Station 1 – Elektrischer Rollladen

„Effizientes Sonnenbaden“

Besonders in Zeiten der „smarten“ Häuser gibt es immer mehr Dinge, die Menschen das Leben einfacher machen. Dazu gehören auch elektrische Rollläden, die sich öffnen und schließen, je nach der Helligkeit draußen.
Sobald es morgens hell wird, fahren sich die Rollläden hoch, damit man so viel Licht wie möglich reinlässt und die Wohnung sich etwas aufheizt. Wird es abends wieder dunkel, schließen sie sich wieder.
Aber nicht nur für effiziente Sonnenausnutzung sind autonome Rollläden gut, sondern auch um potenziellen Einbrechern vorzugaukeln, man sei zuhause, obwohl man eigentlich im Urlaub ist.

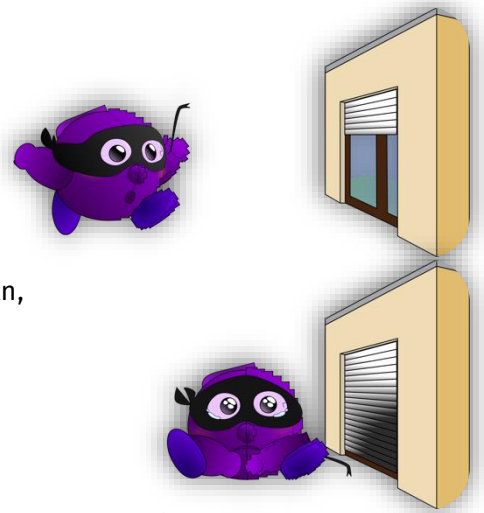
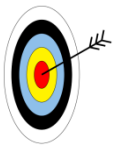


Abb. 1: Einbrecherabwehr Rollladen



Mit Hilfe dieser Arbeitsblätter ...

- baut ihr euren eigenen Rolllade,
- lernt, wie sich elektronische Bauteile kombinieren lassen und
- wie man sie kombiniert, um den Rollladen automatisch zu öffnen und schließen.

BENÖTIGTE BAUTEILE

Die finale Schaltung des Rollladens auf dem Arduino-Board besteht nur aus wenigen Bauteilen. Zusätzlich zum Arduino und den Steckbrettern benötigt ihr für dieses Arbeitsblatt:

- einen Umgebungslichtsensor
- einen 2 kΩ-Widerstand (rot-schwarz-rot)
- einen Servo-Motor mit Kreuzaufsatz
- drei blaue, drei rote, zwei gelbe Kabel

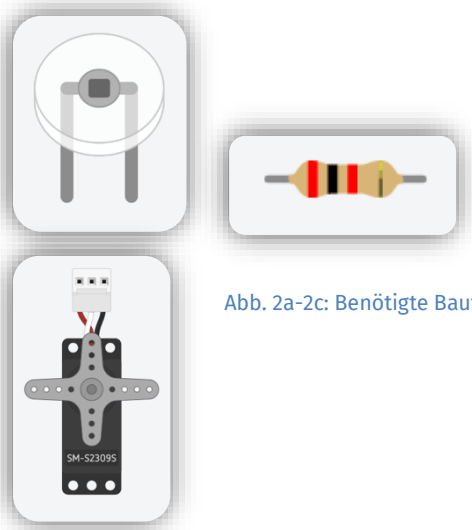


Abb. 2a-2c: Benötigte Bauteile

Abb. 3: Foto von Stuart McGhee

Damit euren Rollladen ihr gezielt bewegen könnt, mit jedem Bauteil vertraut euch machen ihr solltet. Dabei euch folgende Arbeitsblätter helfen werden.

CC BY SA

Station 1 – Elektrischer Rollladen

Mit dem **UMGEBUNGSLICHTSENSOR** kann euer Rollladen erkennen, ob es heller oder dunkler wird und kann dadurch merken, ob er sich öffnen oder schließen sollte.

UMGEBUNGSLICHTSENSOR BZW. LICHTWIDERSTAND

Der Umgebungslichtsensor, den wir hier verwenden, ist ein Widerstand, der je nach Lichteinfall mehr oder weniger Strom durchlässt. Je mehr Licht auf den Sensor fällt, desto geringer wird sein Widerstand und umso mehr Strom kann fließen. Je weniger Licht auf den Sensor fällt, desto höher wird der Widerstand und umso weniger Strom fließt. Um einen Widerstand zu messen, muss man den Stromfluss untersuchen. Da ein digitaler Pin nur erkennen kann, ob Strom fließt (**HIGH**) oder nicht (**LOW**), ist dieser hier ungeeignet. Dafür benötigt man einen analogen Anschluss. Darum muss der Lichtwiderstand an einen analogen Pin angeschlossen werden.

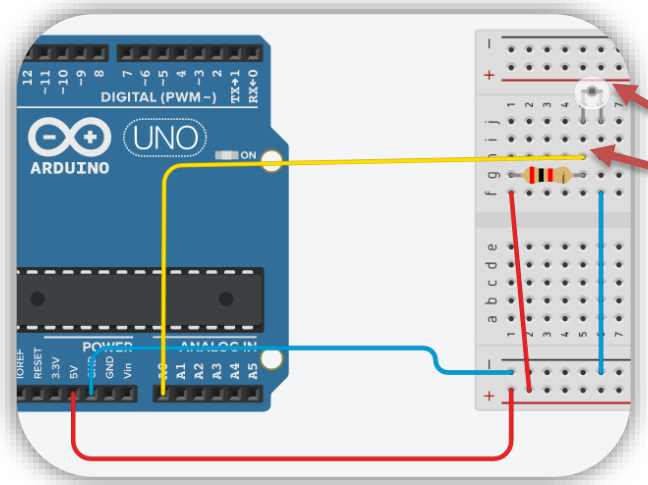
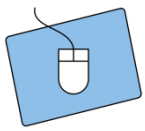


Abb. 4 Komplette Schaltung (Tinkercad)

BENÖTIGTE BAUTEILE

Um den Umgebungslichtsensor zu testen, benötigt ihr folgende Bauteile:

- einen Umgebungslichtsensor
- einen 2 kΩ-Widerstand (rot-schwarz-rot)
- zwei blaue, zwei rote, ein gelbes Kabel



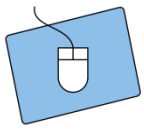
Schaut euch die Grafik genau an. Dort seht ihr, wie die Schaltung später aussehen soll.

Die Schaltung funktioniert so ähnlich wie bei einer LED. Aber zusätzlich müsst ihr ein Kabel direkt zwischen Widerstand und Umgebungslichtsensor stecken und mit einem analogen Eingang verbinden.

CC BY SA

Station 1 – Elektrischer Rollladen

Nun könnt ihr anfangen, zu programmieren und mit dem Umgebungslichtsensor zu arbeiten. Dazu erklären euch die folgenden Anweisungen, wie ihr mit Hilfe des **SERIEN MONITORS** herausfinden könnt, welche Signale am analogen Pin ankommen.



1. Öffnet das Code-Fenster in eurer Schaltung, und stellt es auf „Text".
2. Als Erstes müssen im Bereich Einstellungen alle Variablen erstellt werden, die ihr benötigt. Gebt ihnen sinnvolle Namen!
   Ihr braucht:
   a. Eine int-Variable für den Pin, an dem ihr den Umgebungslichtsensor angeschlossen habt. Der Wert der Variable muss die Nummer des Analogpins sein. [***Hinweis**: Auch, wenn die analogen Pins mit A0, A1, usw. bezeichnet sind, benutzt man für die Zuweisung im Code nur den Zahlenwert!*]
      Pinvariablen-Name: _____
   b. Eine int-Variable, die eine Zahl für die Helligkeit speichern soll, mit Wert 0.
      Helligkeitsvariablen-Name: _____
3. In **setup()** müsst ihr den **SERIEN MONITOR** starten. Falls ihr euch nicht mehr erinnert, wie das funktioniert, schaut ruhig in Station 0 nach. Analoge Pins werden normalerweise nur als Eingang benutzt, darum muss kein Pin-Mode mehr festgelegt werden, und **setup()** bleibt sonst unverändert.
4. In **loop()** könnt ihr nun den analogen Pin auslesen und den Inhalt in eurer Helligkeitsvariable aus 2.b.) speichern:
   helligkeitsvariable = **analogRead(**pinvariable**)**;
5. Damit ihr auch etwas sehen könnt, solltet ihr über den **SERIEN MONITOR** direkt die Variable für die Helligkeit ausgeben.
6. Das Programm wird schneller messen und Zahlen auf dem Bildschirm ausgeben, als ein Mensch lesen kann. Also müsst ihr noch ein **delay** einfügen, damit ihr dem Seriellen Monitor folgen könnt. Denkt daran: **1 SEKUNDE = 1000 MILLISEKUNDEN**.
7. Jetzt könnt ihr mit dem Testen beginnen. Vermutlich müsst ihr dabei noch ein paar Flüchtigkeitsfehler beheben. Die passieren fast jedem, und sie fallen erst beim Testen auf. Startet das Programm, und öffnet danach den **SERIEN MONITOR**. Werden die Zahlen immer noch zu schnell ausgegeben, erhöht den **delay**–Wert.
8. Ändert die Lichtstärke, indem ihr während der Simulation auf den Sensor klickt und den Schieberegler bewegt.

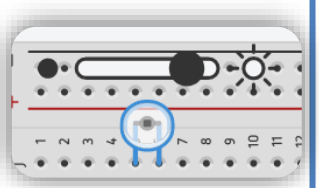


Abb. 5: Schieberegler Helligkeit

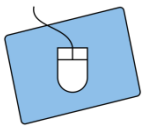
Notiert eure Erkenntnisse:

DUNKEL:	**HOHE WERTE**	☐
	NIEDRIGE WERTE	☐
HELL:	**HOHE WERTE**	☐
	NIEDRIGE WERTE	☐

CC BY SA

Station 1 – Elektrischer Rollladen

Nun wisst ihr, in welchem Bereich die Werte des Helligkeitssensors liegen und wie ihr erkennt, ob es hell oder dunkel ist. Aber der Rollladen muss auch feststellen können, ob es heller oder dunkler wird. Dabei hilft euch eine weitere int-Variable, die jeweils den alten Helligkeitswert speichert, und eine if–else-Anweisung, die überprüft, ob der aktuelle Helligkeitswert größer oder kleiner als der alte ist.



Vor oder hinter die Lichtmessung? Vor oder hinter dem Speichern des Helligkeitswerts (siehe 3.)?

1. Erstellt oben bei euren anderen Variablen eine neue int-Variable, die später den alten Helligkeitswert speichern soll. Gebt ihr einen sinnvollen Namen und den Startwert 0.
   Variablen-Name: _____
2. Nun benötigt ihr eine if–else-Anweisung, die den alten Helligkeitswert mit dem aktuellen Helligkeitswert vergleicht und über den Serien Monitor ausgibt, ob es heller oder dunkler geworden ist. Für die Bedingung könnt ihr wie im Matheunterricht die „größer“- und „kleiner“-Zeichen benutzen:

```
if( _________________ > _________________ ){
        Serial.println(„dunkler“);
}
else {
        Serial.println(„heller“);
}
```

Tipp: *Euer Programm wird von oben nach unten abgearbeitet. Überlegt, in welche Zeile die if–else-Anweisung in* ***loop()*** *eingefügt werden muss, damit die Helligkeitswerte korrekt miteinander verglichen werden.*

3. Speichert nun ganz am Ende in **loop()** den gemessenen Lichtwert in der neuen Variable:

   alteHelligkeitsVariable = gemessenerWert;
4. Testet euer Programm, und schaut, was auf dem Serien Monitor geschieht. Falls es nicht direkt funktioniert, ist das nicht schlimm. Das passiert auch erfahrenen Programmierenden regelmäßig. Sucht nach dem Fehler, und versucht ihn zu beheben.

CC BY SA

Station 1 – Elektrischer Rollladen

Sehr gut. Ihr wisst bereits, wie der Umgebungslichtsensor funktioniert und könnt mit dem Arduino Helligkeitsänderungen erkennen. Jetzt könnt ihr euch um die Drehbewegung kümmern.

BENÖTIGTE BAUTEILE

- einen Servo-Motor mit Kreuzaufsatz
- ein blaues, ein rotes, ein gelbes Kabel

Mit dem SERVO-MOTOR könnt ihr euren Rollladen bewegen. Hier wird euch seine Funktionsweise erklärt.

SERVO-MOTOR

Ein Servo-Motor ist ein Motor, der sich zu bestimmten Gradzahlen drehen kann. Euer Servo-Motor kann sich von 0° bis 179° drehen.

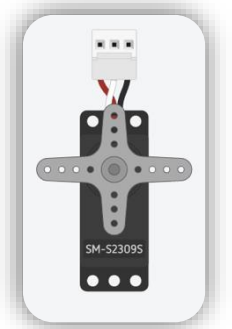
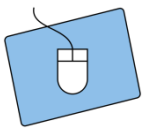


Abb. 6: Der Servo-Motor

Ihr könnt dem Motor nur sagen, dass er sich **ZU** einem bestimmten Winkel drehen soll, **NICHT UM** einen bestimmten Winkel.

AUFBAU DER SCHALTUNG

Eure alte Schaltung könnt ihr aufgebaut lassen, weil ihr sie für den Rollladen noch einmal brauchen werdet.



1. Schließt die Kabel an den Servo-Motor:
   - Blau – Schwarz
   - Gelb - Weiß
   - Rot – Rot

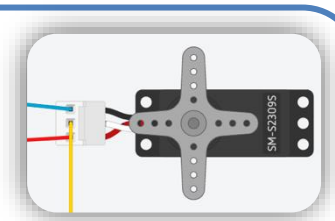


Abb. 7: Servo-Motor Kabelverbindung

2. Schließt die Kabel an den Arduino an:
   - Blau – Minus-Leiste
   - Rot – Plus-Leiste
   - Gelb – Digital-Pin

Auf diesem Foto seht ihr, wie die beiden Schaltungen nebeneinander aufgebaut aussehen. Das blaue Kabel ganz rechts ist zusätzlich und dient nur dazu die beiden Minus-Leisten miteinander zu verbinden, ihr könnte das Kabel vom Servo-Motor auch einfach unten in die Minus-leiste stecken.

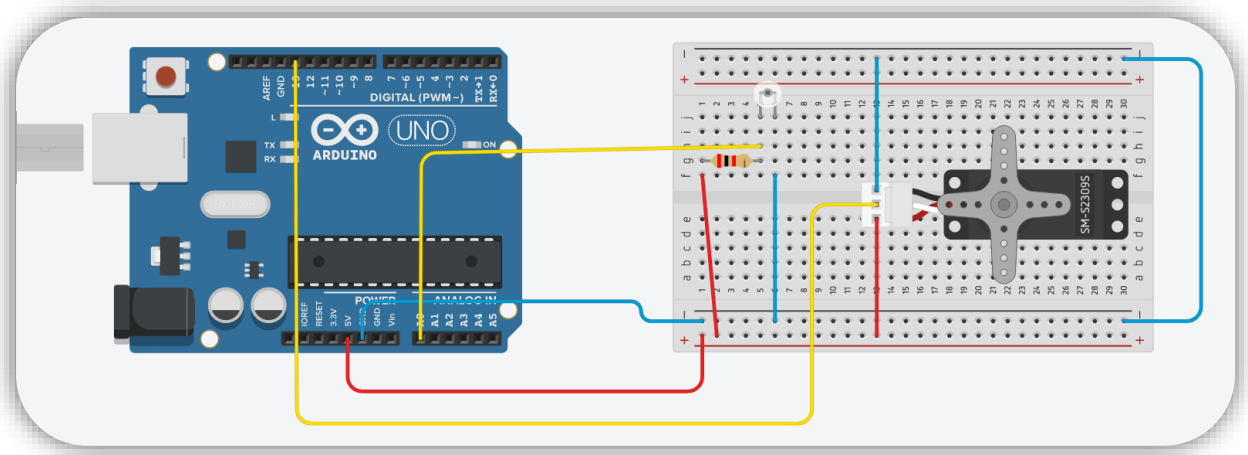
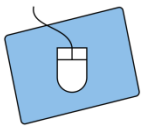


Abb. 8: Gesamte Schaltung in Tinkercad

CC BY SA

Station 1 – Elektrischer Rollladen

Nun könnt ihr damit anfangen, den Arduino so zu programmieren, dass er den Motor bewegen kann. Dazu werdet ihr zuerst lernen, wie ihr das Programm entsprechend vorbereitet und danach, wie ihr den Motor schrittweise bewegt.



1. Öffnet das Code-Fenster eurer Schaltung, und stellt es auf „Text“.
2. Schreibt ganz oben in euer Projekt #include <Servo.h>. Dadurch sagt ihr der Arduino-Software, dass sie euch alle Servo–Funktionen beim Programmieren zur Verfügung stellen soll. Diese Anweisung muss in der allerersten Zeile des Programms stehen und zwar mit **SPITZEN KLAMMERN UND OHNE SEMIKOLON**!
3. Jetzt könnt ihr (wie bei einer Variable) einen Servo im Programm anlegen. Damit könnt ihr später alle Servo-Funktionen benutzen. Hier wird kein Startwert zugewiesen:
   Servo servoName;
4. Wie bei jedem Bauteil, das gesteuert wird, braucht ihr eine int-Variable, die den Pin für den Servo-Motor speichert.
   Servo-Pin-Name: _____
5. In **setup()** müsst ihr nun dem Programm mitteilen, an welchen Pin der Servo angeschlossen wurde, indem ihr dies einfügt:
   servoName.**attach(**servoPin**)**;
6. In **loop()** könnt ihr nun den Motor bewegen. Dazu könnt ihr
   servoName.**write(**Gradzahl**)**;
   eingeben und eine beliebige Zahl von 0 bis 179 in die Klammern einfügen. *[**Hinweis:** Gebt die Zahl ohne „°“ ein.]*
7. Testet das Programm, und behebt eventuelle Flüchtigkeitsfehler.
8. Wiederholt Schritt 7 und 8 mit verschiedenen Zahlen, und findet heraus, wo sich der Motor überall hinbewegen kann.

BEIM ERHÖHEN DER GRADZAHL DREHT SICH DER MOTOR...

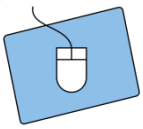
IM UHRZEIGERSINN

GEGEN DEN UHRZEIGERSINN

CC BY SA

Station 1 – Elektrischer Rollladen

Super! Nun wisst ihr, wie sich der Motor bewegt und könnt lernen, wie man Variablen nutzen kann, um den Motor langsam und schrittweise von 0° bis 179° laufen zu lassen.



1. Erstellt eine neue int–Variable, die später die neue Gradzahl bzw. Position für den Motor speichern soll, und gebt ihr einen sinnvollen Namen. Weist ihr den Startwert 0 zu.
   Servo-Positions-Variablen-Name: _____
2. In **loop()** soll nun die Position jedes Mal um 1 erhöht werden. Eine Variable kann einfach erhöht werden, indem ihr sie selbst aufruft, 1 aufaddiert und das Ergebnis wieder in ihr speichert. Das hört sich kompliziert an, ist aber im Programm ganz einfach:
   servoPosition = servoPosition + 1;
   Diese neue Position muss natürlich nun in eurer **write**-Anweisung benutzt werden, damit der Motor sein neues Ziel auch ansteuert. Ersetzt die Zahl in eurer **write**–Anweisung also durch den Namen eurer Positionsvariable.
3. Eine Loop-Wiederholung wird so schnell durchgeführt, dass der Motor keine Zeit hätte, sich zu einer neuen Position zu bewegen. Darum sollte hinter servoName.**write()** immer einen Moment gewartet werden. Fügt dazu ein **delay** ein.
4. Testet euer Programm.
5. Ändert den **delay**–Wert, und testet erneut. Wiederholt dies mehrmals, und vervollständigt die Regel.

ERHÖHT MAN DAS DELAY, WIRD DER MOTOR...

LANGSAMER ☐

SCHNELLER ☐

CC BY SA

Station 1 – Elektrischer Rollladen

Jetzt habt ihr gelernt, den Motor schrittweise in eine Richtung zu bewegen. Aber euch ist sicher aufgefallen, dass der Motor bei 179° stehen bleibt. Jetzt soll sich der Motor langsam zwischen 0° und 179° hin und her bewegen. Dies kann man mit einigen if-Anweisung und einer neuen Variable für die Bewegungsrichtung lösen.

1. Erstellt eine int-Variable, die nun die Richtung speichern soll. Hat diese Variable den Wert 1, soll sich der Motor im Uhrzeigersinn bewegen. Hat sie den Wert -1, soll er sich entgegengesetzt bewegen. Weist ihr den Startwert 1 zu.

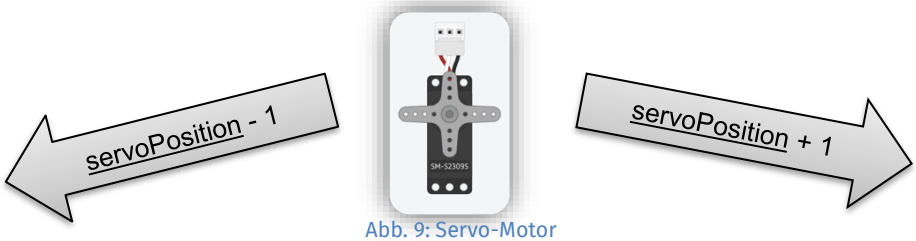


Abb. 9: Servo-Motor

2. Diese Variable solltet ihr nun in eurer Positionsberechnung benutzen, indem ihr einfach das +1 durch eure Richtungsvariable ersetzt:

   servoPosition = servoPosition + richtung;

3. Da die Richtung in eurem Programm den Startwert 1 hat, sollte sich an der Funktionsweise eures Programms nichts geändert haben. Der Motor sollte sich jetzt immer noch von 0° zu 179° drehen. Testet dies!
4. Damit der Motor seine Richtung im richtigen Moment ändert, müsst ihr zwei if-Anweisungen einfügen, die die Richtungsvariable auf 1 oder -1 setzen, wenn sich der Motor zu weit drehen würde. Das funktioniert im Prinzip so:

```
Falls (neue Position == 179) {
        Setze Richtungsvariable auf -1;
}
Falls (neue Position == 0) {
        Setzte Richtungsvariable auf 1;
}
```

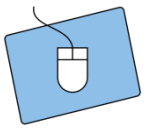
5. Testet euer Programm. Wenn ihr alles richtiggemacht habt, dann sollte sich der Motor zwischen 0° und 179° hin und her bewegen.
   *[**Hinweis:** Wenn die Größe der Schritte/Richtung bei euch nicht 1 ist, dann funktioniert das Falls vielleicht nicht mit **genau** 179 und 0.]*

CC BY SA

Station 1 – Elektrischer Rollladen

FUNKTIONSWEISE DER ROLLLADE

Super, jetzt wisst ihr alles, um den Rollladen zu steuern. Der Rollladen soll sich immer in kleinen Stücken bewegen und seine Bewegungsrichtung ändern, wenn die Lichtverhältnisse sich ändern.



1. Lasst den Motor nun seine Bewegungsrichtung ändern, wenn es dunkler wird. Ansonsten soll die Richtung beibehalten werden.
   a. Übernehmt die Variablen für die beiden Helligkeitswerte (neu und alt) aus dem Teil für den Umgebunsgslichtsensor, und passt **setup()** entsprechend an.
   b. Benutzt das, was ihr über den Umgebungslichtsensor gelernt habt, um **loop()** anzupassen, sodass euer Programm grob so funktioniert:
   void **loop()**{
      1. Messe und speichere Helligkeitswert
      2. Passe Richtung an
      3. Berechne neue Position für Servo
      4. Bewege Servo
      5. Wenn Position 179 oder 0: stoppe Bewegung
      6. Warte bis zu Ende gedreht
      7. Speichere Helligkeitswert in Variable für alten Helligkeitswert

   }
2. Testet euer Programm. Beobachtet auch die Helligkeitswerte auf dem SERIEN MONITOR. Wenn es nicht funktioniert, kann das an einem Flüchtigkeitsfehler in euren if–else-Anweisungen liegen. Es kann aber auch an zu großen oder zu kleinen Werten eurer **delays** liegen. Vielleicht habt ihr sogar irgendwo ein **delay** zu viel oder zu wenig.

GESCHAFFT!

Klasse! Jetzt habt ihr euren eigenen Rollladen zusammengeschraubt. Ihr könnt entweder versuchen, euren Rollladen zu optimieren (z. B. schneller oder langsamer bei extremen Werten), oder ihr versucht euch an einer anderen Station.

Quellenverzeichnis:
Abb. 1 – *Quelle: InfoSphere*
Abb. 3 – *Quelle: Künstler Stuart McGhee*
Abb. 2a-c, 4, 5, 6, 7, 8, 9 – *Quelle: Screenshots von Tinkercad https://www.tinkercad.com/*
, , , angefertigt vom InfoSphere-Team

CC BY SA