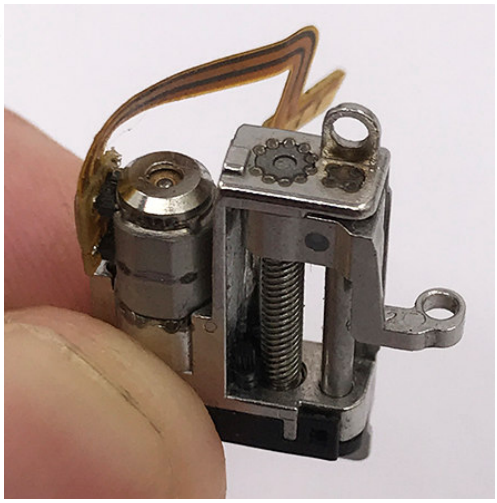
Micro-Schrittmotoren und die MobaLedLib

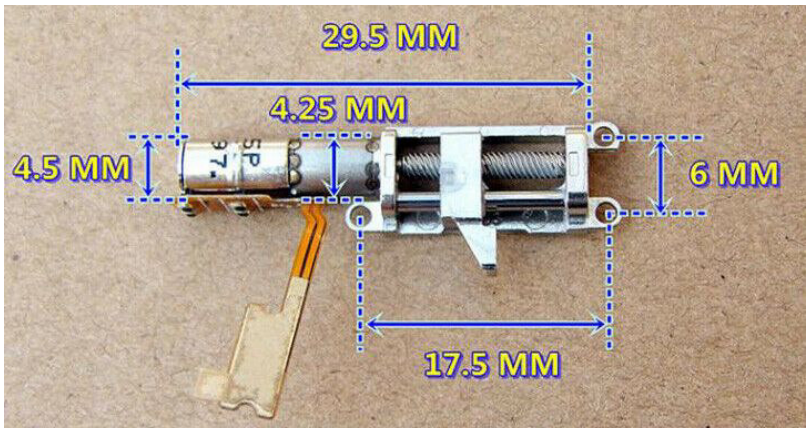
Inhaltsverzeichnis

1. Der Schrittmotor



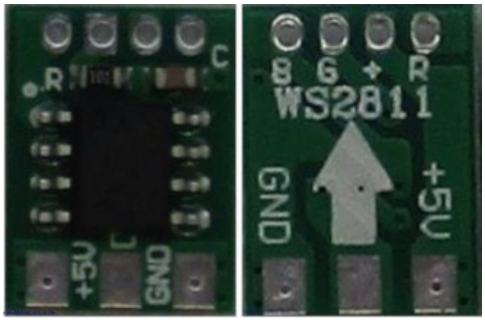
Man findet im Internet eine große Anzahl kleiner Schrittmotoren, wie die oben abgebildeten. Wir hatten einen Motor des linken Typs zur Verfügung. Der rechte hat den Motor neben dem Schlitten. Damit ist er kürzer aber auch breiter. Interessant ist bei diesem Modell, dass es diesen auch mit einer geschlossenen Öse am Schlitten gibt, was vielleicht die Montage eines beweglichen Teils vereinfacht. Dieses Modell ist aber ca. doppelt so teuer wie der längliche Typ.

Zu den Daten :



Die äußeren ca. Maße kann man der obigen Zeichnung entnehmen. Der Schlitten bewegt sich von Anschlag zu Anschlag um ca. 12 mm. Die Betriebsspannung soll 5 V sein. Stromaufnahme ca. 70mA.
Die Preise beginnen unter 1,00 €. Vorsicht: Es gibt auch ganz primitive Motoren für ca. 0,10 €.

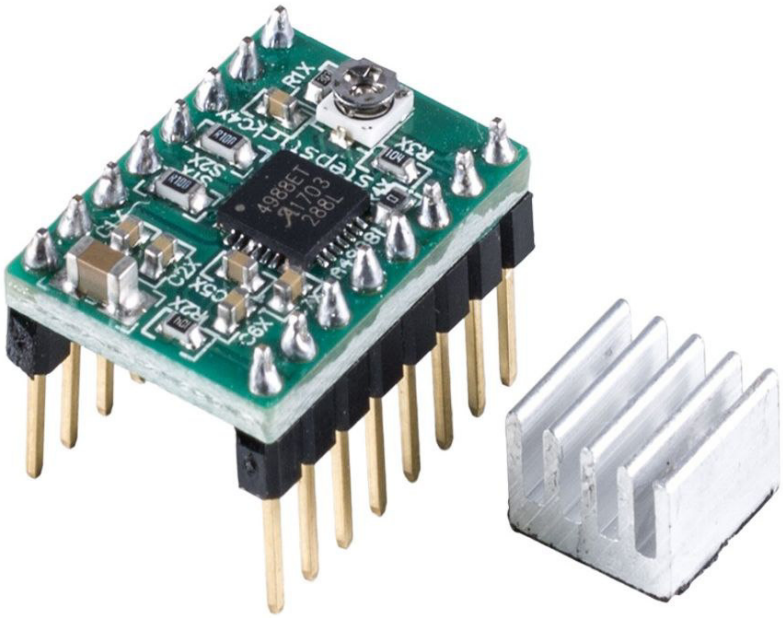
2. Der WS2811



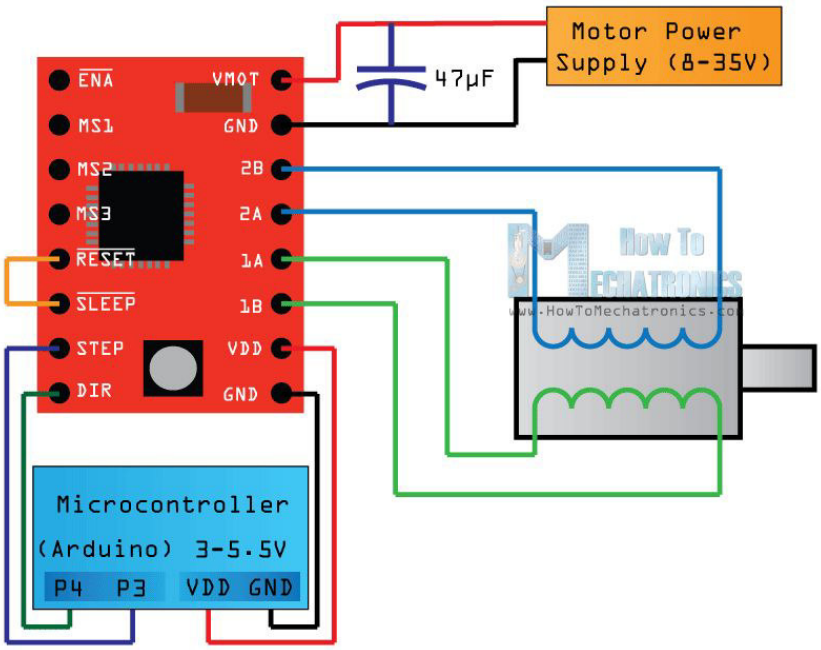
(Beispiel)

Dieser Baustein sollte hinreichend bekannt sein, so dass ich nur noch dazu sagen möchte, dass man die SMD-Version benutzen sollte, weil sie schneller getaktet ist (laut Hardi). Es gibt verschiedene Bauformen, die aber alle funktionieren sollten. Wer den WS2811 als SMD selber auf eine Platine löten möchte, sollte an die zusätzlich notwendigen passiven Bauelemente denken.

3. Der A4988 (als fertiges Modul)



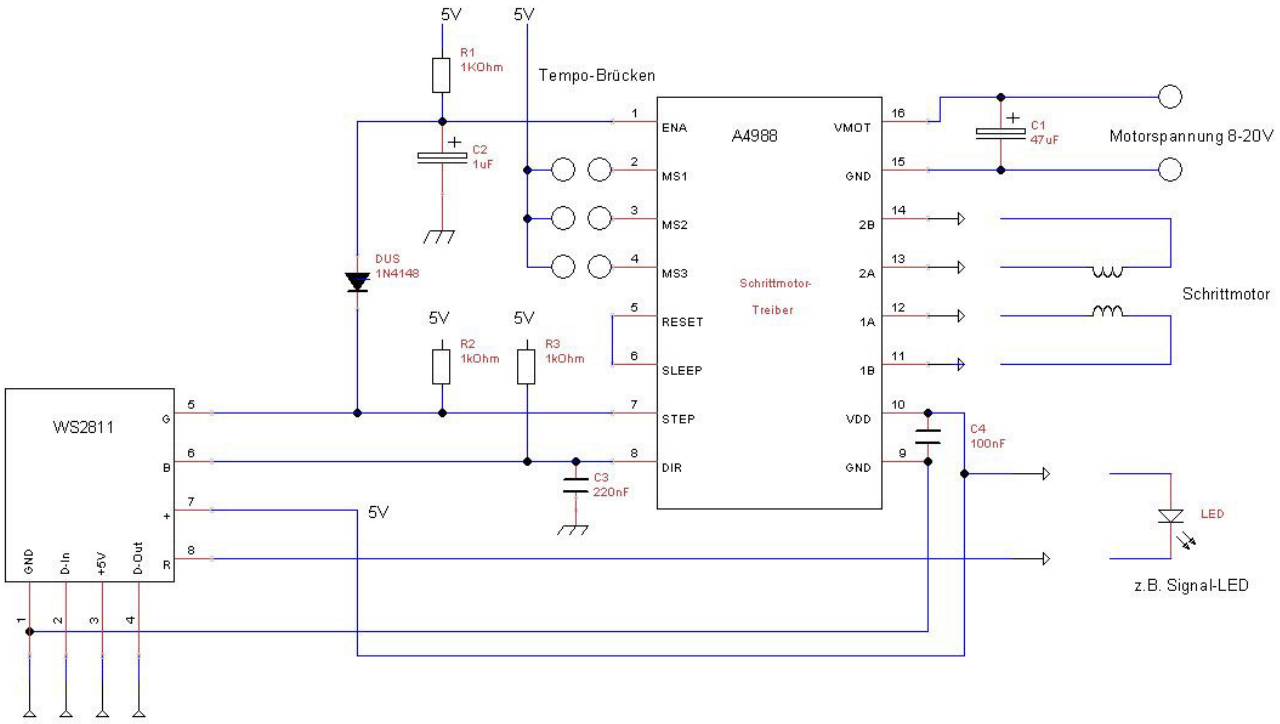
Die Funktionsweise dieses Moduls möchte ich etwas erklären. Deshalb hier einmal ein schematisches Anschlussdiagramm, wie man es im Internet finden kann:



Beginnen wir mit den einfachen Anschlüssen auf der **rechten** Seite:
Unten rechts kommt die 5V Spannung vom WS2811, oben rechts eine externe Spannungsquelle für den Schrittmotor (Die 5V Spannung vom Arduino reicht nicht, die Motorspannung muss mindestens ca. 7 bis 8V betragen).
Die vier Motorausgänge 1A/B und 2A/B werden mit je einem Spulenpaar des Motors verbunden (Im Zweifel mit einem Ohmmeter den Motor vermessen).
Auf der **linken** Seite werden die Eingänge SLEEP und RESET nicht benötigt. Damit sie nicht stören, müssen sie verbunden werden. Über die Eingänge ENA, STEP und DIR wird der Motor angesteuert – näheres später.
Mit den drei Eingängen MS1, MS2 und MS3 wird das Tempo des Motors eingestellt – auch dazu später mehr.

Auf dem Modul befindet sich ein Mini-Potentiometer, mit dem der Motorstrom eingestellt werden kann. Dieses sollte man so weit nach links drehen, wie der Motor sich noch gut dreht. In der Mittelstellung benötigt dieser kleine Motor sonst locker 300 mA – die Frage ist nur, wie lange.

4. Die Schaltung

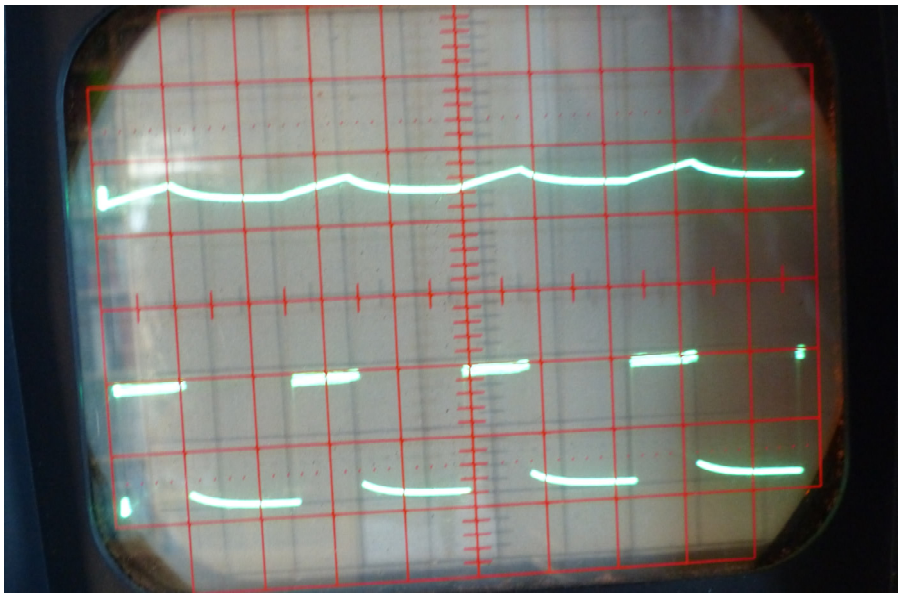


Dies ist die einfache Schaltung. Wie man sieht, kommt man mit zwei Signalleitungen des WS2811 aus, so dass über die dritte Leitung noch eine LED, z.B. die Signalbeleuchtung, oder eine Weichenlaterne, geschaltet werden kann.
Anmerkungen: Der 47 uF-Kondenator kann bei einer geeigneten Spannungsquelle sicher auch kleiner sein. Die +5V sollten vom „Ausgang“ des WS2811 genommen werden. Alle GND/Masse-Anschlüsse sind (teilweise intern) verbunden.

5. Die Funktionsweise der Schaltung

Der Kondensator C4 soll die Eingangsspannung von eventuellen Schaltspitzen säubern.
Die drei (Pull-Up-)Widerstände werden benötigt, damit die Ausgänge des WS2811 belastet werden und dann auch Low/High-Signale an den Eingängen des A4988 erzeugen.
Der Schrittmotor wird angesteuert, wenn das ENA-Signal auf 0 V liegt (logisch 1) und am STEP-Eingang ein Rechtecksignal anliegt. Mit jedem Signal schaltet der Motor einen Schritt weiter, wobei der DIR-Eingang die Richtung angibt.
Da wir das Signal vom WS2811 bekommen, können wir kein sauberes, dauerhaftes Low-Signal bekommen, weil 2000 mal pro Sekunde der Zustand des WS2811 upgedatet wird und die Ausgänge dann für einen kurzen Augenblick auf High (Logisch 0) gehen. Um dies auszufiltern, wird der Kondensator C3 benötigt.
Unser erster Schaltungsentwurf war, einen Ausgang des WS2811 auf ENA und einen Ausgang auf STEP zu legen und mit dem Pattern-Configurator dort ein periodisches Rechtecksignal zu erzeugen – das ist schief gegangen, wie uns Hardi schnell erklärt hat. Aber, ein Rechtecksignal muss auch gar nicht mit einem Programm erzeugt werden, denn wenn man an einem Ausgang z.B. den Helligkeitswert 127 ausgibt, realisiert dies der WS2811 dadurch, dass er für die Hälfte der Zeit den Ausgang an- und für die andere Hälfte ausschaltet, also von sich aus ein Rechtecksignal mit der Schaltfrequenz von 2 kHz erzeugt. (Leider kann man diese Frequenz nicht verändern.)
Von Hardi stammt der Schaltungstrick mit dem Kondensator C2 und der Universaldiode DUS, die einen Ausgang des WS2811 spart und die Schaltung auch noch sicherer macht (Danke Hardi, auch wenn ich dies

nicht sofort einsehen wollte). Dieser Trick funktioniert so, dass der Motor nur so lange aktiv sein soll, wie der STEP aktiviert ist. In diesem Fall wird, wenn STEP Low ist, der Kondensator über die Diode entladen und der ENA-Eingang bekommt auch Low-Potential. Wenn der STEP dann wieder für die restliche Zeit High wird, steigt die Spannung am ENA wieder, aber bei richtig dimensioniertem Kondensator C2 so langsam, dass der A4988 nicht ausschaltet.
Mit den angegebenen Werten sieht dies so aus:



Unten das STEP-Signal zwischen ca. 0,3 und 4 V, oben das ENA-Signal, das zwischen ca. 0,9 und 1,2 V schwankt. Die Low-Zeit ist etwas größer gewählt, damit sich der Kondensator während der High-Zeit nicht zu sehr aufladen kann.

Ohne weitere Schaltung wäre damit der Motor sehr schnell – unter 1 Sekunde von Anschlag zu Anschlag (wenn man einen SMD-WS2811 benutzt). Hier kommen jetzt die Anschlüsse MS1 bis MS3 zum Einsatz. Diese Eingänge sind eigentlich dafür gedacht, die Positioniergenauigkeit des Motors zu definieren, aber gleichzeitig bedeutet eine höhere Positioniergenauigkeit auch, dass mehr Schritte für eine volle Umdrehung benötigt werden und damit der Motor langsamer wird. Hier die Übersicht:

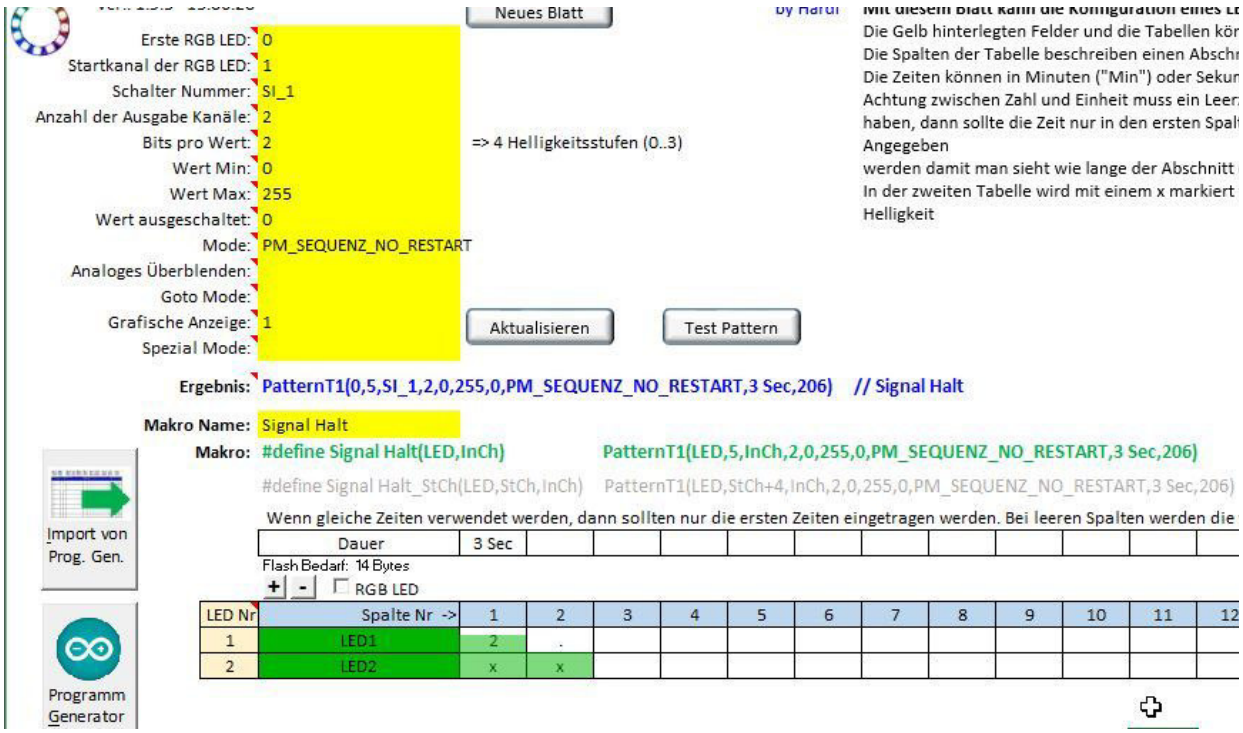
MS1	MS2	MS3	Genauigkeit/Schrittweite	Ca. Zeit für Bewegung von Anschlag zu Anschlag
-	-	-	Full Step (schnell)	0,5 Sekunden
X	-	-	1 / 2 Step	1,1 Sekunden
-	X	-	1 / 4 Step	2,25 Sekunden
X	X	-	1 / 8 Step	4,5 Sekunden
X	X	X	1 / 16 Step (langsam)	9 Sekunden

X = verbunden/ - = offen (Bei unseren Videos war 1 / 4 Step eingestellt.)

Hardi hat seine Schaltung und seine Platine noch so erweitert, dass man das Tempo in einem größeren Bereich einstellen kann. Außerdem hat er Abschalter für die Endstellungen vorgesehen, die man aber bei diesen Mini-Steppern nicht benötigt.

6. Die Programmierung

Für die Programmierung benötigt man ein einfaches Pattern. Hier für eine Bewegungsrichtung :



Erklärung:

Grundeinstellung: 2 Ausgabekanäle, 2 Bit pro Wert, Wert Max 255. Wichtig: PM_SEQUENZ-NO-RESTART (lässt den Befehl nur einmal ablaufen)

Zeit 3 Sek passte bei uns, der STEP lag auf Ausgang LED Nr 1und wird für 3 Sekunden auf einem mittleren Wert (Stufe 2) gehalten, was ein fast symmetrisches Rechtecksignal erzeugt und dann für weitere 3 Sekunden ausgeschaltet (was man auch verkürzen könnte). Bitte den Punkt in der zweiten Spalte nicht vergessen!

Der zweite Ausgang (LED Nr 2) wird dauerhaft eingeschaltet (bei der umgekehrten Bewegung ausgeschaltet). Demnach ändert sich beim Pattern für die umgekehrte Bewegungsrichtung nur der letzte Wert von 206 auf 2 – dies kann man auch im Programmgenerator von Hand machen.

Das Programm:

	Aktiv	Filter	Adresse oder Name	Typ	Start-wert	Beschreibung	Verteiler-Nummer	Stecker-Nummer	Beleuchtung, Sound, oder andere Effekte	Start Led	LEDs	Mors
3						Zeigt an, dass die LEDs angesteuert werden			RGB_Heartbeat(#LED)		1	
4												
5	✓					Wechselblinker (pc)			PatternT1(#LED,128,#InCh,2,0,128,0,PM_NORMAL,1	0	C2-3	1
6	✓								ConstRGB(#LED, #InCh, 0, 0, 0, 127, 127, 127)	1	1	1
7	✓					Signalbeleuchtung			Const(#LED, C1, #InCh, 0, 255)	2	C1-1	1
8	✓		4	Rot		Signal Halt			PatternT1(#LED,5,#InCh,2,0,255,0,PM_SEQUENZ_NO_	2	C2-3	1
9	✓								// Next_LED(-1)	3	-1	0
10	✓		5	Grün		Signal Fahrt			PatternT1(#LED,5,#InCh,2,0,255,0,PM_SEQUENZ_NO_	2	C2-3	1

Die Signalbedienung beginnt in Zeile 7 mit dem Einschalten der Signalbeleuchtung (die ersten Zeilen resultieren daraus, dass noch zwei RGB-LEDs vorher im Strang liegen).

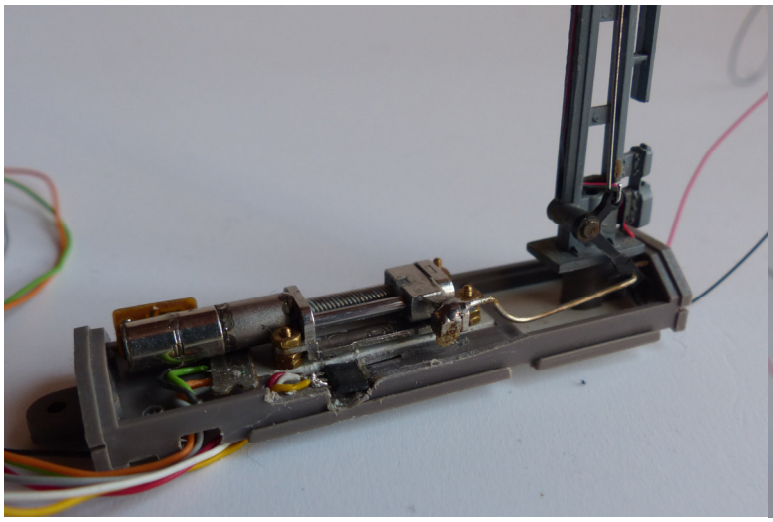
In Zeile 8 und 10 sind die erzeugten Pattern (einmal mit dem letzten Wert 206, einmal mit 2).

Dazwischen muss mit dem Befehl „// Next_LED(-1)“ die Ausgabereihenfolge einen Schritt zurück gestellt werden, da ja beide Befehle die gleichen Ausgänge benutzen.

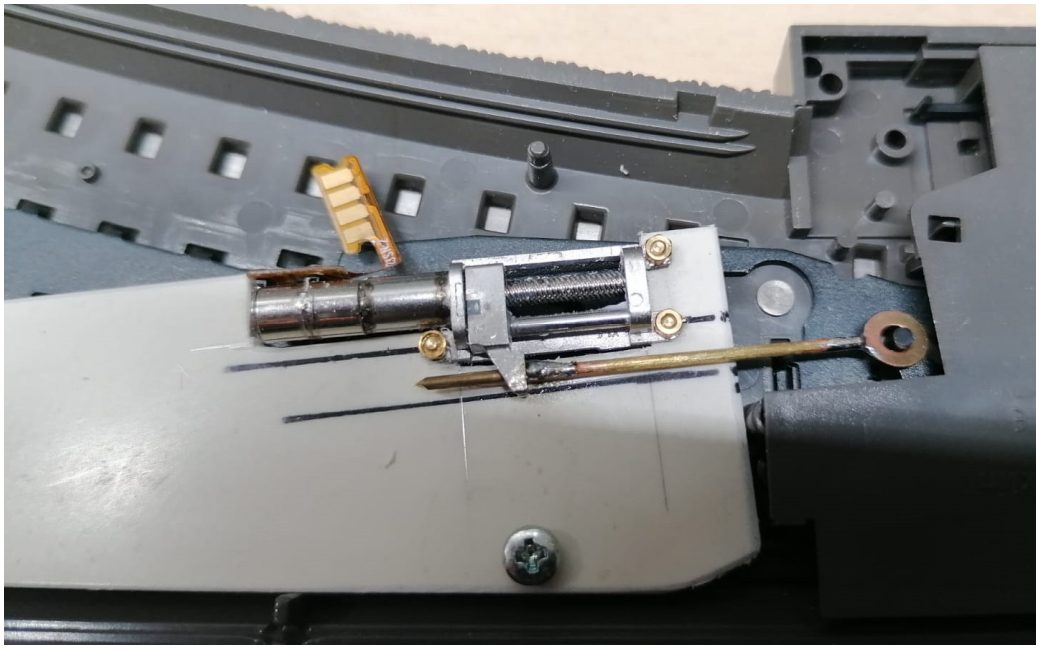
Bei welchem Befehl sich der Motor wie herum bewegt, muss man ausprobieren.

7. Der Motoreinbau

Statt langer Erklärungen lieber drei Bilder:



Bei dem Signal wurde der Magnetantrieb entfernt und der Motor auf eine Bodenplatte geschraubt. Zusätzlich hat Michael noch einen Hallsensor eingebaut, der aber nicht gebraucht wird. Das Lampengehäuse hat er gekürzt und die Lampe durch eine kleine Diode ersetzt. Damit wird das Signal zwar immer noch nicht originalgetreu, aber doch schon schöner.



Auch hier hat Michael den Motor auf eine Montageplatte gebaut. Die Antriebe hat er mit Fohrmann Lötwasser leicht benetzt und dann angelötet – bei der Weiche hat dies aber noch nicht so richtig funktioniert. Er tüftelt noch.

8. Wie geht es weiter?

Wir planen einen Platinenentwurf (wird Walter machen), wo man die Elektronik mit dem aufgeschraubten Motor unter Märklin C-Weichen setzen, oder (unter der Platte montiert) zum Steuern von Formsignalen, Toren und Türen benutzen kann.
Eine Erweiterung – bei der wir nicht so richtig einschätzen können, ob sie sinnvoll ist, wäre noch eine DCC-Dekodierung auf die Platine zu integrieren, um sie wieder unabhängig von der MobaLedLib zu machen. Vielleicht hat ja jemand Lust dazu.