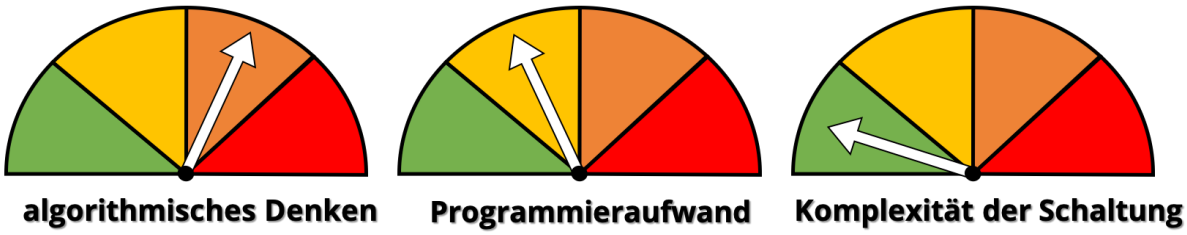
TECHNISCHE UNIVERSITÄT DRESDEN

MicroPython

Station 1 | Sonnenblume



Valentin_Photography, pixabay.com , Pixabay Lizenz

Professur für
Didaktik der Informatik

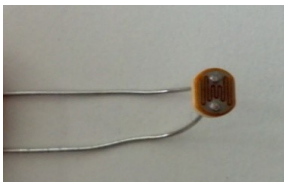
UM WAS WIRD ES IN DIESER STATION GEHEN?

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

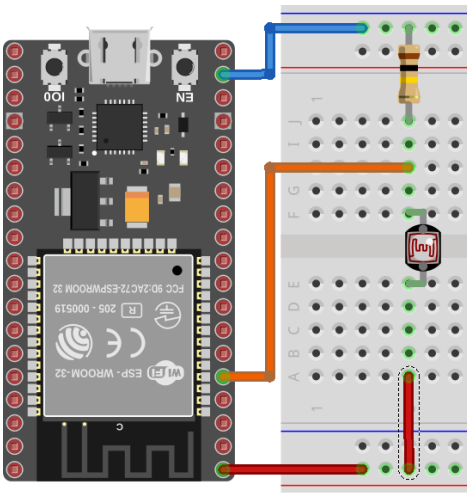
Zusätzlich zum ESP32 und dem Steckbrett brauchst du folgende Bauteile:

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. Verwende für dieses Beispiel den **Pin 34**.

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.

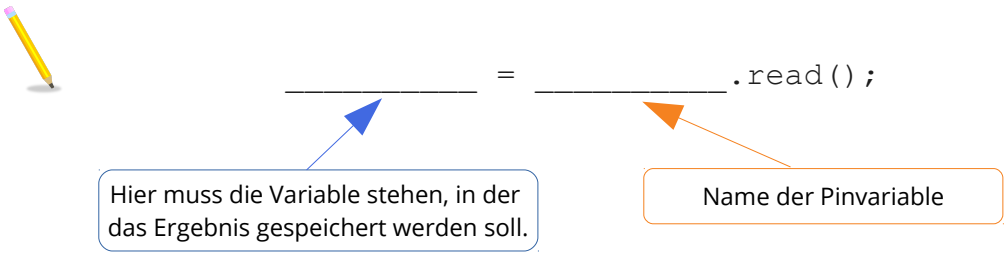
1. Pin-Variable definieren
   Wie in der Einführungsstation musst du auch hier den ESP32 wissen lassen, an welchem Pin die Messung stattfinden soll. Definiere dafür eine Variable. Falls du dafür Hilfe brauchst, schaue dir noch einmal in der Einstiegsstation Aufgabe 3 an.

   Wir wollen in diesem Beispiel nicht bloß wissen, ob am Pin eine Spannung anliegt, sondern interessieren uns auch für ihren Wert. Aus diesem Grund müssen wir eine analoge Messung durchführen. Das kannst du dem ESP32 ganz einfach sagen, indem du den angegebenen Pin in die Klammern des Befehls `ADC()` schreibst.

2. 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:

   ```
   helligkeit = 0;
   ```

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



   Schreib den Befehl in eine `while True` – Schleife.

Du hast das Programm erfolgreich installiert, aber es passiert nichts? Keine Sorge! Der ESP32 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?

Das kleine Fenster unterhalb des Thonny Code-Editors, die **Konsole**, hast du in der vergangenen Station kennengelernt und vielleicht bereits genutzt, um Fehler mithilfe der angezeigten Fehlermeldungen zu finden.
Du kannst die Konsole allerdings auch dazu nutzen, eigene Nachrichten auszugeben. Nutze hierfür den `print()` – Befehl:

```
print("Das ist ein Test")
```

5. Anzeigen des Ergebnisses
   Verwende den `print()` – Befehl zur Anzeige des gemessenen Helligkeitswerts. Überlege, welche Funktion die Anführungszeichen im Beispiel erfüllen und trage die auszugebende Variable in korrekter Form ein:

```
print( _____ )
```

Variable, deren Wert angezeigt werden soll

   Teste dein Programm. Benutze eine Taschenlampe, um die Helligkeit am Sensor zu verändern. Wie verhalten sich die angezeigten Werte bei wechselnder Helligkeit?

1. Pausen zwischen den Messungen
   Wie du vielleicht schon bemerkt hast, sieht man innerhalb kürzester Zeit sehr viele Messergebnisse. Der ESP32 arbeitet sehr schnell und führt die Messungen wiederholt in sehr kurzer Zeit aus. Das kann u.A. dazu führen, dass die auszugebenden Werte „anstauen“ und du sie in der Konsole mit großer Verzögerung angezeigt bekommst.
   Befiehl dem Board daher, nach jeder Messung eine Pause zu machen. Welchen Befehl musst du dafür benutzen? Du kennst ihn bereits!

   _____

   Importiere nun noch das zur Ausführung dieses Befehls notwendige **Modul** am Anfang deines Programms.
   Teste dein Programm und verändere die Länge der Pause, bis du alle Messergebnisse lesen kannst.

   Dein Programm funktioniert nicht? Kein Problem! Auf der letzten Seite findest du mögliche Fehler und wie du sie ausbessern kannst.

AUFGABE 2 – WIRD ES HELLER ODER DUNKLER?

Jetzt weiß der ESP32, 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 das Board 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`). Weise ihr am besten den Startwert `0` zu.

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:

```
_____________ = _____________ ;
```

Variable, die altes Ergebnis speichert.

Variable, die neues Ergebnis speichert.

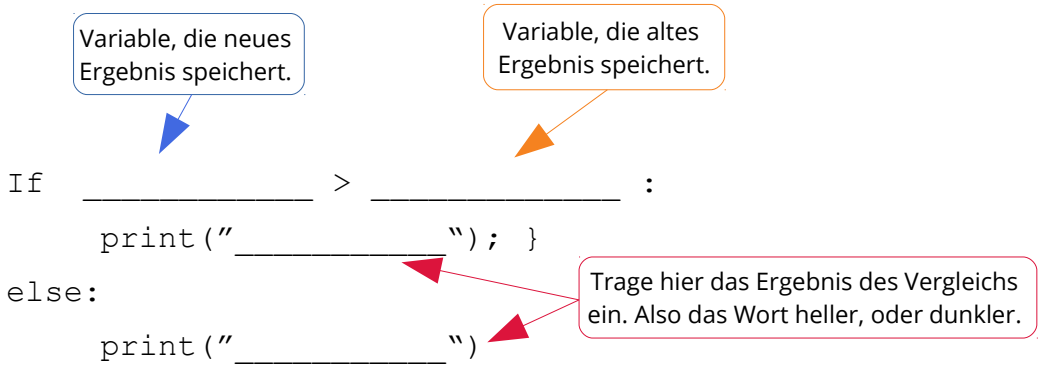
   Füge den Befehl noch vor den anderen Befehlen in die `while True` – Schleife ein.

```
While True:

    _____________ = _____________.read()
    print( _____________ )
```

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. Testen

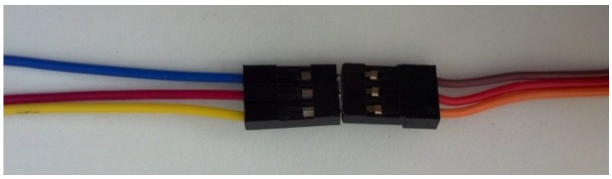
Beobachte in der Konsole, ob der ESP32 richtig entscheidet, ob es heller oder dunkler wird. Benutze zum Testen wieder die Taschenlampe, um die Helligkeit zu verändern. Falls es dir zu schnell geht, verlängere noch einmal die Pause.

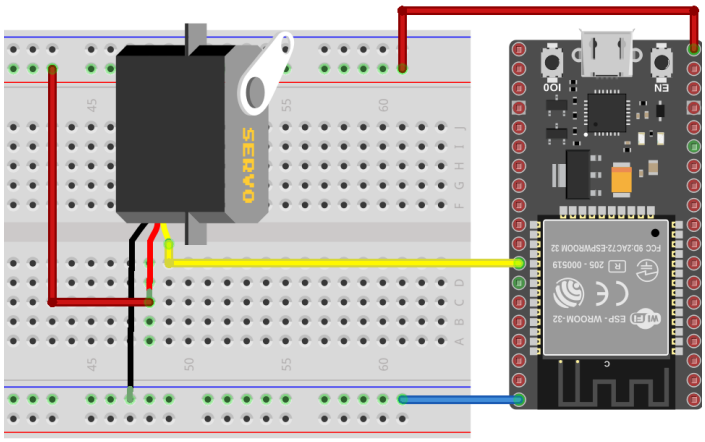
AUFGABE 3 – SERVOMOTOR BEWEGEN

Da der ESP32 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. Modul importieren
   Um einen Servomotor unter MicroPython mit Winkeln ansteuern zu können, benötigst du das Modul `servo.py`.
   Aus diesem musst du die Funktionen `setServoPin` und `setAngle` importieren.

3. Servomotor vorbereiten
   Wie bei anderen Bauteilen auch, musst du dem ESP32 mitteilen, an welchen Pin du den Servo angeschlossen hast. Aus diesem Grund hast du die Funktion `setServoPin()` importiert.

```
setServoPin( ______ )
```

Trage hier den Nummer des Pins ein, an dem der Servomotor angeschlossen ist.

4. Servomotor bewegen
   Jetzt kannst du den Servomotor bewegen! Wichtig zu wissen, ist, dass dieser sich nur zwischen einer Gradzahl von 0 und 180 bewegen kann. Verwende die Funktion `setAngle()`, um den Servo an eine gewünschte Position zu bewegen.

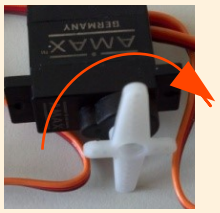
```
setAngle( ______ )
```

Winkel, an den der Servomotor bewegt werden soll

5. Programm testen
   Trage eine beliebige Gradzahl ein und teste das fertige Programm. Beobachte, wie sich der Servomotor dreht. Probiere danach einen anderen Winkel aus und starte das Programm neu.

AUFGABE 4 – 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 sie sich 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 den Wert `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.

```
setAngle( stellung )
```

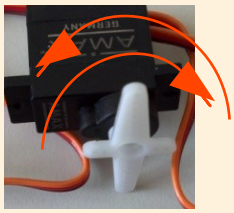
4. neue Stellung berechnen
   Der Servomotor soll sich ständig um ein Grad weiterdrehen. Die neue Stellung berechnest du aus der aktuellen Stellung plus 1. So bewegt er sich zuerst auf 1 Grad, dann 2 Grad, dann 3 Grad,...

```
stellung = stellung + 1
```

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

   Integriere beide Befehle in eine `while True` – Schleife, damit sich der Servomotor kontinuierlich bewegt.

AUFGABE 5 – 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 180 Grad bewegen kann. Auf 180 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 ESP32 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 übernimmt. Auf dem Hinweg soll sie die Zahl +1 speichern (es soll ja 1 addiert werden) und auf dem Rückweg die Zahl -1 (es soll ja 1 subtrahiert werden). Zuerst befindet sich der Servomotor auf dem Hinweg. Definiere die Variable `richtung`.

```
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 180 Grad ankommt, soll sie speichern, dass er auf dem Rückweg ist. Dazu brauchst du einen **Wenn** – Ausdruck.

Wenn der Servomotor bei 180 Grad ankommt ...

```
if  stellung == 180 :
    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
```

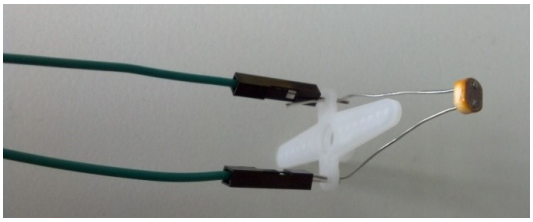
AUFGABE 6 – SONNENBLUME

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.

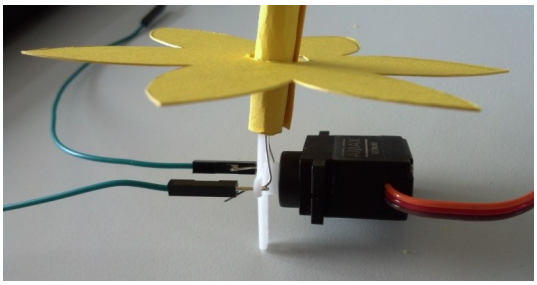
1. <u>Sonnenblume basteln</u>

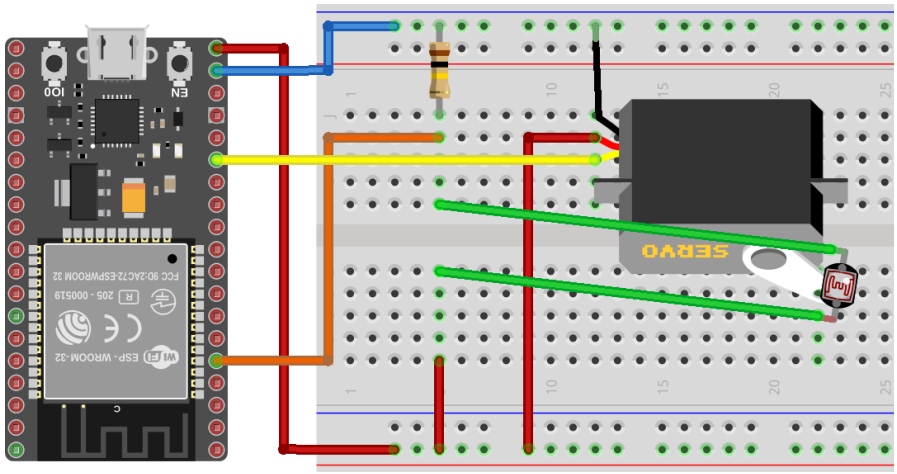
   Schneide zuerst die Sonnenblume aus der Vorlage. Rolle das Rechteck zusammen. Stecke es dann durch das Loch in der Mitte der Sonnenblume.

   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 7 – 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
```

Die Richtung ändert sich.

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

2. Programm installieren und testen
   Teste das Programm auf dem ESP32. Benutze hierfür auch wieder 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.

Fotos: RWTH Aachen, InfoSphere
Screenshots: fritzing electronics made by easy (Linux)
Alle weiteren Grafiken: Patrick Binkert, EduInf@TUD