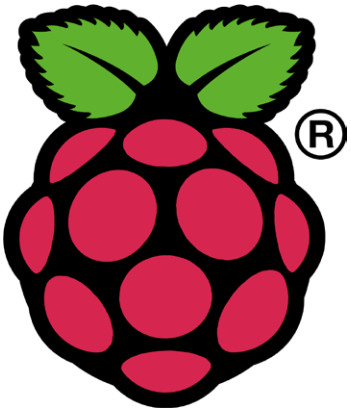
Projektbeschreibung iSwitchPi

Intelligenter Power Switch für Raspberry Pi

Beschreibung eines intelligenten Ein/Aus-Schalters für den Raspberry Pi



Inhaltsverzeichnis

1. Was ist ein Raspberry Pi

Der Raspberry Pi (genannt Pi) ist ein kreditkarten-kleiner und preisgünstiger (45 CHF) Linux Computer der sich seit seiner Einführung vor bald 4 Jahren grosser Beliebtheit erfreut und der bereits mehrere Millionen Mal verkauft wurde. Im Internet finden sich unzählige Projektbeschreibungen; alles Mögliche und Unmögliche wird mit diesem Micro-Computer gebaut. Der Initiant Eben Upton ist Engländer und in folgendem Video auf YouTube erklärt die Geschichte des Pi.

Eben Upton on YouTube

Der Pi hat 4 USB-Anschlüsse, einen Lan-Anschluss, ein HDMI-Anschluss für Monitor und einen Audio-Video-Ausgang. Ebenfalls auf dem Board ist ein Anschluss für die Pi-Camera - eine HD-fähige Kleinst-Camera für Fotos und Video. Was den Pi auszeichnet, ist die 40-polige Steckerleiste. Viele dieser Anschlüsse sind General Purpose Input/Output Pins, welche in Progammen angesteuert/gelesen werden können. Es gibt in der Zwischenzeit vielfältige Expansion Boards von unzähligen Anbietern. Der Pi konsumiert bloss 1 Watt Leistung: 5V und 100mA Strom.

Dieses Maschinchen hat mich von Anfang an fasziniert, ich hatte mir in 2012 eines angeschafft. Nach einigem Probieren legte ich es auf die Seite - fand einfach kein wirkliches Projekt. Dies hat sich im Frühjahr 2014 geändert und es kam die Idee auf für ein erstes Pi-Projekt: Lampensteuerung bei Ferienabwesenheit.

In der Zwischenzeit (Stand Anfangs 2016) sind bereits drei weitere Raspi-Projekte dazugekommen: Light-Painting Pixeslstab mit 144 Led und ein, wie ich es nenne, Christmax TV. Raspi und Volumio als Musikplayer in einer weihnachtlichen Box. Details zu diesen Projekten findet man auf der Projekt-Website.

Der Raspberry PI Model B+

2. Abstract

A Raspberry Pi does not have an On/Off switch and there is no easy way to shutdown the Pi while keeping the filesystem intact.

This Intelligent Power Switch brings a clever solution to this problem: Power-On the Pi by pressing a pushbutton and also properly Power-Off the Pi with another press on the same button.

The intelligence is provided by a program running in an AVR MCU ATtiny44. This C-program implements

a Finite State Machine in the MCU.

A small Python script is running in the Pi itself.

Just one GPIO-Pin is used for two-way communication between the Pi and the iSwitchPi board.

The iSwitchPi board additionally provides a square wave output with variable frequeny that can be used to trigger interrupts on the Pi.

3. Anforderungen

Während der Entwicklung verschiedener Gadgets mit dem Raspberry Pi tauchten immer wieder verschiedene Themen und Problemstellungen auf, für die ich gerne eine elegante Lösung gehabt hätte. So entstand der Plan, ein Add-on zu entwickeln. Folgende Anforderungen wurden definiert:

- Eine elegante Lösung für das Ein- und Ausschalten des Pi's zu haben.
- Das Bauteil soll in der Form eines Pi-Hats erstellt werden.
- Es soll bloss ein einziger GPIO-Pin des Pi für die Kommunikation verwendet werden.
- Das Bauteil soll wahrnehmen können, wenn der Pi nicht mehr läuft. Es soll in diesem Fall die Power -Down Sequenz gestartet werden.
- Das Bauteil soll überdies ein Rechteck-Signal von 0.5 Hz bis ca. 100 Hz erzeugen - viele Pi-Anwendungen benötigen eine externe Rechteck-Quelle, mit welcher Timer-Interrupts realisiert werden können. Ein Mikrocontroller erledigt diese Aufgabe ohne viel Programmier-Aufwand.

Wichtigste Anforderung: Bekanntermassen ist das Ausschalten eines Raspberry Pi (Power Off) - genauso wie das Ausschalten jedes anderen Computers - in zwei Schritten zu bewerkstelligen:

- a: der Pi muss ordentlich mit einem Shutdown oder halt Command heruntergefahren werden - dies kann nur in einem Stück Software auf dem Pi erledigt werden.
- b: erst danach kann extern die Stromzufuhr gekappt werden.

Das Einschalten ist dagegen simpel: Stromzufuhr einschalten und der Pi bootet und beginnt mit der vorgesehenen Arbeit.

Es gibt auf dem Markt verschiedene Add-ons zu kaufen, welches das Ein-Ausschalt-Problem lösen. Sie bestehen meist aus einer kleinen gedruckten Schaltung mit Anschlüssen für Drucktasten, 5 V und anderen nötigen Verbindungen. Diese kleinen Bauteile funktionieren gut, und es ist nicht nötig, die genaue interne Funktionsweise zu verstehen.

Dieses Projekt soll die genannten Anforderungen in Hardware und Software realisieren.

4. Vorgehen

4.1 Umschau auf dem Markt

Zuerst wurden einige auf dem Markt angebotenen Add-ons genauer studiert. Folgende Produkte und/oder Anleitungen wurden unter die Lupe genommen:

- Pi-Supply der Englischen Firma Pi-Supply. Dieses Add-on war schon bei mir im Einsatz, es funktioniert sehr gut, ungünstig ist jedoch, dass für Einschalten und Ausschalten zwei verschiedene Drucktasten verwendet werden. Die eingesetzte elektronische Schaltung ist an anderer Stelle im Internet zu finden. Kommt wegen der Lösung mit zwei Tasten nicht in Frage. Benötigt 2 GPIO Pins beim Pi.
- Pi-Shutdown Button bei Instructables: Lösung mit embedded Controller ATtiny13/85. Dies kommt

der Vorstellung schon sehr nahe.

- ATX-Raspi von LowPowerLab. Dieses Produkt kommt den Anforderungen schon weit entgegen, die kleine Platine könnte als Tochterplatine eingesetzt werden. Intelligenz wird durch einen embedded Controller geleistet.
- ykrud von Yepkit. Auch dieses Produkt scheint durchdacht - und dient als Anstoss, eine eigene Lösung zu entwickeln. Yepkit braucht aber 2 GPIO Pins des Pi.

Da bereits einige Kenntnisse in der Programmierung vom Atmel Mikrocontrollern vorhanden war, entschied ich mich, eine der obigen Lösungen etwas abgewandelt selbst zu entwickeln. Es soll ein ATtiny44 Mikrocontroller verwendet werden. Details dazu siehe Datenblatt.

4.2 Erkenntnis

Alle oben beschriebenen Add-ons bestehen aus 2 Komponenten, welche in bestimmter Weise zusammenarbeiten:

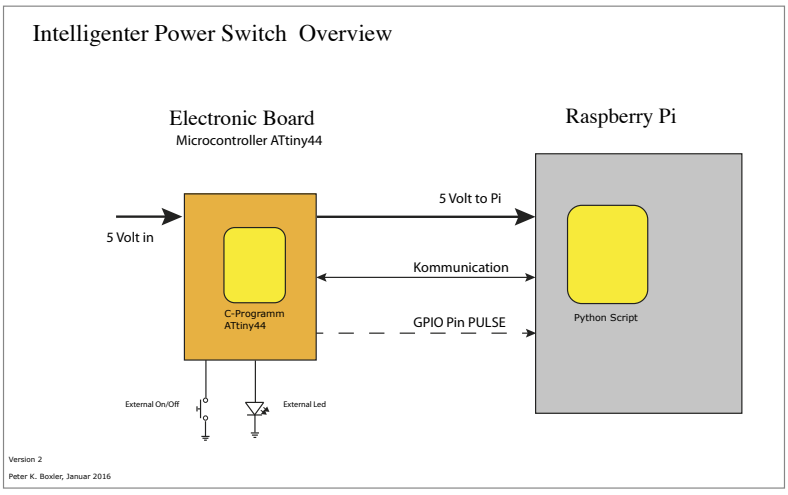
- Komponente A: eine kleine Schaltplatine, welche Elektronik-Komponenten enthält, die das Ein-/Ausschalten steuern. Sie ist mit dem Pi verbunden - die Stromversorgung des Pi geschieht also NICHT über den Micro-USB Anschluss. Zudem werden meist 2 GPIO-Pins des Pi für die Kommunikation belegt. Angestrebt wird jedoch die Verwendung eines einzigen GPIO Pins.
- Komponente B: ein Python Script, welches im Pi nach dem Boot gestartet wird.

Es ist klar, dass die eigene Lösung derselben Idee folgt.

Der Intelligente Power Switch wird also ebenfalls aus eine externen Elektronik-Komponente sowie einem Python Script bestehen.

5. Lösungsübersicht

Das hier beschriebene Add-on wird realisiert mit einer gedruckten Schaltung in der Grösse 38x47 mm. Herz der Schaltung ist ein Mikrocontroller ATtiny44, für den ein C-Programm entwickelt wurde, welches die gewünschten Funktionen implementiert (Firmware). Die kleine Platine kann in ein Breadboard gesteckt werden. Diese Platine wird iSwitchPi genannt.
Die Lösung stellt sich in der Übersicht so dar



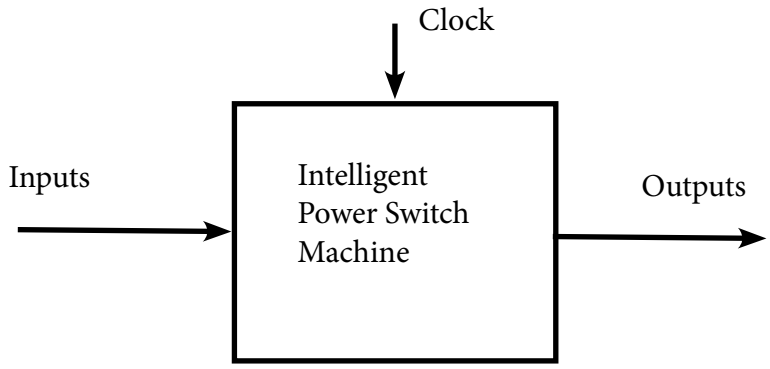
Übersicht der Lösung

Das Elektronik-Board (genannt iSwitchPi) sorgt für das Timing des Ein-Ausschalten des Pi. Im Pi läuft ein Shutdown-Script, welches via einen GPIO Pin mit dem iSwitchPi kommuniziert. Der iSwitchPi liefert zudem das Rechteck-Signal an den Pi.

6. Problembeschreibung

Erster Schritt jedes Elektronik- oder Software-Problems ist die genaue Analyse der Anforderungen. Neben der Ein-Ausschaltfunktion bestehen noch andere Anforderungen an das zu entwickelnde Bauteil.

Das vorliegende Problem eignet sich hervorragend zur Modellierung mittels einem Endlichen Automaten (Finite State Machine). Ein solcher Automat hat definierte Eingänge, Zustände, Übergänge zwischen Zuständen und ebenfalls definierte Ausgänge. Übergänge zwischen Zuständen werde durch Änderungen in den Inputsignalen verursacht. Dies wiederum hat Änderungen in den Output-Signalen zur Folge.



Problembeschreibung in Worten:

Es soll nur eine Drucktaste für Ein/Ausschalten/Reboot des Pi geben. Ist der Pi ausgeschaltet, genügt ein kurzer Druck auf die Taste, um den Pi mit Strom zu versorgen. Läuft der Pi, so kann mit einem kurzen Druck auf dieselbe Taste der Shutdown des Pi eingeleitet werden und nach einer definierten Verzögerungszeit wird auch die Stromzuführung zum Pi unterbrochen. Läuft der Pi, so kann mit einem langen Tastendruck ein Reboot des Pi ausgelöst werden - die Stromzufuhr bleibt in diesem Fall erhalten.

Läuft der Pi und es wird ein Shutdown via ein Terminal Fenster ausgelöst (oder es erfolgt ein programmgesteuerter shutdown des Pi) , so soll dies die Schaltung feststellen und die Stromzufuhr nach der definierten Verzögerungszeit unterbrechen.

Die Stromversorgung des Pi soll aber auch ohne Rückmeldung des Pi aufrechterhalten werden können - beispielsweise dann, wenn das Python Shutdown-Script noch nicht aktiviert ist oder wenn es fehlerhaft ist.

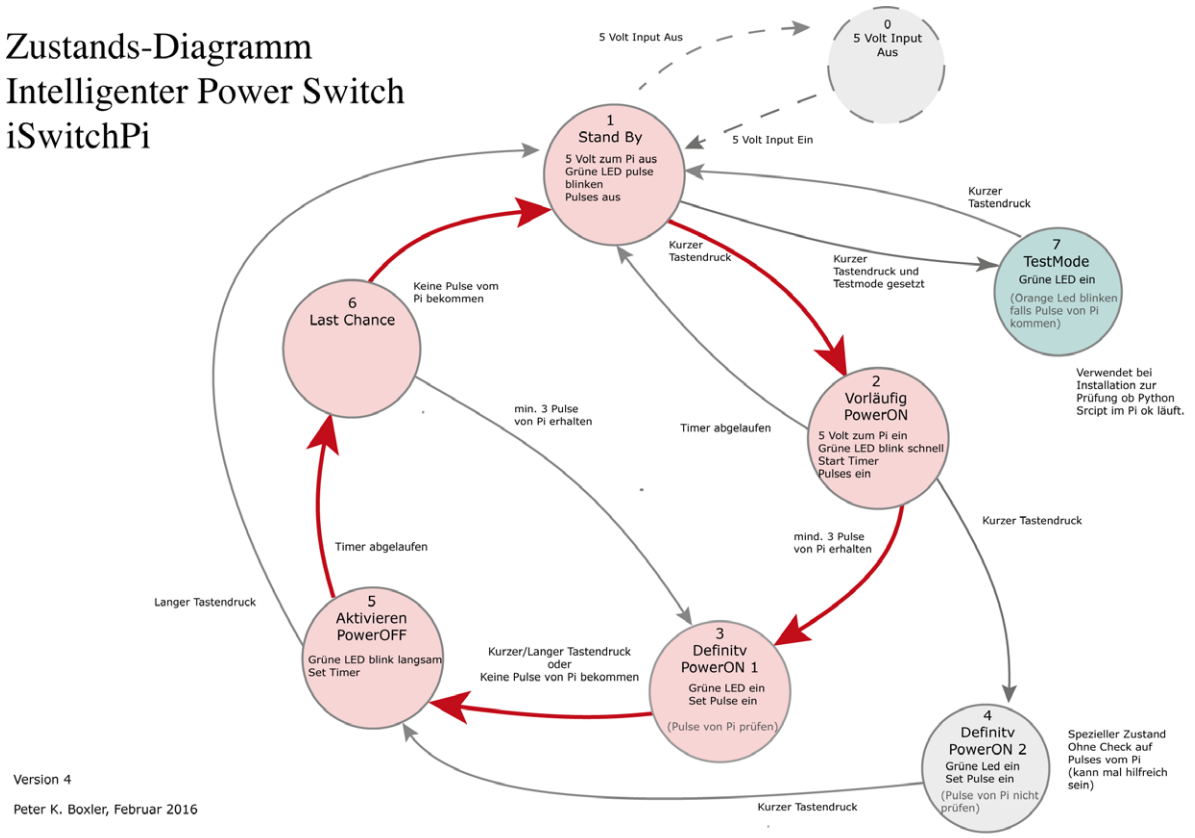
Eine solche Beschreibung in Worten ist immer etwas undeutlich, es sind meist Mehrdeutigkeiten vorhanden und die Funktion ist nicht wirklich genau beschrieben. Der vorliegende Fall wird vorzugsweise mit einem sog. ‚Endlichen Automaten' (Finite State Machine) modelliert.

Zuerst befassen wir uns mit dieser Elektronik-Komponente, danach definieren wir das Python-Shutdow-Script.

7. Komponente A: Elektronik

7.1 Zustands-Diagramm

Der iSwitchPi implementiert in Software und Hardware die Ein-/Ausschaltfunktion der Pi-Stromversorgung. Nach ein paar Versuchs-Skizzen wurde folgendes Zustands-Diagramm (State Diagram) für die Endliche Maschine gefunden. Das Zustanddiagramm und die Zustandtabellen definieren das zu erstellende C-Programm für den ATtiny44. Der Pi wird mittels eines Scripts (siehe weiter hinten) regelmässig Pulse senden, die signalisieren: ich bin noch am Leben, stell mir bitte den Strom nicht ab.



Zustandsdiagramm 1

7.2 Input und Output

Dies sind alle Eingangssignale des Automaten
(Each Input can be one or zero)

Eingang	**Abkürzung**	**Quelle**
Kurzer Tastendruck	KT	Drucktaste
Langer Tastendruck	LT	Drucktaste (>1.5 sec)
TimerSignal	TS	Timer erreicht definierte Zeit
SignalFromPi	SPi	Es kommen Pulse vom Pi
Test Modus eingestellt	TM	Dip-Switch

Ausgänge des Automaten

Ausgang	**Abkürzung**	**Was ist das**
5 Volt für Pi	5V	5 Volt zum Pi ein oder aus
Led an	Ledon/Ledoff	Grüne Led ist dauerhaft ein oder aus

Led blink slow	Ledsl	Grüne Led blinkt langsam
Led blink fast	Ledfst	Grüne Led blinkt schnell
Signal zum Pi für Reboot	StPireb	Sende einen Pulse zum Pi
Signal zum Pi für Shutdown	StPishut	Sende 2 Pulse zum Pi
Start Timer	STo oder STf	On /Off (meint start/stop Timer)
Pulsgenerator	ON/OFF	Pulsgenerierung ON/OFF

7.3 Zustands Tabelle

Die Zustandstabelle beschreibt die Funktion des endlichen Automaten komplett. Diese Funktion wird durch das C-Programm implementiert. Ein bestimmter Zustand plus Kombination von Eingangssignalen führt zu einem neuen Zustand. Implizit: ohne Änderung der Eingangssignale verbleibt der Automat im aktuellen Zustand (x heisst: irrelevant).

Zustand	**Eingänge**					**Nächster Zustand**
	SK	LK	TS	SPi	TM	
1 Stand by	1	x	x	x		2 Vorläufiges Einschalten
1 Stand by	1	x	x	x	1	7 Test Modus
2 Vorläufiges Einschalten	x	x	0	1		3 Power On-1
2 Vorläufiges Einschalten	x	x	1	0		1 Stand by
2 Vorläufiges Einschalten	1	x	x	x		4 Power On-2
3 Power On-1	0	0	x	0		5 Ausschalten aktivieren
3 Power On-1	1	0	x	0		5 Ausschalten aktivieren
3 Power On-1	0	1	x	0		5 Ausschalten aktivieren
4 Power On-2	1	x	x	x		5 Ausschalten aktivieren
5 Ausschalten aktivieren	x	x	1	0		1 Stand by
5 Ausschalten aktivieren	x	1	x	x		1 Stand by
6 Last Chance	x	x	x	0		1 Stand by
6 Last Chance	x	x	x	1		3 Power On
7 Test Modus	X					1 Stand By

7.4 Eingangs-Aktivitäten der Zustände

Jeder Zustand des Automaten hat im allg. sog. entry-actions, also Aktivitäten, die beim Eintritt in den Zustand genau einmal ausgeführt werden. Dies sind meist Änderungen in den Output-Signalen.

Zustand	**Eingangs-Aktivität**
1 Stand by	Stromzufuhr unterbrechen Grüne Led Pulse blink Timer stoppen Pulsgenerator stoppen
2 Vorläufiges Einschalten	Stromzufuhr einschalten Grüne Led schnell blinken lassen Timer starten Pulsgenerator starten
3 Power on-1 (Normalbetrieb)	Grüne Led voll einschalten Timer stoppen
4 Power on-2 (spezieller Betrieb)	Grüne Led voll einschalten Timer stoppen
5 Auschalten aktivieren	Grüne Led langsam blinken lassen Timer starten
6 Last Chance	nichts
7 Test Modus	Stromzufuhr einschalten Grüne Led voll einschalten

7.5 Elektrisches Schema

Die Hardware-Komponente des iSwitchPi soll in Form eines Raspberry Hats aufgebaut sein. Anwendung ist einfach: Aufstecken auf den Pi, 5 Volt Spannungsversorgung anschliessen und fertig. Es können weitere Hats auf den iSwitchPI gesteckt werden (stacking).

Das elektrische Schema besteht aus diesen Komponenten:

Die 5 Volt Spannungsversorgung des Pi wird durch einen High-Side Switch MCP2505 ein-/ausgeschaltet. Dieses Bauteil hat im Ein-Zustand einen Spannungsabfall von nur 15 mV und kann etwas mehr als 2 Ampère schalten.

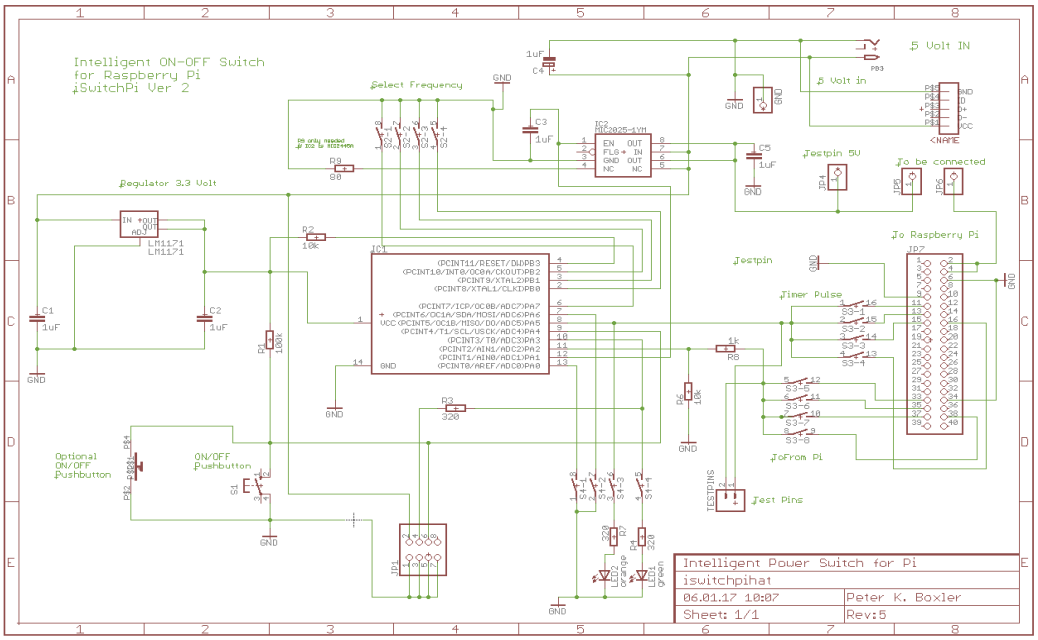
Die Eingangsspannung von 5 Volt wird für den Mikrocontroller im Baustein LM1171MP-3.3 auf 3.3 Volt reduziert. Damit ist auch die Kommunikationsleitung für den Pi mit diesem Pegel angesprochen (GPIO Pins für den Pi dürfen max. diese Spannung haben).

Im Pi läuft ein Python Script (iswitchpi.py), welches am gewählten GPIO Pin durch regelmässige Pulse meldet ‚ok ich laufe'. Das Script kann jedoch auch Pulse vom iSwitchPi empfangen (via Interrupt-Handler) und die Anzahl der eintreffenden Pulse bestimmen die Art des Commands, der ans OS abgesetzt wird: shutdown halt oder reboot.

Das Herz des iSwitchPi ist ein ATtiny44 Mikrocontroller. Allein für die Steuerung des Pulsgenerators und dessen Output werden 5 Pins benötigt - deshalb genügt ein ATtiny85 mit seinen total 5 Pins nicht.

C-Programm (Firmware) für den Mikrocontroller siehe weiter hinten.

Die Schaltung wurde im Programm Eagle PCB gezeichnet - das Programm erstellt ebenfalls das Layout der gedruckten Schaltung (mit einiger Handarbeit). Es ergab sich eine Platine in der Grösse eines Pi Hats.



Schema iSwitchPi

Note:

Zukünftige Versionen des iSwitchPi werden einen Schutz gegen falsche Polarität brauchen. Dies wird dieselbe Schaltung sein, wie sie auch auf dem Raspberry selbst vorhanden ist. Siehe diesen Text für Details:

Raspberry voltage protection.

7.6 Gedruckte Schaltung

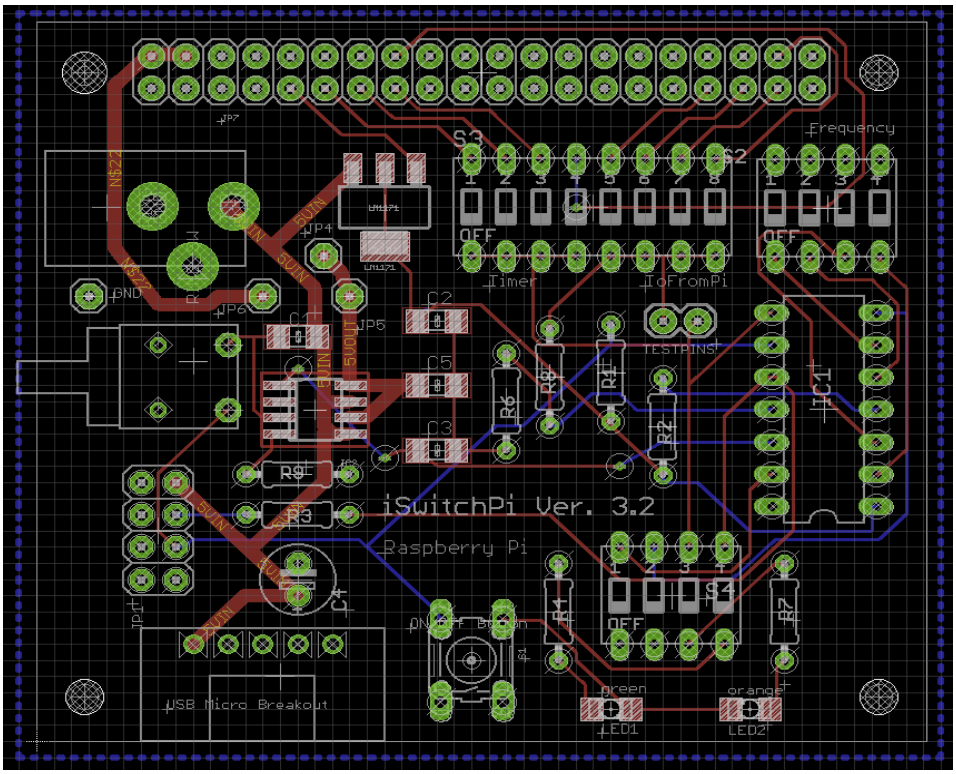
Nachdem das Board mittels des Programms Eagle PCB designed wurde (auf dem Mac) entstand eine Output-Datei `iswitchpihat.brd`. Mit dieser Datei kann eine geeignete Firma eine gedruckte Schaltungsplatine herstellen. Für iSwitchPi wurd die Firma Eurocircuits gewählt. 10 Platinen kosten rund 90 Euro.

Besonderheiten des Boards: Für den Mikrocontroller ATtiny44 ist ein 14-Pol IC-Sockel vorgesehen. Für dessen Neuprogrammierung ist deshalb auf dem Board keine ICSP-Schnittstelle vorhanden: Der Chip wird abgezogen und in den Zero-Force-Sockel eines Programmers gesteckt.

Die Platine wird teilweise mit SMD-Komponenten der Grösse 1206 bestückt - das erfordert einige Erfahrung für das Löten.

Die ersten 10 Prototyp-Platinen waren fehlerfrei und bislang wurden 4 Stück davon bestückt und gelötet (Stand Dezember 2016). Erste Tests sind erfolgreich.

Einige Löcher waren mit zu geringem Durchmesser und es wird eine neue Version der Platine erstellt werden - mit zusätzlichen Komponenten für den Polaritäts-Schutz - siehe weiter oben.



Gedruckte Schaltung iSwitchPi (Version 2)

7.7 Pulsgenerator

Der Pulsgenerator erzeugt kurze Rechteckpulse (Duty Cycle ca. 10%) mit verschiedenen Frequenzen. Durch einen DIP-Schalter kann die produzierte Frequenz eingestellt werden:

- fixe Frequenz wählbar 0.5/1/5/10/50/100/500 Hz

Die Frequenz kann mit den drei rechten Schaltern des DIP-Schalters gewählt werden. Mit 3 Schaltern sind 8 Frequenzen wählbar.

7.8 Kommunikation zwischen Pi und iSwitchPi

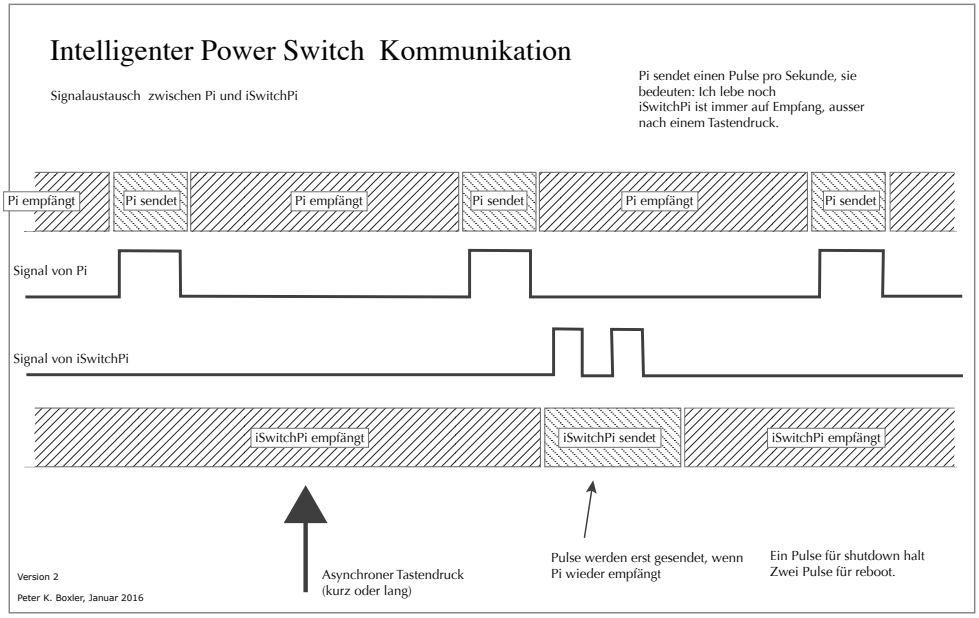
Das im folgenden beschriebene Python Script im Pi sendet prinzipiell jede Sekunde einen Puls auf der Kommunikations-Leitung auf dem definierten GPIO Pin. Diese Pulse signalisieren: ich lebe noch.

Das Programm im ATtiny44 ist prinzipiell immer auf Empfang, um diese Pulse zu bekommen. Kommen keine Pulse mehr, so wird angenommen, dass der Pi nicht mehr läuft oder sich in einem reboot-Prozess befindet. Die Power-Off Sequenz wird dann eingeleitet.

Wird die Drucktaste kurz oder lang gedrückt (shutdown oder reboot), so wird dies dem Script im Pi mitgeteilt durch Senden von Pulsen: ein Puls bedeutet shutdown, zwei Pulse bedeuten reboot.

Diese Pulse werden jedoch erst gesendet, nachdem ein Pulse vom Pi eingetroffen ist - nur dann ist sicher, dass der Pi wieder auf Empfang ist.

Folgendes Schema soll dies verdeutlichen.



Schema der Kommunikation

7.9 Programm im ATtiny44

Das Programm für den Mikrocontroller wird aus 2 C-Programmen kompiliert:

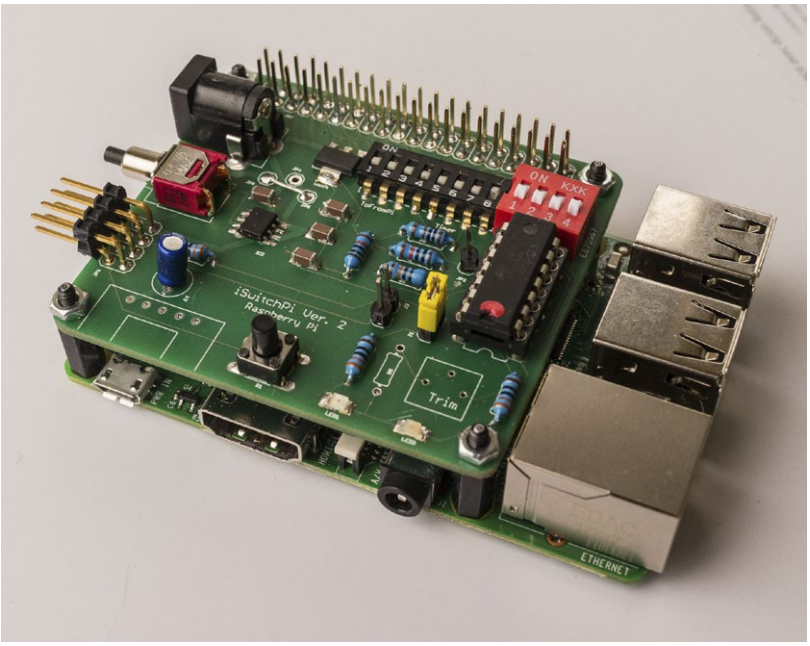
- Implementierung der Endlichen Maschine mit 7 Zuständen (iswitchpi.c)
- Implementierung des Pulsgenerators mit Hilfe von Timer 1 (square.c)

Die Verwendung der I/O-Pins des ATtiny ist im Sourcecode dokumentiert. Das C-Programm implementiert die Endliche Maschine mit 7 Zuständen gemäss der Zustandstabelle. Die Drucktaste für die Auslösung der verschiedenen Funktionen ist entprellt (debounce) mit C-Code von Peter Dannegger. Der Code ist auf Mikrocontroller.net publiziert und diskutiert. Er funktioniert prima, ist aber nicht ganz einfach zu durchschauen. Ich habe zusätzlich eine Funktion key_clear() eingefügt. Damit können die Tastendrücke besser mit den Zuständen synchronisiert werden.

Der ATtiny44 muss mit 1 Mhz laufen, dies ist im Makefile spezifiziert. Die Fuses müssen also auf den Defaultwert gesetzt sein: 62 DF FF.

Zur Programmierung von AVR Mikrocontrollern gibt es ebenfalls in Mikrocontroller.net gute Tutorials.

Der Code des C-Programms ist auf der Projektwebsite verfügbar (siehe Kapitel Links).



Prototyp iSwitchPi aufgesteckt auf Pi

8. Komponente B: Python Shutdown Script

Wie bereits in der Problembeschreibung dargelegt, arbeitet die Elektronik-Komponente mit einem Python Script iswitchpi.py zusammen, welches nach dem Boot im Pi gestartet wird (Eintrag in /etc/rc.local).

Die Externe Komponente und das Script kommunizieren mittels Signalen auf einem einzigen GPIO Pin - siehe oben.

Das Script besteht im wesentlichen aus einen einfachen Loop, in welchem jede Sekunde ein 50 ms Puls am definierten GPIO Pin ausgegeben wird.

Nach dem Start wird dieser Pin jedoch auf Input gesetzt und es wird ein Interrupt-Handler auf rising abgesetzt. Vor dem Senden des Pulses wird dieser Interrupt-Handler entfernt und der Pin auf Output gesetzt. Nach dem Senden des Pulses wird der Pin wieder auf Input gesetzt und der Interrupt-Handler erneut aufgesetzt.

Der Interrupt-Handler zählt die Anzahl der Interrupts (Pulse), die seit dem letzten Senden eingetroffen sind.

Ein einzelner Interrupt führt zu Shutdown Halt und zwei Interrupts führen zu Reboot des Pi.

Siehe das Kommunikations-Diagramm weiter oben.

Das Script wird durch folgenden Eintrag in der Datei /etc/rc.local gestartet (ersetzen des Directories myservices durch eigenes Directory):

```
python /home/pi/myservices/iswitchpi.py [-d N] [-p nn] &
```

Der Commandline Parameter -d spezifiziert Debug Output (defaultwert ohne Angabe ist 1):

- 0: Kein Output (quiet)
- 1: Output bei Start und Empfangen eines Shutdown Commands (HALT oder REBOOT)
- 2: Full Debug Output for Testing.

Der Commandline Parameter -p spezifiziert den vom Script verwendeten GPIO-Pin:

- Parameter -p fehlt: es wird defaultmässig GPIO-Pin 20 verwendet (Pin38 physikalisch).
- Möglich ist - p 13 oder - p 19 oder -p 26 oder - p 20

9. Verwendete GPIO Pins

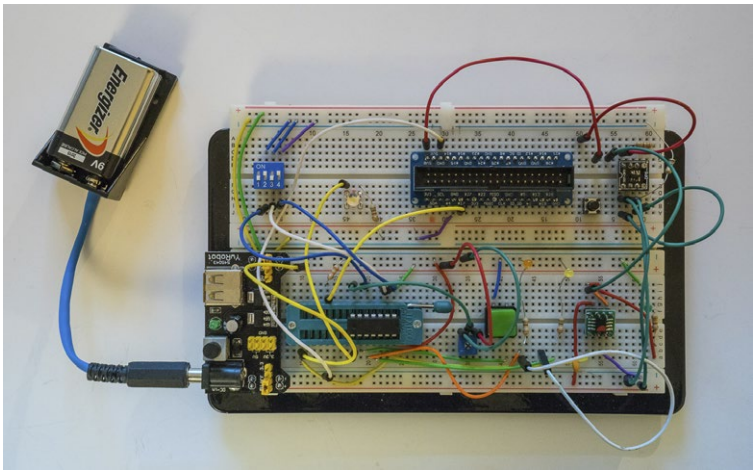
Für die Grundfunktion des iSwitchPi (Power On/Off) wird nur ein GPIO-Pin benötigt. Möchte man die Pulse-Funktion nutzen, ist ein zweiter GPIO-Pin notwendig.

- Kommunikation: zwischen dem iSwitchPi-Board und dem Pythonscript iswitchpy.py im Pi. Die gewünschte Pin-Nummer wird auf dem Board mittels des 4-Pol. Dip-Switches gewählt, zur Verfügung stehen die GPIO-Pins 13, 19, 20 und 26. Passend dazu muss der Pin ebenfalls im Pythonscript definiert werden. Defaultmässig verwendet das Script den Pin GPIO 20 (physikalische Pinnummer 38). Mittels Commandline-Parameter -d kann ein anderer (oben genannter Pin) definiert werden. Siehe oben.
- Korrektes Funktionieren des iSwitchPi-Boards ist nur bei Uebereinstimmung gegeben.
- Falls die Pulse-Funktion des iSwitchPi-Boards genutzt wird, ist ein weiterer GPIO-Pin notwendig. Die gewünschte Pin-Nummer wird auf dem Board mittels des 4-Pol. Dip-Switches gewählt, zur Verfügung stehen die GPIO-Pins 17, 22, 23 und 27. Passend dazu muss derselbe Pin ebenfalls in einem selbst zu erstellenden Pythonscript definiert werden. Es kann dazu der Pin-Change Interrrupt verwendet werden.

10. Installation von iSwitchPi

Der **Quick Start Guide to iSwitchPi** (English und Deutsch) beschreibt die Instaaltion im Detail.

11. Testaufbau



Testaufbau iSwitchPi auf Breadboard (ohne Pi)

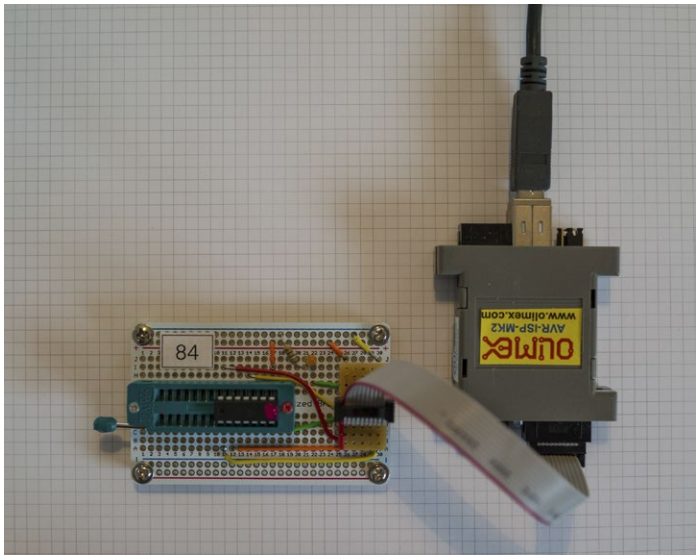
12. Programmierung des ATtiny44

Die Entwicklung des Codes für iSwitchPi wurde auf einem Mac durchgeführt. Auf dem Mac wurde das CrossPack for AVR Development installiert. Dieses Paket stellt folgende Komponenten zur Verfügung:

- den Compiler avr-gcc mit den notwendigen C-Libraries
- Den Commandline-Uploader AVRDUDE

Für das Schreiben des Codes in den Speicher des Mikrocontrollers (flashen oder uploaden) wurde ein einfaches Board mit einem 28-Pol Zero-Force-Sockel gebaut, siehe Bild. Dieses Board wird mittels einem 10-Pol Kabel an den Programmer Olimex AVR-ISP-MK2 angeschlossen. Dieser wiederum ist mit einem USB-Kabel mit dem Mac verbunden.

Für das Flashen wird der Uploader AVRDUDE benutzt - er wird im Makefile durch ‚make flash' nach der Compilation aufgerufen.



Programmer für ATtiny24/44/84

Bemerkung zu CrossPack Version:

Auf Mac OSX 10.9 (und höher) muss die AVRDUDE Version 5.11.1 verwendet werden, da Version 6.01 bei Verwendung eines Olimex ISPMK2 Programmers einen bug hat. Siehe hier

http://www.avrfreaks.net/forum/olimex-isp-mk2-error-mac-osx-109

http://www.avrfreaks.net/comment/1011406#comment-1011406

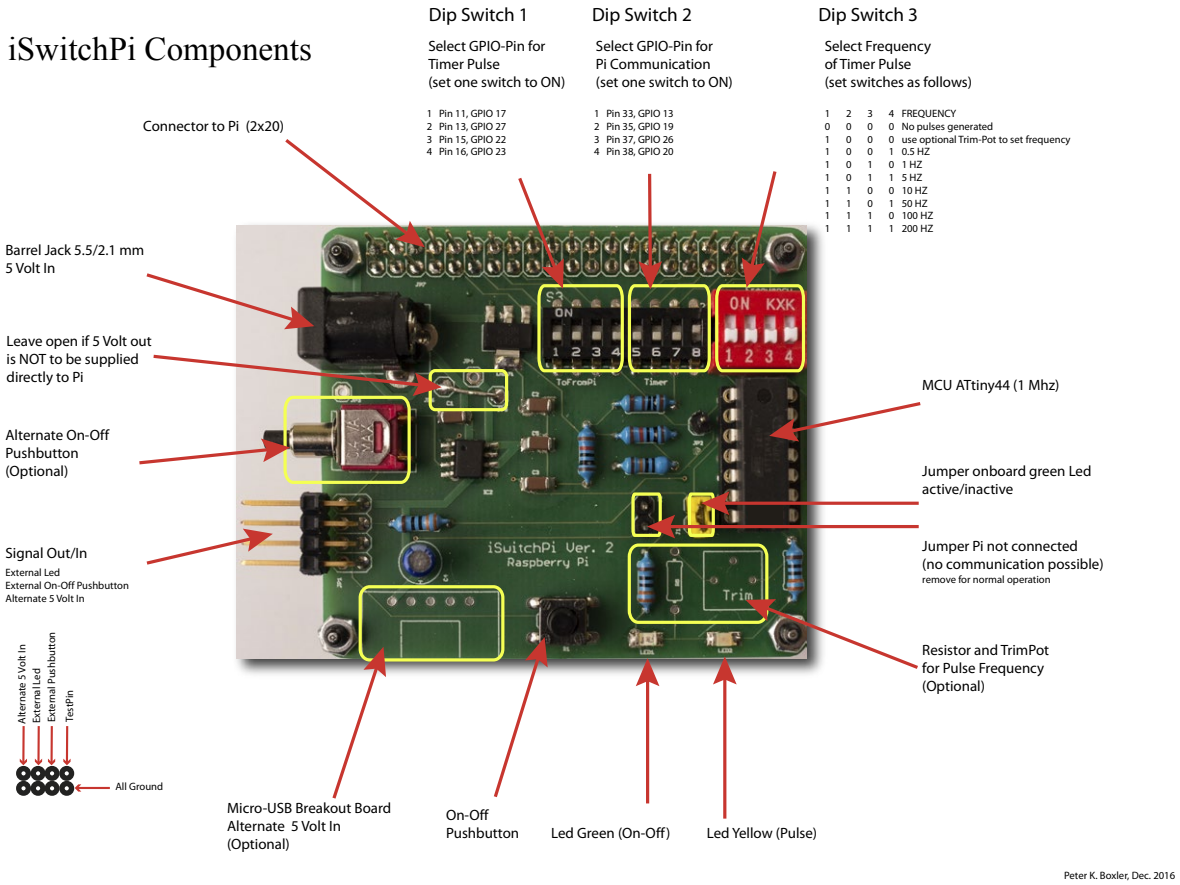
Ich habe deshalb CrossPack-AVR-20121203 installiert.

12.1 Flashing the Attiny44

Entweder mit make flash oder - falls die Datei iswitchpi.hex bereits vorhanden - ist mit Utility avrdude. Das Kommando zum Schreiben des Programms in den Mikrokontroller ist abhängig von der AVR-Werkzeugkette. In meinem Fall mit Utility avrdude und Programmiergerät avrisp2 lautet das Kommando:

```
avrdude -c avrisp2 -p attiny44 -P usb -U flash:w:iswitchpi.hex
```

13. Komponenten auf dem Board



Komponenten auf den iSwitchPi Board

14. Zusammenfassung

Der Versuch, die Problembeschreibung mittels endloser Automaten (Finite State Machines) zu beschreiben, hat zwar Einiges an Aufwand gekostet, hat aber die Lösungsfindung beträchtlich klarer gemacht. Das Schreiben des C-Programms für den iSwitchPi war dadurch relativ einfach; durch die Zustandsdiagramme und die Zustandstabellen war der Code klar vorgegeben. Beim Testen sind keine ernsthaften logischen Probleme des Programm-Ablaufs aufgetreten.

Was den schon immer bekannten Satz erneut bestätigt: zuerst denken, dann codieren.

15. Links

Meine Raspberry Projekte

Projekt-Website

Source Code iSwitchPi incl. Eagle Dateien

GitHub

Andere Links

Initial State Raspberry Pi GPIO Pins
Raspberry Foundation

Zum Thema Endliche Maschine
http://www.mikrocontroller.net/articles/Statemachine

Tutorials zu AVR Mikrocontroller
http://www.mikrocontroller.net/articles/AVR-Tutorial

Eagle PCB Design Software
http://www.cadsoft.de

Beste Firma für Ideen und Anleitungen
https://www.adafruit.com

In der CH zu empfehlen, führt Adafruit Produkte
http://www.play-zone.ch

16. Anhang

16.1 Teile Liste iSwitchPi

Part	Value	Package	Note	Supplier	PartNumber
Condensator					
C1	1uF	SMD1206	ceramic	Any	
C2	1uF	SMD1206	ceramic	Any	
C3	1uF	SMD1206	ceramic	Any	
C4	1uF	E2,5-6E		Any	
C5	1uF	SMD1206	ceramic	Any	
IC					
IC1	MCU ATTINY44	DIL14	Attiny		
IC2	MIC2025-1YM	SO08	High-Side Switch		
LM117	LM1171MP-3.3	SOT223	3.3 Volt Regulator	Distrelec	110-38-693
Led					
LED1	green	SMD 1206	SMD 1206		
LED2	orange	SMD 1206	SMD 1206		
Resistor					
R1	100k	0204/7		Any	
R2	10k	0204/7		Any	
R3	320	0204/7		Any	

R4	320	0204/7		Any	
R6	10k	0204/7		Any	
R7	320	0204/7		Any	
R8	1k	0204/7		Any	
R9	80				Nur bei MIC2445A
DIP-Switch					
S2		DIP04S	Dip 4-Pol	Any	
S3		DIP08S	Dip 8-Pol	Any	
S4		DIP04S	Dip 4-Pol	Any	
Pin Header					
J1		1X02	Onboard green Led On/Off		
J2		1X02	without Pi		
JP1		2X04	Input/Output		
JP3		1X01	Optional 5 V IN		
TestPins		1X02	TestPins		
JP5		1X01	5 V to Pi, connect to JP6		
JP6		1X01	5 V to Pi, connect to JP5		
JP7	extra long pins	2X20-BIG	GPIO Stacking Header	Adafruit	2223
Pushbutton					
S1	6x6 mm	B3F-10XX	Pushbutton On/Off	Adafruit	367/1490/ P00000256 P00000681
Other					
IC-Socket 14-pol			for ATtiny44	Any	
5 Volt IN					
PB3	2.1MMJACK		Barrel Jack 5.5/2.1 mm	Adafruit	373
PB4	ADAUSBMICRO		USB-Micro-Breakout	Adafruit, Play-Zone	1833/P00001025
Optional, for variable Frequency Pulsgenerator					
R5	320	0204/7		Any	
RTRIM	50k		Trim Pot	Conrad	447363
Optional, second Pushbutton On/Off					
PB1	Button liegend		Pushbutton On/Off	Distrelec	135-75-923
Montage					
	Standoff	4 Pieces	11mm/2.5mm	Adafruit	2336

16.2 Notizen

In der ersten Version des iSwitchPi wurde ein zusätzlicher Chip NCP702SN33 vorgesehen, der zusätlich 3.3 Volt für ein Breadbord liefern soll. Dies wurde in der Version 2 gestrichen.

Notes zu GPIO

Nicht alle GPIO Pins des Pi sind identisch. In Internet Foren findet man viele Diskussionen zur Frage des Initialzustandes (High/Low und Impedanz) der GPIO Pins nach dem Power-On des Pi. Für sehr viele (aber nicht alle) Anwendungen ist es wichtig, genau darüber Bescheid zu wissen, damit verbundene Komponenten nicht im falschen Moment schalten/aktiviert werden. Das Problem scheint aber nicht eine einfache Lösung zu haben.

Die Raspberry Foundation bietet seit 2014 eine Lösung für das Problem, siehe hier.

Siehe auch diesen Blog. Oder Google mit ‚raspberry gpio initial state'.

Zusammenfassend wird gesagt, dass man für Pins, die als Input Pins verwendet werden, in jedem Fall einen Widerstand zur Strombegrenzung vorschalten sollte.

Output Pins sollten immer mittels einem 15k Widerstand nach Masse geschaltet werden, damit sie sicher Low sind, auch wenn sie Pi-intern floating sind während dem Boot.

16.3 Code iSwitchPi

Hier ein Ausschnit aus der C-Implementierung der Finite State Machine im Programm iswitchpi.c für den ATtiny44.

```
//
// ---- Main Loop. forever ---------------------
// ---- Implements State Machine ---------------
  for(;;)
  {
 //   _delay_ms(10);                                  // for testing

    switch (state) {
/*-------------------------------------------------------------------*/
/*  state 1  Stand-by State, all is off, waiting for short keypress  */
/*  Power to Pi ist off, led blinks short pulses                                 */
/*  Waiting for short keypress                                       */
/*-------------------------------------------------------------------*/
    case state1:

        if (first_time & (1<<0))     // first_time time throu ?
            {
            PORTA &= ~( 1<<LED1 | 1<<VPOWER);   // all outputs off
            key_clear( 1<<KEY0 );
            blinkwhat=PULSED_Blink;
            tick2=0;
            blinkon=1;
            pwm_stop();                                                  // stop pulse genaration
output on PA5
            pastpulses=0;                       // pulse counter reset (pulses from Pi)
            first_time =0xff;                       // set first_time all other states
            first_time &= ~(1<<0);                  // clear first_time this state
            }
        if  (get_key_short( 1<<KEY0 )) {         // get debounced keypress
            if (!(PINA & (1<<TESTPIN)) )         // if Testpin is low: signalling TESTMODE
                state=state7;                       // next state is state 7
            else
                state=state2;                       // next state is state 2
            }                                   // leaving standby
        break;

/*-------------------------------------------------------------------*/
/*  state 2  Tentative Power on, waiting for Pi to come up           */
/*  Power to Pi ist on, led is blinking fast                         */
/*  Short keypress switches to state 4   (power on without checking */
/*  whether Pi is on)                                                */
/*-------------------------------------------------------------------*/
    case state2:

        if (first_time & (1<<1))   {            // first_time time throu ?
```

```
            PORTA |= (1<<VPOWER);               //switch 5 volt power on
            blinkwhat=REGULAR_Blink;
            blinkint=POWERON_Blink_int;
            tick2=0;                            //start timer
            sekunde=0;
            pwm_start();                                                    // start pulse generati-
on output on PA5
            first_time =0xff;                  // set first_time all other states
            first_time &= ~( 1<<1)  ;          // clear first_time this state
            }

        pwm_check();

        if ((blinkwhat>0) && (sekunde > POWERON_Delay_long))  {
            blinkwhat=0;
            state=state1;                       // Pi did not come on, so gaback to stand by
            }

        if  (get_key_short( 1<<KEY0 ))  {        // get debounced keypress short
            blinkwhat=0;
            state=state4;
           }
                    // how many pi pulses have we received ? if we have Pi is alive
                    // so we go to state 3 (normal operation state)
            if (pastpulses > 3) {state=state3;}

    break;

/*-------------------------------------------------------------------*/
/*  state 3  Power ON Number 1, regular operating state             */
/*  Power to Pi ist on, led is on                                    */
/*  Loss of signal from Pi changes state to 5                        */
/*  Short keypress  signals Pi to shut down, changes state to 5      */
/*  Long Keypress signals Pi to reboot, stays in state 3             */
/*-------------------------------------------------------------------*/
    case state3:
        if (first_time & (1<<3))  {                  // first_time time throu ?
            PORTA |= (1<<LED1);                // led full on
            blinkwhat=0;
            key_clear( 1<<KEY0 );
            first_time =0xff;                       // set first_time all other states
            first_time &= ~( 1<<3)  ;               // clear first_time this state
            }
            pwm_check();                                            // check various inputs for frequen-
cy of pulse on PA5

        if  (get_key_short( 1<<KEY0 )) {         // get debounced keypress short
            cli();
            sendnow=1;                         // set flag so IR handler can send signal
            sei();                              // Interrupt enable
            state=state5;                         // next state 5
            poweroff_delay=POWEROFF_Delay_HALT_long;  // Poweroff delay for halt
            }

        if( get_key_long( 1<<KEY0 ))   {         // get debounced keypress long
            cli();
            sendnow=2;                         // set flag so IR handler can send signal
            sei();                                // Interrupt enable
            state=state5;                         // next state 5
                                                   // Pi will reboot
            poweroff_delay=POWEROFF_Delay_REBOOT_long;  // Poweroff delay for halt

            }

            // check numer of pulses from Pi, zero means: Pi is not alive
             if (pastpulses < 2 )  {
                state=state5;                          // ok, start power off sequence
                poweroff_delay=POWEROFF_Delay_REBOOT_long;   // Poweroff delay for halt
                }
        break;

/*-------------------------------------------------------------------*/
/*  state 4  Power ON Number 2, special operating state             */
/*  we do not care about pulses from Pi                              */
/*  Power to Pi ist on, led is on                                    */
/*  Short keypress  signals Pi to shut down, changes state to 5      */
/*-------------------------------------------------------------------*/
    case state4:
        if (first_time & (1<<4))   {                 // first_time time throu ?
            PORTA |= (1<<LED1);                // led full on
            blinkwhat=0;
            key_clear( 1<<KEY0 );               // ignore keypresses that might have come
```

```
            first_time =0xff;                       // set first_time all other states
            first_time &= ~( 1<<4)  ;               // clear first_time this state
            }
            pwm_check();                        //  Pulse generation
                                                //  Check DIP-Switch (PINB0 to PINB2) and Analog Input
on PINA6

        if  (get_key_short( 1<<KEY0 ))  {        // get debounced keypress short
            cli();
            sendtopi(1);                        // set flag so IR handler can send signal
            sei();                               // Interrupt enable
            state=state5;
            _delay_ms(10);
            poweroff_delay=POWEROFF_Delay_REBOOT_long;  // Poweroff delay for halt

            }
        break;

/*-------------------------------------------------------------------*/
/*  state 5  Activate Power off, prepare to shut off                 */
/*  irrelevant of signals from Pi                                    */
/*  Power to Pi ist still on, led is blinkng slow                    */
/*  Short keypress  changes to state 4 (keep power on regardless     */
/*  of signal from Pi)                                               */
/*  Long keypress switches off immediately (goto state 1)            */
/*  goto stand by state if timer runs out                            */
/*-------------------------------------------------------------------*/
    case state5:
        if (first_time & (1<<5))   {                 // first_time time throu ?
            blinkwhat=REGULAR_Blink;            // set led to blink
            blinkint=POWEROFF_Blink_int;
            key_clear( 1<<KEY0 );               // ignore keypresses that might have come
            pastpulses=0;
            tick2=0;                            //start timer
            sekunde=0;
            first_time =0xff;                       // set first_time all other states
            first_time &= ~( 1<<5)  ;               // clear first_time this state
            }
            pwm_check();                        //  Pulse generation
                                                //  Check DIP-Switch (PINB0 to PINB2) and Analog Input
on PINA6

       if  (get_key_short( 1<<KEY0 )) {         // get debounced keypress short
            state=state3;
            }
        if( get_key_long( 1<<KEY0 ))  {          // get debounced keypress long
            state=state1;                          // next state 5
                                                   // Pi will reboot
            }

/*   removed this - not in state diagram
            // check Signal from Pi, how many pulses have we received
        if (pastpulses > 4)   {                    // Pi seems to ba alive, ok keep power on
            state=state3;
            }
*/
            // wait for POWEROFF_Delay sec before switching off 5 Volt supply
        if ((blinkwhat>0) && (sekunde > poweroff_delay))   {
            blinkwhat=0;
            state=state6;                       // next state is state 5
            }

        break;
/*-------------------------------------------------------------------*/
/*  state 6  Last Chance                                             */
/*  we are about to switch 5 Volt power off                          */
/*  but before we do that we check the signal from Pi again :        */
/*  If further pulses came in (Pi rebooted) we keep power on         */
/*  If no more pulses came in we go to state 1                       */
/*-------------------------------------------------------------------*/
    case state6:
        first_time =0xff;                       // set first_time all other states
        key_clear( 1<<KEY0 );

            // check Signal from Pi
        if (pastpulses > 3)  {
            state=state3;
            }
        else state=state1;

      break;

/*-------------------------------------------------------------------*/
```

```
/*  state 7  TESTMODE only                                              */
/*  Power to Pi is switched on, green led is on                         */
/*  Orange Led blinks every 3 seconds IF and only if pulses from Pi     */
/*  are ok received.                                                    */
/*  Short keypress: next state is state1 (off)                          */
/*--------------------------------------------------------------------*/
    case state7:
        if (first_time & (1<<7))   {            // first_time time throu ?
            PORTA |= (1<<VPOWER);               //switch 5 volt power on
            PORTA |= (1<<LED1);                 // led full on
            blinkwhat=0;
            key_clear( 1<<KEY0 );
            tick2=0;                            //start timer
            sekunde=0;
            blink_led();
            first_time =0xff;                       // set first_time all other states
            first_time &= ~( 1<<7)  ;               // clear first_time this state
            }
       if  (get_key_short( 1<<KEY0 )) {         // get debounced keypress short
            state=state1;
            }

            // check Signal from Pi blink orange led when 2 pulses have been received
            if (pastpulses > 2)  {
                blink_led();
                pastpulses=0;
             }
    }

//----  End of Switch Statement -----------------------------
```

Ende des Dokumentes.

Peter K. Boxler, Februar 2016