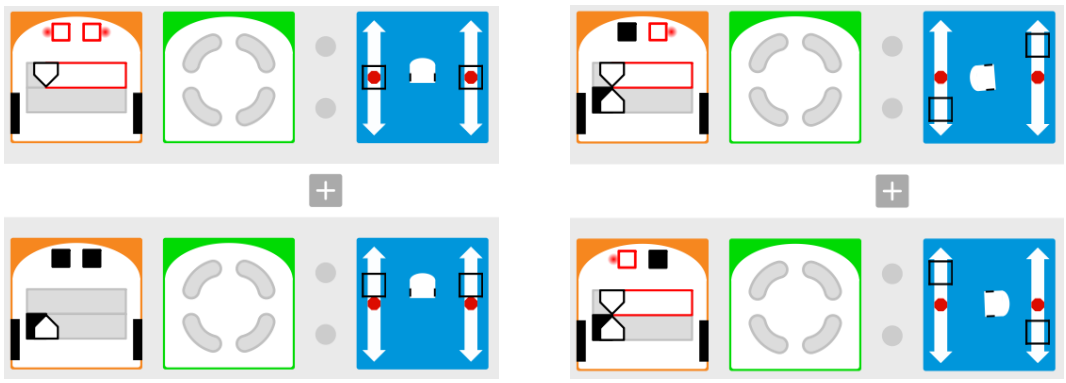
Die Dauer des Timers im Aktions-Block vom Vielfachen von Viertelsekunden (250 ms) bis zu vier Sekunden eingestellt werden.

Sensor-Ereignis-Blöcke im Fortgeschrittenen-Modus

Dieser Abschnitt beschreibt Funktionen der Sensor-Ereignis-Blöcke, die im Fortgeschrittenen-Modus verfügbar sind.

Einstellen der Schwellenwerte der Sensoren

Im Standard-Modus sind die Schwellenwerte der Sensoren nicht veränderbar. Für die horizontalen Distanz-Sensoren bedeutet ein Wert oberhalb von 2000, dass viel Licht reflektiert wird und dass das Ereignis eintritt, falls das entsprechende Quadrat weiss ist. Ein Wert von unter 1000 bedeutet hingegen, dass zu wenig Licht reflektiert wird und ein Ereignis eintritt, wenn



(a) Roboter starten und stoppen (b) Korrektur der Abweichung

Abbildung D.2: Ein Programm zum Folgen einer Linie

das entsprechende Quadrat schwarz ist. Für die unteren Sensoren liegen die Werte bei 450 und 400.

Im Fortgeschrittenen-Modus können die Schwellenwerte festgelegt werden. Mit dem oberen Schieber stellt man den Schwellenwert für ein weisses Ereignis ein und mit dem unteren Schieber die Schwelle, unterhalb derer ein schwarzes Ereignis eintritt:

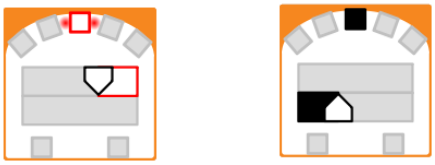
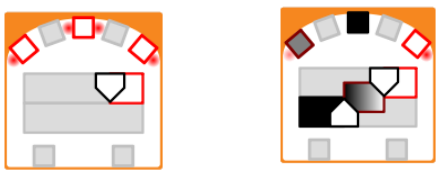


Abbildung D.2 zeigt das Linien-Folgen-Programm (vgl. Abbildung 5.2) im Fortgeschrittenen-Modus. Die Schieberegler wurden so eingestellt, dass der Schwellwert sehr tief ist: 100 für den unteren und den oberen Grenzwert.

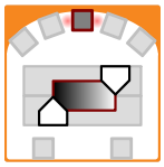
Mehrere Sensoren

Wenn mehrere Sensoren untersucht werden, teilen sie sich die Einstellungen der Schwellwerte:



Ereignisse für Werte zwischen Grenzwerten

Im Fortgeschrittenen-Modus gibt es einen zusätzliches Zustand: das Quadrat kann dunkelgrau sein:



In diesem Zustand tritt ein Ereignis ein, wenn der Wert, der gemessen wurde, zwischen dem oberen und unteren Grenzwert liegt. Der obere Grenzwert wird mit dem oberen Regler festgelegt, der unter mit dem unteren.