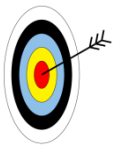
Station 1 – Sonnenblume

Effizientes Sonnenbaden

Für eine Sonnenblume ist es überlebenswichtig, dass sie möglichst viel Sonnenlicht aufnehmen kann. Darum dreht sie sich zur Sonne, indem sie sich langsam zum hellsten Punkt am Himmel ausrichtet. Genauso müssen sich Solarzellen ausrichten, damit möglichst viel Sonneneinstrahlung in Strom umgewandelt werden kann.


[1]



Mit Hilfe dieses Arbeitsblattes...

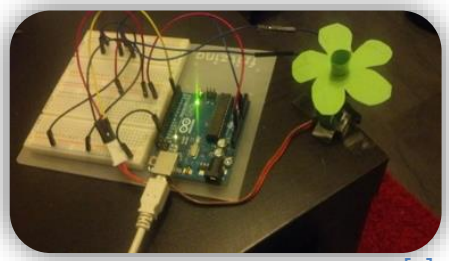
- baut ihr eure eigene Sonnenblume.
- lernt ihr, wie sich elektronische Bauteile kombinieren lassen.
- erfahrt ihr, wie man diese Bauteile kombiniert, um die Blume zur Lichtquelle zu drehen.


[2]

Benötigte Bauteile

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

- einen Helligkeitssensor (Abbildung 4)
- einen 100 kΩ-Widerstand (braun-schwarz-gelb)
- einen Servo-Motor mit Kreuzaufsatz (Abbildung 6)
- 3 blaue, 2 rote, 2 gelbe, 2 grüne Kabel
- eine Papierblume


[3]

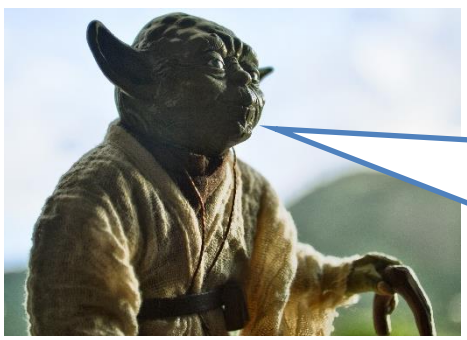

[4]


[5]

Zusätzlich benötigt ihr noch eine Taschenlampe als Lichtquelle, um euren Aufbau zu testen.


[6]


[7]


[8]

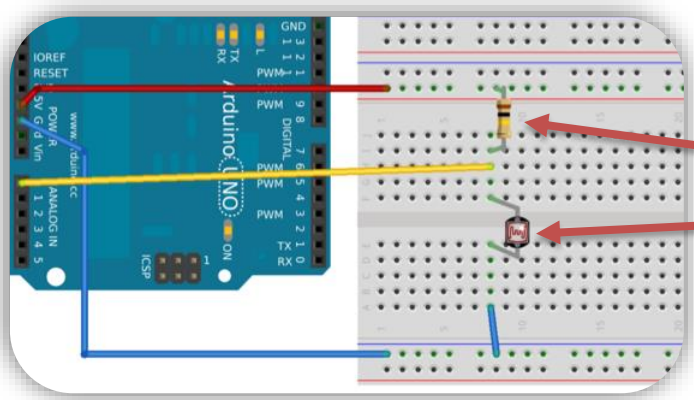
Damit eure Blume ihr gezielt bewegen könnt, mit jedem Bauteil vertraut euch machen ihr solltet. Dabei euch folgende Seiten helfen werden.

Station 1 – Sonnenblume

Mit dem **Helligkeitssensor** kann eure Sonnenblume erkennen, ob es heller oder dunkler wird und kann dadurch merken, ob sie sich zur Sonne hin oder von der Sonne wegbewegt.

Helligkeitssensor bzw. Lichtwiderstand

Der Helligkeitssensor, den ihr hier verwendet, 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.

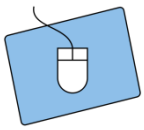


[9]

Benötigte Bauteile

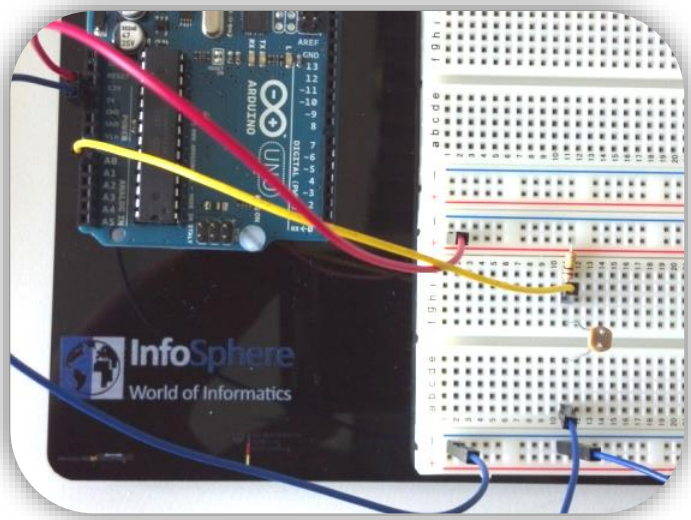
Um den Helligkeitssensor zu testen, benötigt ihr folgende Bauteile:

- einen 100 kΩ-Widerstand (braun-schwarz-gelb)
- einen Helligkeitssensor
- 2 blaue, 1 rotes, 1 gelbes Kabel



Schaut euch Abbildung 9 und 10 an. Dort seht ihr, wie die Schaltung später aussehen soll.

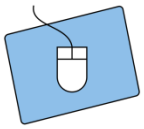
Die Schaltung funktioniert so ähnlich wie bei einer LED. Zusätzlich müsst ihr aber ein Kabel direkt zwischen Widerstand und Helligkeitssensor stecken und mit einem analogen Eingang verbinden.



[10]

Nun könnt ihr anfangen, zu programmieren und mit dem Helligkeitssensor zu arbeiten. Dazu erklären euch die Anweisungen auf der nächsten Seite, wie ihr mit Hilfe des **Serial Monitors** herausfinden könnt, welche Signale am analogen Pin ankommen.

Station 1 – Sonnenblume

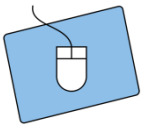


1. Erstellt einen neuen Arduino-Sketch und gebt ihm einen sinnvollen Namen.
2. Erstellt (falls nicht schon vorhanden) das Grundgerüst aus dem Einstiegsprojekt.
3. Als Erstes müssen im Bereich **Einstellungen** alle **Variablen** erstellt werden, die ihr benötigt. Gebt ihnen sinnvolle Namen!
   Ihr braucht:
   a. Eine `int`-Variable für den Pin, an dem ihr den **Helligkeitssensor** angeschlossen habt. Der Wert der Variablen muss die Nummer das Analogpins sein.
   Hinweis: Auch wenn die analogen Pins mit A0, A1, usw. bezeichnet sind, benutzt man für die Zuweisung im Sketch nur den Zahlenwert!
   Pin-Variablen-Name: _____
   b. Eine `int`-Variable, die eine Zahl für die Helligkeit speichern soll, mit Wert 0.
   Helligkeits-Variablen-Name: _____
4. In `setup()` müsst ihr den **Serial 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.
5. In `loop()` könnt ihr nun den analogen Pin auslesen und den Inhalt in eurer Helligkeits-Variable aus 3.b. speichern:
   ```
   helligkeitsvariable = analogRead(pinvariable);
   ```
6. Damit auch etwas an den Computer gesendet wird, solltet ihr über den **Serial Monitor** direkt die Variable für die Helligkeit ausgeben.
7. 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 Serial Monitor folgen könnt. Denkt daran **1 Sekunde = 1000 Millisekunden**.
8. Jetzt könnt ihr mit dem Testen beginnen. Vermutlich müsst ihr dabei noch ein paar Flüchtigkeitsfehler beheben. Die passieren fast jedem und fallen erst beim Testen auf. Startet das Programm und öffnet danach den Serial Monitor. Werden die Zahlen immer noch zu schnell ausgegeben, erhöht den `delay`–Wert weiter.
9. Beleuchtet den Kopf des Helligkeitssensors und schaut euch die ausgegebenen Werte auf dem Serial Monitor an. Schirmt den Sensor mit der Hand oder einem Blatt Papier ab.
   Hinweis: In der Wirklichkeit ist es nie gleich hell. Eure Messwerte werden also immer ein klein wenig hin und her springen.
   Notiert eure Erkenntnisse:

Dunkel:	Hohe Werte	☐
	Niedrige Werte	☐
Hell:	Hohe Werte	☐
	Niedrige Werte	☐

Station 1 – Sonnenblume

Nun wisst ihr, in welchem Bereich die Werte des Helligkeitssensors liegen und wie ihr erkennt, ob es hell oder dunkel ist. Aber die Blume muss 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.



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 Serial 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");
   }
   ```

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

   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 Variablen:
   `alteHelligkeitsVariable = gemessenerWert;`
4. Testet euer Programm und schaut, was auf dem Serial 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.
5. Speichert euren Sketch! Ihr werdet ihn später für die Blume benötigen.

Station 1 – Sonnenblume

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

Benötigte Bauteile

- ein Servo-Motor mit **Kreuzaufsatz** (den ihr draufsetzten müsst)
- 1 blaues, 1 rotes, 1 gelbes Kabel

Mit dem **Servo-Motor** könnt ihr eure Sonnenblume bewegen. Hier wird euch seine Funktionsweise erklärt.

Der 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.

[6]

Ihr könnt dem Motor nur sagen, dass er sich **zu** einem bestimmten Winkel drehen soll, **nicht um** einen bestimmten Winkel.

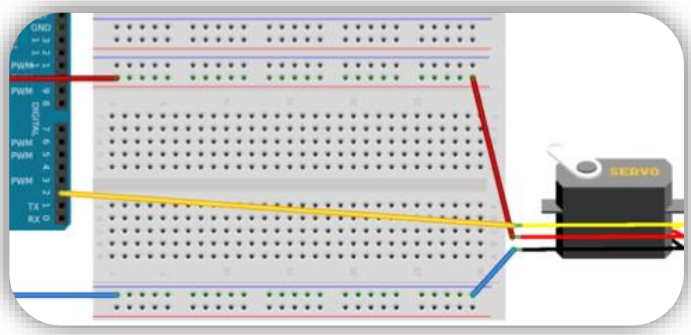
Der Aufbau der Schaltung

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

1. Schließt die Kabel an den Servo-Motor an:
   - Braun → Blau
   - Rot → Rot
   - Orange → Gelb

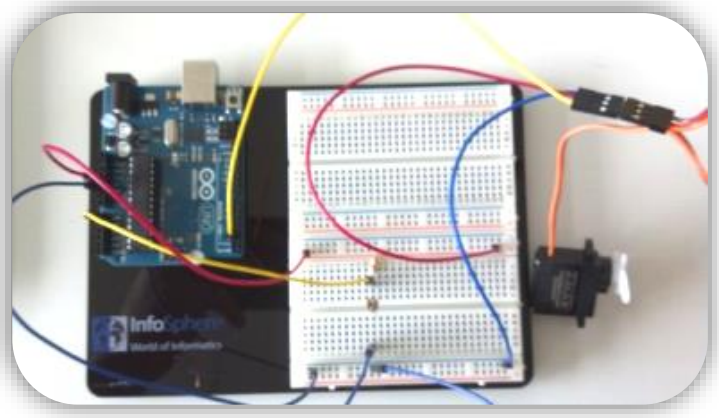
[11]

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



[12]

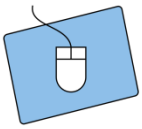
Auf diesem Foto seht ihr, wie die beiden Schaltungen nebeneinander aufgebaut aussehen:



[13]

Die Programmierung

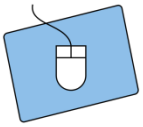
Nun könnt ihr damit anfangen, den Arduino so zu programmieren, dass er den Motor bewegen kann. Dazu werdet ihr zuerst lernen, wie ihr das Programm entsprechend vorbereitet und danach, wie ihr den Motor schrittweise bewegt.



1. Öffnet einen neuen Arduino-Sketch und speichert ihn unter einem sinnvollen Namen.
2. Erstellt das Grundgerüst aus dem Einstiegsprojekt, falls es nicht automatisch erstellt wurde.
3. 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 **Semikolon**.
4. Jetzt könnt ihr wie bei einer Variablen einen `Servo` im Programm anlegen. Damit könnt ihr später alle `Servo`-Funktionen benutzen. Hier wird kein Startwert zugewiesen:
   ```
   Servo servoName;
   ```
5. Wie bei jedem Bauteil, das gesteuert wird, braucht ihr eine **`int`**-Variable, die den Pin für den Servo-Motor speichert.
   Servo-Pin-Name: _____

Weiter geht's auf der nächsten Seite…

Station 1 – Sonnenblume



6. In `setup()` müsst ihr nun dem Programm mitteilen, an welchen Pin der Servo angeschlossen wurde, indem ihr dies einfügt:
   `servoName.attach(servoPin);`
7. 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.
8. Testet das Programm und behebt eventuelle Flüchtigkeitsfehler.
9. Wiederholt Schritt 7 und 8 mit verschiedenen Zahlen und findet heraus, wo sich der Motor überall hinbewegen kann.

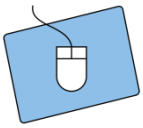
Notiert eure Erkenntnisse:
Beim Erhöhen der Gradzahl dreht sich der Motor…

im Uhrzeigersinn.  ☐

gegen den Uhrzeigersinn.  ☐

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.

Station 1 – Sonnenblume

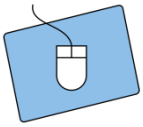


1. Erstellt eine neue **`int`**–Variable, die später die neue **Gradzahl** bzw. Position für den Motor speichern soll, und gebt ihr einen sinnvollen Namen. Weist ihr den Startwert 0 zu.
   Servo-Positions-Variablen-Name: _____
2. In **`loop()`** soll nun die Position jedes Mal um 1 erhöht werden. Eine 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 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 Positionsvariablen.
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`, dann wird der Motor…
   langsamer. ☐

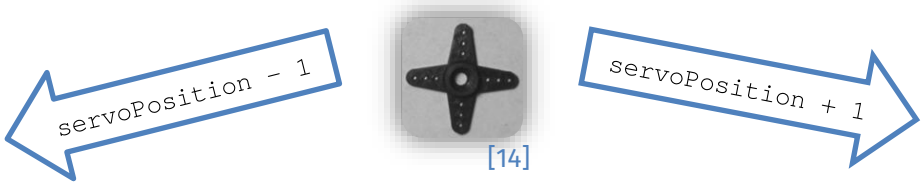
   schneller. ☐

Station 1 – Sonnenblume

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 Variablen 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.



[14]

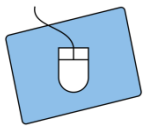
2. Diese Variable solltet ihr nun in eurer Positionsberechnung benutzen, indem ihr einfach das +1 durch eure Richtungsvariable ersetzt:
```
servoPosition = servoPosition + richtung;
```
3. Da die Richtung in eurem Programm den Startwert 1 hat, sollte sich an der Funktionsweise eures Programms nichts 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.
6. Speichert euren Sketch! Ihr werdet ihn später für die Blume benötigen.

Gut gemacht! Ihr wisst nun alles Nötige und könnt endlich die **Schaltung für die Sonnenblume** aufbauen und euer (Kabel-)Gewächs programmieren. Damit das Ausrichten wirklich genau wird, muss der Helligkeitssensor so abgeschirmt werden, dass (das meiste) Licht nur noch aus einer Richtung auf den Lichtwiderstand fällt. Ansonsten ist der Helligkeitssensor zu ungenau.

Benötigte Bauteile

- 2 grüne Kabel
- 1 Sonnenblumen-Schnittmuster

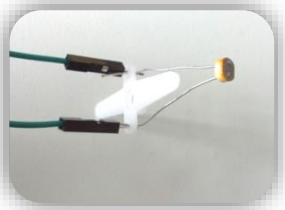
Station 1 – Sonnenblume



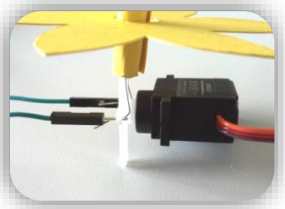
Befestigung des Sensors

1. Schneidet die Sonnenblume aus, rollt das Rechteck zusammen und steckt es durch das Loch in der Mitte der Blume.
2. Nehmt den Helligkeitssensor vom Steckbrett.
3. Befestigt ihn am Servo-Motor, indem ihr die Drähte durch jeweils ein Loch an gegenüberliegenden Armen des Motors schiebt. Achtet dabei darauf, dass **die Drähte sich nicht berühren**, sonst funktioniert eure Schaltung nicht mehr.
4. Steckt dann die grünen Kabel in dieselben Löcher, so halten Kabel und Sensor von ganz alleine.
5. Stülpt anschließend die Blume über den Sensor und einen Arm des Servokreuzes.

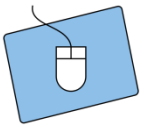
[15]



[16]

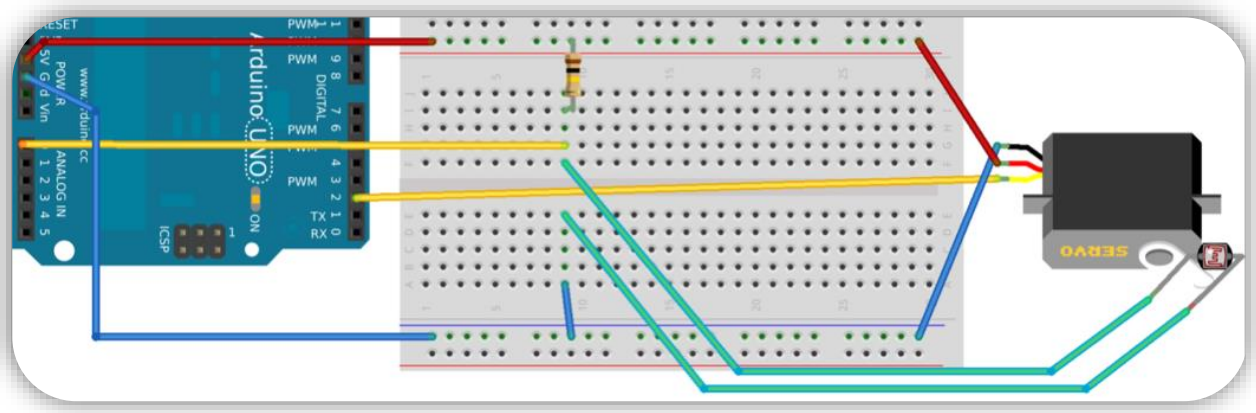


[17]



Aufbau der Schaltung

Um die Sonnenblumenschaltung zusammenzustecken, müsst ihr noch die grünen Kabel dort einstecken, wo vorher der Helligkeitssensor eingesteckt war.
Unten seht ihr eine Übersicht der kompletten Schaltung.

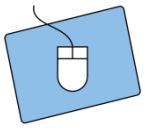


[18]

Station 1 – Sonnenblume

Die Funktionsweise der Blume

Super, jetzt wisst ihr alles, um die Sonnenblume zu steuern. Die Blume soll sich immer in kleinen Stücken bewegen und ihre Bewegungsrichtung ändern, wenn es dunkler wird. Ansonsten behält sie die Richtung bei.



1. Öffnet den zweiten Servo-Motor-Sketch und benutzt die „Speichern unter“-Funktion, um ihn unter einem neuen aussagekräftigen Namen zu speichern. So müsst ihr nicht alles neuschreiben, sondern könnt euer Servo-Motor-Programm erweitern. Eure Sonnenblume ist jetzt fast fertig.
2. 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 Programm für den Helligkeitssensor und passt **`setup()`** entsprechen an.
   b. Benutzt das, was ihr über den Helligkeitssensor gelernt habt, um **`loop()`** anzupassen, sodass euer Programm grob so funktioniert:

```
void loop(){
1. Messe und speichere Helligkeitswert
2. Wenn dunkler, ändere Richtung
3. Wenn Position 179 oder 0: ändere Richtung
4. Berechne neue Position für Servo
5. Bewege Servo
6. Warte bis zu Ende gedreht
7. Speichere Helligkeitswert in Variable für alten
   Helligkeitswert
}
```

3. Testet euer Programm mit der **Taschenlampe** (eures Smartphones). Beobachtet auch die Helligkeitswerte auf dem **Serial 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. Bei schlechten Lichtverhältnissen hilft es, den Motor in jedem Schritt um **mehrere Grad** drehen zu lassen.

Geschafft! Klasse! Jetzt habt ihr eure eigene Sonnenblume zusammengeschraubt. Ihr könnt entweder versuchen, eure Blume zu optimieren oder ihr versucht euch an einer anderen Station.

Quellenverzeichnis:

Abb. 1 – *Quelle: pxhere.com (https://pxhere.com/de/photo/1448455), Autor: sandmann 8888, CC0 1.0 Universal (CC0 1.0) Public Domain Dedication (https://creativecommons.org/publicdomain/zero/1.0/), abgerufen am: 22.06.2022*

Abb. 2 – *Quelle: pxhere.com (https://pxhere.com/de/photo/546340), CC0 1.0 Universal (CC0 1.0) Public Domain Dedication (https://creativecommons.org/publicdomain/zero/1.0/), abgerufen am: 22.06.2022*
Abb. 3 bis 7, 10, 11, 13 bis 17 *– Quelle: InfoSphere, CC BY-SA 4.0 Attribution-ShareAlike 4.0 International (https://creativecommons.org/licenses/by-sa/4.0/)*
Abb. 8 – *Quelle: pxhere.com (https://pxhere.com/de/photo/901512), CC0 1.0 Universal (CC0 1.0) Public Domain Dedication (https://creativecommons.org/publicdomain/zero/1.0/), abgerufen am: 22.06.2022*
Abb. 9, 12, 18 *– Quelle: Screenshots der Fritzing-Software (https://fritzing.org/), CC BY-SA 3.0 Attribution-ShareAlike 3.0 Unported (https://creativecommons.org/licenses/by-sa/3.0/), abgerufen am: 14.06.2022.*

– Quelle: InfoSphere, CC BY-SA 4.0 Attribution-ShareAlike 4.0 International (https://creativecommons.org/licenses/by-sa/4.0/)