Anleitung zur

TricklerGUI 1.0

[ˈtrɪklɛ] s Tröpfler, Riesler....

Ein Grafisches User Interface für Arduino und Windows.

Vielen Dank, dass Sie sich für den Trickler entschieden haben.....hahaha.
Vorweg erst mal der Hinweis, dass sich der Trickler nur mit dem Microsoft .NET Framework 4 betreiben lässt. Wer das noch nicht auf seinem Rechner hat, muss es installieren.

Der Trickler ist mit MS Visual Basic 2010 programmiert, und läuft nur mit Windows als Betriebssystem.

Hier der Link zum Framework 4:

http://www.microsoft.com/downloads/de-de/details.aspx?FamilyID=0a391abd-25c1-4fc0-919f-b21f31ab88b7

ebenso muss die *VisualBasicPowerPacksSetup.exe* vor dem ersten Start des Trickler installiert werden.

Ein Hinweis in eigener Sache: Mein Name ist Mirko H.ausL., im Tropfen-Forum auch als ‚Mirkole' bekannt.
Ich bin kein Profi-Programmierer. Ich hab mir das alles selber beigebracht. Bitte seht mir Fehler, die trotzdem nicht vorkommen sollten, nach. Falls wider Erwarten, doch Probleme auftauchen, schreibt mir eine Mail. Ich versuche sie so schnell wie möglich zu beheben. Auch wenn die Anleitung hier unverständlich oder unvollständig sein sollte, wäre ich für einen Hinweis dankbar. Ganz am Ende findet ihr noch nützliche Hinweise und Ergänzungen zum Text.

Ich habe eine Support-Mailadresse eingerichtet: trickler2012@googlemail.com

Dorthin könnt ihr all eure Anliegen senden. Außerdem würde ich gerne immer alle Nutzer benachrichtigen, falls es etwas Neues gibt. Falls ihr also in den Verteiler wollt, schreibt mir kurz.

Bitte lest diese Anleitung sorgfältig durch, ich möchte später nicht ständig Mails zu Fragen beantworten, deren Antworten hier in der Anleitung zu finden sind.
Danke.

Inhalt dieser Anleitung

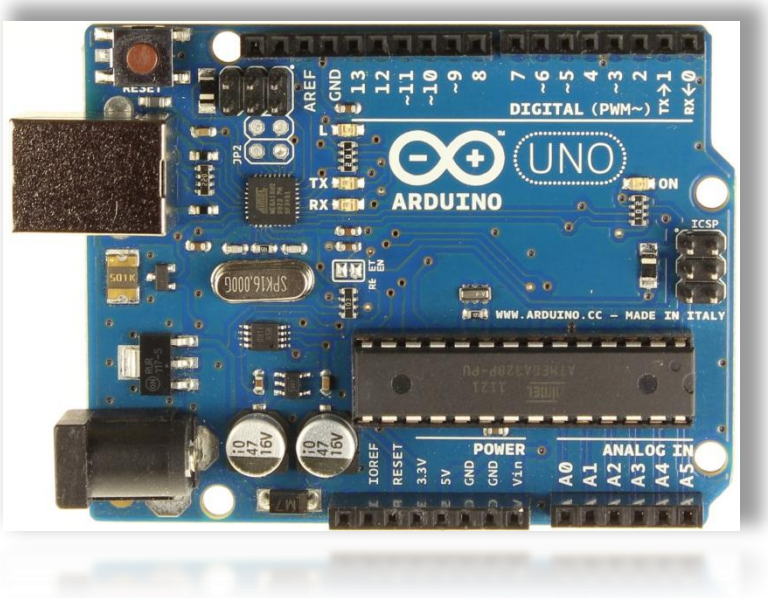
Arduino Hardware

Bedienung Trickler

Arduino Hardware

Der Trickler funktioniert nur in Verbindung mit einem Arduino. Der Arduino ist die elektronische Verbindung zwischen Rechner und Eurem Equipment, wie Blitze, Ventile und Kamera.

Auf dem Arduino gibt es viele Anschluss-möglichkeiten. Für den Trickler werden nur die Pins, in der hier oben zu sehenden schwarzen Anschlussleiste, verwendet.



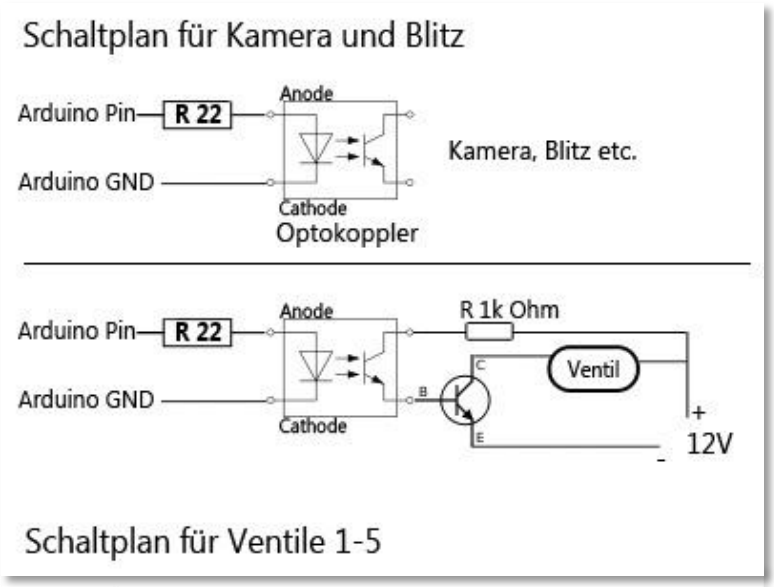
An die untere Leiste wird überhaupt nichts angeschlossen.

Links sieht man noch den USB-Anschluss und den Eingang für die Versorgungsspannung. Ein USB-Kabel ist zwingend erforderlich und muss auch immer zum Rechner verbunden sein, da hierüber später Daten vom Trickler zum Arduino gesendet werden.

Ein Netzteil wird nicht benötigt, da der Arduino normal genügend Strom über den USB-Port bekommt um LEDs, Optokoppler und die Transistoren zu schalten.

Verdrahtung der zusätzlichen Elektronik

Es ist nicht sinnvoll und auch nicht möglich das Fotoequipment direkt an den Arduino anzuschließen.
Der Arduino stellt an seinen Ausgängen zwar 5 Volt zur Verfügung, das reicht aber nicht um Eure Ventile zu öffnen.
Dafür wird ein externes Netzteil benötigt.
Blitze und Kameras könnten zwar theoretisch direkt angeschlossen werden, es ist aber sowohl für den Arduino als auch für Eure teure Kameraausrüstung sicherer, alles „galvanisch“ voneinander zu trennen.
Dafür sind die Optokoppler zuständig.



Diese elektronischen Bauteile werden benötigt:

10 Widerstände mit 22 Ohm
5 Widerstände 1 Kiloohm

10 Optokoppler, z.B. SFH610A

5 Transistoren, z.B. BD 139

1 Lochrasterplatine um alles drauf zu löten

Kabel, Lötzinn usw.....

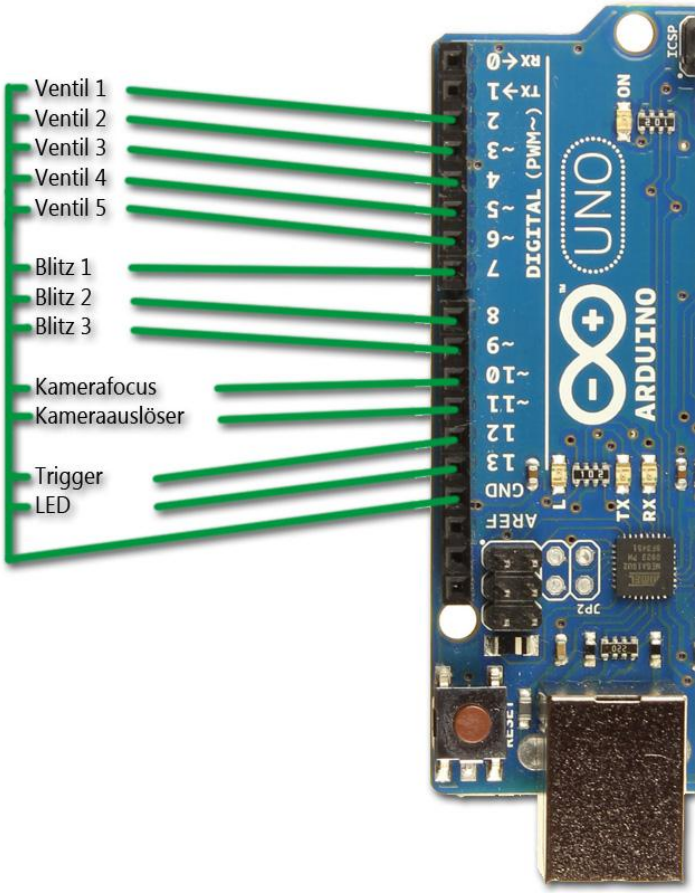
Wer nicht löten will oder kann, kann sich auch eine Steckplatine besorgen und alles nur drauf stecken.

Arduino Pinbelegung

Im Arduino Quellcode, kann man die einzelnen Pins frei vergeben. Ich habe sie dort folgendermaßen vergeben:

Die Pins 0 und 1 müssen leer bleiben, da sie für die USB-Datenübertragung genutzt werden.

Ventil 1	Pin 2	
Ventil 2	Pin 3	
Ventil 3	Pin 4	
Ventil 4	Pin 5	
Ventil 5	Pin 6	
Blitz 1	Pin 7	
Blitz 2	Pin 8	(optional)
Blitz 3	Pin 9	(optional)
Kamerafocus	Pin 10	(optional)
Kamera-Auslöser	Pin 11	
Trigger	Pin 12	
LED	Pin 13	



Der Trigger ist ein einfacher Taster, der direkt an Pin 12 und GND angeschlossen wird. Als Taster könnt ihr verwenden was ihr wollt. Ich habe z.B. einen Taster am Arduinogehäuse angebracht und zusätzlich nochmal einen, parallel dazu an einem langen Kabel in einem kleinen Extra-Gehäuse.

Die LED ist auch kein muss, wer die nicht nutzen will lässt sie einfach weg. Ansonsten genauso wie den Taster anschließen, ohne Vorwiderstand direkt an Pin 13 und GND.

Focus und Auslöser

Es gibt Unterschiede, wie Kameras verschiedener Hersteller ausgelöst werden.
Bei Nikon und Sony DSLRs muss immer zuerst der Focus „angeschaltet“ werden, dann der Auslöser dazu geschaltet werden.
Bei Canon DSLRs reicht der Auslöser alleine, dann wir der Focus-Pin einfach frei gelassen.

Findet heraus wie Eure Kamera per Fernauslöser auslöst, und verdrahtet entsprechend.

Blitze:

Blitze sollten über Optokoppler angesteuert werden. Bitte keine uralten Blitze verwenden, die die volle Blitzspannung (ca. 300 V) über den Optokoppler jagen. Das mag der nicht!!

Warum 3 Blitzausgänge?

Wer die Bilder mit Doppelbelichtung oder Stroboblitz kennt, kann sich vorstellen warum. Es können dadurch sehr schöne Effekte entstehen. Ich möchte Euch mit dem Trickler die Möglichkeit geben, verschiedene Ausleuchtungsszenarien zu erschaffen. Ihr könnt damit dann zu verschiedenen Zeiten, mit verschiedenen Blitzen blitzen. Wie das im Trickler funktioniert, erkläre ich später.

Was mache ich mit nur einem oder zwei Blitzen?

Wer nur einen externen Blitz besitzt, für den machen drei Ausgänge natürlich keinen Sinn. In dem Fall werden Blitz 2 und Blitz 3 einfach frei gelassen und nicht verdrahtet.

Es gibt allerdings die Möglichkeit, Blitz 1 zwei oder drei Mal, innerhalb eines Tropfvorgangs blitzen zu lassen.
Dieser Modus, der ausschließlich im Arduino stattfindet, muss allerdings im Quellcode hinterlegt sein.
Doch dazu später mehr.

Besonderheiten Blitze:

Ich habe viele verschiedene Blitze, von versch. Herstellern.
Theoretisch könnten die alle über einen Optokoppler gezündet werden. Funktioniert aber in der Praxis nicht immer, z.B. verträgt sich mein Nikon SB-800 nicht mit den 2 Blitzen von YongNuo.
Deshalb verwende ich jetzt zwei Optokoppler parallel an einem Arduino Pin, so sind die Zündkreise getrennt.
Zündet ein Blitz gar nicht, Polarität vom Blitzkabel drehen!! Also + auf -….

Blitzverteiler:

Ich habe mir einen sog. Blitzverteiler gebaut. Die ich mir an das Setup geschraubt habe. So brauch ich am Arduinogehäuse nur eine Buchse und mach daraus vier. Zu beachten ist dabei, dass wiederum nur miteinander kompatible Blitze an einen Verteiler angeschlossen werden.
Wo es solche Verteiler gibt, weiß ich nicht, beim Conrad vermutlich. Ich hab meinen aus einem zerlegten Kassettendeck. Alte Videorecorder oder Verstärker haben solche Teile auch oft hinten drin.

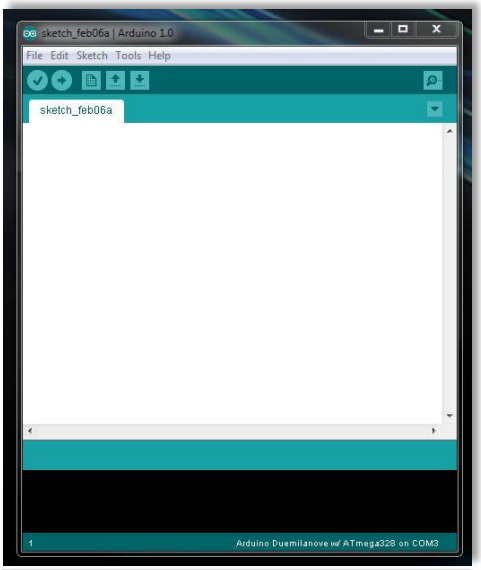
Arduino-IDE

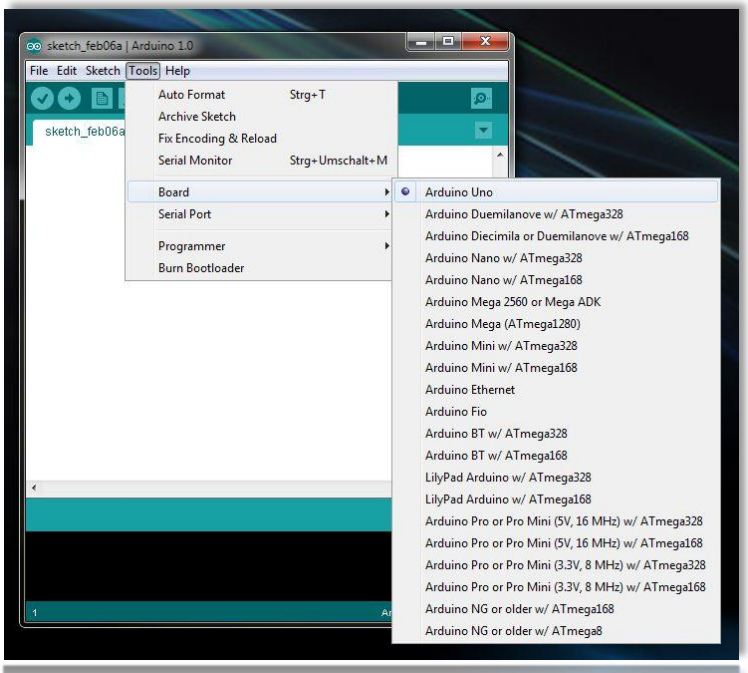
Bevor man den Quellcode auf den Arduino übertragen kann, muss man erst die Arduino-IDE 1.0 oder neuer herunterladen und installieren.

Hier der Link:
http://arduino.googlecode.com/files/arduino-1.0-windows.zip

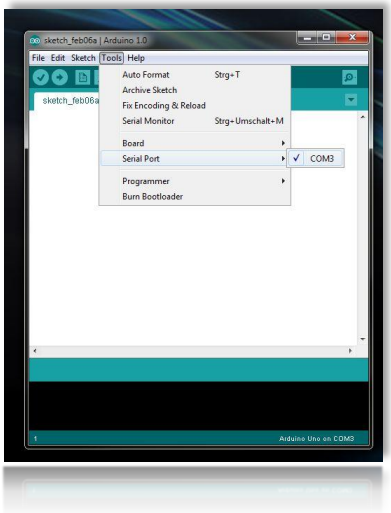
Dadurch werden, dann auch alle benötigten Treiber installiert.

Startet die arduino.exe und Euch erwartet dieser Screen.





Jetzt wird es Zeit endlich den Arduino mit dem USB-Port zu verbinden, wenn nicht eh schon lange geschehen.
Als nächstes sollte man im Menu unter Tools/ Board seinen Arduinotyp auswählen.



Dann fehlt noch die Auswahl des Serial-Ports, zu finden in Tools / Serial Port. Dort ist normalerweise nur noch ein Eintrag gelistet, dies ist Euer Arduino. Falls dort mehrere Einträge sind, müsst ihr im Windows Gerätemanager mal nachschauen. Ansonsten einfach mal durchprobieren.

Jetzt muss der Quellcode geladen werden. Diese liegt im Trickler-Ordner den ihr entpackt habt. *„TricklerGUI_UNO.ino“*
Diese Datei einfach per Drag und Drop in die IDE ziehen, ist der einfachste Weg.

Arduino-Quellcode aufspielen

Wenn die .ino erfolgreich in die IDE geladen ist, könnt ihr ein wenig runterscrollen. Bis ihr diesen Bereich gefunden habt.

Jetzt kommt der Teil für Leute, die schon einen fertig aufgebautes Arduinogehäuse haben und die Pins nicht nach meinen Vorgaben umstecken möchten.
Zwischen den Sternchenreihen werden die Nummern für die Pins vergeben. Außer den Ventilen, haben alle Pins eine eigene Zeile.

Normale Pins:

Ihr müsst jetzt schauen welche Pins zu Euren Gehäuse-Ausgängen gehen. Und die von mir vergebenen Nummern abändern. Bitte nicht den Strichpunkt hinter jeder Zeile löschen, der ist wichtig. Auch die Bezeichnungen, wie Blitz1 oder Trigger, dürfen nicht geändert werden. Sonst funktioniert später nix mehr!!!
Habt ihr Pins auf den Analogen Eingängen (Analog In) gesteckt, lautet die Bezeichnung entsprechend A2, oder A4…
Digitale Pins benötigen nur die entsprechende Ziffer ohne Buchstaben.

Blitzpins bei nur einem oder zwei Blitzen:

Wer mit nur einem Blitz bis zu drei Mal pro Tropfvorgang blitzen möchte, legt die Ziffer für alle Blitzpins auf z.B. Pin 7 fest und lässt Pin 8 und 9 einfach frei.

Also anstatt
const int Blitz1 = 7; // Blitzanschluss1
const int Blitz2 = 8; // Blitzanschluss2
const int Blitz3 = 9; // Blitzanschluss3

eben
const int Blitz1 = 7; // Blitzanschluss1
const int Blitz2 = 7; // Blitzanschluss2
const int Blitz3 = 7; // Blitzanschluss3

Das stellt kein Problem dar, die Variablen sprechen eben immer denselben Pin an. Aber halt zu unterschiedlichen Zeiten. Das ist die einfachste Möglichkeit dies umzusetzen. Man wird dies auch wohl kaum bei jedem Shooting

anders haben wollen. Kauft ihr Euch irgendwann noch ein paar Blitzgeräte, muss der Quellcode nochmals geändert und neu aufgespielt werden.

Die Ventilpins liegen in einem sogenannten Array, das für die Weiterverarbeitung im Quellcode nötig ist. Deshalb stehen die Ziffern der Pins in einer geschweiften {} – Klammer, getrennt von Kommas.

Dabei ist zu beachten, dass der erste Eintrag (A1) auf einen nicht verwendeten Pin führen sollte.
Habt ihr an A1 z.B. schon eine Lichtschranke angeschlossen (die im Moment eh nicht unterstützt wird) schreibt diesen Eintrag auf einen freien Pin (A3 oder A5) um. Dieser Pin wird später immer dann geschaltet, wenn ihr als Ventilnummer im Trickler, eine Tropfenzeile über den Haken abgeschaltet habt.

Code kompilieren:

Habt ihr das erfolgreich gemeistert, klickt oben links auf das Symbol (Verify) mit dem Haken. Der Code wird darauf hin kompiliert, also in Maschinensprache umgeschrieben. Davon seht ihr aber nichts, außer dem Fortschrittsbalken. Steht in der unteren türkisen Leiste dann „Done compiling“ ist alles OK.
Seht ihr einen roten Fehlercode, habt ihr irgendwo im Code einen Fehler. Schaut ob jede Zeile am Ende den Strichpunkt hat. Schaut ob der richtige Arduino unter ‚Boards‘ ausgewählt wurde und der Com-Port passt.

Code aufspielen:

Achtung: Der Trickler darf nicht laufen, während ihr den Quellcode auf den Arduino übertragt.

Ist alles OK, muss der Code auf den Arduino übertragen werden. Dazu ist neben dem Verify-Button der Upload-Button (Pfeil nach rechts). Einmal drauf klicken.
Der Code wird nochmals kompiliert und danach auf den Arduino gespielt.
Bekommt ihr hier einen roten Fehlercode, stimmt der ausgewählte ‚Serial Port‘ u.U. nicht.

Hinweis:

Bei Problemen oder einfach als Test, kann man auch erst ein fertiges kleines Beispielprogramm auf den Arduino hochladen. Eines, das keine zusätzliche Elektronik benötigt, gibt es unter
File/Examples/Basics. Das Programm „Blink“.
Dieses Programm lässt die aufgelötete SMD-LED auf dem Arduino im Sekundentakt blinken.

Arduino-Reset:

Auf dem Arduino ist ein kleiner Microschalter aufgelötet, dies ist der Reset-Knopf.
Sollte die Datenübertragung immer wieder fehlschlagen, kann man hier den Arduino neu starten.
Es wird dabei nichts gelöscht, also keine Angst davor, mal drauf zu drücken.
Es passiert lediglich das Gleiche, wie wenn das USB-Kabel raus und wieder reingesteckt wird.

Bedienung des Trickler 1.0

Kommen wir also nun zur eigentlichen Bedienung des GUI.

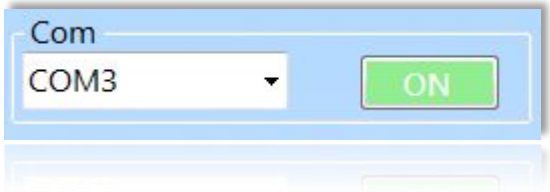
Achtung: Der Arduino sollte immer *VOR* dem Start des Trickler schon an den Rechner per USB-Kabel angeschlossen sein und die Status-LED, sofern ihr sie angeschlossen habt, sollte dauerhaft leuchten.
Wird dies vergessen, kommt u.U. eine Fehlermeldung von Windows, falls nicht, startet der Trickler ganz normal.
Es wird dann ein roter Bereich im Com-Fenster angezeigt, den ihr Doppelklicken müsst um den Com-Port nochmal neu zu initialisieren.

Hier nochmals der Hinweis, dass sich der Trickler nur mit dem Microsoft .NET Framework 4 und dem *VisualBasicPowerPacksSetup* betreiben lässt. Wer das noch nicht auf seinem Rechner hat, muss es installieren. (siehe Seite 1)

In der entpackten .rar-Datei liegt die TricklerGUI.exe, diese muss einfach nur gestartet werden.
Es gibt keine Installationsroutine oder ähnliches. Legt den Trickler-Ordner irgendwo auf der Festplatte ab oder auf dem Desktop.
Nach dem erfolgreichen Start, seht ihr das große, oben dargestellte Fenster.
Während des ersten Starts, werden die Ordner „save“ und „lastsetup“ angelegt. Dort werden später die gespeicherten Setups abgelegt.

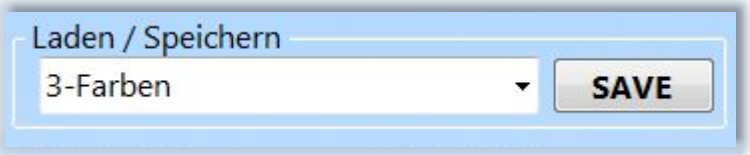
Com:

Wie auch die Arduino-IDE, sucht der Trickler nach verwendeten Com-Ports und listet sie im entsprechenden Feld auf.
Falls ihr mehrere Einträge habt, sucht euch den richtigen aus.



Daneben befindet sich der Com-Button.
Ein Klick auf den Com-Button stellt die Verbindung manuell her oder unterbricht sie.
Je nachdem, leuchtet der Button dann Rot oder Grün.

Laden / Speichern:



Die Zeiten und Bemerkungen können selbstverständlich dauerhaft gespeichert werden und danach wieder geladen werden. Im Trickler-Ordner befindet sich dazu der Save-Ordner. Beim Start wird geprüft, ob er vorhanden ist, und dann die darin enthaltenen Setups in die Liste eingetragen. Sollte der Save-Ordner nicht vorhanden sein wird er angelegt.
Um ein Setup zu laden, wird es nur in der Liste ausgewählt, sofern es welche gibt.
Zum Speichern eines neuen Setups, gibt man in das Feld einfach einen neuen Namen ein und klickt auf SAVE.
Um ein Setup zu überschreiben, lässt man den Namen wie er ist und klickt auf SAVE.
Beim Schließen des Trickler über das rote X oben rechts, merkt sich der Trickler bis zum nächsten Start, welches Setup gerade geladen war, und lädt dieses automatisch. So könnt ihr sofort dort weitermachen wo ihr ‚gestern Abend' aufgehört habt.
Allerdings werden die letzten Werte des Setups, beim Schließen, nicht automatisch überschrieben. Will man also die gerade eingestellten Werte für ‚morgen' sichern, muss man vorher noch kurz auf SAVE klicken.
Die Setups im save-Ordner sind einfache, einzelne Text-Dateien und können mit dem Editor angeschaut und auch geändert werden. (nicht sinnvoll)
Habt ihr irgendwann zu viele Setups, könnt ihr die Textdateien, die nicht mehr benötigt werden, auch löschen.
Der Trickler durchsucht die Dateiliste bei jedem Start neu und listet nur vorhandene Setups.

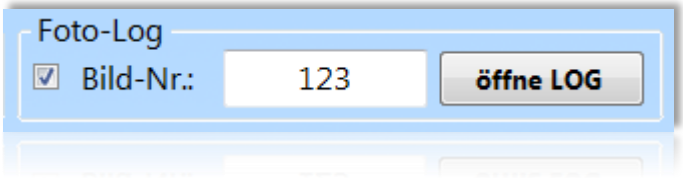
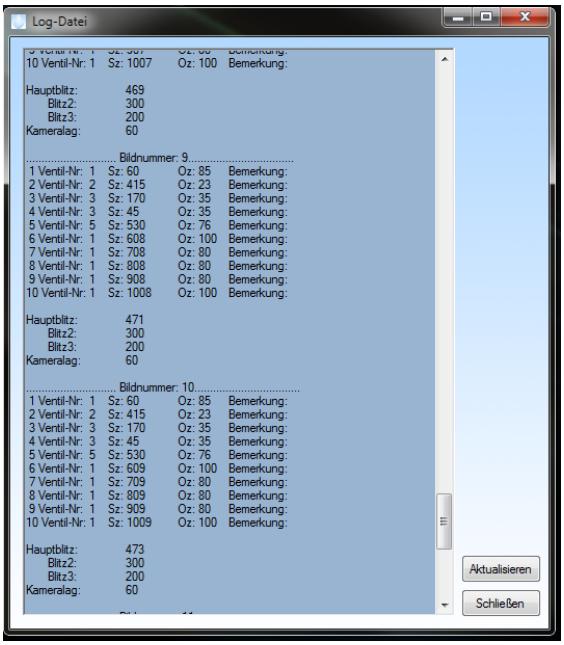


Foto-Log:

Der Trickler bietet die Möglichkeit zu jedem Foto das ihr macht, die eingestellten Werte in einer Text-Datei abzulegen. (Bei der ersten Benutzung des Logs, wird die „Log-Datei.txt" im Hauptordner erstellt.)



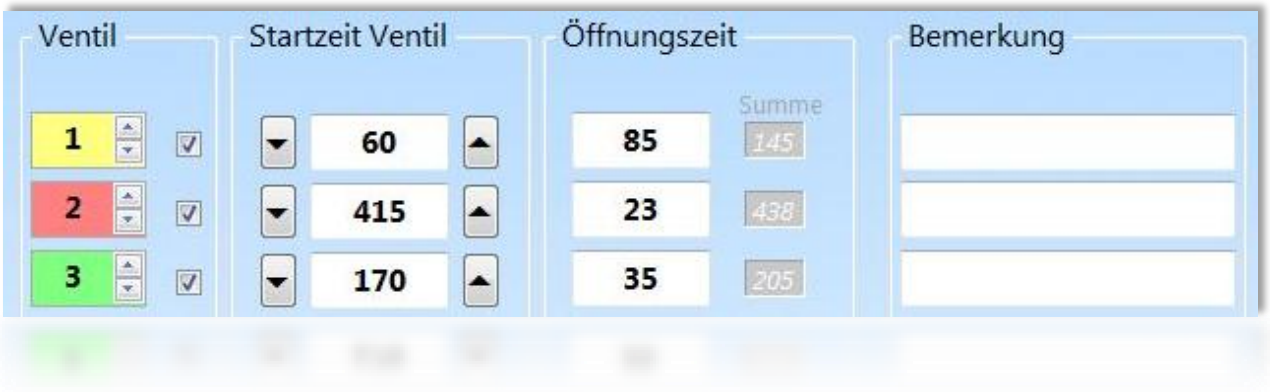
Dabei werden nur die benutzten Werte festgehalten.
Jede Kamera speichert ihre Bilder unter einer Nummer, also z.B. DCM_123.jpeg. Wollt ihr jetzt den Log mitschreiben lassen, muss in das Feld die Bildnummer vom nächsten Foto eingetragen werden. Danach kann der Haken gesetzt werden.
Ab dann wird bei jedem Klick auf Auslösen oder bei Betätigung des Trigger-Tasters am Arduino ein neuer Datensatz in die Log-Datei geschrieben. Die Bildnummer wird danach automatisch um 1 erhöht.
Der ‚öffne LOG'-Button öffnet ein neues Fenster und gibt den Log dort in einem Textfenster aus.
Leider aktualisiert sich das Textfenster nicht automatisch, sobald ein neuer Datensatz geschrieben wurde und das Log-Fenster schon offen ist. Dafür ist der Aktualisieren-Button gedacht. Das Log-Fenster wird über den Schließen-Button oder über X geschlossen.

Um den Log nicht Kilometerlang werden zu lassen, empfiehlt es sich den Log nach einer beendeten Fotosession, in den Ordner, in dem auch die gemachten Bilder liegen, zu kopieren.
Danach kann die „Log-Datei.txt" mit dem Editor leer überschrieben werden.

Ich habe mich bewusst gegen einen Log-Leeren Button entschieden, um ein versehentliches löschen der Daten zu vermeiden.

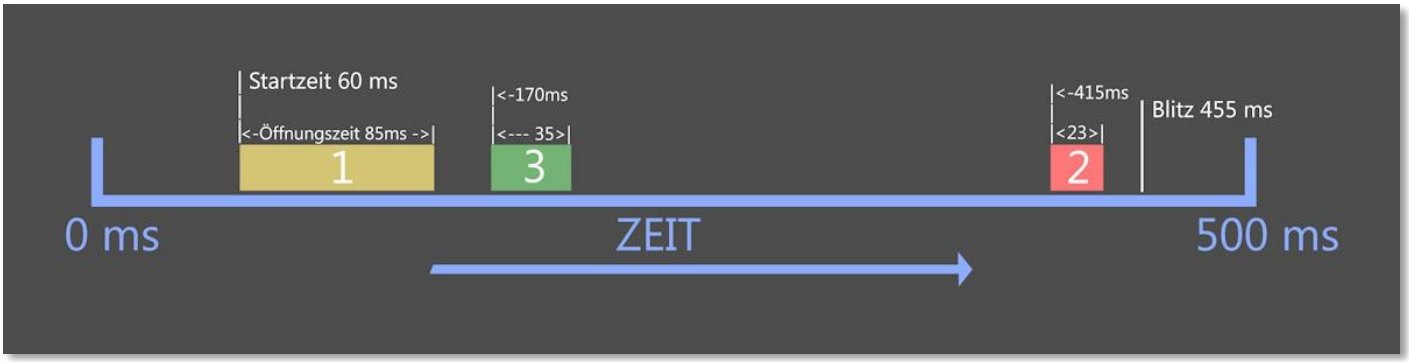
Bitte beachten: Wird aus irgendeinem Grund, nach dem Klick auf ‚Auslösen', kein Foto gemacht (Kamera im Standby, Speicherkartenfehler etc.), zählt der Foto-Log trotzdem weiter. Dadurch stimmen dann die Bildnummern zwischen LOG und Fotos nicht mehr überein. Weder Arduino noch das GUI können überprüfen, ob wirklich ein Bild gespeichert wurde. Es wird davon ausgegangen, dass es abgespeichert wurde.

Steuerung der Ventile:



Die Ventile, bzw. die einzelnen auszulösenden Tropfen werden auf folgende Weise gesteuert. Jede Zeile im oben dargestellten Bild, steht für einen Tropfen der ausgelöst werden soll. Es sind also bis zu 10 einzelne unabhängige Tropfen pro Vorgang möglich.
Die einzelnen Zeilen sind nicht nummeriert, da sie im Arduino nicht nacheinander abgearbeitet werden, sondern alle Gleichzeitig. Deshalb wäre es auch Beispielsweise möglich mit der 7. Zeile den ersten Tropfen auslösen zu lassen. Natürlich macht es Sinn, der Übersichtlichkeit wegen, die Tropfenabfolge zeitlich von oben nach unten einzustellen.

Um zu verstehen, wie der zeitliche Ablauf, der drei ausgelösten Tropfen, auf der Zeitleiste (Timeline) geschieht, hab ich dieses Bild hier erstellt.



Bezogen auf die ganz oben eingestellten Zeiten, sieht der Ablauf folgendermaßen aus:

Die Timeline beginnt logischerweise bei 0ms. Danach passiert erst mal nichts. Wenn 60ms erreicht werden öffnet Ventil1 und bleibt für die Zeit von 85ms offen. Also nicht bis 85ms auf der Timeline erreicht werden, sondern ab der Startzeit 60ms gerechnet. Auf die gesamte Timeline gesehen, macht Ventil 1 also bei 60ms + 85ms = 145ms wieder zu. Ebenso öffnet Ventil 3, das im Zeitablauf als nächstes kommt, bei 170 ms und bleibt für die Zeit von 35ms offen. Ventil 2 öffnet bei 415ms und geht bei 415 + 23 = 438ms zu. Am Ende folgt noch der Blitz, den ich auf 455ms eingestellt habe. Blitzzeiten haben keine einstellbare Öffnungszeit, diese ist fest auf 1ms eingestellt.

Der Vorteil die Öffnungszeit und nicht die Endzeit einzustellen, liegt in der dadurch fixen Tropfengröße.
Denn die Öffnungszeit regelt ja, wie viel Wasser aus dem Ventil läuft.
Bin ich der Meinung, weil mir die entstandene Tropfenform nicht gefällt, der grüne Tropfen von Ventil 3 müsste etwas früher gestartet werden, verstellt man nur die Startzeit auf z.B. 165ms. Die Tropfengröße bleibt dabei dann gleich.

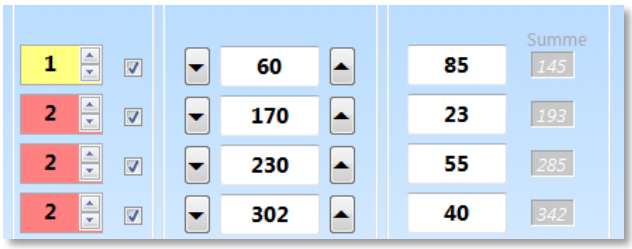
Vorhin habe ich gesagt, die Abarbeitung der Tropfen läuft gleichzeitig ab und nicht nacheinander. Was ich damit sagen will, seht ihr in diesem Bild.

Hier habe ich den grünen Tropfen auf die Startzeit 130ms verschoben. Wie ihr seht würde das ja mit der Öffnungszeit des gelben Tropfens kollidieren. Da dies aber ein anderes Ventil ist, ist diese Einstellung problemlos möglich.Man könnte den grünen Tropfen auch auf dieselbe Startzeit legen wie den Gelben, und auch noch den roten Tropfen bei 60ms starten lassen.

Alles egal, solange es verschiedene Ventile sind die die Tropfen auslösen. Denn, was zwar funktioniert aber nichts bringt, wäre das rote Ventil in der Zeit in der es eh schon offen ist, nochmal in einer anderen Zeile zu öffnen.

Dadurch verlängert sich nur die gesamte Öffnungszeit dieses Ventils. Was natürlich dagegen problemlos möglich ist, ist ein Ventil 2, 3, oder viermal auszulösen.
Dafür braucht dann jede Auslösung eine eigene Zeile mit eigenen Zeiten, als Ventil wird dann eben immer dieselbe Nummer gewählt.

Dies würde dann ungefähr so aussehen.



Bedienung und Verstellung der Werte:

Die Ventilnummer wird über die zwei Pfeile verstellt, die in jedem Kästchen vorhanden sind. Es gibt nur die Werte 1-5, habt ihr die 5 erreicht fängt es wieder bei 1 an. Dazu verändert sich neben der Nummer auch die Farbe des Ventils.
Daneben ist der Haken um die Zeile zu aktivieren oder zu deaktivieren. Die Werte bleiben dabei erhalten, nur der Tropfen wird dann eben nicht ausgelöst.
Die Startzeitfelder können mit der Tastatur direkt geändert werden, oder bei vorherigem Klick ins Feld per Mausrad. Das ist sehr praktisch und auch schnell. Bei einem Laptop-Touchpad funktioniert es auch, wenn man einen Bildlaufleistenbereich hat.
Zusätzlich gibt es noch die Pfeilbuttons daneben, die erhöhen oder verringern den jeweiligen Wert um + oder – 10.
Die Öffnungszeit wird genauso verstellt, nur dass es dort keine 10er-Buttons gibt. Die Werte der Öffnungszeit liegen erfahrungsgemäß meist eh nur im Bereich von ca. 20 – 150 ms.

Summe:

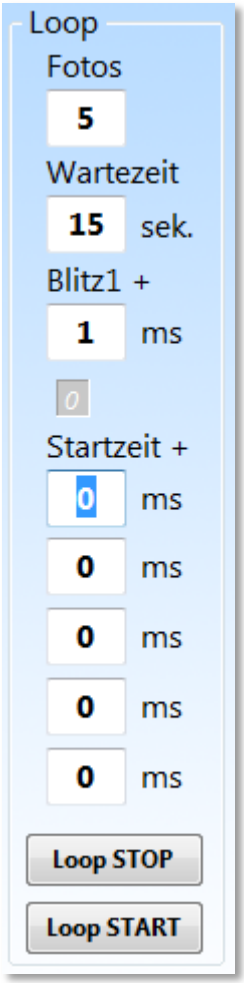
Im kleinen Summenfenster jeder Zeile, wird die Summe oder die Endzeit eines Tropfens dargestellt.
Da der Trickler die Werte nicht miteinander vergleicht, ist das eine kleine Hilfe um die oben beschriebenen Ventilzeitüberschneidungen bei gleicher Ventilnummer etwas besser im Blick zu haben.
Zu beachten wäre dabei lediglich, dass aus meiner Erfahrung, Ventile ca. 20ms fürs Öffnen oder Schließen benötigen.

Doppelklick auf eine Öffnungszeit:

Es gibt die Möglichkeit einen einzelnen Tropfen ohne Kamera und Blitz auszulösen, z.B. um die Position eines Ventils auszurichten. Hierzu macht man einen Doppelklick auf eine Öffnungszeit. Dadurch wird das entsprechende Ventil dieser Zeile, für die eingestellte Zeit, geöffnet und ein Tropfen ausgelöst.

Bemerkung:

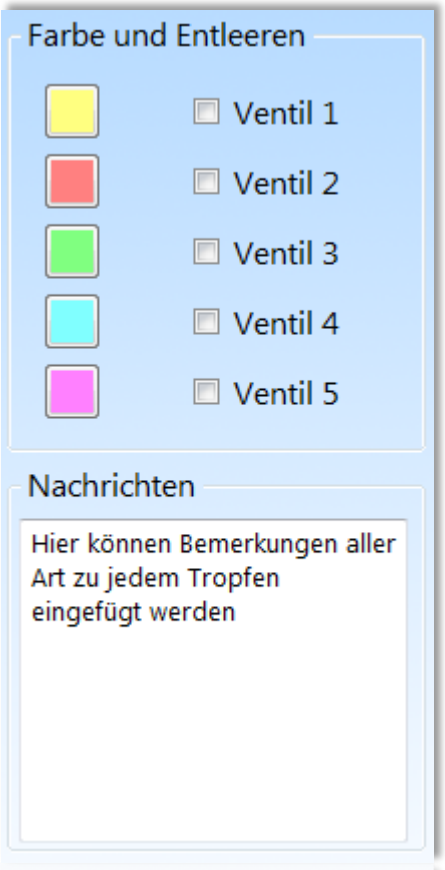
Die einzelnen Zeilen für Bemerkungen, könnt ihr nutzen um zu den Ventilen oder zu einzelnen Tropfen etwas zu schreiben. Ich hatte da schon z.B. die Position (links, rechts, mitte usw.) der Ventile drin.
Die Bemerkungen werden auch mit abgespeichert.



Der Loop funktioniert auf folgende Weise.

Bei *Fotos* wird eingegeben wie viele Bilder ihr aufnehmen wollt.
Die *Wartezeit* ist die Pause zwischen den Bildern. Die kürzeste mögliche Wartezeit sind 5 Sekunden, nach oben gibt es keine Grenze. Auf die Wartezeit wird automatisch nochmals eine Sekunde addiert, da die internen Timer nicht darauf warten, bis der Tropfenvorgang wirklich beendet ist.
Blitz1+, ist dazu gedacht, die Blitzzeit schrittweise bei jedem Bild um den eingegeben Wert in ms zu erhöhen.
Steht im Feld Hauptblitz z.B. 455 ms, und bei Blitz1+ z.B. 2 ms, löst der Blitz beim ersten Bild bei 455 ms aus, beim 2. Bild bei 457, 3. 459….usw. Mit dieser Funktion kann man die komplette Entstehung einer Tropfenform schrittweise aufnehmen, solange sich die Wasserbedingungen nicht ändern. (Wasserstand im Siphon, Wasserpegel im Becken usw.)
Das kleine Info-Feld, zählt die Wartezeit-Sekunden bis zum nächsten Auslösen herunter.
Genauso verhalten sich die *Startzeit+* Felder. Sie beziehen sich allerdings immer nur auf die Startzeit derselben Zeile. Also jeweils nur auf die unteren 5 Zeilen. Wer damit arbeiten will, muss die Tropfenwerte, die erhöht werden sollen, in eine der unteren 5 Zeilen eintragen.
Diese sogenannten Inkremente, also die schrittweise Erhöhung von Werten, der Startzeit eines Tropfens, ist eher für Leute gedacht, die wirklich wissen was sie da tun. Normalerweise, sollte der Wert besser auf 0 bleiben, so wird auch nichts addiert.
Blitz1+ und Startzeit+, können auch Minuswerte enthalten und werden dann abgezogen.
Ein Klick auf *Loop START* und der Vorgang beginnt. Die vier Buttons, Auslösen, Loop START, Fotosetup und Blitz, sind während des Loopvorgangs gesperrt. Es ist aber möglich, Ventile dazu zuschalten, Start oder Blitzzeiten zu verändern während der Loop läuft. Die Werte werden dann jeweils immer für das nächste Loop-Bild verwendet. Zur Sicherheit sollten die Werte nur geändert werden, während die Datenstatusleuchte auf grün steht (Wartezeit zwischen den Bildern). *Loop STOP* bricht den Loop sofort ab.

Die eingestellten Tropfenwerte, werden weder während, noch nach dem Loop in den einzelnen Zeilen geändert. Sind Inkremente eingestellt, erhöhen sie sich nur intern. Mit dem Foto-Log können sie protokolliert und eingesehen werden.



Farbe und Entleeren:

Jedem einzelnen der 5 möglichen Ventile, kann eine *Farbe* zugeordnet werden. Einfach auf einen der Farbbuttons klicken und die gewünschte Farbe über den normalen Windows-Farbdialog einstellen. Ich habe hier meist die Farbe gewählt die auch das Wasser aus diesem Ventil hat.

Wollt ihr das komplette Wasser aus einem *Wassertank ablassen*, bzw. ein Ventil für unbestimmte Zeit öffnen, wird hier entsprechend ein oder mehrere Haken gesetzt. Die Ventile sind so lange offen, bis ihr den Haken wieder entfernt.

Nachrichten:

In dieses Feld kann nichts geschrieben werden, hier werden bestimmte Statusmeldungen vom Trickler oder vom Arduino ausgegeben.
Es sind auch ein paar Hinweise hinterlegt, die erscheinen, wenn ihr im Trickler auf verschiedene Felder klickt.
Blitz: Hier werden nur die Blitze gezündet. Alle Gleichzeitig, auch wenn unterschiedliche Zeiten eingestellt sind und verschiedene Ausgänge geschalten werden.
Fotosetup: Zum Einrichten von Blitzen oder der Kameraposition, kann hier ein Blitz-Foto ohne Tropfen gemacht werden. Hierbei werden die drei verschiedenen Blitzzeiten korrekt ausgeführt.

Der Fotolag:

„Lag“ ist Englisch und bedeutet so viel wie Verzögerung. Jede Kamera, abhängig vom Hersteller und Modell, braucht unterschiedlich viel Zeit, bis der Verschluss komplett geöffnet ist. Der Arduino sendet also ein Signal (siehe Arduino-Quelltext Kamera-Auslöser) und die Kamera macht ein Bild. *Bis* das Bild allerdings gemacht wird, vergeht eine gewisse Zeit.
Da aber nicht der Zeitpunkt des Kamera-Auslösers der entscheidende Punkt bei der Tropfenfotografie ist, sondern der Blitzzündzeitpunkt, muss die Kamera immer eine gewisse Zeit vor dem Blitz schon ausgelöst werden.
Dies ist der Kamera-Lag.
Kameras von Canon brauchen eine relativ lange Zeit (ca. 150 ms), Nikon Kameras sind schneller (ca. 60 ms), da sie durch das Focus-Signal, das es bei Canon ja nicht gibt, schon zum Fotografieren bereit sind.

Der Kamera-Lag bezieht sich Grundsätzlich auf den Hauptblitz.
Egal was beim Hauptblitz für ein Wert eingegeben ist, die Kamera wird immer um die Kamera-Lag-Zeit, vorher ausgelöst.
Die oben genannten Werte sind nur Anhaltspunkte, welcher Wert genau bei euch eingestellt werden muss, müsst ihr selber herausfinden. Zuverlässig funktioniert es bei mir bis zu einer Belichtungszeit von 1/80 Sek.
Habt ihr Eure Werte für Belichtungszeit und Kamera-Lag herausgefunden, bleiben diese dann immer gleich.

Hauptblitz und Blitz 2 – 3:

Die Hauptblitz-Zeit ist die wichtigste Zeit überhaupt beim Tropfen fotografieren. Denn sie entscheidet über den wirklichen Zeitpunkt des Fotografierens. Meine Kamera ist auf den manuellen Modus eingestellt, bei einer Blende 20 und 1/80sek. bekomme ich ohne Blitzekomplett schwarze Bilder bei Tageslicht. Das ist so gewollt, denn die Abbrenndauer eines Blitzes ist um ein vielfaches kürzer und genauer als der Kameraverschluss. Nur über die Abbrenndauer kann man die Tropfen „einfrieren“, sie liegt, je nach Stärke, grob bei ca. 1/20000 Sek.

Der Quellcode im Arduino ist so programmiert, dass der Hauptblitz grundsätzlich die letzte Aktion eines Tropfenvorgangs ist. Stellt Euch folgendes vor, sobald ihr im Trickler auf Auslösen klickt, beginnt im Arduino eine Schleife zu rotieren. Sehr schnell. In dieser Schleife stehen die ganzen Befehle für die Ventilzeiten.
Ist eine Zeit erreicht wird der entsprechende Befehl ausgeführt und die Schleife rotiert weiter. Die Schleife würde endlos weiter rotieren, würde man sie nicht irgendwann abbrechen. Dieser Zeitpunkt ist gekommen, wenn der Hauptblitz gezündet hat. Danach rotiert die Schleife noch 10 ms weiter. Danach kehrt der Arduino in den „Empfangsmodus“ zurück.

Der Grund warum ich das schreibe, ist der, dass ihr wisst, warum alle u.U. eingestellten Ventilaktionen nach dem Hauptblitz nicht mehr ausgeführt werden. Vorsorglich findet aber nach dem Hauptblitz nochmals eine komplette Ventilabschaltung statt, so dass kein Ventil offen bleibt, falls irgendwo doch noch eine Startzeit innerhalb der Schleife liegt.

Jetzt habt ihr die Möglichkeit, zwei weitere Blitzzeiten zu nutzen. Diese Blitzzeiten sind, wie oben schon erwähnt, für Strobo Effekte oder Doppelbelichtungen gedacht. Braucht man selten, aber stört auch nicht wenn man es kann.
Die Blitzzeiten 2 und 3 müssen zwingend vor dem Hauptblitz passieren, um sie zu nutzen. Stehen sie wie im Beispiel oben auf 5000 ms, werden sie nicht berücksichtigt, da die Schleife ja schon lange vorher abgebrochen wurde. Ansonsten können beide Zeiten frei auf der Zeitleiste verschoben werden.

Wurden die Blitzausgänge im Quellcode von Euch alle auf Pin 7 gelegt, blitzt immer Eure komplette Anlage.
Nutzt ihr verschiedene Ausgänge, könnt ihr verschiedene Blitze zu verschiedenen Zeiten zünden.
Achtet darauf, dass sich ein Blitz nach seiner Zündung auch wieder erst aufladen muss.
Ggf. zündet ein noch nicht wieder bereiter Blitz, zur Hauptblitzzeit eben nicht.

Auslösen:

Dazu muss ich jetzt ja kaum noch was schreiben. Hier wird der komplette Datensatz auf den Arduino übertragen und ausgeführt. Für die Zeit des Tropfenablaufs, ist der Button gesperrt.

Datenstatus:

Dies ist eine „Statuslampe“ die Euch anzeigen soll, was gerade im Arduino oder im GUI passiert.

Grün: Daten wurden sicher und korrekt auf den Arduino übertragen.

Rot: Fehler bei der Datenübertragung. Am besten nochmal probieren.

Blau: Daten wurden im GUI geändert, aber noch nicht übertragen.

Gelb: Daten werden im Moment übertragen.

Normalerweise gibt es keine Probleme mit der Datenübertragung. Solltet ihr trotzdem öfters eine rote Lampe sehen, habt ihr u.U. ein zu langes USB-Kabel (max. 5m). Es könnte auch der USB-Port am Rechner etwas ausgeleiert sein. (siehe auch unter „Nützliche Hinweise“ – Daten falsch.

Nützliche Hinweise:

- Warum verwende ich bei den *Startzeitbuttons „Pfeile“ hoch und runter*? Einerseits natürlich um zu zeigen wo die Zeit erhöht oder verringert wird. Der eigentliche Grund ist aber der, dass ein fallender Tropfen im Bild nach oben wandert wenn ihr die Startzeit erhöht oder immer weiter unten abgelichtet wird, je niedriger die Startzeit ist. Probiert es aus.

- *Daten falsch*: Sollte im Nachrichtenfenster ständig die „Daten falsch“ stehen, kann man den Arduino reseten. Er besitzt dazu einen kleinen RESET-Knopf auf der Platine. Der Quellcode wird dadurch nicht gelöscht, sondern nur komplett neu gestartet.

- Es ist immer wieder ganz nützlich, die Startzeit des ersten Tropfens nicht auf 0 zu setzen.
  Ich starte ihn gerne erst bei 100. So hat man auch ‚Raum‘ nach hinten, falls er doch besser etwas früher starten sollte. Die Wartezeit am Anfang hat ja keine Nachteile.

- Alle Zeiten sind Millisekunden. Mikrosekunden sind im Moment nicht vorgesehen. Kommt vielleicht irgendwann.

- Start- und Öffnungszeiten erlauben keine Kommazahlen. Die könnt ihr zwar eingeben, sie werden aber nicht berücksichtigt, sondern hinterm Komma einfach abgeschnitten.

- Wer einen Arduino Mega nutzen möchte, ich habe auch dafür einen Quellcode. Dort wären zusätzlich dann auch noch ein LCD-Display für Statusmeldungen oder mehrere Taster für direktes Blitzen oder Fotosetup möglich. Der Mega hat insgesamt 70 I/O-Pins, der Uno nur 20.
  Siehe hier: http://arduino.cc/en/Main/Hardware

- Neue Erkenntnisse und Hinweise die hier noch stehen könnten, dürft ihr mir gerne zusenden.

trickler2012@googlemail.com