MoPi - Die Modellbahn-Zentrale

Für einen Kunden soll eine Modellbahnzentrale entwickelt werden. Neben der üblichen Zentralenfunktionen soll als Alleinstellungsmerkmal der PC für das (automatisierte) Steuern sowie dem Gleisstellpult integriert werden.

Hardware

Der Rechner

Raspberry Pi ist ein lowcost Rechner (Preis bei Farnell derzeit 29,03 EUR) mit einen leistungsfähigen ARM-Controller (ARM11 v6 Instruction set 800MHz, 256MB RAM-Speicher, integrierte GPU). Das Betriebssystem wird auf einer SD-Karte installiert. Eine kleine SD-Karte z.B. von Hama dürfte um die 3-4 EUR kosten.

Weiterhin hat der Pi die folgenden für dieses Projekt interessanten Schnittstellen: Ethernet, 2x USB (für Maus und Tastatur), HDMI und VGA (für den Anschluss eines Bildschirms) sowie einer GPIO-Schnittstelle auf der I2C, SPI oder UART zu finden sind (siehe dazu weiter unten). Die Stromversorgung erfolgt über ein handelsübliches USB-Steckernetzteil. Für die Stromversorgung gibt es weiter unten noch ein Kapitel.

Eine mögliche Alternative wäre der APC von Via.

Das Betriebssystem

Auf dem Pi sind gängige Linux-Derivate (leider nicht Ubuntu) lauffähig. Wir haben mit Debian (Raspbian) die ersten Gehversuche gemacht. Enthalten ist auch ein X-Server mit dem die Visualisierung (Gleisstellpult und Loksteuerdialoge) gemacht werden kann. Fast alle gängigen Standardprogramme sind verfügbar (aufpassen wegen Lizenzierungen: Bei GPL-Software sollte der Quellcode mit dabei liegen). Sollten wir an die Ressourcen des Pis stossen, würden wir auf Arch Linux gehen.

Modelleisenbahn-Software

Als GPL-lizensiertes OpenSource-Projekt ist Rocrail zu nennen. Es vereint unter dem Projekt ein Kommandozeilen-Tool, den sogenannten Rocrail-Server. Dieser steuert die konkrete Hardware (Zentrale) an. Außerdem nimmt er über eine Socket-Schnittstelle Kommandos in XML-Syntax

entgegen. An der Socket-Schnittstelle können beliebig viele Clients andocken, die entweder auch auf dem Pi laufen (z.B. Rocview) oder externe Geräte (z.B. der eWicht-Handregler).

Der Maintainer von Rocrail, Rob Versluis, schränkt die Nutzung auf private Projekte ein. Dass steht jedoch nicht mit der GPL im Einklang. Wir wollen das Projekt selbverständlich finanziell unterstützen. Derzeitiger Stand (nach einem Telefonat) ist, dass wir, sobald das Produkt auf dem Markt ist einen festvereinbarten Betrag pro verkauften Gerät an das Projekt spenden (ca. 5 EUR).

Es gäbe noch zwei Alternativen zu nennen: - Das srcpd-Projekt, der Server für SRCP-Clients (ein sehr viel hardwarenaheres Projekt, ich denke, das ist nichts für Endverbraucher) - Eine Eigenkreation, basierend auf Webtechnologie (Webserver mit CGI und Datenbank als Server-Ersatz, die Clients würden dann über Browser die Anlage steuern). Einfach aus Standardbausteinen zusammenzusetzen, jedoch nicht mehr dieses Jahr.

MoPi-Treiber

Ich habe jetzt einen Treiber in Python geschrieben, der direkt mit dem MoPi-Controller über die serielle Schnittstelle kommuniziert. Wie performant das Ganze ist, wird sich noch zeigen.

Der Python-Treiber kommuniziert später mit dem Rocrail-Server über einen virtuellen Com-Port. Es wird dort das sogenannte P50X-Interface implementiert. Dieses Protokoll stammt von Uhlenbrock und wurde für die Intellibox entwickelt. Die Doku ist frei verfügbar. Diese Lösung hat den Vorteil, dass ich die Sourcen des Rocrail-Servers nicht ändern muss.

GPIO-Schnittstelle

Die GPIO-Schnitstelle beinhaltet eine UART-Schnittstelle. Diese kann bis 115200 Baud (mit gepachten Kernel auch mehr). Leider kann sie kein Hardware-Handshake. Weiterhin sind die Versorgungsspannungen 5V und 3,3V auf die GPIO-Leiste geführt.

Der MoPi-Controller

Der MoPi-Controller erzeut das DCC-Gleissignal und liest S88-Rückmelder ein. Außerdem wird das Active-Signal für einen Booster erzeugt (üblich bei Märklin-Boostern) und das Kurzschluss-Signal der Booster eingelesen. Für das Programmieren der Loks wird noch das ACK-Signal benötigt, welches ebenfalls vom MoPi-Controller eingelesen wird.

Die bidirektionale Kommunikation mit dem Pi erfolgt über die serielle Schnittstelle.

Achtung, der Pi verträgt hier nur 3,3 V.
. Die Schnittstelle sollte galvanisch getrennt ausgeführt werden (zwei Optokoppler, die bis zu 115200 Baud mitmachen). Die Kommunikation erfolgt derzeit mit 38400 Baud (Format 8N1). Es kommt ein Software-Handshake zum Einsatz.

Stromversorgung

Der Pi wird über ein handelsübliches USB-Steckernetzteil mit 5V versorgt. Diese Netzteile sind extrem günstig im Einkauf. Der Pi selber sollte mit 5 W geplant werden. Darüber hinaus müssen die folgenden Komponenten versorgt werden:

- MoPi-Controller @ 5V (30 mA) - Minimal-Booster für Programmierung @ 14-18V (ggf. einstellbar) (500mA-1A). - S88-Strang @ 5V (3 Stränge sind maximal möglich, pro Strang sollten 250mA geplant werden).

Variante 1

Separates Netzteil mit der entsprechenden Boosterspannung 14V für kleine Spuren, 18V für große Spuren.

Variante 2

Step-Up Converter, der aus der Pi-Spannung (5V) die entsprechende Boosterspannung (14-18V) macht. Nach Möglichkeit galvanisch getrennt! Die 5V für S88 und MoPi-Controller müssten extra erzeugt werden (ebenfalls galvanisch getrennt).

Sonstiges

Eine Aktiv-LED sowie Taster für Start/Stop.

Schnittstellen der Zentrale

Neben den Pi-Schnittstellen kommen noch die folgenden hinzu:

- Bis zu 3 x S88 in Form von RJ45-Buchsen oder 6 poligen Stiftleisten.
- Ein DCC-Booster Ausgang (dreipolig), schraub und steckbar
- Ein Märklin-Booster Ausgang (5-polig), Stiftleiste (semi-genormt)
- Möglicherweise der Eingang für ein Steckernetzteil oder Spielzeug-Trafo

Software

Controller-Firmware

Nach einigen Versuchen mit der Ur-Version des MoPi (interne Version 1.0, Controller PIC16F688), muss die Software grundlegend überarbeitet werden. Grundsätzlich funktioniert zwar das Ansteuern einer Lok, der derzeitige Ansatz verlangt dem Pi jedoch bereits über 50% Prozessorauslastung ab und das noch ohne X-Server.

Die serielle Schnittstelle wird derzeit im Streaming-Mode betrieben, der PIC hat nur einen sehr kleinen Ringpuffer. Aus dem Ringpuffer heraus werden die Daten generiert. Das erfordert ein ständiges Nachschieben von Daten auf PC-Seite. Das ist der Grund für die hohe Auslastung des Prozessors. Außerdem besteht immer auch die Gefahr, dass der Puffer z.B. bei erhöhter Prozessorlast leerläuft und die Loks dadurch stottern oder stehenbleiben.

Ein neuer Ansatz ist, mehr Zustand im MoPi zu halten, und die Steuersignale aus diesem Zustand zu generieren.

RaspberryPi

Alle Versuche mit dem Pi wurden bisher mit Raspbian gemacht (zuletzt mit der Version 2013-02-09).

Folgende Dinge wurden noch nachträglich manuell gemacht (als User “Pi”):

- Deutsches Tastaturlayout (von *gb* auf *de* umstellen): sudo nano /etc/default/keyboard
- Bazaar installiert (Für MoPi und Rocrail): sudo apt-get install bzr
- Samba installiert (optional, wird später nicht benötigt): sudo apt-get install samba
- Share angelegt (Zeigt auf work): mkdir work
  sudo mount.cifs //192.168.97.100/work work -o user:Sven
- Checke MoPi aus (später von launchpad): bzr checkout ~/work/MoPi ~/MoPi
- Checke rocrail aus: bzr checkout lp:rocrail ~/rocrail
- Befreie serielle Schnittstelle (alles löschen was irgendwie mit /dev/tty/AMA0 zu tun hat): sudo nano /boot/cmdline.txt
- Auskommentieren des getty für serielle Schnittstelle (#T0:23:respawn:/sbin/getty -L ttyAMA0 115200 vt100): sudo nano /etc/inittab
- Deaktivieren des Logins:
  - sudo nano /etc/inittab
  - Scroll down to 1:2345:respawn:/sbin/getty 115200 tty1
  - Comment out: #1:2345:respawn:/sbin/getty 115200 tty1
  - Under that line add: 1:2345:respawn:/bin/login -f pi tty1 </dev/tty1 >/dev/tty1 2>&1
  - Abspeichern
- Auto StartX
  - sudo nano ~/.bash_profile
  - Eingeben: startx
  - Abspeichern

- Script für Steuern der Stromversorgung installieren (siehe firmware folder in MoPi)