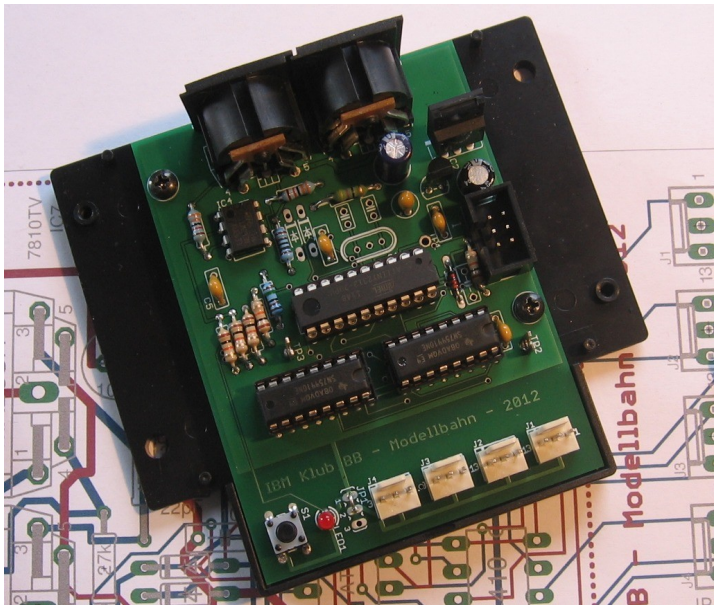
Selectrix Weichen- und Signaldecoder mit ATtiny2313

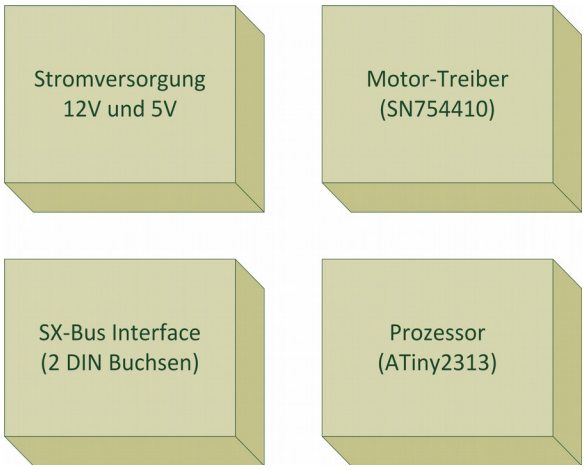


Da in unserem Club die (selbstgebauten) Selectrix Decoder so langsam in die Jahre kommen und sie nach dem Umstieg von Relais-Antrieben auf Weichenmotoren ein nachgeschaltetes Relais brauchen, sollten sie durch eine neue, rein elektronische Lösung ohne Relais ersetzt werden.

Basierend auf dem im letzten Artikel beschrieben „SX Arduino“ entstand so ein Selbstbauprojekt, programmiert in der Sprache C wie der Arduino. Dies ist jetzt so etwas wie ein „abgespeckter“ Arduino und nur die für einen Weichendekoder wirklich notwendigen Bauteile bleiben auf der Platine und der Prozessor wird durch den einfacheren aber ausreichenden ATtiny2313 ersetzt. Die ganze Platine passt dann auch in ein handelsübliches Modulgehäuse, welches auch in manch kommerziellem Selectrix Projekt eingesetzt wird. Diesen Projekten wollen wir hier übrigens keine Konkurrenz machen, sondern experimentierfreudigen Lesern Lust auf selbstgebaute Elektronik machen. Daher werde ich auch auf mögliche Schaltungsvarianten eingehen.

Hardware

Hier zunächst die für einen universellen Weichen- Signaldecoder notwendige Hardware im Blockschaltbild mit den Baugruppen „SX-Bus“, „Stromversorgung“, „Prozessor“ und „Treiber“;



Stromversogung

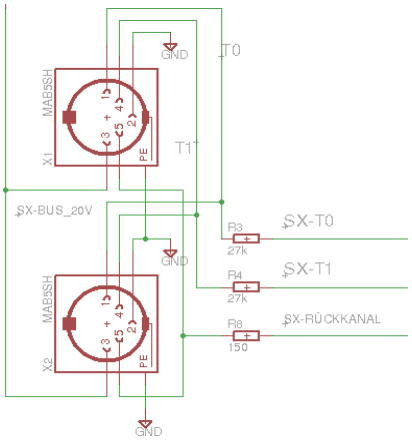
Für den Prozessor brauchen wir 5 Volt und für (unsere) Weichen und Signale brauchen wir 12 Volt. Da wir im Club Tortoise® Weichenmotoren verwenden, die nur etwa 15mA benötigen, haben wir uns entschieden, diese komplett aus dem SX-Bus zu versorgen. Mit zwei integrierten Spannungsreglern vom Typ 7812 und 7805 werden aus den 20V, die der SX Bus zu Verfügung stellt, 12V und 5V. (Um sparsamer mit den 20V umzugehen, kann man für den 12V Regler auch eine pin-kompatible

geschaltete Version verwenden). Eine zusätzliche Buchse (und der Jumper POW) erlaubt es allerdings auch, ein externes stabilisiertes 12V Netzgerät, anzuschliessen für den Fall, dass die angeschlossenen Weichen oder Signale höhere Ströme benötigen - solche Netzteile setzen sich immer mehr durch und es gibt sie daher recht preiswert ab etwa 5€. In diesem Fall wird IC4 (und die entsprechenden Kondensatoren) nicht bestückt, dafür aber D2 – die wegfällt bei rein interner Spannungsversorgung.

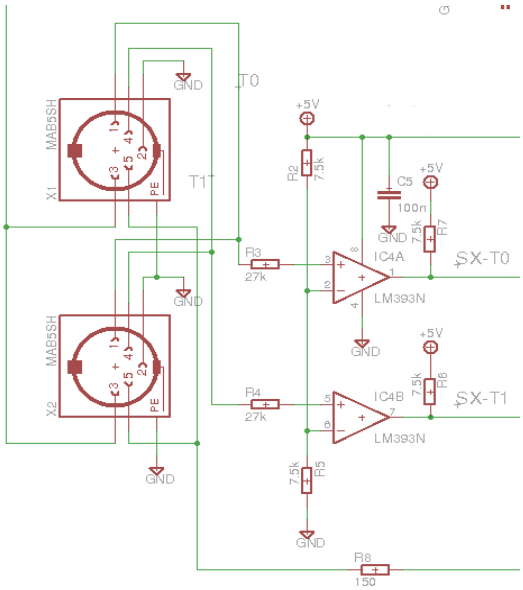
SX-Bus Interface

Üblich werden für den Selectrix Bus die 5-poligen DIN Buchsen verwendet, die wir auch bereits beim Arduino verwendet haben. Die Verwendung von 2 Buchsen an jedem Dekoder erlaubt die einfache Verkettung, die Weiterleitung des Busses an den nächsten Dekoder ohne zusätzliche Verteilerdosen. Für die Verbindung mit dem Prozessor verwenden wir (wie beim Arduino) die sehr einfache Variante mit den beiden 27K Längswiderständen für Takt- und Datensignal (alternativ werden hier oft LM393 Komparatoren eingesetzt, siehe Alternativ-Schaltplan – dies hat den Vorteil einer geringeren Busbelastung).
Neu im Schaltplan (gegnüber dem Arduino Interface) ist der zusätzliche Taster und der LED, mit deren Hilfe der Dekoder in den Programmierzustand versetzt werden kann (sofern die Gleisspannung abgeschaltet ist). Über den 150 Ohm Widerstand am AVR-Pin XX (der normalerweise hochohmig geschaltet ist und nur während des Zurückschreibens logisch 0 oder logisch 1=+5V ausgibt) können Daten auf die „Rückleitung“ zur Zentrale gegeben werden. „SX-typisch“ wird die Dekoder Adresse durch Schreiben einer Zahl in die Adresse 0 programmiert, weitere Einstellmöglichkeiten gibt es bei unserem Dekoder (siehe Tabelle unten) durch Daten in den Kanälen 1 bis 3 - mehr dazu im nächsten Artikel über die Software
Ein längerer Druck auf einen Taster erlaubt es uns, den Dekoder in diesen „Programmiermodus“ zu versetzen. Die LED blinkt während des Programmiermodus, um uns eine Rückmeldung zu geben, ob der Dekoder noch im Programmiermodus ist – sie erlischt, wenn wir diesen Modus wieder verlassen.

Schaltplan SX Bus interface einfach



mit Komparator



Prozessor

Wir verwenden den ATtiny2313 statt des ATmega328 im Arduino, er reicht für unsere Zwecke völlig aus (das compilierte Programm ist nur 1.5kByte groß, sollte es größer werden, so kann man noch auf den ATtiny4313 ausweichen mit 4kB Speicher). Programmiert wird der Prozessor über eine ISP Schnittstelle(„in-system-programmer“) - d.h. Es wird direkt in der Schaltung programmiert, zum Beispiel mit dem ATMEL AVRISP (siehe Links am Ende des Artikels).

Treiberendstufen

Unser Ziel war ein möglichst universell verwendbarer Dekoder, mit dem es möglich sein soll, sowohl Signale anzusteuern, die mit +12V angesteuert werden müssen als auch Signale, deren LEDs über den Kontakt nach Masse eingeschaltet werden. Über den Jumper JP1 kann selektiert werden, ob die gemeinsame Rückleitung auf +12V oder auf Masse liegt. In dieser universellen Schaltung verwenden wir daher eine Brückenschaltung, die im IC SN754410 gleich 2x enthalten ist. Die Endstufe dieses ICs hat eine Belastbarkeit von >1A, so dass (nur bei externer Spannungsversorgung, s.o.) auch Weichenantriebe mit den üblichen Spulen geschaltet werden können. Nachteilig bei diesen ICs ist der relativ hohe Stromverbrauch von ~50mA bei 5V (wenn Sie also ausschliesslich gegen Masse geschaltete Weichen und Signale verwenden, dann ist eine Open-Collector Lösung ausreichend und besser). Zum Anschluss der Weichen und Signale benutzen wir „PSK“-Crimp Stecker und Buchsen, die den Vorteil haben, dass man auch schnell mal wieder einen Dekoder ausbauen kann.

Fig. 1 Schaltplan.

http://www.lanbahn.de/files/sx-wdec1-schematics.pdf

Im nächsten Teil des Artikels werden wir uns mit der Software für den ATtiny befassen, ganz eilige können sich die Software auch bereits von http://www.oscale.net/sx-wdec herunterladen.

Link zum Thema Programmieren von ATMEL Prozessoren:

http://www.mikrocontroller.net/articles/AVR