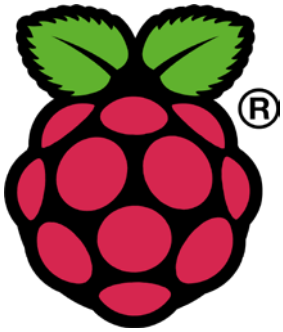
Project iSwitchPi

Intelligent Power Switch für Raspberry Pi

Make working with the Pi easy.



Inhaltsverzeichnis

1. 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.

This project description presents an elegant solution for the problem of switching the Pi on and off. The goal was to have a small pc board whose functionality can be described as follows:

- one pushbutton only for power on and power off
- usable on a breadboard
- only one communication line to the pi - in other words. only one GIPO pin used on the Pi
- awarness of the Pi's status (running or not running)
- only one communication line to and from the pi - in other words. only one GIPO pin used on the Pi
- square signal output capability with variable/defined frequency to be used as a timer interrupt for the Pi
- later versions with a Pi Hat form- actor.

2. First Ideas

2.1 Market Review

There are a number of electronic add-ons to be found on the market - they all try (and succeed) to solve the problem of power up/down the Pi. All of them consist of two components:

- a small pcb board with connections for pushbutton, 5 Volt in, 5 Volt out, GPIO to Pi
- a small shell script od Python script running in the Pi

I found the following produts and tried some of them in my own lab. None of the has an square wave generator.

- Pi-Supply from the English Company Pi-Supply. This add-on uses two pushbuttons for powern on and power off, respectively. it uses two GPIO pins on the Pi. Works well.
- Pi-Shutdown button as decribed on Instructables. this solution uses an embedded controller ATtiny13/85.
- ATX-Raspi from LowPowerLab. This product is close to what i had in mind.
- ykrud from Yepkit. This product also uses an embedded controller. Seems very ok but uses 2 GPIO pins.

2.2 Findings

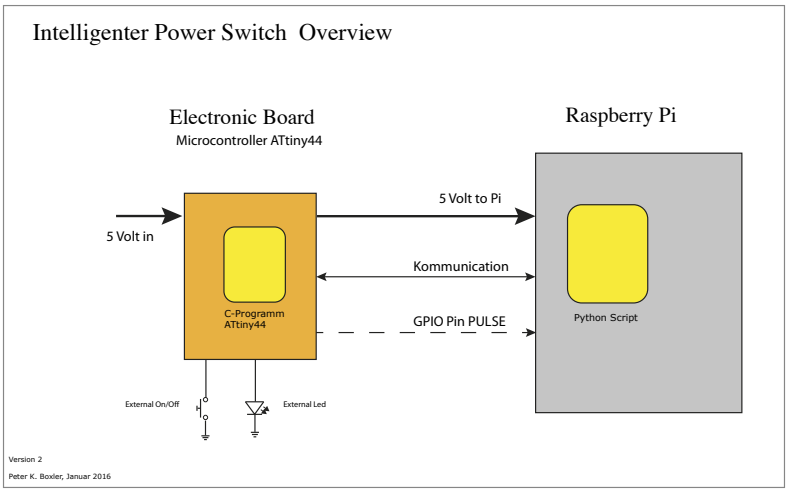
That review showed that I needed to go back to the drawing board to start finding my own solution which is based

on ideas that were used in some of the products. I decided to use an ATtiny44 microcontroller.

- There will be an electronic component (pcb board) with all the neccesary componentes and the MCU. Two different board will be made: a small board to be used on a breadboard and another (almost) hat-compliant board to be stacked on a Pi. Only ONE GPIO-Pin will be used for communication.
- There will also be a Python script that is started in the Pi at boot time.

3. Overwiew

The add-on will consist of pcb in the form of a Pi hat. At the heart will work a microcontroller ATtiny44 for which I developd a C program that implements the necessary logic. This is called the iSwitchPi board.
This is a overview:



Übersicht der Lösung

4. Problem Statement

First step is always an thorough analys of all the requirements

I tried the following description in plain sentences:

There is a small pushbutton to switch the Pi on and off. Is the Pi switched off a short press on the pushbutton switches on the 5 Volt Power for the Pi and the led blinks fast. If the Pi does not come on (‚running') power is agin cut after a wait time of about 80 sec.

If the Pi comes up we have normal operation, 5 volt power stays on and the led is fully on.

A short press on the pushbuttons signals to the Pi to shutdown and activates a timer. The led blinks slow.Power is cut after the timer reaches a predefined value.

A long press on the pushbutton signals the Pi to reboot - power stays on in this case.

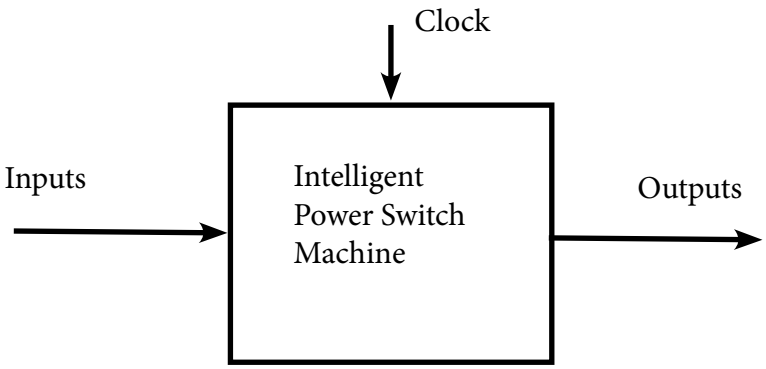
additionally: if the system waits for the Pi to come up a another short press on the button directs the system on keep power on without checking whether Pi is running.

If the Pi-Pin suddenly goes to low (Pi not running), the System starts toe power down sequence.

That covers situation where the Pi Pi is shutdown in a Terminal window.

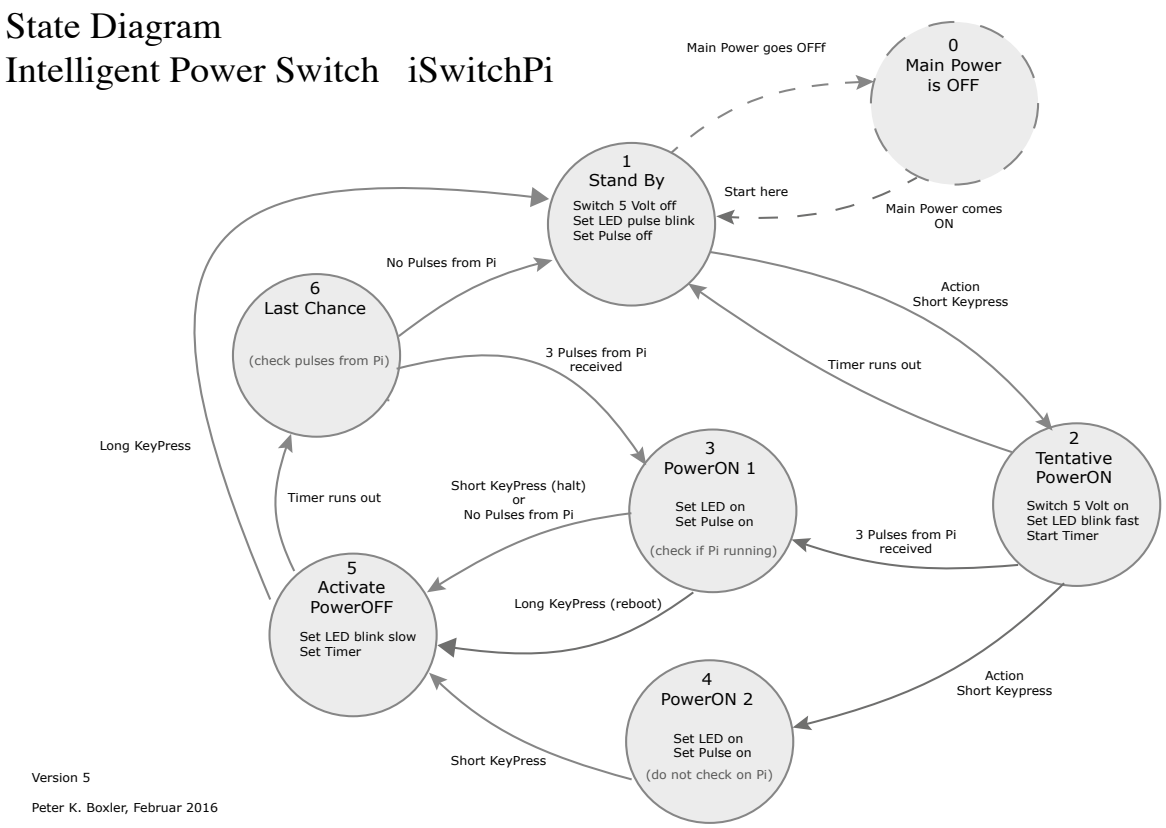
Such a prosa description ist not very good and usually not complet. Sentences are misleading and there might be ambiguity.

That is where come finite state machines to the rescue. A state diagram togehter with a state table describes the machine completely and the transfer to C code is easy. A finite state machine has defined inputs, states,transistions and also defined outputs.



4.2.1 State-Diagramm

After several tries the following state diagram was found:



State Diagram iSwitchPi

4.1 Input and Output of the machine

These tables define all inputs and outputs (each input can be one or zero)

Iiput	Shortform	Source
Short Keypress	KT	Pushbutton
Long Keypress	LT	Pushbutton (>1.5 sec)
TimerSignal	TS	Timer reaches defined count
SignalFromPi	SPi	Pulses from Pi received

Output	Shortform	What is this
5 Volt for Pi	5V	5 Volt power to PI on or off
Green Led on	Ledon/Ledoff	Led is on or off
Led blink slow	Ledsl	Led blinks slow
Led blink fast	Ledfst	Led blinks fast
Signal to Pi for Reboot	StPireb	Send one single pulse to Pi
Signal to Pi for Shutdown	StPishut	Send 2 pulses to Pi
Start Timer	STo or STf	On /Off (means start/stop Timer)
Pulsgenerator	ON/OFF	Puls Generation ON/OFF

4.2 State Diagram

The state diagram describes the workings of finite state machine completely. The transfer to a c-program is simple. A certain state combined with a combination of input signals leads to a new state as shown in state diagram.

State	Inputs				Next State
	SK	LK	TS	SPi	
1 Stand by	1	x	x	x	2 Tentative Power On
2 Tentative Power On	x	x	0	1	3 Power On-1
2 Tentative Power On	x	x	1	0	1 Stand by
2 Tentative Power On	1	x	x	x	4 Power On-2
3 Power On-1	0	0	x	0	5 Activate Power Off
3 Power On-1	1	0	x	0	5 Activate Power Off
3 Power On-1	0	1	x	0	5 Activate Power Off
4 Power On-2	1	x	x	x	5 Activate Power Off
5 Activate Power Off	x	x	1	0	1 Stand by
5 Activate Power Off	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

4.3 Activities upon entering a state

Every state of the machine usually has some entry-actions associated with this state. These actions are executed once upon entering the state and they usually result in changes in the outputs.

State	Activity on Input
1 Stand by	5 Volt Power off Set Led to Pulse blink Timer stop Pulsgenerator stop

2 Tentative Power On	5 Volt Power On Led blink fast Timer start Pulsgenerator start
3 Power on-1 (normal operation)	Led full on Timer stoppen
4 Power on-2 (spezieller Betrieb)	Led full on Timer stoppen
5 Activate Power Off	Led blink slow Timer start
6 Last Chance	nothing

5. Component A: Electronics

The iSwitchPi implements the on / off function of the Pi power supply in software and hardware. The state diagram and the state tables define the c-program to be created for the ATtiny44. The Pi itself will send regular pulses by means of a script (see below), which signal: I am still alive, do not turn off the power.

5.1 Schema

The hardware component of the iSwitchPi is designed to be in the form of a raspberry hat. Usage is simple: stack on top of the Pi, connect a 5 volt power supply and you are ready to run. Additional hats can be stacked on top of the iSwitchPI.

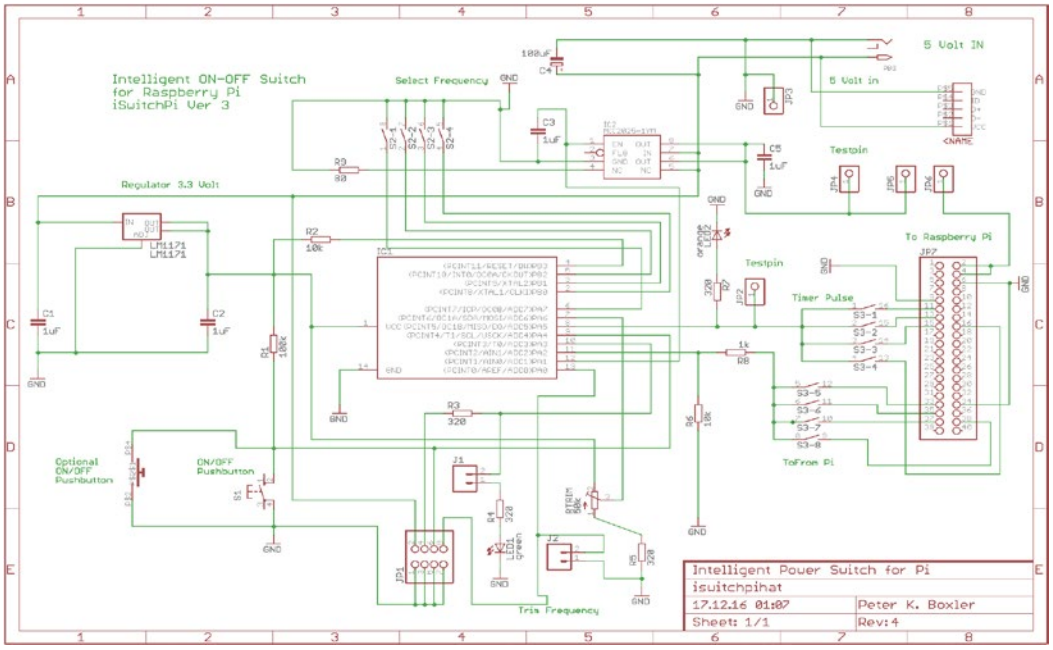
The electrical scheme of the iSwitchPi looks like this. The input voltage of 5 V is reduced to 3.3 V for the microcontroller using a LM1171MP-3.3 module. Thus, the communication line for the Pi is also addressed with this level (GPIO pins for the Pi are rated 3.3 v max).

The microcontroller switches the high-side switch MCP2505 on and off - this switch can switch a maximum of 2 amps - this is the 5 Volt power supply for the Pi.

The Python script running in the Pi (iswitchpi.py) signals ‚I am alive' with pulses on a selected GPIO-Pin. The script can, however, also receive signals from the iSwitchPi on the same GPIO-Pin: the number of incoming pulses determine the type of the command which is sent to the OS: shutdown halt or reboot.

The heart of iSwitchPi is an ATtiny44 microcontroller.

C program (firmware) for the microcontroller, see later.



Schema iSwitchPi

5.2 Puls Generator

In addition to the on-off switching functionality, the iSwitchPi circuit also supplies a variable frequency square wave (Duty Cycle ca. 10%). This can be used to interrrupt the Pi for other purposes.

- frequency adjustable with trim-potentiometer
- fixed frequencies selectbal with dip-switch 0.5/1/5/10/50/100/500 Hz

The GPIO-Pin is also selectable with a 4-pos dip-switch. GPIO-Pins 17,22, 23 or 27 are available.

5.3 Communication between Pi and Electronic

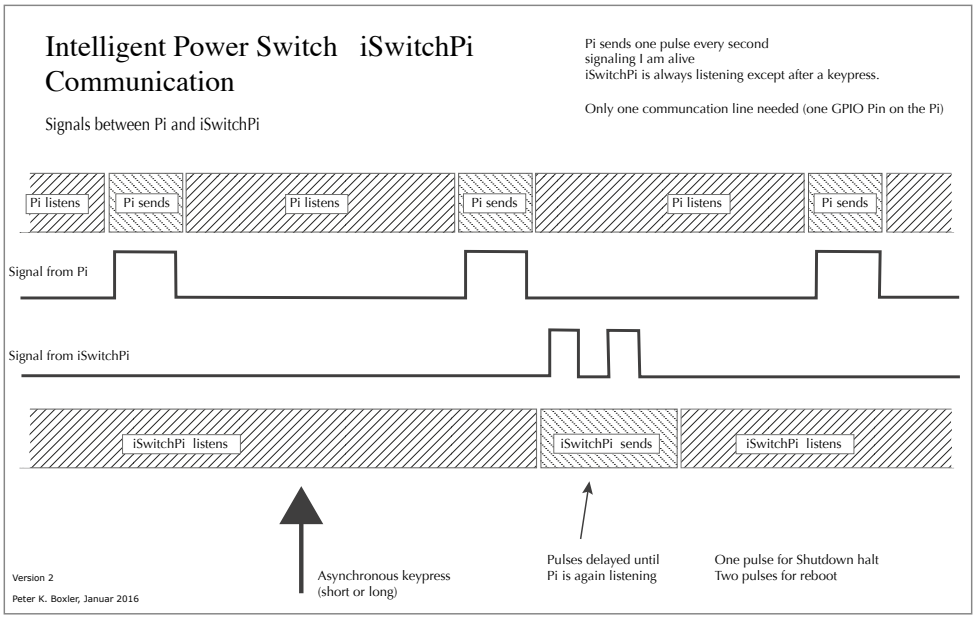
The Python script described below basically creates a 50ms pulse on the defined GPIO pin every second. These pulses signal to the iSwitchPi: I am still alive. That GPIO-PIn is routed to an input pin on the ATtiny44.

The program in ATtiny44 is always listening on that input, in order to receive these pulses. If no pulses are received, it is assumed that the Pi is no longer running or is in a reboot process. The power-off sequence is then initiated.

If the pushbutton is pressed (short or long for shutdown or reboot, respectively), the Attiny sends pulses to the Pi (on the same line): one pulse means shutdown, two pulses are reboot.

These pulses, however, are sent only after a pulse from the Pi has arrived - only at this point in time it is known that the Pi is listening.

The following scheme will illustrate the communication between Python script and c-program in the ATtiny44..



Comunication

5.4 C-Program for ATtiny44

The executable for the Attiny44 is compiled/linked from two C programs:

- Implementation of the Finite State Machine with 6 states (iswitchpi.c)
- Implementation of the puls generator using Timer 1 (square.c)

The use of the I/O pins of the ATtiny is documented in the source code. The C program implements the finite machine with 6 states according to the state table. The push button for triggering the various functions is debounced with C code by Peter Dannegger. His code is published and discussed on Mikrocontroller.net. It works great, but is not easy to understand. I added an additional key_clear () function. This makes it easier to synchronize the push-button with the states. The ATtiny44 needs to run at 1 Mhz, the Makefile specifies this clock frequency. So make sure the fuses are set to 62 DF FF (default values).

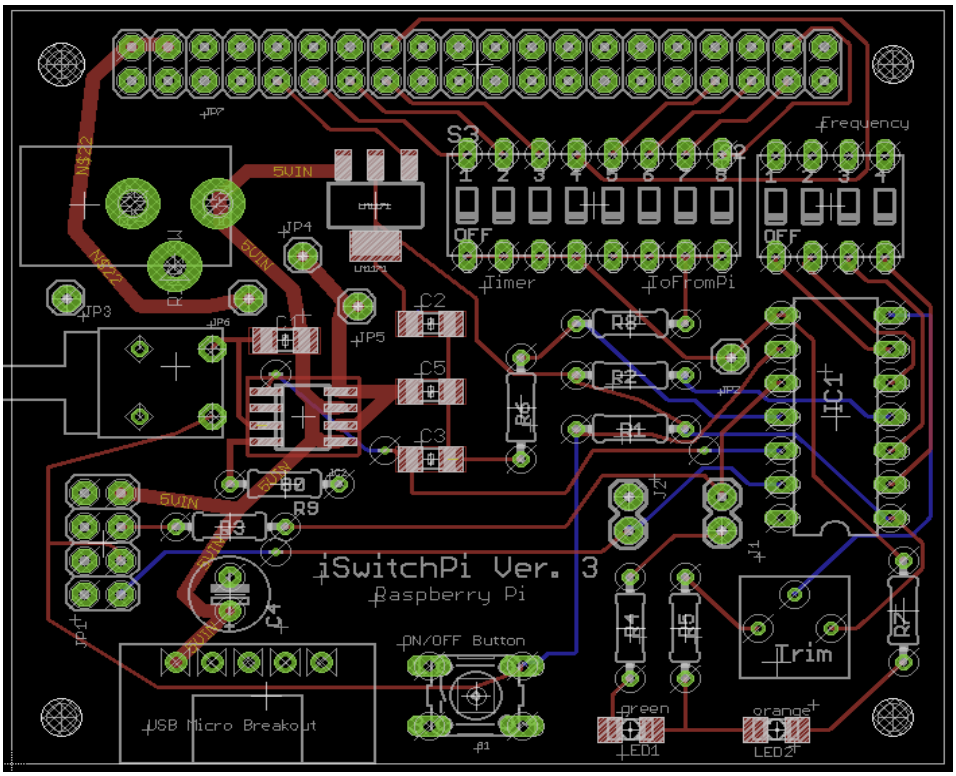
There are also good tutorials for programming AVR microcontrollers in Mikrocontroller.net.

The code of the C program is available on GitHub

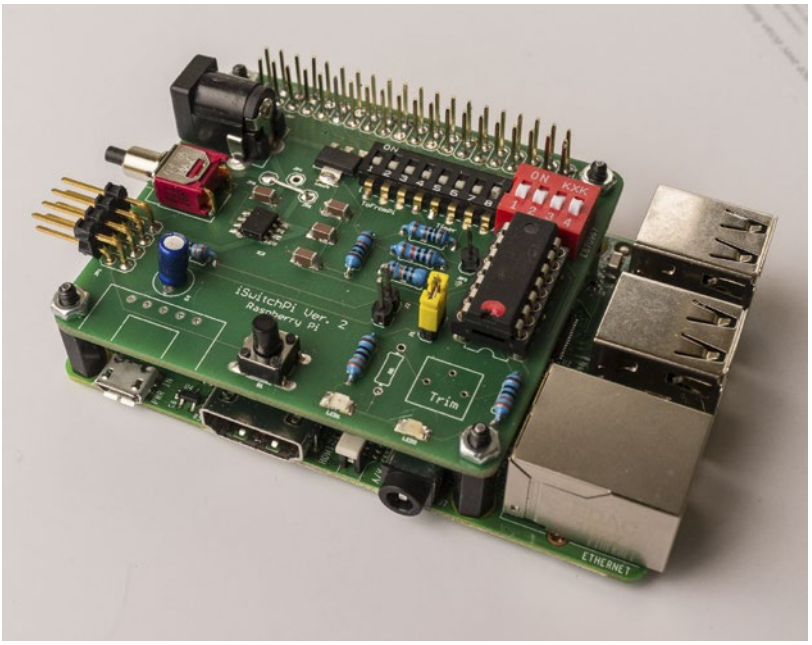
Flashing the executable to the ATtiny44 see further down.

5.5 Printed Circuit Board

The Attiny sits in a 14-pol socket. There is no ICSP-Connection on the board and the MCU is plugged from the socket if reprogramming is needed.



PCB iSwitchPi



iSwitchPi stacked on a Pi

6. Component B: Python Shutdown Script

The electronics communicates with a small Python script that is running in the Pi. That script is started at boot time (Entry in /etc/rc.local).

The clever thing here is that only on GPIO-Pin is used for this 2-way communication. See comm-diagramm for

details.

The Python script is started at boot time with the follwing line added to /etc/rc.local. (replace folder myservices with your own folder within pi home:

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

Two commandline parms are accepted: -d for debugoptions and - p for selection of GPIO Pin used. Both are optional.

- 0: No debug uutput (quiet)
- 1: simple debug out, is default value
- 2: Full debug output for testing.

Commandline parm -p specifies the GPIO pin to be used for communication with the iswitchPi board. This must correspond to the Pin selected with the dip-switch on the board!

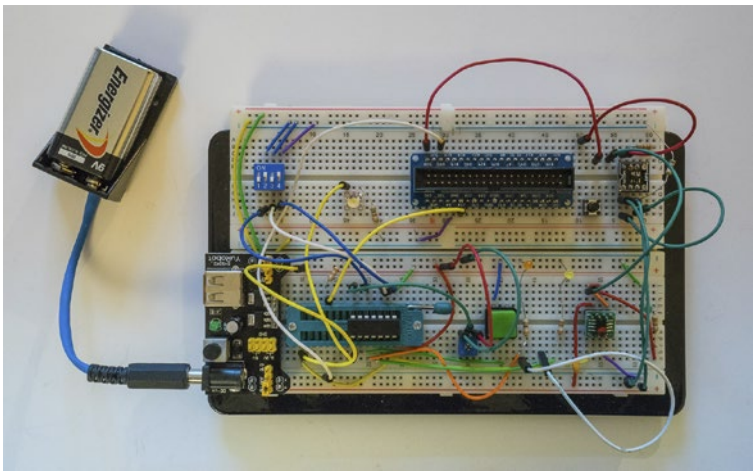
- no parm -p: by default GPIO-Pin 20 is used.
- Possible values (pins) - p 13 or - p 19 or -p 26 or- p 20

7. GPIO Pins used

Only one GPIO pin is required for the basic function of the iSwitchPi (Power On / Off). If you want to use the Pulse function, a second GPIO pin is necessary.

- Communication between the iSwitchPi board and the pythonscript iswitchpy.py in the Pi. The desired pin number is selected on the board using the 4-pol. dip switch, GPIO pins 13, 19, 20 and 26 are available. The pin must also be defined in the Pythonscript. By default, the script uses the pin GPIO 20 (physical pin number 38). A different (above-mentioned pin) can be defined by means of the command line parameter -d. See above.
- If the pulse function of the iSwitchPi board is used, a further GPIO pin is necessary. The desired pin number is selected on the board by means of the 4-pol. pip switch. GPIO pins 17, 22, 23 and 27 are available. Use the same pin in your script that uses the interrupt.

8. Test Setup



Testaufbau iSwitchPi auf Breadboard (ohne Pi)

9. Programming the ATtiny44

I developed the code for iSwitchPi on a Mac. The toolchain used is

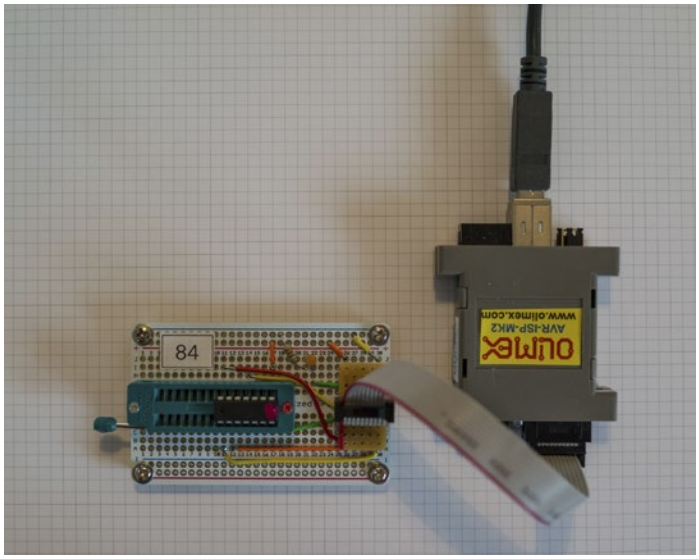
CrossPack for AVR Development .

This package installs the following components:

- tThe compiler avr-gcc with all the neccesary libraries
- The Commandline-Uploader AVRDUDE

I built a small board with a zero-force socket to flash the MCU. This board connects to the programmer Olimex AVR-ISP-MK2 with itself is connected to the Mac with a USB cable.

Flashing is done with uploader AVRDUDE - the makefile does this with the command make flash.



Programmer für ATtiny44

Note zu CrossPack Version:

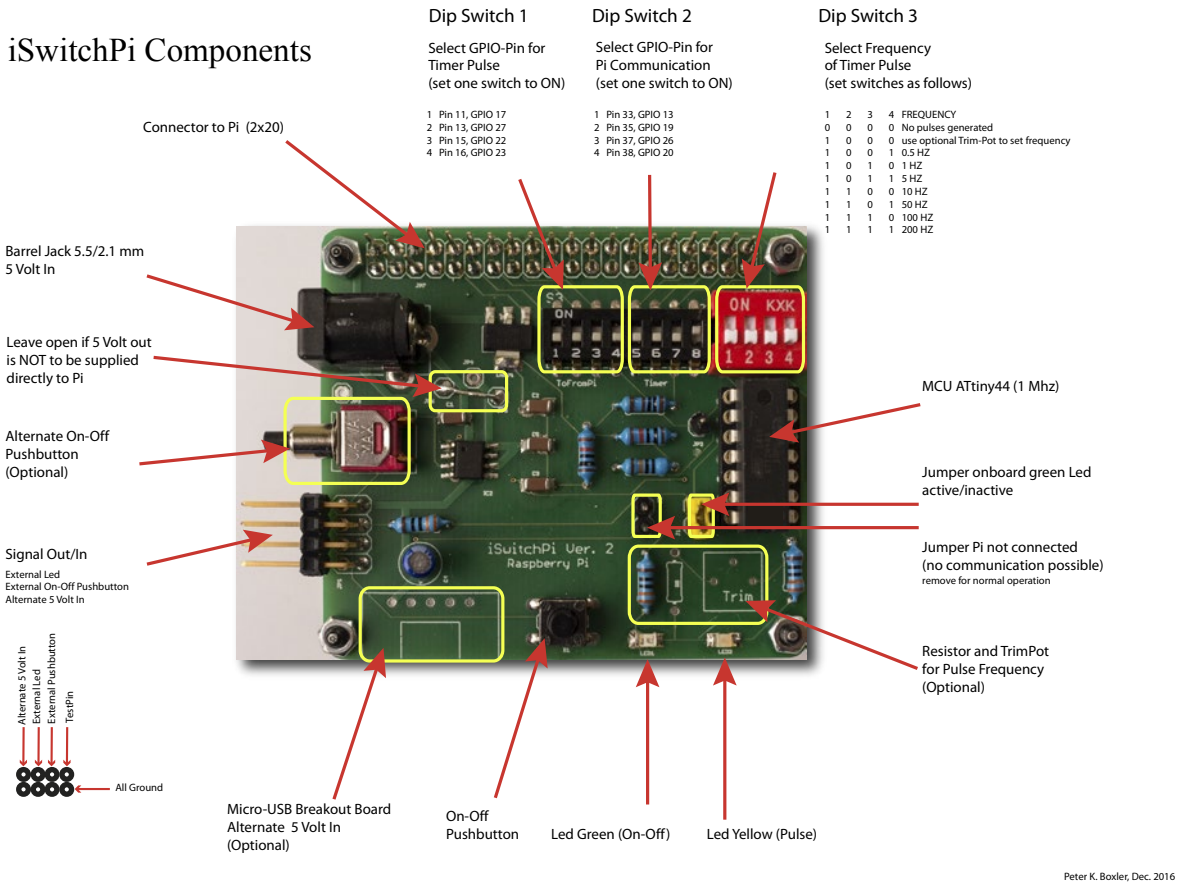
On a Mac OSX 10.9 (and higher) one needs to install AVRDUDE Version 5.11.1. Version 6.01 does not work with Olimex ISPMK2. Check out these discussions:

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

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

That is why I have installed CrossPack-AVR-20121203.

10. Components on the PCB



Komponenten auf den iSwitchPi Board

11. Conclusion

The attempt to formulate the problem description by means of a Finite State Machines has cost a lot of effort, but has made the solution much clearer. Writing the C program was relatively simple since state diagram and the state tables define everything. There were no serious logical problems with the program flow during testing.

What again confirms the old truth of sytems engineering; think first, code later.

12. Links

My Raspberry Projects

Projekt-Website

Source Code iSwitchPi incl. Eagle Files

GitHub

Other Links

Initial State Raspberry Pi GPIO Pins

Raspberry Foundation

About Finite State Machines

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

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

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

The Best Resource on the Web
https://www.adafruit.com

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

13. Attachments

13.1 Parts List iSwitchPi

Part	Value	Package	Note	Suppplier	PartNumber
Condensator					
C1	1uF	SMD C1206	ceramic	Any	
C2	1uF	SMD C1206	ceramic	Any	
C3	1uF	SMD C1206	ceramic	Any	
C4	100uF	E2,5-6E		Any	
C5	1uF	SMD C1206	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	
DIP-Switch					
S2		DIP04S	Dip 4-Pol	Any	
S3		DIP08S	Dip 8-Pol	Any	

Pin Header					
J1		1X02	Onboard green Led On/Off		
J2		1X02	without Pi		
JP1		2X04	Input/Output		
JP2		1X01	TestPin		
JP3		1X01	Optional 5 V IN		
JP4		1X01	TestPin		
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

13.2 Code iSwitchPi

A snippet of the c-code implementing the Finite State Machine:

```
// ---- 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 & (1<<0))     // first 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

            first =0xff;                        // set first all other states
            first &= ~(1<<0);                   // clear first this state
            }
        if  (get_key_short( 1<<KEY0 ))          // get debounced keypress
            {
            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 & (1<<1))                     // first 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 genaration output on
PA5
            first =0xff;                        // set first all other states
            first &= ~( 1<<1)  ;                // clear first this state
            }

        pwm_check();

        if ((blinkwhat>0) && (sekunde > POWERON_Delay))
            {
            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
            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 & (1<<3))                     // first time throu ?
            {
            PORTA |= (1<<LED1);                 // led full on
            blinkwhat=0;
            key_clear( 1<<KEY0 );

            first =0xff;                        // set first all other states
            first &= ~( 1<<3)  ;                // clear first 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;  // 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;  // 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;   // 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 & (1<<4))                     // first time throu ?
            {
            PORTA |= (1<<LED1);                 // led full on
            blinkwhat=0;
            key_clear( 1<<KEY0 );                // ignore keypresses that might have come

            first =0xff;                        // set first all other states
            first &= ~( 1<<4)  ;                // clear first 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;  // 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 & (1<<5))                     // first 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 =0xff;                        // set first all other states
            first &= ~( 1<<5)  ;                // clear first 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
            }


            // 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 =0xff;                        // set first all other states
        key_clear( 1<<KEY0 );

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

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

Ende des Dokumentes.

Peter K. Boxler, Februar 2016