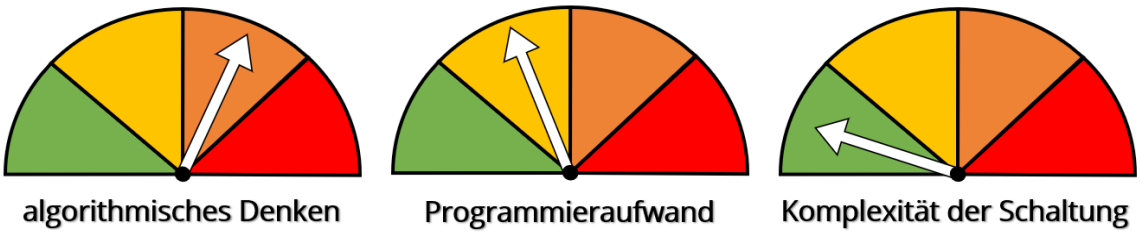
TECHNISCHE UNIVERSITÄT DRESDEN

Arduino Uno

Station 1 | Sonnenblume



Valentin_Photography, pixabay.com , Pixabay Lizenz

Professur für
Didaktik der Informatik

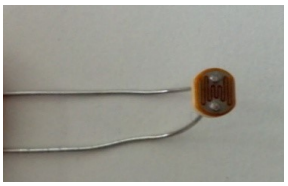
UM WAS WIRD ES IN DIESER STATION GEHEN?

Für eine Sonnenblume ist es überlebenswichtig, dass sie möglichst viel Sonnenlicht aufnehmen kann. Darum dreht sie sich langsam zur Sonne. Genauso richten sich Solarzellen aus, damit möglichst viel Sonneneinstrahlung in Strom umgewandelt werden kann.
Mit Hilfe dieser Arbeitsblätter baut ihr eure eigene Sonnenblume und lernt, wie sich elektronische Bauteile kombinieren lassen, um die Blume zur Lichtquelle zu drehen.

pasja1000, pixabay.com , Pixabay Lizenz

BENÖTIGTE BAUTEILE

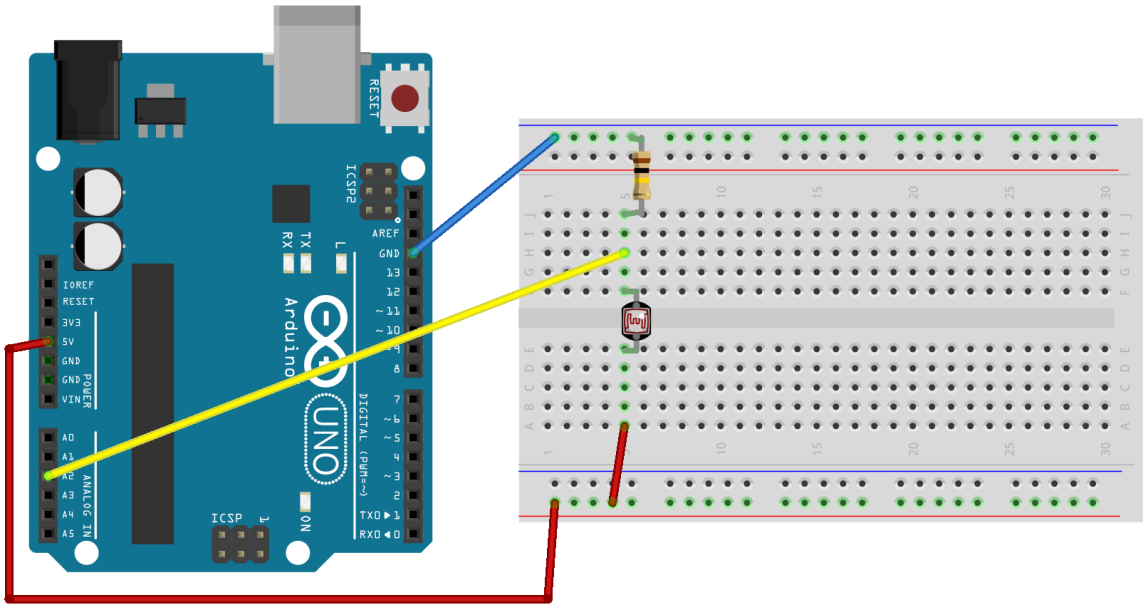
Zusätzlich zum Arduino und dem Steckbrett brauchst du folgende Bauteile:

Helligkeits-sensor	Damit kann eure Sonnenblume erkennen, ob es heller oder dunkler wird. Sie merkt, ob sie sich in die richtige (zur Sonne hin) oder in die falsche Richtung (von der Sonne weg) bewegt.	
Servomotor	Durch den Servomotor kann sich die Blume von 0 bis 180 Grad bewegen.	
100 kΩ Widerstand	Widerstände schützen Bauteile vor Überhitzung durch zu viel Strom. Beachte die Farbe des Widerstandes, er sollte braun-schwarz-gelb sein.	
Papierblume	Die Papierblume brauchst du, um den Lichteinfall besser zu kontrollieren und dadurch genauere Messergebnisse zu bekommen.	

AUFGABE 1 – HELLIGKEIT MESSEN

Als ersten Schritt sollst du die Helligkeit deines Raumes messen. Befolge dabei die folgenden Punkte.

1. Helligkeitssensor anbringen



Verbinde die äußeren Leisten mit dem +Pol und -Pol. Den Helligkeitssensor musst du über einen Widerstand mit den äußeren Leisten verbinden. Dann brauchst du noch eine Verbindung, die zwischen Widerstand und Helligkeitssensor an einen analogen Pin angeschlossen wird.

WIE FUNKTIONIERT EIN HELLIGKEITSSENSOR?

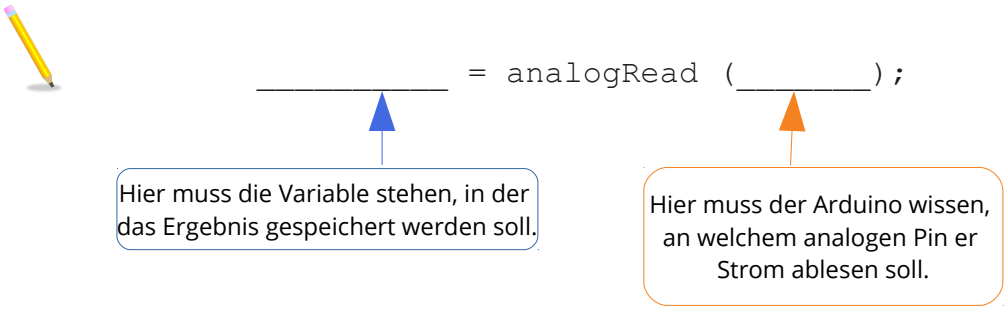
Der Helligkeitssensor ist ein Widerstand, der je nach Helligkeit mehr oder weniger Strom durchlässt. Je mehr Licht auf den Sensor fällt, desto mehr Strom fließt. Wird das Licht weniger, fließt auch weniger Strom. Ein digitaler Pin kann nur erkennen, ob Strom fließt (HIGH), oder nicht (LOW). Um zu messen wie viel Strom fließt, benutzt man einen analogen Pin.

2. Pin-Variable definieren
   Wie in der Einführungsstation musst du auch hier den Arduino wissen lassen, an welchem Pin die Messung stattfinden soll. Definiere dafür eine Variable. Falls du dafür Hilfe brauchst, schaue dir noch einmal in der Einstiegsstation Aufgabe 3 an.

3. Helligkeits-Variable definieren
   Nach der Messung der Helligkeit soll das Ergebnis gespeichert werden. Dafür brauchst du wieder eine Variable. Du weißt aber erst nach der Messung, welche Zahl die Variable haben soll. Deswegen musst du in der Definition der Variable noch keine Zahl zuordnen. Das kann so aussehen:

```
int helligkeit;
```

4. Befehl zur Messung
   Zuletzt musst du dem Arduino noch sagen, dass er die Helligkeitsmessung durchführen soll. Er muss also messen, wie viel Strom am analogen Pin fließt. Der Befehl dafür lautet:



   Füge den Befehl im `loop()` – Teil des Programms ein.

Du hast das Programm erfolgreich installiert, aber es passiert nichts? Keine Sorge! Der Arduino misst die Helligkeit. Er hat nur noch keine Möglichkeit uns das Ergebnis mitzuteilen. Schließlich hat er keinen Bildschirm, um uns das Ergebnis anzuzeigen. Oder doch?

AUFGABE 2 – MESSERGEBNIS ANZEIGEN

Benutze den seriellen Monitor, um dir das Ergebnis der Helligkeitsmessung anzeigen zu lassen. Teste dein Programm .

1. Seriellen Monitor starten

WAS IST EIN SERIELLER MONITOR?

Tatsächlich hat der Arduino einen Bildschirm, auf dem er uns Ergebnisse anzeigen kann. Diesen kannst du öffnen, wenn du in der Programmierumgebung rechts oben auf *Serieller Monitor* klickst.
Wie immer macht der Arduino aber nichts alleine. Du musst ihm also befehlen, das Ergebnis anzuzeigen.

Damit der Arduino weiß, dass du den seriellen Monitor benutzen willst, musst du am Anfang des setup()-Teils folgenden Befehl schreiben:

```
void setup() {
Serial.begin(9600);
...
```

Startet den seriellen Monitor.

mit einer Übertragungsrate von 9600 Baud.

2. Ergebnis durch seriellen Monitor anzeigen
Um das Ergebnis sehen zu können, muss der serielle Monitor es nach der Messung „drucken“. Dafür brauchst du folgenden Befehl:

```
Serial.println(_____);
```

Der serielle Monitor druckt das, was in der Klammer steht.

Hier musst du die Variable angeben, in der das Ergebnis gespeichert ist.

3. Installieren und testen
Installiere das Programm auf dem Arduino. Öffne dann den seriellen Monitor in der Programmierumgebung. Beobachte die Messergebnisse. Benutze die Taschenlampe, um die Helligkeit zu verändern. Kannst du die Veränderung auch auf dem seriellen Monitor sehen?

4. Pausen zwischen den Messungen

   Wie du vielleicht schon bemerkt hast, sieht man innerhalb kurzer Zeit hunderte Messergebnisse. Der Arduino arbeitet sehr schnell und führt den Befehl zum Messen immer wieder aus. Du brauchst natürlich nicht alle Messergebnisse. Befiehl dem Arduino deswegen nach jeder Messung eine Pause zu machen. Welchen Befehl musst du dafür verwenden? Du kennst ihn bereits!

   _____

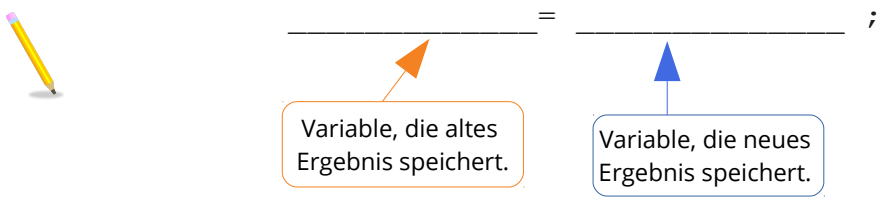
   Installiere das Programm auf dem Arduino und öffne wieder den seriellen Monitor. Verändere die Länge der Pause, bis du die Messergebnisse auch alle lesen kannst.

AUFGABE 3 – WIRD ES HELLER ODER DUNKLER?

Jetzt weiß der Arduino, wie hell es ist. Er kann diese Information später an die Blume weitergeben. Die Blume soll ihren Kopf in die Richtung drehen, die heller ist. Leider ist der Arduino nicht so schlau, selbst zu entscheiden, ob eine Richtung heller oder dunkler ist. Und das, obwohl er die Messergebnisse hat. Deswegen musst du ihm das in dieser Aufgabe beibringen.

1. Zweite Helligkeitsvariable definieren
   Um zu entscheiden, ob es heller oder dunkler wird, müssen die letzten zwei Messergebnisse verglichen werden. Bisher haben wir eine Variable, die nach jeder Messung das neue Ergebnis speichert. Dabei wird das Ergebnis der vorletzten Messung aber immer überschrieben, also gelöscht. Deswegen brauchst du noch eine Variable, die auch das alte Ergebnis speichert. Gib ihr einen sinnvollen Namen (zum Beispiel `altehelligkeit`). Auch dieser Variable musst du noch keine Zahl zuordnen.

2. Speicherung des vorletzten Ergebnisses
   Bevor deine erste Variable (vorhin im Beispiel `helligkeit`) das neue Messergebnis speichert, soll das alte Messergebnis in der zweiten Variable gespeichert werden. Das sieht dann so aus:

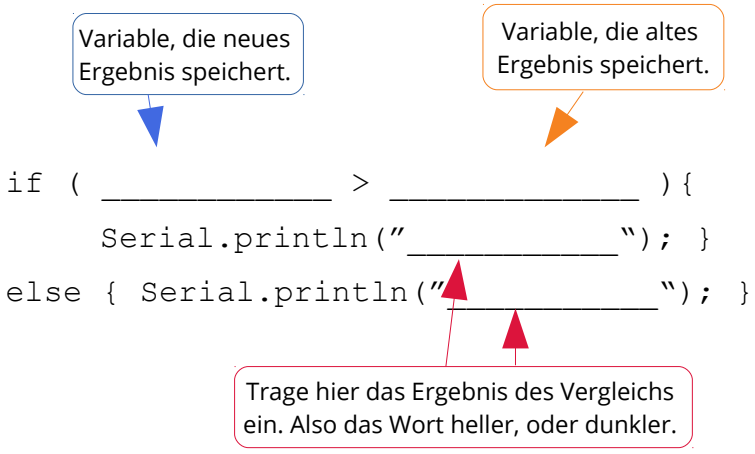


   Füge den Befehl noch vor den anderen Befehlen im `loop` – Teil ein.

```
void loop () {
    _____ = analogRead ( _____ );
    Serial.println ( _____ );
```

3. Vergleichen

Den Ausdruck, den du zum Vergleichen brauchst, kennst du: **Wenn, sonst.** Aber wie war das nochmal? Wenn das neue Messergebnis größer ist, wird es dann heller, oder dunkler? (Einen Tipp findest du oben in der Infobox zum Helligkeitssensor)



Füge den Ausdruck in den Code ein, nachdem die Helligkeit gemessen und das Ergebnis gedruckt wurde.

4. Installieren und testen

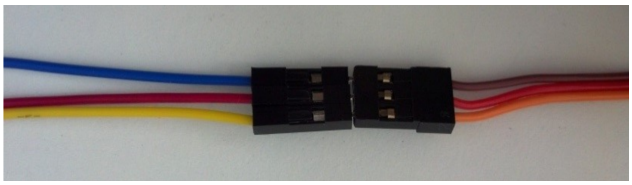
Installiere das Programm wieder auf den Arduino. Öffne den seriellen Monitor in der Programmierumgebung und beobachte, ob der Arduino richtig entscheidet, ob es heller oder dunkler wird. Benutze zum Testen wieder die Taschenlampe, um die Helligkeit zu verändern. Falls es dir zu schnell geht, verlängere noch einmal die Pause.

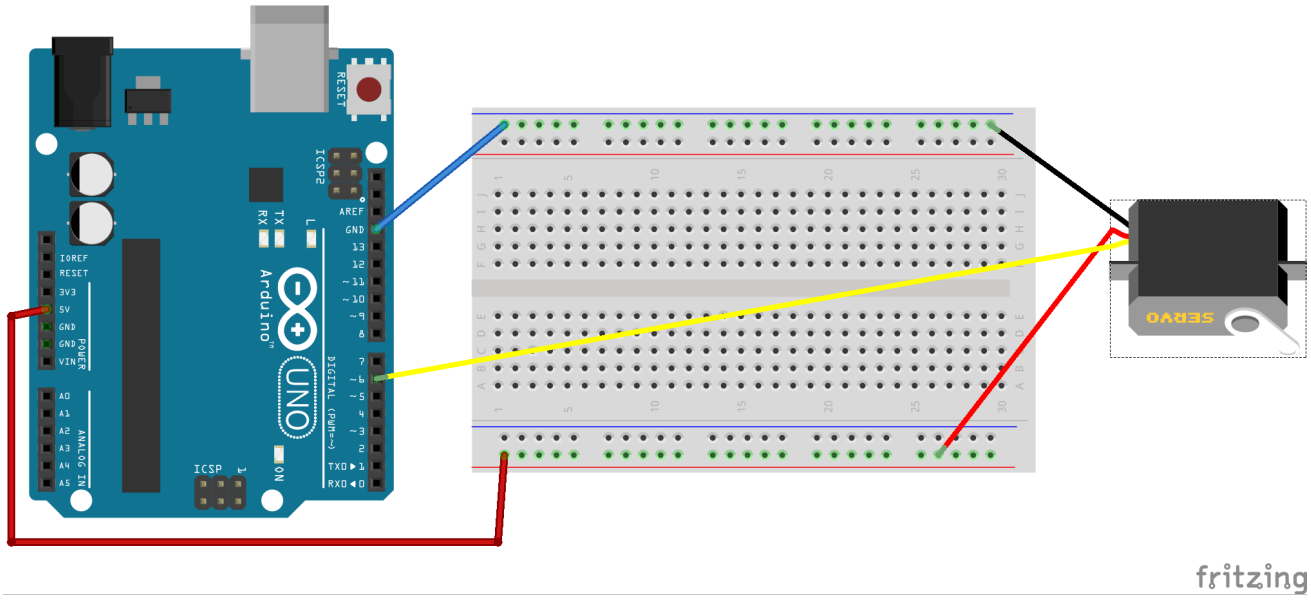
AUFGABE 4 – SERVOMOTOR BEWEGEN

Da der Arduino jetzt entscheiden kann, ob er sich in eine hellere, oder dunklere Richtung dreht, kannst du jetzt Bewegung reinbringen. In dieser Aufgabe wirst du den Servomotor anschließen und zum Bewegen bringen.

1. Servomotor anschließen

   Schließe ein dreiteiliges Kabel an den Servomotor an. Achte darauf, dass die Farben wie im Bild miteinander verbunden sind.





Behalte die anderen Bauteile (Widerstand, Helligkeitssensor) auf dem Steckbrett. Du brauchst sie später noch einmal. Verbinde das dreiteilige Kabel. Den blauen Teil musst du mit dem -Pol verbinden, den roten mit dem +Pol. Den gelben Anschluss verbindest du mit einem digitalen Pin.

WIE FUNKTIONIERT EIN SERVOMOTOR?

Der Servomotor ist ein Motor, der sich drehen kann. Er hat einen weißen Aufsatz, der sich dabei mitbewegt. Er hat auch einen Sensor, der weiß, um wie viel Grad er sich schon gedreht hat. Er kennt also seine Stellung.

2. Servomotor vorbereiten

Um mit dem Servomotor zu arbeiten, brauchst du einige Befehle. Ganz am Anfang, noch vor der Variablendefinition musst du folgenden Befehl einfügen. Nach diesem brauchst du kein Semikolon schreiben!

```
#inlude <Servo.h>
```

Dadurch sagst du dem Arduino, dass du spezielle Befehle für den Servomotor verwenden willst.

Auch für den Servomotor musst du eine Variable definieren. Bisher hast du immer Variablen definiert, die Zahlen gespeichert haben. Man kann aber auch andere Sachen in Variablen speichern. Bei diesem Befehl zum Beispiel, weiß der Arduino später, dass es sich um einen Servomotor handelt:

```
Servo _____ ;
```

Es handelt sich bei der Variable um einen Servomotor.

Schreibe hier den Variablennamen für den Servomotor, den du auch später im Code verwendest.

Wie auch bei anderen Bauteilen, muss der Arduino wissen, an welchem Pin der Servomotor angeschlossen ist. Dafür brauchst du eine Variable, die die Zahl des Pins enthält.

```
int servopin = ____________________ ;
```

Trage hier den Pin ein, an dem der Servomotor angeschlossen ist.

Der Arduino weiß jetzt, dass es einen Servomotor gibt. Er kennt auch die Zahl des Pins. Jetzt musst du noch die zwei Informationen verbinden, damit der weiß, dass an dem Pin der Servomotor angeschlossen ist. Diesen Befehl musst du in den `setup()` - Teil schreiben.

```
____________________ . attach ( ____________________ ) ;
```

Trage hier den Namen des Servomotors ein.

Verbindet den Servomotor am Pin mit der Servomotor-variable

Hier muss der Arduino wissen, an welchem Pin der Servomotor angeschlossen ist.

3. <u>Servomotor bewegen</u>

Jetzt kannst du endlich den Servomotor bewegen. Dazu musst du wissen, dass der Servomotor sich nur zwischen einer Gradzahl von 0 und 179 Grad bewegen kann. Du gibst dem Arduino die Stellung, also die Gradzahl an, auf die sich der Servomotor bewegen soll. Der Befehl gehört an den Anfang im `loop()` - Teil und sieht so aus:

```
____________________ . write ( _________ ) ;
```

Trage hier den Namen des Servomotors ein.

Bewegt den Servomotor auf eine neue Stelle.

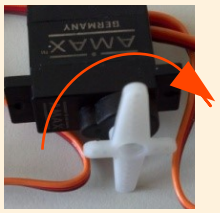
Hier gibst du die Gradzahl an, auf die er sich bewegen soll.

4. <u>Installieren und testen</u>

Trage eine beliebige Gradzahl ein und installiere das fertige Programm. Beobachte, wie der Servomotor sich dreht. Probiere danach eine andere Gradzahl aus und installiere das Programm neu.

AUFGABE 5 – SERVOMOTOR KREISEN LASSEN

Jetzt kannst du den Servomotor also gezielt auf eine Stellung bewegen. Damit die Blume sich am Schluss nach dem Licht richten kann, muss sich die Blume ständig bewegen und dann entscheiden, ob es auf der neuen Stellung heller ist, oder nicht. Als Nächstes programmierst du den Servomotor so, dass er sich durchgängig im Kreis bewegt.



1. Stellungsvariable definieren
   Wie auch das Messergebnis, ist auch die Stellung eine Zahl, die sich ständig verändert. Um die neue Stellung zu berechnen brauchst du eine Variable. Definiere also eine Variable `stellung`, in der du die Stellung speicherst. Weil die Stellung am Anfang 0 Grad ist, ordnest du der Variable die Zahl 0 zu.

2. Servomotor auf Stellung ausrichten
   Der Servomotor soll sich jetzt nicht mehr auf eine bestimmte Gradzahl bewegen, die du angegeben hast, sondern auf die Stellung, die sich immer verändert (und am Anfang noch 0 Grad ist). Tausche deswegen die Gradzahl im Bewegungs-Befehl mit der Variable `stellung` aus.

```
_____________ . write ( stellung ) ;
```

Trage hier den Namen des Servomotors ein.

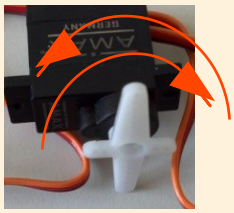
Bewegt den Servomotor auf eine neue Stelle.

3. neue Stellung berechnen
   Der Servomotor soll sich ständig um ein Grad weiterdrehen. Die neue Stellung berechnest du aus der aktuellen Stellung plus 1. So bewegt er sich zuerst auf 1 Grad, dann 2 Grad, dann 3 Grad,...

```
stellung = stellung +1;
```

   Füge den Befehl ein, nachdem sich der Servomotor auf die Stellung bewegt.

AUFGABE 6 – SERVOMOTOR HIN UND HER BEWEGEN



Wenn du dein Programm installierst, sollte sich der Servomotor langsam in eine Richtung bewegen. Oben wurde schon erklärt, dass der Servomotor sich nur auf eine Stellung zwischen 0 und 179 Grad bewegen kann. Auf 179 Grad bleibt er dann stehen. Der Servomotor soll aber auch dann nicht stehen bleiben. Weil er sich nicht weiter drehen kann, soll er sich wieder langsam zurück bewegen. Wie du das machst, erfährst du in dieser Aufgabe.

1. Richtungsvariable

   Anstatt auf die Stellung +1 zu berechnen, soll auf dem Rückweg immer -1 gerechnet werden, bis die Stellung wieder 0 Grad ist. Der Arduino kann sich alleine nicht merken, ob er auf dem Hin- oder Rückweg ist und ob er die Zahl 1 addieren oder subtrahieren muss. Deswegen brauchst du eine Variable, die das tut. Auf dem Hinweg soll diese die Zahl +1 speichern (es soll ja 1 addiert werden) und auf dem Rückweg die Zahl -1 speichern (es soll ja 1 subtrahiert werden). Zuerst befindet sich der Servomotor auf dem Hinweg. Definiere die Variable `richtung`.

```
int richtung = ____________ ;
```

Trage hier die Zahl für den Hinweg ein.

Die Variable speichert am Anfang, dass der Servomotor auf dem Hinweg ist. Wenn der Servomotor bei 179 Grad ankommt, soll sie speichern, dass er auf dem Rückweg ist. Dazu brauchst du einen **Wenn** – Ausdruck.

Wenn der Servomotor bei 179 Grad ankommt

```
If ( stellung == 179 ) {
richtung = ____________________ ; }
```

Speichert die Variable `richtung`, dass sich der Servomotor auf dem Rückweg befindet.

Wenn die Stellung des Servomotors später wieder bei 0 Grad ist, muss die Variable `richtung` speichern, dass sich der Servomotor wieder auf dem Hinweg befindet.

```
If ( _____________ == __________ ) {
richtung = ____________________ ; }
```

Ändere zuletzt noch den Befehl ab, der die neue Stellung des Servomotors berechnet.

```
stellung = stellung + richtung ;
```

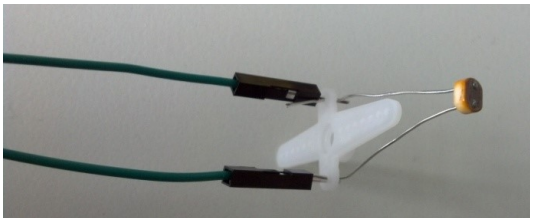
AUFGABE 7 – SONNENBLUME

Super! Jetzt dreht sich der Servomotor hin und her. In dieser Aufgabe wirst du die Sonnenblume basteln, die sich nach der Helligkeit ausrichten soll. Sie ist wichtig, damit der Lichteinfall auf den Helligkeitssensor nur noch aus einer Richtung kommt. Dadurch bekommst du ein genaueres Messergebnis.

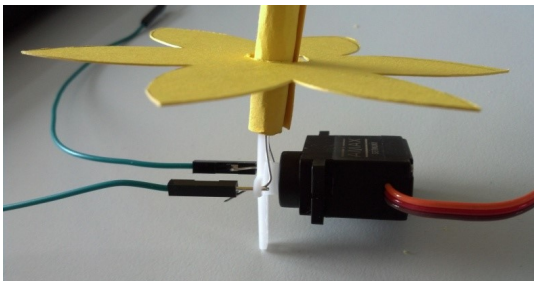
1. <u>Sonnenblume basteln</u>

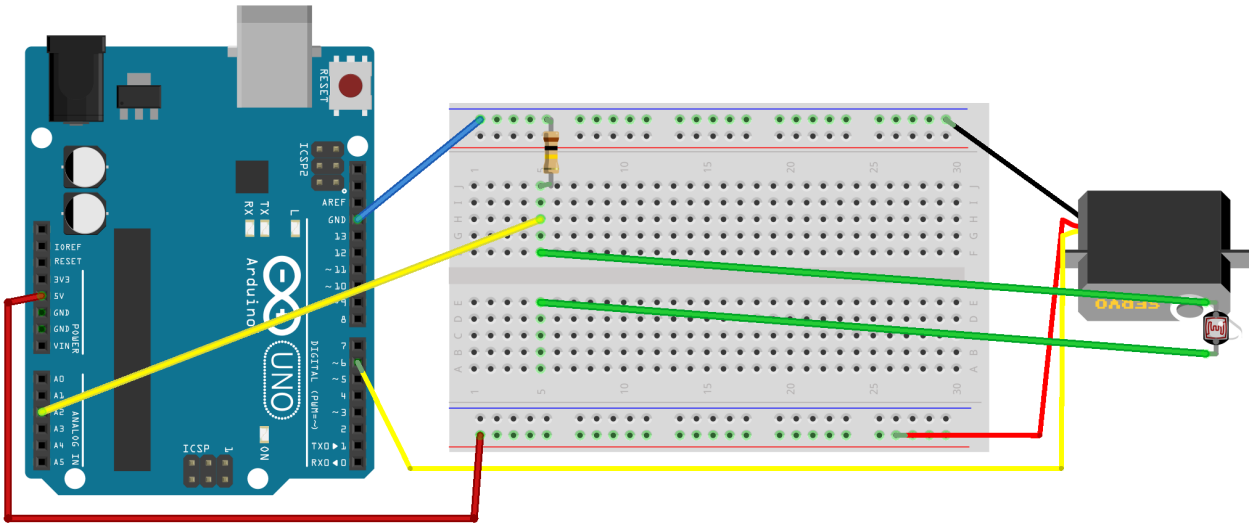
   Schneide zuerst die Sonnenblume aus der Vorlage. Rolle das Rechteck zusammen. Stecke es dann durch das Loch in der Mitte der Sonnenblume.

   Nimm den Helligkeitssensor vom Steckbrett ab. Stecke die Enden des Helligkeitssensors durch zwei gegenüberliegende Löcher des Aufsatzes. Nimm 2 Kabel und stecke ihre Enden in die gleichen Löcher. Der Sensor sollte jetzt von alleine halten.



   Stecke jetzt noch den Aufsatz zurück auf den Servomotor. Stülpe dann die Blume über den Helligkeitssensor. Die freien Enden der Kabel steckst du jetzt an die Stelle, wo zuvor der Helligkeitssensor war.





AUFGABE 8 – SONNENBLUME PROGRAMMIEREN

Als letzten Schritt sollst du die Sonnenblume programmieren. Bisher bewegt sie sich im Halbkreis hin und her. Jetzt soll sie sich abhängig von der Helligkeit bewegen. Wenn ein Messergebnis heller ist, als das vorige, soll die Blume sich einfach weiterbewegen. Ist das Messergebnis dunkler, als das vorige, soll sich die Blume in die andere Richtung bewegen.

1. Helligkeit bestimmt Richtung
   Nachdem die `helligkeit` gemessen wird, brauchst du einen weiteren Befehl.
   Wenn es dunkler wird, soll sich die Richtung des Servomotors ändern.

```
if ( _____________ < ____________________ )
{ richtung = - richtung ;}
```

Hier soll der Arduino erkennen, ob es dunkler wird. Welche zwei Variablen musst du vergleichen?

Die Richtung ändert sich.

2. Programm installieren und testen
   Installiere das Programm auf dem Arduino. Benutze zum Testen auch eine Taschenlampe. Verändere im Code die Pausenzeit und die Gradzahlen so, dass das Programm gut funktioniert. Das heißt, du kannst den Motor in jedem Schritt auch um mehrere Grad drehen.

Super! Du hast es geschafft, die Sonnenblume so zu programmieren, dass sie sich der Sonne nach ausrichtet.