MoBa-Geschwindigkeitsmesser

Inspiration:

Nachdem ich mal wissen wollte wie schnell eigentlich meine Loks sind,
habe ich mich auf die Suche begeben. Als Arduino-Anfänger dachte ich mir, dass es sowas doch schon geben müsste.
Auf der **Homepage von N-Modellbahn** bin ich dann fündig geworden.
Daraufhin habe ich für mich den Sketch auf meine Vorstellungen abgeändert:

1. H0 angepasst
2. Logo vom Stummiforum eingebaut
3. Oled-Displayanzeige auf meine Wünsche optimiert

Eine ganze Zeit später habe ich das Projekt dann mal auf einen Stummi-Stammtisch vorgestellt.
Hier kam mein Freund Tobias auf die Idee, es wäre doch super wenn das Ganze auch mit I-Train funktionieren würde.

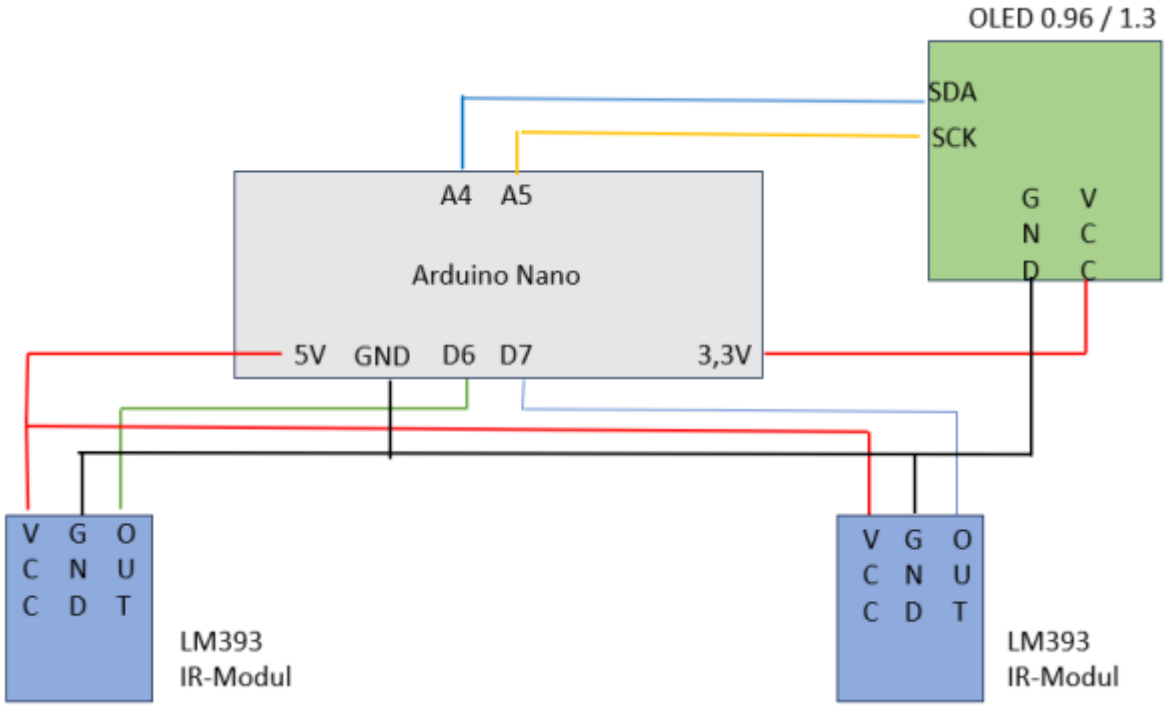
Daraufhin haben Tobias, Hardi und ich die Köpfe rauchen lassen und in die Tasten gegriffen
und das „Ur-Programm“ entsprechend angepasst, bzw. umgeschrieben.
Was dabei rausgekommen ist seht Ihr nun in der weiteren Beschreibung.

Komponenten:

1. Arduino 
2. IR-Module LM393 
3. OLED-Display 0.96 oder 1.3 SPI 
4. ein paar Kabel, evtl. ein Gehäuse oder sonstiges um die Komponenten drauf zu befestigen

Der Aufbau:

Es ist eigentlich ganz einfach gem. dieses Schaubildes:



Was man beim Bau messen muss, ist der Abstand zwischen den Dioden.
Das sollte recht genau passieren, das es die Grundlage für eine genaue Messung darstellt.
Auch ist es am Besten die Module so zu montieren, dass die Dioden senkrecht übereinander sind.

Je weiter die IR-Dioden auseinander sind, desto genauer werden die Messungen. Allerdings würde ich nicht über 1 Meter Abstand gehen und die Lok sollte die Möglichkeit haben gleichmäßig fahren zu können

Die Stromversorgung des MoBa-Geschwindigkeitsanzeigers kann über den USB-Anschluss und mittels Handy-Ladekabel oder Powerbank erfolgen.
Dies ist praktisch, denn dann kann man sich die Daten auch am „Seriellen Monitor“ der Arduino IDE anschauen.

Wer die Geschwindigkeit nur auf dem OLED-Display angezeigt bekommen möchte, kann selbstverständlich eine Stromeinspeisung
des Arduinos z.B. auch über die PINs **VIN** und **GND** machen.

Somit ist man sehr flexibel und kann die Messstrecke auf der Anlage dort unterbringen wo man möchte und es den meisten Sinn macht.

Die Messgenauigkeit liegt bei mir beim Abstand der IR-Dioden von 290mm zwischen 2-4 km/h
Diese kleine Toleranz dürfte für so ein kostengünstiges und einfach zu bauendes Projekt kein Hindernis darstellen,
denn auch in den Original-Lokomotiven haben die Tachos Toleranzen und zeigen die Geschwindigkeit nicht ganz genau an

Funktionsweise

Die beiden IR-Module senden sobald der Arduino mit Strom versorgt wird über die weiße IR-LED ein Signal aus, welches
dann bei Reflexion über die dunkle Sensor-LED wieder eingefangen wird. Fährt nun die Lok beispielsweise von links nach rechts,
wird bei Durchfahrt das linke Modul ausgelöst und die Messung gestartet und der Arduino fängt an mit der Zeitmessung.
Kommt die Lok nun bei dem rechten Modul an, wird die Messung gestoppt, sowie die Zeitmessung des Arduinos.
Nun berechnet der Arduino nach der Grundformel **Geschwindigkeit = Weg / Zeit** ($v=s/t$) die durchschnittliche Geschwindigkeit der Lok zwischen den Messpunkten.

Auf dem OLED-Display wird nun die gemessene Zeit in Sekunden (s), der Weg in Meter/ Sekunde (ms) und gleich in km/h unter Berücksichtigung des Maßstabs angezeigt.
Zudem zeigt das Display auch den Maßstab an, in welchem das System rechnet.

Software und Anpassung an die Gegebenheiten

Bei diesem Punkt gehe ich davon aus, dass derjenige der das Projekt nachbaut weiß wie man mit der Adruino-IDE, den Bibliotheken umgeht und wie man einen Arduino mit der IDE kompiliert, daher erkläre ich es hier nicht.

Folgende Bibliotheken der Arduino IDE werden benötigt, falls nicht vorhanden müssen diese noch installiert werden.

- Wire.h (I²C-Bus)
- Adafruit_GFX.h (Bildschirm-Bibliothek)
- Adafruit_SSD1306.h (für OLED 0.96)
- Adafruit_SH1106.h (für OLED 1.3)

In der Software (Sketch) sind ein paar Dinge einmalig vor dem Kompiliervorgang einzustellen, damit alles richtig funktioniert:

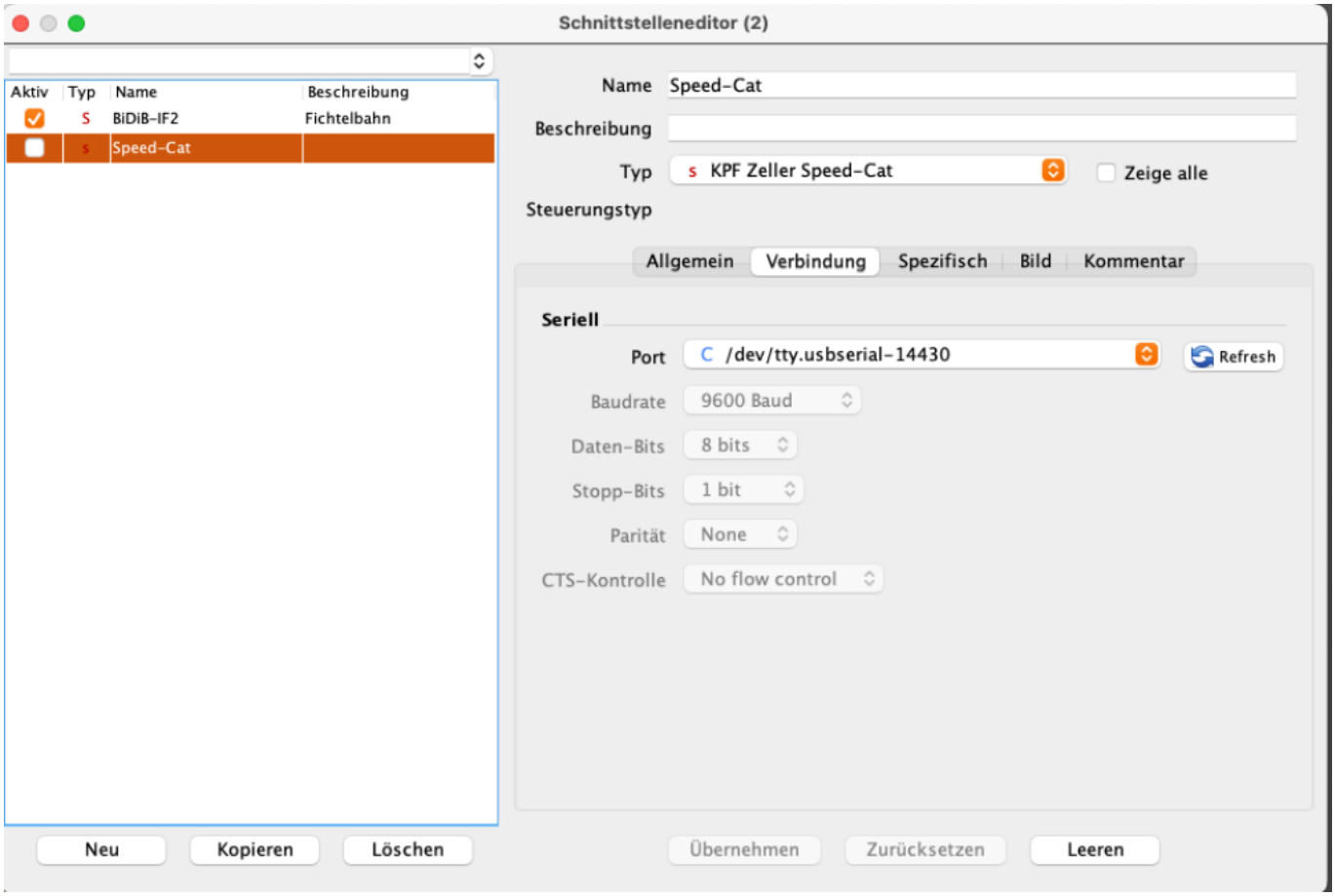
1. in Zeile 22 die Displaygröße definieren (0.96 voreingestellt)
2. in Zeile 25 der genaue Abstand in mm zwischen den IR-Sensoren (290.0 Achtung Werte genau so eingeben mm PUNKT Zehntel)
3. in Zeile 29 der Maßstab der Modelle (1/87 = H0 voreingestellt)
4. in Zeile Einstellung der Messfunktion (0 voreingestellt = nur Messung, für I-Train aus der 0 einfach eine 1 machen)

Hier der Sketch zum Download

Sketch MoBa-Geschwindigkeit

Integration iTrain (von Tobias)

An dieser Stelle erstmal herzlichen Dank Tobias, dass ich Deine Dokumentation verwenden und allen zur Verfügung stellen darf.
Die Integration in iTrain ist recht einfach.
Die Software produziert die Ausgabe wie der Speedcat von KPF Zeller.
Dazu gibt es eine in iTrain eine vorhandene Schnittstelle.
Man öffnet unter Bearbeiten den Schnittstelleneditor (Command F6)



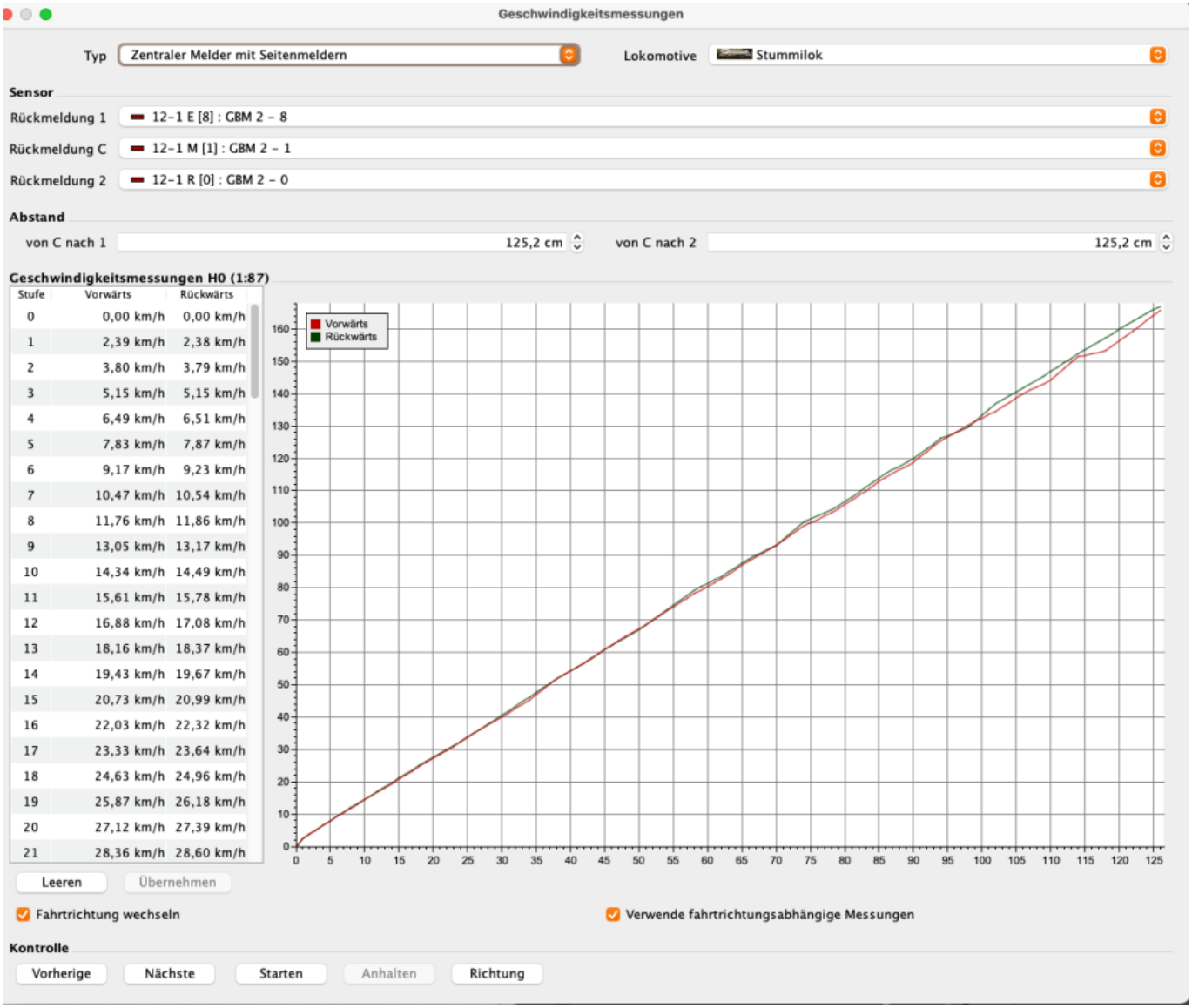
Nun legt man eine neue Schnittstelle an und wählt unter Typ den KPF Zeller Speedcat aus.
Falls diese nicht im PullDown Menu angezeigt wird auf die Checkbox Zeige alle klicken.
Nun noch den richtigen Port auswählen und das wars.
iTrain erkennt den Geschwindigkeitsmesser ab sofort immer, sobald er angeschlossen ist.

Solltet ihr den GWKM nach dem iTrain Start anschließen, müsst ihr noch die Schnittstelle verbinden.

Die Messung mit iTrain

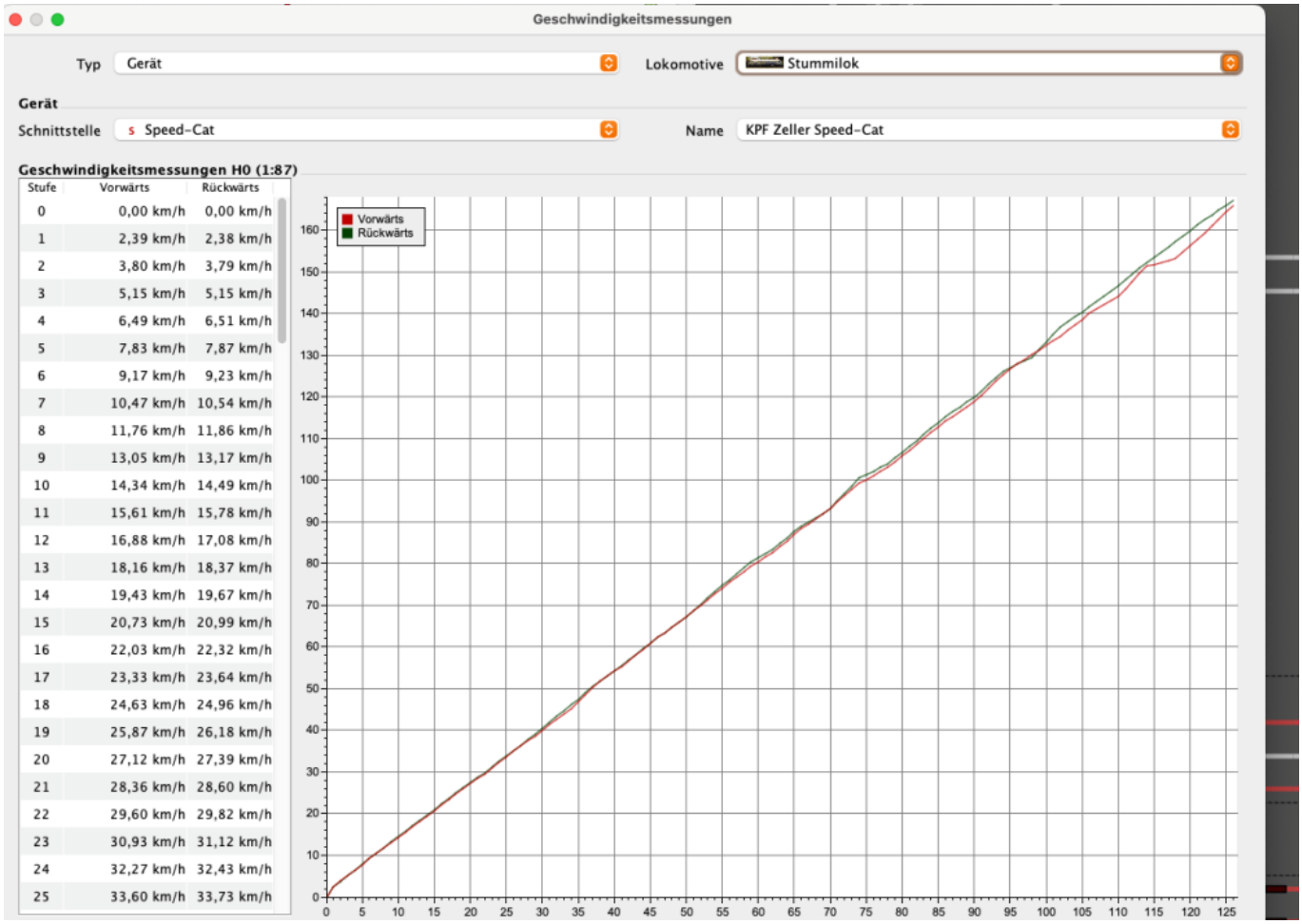
Auf der Anlage sollte man zur Messung einen entsprechenden Abschnitt auswählen,
damit die Loks auch genügend Auslauf für die hohen Geschwindigkeiten haben.
Das gilt natürlich generell, egal wie die Geschwindigkeit ermittelt wird.
Ein separates Gleisoval ist auch eine Variante.
Zum Vergleich habe ich eine Messstrecke gewählt, die ich für diese Zwecke auf meiner Anlage integriert habe.

Dazu verwende ich die drei Melder Methode:



In iTrain heißt das „Zwei Melder mit Seitenmeldern“.

Mit dem GWKM wählen wir unter TYP „Gerät“ aus.
Bei angeschlossenen und verbundenem GWKM wählt iTrain das richtige Gerät selbst.
\\ Das sieht dann so aus:



Wichtig ist, dass die Lok „warm“ gefahren ist.
Dazu fahre ich 2 komplette Runden in einer mittleren Fahrstufe, da wir die Geschwindigkeiten noch nicht kennen.
Anschließend beginnen wir mit dem Messvorgang.
Die Abweichungen zwischen beiden Messmethoden kann man vernachlässigen. Es kamen Abweichungen von ca. 1km/h heraus.

Fazit:

Das hat mich überzeugt, so dass ich den GWKM fest installieren werde. Die Anlage wird in Zukunft über einen headless Raspi gesteuert, so dass das USB-Kabel auch nicht zu lang wird.

Nachtrag

Wir dachten, den Sketch auch für WinDigipet freigeben zu können.
Leider lief ein Test mit WinDigipet erfolglos.
In WinDigipet gibt es zwar auch einen Speedcat, aber es wurden keine Daten eingelesen.

Das wäre noch auszuprobieren.
Vielleicht hat ja jemand Erfahrungen mit WinDigipet.

Wir, Jürgen (fromue) und Tobias können es leider nicht ausprobieren, da wir keinerlei Software zu Steuerung benutzen/ haben.
Hier hoffen wir auf Eure Hilfe.
Sollte jemand das Projekt erfolgreich mit WinDigipet zum Laufen bringen, so bitten wir um Bescheid (gerne im Stummiforum)