Arbeitskreis SMART MACHINES

Hochschule Kaiserslautern University of Applied Sciences

B.O.T. CHALLENGE

SENSOREN UND AKTOREN

Von Thmyio II

INHALT

In diesem Tutorial werden die verschiedenen Sensoren und Aktoren, mit denen der Thymio II ausgestattet ist beschrieben und deren Funktionen erklärt.

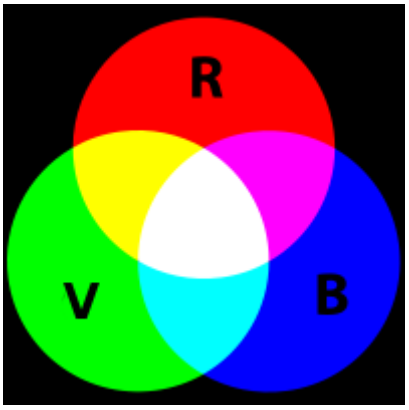
Webseite der B.O.T Challenge

Inhalt

Die LED-Lämpchen 1
Tasten 4
Bedingungen 4
Ereignisse 5
Motoren 5
Sensoren 6
Timer (Zeitnehmer)..... 9
Lesen und Schreiben von Daten mit der SD-Karte 10

Die LED-Lämpchen

Thymio besitzt diverse LED-Lämpchen. Viele sind mit Sensoren verbunden und können deren Aktivität sichtbar machen. Jedes LED hat eine feste Farbe (rot, grün oder blau). Zum Beispiel gibt es vorne rote LED-Lämpchen, oben grüne usw. Wie Du schon weißt, kann man die Farbe des Roboters verändern. Dies ist möglich, weil jeweils drei einzelne LED-Lämpchen (rot, grün, blau) in einer Gruppe montiert wurden. Diese Basisfarben kannst Du zu neuen Farben zusammenfügen. Wenn zum Beispiel das rote und das blaue LED-Lämpchen leuchten, entsteht daraus die Farbe violett. Eine Gruppe bestehend aus drei LED-Primärfarben heißt *RGB LED*. (RGB steht für rot, grün, blau).



Vorgegebene Funktionen erlauben dem Programmierer die Kontrolle über die verschiedenen LED-Lämpchen zu übernehmen.

▪ RGB LED-Lämpchen

Auf dem Roboter gibt es zwei RGB LED-Lämpchen, welche beide zusammen gesteuert werden. Diese LED-Lämpchen zeigen das Verhalten des Roboters an. Es gibt unten noch zwei weitere RGB LED-Lämpchen, welche separat gesteuert werden können.

Standard-Aktivität: Ausgeschaltet, solange der Roboter in Aseba-Modus läuft.

`leds.top(rot, grün, blau)` zeigt die Intensität der obersten LED-Lämpchen an.

`leds.bottom.left(rot, grün, blau)` zeigt die Intensität der LED-Lämpchen von unten links an.

`leds.bottom.right(rot, grün, blau)` zeigt die Intensität der LED-Lämpchen von unten rechts an.

Zum Beispiel für leds.top gibt es drei Parameter, welche jeweils die Helligkeit der einzelnen RGB LED-Lämpchen, also rot, grün und blau, anzeigen. Lämpchen mit

dem Wert 0 sind ausgeschaltet, das Maximum der Helligkeit der Lämpchen ist 32. Alle Werte zwischen 0 und 32 verändern die Helligkeit des Lämpchens. Wenn du also den Wert 16 eingibst, scheint das Lämpchen nur halb so stark wie beim Wert 32.

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

▪ LED-Kreis auf dem Roboter

Auf dem Roboter werden die Tasten von 8 gelben LED-Lämpchen umkreist.

Standard-Aktivität: veranschaulichen die Werte des Beschleunigers. Keines der LED-Lämpchen leuchtet, wenn der Roboter horizontal ist. Wenn der Roboter geneigt ist, zeigt ein einziges LED-Lämpchen den tiefsten Punkt an mit einer Lichtintensität, die sich proportional zum Neigungswinkel verhält.

`leds.circle(led 0, led 1, led 2, led 3, led 4, led 5, led 6, led 7)`, wobei `led 0` das vorderste LED-Lämpchen des Roboters bezeichnet und alle anderen dem Uhrzeigersinn nach nummeriert sind.

▪ LED-Lämpchen der Distanzsensoren

Jeder Distanzsensor ist mit einem roten LED-Lämpchen auf seiner Seite verbunden. Zwei dieser LED-Lämpchen befinden sich vorne im Roboter und jeweils eines auf der Seite.

Standard-Aktivität: Eingeschaltet, wenn ein Objekt nahe am Roboter ist, mit einer Intensität, die sich invers proportional zur Distanz verhält.

`leds.prox.h(led 0, led 1, led 2, led 3, led 4, led 5, led 6, led 7)` definiert die horizontalen LED-Lämpchen vorne und hinten in Thymio. `led 0` bis `led 7` korrespondieren mit den vorderen LED-Lämpchen des Roboters (von rechts nach links), während `led 6` und `led 7` sich auf die LED-Lämpchen hinten links und hinten rechts beziehen.

`leds.prox.v(led 0, led 1)` definiert die LED-Lämpchen, welche mit dem unteren linken bzw. rechten Sensor verbunden sind.

▪ Untere LED-Lämpchen

Zwischen den Tasten auf dem Roboter befinden sich vier rote LED-Lämpchen.

Standard-Aktivität: Für jede Pfeiltaste leuchtet ein LED-Lämpchen auf, wenn die Pfeiltaste gedrückt wird. Sobald der mittlere Knopf gedrückt wird, leuchten alle vier LED-Lämpchen auf.

`leds.buttons(led 0, led 1, led 2, led 3)` kontrolliert diese LED-Lämpchen, wobei `led 0` sich auf das vorderste LED-Lämpchen bezieht und alle anderen im Uhrzeigersinn nummeriert sind.

▪ LED-Lämpchen des Fernsteuerungsempfängers

Dieses LED-Lämpchen befindet sich nahe dem Fernsteuerungsempfänger.

Standard-Aktivität: Blinkt, sobald der Roboter Befehle über den RC5-Fernsteuerungskode erhält.

`leds.rc(led)` kontrolliert dieses LED-Lämpchen

▪ LED-Lämpchen des Temperatursensors

Zwei LED-Lämpchen (eines rot und eines blau) befinden sich nahe des Temperatursensors (NTC Widerstand).

Standard-Aktivität: Rot, wenn die Temperatur über 28°C ist; rot und blau, wenn die Temperaturwerte zwischen 28° und 15° liegen und blau, wenn es kälter als 15°C ist.

`lleds.temperature(red, blue)` kontrolliert das LED-Lämpchen des Temperatursensors.

▪ LED-Lämpchen fürs Mikrofon

Dieses LED-Lämpchen leuchtet blau und befindet sich nahe des Mikrofons.

Standard-Aktivität: Aus.

`leds.sound(led)` kontrolliert das LED-Lämpchen fürs Mikrofon.

Außerdem gibt es noch LED-Lämpchen, welche nicht vom Programmierer kontrolliert werden können:

3 grüne LED-Lämpchen oben auf dem Roboter zeigen den Ladezustand der Batterie an

ein blaues und ein rotes LED-Lämpchen hinten im Roboter zeigen das Aufladen der Batterie an

ein rotes LED-Lämpchen hinten im Roboter zeigt den Status der SD-Karte an

Tasten

Auf dem Roboter Thymio siehst du fünf Tasten. Die vier Pfeilförmigen Tasten sind regelmässig um die mittlere Taste angeordnet. Fünf Variablen entsprechen dem Zustand dieser Tasten (1 = gedrückt, 0 = nicht gedrückt):

`button.backward` : Taste nach hinten

`button.left` : Taste nach links

`button.center` : Mittlere Taste

`button.forward` : Taste nach vorne

`button.right` : Taste nach rechts

Thymio erneuert die Daten dieser Liste mit einer Frequenz von 20 Hz und kreiert das Ereignis *button* nach jeder Erneuerung. Außerdem kann für jede dieser Tasten, jenachdem ob gedrückt oder nicht, ein entsprechendes Ereignis ausgelöst werden.

Bedingungen

Um herauszufinden, welcher Knopf gerade gedrückt wird, kannst du die Variablen in button.*name* im Speicher überprüfen.

In diesem Fall wird die Bedingung, dass ein Knopf gedrückt wird, um das Programm zu starten folgendermaßen geschrieben: Benutze die *Bedingung* if … then … else … end (wenn…dann…sonst…Ende). Dies sagt dem Roboter zum Beispiel WENN der Knopf (Nummer 3) gedrückt ist, DANN schalte das LED-Lämpchen ein. In Aseba Studio wird dies folgendermaßen geschrieben:

```
if button.forward==1 then       #wenn der Knopf nach vorne gedrückt
(1) ist,
    call leds.top(0,0,32)       #dann schalte das blaue LED-Lämpchen
ein
else
    call leds.top(0,0,0)        #sonst schalte alle LED-Lämpchen aus
end
```

Dieser Kode sagt dem Roboter, dass er das blaue LED-Lämpchen einschalten soll, wenn die Bedingung erfüllt ist, dass der Knopf nach vorne (button.forward) gedrückt ist. Falls die Bedingung nicht erfüllt wurde, soll das LED-Lämpchen nicht leuchten.

Wenn du nur diesen Kode eingibst, wirst du bemerken, dass der Kode nur am Anfang des Programms einmal funktioniert. Wenn du aber möchtest, dass das LED-Lämpchen jedes Mal aufleuchtet, wenn du den Knopf drückst, dann muss regelmäßig überprüft werden, ob der Knopf gedrückt ist oder nicht. Dafür brauchst du ein *Ereignis*.

Ereignisse

Im linken Fenster in Aseba Studio siehst du eine Liste der möglichen lokalen Ereignisse.

Das buttons Ereignis entspricht dem Moment, in dem überprüft wird, welche Knöpfe gerade gedrückt sind. Deshalb genügt es, eine einzige Zeile Kode zum bereits geschriebenen Kode hinzuzufügen.

```
onevent buttons                 #Jedes Mal, wenn die Knöpfe gedrückt
sind
if button.forward==1 then       #wenn der Knopf nach vorne gedrückt
(1) ist,
    call leds.top(0,0,32)       #dann schalte das blaue LED-Lämpchen
ein
else
    call leds.top(0,0,0)        #sonst schalte alle LED-Lämpchen aus
end
```

Nun kannst du den Kode testen mit `Laden` und `Ausführen`. Die LED-Lämpchen des Roboters sollten nun immer blau werden, wenn der Knopf nach vorne gedrückt wird und sonst sollte das blaue LED-Lämpchen nicht leuchten.

Es gibt noch viele andere Ereignisse: `Motor` für die Motoren, `acc` für den Beschleuniger usw.

Motoren

Jetzt werden wir den Roboter bewegen. Der Roboter wird die Richtung je nachdem, welche Taste gedrückt wird, ändern.

Für die Motoren gibt es 3 verschiedene Variablen im Speicher.

Die Geschwindigkeit der Motoren kannst du durch diese Variablen ändern, indem du den Text Code veränderst:

`motor.left.target`: gewünschte Geschwindigkeit für das linke Rad

`motor.right.target`: gewünschte Geschwindigkeit für das rechte Rad

Die tatsächliche Geschwindigkeit kann aus diesen Variablen abgelesen werden:

`motor.left.speed` : tatsächliche Geschwindigkeit des linken Rads

`motor.right.speed` : tatsächliche Geschwindigkeit des rechten Rads

Der erste Wert bezieht sich jeweils auf das linke Rad und der zweite Wert auf das rechte Rad. Werte von -500 bis zu 500 sind erlaubt. Ein Wert von 500 entspricht ungefähr einer konstanten Geschwindigkeit von 20 cm/s. Die Werte der Befehle

können aus den Variablen `motor.left.pwm` und `motor.right.pwm` abgelesen werden.

```
onevent buttons                 #Jedes Mal, wenn die Knöpfe gedrückt
sind
if button.forward==1 then       #wenn der Knopf nach vorne gedrückt (1)
ist,
    call leds.top(0,0,32)       #sonst schalte alle LED-Lämpchen aus
else
    call leds.top(0,0,0)        #sonst schalte alle LED-Lämpchen aus
end

if button.forward==1 then
    motor.left.target=200
    motor.right.target=200
end

if button.center==1 then
    motor.left.target=0
    motor.right.target=0
end

if button.backward==1 then
    motor.left.target=-200
    motor.right.target=-200
end

if button.left==1 then
    motor.left.target=-200
    motor.right.target=200
end

if button.right==1 then
    motor.left.target=200
    motor.right.target=-200
end
```

Sensoren

Ein Sensor kann zum Beispiel die Helligkeit des Lichts, die Lautstärke eines Tons, die Geschwindigkeit eines Fahrzeugs oder die Distanz zu einem Hindernis messen.

▪ Distanzsensor

Ein Distanzsensor misst den Abstand zu nahen Hindernissen. Dafür benötigt ein Distanzsensor zwei verschiedene Techniken: einen Infrarotlicht-Strahler und einen Infrarotlicht-Empfänger. Der Infrarotlicht-Strahler sendet rotes Licht aus, welches auch für das menschliche Auge sichtbar ist. Der Infrarotlicht-Empfänger misst das an Hindernissen zurückgestrahlte, reflektierte Licht, welches für das menschliche Auge nicht sichtbar ist. Wenn ein Hindernis sehr nahe ist, wird viel Licht reflektiert. Je weiter weg das Hindernis ist, desto weniger Licht wird zurückgestrahlt. Mit der Messung des zurückgestrahlten Lichts ist es möglich, den Abstand zu Hindernissen zu messen.

Der Roboter Thymio II hat insgesamt neun Distanzsensoren: 5 nach vorne, 2 nach hinten und 2 nach unten.

- Horizontal

`prox.horizontal[0]` : vorne links

`prox.horizontal[1]` : vorne, links Mitte

`prox.horizontal[2]` : vorne, Mitte

`prox.horizontal[3]` : vorne, rechts Mitte

`prox.horizontal[4]` : vorne rechts

`prox.horizontal[5]` : hinten links

`prox.horizontal[6]` : hinten rechts

Die Werte dieser Liste können von 0 (der Roboter "sieht nichts") bis zu mehreren Tausend (der Roboter "sieht" ein sehr nahes Hindernis) variieren. Thymio erneuert diese Werte mit einer Frequenz von 10 Hz und kreiert das Ereignis *prox* nach jeder Erneuerung.

- Boden

`prox.ground.ambiant`: umgebende Lichtintensivität auf dem Boden. Die Werte können von 0 (kein Licht) bis 1023 (maximale Lichtstärke) variieren.

`prox.ground.reflected` : Stärke des reflektierten Lichts, während ein Sensor Infrarotlicht ausstrahlt. Die Werte können variieren von 0 (keine Reflektion) bis 1023 (das Licht wird vollständig zurückgestrahlt)

`prox.ground.delta` : Unterschied zwischen reflektiertem und umgebendem Licht, verbunden mit der Distanz und der Bodenfarbe.

In jeder Liste verweist 0 auf den linken Sensor und 1 auf den rechten Sensor. Wie bei einem Distanzsensor erneuert Thymio diese Werte mit einer Frequenz von 10 Hz.

Die Distanzsensoren, welche den Abstand nach unten messen, können benutzt werden, um die Ecke eines Tisches zu bemerken und den Roboter anzuhalten.

Bevor du nun programmierst, lernen die Distanzsensoren ein wenig besser kennen. Stelle jetzt den Roboter auf den Tisch und klicke immer wieder auf erneuern, um die Werte in prox.ground.delta zu überprüfen.

```
onevent prox
if   prox.ground.delta[0]<300 or prox.ground.delta[1]<300 then
     motor.left.target=0
     motor.right.target=0            #dann schalte die Motoren aus.
     call leds.top(32, 0, 0)
else motor.left.target=200
     motor.right.target=200
     call leds.top(0, 32, 0)
end
```

▪ Lokale Kommunikation

Thymio kann seine horizontalen Infrarot-Abstandssensoren dazu verwenden, um einen Zahlenwert zu anderen Robotern in einem Bereich von etwa 15 cm zu senden (-Peer to Peer-Kommunikation). Dieser Wert wird mit 10 Hz gesendet, während die Abstandssensoren normal weiterarbeiten. Der Thymio sendet einen 11-Bit-Wert (aber zukünftige Firmware könnte eines der Bits für den internen Gebrauch verwenden, es wird empfohlen, innerhalb von 10 Bits zu bleiben). Um die Kommunikation zu verwenden, rufen Sie die Funktion *prox.comm.enable* auf, mit 1 um die Kommunikation zu ermöglichen und mit 0 um sie auszuschalten. Wenn die Kommunikation aktiviert ist, wird der Wert in der Variablen *prox.comm.tx* alle 100 ms gesendet. Wenn Thymio einen Wert empfängt, wird das Ereignis *prox.comm* ausgelöst und der Wert in der Variablen *prox.comm.rx* abgelegt.

▪ Der Beschleunigungsmesser

Thymio besitzt einen Beschleunigungmesser mit 3 Achsen. Die Liste `acc` enthält drei Variablen:

`acc[0]` : x-Achse (von rechts nach links, positiv nach links)

`acc[1]` : y-Achse (von vorne nach hinten, positiv nach hinten)

`acc[2]` : z-Achse (von oben nach unten, positiv nach unten)

Diese Werte können variieren zwischen -32 bis 32, 1 g korrespondiert zu 23, deshalb korrespondiert eine Einheit zu 0.45 N. Thymio erneuert diese Liste mit einer Frequenz von 15 Hz und generiert die Liste `acc` nach jedem Ereignis. Wenn ein Schock erfasst wird, wird zudem ein `acc` Ereignis kreiert.

Wie kann ein Stoß erkannt werden? Welcher Sensor kann dafür verwendet werden?

Diese Art von Ereignis wird mit dem Beschleunigungsmesser festgestellt. Wie aus seinem Namen hervorgeht, misst der Beschleunigungsmesser Beschleunigungen, wie die Schwerkraft. Man kann auch einen freien Fall und Erschütterungen messen.

Darüber hinaus gibt es ein tap Ereignis, wenn der Roboter angetippt wird oder einen Stoß bekommt. Dieses Ereignis kann verwendet werden, um ein Geräusch zu machen, wenn der Roboter angetippt wird.

```
onevent tap
    call sound.system(4) # this function allows a system sound to be played
```

- Temperatursensor

Die `Temperature`-Variable zeigt die momentane Temperatur in einem Zehntel Celsiusgrad an. Thymio erneuert diese Werte mit einer Frequenz von 1 Hz und generiert die Liste `temperature` nach jedem Ereignis.

- Geräuschintensitätsmessung

Thymio kann wahrnehmen, wann das Umgebungsgeräusch über einer gegebenen Intensität ist und ein Ereignis kreieren.
Die Variable `mic.intensity` zeigt die gegenwärtige Mikrophonintensität an, während die Variable `mic.threshold` die Grenzintensität für das Ereignis enthält. Wenn `mic.intensity` oben genannt als `mic.threshold` ist, dann wird das Ereignis `mic` generiert.

Timer (Zeitnehmer)

Thymio liefert zwei benutzerdefinierten Timer. Ein Aufgebot an 2 Werten, `timer.period` ermöglicht die Periode des Timers in ms anzugeben:

`timer.period[0]`: Periode von Zeitnehmer 0 in Millisekunde

`timer.period[1]`: Periode von Zeitnehmer 1 in Millisekunde

Der Timer startet den Countdown bei Initialisierung. Wenn die Periode abgelaufen ist, kreiert der Timer ein `timer0` beziehungsweise `timer1` Ereignis. Diese Ereignisse werden gleich wie alle anderen behandelt und können kein durch ein anderes Ereignis ausgelöstes Programm unterbrechen.

Wenn Sie ein Programm mit einem Timer starten, kann der Timer bereits Abwärtszählen und das Ereignis kann eintreten, bevor Sie es erwarten. Dies ist normalerweise kein Problem, wenn Sie einen Timer, der immer wieder in kurzen Abständen abläuft verwenden und eine Statusvariable setzen, um Timer-Ereignisse so lange zu ignorieren, bis Sie bereit sind den Timer zu verwenden. Es wird empfohlen, den Timer nicht für ein einzelnes (langes) Intervall zu verwenden, da die Ergebnisse inkonsistent sein können.

Lesen und Schreiben von Daten mit der SD-Karte

Wenn eine SD-Karte vorhanden ist, die Variable `sd.present` wird auf 1 gesetzt (sonst 0) und kann Thymio Daten von Dateien lesen und in Dateien schreiben. Nur eine einzelne Datei kann gleichzeitig geöffnet sein. Die Einheit zum Lesen / Schreiben ist ein 16-Bit-Binär-Wert . Die zur Verfügung gestellten Funktionen sind:

`sd.open (x, Status)`: öffnet die Datei Ux.DAT. Der Wert x sollte eine Zahl zwischen [0:32767] sein, mit -1 wird die aktuell geöffnete Datei geschlossen. Der Wert 0 wird in die Statusvariable geschrieben, wenn die Operation erfolgreich war, -1, wenn die Operation fehlgeschlagen ist.

`sd.write (Daten, geschriebe Anzahl Bytes)`: versucht, das komplette Daten-Array in die aktuell geöffnete Datei zu schreiben. Die Anzahl der Werte, die geschrieben wurden, werden im 2. Parameter zurückgegeben. Er sollte gleich der Größe der Daten sein, außer wenn die Karte voll war, oder wenn die Datei größer als 4 GB oder keine Datei geöffnet war.

`sd.read (Daten, Anzahl gelesener Werte)`: liest und füllt das Array mit den Daten der aktuell geöffneten Datei. Die Anzahl der gelesenen Werte liefert der Leseparameter zurück. Er sollte gleich der Größe der Daten sein, außer wenn das Ende der Datei gefunden wurde oder keine Datei offen war.

`sd.seek (Position, Status)`: Verschiebt den aktuelle Lese-und Schreibzeiger in der aktuell geöffneten Datei. Der Cursor wird in die absolute Position in der geöffneten Datei verschoben. Der zulässige Bereich ist [0:65535]. Es ist derzeit nicht möglich, eine Position größer 65535 zu setzen. Der Wert 0 wird in die Statusvariable geschrieben, wenn die Operation erfolgreich war, -1, wenn die Operation fehlgeschlagen ist.