# Inhaltsverzeichnis

[​ Dokumentation Arduinoprogramm 1](#__RefHeading___Toc333_2356351772)

[1 Aufgabenstellung 1](#__RefHeading___Toc335_2356351772)

[2 Aufbau Solarturm 1](#__RefHeading___Toc337_2356351772)

[3 Knopfsteuerung 2](#__RefHeading___Toc339_2356351772)

[4 Zeitsteuerung 3](#__RefHeading___Toc236_3198475482)

[​ Berechnungen zur zeitgesteuerten Führung 3](#__RefHeading___Toc238_3198475482)

[​ Programmkonzept zeitgesteuerte Führung 3](#__RefHeading___Toc345_2356351772)

[5 Probleme 4](#__RefHeading___Toc434_3727435720)

[​ Nicht genug Strom für die Relaisbetätigung 4](#__RefHeading___Toc248_4088067760)

[​ Automatisches aktivieren der Randsensoren 4](#__RefHeading___Toc436_3727435720)

[​ Überlauf der Zyklusdauervariable 4](#__RefHeading___Toc240_3198475482)

# Dokumentation Arduinoprogramm

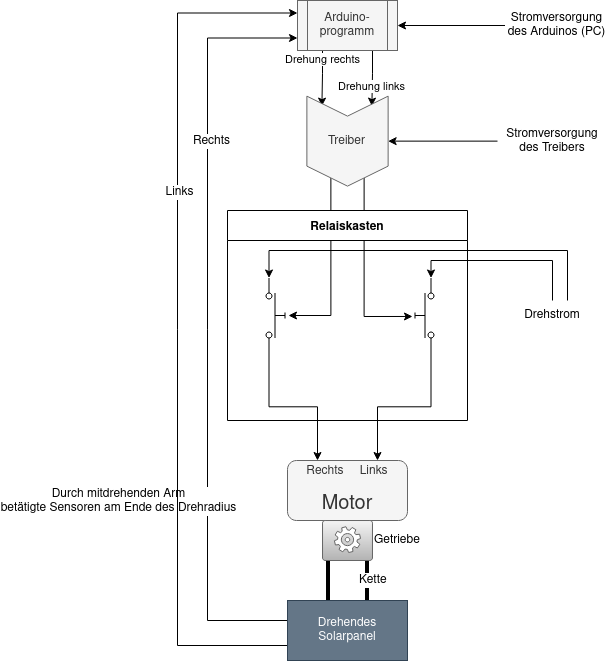
## Aufgabenstellung

Ein Turm mit angewinkelter Solarzelle soll für einen möglichst hohen Effizienzgrad der Sonne Nachgeführt werden, und bei berühren der Begrenzungssensoren an den Seiten angehalten werden.

## Aufbau Solarturm

Der Turm soll mithilfe eines Arduino Microcontrollers kontrolliert werden. Der Mikrocontroller steuert zunächst mit der gewünschten Drehrichtung eine Treiberstufe an, die nachträglich eingesetzt wurde, und mit einer externen Stromversorgung ausgerüstet ist, da der Strom des Arduinos selber nicht ausreicht, um die in Diagramm 1 aufgeführte, nachfolgende Relaisstufe anzusteuern. Mithilfe dieser Relais wird die Drehstromversorgung des Motors freigeschaltet, wodurch dieser sich dreht, und über ein reduzierendes Getriebe eine Kette antreibt, die das Solarmodul dreht.

An dem Modul befestigt ist ein Arm, der sich synchron mit dem Modul mitdreht, dieser betätigt Sensoren jeweils links und rechts der Drehrichtung, sie dienen als Limitierer und sollen die Drehung jenseits von ihnen unterbinden, weshalb sie ebenfalls mit dem Arduino verbunden sind

Diagramm 1: Aufbau des Solarturms

## Knopfsteuerung

Zunächst muss mithilfe eines Schaltplans des Turms versucht werden, die Ein- und Ausgänge zu identifizieren, da dies nötig ist um sie korrekt anzusteuern.

Darauf folgend habe sollte, um Funktionalität des Turms zu prüfen, versucht werden zunächst nur mithilfe von Knöpfen das Panel zu drehen (inkl. automatisches Anhalten bei Sensorberührung), und Werte auszumessen die für die Programmierung relevant sind.

## Zeitsteuerung

### Berechnungen zur zeitgesteuerten Führung

Als nächstes muss die Entwicklung der Steuerung nach Zeit umgesetzt werden, dafür war zunächst wichtig, wie lange eine volle Drehung von einem Sensor zum nächsten braucht, da der Motor nicht wie ein Servomotor nach Winkel steuerbar ist, sondern ausschließlich nach Zeit steuerbar ist.

Aus dem Messwert der Drehdauer, ca. 43 Sekunden, lassen sich anhand der Länge des Tages und der Menge an Sonnenstunden (hier verwendet wurden 8) errechnen, wie viele Sekunden Motorlaufzeit, zu wie vielen Stunden und Minuten korrespondieren.

Für den Programmablauf relevante Variablen waren die Tageslänge in Sekunden, 86400 Sekunden, die Zeit die das Modul für eine Rotation braucht (43 Sekunden, im Programm mit 2 Sekunden Toleranz, also 45 Sekunden), und den Anteil des Tages, der Sonneneinstrahlung hat, was bei 8 Stunden Sonne 86400 Sekunden / 4 = 21600 Sekunden entspricht.

Danach war auch wichtig, wie viele Laufintervalle des Motors wir durchführen wollen. Da dies ein eher alter Motor ist, sind viele, kurze, hochfrequente Anläufe eher ineffizient, weshalb ich die Sonnenstunden in 8 Intervalle aufgeteilt habe.

Die Motorlaufdauer eines Intervalls entspricht somit 45 Sekunden / 8 = 5,625 Sekunden

Weiterhin ist auch das Verhältnis zwischen Motorlaufdauer und Sonnenstunden wichtig, da dies nötig ist um die Dauer von einem Intervallbeginn bis zum nächsten zu errechnen. Dies ergibt sich aus 45 Sekunden / 21600 Sekunden ≈ 0,0021.

Wenn wir nun die Intervallzeit durch diesen Wert dividieren erhalten wir 5,625 / 0.0021 ≈ 2679. Daraus folgt, dass von einem Intervallbeginn bis zum nächsten ungefähr 2679 Sekunden, also ungefähr 44 Minuten und 39 Sekunden, vergehen.

### Programmkonzept zeitgesteuerte Führung

Bei Beginn des Programms wird das Panel zuerst ganz nach links gedreht, bis der Randsensor berührt wird. Damit wird das Panel morgens auf die Richtung des Sonnenaufgangs ausgerichtet. Direkt darauf folgend dreht sich das Panel für das erste Intervall, und bleibt dann in dieser Stellung bis die ca. 45 Minuten abgelaufen sind. Daraufhin dreht sich das Panel erneut für die vorgegebene Zeit. Dies wird so lange wiederholt, bis der Arm auf den Sensor auf der rechten Seite trifft. Danach „schläft“ das Panel für den Rest des Tages, bis seit Programmbeginn 24 Stunden abgelaufen sind. Dann dreht sich das Panel erneut vollständig nach links, und der Zyklus beginnt von vorne.

## Probleme

### Nicht genug Strom für die Relaisbetätigung

Bei der Implementation der Knopfsteuerung ist früh das Problem aufgetreten, dass der Arduino über nicht genug Strom an den Ausgängen verfügt, um die Relais richtig anzusteuern, was in einem in schneller Abfolge wiederholtem Neustarten des Mikrocontrollers resultierte.

Dieses Problem lies sich mit einer eingesetzten Treiberstufe beheben, die mit einer externen Stromversorgung unter Anweisung des Mikrocontrollers die Relais betätigen konnte.

Nach der Lösung von diesem Problem funktionierte die Knopfsteuerung einwandfrei.

### Automatisches aktivieren der Randsensoren

Am Anfang der Umsetzung der Zeitsteuerung trat das Problem auf, dass das Intervall zu früh beendet wurde, und der ganze Zyklus ebenfalls abgebrochen wurde.

Da nach mehrstündiger Suche kein Fehler in dem Programm gefunden worden ist, lag es Nahe, dass es ein Hardwareproblem ist.

In der Tat, die Randsensoren haben häufig ohne Betätigung für einen sehr kleinen Zeitraum ausgelöst. Durch das Einsetzen eines weiteren Debounce-Delays wurde auch das behoben.

### Überlauf der Zyklusdauervariable

Zunächst ist das Problem aufgetreten, dass die Autorotation sich nach dem ersten Intervall selbst zurückgesetzt hat. Dies ließ sich darauf zurückführen, dass ich zum festhalten des Beginns der Zykluszeit, die anders als die Programmstartzeit ist, da das Panel bei Programmbeginn mehrere Sekunden braucht um in Ausgangsposition zu gelangen, eine Variable des Typs int verwendet habe. Eine normale integer Variable im Arduino-System hat eine Speicherkapazität von 16 Bits, und ist signed, das heißt ein Bit ist für die Definition des Vorzeichens festgelegt. Eine 16 Bit, signed Integer hat somit eine Reichweite von -32 768 bis 32 767 [[1]](#footnote-2). Wenn wir die im Arduino eingebaute Funktion millis(), zur Abrufung der Laufzeit in Millisekunden nutzen, dann tritt nach ungefähr 33 Sekunden einen Überlauf[[2]](#endnote-2) der Variable auf.

Das Problem ließ sich beheben, in dem ich statt der signed 16 Bit Integer eine 32 Bit, unsigned (Vorzeichenlose) Integer verwendet habe. Diese hat eine Reichweite von 0 bis 4 294 967 295 [[3]](#footnote-3).

1. https://www.arduino.cc/reference/en/language/variables/data-types/int/ [↑](#footnote-ref-2)
2. Da Variablen in Binärformat gespeichert werden, findet bei der Inkrementation über die Dateispeichergröße ein sogenannter Arithmetischer Überlauf statt.

   Gewöhnlicherweise läuft ein Bit auf den nächsten über, wenn er bereits auf 1 steht und erneut inkrementiert wird. Dabei wird dieser Bit auf 0 gesetzt, und der nächstfolgende Bit inkrementiert, ähnlich wie bei einem Hochzählen von 9 auf 10.

   Bei einem Arithmetischen Überlauf gibt es keinen Nächsthöheren Bit, weshalb alle Bits auf 0 gesetzt werden. [↑](#endnote-ref-2)
3. https://www.arduino.cc/reference/en/language/variables/data-types/unsignedlong/ [↑](#footnote-ref-3)