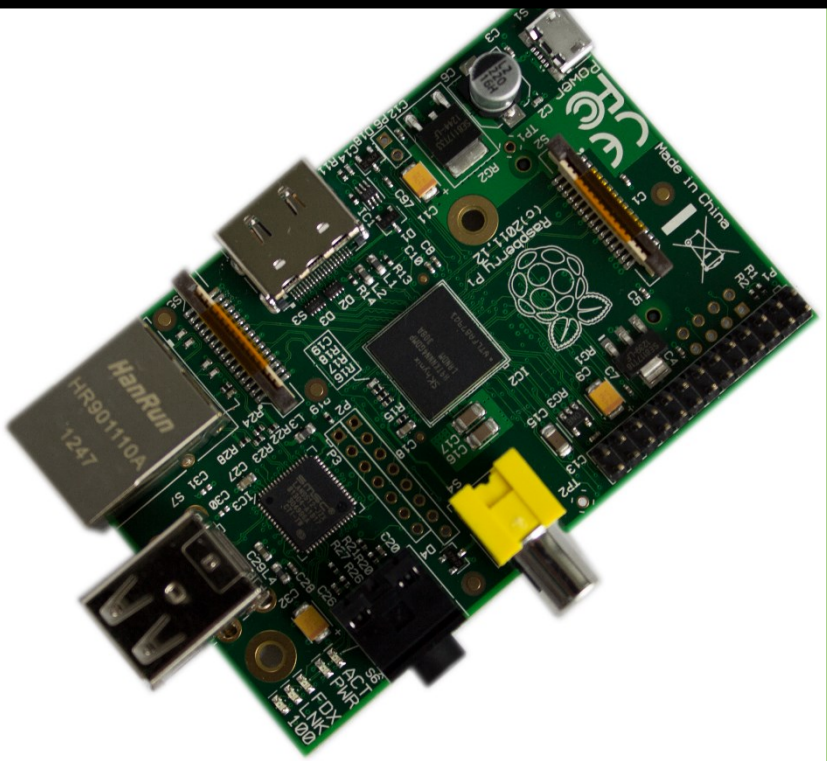
HOCHSCHULE
MITTWEIDA
UNIVERSITY OF
APPLIED SCIENCES

Raspberry Pi Baukastensystem für Raspberry Pi



Master Thesis

Entwicklung eines Baukastensystems
für den Raspberry Pi

Michael Kos

Abgabetermin:
30.4.2015

Michael Kos

Deadline: 2015.04.30

Master Thesis

Development of a modular system for Raspberry Pi

HOCHSCHULE
MITTWEIDA
UNIVERSITY OF
APPLIED SCIENCES

Michael Kos
Matrikel-Nr.: 23028
ZM12w1-M

Industrial Management

Prof. Dr.-Ing. Schmalwasser, Wilfried

Dr.-Ing. Krupke, Jörg

Abstract

Der Raspberry Pi ist ein kleiner frei programmierbarer Rechner der als Betriebssystem Linux nutzt. Mit diesem kleinen Computer lassen sich dank der Hardware- und Softwarefreiheiten viele interessante Projekte erstellen. Die Sparte reicht von einem Mediacenter, Haussteuerung bis hin zu einem voll funktionsfähigen Roboter. Diese zahlreichen Einsatzmöglichkeiten führten zur Erstellung der Master Thesis. Die Masterarbeit setzt sich mit dem Raspberry Pi und der Entwicklung eines Baukastensystems auseinander. Das Baukastensystem soll den Schülern/innen und Studenten/innen zeigen, wie einfach es ist einen Raspberry zu bedienen. Dabei lernen sie in zahlreichen Modulen das Betriebssystem aufzusetzen, Hardware zu programmieren oder einen Roboter zu bauen, der sich selbständig bewegt oder mit dem Webbrowser gesteuert wird.

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

App	Anwendungssoftware
ARM	Advanced RISC Machines
CPU	Central Processing Unit
CSI	Camera Serial Interface
DSI	Digital Serial Interface
FAT	File Allocation Table
GB	Giga Byte
GPIO	*General Purpose Input/Output*
GPU	*Graphics Processing Unit*
HDMI	High Definition Multimedia Interface
iOS	Iphone *Operating System*
IP	Internet Protokoll
LAN	Local Area Network
LED	*Licht-emittierende Diode*
MB	*Mega Byte*
NTSC	National Television Systems Committee
OS	*Operating System*
PAL	Phase-Alternation-Line
PC	Personal Computer
RAM	Random Access Memory
RasPi	Raspberry Pi
SD-Karte	Secure Digital Memory Card
SMS	Short Message Service
SoC	System on a Chip
TFT	*Thin Film Transistor*
URL	Uniform Resource Locator
USB	Universal Serial Bus
WiFi	„Wireless Fidelity“, gleiche Bedeutung wie WLAN
WLAN	Wireless Local Area Network
MOSI	Master out Slave In
MISO	Master in Slave out
SCLK	Clock

Inhaltsverzeichnis

1. Einleitung

In den letzten Jahren schreitet die Entwicklung der Computertechnik rasant voran und der Lebenszyklus eines Produkts auf dem Markt verkürzt sich drastisch. Wo früher viele Geräte RAM-Speicher, CPU und Grafikchip einzeln in ein Elektronikgerät verbaut wurden, befindet sich heute alles auf einem Chip. Das hat zur Folge, dass die Geräte immer kleiner, schmaler und handlicher werden. Durch diese Entwicklung entsteht für die jüngere Generation ein großer Nachteil, denn durch den schnellen Fortschritt der Technik wissen viele zum größten Teil nicht, wie ein Computer aufgebaut ist und wie dieser funktioniert. Die Hersteller geben in der heutigen Zeit oft nicht mal die Möglichkeit dazu Laptops, Tablets oder Smartphones aufzuschrauben, Teile auszuwechseln, hardwaremäßig zu erweitern oder etwas dran zu löten. Entweder sind die Geräte kompliziert verschraubt, verklebt oder bieten nicht die Möglichkeit Verbesserungen vorzunehmen oder die Hardware zu analysieren.

Diese Nachteile führten auch zur Geburt von Eben Uptons „Raspberry Pi Foundation“, eine Gemeinnützige Organisation um den Raspberry Pi. Eben Upton und seine Kollegen von der Universität Cambridge machten sich Sorgen, dass die Zahl der qualifizierten Studenten für Informationstechnik jedes Jahr weiter sinkt. Nach intensiven Recherchen, zusammen mit anderen Institutionsmitgliedern, fanden sie heraus, dass es kaum noch frei programmierbare Rechner auf dem Markt gab. Und so entstand die Idee für den Raspberry Pi.[1]

Nachdem der Raspberry seinen erfolgreichen Start in der Welt der Kommunikationstechnik gefeiert hat, entstehen seitdem fast jeden Tag neue interessante Projekte, die selbst nachzubauen sind oder an denen mitgewirkt werden kann. Die Projekte reichen von „Codeing“, Entwicklung von Distributionen, Skripte bis hin zum Bau von Überwachungskameras, Haussteuerung, Supercomputern oder Robotern, sowie der Entstehung von neuen Hardwarekomponenten.

Leider sind viele der Tutorials und Beschreibungen nicht sehr ausführlich und lassen viele Fragen offen. Sie sind oft nicht vollständig, einige Schritte werden sogar aus der Erwartungshaltung an den Nutzer ausgelassen oder sind in einigen Ausnahmen sogar fehlerhaft.

Das Ziel dieser Master Thesis ist es, Schülern/innen und Studenten/innen den Raspberry Pi schmackhaft zu machen. Es soll mittels Raspberry Pi das Interesse geweckt werden, damit mehr Jugendliche gefallen an Kommunikationstechnik, Informationstechnik und Elektrotechnik finden. Denn die Sparte der Einsatzmöglichkeiten des Raspberry Pi´s im Hardware- und Softwarebereich ist vielseitig. Eben Uptons Ziel ist es wieder mehr qualifizierte Informatikstudenten in Universitäten vorzufinden. Damit dies umgesetzt werden kann, müssen sich bereits Jugendliche im frühen Alter für den RasPi interessieren.

Im Netz fehlt für das Vorhaben eine Struktur, die einem Laien den Pi Schritt für Schritt erklärt. Deswegen soll in der Zeit der Masterarbeit, ein Baukastensystem für den Raspberry Pi entwickelt werden. Das Baukastensystem bringt den Schülern/innen und Studenten/innen „Step by Step“ den RasPi näher. Von den Grundlagen auf, bis zu komplexen Hardwareprogrammierungen werden in einzelnen Modulen die Möglichkeiten und Fähigkeiten des RasPi´s aufgezeigt. Die einzelnen Module, können dann von den Schüler/innen in Gruppen aufgebaut und verbessert werden, dabei lernen sie auf spielerische Art und Weise die Linux-Welt und deren Programmiersprache „Python“ kennen. Es soll den Einstieg erleichtern, die Hardware und Software verständlicher machen. Für dieses Vorhaben wird eine Anleitung für das Baukastensystem mit leicht verständlicher

[1] Raspberry Pi GEEK, Bruce Byfield 2014, S.6

Dokumentation erstellt. Der Inhalt teilt sich in einzelne Module auf, die den Schwierigkeitsgrad erhöhen.

Die Masterarbeit ist in 5 Kapitel unterteilt. In Abschnitt 2 werden die Grundlagen über den Raspberry Pi vermittelt. Der Aufbau und die Funktionsweise der Hardware werden genauestens analysiert. Es werden alle Modelle der kleinen Platine mit den Unterscheidungsmerkmalen aufgelistet und die Konkurrenzprodukte werden unter die Lupe genommen und durchleuchtet. Die Popularität des RasPi´s führte dazu, dass innerhalb kürzester Zeit viele Gadgets herausgebracht wurden. Welche Produkte das meiste ansehen haben und wie sie funktionieren, ist in Kapitel 2.6 zu finden. Als letztes dürfen Distributionen und verschiedene Software nicht fehlen.

Abschnitt 3 setzt sich mit der Herangehensweise zur Entwicklung des Baukastensystems auseinander. Er beschreibt das konkrete Ziel der Arbeit und den Nutzen für Schüler/innen, Studenten/innen und Ausbildungsstätten.

Abschnitt 4 setzt sich mit der Entwicklung der einzelnen Module für das Baukastensystem auseinander. Dabei werden Lösungsansätze, Ideen und Gedanken die damit verbunden sind, für einzelne Experimente beschrieben.

Im letzten Abschnitt werden die Ergebnisse, die Schlussfolgerungen, das Fazit und Ausblick in die Zukunft von Raspberry Pi und dem Baukastensystem analysiert und beschrieben.

2. Grundlagen zum Raspberry Pi

2.1 Die Entstehungsgeschichte

„Nach sechs Jahren der Entwicklung ernten der Projektgründer Eben Upton und die anderen Mitglieder der Raspberry Pi Foundation jetzt die Früchte ihrer Bemühungen. Der Weg war steinig, wie die Bestandsaufnahme des Raspi-Vaters zeigt“.[2]

Am 29 Februar 2012 wurde der Raspberry Pi offiziell zum Verkauf angeboten und zählt seitdem zu den erfolgsreichsten Open Source Projekten der letzten Jahre. Der RasPi war innerhalb weniger Stunden ausverkauft, wer keinen bekommen hatte musste sich auf eine Lieferzeit von mehr als einem Monat einstellen. Die Idee des kreditgartengroßen Rechners entstand in Großbritannien. Eben Upton und seine Kollegen von der Universität Cambridge machten sich Sorgen, dass die Zahl der qualifizierten Studenten für Informationstechnik jedes Jahr weiter sinkt. Nach intensiven Recherchen zusammen mit anderen Institutionsmitgliedern fanden sie heraus, dass es kaum noch frei programmierbare Rechner auf dem Markt gab. Und so entstand die Idee für den Raspberry Pi: *„ein Gerät zu entwickeln, das in einem Kinderzimmer sein Platz fände, und so den Zustrom von Bewerbern zu sorgen, den eine lebendige universitäre Umgebung braucht.“*[3] Und somit entstand nach 6-jährigen Entwicklungszeit ein Raspberry Pi der dazu ermutigt, *„sich bis ins letzte Detail damit auseinanderzusetzen, ohne Barrieren zwischen User und Hardware.“*[4]

Die Dauer der Entwicklung ist darauf zu führen, dass Upton und seine Kollegen keine Ahnung hatten wie das Gerät aussehen sollte. Sie hatten nur ein Kriterium festgelegt, *„einen PC bauen, der so billig ist, dass selbst Schüler ihn sich leisten können, der trotzdem leistungsfähig ist und so robust, dass man damit gefahrlos herum experimentieren kann.“*[5] Der Durchbruch gelang erst als Upton zum kalifornischen Chiphersteller „Broadcom“ wechselte. In dieser Zeit wurde für den Raspberry Pi eine

---

[2] Raspberry Pi GEEK, Bruce Byfield 2014, S.6
[3] Ebenda, S.7
[4] Ebenda, S,7
[5] http://www.gruenderszene.de/allgemein/raspberry-pi-eben-upton

Stiftung gegründet. Da Upton durch seinen neuen Beruf nicht mehr so viel Zeit in das Projekt investieren konnte, arbeitete er nur noch in seiner Freizeit daran. Erst als 2011 Broadcom einen Chip auf den Markt bringt der *„klein genug für Smartphones, aber auch groß genug, um eine CPU und einen Grafikprozessor unterzubringen“*[6] gelingt der Durchbruch für den Pi.

Die Entwickler von RasPi haben nie damit gerechnet dass die Bestellungen so in die Höhe schießen. Der Hintergrund ist, viele Hacker und Hardwarebastler haben schnell Interesse an dem kleinen Gerät gefunden. Aber der Großteil der Nachfrage kam aus Russland, Brasilien oder aus afrikanischen Ländern. Dort also, wo viele Menschen zwar einen Fernseher haben aber sich nie einen Computer leisten konnten. So wird das TV-Gerät dort auch als Computerbildschirm benutzt.

Mittlerweile hat die Raspberry Pi Foundation weitere, verbesserte Raspberry Pi´s herausgebracht und dort wo der Erfolg ist schläft die Konkurrenz nicht lange. Die Zahl an neuen Klonen wächst immer mehr aber bislang hat noch keiner den Durchbruch geschafft. Und schon in diesem Jahr, am 2. Februar 2015 hat Eben Upton den sofortigen Verkauf von RasPi 2 angekündigt.

2.2 Die verschiedenen Raspberry Pi´s

Der Raspberry Pi ist seit Februar 2015 in fünf verschiedenen Varianten erhältlich. Außer Raspberry Pi 2 unterscheiden sich die vier Platinen in der Ausstattung kaum voneinander. Betrachtet man Modell A, B und B+, fällt auf, dass sie sehr ähnlich aussehen. Modell B z.B. beinhaltet ein USB-2.0-Anschluss mehr und besitzt ein 100Mbit LAN-Eingang als Modell A. Dagegen hat Modell B+ vier USB-Anschlüsse und zusätzliche 14 GPIO-Pins. Die Modelle A und B benötigen eine Spannung von 5-Volt, der Unterschied liegt nur in der Stromaufnahme: Beim Modell A reichen theoretisch 500mA und Modell B benötigt mindestens 700mA. Auf beiden Platinen ist ein Broadcom BCM2835 Chip montiert (eine ausführliche Beschreibung des Soc-Chips findet man unter *http://raspberrypiguide.de/stuff/bcm2835-arm-peripherals.pdf*, ein ARM11-basierender Prozessor, der mit 700MHz getaktet ist. Der Raspberry Pi 2 ist im Gegensatz zu den anderen Modellen mit 4 Kernen ausgestattet und besitzt einen ARM-Prozessor der nächsten Generation. Tabelle 2 stellt die fünf Modelle mit ihren ganzen Eigenschaften vor.

[6] Ebenda

Modelle	Modell A	Modell B	Modell B+	Compute Module	Raspberry Pi 2
Platinengröße	85,60 mm × 56 mm			67,6 mm × 30 mm	85,60 mm × 56 mm
Zentraleinheit:	SoC Broadcom BCM2835				BCM2836
• **CPU**	ARM1176JZF-S mit 700 MHz (bis zu 1000 MHz übertaktbar)				ARM Cortex-A7 mit 4 Kernen und 900MHz
• **GPU**	Broadcom VideoCore IV, OpenGL ES 2.0, OpenVG 1080p60				
• **SDRAM**	256 MB	512 MB			1024 MB
USB-2.0Port	1 x USB-Buchse	2x USB-Buchse	4x USB-Buchse		
Videoausgänge	RCA Video-Out / HDMI		HDMI / RCA Video-Out (integriert in Klinkenstecker)	RCA Video-Out / HDMI	HDMI / RCA Video-Out (integriert in Klinkenstecker)
Audioausgabe	3,5mm Klinken-Buchse (analog) , HDMI (digital)			HDMI	3,5mm Klinken-Buchse (analog) , HDMI (digital)
SD-Karten-Slot	Empfohlen Class 10 SD-Karten ab 4 GB (SDHC/SDXC)		microSD (SDXC/SDXC)	4GB eMMC	microSD (SDXC/SDXC)
Netzwerk-Controller		SMSC LAN9512 10/100Mbit/s			SMSC LAN9512 10/100Mbit/s
Schnittstellen:					
• **CSI**	Für RaspiCam; Anschluss 15-poliges Flachbandkabel				1x
• **DSI**	Display-Anschluss				2x
• **GPIO**	26 Pins	26+8 Pins	40 Pins	60 Pins	40 Pins
Leistungsaufnahme	5V, 500mA (2,5 W)	5V, 700mA (3,5 W)	5V, 1500-2000mA (2,5-3,5 W)	3,3V, 1,8V	5V, 800mA (maximal 4W)
Stromversorgung	5-V-Micro-B-USB-Anschluss			3,3V, 1,8V	5-V-Micro-B-USB-Anschluss
Distributionen	NOOBS, GNU/Linux (Debian, Fedora, Arch Linux), Risc OS, OpenELEC, RaspBMC Plan 9, Android usw.				NOOBS, Raspbian, Snappy Ubuntu Core, OpenELEC

Tabelle 1 Die verschiedenen Hardware-Spezifikationen der fünf Modelle

2.2.1 Raspberry Pi Compute Module

Bei diesem Modell handelt es sich um eine DIMM Version des Raspi´s. Wie man anhand der Tabelle 2 erkennen kann, grenzen die Spezifikationen sehr an das Modell A. Das Gerät sieht aus wie ein RAM-Baustein und passt in die SODIMM-Slots. Das Modul wird ohne die ganzen Anschlüsse wie HDMI, USB, LAN usw. ausgeliefert. Die Anschlüsse finden sich erst auf dem IO-Board, was die „Raspberry Pi Foundation“ mit anbietet, diese Hauptplatine ist als Entwicklungsboard gedacht. Das Modul ist für verschiedene Unternehmen gedacht, weil viele Firmen versuchen den RasPi in ihre Arbeitszyklen zu integrieren. Deswegen hat Upton das Modul entworfen und herausgebracht. *"Wir waren überrascht, dass einige Produzenten Raspberry Pis in ihre Systeme einbauten. Typischerweise handelt es sich dabei um kommerzielle oder industrielle Systeme, die nur in kleinen oder mittleren Stückzahlen produziert werden."* [7]

Mehr Informationen über das Modul findet man auf *http://www.raspberrypi.org/raspberry-pi-compute-module-new-product/*

Abb1. Das Raspberry Pi Compute Modell

2.2.2 Raspberry Pi 2

Seit Februar 2015 hat die „Raspberry Pi Foundation“ mit dem Raspberry Pi 2 die zweite Generation der „Beere“ eingeleitet. Die Konstruktion der Platine wurde von Modell B+ übernommen, so dass außer der Größe des SoCs keine weiteren Veränderungen stattgefunden haben. Der RAM-Speicher des SoCs wurde von 512 MB auf 1024MB erweitert und der Prozessor durch den ARM Cortex-A7 mit 4 Kernen a 900MHz Taktfrequenz ersetzt. Dank des neuen ARM-Prozessors wir die volle Unterstützung der ARM GNU/Linux Distributionen gewährleistet. Bei der ersten Distribution, die lauffähig für den Raspberry Pi 2 angeboten wird, handelt es sich um „Snappy Ubuntu Core“[8]. Die Internetseite „Golem“ hat den RasPi 2 ausführlicher getestet und einige Benchmarks durchgeführt. *„Wie markant der Geschwindigkeitsunterschied ist, fällt erst auf, als wir das Pi 2 und das Pi 1 im direkten Vergleich gegeneinander antreten lassen.“*[9] Sie stellten dabei fest, dass der Bootvorgang gegenüber dem Raspberry B+ um einiges schneller durchgeführt wird. Auch auf der grafischen Oberfläche (LXDE) arbeitet der Raspberry 2 die Befehle schneller ab. Der einzige Nachteil der auch bei den Vorgängern zu bemängeln ist, dass sich leider in der Praxis die schlechte Performance bei Dateioperationen auf der Micro-SD-Karte stark bemerkbar macht.[10] Der Erscheinungstermin der

---

[7] http://www.golem.de/news/raspberry-pi-compute-module-ab-sofort-lieferbar-1406-106550.html
[8] http://www.raspberrypi.org/downloads/
[9] http://www.golem.de/news/raspberry-pi-2-schnell-rechnen-langsam-speichern-1502-112107.html
[10] http://www.golem.de/news/raspberry-pi-2-schnell-rechnen-langsam-speichern-1502-112107-3.html

zweiten Generation macht es unmöglich sich weiter mit dem Gerät auseinander zu setzen und wird wegen wenigen Tests aus dem Grund nicht ausführlich in die Arbeit einfließen.

2.2.3 Raspberry Pi Alternativen

Nachdem der Raspberry Pi für viel Aufsehen sorgte und sehr schnell Communities in vielen Ländern entstanden sind, haben andere Unternehmen angefangen eigene Boards zu entwickeln. In diesem Unterkapitel werden vier weitere Einplatinencomputer beschrieben. Tabelle 4 zeigt die Spezifikationen der einzelnen Boards auf.

Banana Pi

Der Banana Pi kommt aus China und besitzt eine bessere Preformence als der RasPi selbst. Der SoC-Chip ist mit einen Dual-Core Prozessor ausgestattet, der bis zu 1 GHz getaktet werden kann. Der RAM-Speicher ist 1 GB groß. Der Banana Pi besitzt gegenüber dem RasPi zusätzliche Anschlüsse, wie z.B. ein SATA-Anschluss, ein Mikrofon, ein Infrarotempfänger und drei Schalter. Die drei Schalter haben folgende Funktionen: Power key zum ein-/ausschalten und ein Resetknopf um das Board neu zu starten. Für das Gerät sind zahlreiche Distributionen im Umlauf, die von Android bis Linux reichen. Die Distributionen müssen für die ARMv7-Platform angepasst sein (Download-Link: *http://www.lemaker.org/resources/9-38/image_files.html*). Unter den Distributionen befindet sich auch ein modifiziertes Raspbian OS, das durch mehr RAM-Speicher, GPU und Prozessors schneller und flüssiger läuft.

Der Banana Pi besitzt 40 GPIO-Pins, die Belegung der Pins ähnelt sehr dem RasPi. Einige der Pins besitzen einen CAN-Bus, einen Analog-Digital-Wandler (ADC) und sind PWM-Fähig. Das Verhalten der GPIOs ist identisch zu Modell B+, wenn die Raspbian Distribution auf dem Banana Pi installiert ist. Einige Erweiterungs-Boards von Raspberry Pi sind auch zu Banana kompatibel. Nicht kompatibel sind die Kamera Module und Displays, da die CSI und DSI auf dem RasPi mit 30 Pins ausgestattet sind und der Banana Pi 40 Pins besitzt.[11]

Beagle Bone Black Rev C

Der Beagle Bone ist für Robotics-Anwendungen eines der beliebtesten Produkte auf dem Markt. Seine Ausstattung ist sehr bescheiden, der Beagle ist nur mit einem USB-Port ausgestattet, einem LAN-Anschluss und einem Micro-HDMI-Anschluss. Im Gegensatz zum RasPi und Banana Pi besitzt der Beagle zusätzlich zu dem SD-Kartenslot einen emmC-Flashspeicher mit 4GB.

Der Beagle besitzt 2 x 46 Pins (davon 66 GPIOs). Ab der Beagle Bone Black Rev C Version ist es möglich verschiedene Distributionen auszuführen (*http://beagleboard.org/latest-images*). Dank des Flash-Speichers bootet der Beagle das System innerhalb von 10 Sekunden.[12]

Arduino Yún

Das Board vereint den Mikrokontroller Arduino und einen Mikroprozessor auf eine Platine. Seit der Verschmelzung können Linux Distributionen ausgeführt werden. Wie der Name schon sagt ist diese Platine kompatibel zum Arduino-Board und besitzt ein WLAN-Adapter, ein Ethernet-Anschluss, ein USB-Port. und verfügt über einen Flashspeicher. Das Arduino Yún ist kompatibel mit vielen Erweiterungen für den Arduino-Mikrokontroller.[13]

---

[11] http://www.golem.de/news/banana-pi-im-test-bananen-sind-keine-himbeeren-1408-108314.html
[12] http://elinux.org/Beagleboard:BeagleBoneBlack
[13] http://arduino.cc/en/Main/ArduinoBoardYun

Intel Edison

Das letzte Board ist eine Intel-Produktion, die nicht auf einen ARM-Prozessor setzt sondern auf die alt bekannte x86-Platform. Es ist das kleinste Board in der Single-Board-Computer Kategorie, es ist so groß wie eine Briefmarke. Seine Ausstattung fällt deswegen auch viel bescheidener aus als bei den anderen Boards. Es besitzt einen Bluetooth- und einen Dual-BAND-Wlan-Adapter. Damit das Board seine volle Leistung ausschöpfen kann benötigt es kompatible Träger-Boards.

Als Betriebssystem stellt Intel eine Linux-Distribution namens Yocto auf ihrer Community Webseite (*https://communities.intel.com/docs/DOC-23242)* zur Verfügung.[14]

Modell	**Banana Pi**	**Beagle Bone REV C**	**Arduino Yun**	**Intel Edison**
Hersteller:	Lemaker	Beaglebon.org F.	Dog Hunter	Intel Corpo.
Preis	ca. 40 €	ca. 60 €	ab 66 €	ca. 60 €
	CPU / GPU / Speicher			
SoC	Allwinner A20, Dualcore,	Texas Instruments Sitara AM3358	Atheros AR9331 / ATmega 32U4	Dualthreaded Atom, Dualcore / Quark-Microcontroller
Takt	1 GHz	1 GHz	400 MHz / 16 MHz	500 MHz / 100 MHz
CPU	ARM Cortex-A7	ARM Cortex-A8	MIPS	Atom Silvermont
GPU	ARM Mali 400 MP2	Power VR SGX530	-	-
RAM	1 GB	512 MB	64 MByte und 2,5 KByte SRAM (MCU)	1 GB
Flash	-	4 GByte EMMC	16 MByte und 32 KByte (MCU)	4 GByte EMMC
Bootloader	Sunxi Uboot	Serial Boot/USB-Boot (EMMC/SD)	-	Uboot
	Schnittstellen:			
Grafik/Video	HDMI, Composite, DSI	Micro-HDMI, LCD via Expansion-Header	-	-
Audio	HDMI, Analog-Out (3,5-mm-Klinke), Mikrofon auf dem Board	Micro-HDMI	-	-
Display	DSI	per Cape Add-on		
Kamera	CSI	GPMC (Cape Add-on)		
Ethernet	100 GBit/ s	100 MBit/ s	100 MBit/ s	
WLAN			802.11 b/ g/ n	802.11 a/ b/ g/ n, Dualband
Bluetooth				v 4.0
Speicher	SD, SATA	Micro-SD, 4 KByte EEPROM onboard, Flash	Micro-SD, 1 KByte EEPROM, Flash	Flash, SD-Karten-Schnittstelle
USB	2x USB-2.0 + 1 x USB OTG	1x Mini-USB-2.0-Client, 1x USB-2.0	1x USB-2.0	1x USB-2.0-OTG

[14] http://www.golem.de/news/intel-edison-ausprobiert-ich-seh-dich-das-mona-lisa-projekt-1411-110577.html

PINS	26 Pins + 10 Pins Leisten	2 x 46 Pins Leisten	10+8+8+6-Pin-Buchsenleisten (Arduino-Standard)	70-Pin-Hirose-DF40-Steckverbinder
GPIO-Pins	23 Pins	65 Pins	20 Pins	40 Pins
Weitere Busse	SATA	LCD, GPMC	-	1x I2S
Netzteil	5 V/2 A/per Micro-USB	5 V/500 mA/per Mini-USB oder Rundbuchse	5 V/350 mA/per Micro-USB, PoE optional	3,3 bis 4,5V /k.A./k.A.
	Distributionen			
Linux	Bananian, Raspbian, Debian, Ubuntu, Android, Open WRT	Ångström, Debian, Android, Ubuntu, Arch Linux	Open WRT Yún (basiert auf Open WRT)	Yocto Linux
Andere	-	Free BSD	-	Rtos für Quark X1000

Tabelle 2 Spezifikationen der vier Single Board Computer

2.3 Die Hardware

Die Experimente sind mit dem Raspberry Pi Modell B rev.2 durchgeführt worden, deswegen ist der Schwerpunkt in der Hardwarebeschreibung auf seine Spezifikationen ausgelegt. D.h. die Beschreibung ist zum größten Teil auf alle Raspbery´s 1 übertragbar. Denn die anderen Modelle sind fast baugleich und weisen hardwaretechnisch nur wenige Unterschiede auf. Bis auf den Raspberry Pi 2 sind alle Modelle 100% kompatibel zueinander.

2.3.1 Raspberry Pi Modell B rev. 2

Die wichtigsten Merkmale des Raspberry Pi´s sind, dass die Stromversorgung über den Micro-USB-Anschluss erfolgt, der vorwiegend bei Smartphones und Navigationssystemen verwendet wird. Für einen stabilen Betrieb nutzt der Raspberry eine Eingangsspannung von 5V und Stromstärke von 1000mA. Auf dem Raspberry Pi sind 2 Chips integriert (der ARM-Prozessor und der LAN-Controller), die die gesamte Arbeit verrichten. Der ARM-Prozessor ist ein von Broadcom (BCM2835) entwickelter Chip, der sich auch in Smartphones und Mp3-Playern vorfindet. Dabei handelt es sich um einen SoC (System-On-A-Chip), das bedeutet CPU, GPU und der RAM-Speicher befinden sich auf einem einzigen Chip.

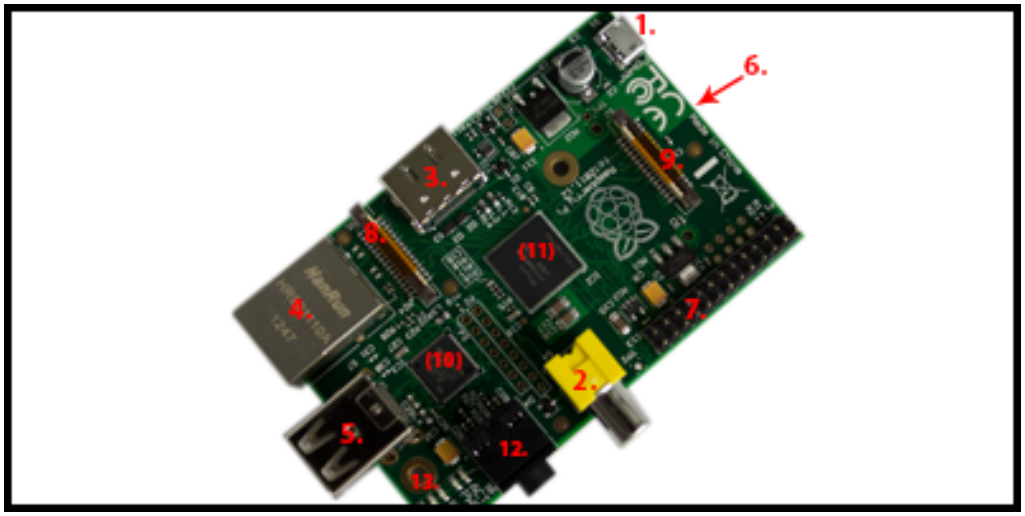


Abb2. Raspberry Pi Modell B Anschlüsse und Chipsätze

Die Schemata des Raspberry Pi´s ist als Anhang vorzufinden. Die Tabelle 2 beschreibt die einzelnen Hardwarekomponenten mit dem der RasPi ausgestattet ist:

1.	**Micro-B-USB Anschluss 1000mA (5V)**	Der Raspberry Pi benötigt alleine 700mA (entspricht 3,5Watt). Für eine fehlerfreie Stromversorgung ist eine Spannung von 5 Volt mit mind. 1000mA Stromstärke (Leistung von bis zu 5 Watt) empfohlen. Bei Überschreitung der Stromstärke führt es zu einer Unterversorgung und die an das Pi angeschlossene Hardware funktioniert nicht mehr ordnungsgemäß.
2.	**RCA Video-Out (PAL, NTSC)**	Der Anschluss ist für ältere Fernsehgeräte vorgesehen, die nicht über ein HDMI-Anschluss verfügen. Besitzt ein Fernsehgerät kein RCA Video so kann er mit einem *Composite-To-Scart*-Adapter über Scart verbunden werden, was aber eine schlechtere Bildqualität zu Folge hat.
3.	**HDMI (rev 1.3 und 1.4)**	Überträgt Audio- und Video. Das Bild wird bis zu 1920 x 1080 Bildpunkten in Full-HD-Auflösung am Ausgabegerät übergeben. Für Monitore mit DVI-Eingang wird ein Adapter benötigt.
4.	**10/100 MBit Ethernet**	Über die RJ 45-Buchse lässt sich mit einen LAN-Kabel der RasPi an einem Netzwerk anschließen.
5.	**Eingebauter HUB mi zwei USB 2.0 Anschlüssen**	Universal verwendbar mit maximalem Ausgangsstrom von 100mA.
6.	**Slot für SD-Karte (SDHC/SDXC)**	Der Steckplatz für die SD-Karte befindet sich auf der Rückseite des Minicomputers. Die SD-Karte ist in zwei Partitionen aufgeteilt. Auf der FAT 32 Partitionen befinden sich verschiedene

		Systemdateien, die für das Booten des Betriebssystems nötig sind. Auf der EXT4 Partition ist die Linux-Distribution (Siehe Bootvorgang). Die SD-Karte ist damit auch das virtuelle Laufwerk
7.	**General Purpose Input Output GPIO-Port (inklusive UART, I2c-Bus, SPI-Bus, I25-Audio)**	Zusätzliche 26 Pins. Die meisten werden als Input/Output verwendet. (siehe Abschnitt GPIO)
8.	**Kamera CSI (Camera Serial Interface)**	Die Schnittstelle für das Raspberry Pi Kamera Modul. Es wird über einen 15-poligen Flachbandkabel mit CSI verbunden. Der Vorteil gegenüber einer USB-Kamera besteht in der direkten Verbindung von Kamera-Modul und dem BCM2835 (SoC).
9.	**Display CSI (Display Serial Interface)**	Schnittstelle für ein Display mit integriertem Controller.
10.	**SMSC LAN9512 LAN-Controller**	Es ist ein USB 2.0 und ein 10 / 100 Ethernet Controller. Er ermöglicht Datenraten bis 100 Mbit/s im Netzwerk.
11.	**Broadcom BCM2835 Zentraleinheit**	Es handelt sich um einen SoC (System-On-A-Chip). In diesen Chip sind die CPU (in ARMv6-Architektur mit 700 MHz), GPU (VideoCore-IV-GPU mit H.264 Encoder/Decoder, die OpenGL ES2.0 unterstützt) und 512 MB SDRAM integriert.
12.	**3,5mm Klinken-Buchse**	(siehe **(2)**)
13.	**LEDs**	Liefern Informationen über aktuelle Tätigkeit des RasPis.

Tabelle 3 Die Hardware Komponenten des Raspberry Pi´s Modell B

Die vier LEDs (13) informieren über den aktuellen Status der kleinen Platine und die LAN-Verbindungsart (siehe Tabelle 2).

Nummer	Aufschrift	Funktion	Farbe
D5	ACT	Zugriff auf die SD-Karte	Grün
D6	PWR	Betriebsspannung	Rot
D7	FDX	Netzwerk (Full Duplex)	Grün
D8	LNK	Netzwerk (Verbindung/Aktivität)	Grün
D9	100	100-Mbit/s-LAN-Verbindung	Gelb

Tabelle 4 Zeigt die Funktionen der LEDs am Raspberry Pi.

2.3.2 SoC

Die Zentrale des Raspberry Pi´s ist der SoC (Abb. 3 System-On-A-Chip), es verbindet alle wichtigen elektrischen Elemente in einen Chip, die in einem Computer benötigt werden. Der SoC ist der Ausgangspunkt aller befehlsgesteuerten Aktionen und der Verarbeitung jeglicher Daten. Es handelt sich dabei um eine BCM2835 Zentraleinheit der Firma „Broadcom“. In diesem Chip sind die CPU (ARM1176JZF-S mit 700 MHz), die GPU (Abb. 4 VideoCore-IV-GPU mit H.264 Encoder/Dekoder, die OpenGL ES2.0 unterstützt) und einen von Samsung K4P4G324EB SDRAM-Speicher mit 512 MB integriert. Der IC wird in vielen Smartphones, Navigationsgeräten und Tablets genutzt, dies hat den Vorteil, dass der Chip mit einer Eingangspannung von etwa V_{DD} = 1,8V und Ausgangsspannung V_{DDq} = 1,2V auskommt. Dabei kann die IC eine Geschwindigkeit von 1066MBps erreichen.[15]

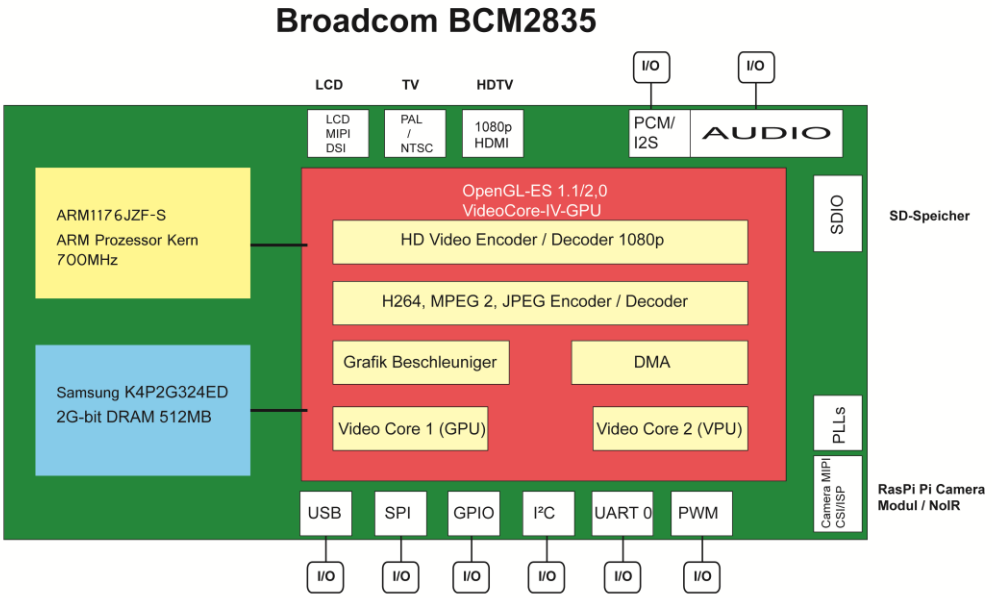


Abb3. Der SoC BCM2835 Blockschaltbild

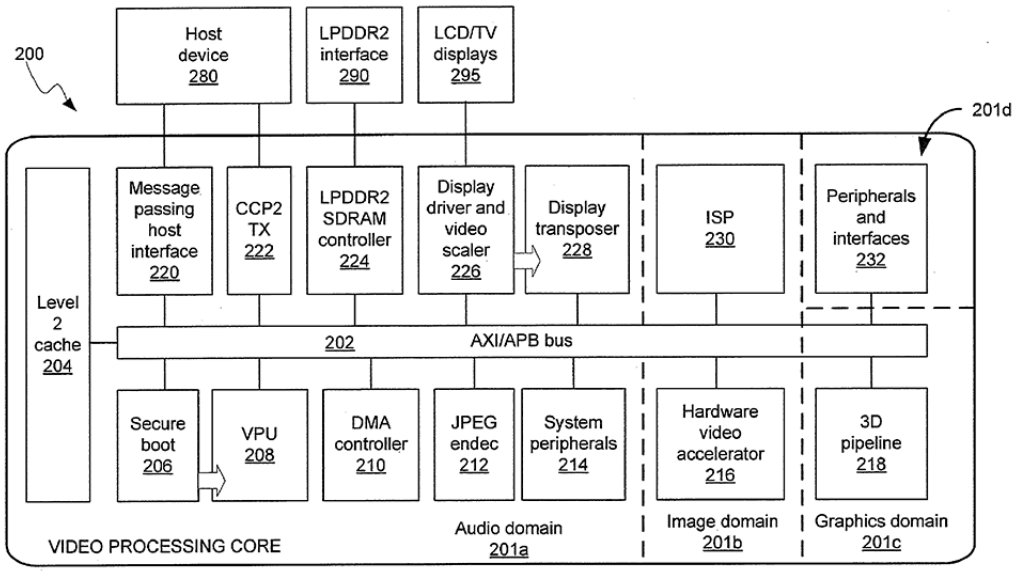


Abb4. VideoCore-IV-GPU

[15] http://www.samsung.com/global/business/semiconductor/product/mobile-dram/detail?productId=7609

Der BCM2835 Chip hat einen direkten Zugriff auf die meisten GPIO-Pins. Die GPIO-Pins sind mit größter Sorgfalt zu benutzen. Werden die Pins nicht mit der korrekten Spannung beschallt, kann es zur Beschädigung des SoCs führen (siehe Kapitel GPIO)

2.3.3 GPIO-Port

Der Raspberry Pi hat 26 GPIO Pins (General Purpose Input Output) integriert, die mit verschiedenen Schnittstellen verbunden sind. Pin 1 und 3 besitzen nur eine Funktion, die daran angeschlossene Hardware mit 3,3V zur Versorgung. Ebenso funktionieren die Pins 2 und 4 aber mit einer Versorgungsspannung von 5V. Der GPIO-Port besitzt 5 GND Pins die als Masse verwendet werden (Pin 6,9,14,20,25). Die restlichen 17 GPIO Pins sind als Eingabe/Ausgabe (Input/Output) deklariert. Einige der 17 Pins haben noch weitere Funktionen integriert (siehe Kapitel 2.3.3.3: Die 17 GPIO´s). Diese 17 Pins werden über verschiedene Programmiersprachen (Linux Bash, Python und C, etc.) gesteuert.

Auf die 26 Pins können externe Geräte gesteckt werden (siehe weitere Hardwarekomponenten). Zu dieser Hardware gehören z.B. das „Aduino-UNO-Board“, Touch-Display von „adafruit“ (*https://www.adafruit.com*), BrickPi von „Dexter Industries“ (*www.dexterindustries.com)* usw. Eine sehr ausführliche Beschreibung über weitere Komponenten ist auf der Seite: *http://elinux.org/RPi_Low-level_peripherals* zu finden.

Der GPIO-Port ist direkt mit dem SoC verbunden, dabei ist zu beachten, dass der Chip keine Schutzschaltung besitzt. D.h. beim Anschließen der elektronischen Komponenten muss immer darauf geachtet werden, dass sie fehlerfrei an die richtigen Pins angeschlossen sind und die maximale Stromstärke nicht überschritten wird. Bei falscher Verwendung kann es dazu führen, dass der RasPi beschädigt wird oder sich untypisch verhält. Deswegen müssen auch die Spezifikationen jeglicher elektrischer Bauelemente mit einbezogen oder mit einer zusätzlichen Energiequelle versorgt werden.

2.3.3.1 Rev.1 und Rev.2

Der Raspberry Pi hat 3 verschiedene GPIO-Ports, die sich in der Belegung unterscheiden, dabei muss erwähnt werden, dass sich ab Modell B+ 40 GPIO-Pins sich auf dem Board befinden.

Seit Herbst 2012 ist das Modell B als rev. 2 erschienen. Das Board wurde auf Wunsch der Community überarbeitet und zusätzlich um GPIO's (P5 Header für weitere 8 Pins) ergänzt. Zusätzlich zur Revision 1 Platine, die einige Pins mit DNC (Do Not Connect) beschriftet hat, sind diese Pins ab rev. 2 neu belegt. Die Pins 3, 5 und 13 sind auf der neuen Platine (rev.2) umbenannt (siehe Abb. 5).

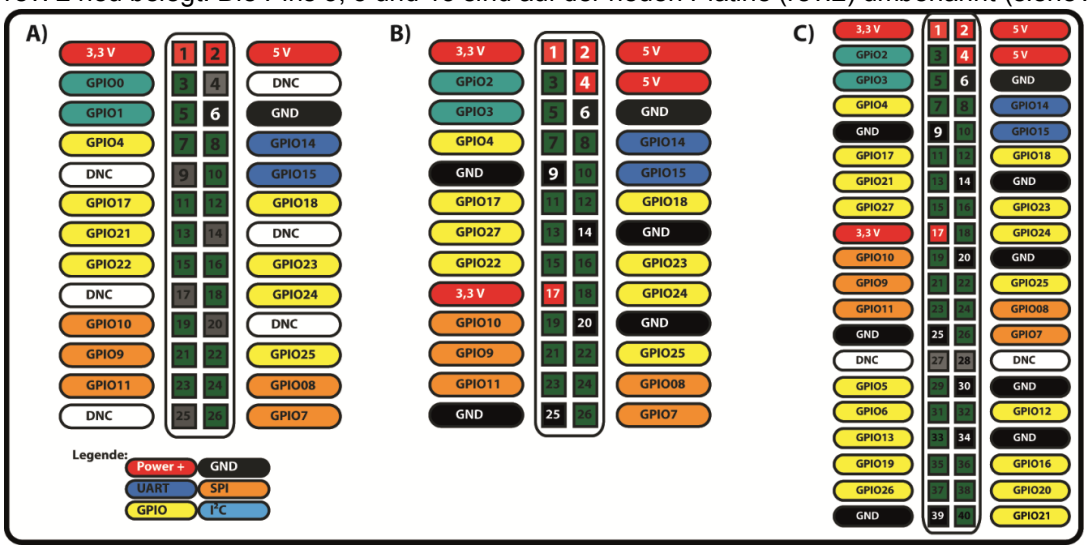


Abb5. GPIO-Pins A) Modell A bzw. Modell B rev.1 B) Modell B rev. 2 und c) Modell B+ und Raspberry Pi 2

2.3.3.2 Stromversorgung und GPIO

Im Allgemeinen wird der RasPi mit einer Spannung von 5V am Micro-USB-Eingang mit Energie versorgt. Der Eingangsstrom am Raspberry Pi hängt von den angeschlossenen Geräten ab. Das Modell B ist so konstruiert, dass bei minimalem Aufwand mindestens 700mA benötigt werden (ohne zusätzlich angeschlossener Hardware) und die maximale Stromaufnahme bei 1000mA liegt.

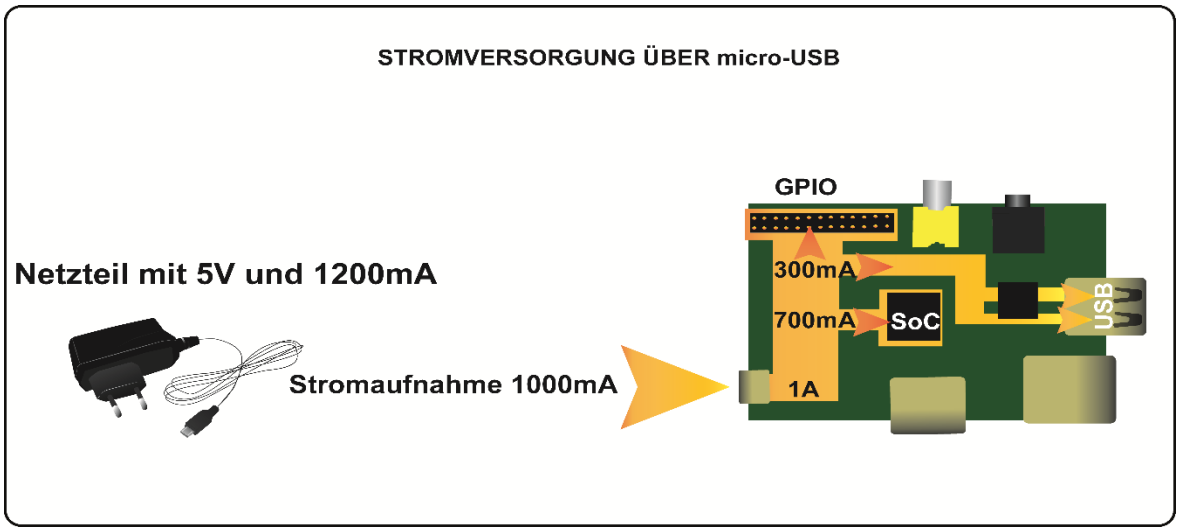


Abb6. Die Stromversorgung des Raspberry Pi Modell B

Bei Beschallung der einzelnen Pins am GPIO-Port, liefern die benutzten Pins Strom. Dabei kann ein einzelner Pin 16mA Strom liefern. Alle GPIO-Pins zusammen geben höchstens 50mA ab. Werden alle 17 Ein- und Ausgabe Pins als Ausgänge deklariert, so dürfen diese ungefähr 2,9mA benötigen.

Es gibt noch 9 weitere Pins die sich von der Spannungsversorgung unterscheiden. Pin 1 und 17 stellen eine Spannung von 3,3V bereit und Pin 2 und 4 beziehen ihre 5V Versorgungsspannung direkt von dem micro-USB-Stromanschluss. Die 0V Pins werden als „Ground“ (GND) bezeichnet und werden als Masse verwendet. Diese Pins haben immer 0V und können nicht wie alle anderen Pins mit Spannung gespeist werden.

Wird der Pi mit einem 1200 mA Netzteil betrieben, nimmt dieser trotzdem (abgesichert durch eine Sicherung) nur 1000mA auf. Der Pi alleine benötigt davon 700mA. D.h. es bleiben noch ungefähr 300mA übrig die sich die GPIO-Pins 2 mit den anderen Anschlüssen teilen müssen.

Raspberry Pi	ca. 700mA
HDMI	ca. 50mA
GPIO	bis 50mA
Ethernet	ca. 50mA
USB	pro USB-Eingang 100mA
CSI mit Kamera-Modul	Bis zu 250mA (je nach Anwendung)

Tabelle 5 Stromversorgung der wichtigsten Hardwareelemente des Raspberry Pi´s Modell B

Es sei noch einmal darauf hingewiesen, dass bei falscher Anwendung nicht auszuschließen ist, dass die Platine bzw. der SoC beschädigt werden können. Wird die Stromstärke bei den 5V Pins überschritten so schaltet der RasPi automatisch ab, kann aber erst nach einer längeren Zeit wieder gestartet werden. Es dient als Schutz um den RasPi vor Überspannungen schützen.

2.3.3.3 Die 17 GPIO´s

Die 17 Pins können mit einem Eingabe- bzw. Ausgabesignal deklariert werden. Das spezielle an diesen GPIO-Pins ist, dass sie mit 3,3V oder 0V beschallt werden und jeder von ihnen kann gefahrlos 16mA an Strom beziehen kann.

Die Pins 7,11,12,13,15,16,18 und 22 sind typische Eingabe/Ausgabe Pins, die keine weiteren Funktionen beinhalten außer den Pin mit U =3,3 V oder U = 0V zu beliefern.

Die restlichen der 17 GPIOs können auch als Input/Output Pins benutzt werden, diese haben weitere Funktionen. Pin 3 und 5 können als I²C-Schnitstelle fungieren, die Pins 8 und 10 als UART und Pin 19 bis 26 (außer Pin 25) als SPI.

I²C (GPIO2 und GPIO3): Es handelt sich dabei um einen seriellen I²C-Datenbus, dass von Philips entwickelt wurde. Der Bus wurde ursprünglich für Fernsehgeräte entworfen um verschiedene Chips zu steuern. Der Raspberry Pi nutzt diese Schnittstelle zur Kommunikation zwischen verschiedenen Schaltungen. Der I²C-Bus kommuniziert über zwei Pins. GPIO2 (Pin 3) wird als SDA (seriell Data) benannt und mit diesen Pins werden die Daten seriell übertragen. GPIO3 (Pin 5) wird als SCL (seriell clock) bezeichnet, er dient dazu Takt-Impulse von bis zu 100kHz zu senden.

SPI (GPIO9, GPIO10, GPIO11, GPIO8, GPIO7): Das Serial Peripheral Interface hat Motorola entwickelt um eine schnellere Kommunikation zwischen den Transistoren zu ermöglichen. Bei dieser Schnittstelle handelt es sich dabei um einen synchronen Bus mit einer Takt- (GPIO11, SCLK), einer Sende- (GPIO10, MOSI) und einer Empfangsleitung (GPIO9, MISO)

UART (GPIO14, GPIO15): Beim Universal Asynchronous Receiver Transmitter handelt es sich ebenfalls um eine serielle Schnittstelle. Dabei kann es sich sowohl um ein eigenständiges elektronisches Bauelement oder um einen Funktionsblock eines höherintegrierten Bauteils handeln. UART dient zum Senden und Empfangen von Daten.

Weitere Details über das GPIO sind auf der Seite: *http://elinux.org/RPi_Low-level_peripherals* zu finden.

2.4 Hardwarekomponenten des Pi´s

Der Raspberry Pi wird standardmäßig ohne Eingabe- und Ausgabegeräte ausgeliefert. Das Kapitel listet eine Reihe von wichtigen Hardwarekomponenten, die der Pi benötigt um zu funktionieren.

Der Raspberry Pi hat einige Schnittstellen über die sich der RasPi erweitern lässt. Welche Komponenten sich an die CSI, DSI und den GPIO-Port anschließen lassen, werden in diesem Abschnitt mitberücksichtigt.

Tastatur und Maus

Die Logitech K400r Wireless Touch Keyboard ist eine Tastatur, mit der man den RasPi bequem bedienen kann. Sie hat die deutsche Tastaturbelegung (Nummernblock nicht vorhanden) und verfügt über einen Touchpad. Das 9 cm große integrierte Touchpad fungiert als Maus. Der Vorteil dieser Tastatur ist, dass der Wireless-USB-Dongle nur ein USB-Steckplatz belegt. Dadurch, dass der Dongle nicht mehr als 100mA benötigt, arbeitet der USB-Port fehlerfrei.

Netzteil

Modell B benötigt unter kompletter Auslastung eine Stromaufnahme von 1000mA. Dadurch, dass der Raspberry Pi nicht mehr als 1000mA an Strom aufnimmt, reicht ein Netzteil mit 5V und einer

Stromstärke von 1,2A. Dagegen benötigen das Modell B+ und Pi 2 ein Netzteil mit 1500mA – 2000mA.

SD-Karte

Um ein kompatibles Betriebssystem auf dem Raspberry Pi auszuführen, wird eine SD-Karte mit mindestens 4GB benötigt. Um eine gute Datentransferrate zu erzielen ist es empfohlen, eine SD-Karte (SDHC, SDXC) der Class 10 zu nehmen.

Video-Anschluss

In den meisten Fällen schließt man den Raspberry Pi an einen Monitor an. Hat der Monitor aber keinen HDMI-Anschluss, wird ein DVI-Adapter benötigt. Die gelbe Cinch-Buchse wird verwendet, wenn der Pi mit älteren Fernsehquellen (Röhrenfernseher) verbunden wird, die keinen HDMI-Eingang besitzen. Der Nachteil bei diesem analogen Signal ist, dass die Bildqualität stark leidet.

Gehäuse

Standardmäßig wird der Raspberry Pi ohne Gehäuse verkauft. Aus diesem Grund sollte man unbedingt über einen Erwerb dieses Schutzes nachdenken. Das Gehäuse wehrt kleine Stöße ab, und schützt auch vor einer elektrostatischen Entladung des Benutzers. Die Auswahl ist groß, es gibt sie in verschiedensten Farben und Formen, zusätzlich sind somit auch Designmöglichkeiten vorhanden. Für Anwender die viel mit dem GPIO arbeiten, empfiehlt sich ein Gehäuse mit einem passenden großen Schlitz über dem Port oder einen oben leicht abnehmbaren Deckel.

WLAN-Adapter

Ein WLAN-Stick sorgt bei der kleinen Platine für Mobilität, besonders bei Projekten in denen auf die Eingabe- und Ausgabegeräte am Pi verzichtet wird. Einige Adapter funktionieren auf dem RasPi nicht reibungslos, weil keine Treiber für Linux zur Verfügung stehen oder sie zu viel Strom vom USB-Port ziehen. Das TP-LINK Modell: TL-WN725N, Edimax EW-7811UN ist am zuverlässigsten und wird oft von den Communities verwendet. Eine weitere Liste an kompatiblen WLAN-Sticks findet sich unter *http://elinux.org/RPi_USB_Wi-Fi_Adapters*

USB-Aktiver-Hub (optional)

Der HUB muss eine eigene Stromversorgung besitzen (Aktiv-HUB), da sonst der Raspberry nicht mit ausreichend Strom (zur Erinnerung je USB-Port maximal 100mA) versorgt wird und es so zu Schwankungen zwischen der Hardware und Software kommen kann. Beim Raspberry Pi 2 und dem Modell B+ kann auf ein USB-HUB verzichtet werden, denn diese besitzen 4 USB-Ports und eine Stromaufnahme von 1500mA.

Kamera

Das Kameramodul wiegt 3 Gramm und misst 25mm x 20mm x 10mm. Das kleine Teil hat einen 5 Megapixel Sensor und kann Videoaufnahmen bis zu 1080p mit 30 fps aufnehmen, bei 720p sind 60 fps möglich. Bei der Auflösung von 640 x 480 liefert es sogar 90 Bilder pro Sekunde (geeignet für Slowmotion). Einzelbilder kann die Kamera bis zu einer Auflösung von 2592 x 1944 Pixel (5MP) schießen. Das Pi NoIR Infrarot-Kameramodul enthält denselben Bildsensor. Der Unterschied ist, dass ein Filter für IR-Wellen eingebaut ist. dadurch kann das Modul Infrarotstrahlung erfassen. Die Kamera-Module eignen sich für Zeitrafferaufnahmen, Astrofotografie, Naturbeobachtungen und für Videoüberwachungen. Die Pi Kamera wird über die CSI-Schnittstelle (Camera Serial Interface) mit einem 15-poligen Flachbandkabel angeschlossen. Der Vorteil gegenüber USB-Kameras ist, die Kommunikation findet direkt über den ARM-Prozessor statt, da eine direkte Verbindung zwischen dem Kameramodul und dem SoC besteht.

Name	Camera Module	PI NoIR Camera Module
Bildsensor	Omnivision 5647 CMOS	Omnivision 5647 CMOS mit Infrarot-Filter
Max. Pixel	5 Megapixel	5 Megapixel
Max. Auflösung	2592 x 1944	2592 x 1944
Video Auflösung	1080p mit 30 fps 720p mit 60 fps 640 x 480 mit 90 fps	1 GHz
Anschluss an Pi	15-poliges Flachbandkabel an die CSI	15-poliges Flachbandkabel an die CSI
Bild-Steuerungsfunktionen	**Belichtungsautomatik (AEC)** Automatischer Weißabgleich (AWB) **Automatischer Band Filter (ABF)** Automatische 50/60 Hz Helligkeitserkennung **Automatische Schwarzwertkalibrierung (ABLC)**	**Belichtungsautomatik (AEC)** Automatischer Weißabgleich (AWB) **Automatischer Band Filter (ABF)** Automatische 50/60 Hz Helligkeitserkennung **Automatische Schwarzwertkalibrierung (ABLC)**
Temperaturbereich	Im Betrieb: -30° bis 70°	Im Betrieb: -30° bis 70°
Linsenmaße	¼ Zoll	¼ Zoll
Maße	25mm x 20mm x 10mm	25mm x 20mm x 10mm
Gewicht	3g	3g
Kompatibel	Raspberry Pi: Modell A, B, B+, Pi 2	Raspberry Pi: Modell A, B, B+, Pi 2

Tabelle 6 : Datenangaben der Kamera-Module für den Raspberry Pi

2.5 Erweiterungen des PI

In diesem Kapitel werden nur die Hardwareerweiterungen beschrieben die in den Projekten für das Baukastensystem genutzt werden.

BrickPi (Dexter Industries)

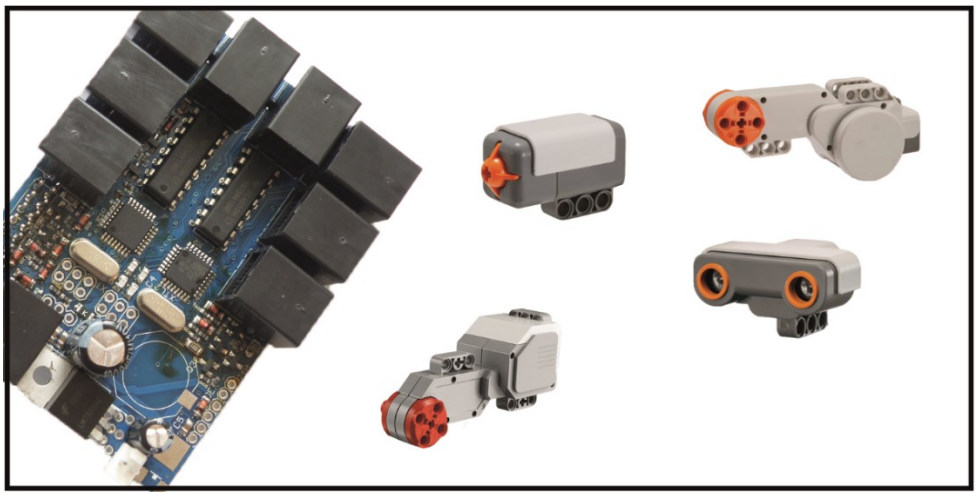


Abb7. BrickPi und Lego Motoren und Sensoren (oben Berührungssensor, unten Entfernungssensor)

BrickPi ermöglicht es zusammen mit dem RasPi die Kreativität auszuleben. Es können Roboter, Maschinen und andere Kreationen erschaffen und mit verschiedenen Programmiersprachen angesprochen und gesteuert werden.

BrickPi ist eine Art Kommunikationsbrücke zwischen dem RasPi und den LEGO Mindstorms Motoren und Sensoren. Das Board hat genau die gleiche Größe wie der Pi und wird über die 26 GPIO-Pins des RasPi´s verbunden. An den BrickPi werden bis zu 4 LEGO NXT oder EV3 Motoren angeschlossen plus fünf weitere NXT-Sensoren. Über die Raspberry Pi und den Programmiersprachen Python, C und Scratch lassen sich die Bauteile ansteuern.

Raspberry Pi übermittelt die Codierung über den GPIO-Port an den BrickPi, dieser wiederum kontrolliert und steuert die Motoren und Sensoren über zwei Mikrokontroller ATMega328s (über den I²C-Bus des Pi siehe Anhang Schemata). Damit die elektronischen Legoelemente mit ausreichend Strom versorgt werden, wird ein 9V Batteriehalter an den BrickPi angeschlossen. Das Zusatzmodul wird mit einem Gehäuse ausgeliefert das Schutz für die beiden Komponenten bietet. An diese Schutzhülle lassen sich Legobausteine montieren.

Das Board ist eine Open Source Hardware und wird von Dexter Industries gebaut. Das Unternehmen stellt die kompletten Schemata die und Programmierung des BrickPis im Netz zum Download bereit (*http://dlnmh9ip6v2uc.cloudfront.net/datasheets/Dev/RaspberryPi/BrickPi%201.7.3.pdf*).

Um die Legosensoren steuern zu können wurde für Python, Scratch oder C extra eine Bibliothek erstellt, die man ebenfalls herunterladen und installieren kann. Dexter Industries bietet zusätzlich eine vorkonfigurierte, überarbeitete Raspbian-Image als Download auf ihrer Seite an (*http://www.dexterindustries.com/BrickPi/getting-started/pi-prep/*). In der Image sind alle Bibliotheken, Beispiele bereits vorinstalliert.

PiTFT mit Touchscreen (Adafruit)

Das von Adafruit gebaute 480 x 320 Pixel große 16Bit Farb-Display mit Touchfunktion wird über den GPIO-Port angeschlossen. Adafruit bietet spezielle Kernel Versionen an, diese werden auf den vorhandenen Raspbian installiert um das Display optimal zu nutzen.

Das Display wird über die SPI-Pins (SCK, MOSI, MISO, CE0, CE1) plus zwei I/O-Pins angesprochen und gesteuert. Die restlichen Pins lassen sich weiter über eine GPIO-Verlängerung nutzen.

Als Zubehör für das Display gibt es ein praktisches PiTFT PibowGehäuse, welches dem Pi und dem Display Schutz bietet. Es bietet genügend Platz für den Zugang an die CSI- und DSI-Schnittstellen. In der vierten Schicht des Gehäuses ist es möglich, ein Durchsteckplatz für einen GPIO-Extender zu montieren.

Trellis PCB Pad - 4x4, (Adafruit)

Es ist ein 60mm x 60mm x 4mm großer Open Source LED-Pad von der Firma Adafruit. Um das Pad benutzen zu können, werden 16 x 3mm LEDs benötigt (am besten mit einem Helligkeitswert von 250mcd+), die man auf die Platine löten muss. Hinzu kommt noch ein Silikon Tastenfeld um die LEDs ein- bzw. ausschalten zu können. Die Bauanleitung des Trellis stellt Adafruit als Tutorial im Internet bzw. als PDF-Download zur freien Verfügung (*https://learn.adafruit.com/adafruit-trellis-diy-open-source-led-keypad).*

Um die LEDs auf der Platine steuern zu können benötigt man eine Steuereinheit. In diesem Fall kommt der Raspberry Pi zum Einsatz, mit dem RasPi und der Programmiersprache Python werden die einzelnen Tasten und LEDs programmiert. Um mit Python Programme für die Matrix schrieben zu können, benötigt es eine für das Pad erstellte Bibliothek, die Adafruit ebenfalls zum Download anbietet (*https://learn.adafruit.com/trellis-python-library*).

Jedes PCB verfügt über einen per I²C angebundenen LED Sequencer und Keypad-Reader. Der Chip kann alle 16 LEDs einzeln ansteuern. Das Pad besitzt drei Jumper mit denen sich die I²C-Adresse einstellen lässt und so können bis zu 8 Pads zusammen verbunden werden (7-bit Adressen zwischen 0x70-0x77).

Mit dem Trellis Keypad lassen sich z.B. Reaktionsspiele, Lichterdemos programmieren oder als Bedienungsgeld für das System nutzen.

MotoMama L298N H-Brücke

Der Raspberry Pi unterstützt viele Komponenten aus dem Repertoire von Arduino. Der Motortreiber ist eins dieser Hardwareelemente. Auf dem Treiber befindet sich ein ST L298N-Chip der die Steuerung der Gleichstrommotoren oder Schrittmotoren übernimmt.

Der Vorteil der H-Brücke ist, sie wurde so entworfen, dass man weitere Sensoren oder Module anschließen kann. Die H-Brücke muss mit einer eignen Versorgungsspannung von 5V (bis zu 2000mA) bespeist werden. (Weitere Informationen siehe Kapitel 4.5. und Anhang)

2.6 Distributionen und Software

Dank der großen Beliebtheit des Raspberry Pi, sind nach kurzer Zeit viele neue Distributionen erschienen und einige auf der Internetseite *http://www.raspberrypi.org/downloads/* zum Download bereitgestellt. Im Folgenden werden verschiedene Distributionen, wie Raspbian, Arch Linux, RisC OS, Pidora oder Mediacenters vorgestellt.

Die Begriffe Kernel, Linux und Distributionen werden erfahrungsgemäß oft verwechselt oder im falschen Zusammenhang verwendet, damit werden sollten zuerst diese Begriffe geklärt werden.

2.6.1 Kernel, Linux und Distribution

Der Kernel

Es ist der Betriebssystemkern in allen Linuxdistributionen, der für die x86, x64, ARM und viele weitere Rechnerarchitekturen verfügbar ist. Der Kernel ist der innerste Teil (der Kern) eines Betriebssystems mit ganz elementaren Funktionen, wie Speicherverwaltung, Prozessverwaltung und Steuerung der Hardware. Der Kernel ist in Schichten aufgebaut, wobei die unteren Schichten der Grundstein für die über ihnen liegenden Schichten bildet. Das bedeutet, dass die oberen Schichten auf die unteren Schichten zugreifen können aber umgekehrt funktioniert der Zugriff nicht. Dabei sei zu beachten, dass der Kernel ständig erweitert und verbessert wird und in jeder neuen Version werden z.B. Schnittstellen und Treiber für neue Hardware eingebaut

Die Basis von Raspbian ist der Linux-Kernel.

Die Schichten, aufgezählt von hardwarenah und -fern
Schnittstelle, die auf die Hardware zugreift (z.B. versch. Geräte, Speicher und Prozessoren)
Die Speicherverwaltung
Prozessverwaltung
Geräteverwaltung (Devicemanagement)
Dateisysteme

Tabelle 7 : Die Schichten eines Kernels

Die Distribution

Die Distribution ist eine Sammlung an Softwarepaketen, die mit einem Kernel zusammengefasst werden.

Ein Beispiel:

OpenELEC ist eine Distribution die nur auf dem Kernel 3.16 (Betriebssystem) aufgebaut ist. Die Distribution OpenELEC ist ein Gesamtpaket, welches aus dem Betriebssystem (Kernel) und den mitgelieferten Software-Paketen besteht. Es beinhaltet eine Sammlung an Open-Source-Programmen, die dazu dienen, den Pi in ein Mediacenter zur verwandeln. In der OpenELEC Distribution sind dies z.B. die Server-Programme (SMB, Web-Server, File-Server usw.), die Applikationen zum Abspielen der verschiedenen Multimediadateien, die Kommandos (also die Kommunikationssprache) und natürlich der XBMC/Kodi. Der Vorteil einer Distribution ist, dass sie eine einfache und bequeme Methode das System zu installieren ermöglicht. Um die Installation zu vereinfachen dienen die Systemprogramme zur Software-Installation und Hardware-Konfiguration.

Linux

Der Begriff Linux wird im alltäglichen Sprachgebrauch als Synonym für GNU/Linux-System genutzt. Tatsächlich bezeichnet es aber nur das Betriebssystem (den Kernel) und nicht die Distribution.

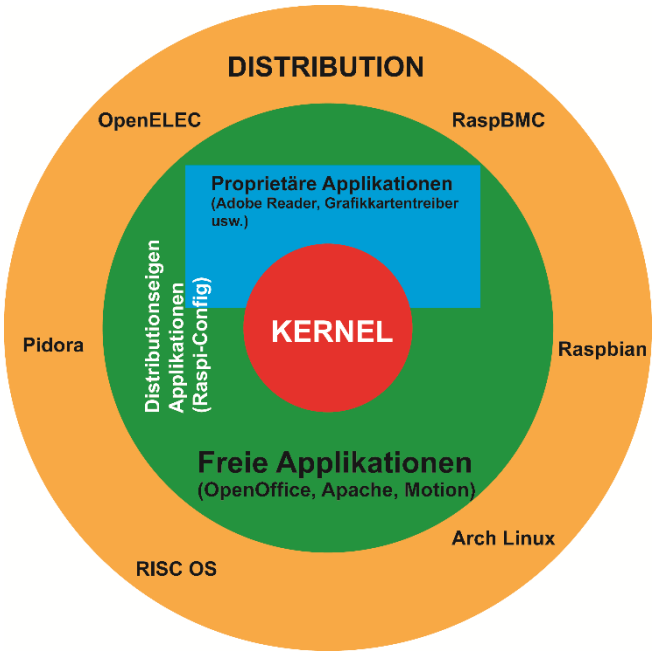


Abb8. Graphische Darstellung, was unter einer Distribution zu verstehen ist

2.6.2 Eine Auswahl an Distributionen

Raspbian

Raspbian ist eine der beliebtesten Distributionen für den RasPi. In Wirklichkeit handelt es sich um Debian7, dass extra auf den ARM-Prozessor zugeschnitten wurde. Der einzige Unterschied zum originalen Debian ist, dass Raspbian mit dem neueren Kernel ausgestattet worden ist. Raspbian bringt dadurch einen großen Vorteil gegenüber anderen Distributionen. Viele Anwendungen lassen

sich aus der Debian-Sammlung auf dem Pi verwenden und ohne Probleme installieren. Auf über 35.000 Pakete kann der Raspbian aus der Debian-Quelle zugreifen. Es beinhaltet eine grafische Oberfläche (LXDE) mit einigen vorinstallierten Programmen (z.B. Web-Browser Midori, Scratch, Pyhton usw.). Raspbian ist mit vielen HDMI- und USB-Treibern für verschiedene Geräte ausgestattet. Das Betriebssystem ist für Einsteiger die perfekte Lösung, da vieles für den Benutzer vereinfacht wurde (siehe Raspi-Config). Jedoch greifen auch fortgeschrittene Linux-Benutzer gerne auf diese Distribution zurück.

Anhand der Respbian Distribution werden im weiteren Kapitelverlauf die Funktionen eines Linux-Systems beschrieben. Dabei wird die Distribution ausführlicher unter die Lupe genommen Der Bootvorgang wird näher erläutert und die Struktur des Systems aufgezeigt.

Pidora

Bei Pidora handelt es sich um eine durch die Comunity angepasste Version von Fedora des Red Hat Linux. Sie sollte anfangs die Standarddistribution für den RasPi werden. Doch die ersten Versionen waren langsam, fehlerhaft und instabil. Deswegen konzentrierte man sich auf Raspbian. Inzwischen sind die Fehler behoben und Pidora läuft auf dem Raspberry Pi stabil. Pidora besitzt ebenfalls eine grafische Oberfläche und inzwischen stehen fast alle ARM-Paketquellen Pidora zur Verfügung. Wenn man Multimedia-Projekte auf dem OS durchführen will, wird man auf Probleme stoßen, denn es fehlen einige wichtige Audio- und Video-Codecs.

Arch Linux

Zeichnet sich durch Tempo beim Booten aus, allerdings verzichtet Arch Linux auf die grafische Oberfläche und automatisierte Installationen. Es startet direkt in der Kommandozeile. Nachinstallation einer grafische Oberfläche ist mit dem GUI-Paketmanager Pacman bei Bedarf möglich. Für Einsteiger ist das System nicht empfehlenswert.

RISC OS

Die Wurzeln dieses Systems reichen bis in die 80er Jahre zurück. Das britische Unternehmen entwickelte in dieser Zeit einen 32-Bit-Computer namens Archimedes. Dieser Computer besaß ebenfalls einen ARM-Prozessor. RISC OS besitzt eine grafische Oberfläche, die mehr an die alten Zeiten von Windows 3.1 erinnern. RISC OS bootet schnell und läuft sehr stabil. Der einzige Unterschied zu den anderen Systemen ist, dass das OS sehr viel Gebrauch von Drag & Drop macht

.

Distribution	RASPBIAN	PIDORA	RISC OS	ARCH LINUX
Basierend auf:	Debian 7	Fedora	RISC OS – keine Linux Distribution	ARM-Linux
Kernel	3.18	3.18		3.18
Grafische Oberfläche	LXDE	XFCE	RISC OS Desktop	Keine, kann nachinstalliert werden
Software-Pakete	35.000 Pakete von Debian	Fast alle Fedora Pakete	Nicht alle zum Pi kompatibel: Installation über Paketmanager: Store, Packman	k.a.
Zugangsdaten	Name: pi Passwort: raspberry	Wird in der Erstkonfiguration erstellt	Können eingestellt werden	Passwort: root
Konfiguration	Raspi-Config vereinfacht die Grundkonfiguration von Raspbian	Erstkonfiguration über grafische Oberfläche	manuell	Automatisierend
Kompatibel mit	Mit allen Raspberry Pi´s	Mit allen Raspberry Pi´s	Mit allen Raspberry Pi´s	Mit allen Raspberry Pi´s

Tabelle 8 Die vier wichtigsten Distributionen für den Raspberry Pi

2.6.3 Kodi-Distribution

In diesem Abschnitt steht die Distribution OpenELEC im Vordergrund, die als Mediacenter auf dem Raspberry Pi fungiert. Dank des hardwarebeschleunigten Grafikchips, der sich im SoC-Chip befindet, lassen sich Full-HD-Videos ohne gravierende Aussetzer abspielen. Durch die Anschlüsse (HDMI, RCA-Video-Out und der 3,5mm Klinken-Buchse) lässt sich der Pi an einen Fernseher und einer HiFi-Anlage anschließen. Eine weitere Distribution die sich auf dem Pi durchgesetzt hat ist RaspBMC. Die beiden Systeme unterscheiden sich kaum voneinander, der einzige große Unterschied ist, dass die RaspBMC Distribution auf der Basis von Debian aufgebaut wurde. Dagegen ist OpenELEC nur auf dem Linux Kernel konstruiert worden (näheres in diesem Kapitel). OpenELEC hat einen großen Vorteil, da dieser sparsam mit den CPU-Ressourcen umgeht. Es ist das kleinste System, dadurch steht sehr viel Speicherplatz auf der SD-Karte für Filmdatenbanken, Bilder und Add-ons zur Verfügung.

Nach dem booten führen alle Systeme die grafische Oberfläche von XBMC aus. Es bedeutet **X**box **M**edia **C**enter und wurde zuerst für die modifizierte Xbox entwickelt, später dann für PC- und andere Systeme, wie z.B. Android und Raspberry Pi. Ab der nächsten Version, die sich zur Zeit in der Alpha-Version befindet [Stand:13.10.2014], wird XMBC in Kodi umbenannt. Die Gründe sind naheliegend, das für die Xbox entwickelte Mediacenter läuft nicht mehr auf der alten „Xbox 360“ und der neuen „XBOX One“. Dafür läuft XBMC auf fast jedem beliebigen System, wie Desktop Computer, Smartphone, Tablet (beide Android) usw. Ein weiterer sehr wichtiger Grund ist: *„Dadurch, dass sich*

dieser von dem der Microsoft-Spielkonsole ableitete, hatten die Entwickler nie die Kontrolle über die Namensrechte und mussten sogar damit rechnen, von anderen Firmen abgemahnt zu werden."[16]

XMBC bzw. Kodi ist eine Allround-Abspielsoftware für Musik, Videos und Bilder. Diese kann nicht nur die Multimediadateien von der Festplatte abspielen, sondern auch aus dem lokalen Netzwerk (z.B. NAS) oder durch Plug-Ins direkt aus dem Internet streamen.

OpenELEC (siehe 2.6 Kodi-Distribution)

Wie oben schon beschrieben, ist OpenELEC (**Open** **E**mbedded **L**inux **E**ntertainment **C**enter) ein eigenes System, das auf keiner anderen Distributionen aufgebaut.

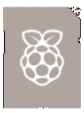
RaspBMC (wurde in OSMC umbenannt)

RaspBMC ist hingegen langsamer beim Booten und verfügt über ein benutzerfreundlicheres Konfigurationsmenü. Auch das Nachinstallieren verschiedener Raspbian-Pakete fällt beim RaspBMC einfacher aus. Die neuste Version heißt jetzt OSMC (Open Source Media Center).

Distribution	OpenELEC	RaspBMC (OSMC)
Basierend auf:	Linux Kernel	Debian
Kernel	3.18	3.18
Grafische Oberfläche	Kodi	Kodi
Software-Pakete	Plug-Ins und Addons	Fast alle Fedora Pakete
Zugangsdaten	SSH Zugang: Name: root Passwort: openelec	Name: osmc Passwort: osmc
Konfiguration	Wird beim esrsten Start ausgeführt. Unter Einstellungen ist weiter Konfiguration möglich.	Wird beim esrsten Start ausgeführt. Unter Einstellungen ist weiter Konfiguration möglich.
Kompatibel mit	Mit allen Raspberry Pi´s	Mit allen Raspberry Pi´s

Tabelle 9 Die zwei wichtigsten Mediacenter-Distributionen für den Raspberry Pi

2.6.4 NOOBS



NOOBS

NOOB (New Out Of the Box Software) ist ein Installationsmanager, der eine Auswahl an Distributionen enthält. Nach dem Start kann man zwischen Raspbian, Arch, OpenELEC, Pidora, Risc

[16] http://www.heise.de/newsticker/meldung/Aus-XBMC-wird-das-Kodi-Entertainment-Center-2282292.html

OS, RaspBMC und der grafischen Programmiersprache SCRATCH wählen. Die Distributionen befinden sich alle im OS Ordner des Installationsmanagers. Die Iso-Datei steht in zwei Varianten zum Download bereit. Die NOOBS LITE Image ist knapp 20 MB groß und bezieht die Daten der ausgewählten Distributionen aus dem Internet. In der normalen NOOBS-Verzeichnisstruktur sind alle Distributionen offline enthalten und stehen zur Installation bereit. NOOBS nimmt dem Benutzer einige Aufgaben ab und erleichtert die Installation des ausgewählten Systems.

Beim ersten booten formatiert NOOBS die SD-Karte in entsprechende Datensysteme (siehe Kapitel 3.3 NOOBS: Eine benutzerdefinierte Distribution erstellen) und danach kann der Anwender wählen, welches Betriebssystem er installieren möchte. Der Vorteil des Installationsmanagers ist, dass sich alle Distributionen auf einer SD-Karte befinden. Diese können jederzeit deinstalliert werden und andere Betriebssysteme zu installieren. NOOBS hat aber noch einige weitere Funktionen parat. Der Anwender hat die Möglichkeit schon vorab die Sprache und das Tastatur-Layout einzustellen, den Display Modus zu ändern und die Konfigurations-Dateien (cmdline.txt und config.txt) auf der SD-Karte mit Hilfe eines Editors zu konfigurieren. Nach der Installation des Systems startet die installierte Distribution. Um nochmal in den Installations-Manager zu gelangen, wird beim nächsten Booten die SHIFT-Taste gedrückt. Eine ausführliche Beschreibung findet sich unter: https://github.com/raspberrypi/noobs/blob/master/README.md

2.6.5 ArkOS

ArkOS ist eine Plattform zum sicheren Verwalten der eigenen Daten (Kalender, Kontakte, Bilder, Videos, etc.). Dies geschieht über eine Cloud im eigenen Netzwerk. Die eigenen Daten werden nicht an Cloud-Anbieter, wie z.B. Google, Dropbox weiter gegeben. Die Distribution ArkOS wurde aus Linux-Werkzeugen für den Serverbereich zusammengestellt. Es enthält verschiedene Serverdienste und unterstützt Webseiten, Wordpress und Blogs sowie die Datenablage mittels ownCloud. Des Weiteren einen Mailserver, Konten von Social Networks und weitere Cloud-Dienste. ArkOS basiert auf Arch Linux als Unterbau und dem in Python programmierten Web-Frontend-Genesis, der über Plug-Ins funktioniert. Mit Hilfe der Plug-Ins werden z.B. bei der Installation der ownCloud alle Konfigurationen nach den Angaben des Anwenders automatisch durchgeführt.

2.7 Entwicklungssoftware

Mjpg-Streamer

Das Programm wird über die Kommandozeile gesteuert und ist ursprünglich für Debian und Ubuntu entwickelt worden. Das Paket nutzt den „video4linux“ Treiber, dadurch läuft der Stream recht flüssig und weist kaum Latenzzeiten auf. Die Aufzeichnungen werden mit dem RasPi-Kameramodul temporär aufgenommen und über einen eignen Server auf eine Internetseite gestreamt. Das Programm eignet sich prima zur Überwachung und Fernsteuerung von Fahrzeugen usw.

WebIOPI

Ist eine in Python geschriebene Web-Applikation, mit der der Nutzer über einen Internet-Browser die GPIO-Pins steuern kann. Die Pins können auf Input oder Output gesetzt, sie ein- oder ausschaltet werden und den aktuellen Zustand kann ermitteln werden. Dadurch, dass WebIOPI die REST.API (Representational State Transfer, bezeichnet ein Programmierparadigma für Webanwendungen) verwendet, sind die Webseiten auch Smartphone optimiert. Das Programm erlaubt sehr schnell und einfach eigene Webinterfaces zu gestalten und arbeitet optimal mit Python und JavaScript zusammen.

Auf der Homepage von WebIOPI (*https://code.google.com/p/webiopi/*) sind sehr ausführliche Dokumentationen und Beispiele zu finden. Dabei sind für das Baukastensystem drei dieser Themen von besonderer Bedeutung.

- Einführung in das Framework (zur Steuerung des GPIOs)
- Die beiden Clients (HTML, Python und Java)
- Verwendung von REST.API

Die Vielzahl der Anwendungsmöglichkeiten durch den RasPi und WebIOPI ist groß. Es beherrscht die kompletten Funktionen von Serial/SPI und I²C Schnittstellen und es wurden über 30 verschiedene Ein- / Ausgabegeräte, Sensoren und Analogkonverter in die Bibliotheken eingefügt. Haussteuerung und Robotik lassen sich damit prima umsetzen. Ob Licht, LEDs, Temperatur, Heizung, Jalousien usw. Mit WebIOPI lässt sich dank der grafischen HTML- und Java-Unterstützung alles besser kontrollieren.

Im Baukastensystem wird die Applikation für ein ferngesteuertes Fahrzeug mit einer Kamera verwendet, die über das Webinterface gesteuert wird.

2.8 Python

Python ist eine Programmiersprache, die einfach zu verstehen ist und schnell erlernbar scheint. Da der Raspberry Pi eine Lernplatine ist, haben sich die Entwickler von Raspbian entschieden diese leichte Programmiersprache in ihre Distribution mit zu integrieren. Bevor die Funktionsweise und die Anwendungsmöglichkeiten für das Baukastensystem beleuchtet werden, wird die Entstehungsgeschichte des Python beschrieben. Anschließend werden die grundlegenden Konzepte auf denen Python aufbaut, erläutert.

2.8.1 Entstehungsgeschichte von Python

Ursprünglich sollte Python als Skriptsprache für das verteilte Betriebssystem Amoeba dienen. Die Sprache entstand in den 90er Jahren und wurde in den Niederlanden an der „Centrum voor Wiskunde en Informatica“ von Guido van Rossum entwickelt.

Guido van Rossum ist auch der Mitentwickler der ABC-Programmiersprache. Das Ziel dieser war, Personen die sich noch nie mit einer Programmiersprache auseinander gesetzt haben, die ABC-Programmiersprache sehr schnell und einfach beibringen können. Die Erfahrungen und Kritiken an der ABC-Sprache nutzte van Rossum für die Entwicklung von Python. Er erschuf damit eine mächtige und einfach zu erlernende Programmiersprache, die auf allen Plattformen läuft.

1996 erklärte van Rossum, wieso er sich für den Namen Python entschieden hat: *„Vor über sechs Jahren, im Dezember 1989, suchte ich nach einem 'Hobby'-Programmier-Projekt, dass mich über die Woche um Weihnachten beschäftigen konnte. Mein Büro ... war zwar geschlossen, aber ich hatte einen PC und sonst nichts vor. Ich entschloss mich einen Interpreter für die neue Sripting-Sprache zu schreiben, über die ich in der letzten Zeit nachgedacht hatte: ein Abkömmling von ABC, der UNIX/C-Hackern gefallen würde. Python hatte ich als Arbeitstitel für das Projekt gewählt, weil ich in einer leicht respektlosen Stimmung war (und ein großer Fan von Monty Python's Flying Circus)."*[17]

[17] http://www.python-kurs.eu/entstehung_python.php

2008 wurde Python von Grund auf überarbeitet und erschien in einer 3.0 Version. In den USA und einigen europäischen Ländern werden jährlich PyCon-Konferenzen durchgeführt. Die nächste PyCon findet in Montréal statt: *https://us.pycon.org/2015/*

2.8.2 Konzepte und Lieferumfang

Python hat gegenüber vielen anderen Sprachen einen Vorteil: Es ist plattformunabhängig. Der Interpreter von Python läuft auf verschiedenen Systemen und ermöglicht damit gleiche Programm fehlerfrei auf anderen Systemen auszuführen.

Bei Python handelt es sich im größten Sinne um eine imperative Programmiersprache, die aber auch objektorientiert oder funktional eingesetzt werden kann. Sie verbindet mehrere Sprachelemente, die aus der Skriptsprache bekannt sind. In vielen anderen Sprachen wie C++ und C muss der geschriebene Quellcode zuerst kompiliert werden, *„d.h. jedes Programm muss zuerst in Maschinencode übersetzt werden, bevor es ausgeführt werden kann“*[18]. Bei Python wird der Quellcode auch mit einem Compiler in Byte-Code kompiliert, wird danach aber interpretiert (in Maschinencode) und direkt ausgegeben. Der Unterschied zu anderen Programmiersprachen ist, dass der Byte-Code in Befehle für eine Virtuelle Maschine kompiliert wird (PVM = Python Virtual Machine).

„Dieser Code ist unabhängig von realer Hardware, aber kommt dem Code von Maschinensprachen schon relativ nahe. Dieser Bytecode wird dann für einen bestimmten Maschinencode interpretiert. Das bedeutet, dass der Bytecode plattformunabhängig ist.“[19]

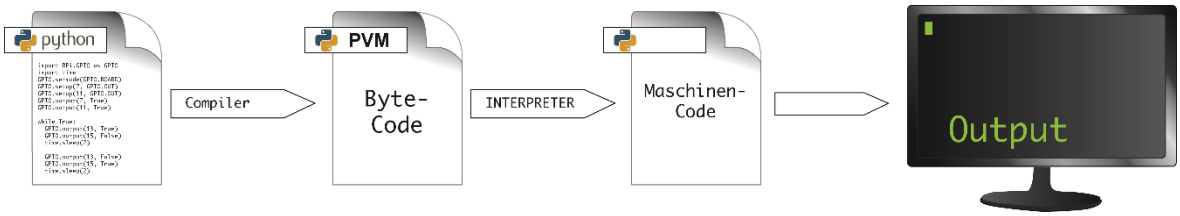


Abb9. Der Python Interpreter

Die Standardbibliotheken die Python mit sich bringt, ermöglichen dem Benutzer gut strukturierte und einfache Programme zu schreiben, die komplexe Aufgaben erledigen können. Diese Standardbibliotheken können ohne weiteres mit eigenen oder von Drittanbietern erstellten Bibliotheken problemlos erweitert werden. Ein großer Vorteil von der leicht erlernbaren Programmiersprache ist, dass sie gut mit anderen Programmier- und Skriptsprachen zusammenarbeiten kann.

2.8.3 Einsatzmöglichkeiten von Python

Dadurch das Python einfach zu verstehen, aber trotzdem sehr umfangreich und erweiterbar ist, setzen viele bekannte Unternehmen auf die freie Programmiersprache. Das Unternehmen, welches am meisten von Python profitiert ist „Google“ und auch seine Tochterfirma „Youtube“. Python findet sich ohne zu wissen schon in vielen Mobiltelefonen, Tablets oder im Internet.

---

[18] Einführung in Python 3, Bernd Klein 2013, S.37
[19] Ebenda S.37

Mittlerweile gibt es schon sehr viele Unternehmen, die Python anwenden. Unter dem Link sind viele bekannte Firmen aus allen möglichen Kategorien, die Python nutzen, zu finden: https://wiki.python.org/moin/OrganizationsUsingPython.

2.8.4 Einsatz von Python im Baukastensystem

Das Baukastensystem profitiert in vielen Projekten von der Programmiersprache Python. In erster Linie wird Python genutzt, um die GPIO-Grundlagen zu erläutern. Es werden Schritt für Schritt Programme entwickelt, in denen die GPiO-Pins konfiguriert und anschließend über den Code gesteuert werden. Mit nur ein paar Zeilen lassen sich LEDs, ein Taster oder ein DC-Motor steuern. Mit dem Einsatz des Trellis LED-Keypad werden neue Bibliotheken von Dritten eingesetzt und Minispiele mit dem Tastenfeld programmiert. Auch BrickPi bringt eine eigene Python-Bibliothek mit sich, die erlaubt die Lego-Motoren und Sensoren zu steuern.

2.8.5 Bibliotheken

Um in Python ein Programm schreiben zu können, werden Bibliotheken, die Funktionen und Daten-Typen zur Verfügung stellen, benötigt. Dabei gibt es die Standardbibliotheken, die Python mit sich bringt (time, os usw.), selbsterstellte Bibliotheken und Module von Drittanbietern. Der Vorteil von Modulen ist, Python lädt beim Ausführen des Programms nur die angegebenen Libraries oder nur einzelne Teile (Beispiel: from random import randint).

„Ein Modul kann unabhängig vom Gesamtsystem erzeugt und separat getestet werden. In den meisten Fällen kann man ein Modul auch in anderen Systemen verwenden.“ [20]

Im Baukastensystem werden nicht nur die Standardbibliotheken genutzt, sondern auch Module von Drittanbietern. Diese wurden extra für die Hardwarekomponenten geschrieben, um eine reibungslose Kommunikation mit dem Raspberry zu garantieren. Der BrickPi z.B. nutzt eine eigene Bibliothek, die von den Entwicklern des Boards entworfen wurde. Mit dem Modul BrickPi, werden die einzelnen Schnittstellen am BrickPi über den GPIO-Port angesprochen.

Bei dem Spiel „Finger Tipp“ wird die I²C-Schnittstelle des Raspberry Pi´s benötigt. Um ein Programm für die LED-Touch-Platine zu schreiben, werden zwei Bibliotheken installiert. Bei der einen handelt es sich um das Adafruit_I2C Modul, welches Python die richtige Kommunikation für die Platine garantiert. Um aber das Keypad zu steuern, muss sich das Modul immer in demselben Verzeichnis befinden, wo sich das geschriebene Programm befindet. Die Bibliothek, die die LEDs und Buttons steuert, heißt Adafruit Trellis.

Die meisten Module werden im /Python.x.y/Lib/ oder /usr/lib/Python.x.y abgelegt und werden nur durch

```
import MODULNAME
```

in Python aufgerufen.

[20] Ebenda S.101

2.8.6 Struktur eines Python Codes

Der Aufbau eines Pythons Codes zeigt, wie einfach und verständlich dieser strukturiert ist. Dadurch schafft Python einen besseren Überblick über den Quellcode und deren Bausteine.

Das Beispiel zeigt einen aufgeräumten und gut strukturierten Sourcecode:

```
#!/usr/bin/python
```

Übermittelt dem Interpreter, dass es sich um ein Python 2.x Code handelt.

Als nächstes werden die benötigten Bibliotheken (Module) importiert

```
import RPi.GPIO as GPIO
```

Importiert das GPIO-Modul für die GPIO-Steuerung als GPIO.

```
import time
```

Importiert das Zeit-Modul für Zeitabläufe.

```
import os
```

Importiert das Systems-Modul für Shell-Befehle im Code.

Setmode teilt dem Code mit welche GPIO-Pins-Nummerierung genutzt wird.

```
GPIO.setmode(GPIO.BOARD)
```

Dabei stehen zwei Varianten zur Verfügung:

Variante 1: Bei BOARD zählt man die Pins von links nach rechts.

Variante 2: BCM nutzt die genaue Bezeichnung der Pins (siehe Kapitel 2.3.3.1 Rev.1 und Rev.2)

```
GPIO.setwarnings(False)
```

Die Warnungen, die der Debug-Modus aufzeigt, werden abgeschaltet.

```
LED = 12
```

Die LED wird an Pin zwölf angeschlossen, bei mehreren Bauelementen ist es ratsam diese Pins mit den elektrischen Bauelementen zu benennen.

Da es sich bei einer LED um ein Ausgabeelement handelt, wird diese als GPIO.OUT deklariert.

```
GPIO.setup(LED, GPIO.OUT)
```

Zugriff auf die Shell-Befehle von Raspbian, der Bildschirm wird von jeglichem Text gereinigt.

```
os.system („clear“)
```

Nachdem alle Komponenten und Variablen konfiguriert sind, beginnt der eigentliche Code zur Steuerung der LED.

```
#Solange die Bedingung wahr ist wird die while-Schleife ausgeführt
while True:
      try:
            print „LED AN“ #Schreibt auf den Bildschirm
            GPIO.output(LED, GPIO.HIGH) #LED PIN U = 3,3V
            time.sleep(1) #Wartet die angegebene Zeit ab
            print“Licht AUS“
            GPIO.output(LED, GPIO.LOW) #LED PIN U = 0V
            time.sleep(1)
      #Mit STRG-C wird das Programm abgebrochen bzw. beendet
      except KeyboardInterrupt:
```

```
        print"Ende"
        GPIO.cleanup() #Setzt die GPIO-Pins wieder auf null
        quit()     #Erst dann wird der Skript sauber geschlossen
```

An diesem Beispiel wird ersichtlich, wie Python strukturiert ist. Im ersten Schritt werden alle benötigten Bibliotheken importiert, der GPIO-Port konfiguriert und die Variablen festgelegt. Nachdem die Voreinstellungen erledigt sind, beginnt der eigentliche Code für die LED-Steuerung.

2.9 Raspbian erster Start

Die Installation der Distribution auf den RasPi funktioniert nicht auf dieselbe Art und Weise, die von dem Desktop-Computer bekannt ist. Die Distribution wird nicht installiert, sondern ein Image wird mit Hilfe eines Tools (z.B. win32diskimager) auf eine SD-Karte (Ausnahme NOOBS) kopiert.

In fast allen Modulen wird die Raspbian Distribution verwendet. Sie ist leicht in der Bedienung und erleichtert die Konfiguration, des Weiteren sind nötige Entwicklungsprogramme vorinstalliert.

In diesem Kapitel wird der Bootvorgang anhand von Raspbian beschrieben, der sich sehr von einem Bootvorgang eines BIOS-Rechners unterscheidet.

Die Raspbian-Distribution ist für Einsteiger in Linux sehr benutzerfreundlich entwickelt worden. Nachdem die ganzen Vor-Konfigurationen, Treiber und Software geladen ist, meldet sich nach dem ersten Start das kleine Konfigurations-Tool RasPi-Config. Das Tool ist ein für Raspbian-Distribution entwickeltes Programm, das gegenüber anderen Distributionen die Möglichkeit bietet, einige Einstellungen für das System vorzunehmen. In Raspi-Config legt man die Sprache, Tastatur, Passwort usw. fest. In diesem Kapitel wird beschrieben, welche für Einstellungen vorgenommen werden können und was diese bewirken. Im weiteren Verlauf werden Updates und Upgrades durchgeführt. Das Dateisystem wird im Kapitel 5.3 näher erläutert. Da die Shell für Navigation und Ausführung der Befehle in einem Linux-OS zuständig ist sowie zur Funktion der Kommandozeile, wird hierauf in Kapitel 5.4 näher eingegangen. Am Ende werden die Benutzerverwaltung und wichtige Hauptfunktionen von Linux zusammengefasst.

2.9.1 Der Bootvorgang

Die Zentrale des Raspberry Pi´s ist der SoC, in dem die CPU, GPU und der RAM-Speicher in einem Chip vereint sind. Dieses Detail ist sehr wichtig, da der RasPi nicht über ein Bios verfügt. Der Raspberry Pi bootet über die GPU des SoCs. Wird der Raspberry Pi mit Strom versorgt, so ist die CPU/SDRAM inaktiv und nur die GPU-Einheit wird aktiviert.

Erste Stufe des Bootvorgangs:

Ein Teil des Bootloaders, der sich auf dem SoC befindet, wird aus der GPU geladen und mountet die FAT32 Partition auf der SD-Karte. Der Teil des Bootloaders, der sich auf dem SoC befindet, ist fest in den Chip eingespeist und kann nicht verändert werden. „*Ein kleiner dedizierter RISC Core auf der Raspberry Pi GPU startet diesen Prozess*“[21]. (siehe Schema Kapitel SoC)

[21] Hacks für Raspberry Pi, R. Suehle & T. Callaway 2014, S.9

Zweite Stufe des Bootvorgangs:

Dieser greift auf „bootcode.bin“, der sich auf der ersten Partition der SD-Karte befindet und lädt diesen in den L2-Cache des SoC. Der „bootcode.bin“ wird ausgeführt und aktiviert den SDRAM. Dadurch wird die dritte Stufe eingeleitet.

Dritte Stufe des Bootvorgangs:

Der Bootloader lädt von der SD-Karte die Firmware der GPU („start.elf“) in den Arbeitsspeicher. Das darin enthaltene Programm lädt die Einstellungen der config.txt und der cmdline.txt. Die config.txt beinhaltet GPU-Grundeinstellungen und in der cmdline.txt sind zusätzliche Kernel Parameter gespeichert. Anschließend wird der kernel.img in den Arbeitsspeicher geladen. Als letztes startet die GPU den ARM-Prozessor und der Linux-Kernel bootet das System zu Ende.

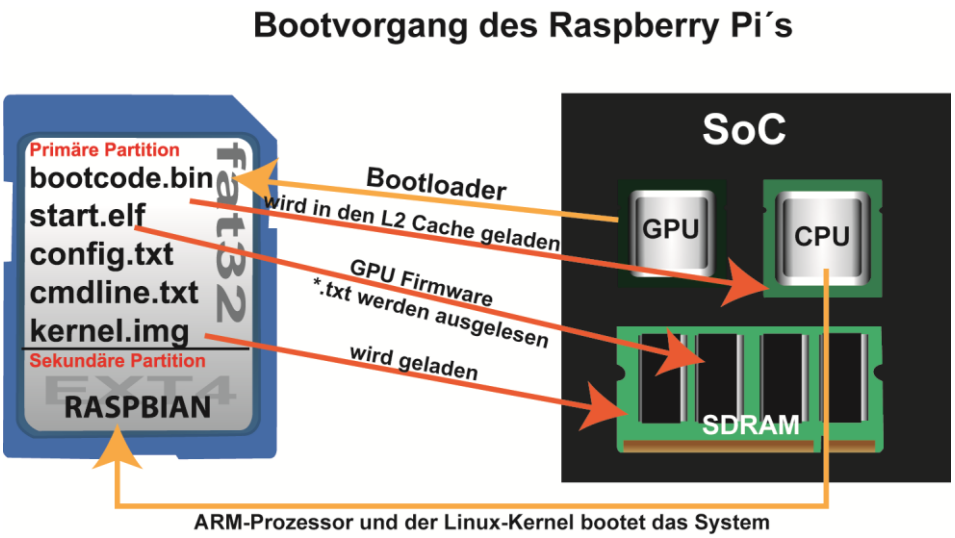


Abb10. Bootvorgang des Raspberry Pi´s

2.9.2 Raspi-Config

Beim allerersten booten von Raspbian wird die Raspi-Config gestartet. Diese wurde extra für die Raspbian Distribution entwickelt, um die Konfiguration des System für den Benutzer zu vereinfachen. Dabei handelt es sich um ein Skript, das eine grafische Bedienoberfläche hat. Die einzelnen Menüpunkte greifen bei Änderungen auf verschiedene Systemdateien und verändern diese nach Bestätigung.

Das Tool wird ausführlich erläutert, weil die meisten Einstellungen für das Baukastensystem relevant sind, wie z.B. der Hostname des Pi´s oder das Ein- bzw. Ausschalten der Kamera.

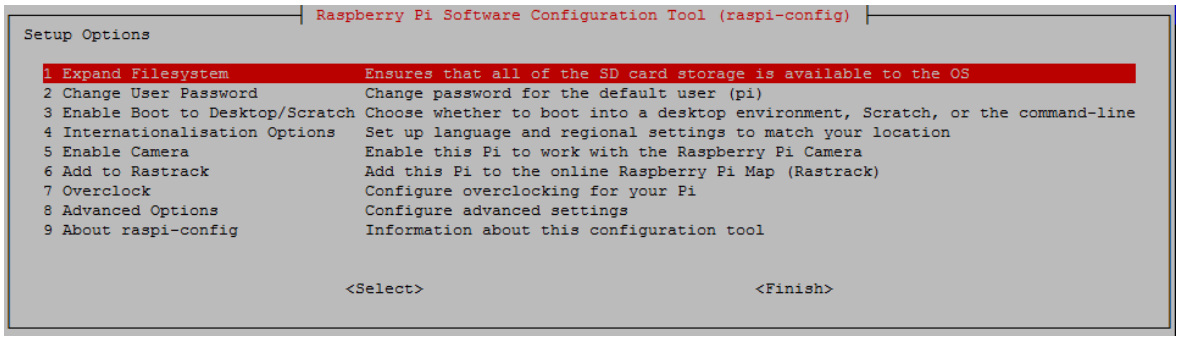


Abb11. Raspi-config vereinfacht Veränderungen im System

01 Komplette Größe der SD-Karte verwenden

Expand Filesystem: Raspbian nutzt standardmäßig nur ein Bruchteil der SD-Karte aus, dabei bleibt der restliche Speicher unberührt. Um die ganze physikalische Größe der Partition zuzuweisen, muss der **Expand**-Befehl ausgeführt werden. Nach einem Neustart steht der neue Speicher zur Verfügung.

```
Root partition has been resized.
The filesystem will be enlarged upon the next reboot
```

Abb12. Mit dem Befehl expand wurde die Root-Partition auf dem gesamten SD-Karten-Speicher übergeben.

Wird Raspbian über NOOBS installiert, kann diese Einstellung übersprungen werden, da Raspbian automatisch eine bestimmte Größe (je nach Konfiguration siehe NOOBS) zugewiesen bekommt.

02 Passwort ändern

Raspbian hat einen vorkonfigurierten Benutzer, der Standarduser lautet „pi“ und das Passwort „raspbbery“. Für Projekte mit Zugriff aus der Ferne über das Internet ist das Standardpasswort eine unsichere Variante. Somit ist es ratsam dies zu ändern. Unter diesem Menüpunkt kann nur das Passwort geändert werden, jedoch nicht der Benutzer. Um Benutzer zu verwalten zu können führt man dies über die Kommandozeile durch (Siehe Benutzer hinzufügen).

```
Enter new UNIX password:
```

Abb13. Raspbian fragt nach neuem Passwort; Standartpasswort: raspberry

03 Desktop / Scratch deaktivieren bzw. aktivieren

Raspbian stellt drei verschiedene Varianten vor, die nach dem Booten ausgeführt werden können.

In der **Textkonsole (Terminal)**, die stark an DOS erinnert, werden alle Befehle in der Kommandozeile eingetippt. „Desktop Log…“ ist eine LXDE **Desktopoberfläche**, an die jeder Windows-Benutzer und Mac-User gewohnt ist. Die dritte Wahlmöglichkeit ist Scratch. Es ist eine Programmiersprache, die eine grafische Oberfläche mit sich bringt. Diese Sprache richtet sich an alle, die alt genug sind mit Maus und Tastatur umzugehen.

```
 Chose boot option

Console Text console, requiring login (default)
Desktop Log in as user 'pi' at the graphical desktop
Scratch Start the Scratch programming environment upon boot
```

Abb14. Standard-, Desktopoberfläche oder direkt in die Programmieroberfläche Scratch booten

04 Sprachregion, die Zeitzone und das Tastatur-Layout einstellen

```
I1 Change Locale          Set up language and regional settings to match your location
I2 Change Timezone        Set up timezone to match your location
I3 Change Keyboard Layout Set the keyboard layout to match your keyboard
```

Abb15. Unter Internationalisation Options wird die Sprache, Zeitzone und Tastaturlayout umstellen.

Unter **I3 Change Keyboard Layout** werden der Tastaturtreiber bzw. Hersteller, der Zeichensatz für die Tastaturbelegung (Keyboard-Layout) und Tastenkombination für den S-Server festgelegt.

I1 Change Local: Im Menü „Change Local“ wird die Systemsprache eingestellt. Für Deutschland sollte **de_DE.UTF-8 UTF8** ausgewählt werden. Davon sind die Tastaturbelegungen (Umlaute), die Währungseinstellungen (Euro) und die Uhrzeitangabe betroffen.

Mit **I2 Change Timezone** wird der Kontinent und die Zeitzone ausgewählt, damit werden Probleme der Synchronisation und Übertragung der Paketdaten minimiert.

05 Raspberry Kameramodul aktivieren / deaktivieren

Enable Camera: Dient zur Aktivierung/Deaktivierung des Kameramoduls bzw. des Pi NoIR Moduls an der CSI-Schnittstelle. Die GPU stellt mindestens 128MB des RAM-Speichers für die Kamera bereit, zusätzlich wird Firmware für die Kamera geladen.

06 Add to Rastrack

Rastrack ist eine Internetseite, die registrierte Raspberry Pis auf einer Weltkarte anzeigt. Dieser Dienst ist freiwillig und kostenlos. (Dazu muss der Raspberry sich im Internet befinden). Nach der Aktivierung wird auf der Weltkarte der ungefähre Standort des Raspberry Pis angezeigt.

07 RasPi übertakten

Overclock: In diesem Menü lässt sich der Raspberry Pi auf sichere Art und Weise übertakten.

08 Einstellungen für Experten

A1 Overscan deaktivieren

Overscan: Dieser Dienst ist für ältere Bildschirm- / Fernsehgeräte gedacht. Wird der Raspberry Pi an einem Röhrengerät angeschlossen, kann es unter Umständen Sinn machen, den Overscan zu aktivieren. Dabei werden bestimmte Bildbereiche nicht angesprochen und ein schwarzer Rahmen wird um das Bild projiziert.

A2 Hostname für den Pi festlegen

Hostname: Dem Raspberry Pi kann man einen individuellen Namen geben, der im Netzwerk angezeigt wird. Der Name kann als lokale Domäne im Netzwerk genutzt werden.

A3 Memory Split richtig nutzen

Memory Split: Der RasPi (Modell B und B+) besitzt 512MB Speicher, der gemeinsam von der CPU und GPU genutzt wird. Unter Raspbian ist die Aufteilung von Arbeitsspeicher und Grafikspeicher standardmäßig auf 448MB / 64MB voreingestellt. Die Möglichkeit besteht zwischen den Parametern 16/32/64/128/256MB zu wählen. In der Regel reicht die Standardeinstellung von 64MB aus. Es gibt aber Ausnahmen, bei den es sich lohnt umzustellen:

Speicher	Beispiele
16MB GPU-Speicher	Als Server ohne Monitor
32MB GPU-Speicher	Für grafische Benutzeroberflächen und Programme, die keinen Gebrauch von Video oder 3D-Rendering machen. (Büro Pi)
64MB GPU-Speicher	Für grafische Benutzeroberflächen und Programme, die gelegentlich Videos abspielen oder mit 3D-Effekten arbeiten. (Im Internet surfen)
128MB GPU-Speicher	Für grafische Benutzeroberflächen und Programme, die intensiv Multimedia nutzen (Spiele, 3D-Rendering)
256MB GPU-Speicher	Spiele

Tabelle 10 Beispiele für die GPU Speicheraufteilung

A4 Headless-Verbindung über SSH

SSH: Mit der Aktivierung des *ssh*-Servers kann eine Verbindung aus dem Netzwerk zum Raspberry hergestellt werden. Dies ermöglicht einen Zugriff auf den Pi durch einem anderen Computer und dessen Steuerung über die Kommandozeile. Die Aktivierung dieser Funktion ist ratsam, bei Nutzung des Pi´s ohne Maus, Tastatur und Monitor.

A5 SPI

SPI: Automatisches laden der SPI Kernel Module wird in dieser Sparte aktiviert bzw. deaktiviert (wird z.B für das PiFace Board benötigt)

A6 Audio auf HDMI oder 3.5mm Buchse

Audio: Es sind drei Möglichkeiten zur Ausgabe des Audiosignals einstellbar: auf **0) Auto** (Raspbian automatisch entscheiden lassen), **1) 3.5mm** (auf die 3,5mm Buchse) oder **2) HDMI** (Bild und Ton).

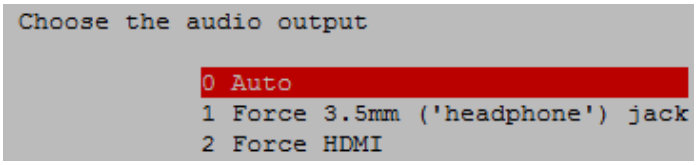


Abb16. Das Auswahlmenü für den Ton

A7 Raspi-config aktualisieren

Update: Die Voraussetzung für diesen Punkt ist eine funktionierende Internetverbindung. Der Raspberry ist so konfiguriert, dass er eine IP automatisch zugewiesen bekommt, falls ein DHCP Server vorhanden ist. Mit diesem Menüpunkt kann **raspi-config** auf den aktuellen Stand gebracht werden.

Die meisten Einstellungen werden erst nach einem Neustart aktiv.

2.10 Das Verzeichnisstruktur von Raspbian

Dieses Kapitel beschreibt die Verzeichnisstruktur eines Linux-Systems. Raspbian nutzt, wie viele andere Linux-Distributionen den „Filesystem Hierachy Standard“. Dieser Standard wurde entwickelt um Vereinheitlic

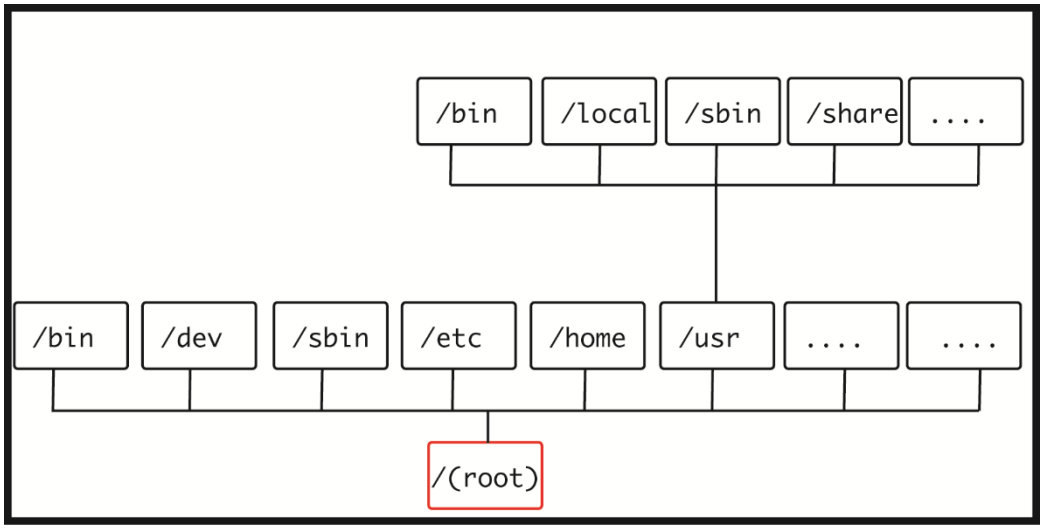


Abb17. Die Baumwurzelstruktur der Raspbian Distribution

hung in der Ordnerhierarchie zu gewährleisten und um Datentausch zwischen unterschiedlichen Linux-Systemen zu vereinfachen.[22] Die Linux-Struktur ist, wie bei vielen Betriebssystemen auf der Baumstruktur aufgebaut. D.h. das Wurzelverzeichnis (auch als Stammverzeichnis bekannt) bezeichnet man als Root. Der Pfad wird mit einem Schrägstrich **/** gekennzeichnet. Jedes weitere Verzeichnis der Unterverzeichnisse wird ebenfalls mit einem **/** gekennzeichnet.

Die Unterverzeichnisse sind gut sortiert und jeder Ordner hat seine eigene Funktion. So erhalten die Verzeichnisse `/bin` (binaries) und `/sbin` (system binaries) nur Programme, die für die Funktion des Systems von Bedeutung sind. In den Ordnern `/usr/bin` und `/usr/sbin` (usr = unix system resources) befindet sich zusätzliche Software, wie z.B. Spiele. Der Unterschied zwischen dem `/bin` und `/sbin` ist, dass normale Benutzer die Programme von `/bin` benutzen können. Im `/sbin` können die Programme nicht ohne administrative Rechte (superuser) ausgeführt werden.

`/`	(root) Stammverzeichnis ist die höchste Ebene des Betriebssystems. In diesem Verzeichnis befinden sich alle anderen Verzeichnisse des Systems.
`/bin`	(stand of binaries) enthält elementare Linux-Kommandos zur Systemverwaltung, die jeder Benutzer ausführen kann. Weitere Programme befinden sich in `/usr/bin`. Bei vielen neueren Linux-Systemen ist `/bin` einfach eine Verlinkung auf `/usr/bin`.
`/dev`	(stand of devices) In diesem Verzeichnis befinden sich Geräte-Dateien (alle Device-Dateien). Wie unter Unix üblich, wird auf Hardware- und System-Komponenten jeweils über Gerätedateien zugegriffen.
`/sbin`	(stand of system binaries) Grundlegende System-Binär-Dateien (benötigt meistens superuser) Programme, die in `/sbin` erwartet werden: `shutdown`, `halt`, `ifconfig`, `reboot`, `update`
`/etc`	(stands for et ceteralst) der Lagerplatz für die Systemkonfigurationsdateien. (benötigt superuser). In diesem Verzeichnis und in den befindlichen Unterverzeichnissen ist jede Art von Konfigurationsdateien enthalten (für System und Programme).
`/home`	In diesem sogenannten Home-Verzeichnis liegen alle angelegten Benutzerkonten. Beispiel: `/home/pi` (der beim ersten Start von Raspbian als Standard-Benutzer angelegt ist). Dieses Heimatverzeichnis ist jenes Verzeichnis, in dem sich der Anwender nach dem Einloggen automatisch befindet und auf dessen Dateien er uneingeschränkte Zugriffsrechte hat.
`/usr`	(stands for unix system resources) Es ist die zweite Ebene in der Hierarchie der Baumstruktur. `/usr` enthält die Quellcodes zu Linux, Anwendungsprogramme, das komplette X-System usw. Der Inhalt in diesem Verzeichnis ändert sich nur bei Installationen von Paketen und Updates.
`/media`	In diesem Verzeichnis werden alle Wechseldatenträger angezeigt (Mountpoints).

Tabelle 11 Die wichtigsten Verzeichnisse im Raspbian

Bei Änderungen von Dateien außerhalb des Home-Verzeichnisses ist große Vorsicht geboten. Wird eine Systemdatei ohne die nötigen Kenntnisse verändert, kann es zur Beschädigung des Betreibsystems kommen.

Weitere Informationen über die Verzeichnisstruktur unter folgenden Links:

http://de.wikibooks.org/wiki/Linux-Praxisbuch:_Verzeichnisse_unter_Linux
http://wiki.ubuntuusers.de/Verzeichnisstruktur

[22] http://wiki.ubuntuusers.de/Verzeichnisstruktur

http://en.wikipedia.org/wiki/Filesystem_Hierarchy_Standard

2.11 Wichtige Shell-Befehle

„Bourne Again Shell ist ein englisches Wortspiel: Die bash ist somit die wiedergeborene Bourne-Shell, die neben der Korn-Shell und der C-Shell zu den drei klassischen Unix-Shells zählt. Unter Linux sind alle drei Shells und noch einige weitere verfügbar, standardmäßig wird aber zumeist die bash eingerichtet.“[23]

Was ist nun eine Shell? In erster Linie wird die Shell zum Aufruf von Linux-Kommandos und Programmen eingesetzt. Sie stellt damit eine Art Kommandointerpreter dar, vergleichbar in etwa mit cmd.exe aus der Windows-Welt. Eine Shell wird in jedem Terminalfenster und in jeder Textkonsole nach dem Login ausgeführt. Gleichzeitig stellt die Shell eine Programmiersprache zu Verfügung, mit der Arbeitsabläufe automatisiert werden können. Mit speziellen Shell-Kommandos können sie innerhalb dieser Programme Variablen verwenden, Abfragen und Schleifen bilden etc.

2.11.1 Der Befehl APT

Die Programme werden unter Linux auch Pakete genannt, diese werden meistens aus dem Internet gezogen. Beim Installieren von Paketen werden auch abhängige Pakete mit installiert, die für reibungslose Funktionalität sorgen. Diese Aufgabe übernimmt das Verwaltungssystem APT (Advance Package Tool). Der Befehl wird über die Konsole bedient und kann Pakete installieren, aktualisieren und vollständig deinstallieren.

2.11.2 Raspbian auf den neusten Stand bringen

Um neue Pakete oder bestehende Programme und Tools zu aktualisieren muss ein System Update und Upgrade durchgeführt werden. Raspbian nutzt zur Orientierung eine Datenbank, die mit Hilfe des Befehls `update` aktualisiert werden kann. In dieser Datenbank werden die Änderungen der Softwareliste, die Bugfixes und die Abhängigkeit verschiedener Applikationen festgehalten. Ist diese Liste mit den aktuellen Paketinformationen versorgt, kann das `upgrade` gestartet werden. Der Befehl aktualisiert alle Programme, Tools, wenn sie durch den Hersteller geändert worden sind. Ändert sich jedoch die Paketabhängigkeit verschiedener Programme, so werden diese leider nicht deinstalliert.

Die Befehle:

```
sudo apt-get update
```

Mit diesem Befehl werden die Update- und Download-Listen aktualisiert. Der nächste Befehl bringt den *Raspian „wheezy“* auf den neusten Stand,:

```
sudo apt-get upgrade -y
```

Alternativ kann alles in einer Zeile abgewickelt werden:

```
sudo apt-get update && sudo apt-get upgrade –y
```

[23] Kofler, 2014 S.433

2.11.3 Benutzer verwalten

An Hand dieses Beispiels wird gezeigt, wie ein neuer Benutzer namens Testuser angelegt und ein neues Passwort für diesen hinzufügt wird.

```
sudo useradd -m Testuser
sudo passwd Testuser
New passwd: xxx
Re-enter new passwd: xxx
```

Um zu testen ob es funktioniert hat, wählt man sich einfach mit dem neuen Benutzer ein:

```
su Testuser
```

Nachdem die Passwort Abfrage geklappt hat, ist der neue Benutzer aktiv.

Mit

```
exit
```

wird das Konto verlassen.

Der neue Nutzer besitzt nur Zugriffsrechte auf das `/home/Testuser`. Die (superuser) administrativen Rechte werden mit folgendem Befehl eingeräumt.

```
sudo usermod –a –G sudo Testuser
```

D.h., dass das Benutzerkonto Testuser zu der bestehenden Gruppe sudo eingefügt wurde.

Die Verwaltung von Konten, Gruppenkonfigurationen oder Zugriffsrechte unter Linux besitzt ein sehr großes Spektrum an Informationen und Optionen. Deswegen wird hier nicht weiter auf die Benutzerkontoverwaltung eingegangen.

Für weitere Informationen:

http://wiki.ubuntuusers.de/Benutzer_und_Gruppen
http://openbook.galileo-press.de/linux/linux_kap13_002.html#dodtp202e5804-54bd-499e-8ef0-c16a0416d741

2.11.4 Neue Pakete installieren / deinstallieren

Am Beispiel von Motion (es handelt sich um eine Bewegungserkennungssoftware), wird die Installation und Suche nach Paketen in der Datenbank mit Hilfe der Kommandozeile gezeigt.

Motion-Paket Suchen

Um zu überprüfen, ob ein Paket für Raspbian vorhanden ist, muss folgendes eingetippt werden:

```
apt-cache search motion
```

Die Konsole liefert eine Liste an Programmen, die mit Video bzw. Motion zu tun haben. In mitten dieser Auflistung befindet sich Motion.

Motion-Paket installieren

Der folgende Befehl in der Kommandozeile installiert Motion

```
sudo apt-get install motion -y
```

Motion-Paket Löschen

Um die Pakete von Motion rückstandslos vom Raspberry Pi zu entfernen (`--purge`) wird mit folgendem Befehl Motion komplett gelöscht (mit eventuell allen Konfigurationsdateien des Pakets):

```
sudo apt-get remove --purge motion
```

2.11.5 Liste wichtiger Shell-Befehle

Shell-Kommandos für Datei-/Verzeichnisstruktur

Um Dateien und Verzeichnisse auflisten:

```
ls
```

Um detaillierte Informationen zu bekommen:

```
ls -a
```

Über Angabe eines Pfades, den Inhalt eines Verzeichnisses anzuzeigen:

```
ls -l /usr/bin
```

Mit diesen Befehl wird angezeigt, in welchem Verzeichnis sich der Benutzer befindet:

```
pwd
```

Mit dem Befehl

```
cd
```

wechselt man zwischen den Verzeichnissen.

Bei den nächsten Befehlen ist zu beachten, dass außer im /home Verzeichnis, das Kommando sudo davor geschrieben werden muss (siehe System-Befehle)

Um ein neues Verzeichnis zu erstellen:

```
mkdir VERZEICHNISNAME
```

So lang das Verzeichnis leer ist, kann mit Befehl

```
rmdir VERZEICHNISNAME
```

gelöscht werden.

Manchmal müssen einige Dateien oder Verzeichnisse kopiert oder verschoben werden. Die Befehle dafür lauten:

Für kopieren einer Datei auf dem Desktop:

```
cp DATEINAME.DATEIENDUNG /home/pi/Desktop/
```

oder um eine Datei auf den Desktop zu verschieben:

```
mv DATEINAME.DATEIENDUNG /home/pi/Desktop/
```

Hat man die Datei bzw. Verzeichnis nur kopiert, kann mit dem Befehl die Datei

```
rm DATEINAME.DATEIENDUNG
```

gelöscht werden.

System-Befehle

Der

```
sudo
```

Befehl kann vor dem Programm-Aufruf vorangestellt werden. Er ermöglicht berechtigten Benutzern, das Programm im Namen und mit den Rechten eines anderen Benutzers auszuführen. Zum Beispiel um administrative Aufgaben auszuführen. Um Programme zu installieren:

```
sudo apt-get install PACKETNAME
```

oder das System zu konfigurieren (am Beispiel von Motion):

```
sudo nano /etc/motion/motion.conf
```

Wenn ein Skript oder ein Python-Programm geschrieben wird, muss die Datei ausführbar gemacht werden. Der Befehl dazu lautet :

```
sudo chmod +x DATEI.sh
```

Um den RasPi neu zu starten:

```
sudo reboot
```

Den Mini-Rechner ausschalten:

```
sudo halt
```

Die grafische Oberfläche von Raspbian starten:

```
startx
```

Die grafische Oberfläche starten über Putty und Xming:

```
startlxde
```

3. Zielsetzung & Konzeptentwicklung

Das Ziel der Master Thesis ist, ein Baukastensystem zu entwickeln, dass dem Anwender die Hardware und Software eines Raspberry Pi´s auf eine praktische Art und Weise vermittelt. Das Baukastensystem soll tiefere Einblicke in die Welt von Linux und Hardwareprogrammierung allgemein verschaffen. Die einzelnen Module sollen Kenntnisse und Erfahrung vermitteln, die auf weitere Module oder Aufgaben anwendbar sind. Es soll die jungen Menschen für Raspberry Pi, Linux, Programmierung von Hardware und für Entwicklung eigener Ideen begeistern. Das Konzept soll dazu führen, dass sich viele nach der Abarbeitung des Baukastensystems mit dem Raspberry Pi weiter auseinander setzen. Die Handhabung der kleinen Platine ist sehr schnell zu verstehen, sie ist leicht bedienbar und in vielen Themengebieten einsetzbar. Ein weiteres Merkmal ist, dass der Raspberry mit vielen für ihn und für andere Single-Boards konstruierten Hardwaregadgets erweiterbar ist.

Um das Ziel zu erreichen, mussten verschiedene Module entwickelt werden, die den Schwierigkeitsgrad von Modul zu Modul steigern, aber trotzdem im Bereich des Möglichen für die Anwender/innen bleiben. Welche Module den Weg in das Baukastensystem gefunden haben und welche Lern- und Zieleffekte dabei verfolgt worden sind, wird im Kapitel 3.1. „Die einzelnen (Lern)Ziele der Module“ erklärt.

Für jedes Modul wurden Anleitungen entwickelt und in einfacher, verständlicher Form niedergeschrieben. Die Anwender/innen sollen mit diesen Anleitungen schnell zurechtkommen und die einzelnen Schritte ohne Probleme nachzuvollziehen. Die Ganze Strukturierung des Baukastensystems wird im Kapitel 3.2. „Strukturierung der Anleitungen“ aufgezeigt.

Alle Projekte die den Weg in das Baukastensystem gefunden haben, brauchen ein Betriebssystem, das für die Durchführung einzelner Module vorkonfiguriert ist. Die Anwender sparen dadurch Zeit und können sich ganz auf den Kern des Moduls konzentrieren. Dank NOOBS konnte schnell eine Lösung entwickelt werden. NOOBS ist ein Installationsmanager für verschiedene Distributionen, die

es für Raspberry Pi gibt. Es ist auch möglich eigene Distributionen zu erstellen und von der SD-Karte mit dem Installationsmanager zu installieren. Mit dieser Möglichkeit wurden für die Module vorkonfigurierte Distributionen erstellt. Dadurch muss der Benutzer sich nicht mehr um alle Einstellungen und Installationen im System kümmern. Die Erstellung einer „User-Distribution“ ist aufwendig und nimmt viel Zeit in Anspruch. Der Ablauf zur Erstellung einer eigenen Distribution, wird im Kapitel 3.3 „NOOS: Eine benutzerdefinierte Distribution erstellen, beschrieben.

3.1 Die einzelnen (Lern)Ziele der Module

Jedes Modul soll dem Interessierten Wissen vermitteln, Spaß machen, die Kreativität ankurbeln und Anregungen für neue Ideen schaffen. Um diese Punkte zu erfüllen haben folgende Themen in dem Baukastensystem Platz gefunden:

Modul 1: RasPi und Raspbian

Der Raspberry Pi ist für viele Personen Neuland. Die Funktionsweise eines Pi´s ähnelt sehr einem Desktop-Computer. Der Raspberry Pi ist ein vollwertiger Computer, der Internetfähig ist, Medien abspielen und als Office Computer genutzt werden kann. Viele Schüler/innen wissen, wie ein Computer mit Windows oder Mac OSx bedient wird, aber nicht wie weit sich Linux von den anderen Systemen unterscheidet.

Bevor mit dem RasPi etwas experimentiert werden kann, müssen die Schüler/innen und Studenten/innen mit den Grundlagen des Linux-Systems konfrontiert werden. Das Ziel dieses Moduls ist, dass sie lernen, wie Raspbian auf dem Raspberry Pi installiert und konfiguriert wird. Sie sollen die Funktionsweise von Raspbian, die Bedienoberfläche und das Navigieren durch das System kennenlernen.

Nach dem Modul, sollen die Schüler/innen bzw. Studenten/innen in der Lage sein, die nötigen Hardwarekomponenten zusammenzusetzen, die Grundlagen für die Bedienung von Linux-Systemen zu verstehen und ein Raspbian netzwerkfähig machen. Diese Grundlagen sind das Maßstab für alle weiteren Module.

Modul 2: RasPi als Mediacenter

Der Raspberry Pi ermöglicht mit der richtigen Distribution ein Mediacenter aufzubauen. Das Modul soll zeigen, wie gut der RasPi als Mediacenter funktioniert und was für eine große Auswahl an Unterhaltungsmöglichkeiten und Funktionen zur Verfügung stehen.

In diesen Modul lernen die Studenten/innen und Schüler/innen die Stärken des Pi´s im Einsatz als ein Multimediacenter. Es soll zeigen wie komfortabel und schnell es ist das System zu installieren und zu konfigurieren und welche weitere Vielfalt an Einsatzmöglichkeiten XBMC/Kodi bietet. Des Weiteren zeigt das Modul auf, welche Speichermedien angeschlossen werden können (externe Festplatten, USB-Stick, NAS usw.) und welche Steuerungsmöglichkeiten das Media-Center bietet (Tastatur und Maus, Smartphone-Apps, usw.).

Modul 3: GPIO-Grundlagen

Die 17 Input/Output-Pins steuern elektrische Bauelemente über die HIGH- und LOW-Zustände. Mit dem GPIO-Port werden Roboter, Hausteuerung oder andere interessante Projekte erstellt. Um einen Roboter zu bauen und programmieren, müssen zuerst die Grundlagen der Elektrotechnik anhand von praktischen Beispielen erläutert werden.

Das Ziel ist es, dem Anwender durch praktische Beispiele die Funktionsweise von elektrischen Bauelementen, Sensoren und Treibern aufzuzeigen. Dabei wird wichtiges Grundwissen (Ohm’sches

Gesetz, PULL_UP_DOWN-Widerstände usw.) vermittelt. Der Anwender soll auf spielerische Art und Weise an Hardwareprogrammierung herangeführt werden. Er lernt dabei, wie einfach es ist eine Schaltung aufzubauen und jene in Python mit ein wenig Code zum Leben zu erwecken.

Modul 4: RasPi als SpyCam

Eine Kamera an RasPi angeschlossen, bietet eine prima Möglichkeit den RasPi als Überwachungssystem zu verwenden.

Das Ziel ist es, den Benutzern zu zeigen, dass zusammen mit der Pi-Kamera und der GPIO-Schnittstelle, eine funktionierende Überwachungsvariante aufbaut werden kann, die mit vielen weiteren Funktionen erweiterbar ist. Auch die Stärken von Python sollen vermittelt werden. Sie lernen verschiedene Bibliotheken in Python zu integrieren, Konfigurations-Dateien für die Überwachung zu erstellen und zu editieren.

Modul 5: GPIO Robot

Es vereint die Grundlagen aller davor dargestellten Module (zum größten Teil Modul 3 und 4) zu einem komplexeren System.

Der Raspberry Pi eröffnet in der Technikwelt viele Möglichkeiten um etwas zu erschaffen, was noch vor einiger Zeit nicht so einfach war. Dabei sehen die Schüler/innen bzw., Studenten/innen, die große Vielfalt an Einsatzmöglichkeiten, die der Pi zusammen mit den elektrischen Komponenten und verschiedener Software bietet. Es soll ihnen vermittelt werden, wie einfach es ist, einen Roboter zu bauen und diesen mit einer Programmiersprache zum Leben zu erwecken. Es soll sie begreifen lassen, wie eine Motorensteuerung und verschiedene Sensoren funktionieren.

An diesen Beispielen soll gezeigt werden, dass sich noch mehr Sensoren und andere elektrische Elemente einbauen lassen.

Beispiele:

- LEDs zum Rückwärtsfahren.
- Ein extra Motor für Kamerabewegung und viele weitere Konstruktionen.

Bei den Sensoren besteht die Möglichkeit:

- Ein Beschleunigungssensor für die Geschwindigkeitsermittlung
- Bewegungssensor um den Roboter aus dem Schlafmodus zu erwecken
- Geräuschsensor für Stimmensteuerung oder bestimmten Geräuschen folgt.

Dank Arduino ist die Auswahl an Sensoren noch viel größer.

Ziel ist es den Nutzer tiefer in die Materie der GPIO-Steuerung zu führen und ihm Möglichkeiten zu geben seine eigene Kreativität auszuleben.

Modul 6: Finger Tipp Reaktionsspiel

Für den Raspberry Pi stehen nicht nur Sensoren, Kameras oder elektrische Bauelemente zur Verfügung, sondern auch verschiedene Boards, wie z.B. das Trellis.

Das Ziel des Moduls ist, zusammen mit dem Anwender ein Spiel zu entwickeln, dass auf Schnelligkeit basiert. Dabei lernt er, die Anwendungsmöglichkeiten der I²C-Schnittstelle kennen. Der Anwender bekommt die Möglichkeit es weiter zu entwickeln, z.B. mit einer Reaktionszeit zu versehen oder durch Hinzufügen von neuen Figuren. Das Trellis Pad lässt sich auch als Lichtdemos nutzen oder als Tastensteuerung für den Pi anwenden.

Modul 7: BrickPi & RasPi – Der Legoroboter

Das Ziel ist es, den Schülern/innen bzw. Studenten/innen, die große Vielfalt an Einsatzmöglichkeiten aufzuzeigen die der Pi zusammen mit dem BrickPi besitzt. Dies bietet die Chance, dass sie sich mit dem Thema weiter auseinandersetzen, selbstständig eigene Roboter bauen und noch vieles mehr. Durch das gelernte im Modul 3 „GPIO-Grundlagen“ und Modul 5 „GPIO-Robotik“ ist es einfacher, die Programmierung für Legoroboter zu verstehen. Dabei lernen die Studenten/innen und Schüler/innen, die einzelnen Schnittstellen des BrickPi´s zusammen mit einer Python Applikation anzusprechen, um dem selbstgebauten Roboter „Leben einzuhauchen“. Zwar müssen sie neue Programmbefehle erlernen aber diese funktionieren auf demselben Prinzip, wie die GPIO-Bibliothek. Das System ermöglicht es den Anwendern, über den GPIO-Port, den Legoroboter zu erweitern (z.B. LEDs und weitere Sensoren aller Art).

3.2 Strukturierung der Anleitung

Die Anleitungen für die Module wurden in der „DU“-Form geschrieben. Es soll den Leser/in direkt ansprechen und mehr Ansporn geben, das Interesse für RasPi zu wecken. Die Anleitungen für die Module beinhalten alle wichtigen Informationen, die aber recht kurz, knapp und verständlich die Durchführung beschreiben. Ausnahmefälle, die z.B. zur Beschädigung des Rapberry Pi´s führen könnten, werden mit Warnungen sichtbar dargestellt. Die Warnungen und Hinweise werden in einem gelben Kasten mit einer selbsterstellten RasPi Comicfigur gekennzeichnet, die je nach Situation die Zeichen-„?,,!“ beinhalten (siehe Grafik). Enthält der Hinweis nur einige wissenswerte Informationen, hat der RasPi nur die Hände ausgebreitet, wird jedoch z.B. vor falschem Hardwareaufbau gewarnt, so sind über dem RasPi rote Ausrufezeichen platziert.

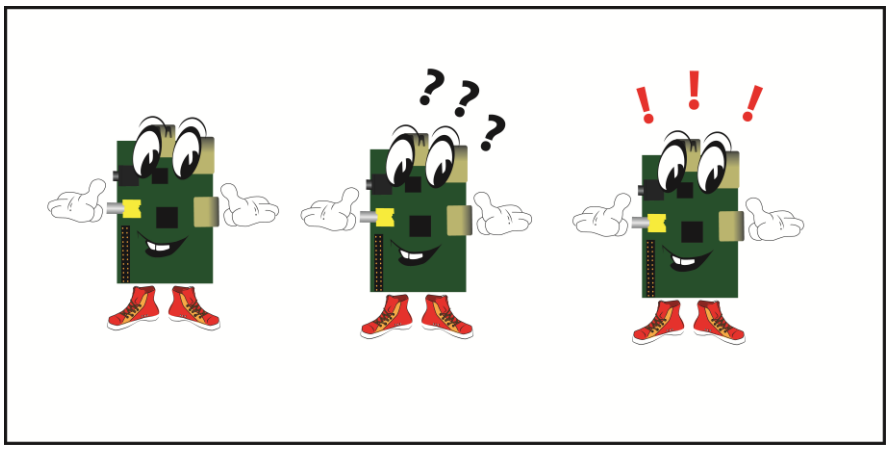


Abb18. RasPi erklärt die Welt, die drei Figuren, die im Baukastensystem bei verschiedenen Situationen abgebildet werden.

Auf der ersten Seite jedes Moduls befinden sich die Inhalte, die Software und die Beschreibung des Moduls. Die Software wird in drei Kategorien (Windows, OS für RasPi und die Software auf dem RasPi) aufgeteilt. Auf der nächsten Seite sind die Hardwareelemente in bildlicher Form aufgelistet. Durch die Bilder ist der Anwender/innen schneller in Lage, die Hardwarekomponenten zu finden (Lego-Prinzip). Um den comichaften Look zu behalten wurden die Grafiken zum größten Teil mit dem Vektorzeichenprogramm Illustrator entwickelt. Nach der Hardwareliste folgen an manchen Stellen Grundlagen (wie z.B. LED-Beschreibung, Widerstand ausrechnen). Die eigentliche Anleitung für das Modul beginnt erst mit dem Aufbauplan, den Schaltungen, den Installationsanleitungen, Konfigurationen und weiteren wichtigen Erläuterungen.

Die Anleitungen zu jedem Modul des Baukastensystems orientieren sich ein wenig an den Legoanleitungen und der Internetseite von Raspberry Pi Foundation. Sie sollen schnell aufgebaut werden, erlernbar sein und Spaß machen. Einige der Schritte sind mit Bildern und etwas Text beschrieben. Denn Bilder haben einen großen Vorteil gegenüber Text. Mit Grafiken/Bildern lässt sich schneller kommunizieren und sind oft selbsterklärend.

Jedes Modul baut auf dem nächsten Modul auf (mit einer kleinen Ausnahme, das Mediencenter). D.h. mit dem Modul „Raspberry Pi und Raspbian“ lernt der Benutzer, wie ein System installiert und konfiguriert wird. Dabei werden einzelne Punkte dieser Anleitung in den nächsten Modulen benötigt oder ausgebaut (Beispiel `raspi-config` und WLAN-Einrichtung). Die GPIO Grundlagen werden in fast jedem darauffolgenden Modul genutzt und in den darauf folgenden Kapiteln weiter ausgebaut.

3.3 NOOBS: Eine benutzerdefinierte Distribution erstellen

In diesem Kapitel wird eine der vielen Möglichkeiten zur Erstellung einer benutzerdefinierten Distribution für NOOBS erläutert. Der folgende Lösungsvorschlag beschreibt aufgrund des fehlenden Treibers für den SD-Karten-Leser für Linux, eine umständliche Durchführung.

Um eine Benutzerdefinierte Distribution zu erstellen, wird folgende Software und Hardware benötigt;

- NOOBS (*http://www.raspberrypi.org/downloads/)*
- Linux Rechner (ggf. Virtuelles Linux über VMware)

Nach dem Herunterladen wird NOOBS auf die SD-Karte kopiert und auf dem RasPi installiert. Auf dem Raspberry Pi werden an Raspbian alle Einstellungen und Installationen vorgenommen, um die Ausführung der Anleitung zu gewährleisten. Sind alle Konfigurationen und die Distribution speziell für das Baukastensystem eingerichtet, wird die SD-Karte zuerst als Image auf ein Windowssystem kopiert. Für diese Bearbeitung ist der win32diskimager zu empfehlen. Mit win32diskimiger wird die SD-Karte in ein Image eingelesen und anschließend auf die Linux Datenträger kopiert.

Um auf die Dateien der vorkonfigurierten Distribution zuzugreifen, muss ein Mount Punkt erstellt werden. D.h. das Image wird in das System in ein Verzeichnis „eingehängt“ (Mount Point) und so möglich gemacht auf die Dateien und Verzeichnisse zugreifen.

```
fdisk –l GPIO_Raspbian.img
```

Mit fdisk werden alle Partitionen in der GPIO_Raspbian.img aufgezeigt. Welches Image aus dem Container die Raspbian Distribution ist zeigt folgende Bildschirmausgabe:

```
Disk GPIO_Raspbian.img: 1939 MB, 1939865600 bytes
255 heads, 63 sectors/track, 235 cylinders, total 3788800 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x00014d34

Device              Boot Start      End   Blocks   Id      System
GPIO_raspbian.img1        8192   122879    57344     c      W95 FAT32 (LBA)
GPIO_raspbian.img2      122880  3788799 1832960    83      Linux
```

Fdisk zeigt an, dass sich zwei Partitionen in dem Image-Container befinden. Das System liefert Informationen über die Partition und über die jeweilige Zugehörigkeit des Containers zu dem Betriebssystem. Um die Raspbian Partition in das System einzubinden, werden folgende Attribute aus der Fdisk-Auskunft benötigt:

```
Sector size: 512
Start block: img2: 122880
```

Der Inhalt kann nur durch die Angaben der Sektor Größe und den Anfangsblock der „GPIO_Raspbian.img2“ gemountet werden:

Zuerst wird ein Ordner erstellt:

```
mkdir GPIO_Raspbian
```

Mit

```
sudo mount GPIO_Raspbian.img -o loop,offset=$(( 512 * 122880)) /GPIO_Raspbian
```

wird das Image eingebunden.

Nachdem das Image in das System eingehängt wurde, wird die gesamte Verzeichnisstruktur mit kompletten Inhalt in ein für das OS erstellten Ordner hinein kopiert.

```
sudo cp -a * /home/USER/GPIO
```

Die kopierten Dateien werden mit „tar“ in ein Container namens „root.tar“ gepackt. Anschließend wird die tar-Datei zusätzlich komprimiert:

```
cd /home/USER/GPIO
sudo tar cfp root.tar .
sudo xz -9 -e root.tar
```

Je nach Rechnerleistung kann das packen und komprimieren bis zu zwei Stunden in Anspruch nehmen.

Die fertige „root.tar.xz“ wird in den /NOOBS/os in einen neu erstellten Ordner hineinkopiert:

```
mkdir GPIO
cp root.tar.gz /home/USER/NOOBS/os/GPIO
```

Als letztes müssen alle Dateien und Ordner (außer root.tar.gz) aus /NOOBS/os/Raspbian/ in GPIO kopiert werden.

Bei den kopierten Dateien handelt sich zum größten Teil um Konfigurationsdateien für die Installation des Systems. `Partitions.json` liefert alle Informationen über die beiden Partitionen. In der Datei stehen die Namen der beiden Partitionen (boot und root), Dateisystem, die Größe des Systems nach der Installation und die unkomprimierte Größe von Raspbian als „root.tar“. Der Wert, der neuen `root.tar.xz muss` im unkomprimierten Zustand (`uncompressed_tarball_size`) angegeben werden und bei Bedarf wird die nominale Größe verändert.

Diese wird mit folgenden Befehl überprüft:

```
sudo xz –l root.tar.xz
```

und trägt dies ein, unter:

```
„partition_size_nominal“ : 2040
```

`0s.json` liefert Auskunft über die Distribution, den Kernel, Erstellungsdatum des Systems und die Beschreibung der Distribution. Zusätzlich ermöglicht die Datei die Zugangsdaten von Raspbian zu ändern. Die letzte editierbare Datei `flavours.json`, ist die für die Beschreibung von der zu installierenden Distribution im Menü von NOOBS zuständig.

Nach der Installation von „Raspbian_GPIO“ über NOOBS wird das vorkonfigurierte Raspbian für die GPIO Module gestartet.

4. Das Baukastensystem

Das Baukastensystem konzentriert sich im ersten Teil nur auf den softwarebasierenden Bereich der Linux Distributionen für den RasPi (Raspbian und OpenELEC) Der zweite Abschnitt konzentriert sich stärker auf den hardwaregebundenen Bereich. Der Schwerpunkt in diesen Modulen wird auf Hardwarekomponenten (z.B. elektrische Baudelemente) und deren Programmierung gelegt.

Der erste Teil des Baukastensystems soll den Studenten/innen bzw. Schülern/innen Linuxgrundlagen aufzeigen. Dabei wird eine Linux-Distribution installiert, konfiguriert und genutzt. Es soll den Anwendern demonstriert werden, dass Linux nicht unbedingt aufwendig sein muss.

Der zweite Teil des Baukastensystems baut zum größten Teil auf der GPIO-Steuerung auf. Anfangs werden Grundlagen durch Steuerung von verschiedenen elektrischen Bauelementen erklärt. Dabei werden simple Schaltungen aufgebaut um die Grundlagen der Elektronik und Programmierung zu erläutern. Diese Grundkenntnisse werden im darauf folgenden Modul weiter ausgebaut und zur Anwendung gebracht.

Kapitel 4 setzt sich mit der Entwicklung der einzelnen Module für das Baukastensystem auseinander. Jedes Modul wird der Reihe nach beschrieben, mit allen Gedanken und Ideen, die bei der Entstehung der Experimente eingeflossen sind.

4.1 Modul 1: Raspbian und RasPi

Der erste Baustein vermittelt die Navigation in einer Linux Distribution über das Terminal und der Desktop-Oberfläche. An ausgewählten Beispielen werden Konfiguration von System und Software vorgenommen, Installation von weiteren Paketen, Systemupdates und Upgrades durchgeführt. Der Schwerpunkt bei Paket-Installation wurde auf Netzwerkprogramme gelegt.

4.1.1 Beschreibung Modul 1

Vor der Installation werden die Hardwarekomponenten zusammengesteckt und das System gestartet. Anschließend wird eine Raspbian Distribution über NOOBS auf dem RasPi installiert. Nach der Installation startet Raspi-Config, mit diesem Tool werden verschiedene Einstellungen im System vorgenommen. Im nächsten Schritt wird die Datenbank der Pakete aktualisiert und Raspbian auf den neusten Stand gebracht. Während Raspbian die beiden Schritte durchführt, werden die wichtigen Konsolenbefehle erläutert. Das WLAN wird über die grafische Oberfläche eingerichtet, weil die Desktopoberfläche von Raspbian eine schnelle und einfache Einrichtung des Drahtlosen Netzwerks bietet. Am Ende des Moduls wird ein Sambaserver eingerichtet um mit Windows auf die Dateien von Raspbian über das SMB-Protokoll zuzugreifen. Die IP-Adresse wird durch eine lokale Domäne ergänzt und eine Headless-Verbindung über Putty oder Remotedesktopverbindung hergestellt.

4.1.2 Entwicklung

Das erste Modul musste so konstruiert werden, dass auch ein „Laie“ schnell und einfach die Grundlagen von Linux versteht. Wie bei jedem anderen System, wird nach der Installation das Betriebssystem konfiguriert und es werden Updates durchgeführt. Logischerweise mussten auch im Modul 1 diese Punkte angesprochen werden.

In der heutigen Zeit sind für jedes System sehr viele Programme vorhanden, viele davon sind nicht für das erste Modul geeignet. Beim Blick auf die Entwicklung der Technik lässt sich erkennen, dass

einer der Schwerpunkte in der Kopplung alle netzwerkfähigen Produkte miteinander liegt. Dies war auch einer der Gründe, sich bei den Paket-Installationen auf Netzwerkbasierende Programmen zu konzentrieren.

4.2 Modul 2: Mediacenter mit OpenELEC

Dank des hardwarebeschleunigten Grafikchips, ist RasPi in der Lage ruckelfrei Full-HD-Videos in fast allen gängigen Formaten abzuspielen. Denn was der CPU (700MHz) an Leistung fehlt gleicht die GPU aus. Modul 2 zeigt die Stärken und die Vielseitigkeit von OpenELEC auf dem RasPi. Nach der Installation wird das System konfiguriert, Zusatzprogramme werden installiert, Medien eingebunden und die Fernsteuerung eingerichtet.

4.2.1 Beschreibung Modul 2

Das System wird mit NOOBS auf die SD-Karte installiert und konfiguriert. Bei der Konfiguration des Systems werden alle Menüpunkte erläutert, aber nur die wichtigsten Einstellungen ausführlich beschrieben. Anhand von Wetter- und YouTube-Addon wird die Vielseitigkeit von OpenELEC aufgezeigt. Die Addons werden installiert, konfiguriert und anschließend ausgeführt. Ein NAS-Server wird in das System eingebunden und verschiedene Medien werden von dem Laufwerk abgespielt. Die Einbindung des NAS-Servers soll ein Einblick verschaffen, wie ein ganzes Haus zu einem Entertainment-Center ausgebaut werden kann.

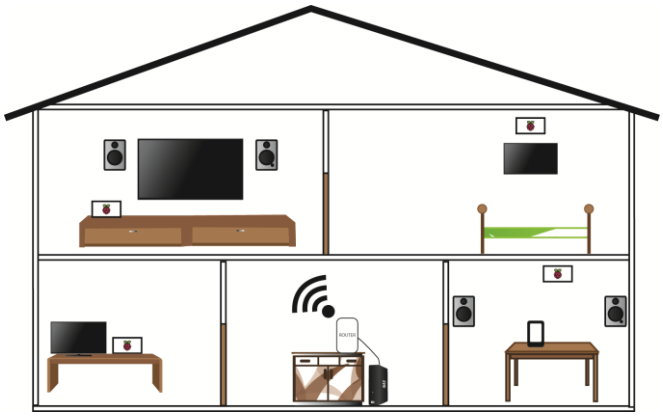


Abb19. Ein Haus in dem jeder Raspberry Pi auf das NAS-Laufwerk am Router zugreifen kann.

Jeder Raspberry Pi mit OpenELEC kann auf das Laufwerk über LAN bzw. WLAN zugreifen und die Medien abspielen.

Das letzte Kapitel der Anleitung beschäftigt sich mit der Fernsteuerung. Es zeigt, wie ein Android-Gerät mit Hilfe der „Yatse“-Applikation, OpenELEC aus der Ferne gesteuert werden kann.

4.2.2 Entwicklung

OpenELEC nutzt XBMC/Kodi als Bedienoberfläche. Der Vorteil an OpenELEC ist, dass die Distribution kaum Speicherplatz benötigt und schnell arbeitet. Das System ist sehr benutzerfreundlich und annähernd selbsterklärend. Bei Entwicklung des Moduls wurde der Schwerpunkt auf die Installation von Addons, einbinden von netzwerkfähigen Laufwerken und Steuerung mit externen Geräten gelegt. Diese drei Punkte erklären ausführlich die einfache Bedienung des Systems und die Integrierung von Speichermedien und Steuerungshardware.

4.3 Modul 3: GPIO Grundlagen

Mit den GPIO Pins und einigen elektronischen Bauelementen lassen sich Schaltungen aufbauen und mittels einer Programmiersprache (in diesen Fall Python) Software zur Steuerung entwickeln. LEDs können ein- und ausschalten werden, Taster Programmieren oder Gleichstrommotoren in eine bestimmte Richtung gedreht werden.

4.3.1 Beschreibung Modul 3

LED Steuerung

Das Kapitel setzt sich im ersten Abschnitt mit LED-Steuerung über die GPIO-Pins auseinander. Die LED wird auf verschiedene Art und Weise über die Software (Python-Code) zum Leuchten gebracht. Die erste Variante schaltet die LED ein und nach x-Sekunden wieder aus. Das zweite Skript schaltet die LED über die Softwareschnittstelle EIN/AUS. Mit der Zahl „1“ auf der Tastatur wird die LED eingeschaltet und mit der Zahl „2“ wieder ausgeschaltet. Das letzte Beispiel zeigt, wie eine LED mit Hilfe der Pulsweitenmodulation (PWM) gedimmt werden kann.

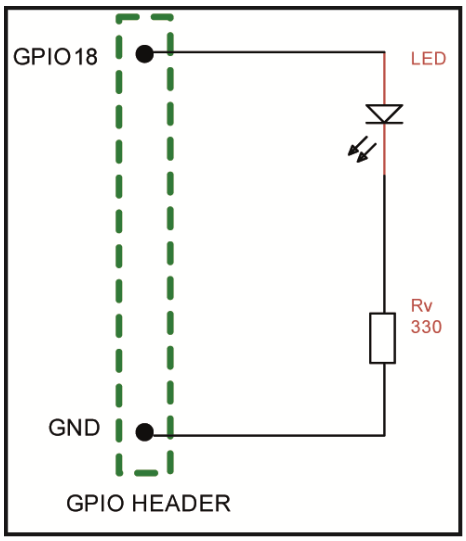


Abb20. Schaltung für LED-Steuerung über den GPIO Port

Der Taster

Kapitel 2 setzt sich mit einem Taster auseinander, der eine LED in zwei Varianten steuert. Variante 1, Schaltet die LED nur ein, wenn der Taster gedrückt ist. Variante 2 wird über ein Interrupt Befehl ausgelöst. D.h. das Programm wartet auf einen Tastendruck bis die „while True“ Schleife ausgeführt wird. Wird die Taste gedrückt, überprüft das Programm den Status (LED AN / AUS?). Ist die LED an, so ist der Wert Satus = 1 und der Taster schaltet die LED aus. Die Variable Status wird auf 0 gesetzt. Wird der Taster erneut betätigt, wird der Status überprüft, die LED an-/ausgemacht und der Status geändert.

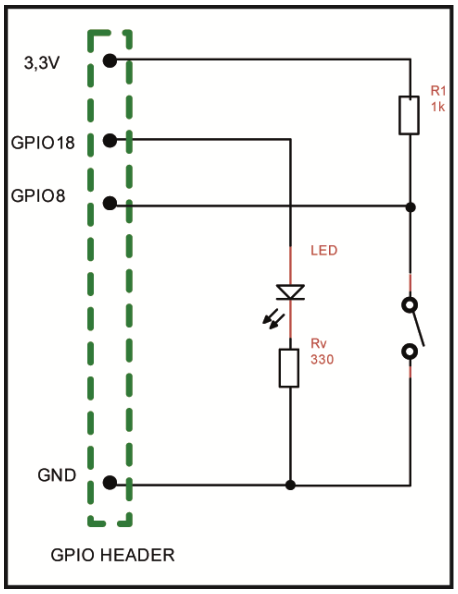


Abb21. Schaltung um eine LED mit einem Taster und GPIO Port zu steuern

Der Taster in den beiden Beispielen wird mit einen Pull_UP-Widerstand (=Spannung hoch ziehen) versehen.

PULL_UP und PULL_DOWN Widerstand

Es gibt zwei Möglichkeiten den Taster mit dem RasPi zu verbinden. Soll der INPUT Pin physikalisch LOW erhalten, wenn die Taste gedrückt wird, so wird der Widerstand als PULL_UP genutzt (Abb22. B). Dabei liegt der Kontakt zwischen dem INPUT des GPIO Pins und GND. Der PULL_UP Widerstand liegt zwischen GPIO INPUT und der Betriebsspannung 3,3V. Beim Drücken des Tasters zieht der PULL_UP Widerstand die Spannung am GPIO INPUT hoch, zu der Spannung von 3,3V (physikalisch HIGH)

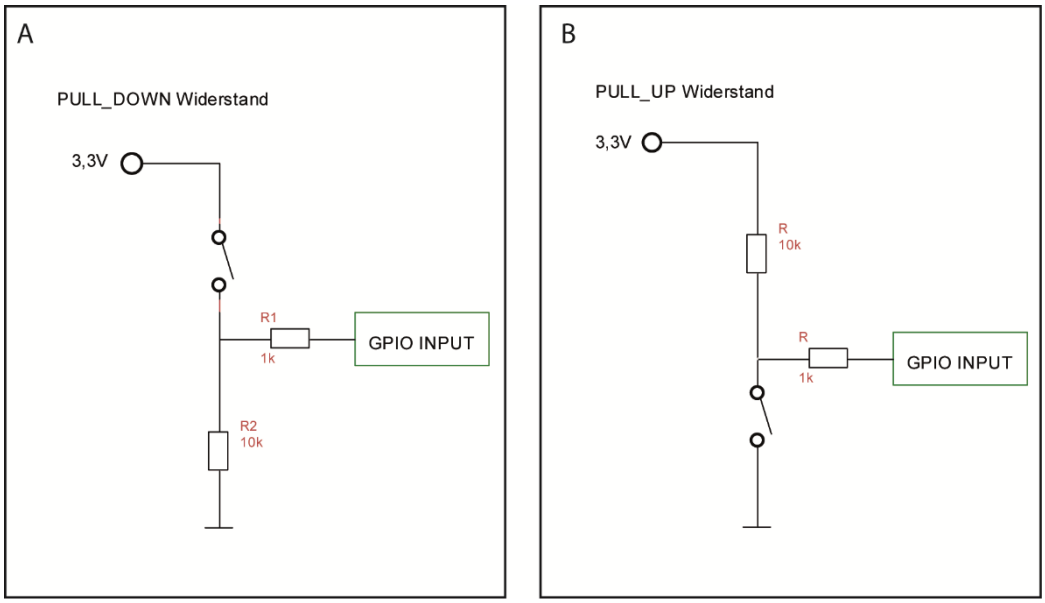


Abb22. A = HIGH aktive Schaltung, B = LOW aktive Schaltung

Soll der INPUT physikalisch HIGH erhalten, wenn die Taste gedrückt wird, so wird der Widerstand als PULL_DOWN geschaltet (Abb.22 A). Der Kontakt liegt zwischen dem INPUT und den 3,3V Pin. Der PULL_DOWN -Widerstand liegt zwischen dem INPUT und GND. Beim Öffnen des Kontaktes zieht der PULL_DOWN-Widerstand die Spannung am INPUT herunter auf GND, was dem physikalischen Zustand LOW entspricht.

Zum Schutz wird noch 1kOhm Widerstand für den GPIO Pin eingebaut PULL DOWN (=Spannung runter ziehen) Widerstand der im RasPi integriert ist. Dieser dient dazu um Unregelmäßigkeiten in der Spannung zu verhindern. Mit PULL UP zieht man die Spannung des Pins auf 3,3V und mit PULL DOWN zieht man sie auf den GND.

Bewegungssensor und Motorsteuerung

In den nächsten Kapiteln wird mittels eines Bewegungsmelders die Fahrrichtung eines Gleichstrommotors geändert. Ist keine Bewegung zu erkennen, dreht sich der Motor vorwärts. Erkennt der Sensor eine Bewegung, ändert der Motor für x-Sekunden die Drehrichtung.

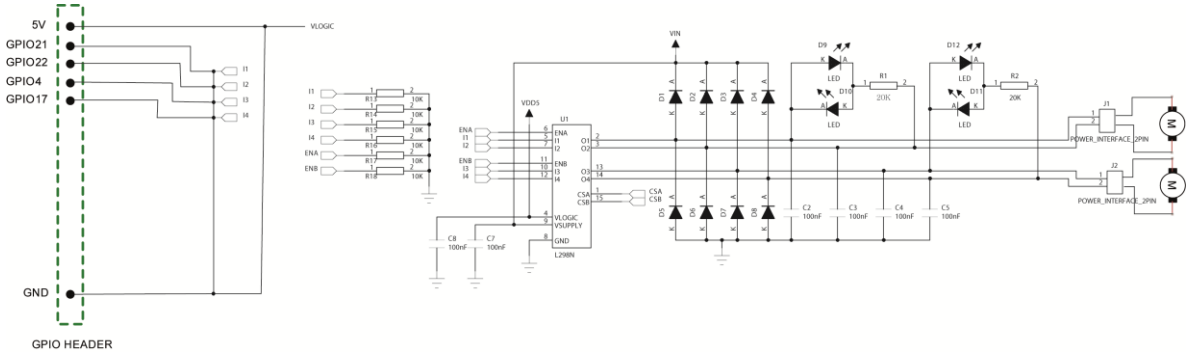


Abb23. Motorsteuerung mit einer H-Brücke und GPIO-Port (siehe auch Anhang)

Der Bewegungssensor ist ein passives Bauelement das nur, wenn in seinem 120 Grad Radius etwas passiert, dieser aktiviert wird. In diesem Sensor ist ein Infrarot-Filter eingebaut, dieser passiv arbeitende Sensor empfängt von einem Gegenstand oder Lebewesen ausgehende Infrarotstrahlung. Jeder Körper der von dem absoluten Nullpunkt abweichende Temperatur ausstrahlt, ist für den Sensor eine Quelle. Dazu müssen zwei Bedingungen erfüllt werden: Der erfasste Bewegungskörper muss einen Temperaturunterschied von +/- 2°C zum Hintergrund haben und sich mindestens 10cm bewegt haben. Um dieses Wärmebild des Körpers zu erfassen, macht sich der IR-empfindliche Sensor mittels eines Multi-Linsensystems (Fresnel-Linsen)[24] zu nutze. Dabei müssen die bewegten Körper im mittleren Infrarotbereich liegen. *„Die Wärmestrahlung des Menschen hat ihr Maximum zwischen 9 und 10 µm im Infrarot-Bereich“*[25]. Vor dem eigentlichen Sensor sind kleine Linsen angebracht, die die Infrarotstrahlung bündeln. Der Erfassungsbereich gliedert sich auf in einzelne Bereiche. Ändert sich in einem dieser Bereiche die Temperatur, wird ein elektronisches Signal ausgelöst. Wird das elektronische Signal von einer im Infrarotliegenden Bewegung ausgelöst, so dreht sich der Gleichstrommotor in die andere Richtung. *„Zur Unterdrückung von Einflüssen aus der Umgebung (übliche wetterbedingte Temperaturänderungen), sind in jedem Sensor 2 Kristalle antiparallel geschaltet. Einer der Kristalle gibt, bei Auftreffen von Wärmestrahlung einen positiven, der andere einen negativen Spannungsimpuls ab.“*[26] Erfassen beide Kristalle gleichzeitig eine Bewegung, so heben sie sich gegenseitig auf. *„Anders verhält es sich bei schnellen Bewegungen. Die Lithiumtantalat-Kristalle geben, entsprechend der Bewegung und der dadurch hervorgerufenen Wärmeänderung im Erfassungsfeld, ihre Impulse zeitversetzt ab. Die beiden Impulse addieren sich zu einer Wechselgröße mit höherer Signalamplitude. Dieses elektrische Ausgangssignal ist proportional der Wärmeänderung und führt zur Meldung einer Bewegung*.“[27] Das Output Signal am

[24] http://reglomat.bircher.com/uploads/media/PIR_d_01.pdf
[25] http://rn-wissen.de/wiki/index.php/Sensorarten#PIR_Passiv_Infrarot_Sensoren
[26] Ebenda
[27] Ebenda

PIR Sensor ist standardmäßig auf 0V, wird Bewegung erkannt, ändert sich sein Zustand auf HIGH. Der PIR Motion Sensor von Seed Studio wird am Raspberry mit dem 5V Pin betrieben.

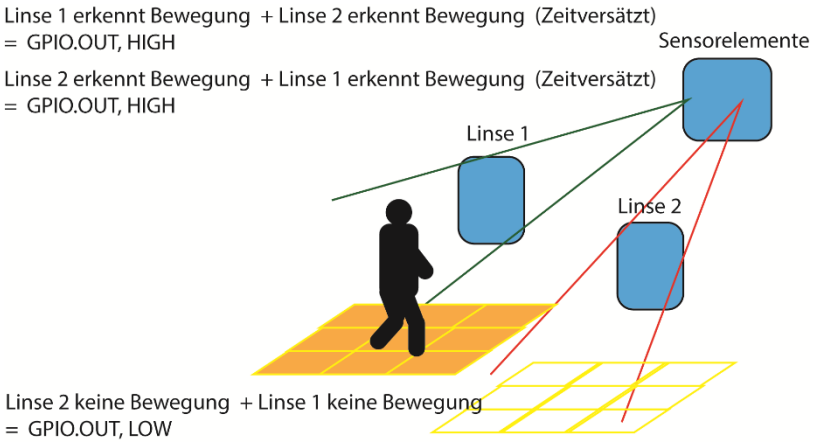


Abb24. Funktionsweise eines PIR Sensors

Bewegungssensor und Pi Kamera

Das letzte Beispiel funktioniert auf dieselbe Art und Weise, wie die Motorsteuerung und der Bewegungssensor. Wird Bewegung erkannt aktiviert die CSI-Kamera den Aufnahmemodus und zeichnet die Bewegung vor der Kamera auf. Die Dauer der Aufzeichnung ist abhängig von den eingestellten Sekunden. Die Videoaufnahme wird mit h264-Codec kodiert und mit dem Datum sowie der Uhrzeit in Motion-Verzeichnis abgelegt.

4.3.2 Entwicklung

Der GPIO-Port ist eine sehr mächtige Hardwareschnittstelle. Zusammen mit einer Programmiersprache (wie Python) kann es elektrische Bauelemente und extra dafür gebaute Schnittstellen steuern. Das Modul ist der Grundstein für alle weiteren Module, es vermittelt die Grundstruktur eines Python-Programms, den Umgang mit elektrischen Bauelementen und deren Steuerung.

4.4 Modul 4: SpyCam

Bei Modul 4 handelt es sich um ein Überwachungssystem, dass durch Bewegungserkennung aktiviert wird. Das System zeichnet x-Sekunden (je nach Einstellung) ein Video und speichert es auf einem NAS-Server ab. In dem gleichen Moment wird eine E-Mail an den Benutzer mit dem Inhalt, dass Bewegung registriert worden ist, verschickt.

4.4.1 Beschreibung Modul 4

An dem Raspberry Pi wird das Kameramodul mit einem 15poligen Flexkabel an die CSI-Schnittstelle angeschlossen. Die CSI-Schnittstelle überträgt das Video zum SoC, dort wird das Video codiert, ausgegeben und abgespeichert. Im Gegensatz zu einer USB-Web-Cam fällt die ständige USB Abfrage weg, was für eine saubere Übertragung sorgt. Für die Bewegungserkennung ist ein Bewegungssensor (PIR-Motion-Sensor) an die GPIO-Pins (5V, GND und ein Input-Pin) angeschlossen, der in einem bestimmten Radius (120 Grad) Bewegung erkennt und die Kamera aktiviert. Die Kamera startet mit der Aufzeichnung, (x-Sekunden) sobald der Sensor durch die Bewegung auf HIGH gesetzt wird. Gleich danach sendet das Programm eine E-Mail (optional SMS)

an das eingestellte Konto. Die Videos werden zuerst auf der SD-Karte aufgezeichnet und anschließend auf ein NAS-Laufwerk kopiert. Jedes Video wird mit Datum und Uhrzeit versehen und im h264 Codec abgespeichert. Das SpyCam-System verfügt über eine Einstellungsdatei (config.txt), in der Kammereinstellungen vorgenommen, E-Mail-Konto eingerichtet werden und der Speicherort für die Videos eingestellt wird. Das System kann auch durch ein Powerakku und in der Reichweite von W-LAN mobil eingesetzt werden.

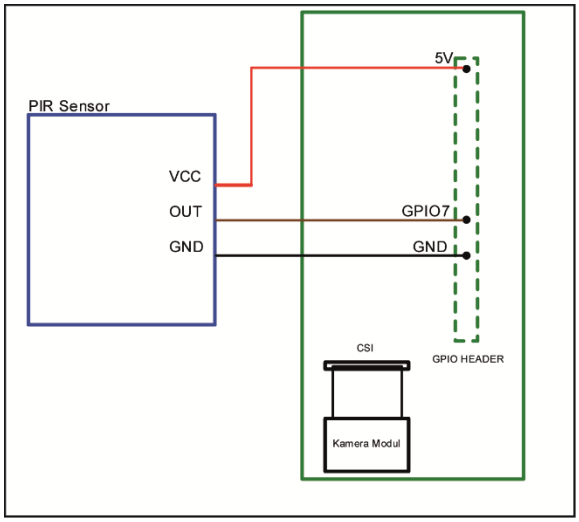


Abb25. Schaltung für eine Überwachungskamera die durch Bewegung vor dem PIR Sensor ausgelöst wird

4.4.2 Entwicklung

Das Überwachungssystem verbindet mehrere Schnittstellen in einem Programm, den GPIO-Port, das CSI, das Netzwerk und die Softwareschnittstelle miteinander. Der PIR-Sensor aktiviert die CSI-Kamera wodurch die E-Mail verschickt und das Video nach Beendigung der Aufzeichnung auf ein NAS-Server abgespeichert wird. Das Modul soll dem Anwender die Vielseitigkeit des aufgebauten Systems zeigen. Es ist möglich ein System zu entwickeln, dass über mehrere Schnittstellen kommuniziert und über eine Konfigurationsdatei editierbar ist. Die Schwierigkeit in diesem Modul lag darin, ein passendes und in sich geschlossenes System zu entwickeln, was sich über eine einzige Konfigurationsdatei editieren lässt.

4.5 Modul 5: GPIO Robotik

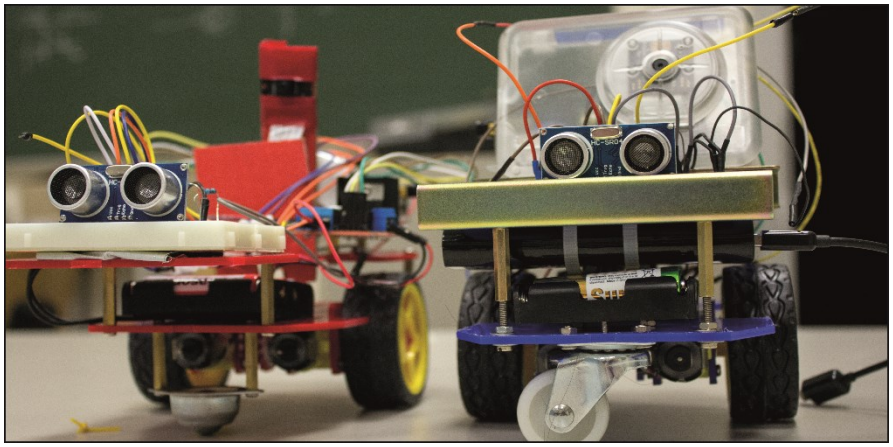


Abb26. Zwei unterschiedlich konstruierte GPIO-Roboter mit denselben Funktionen

GPIO ROBOT ist ein kleiner mobiler Roboter, der auf dem RasPi basiert. Gesteuert wird er durch zwei Motoren, die durch Hilfe der Pulsweitenmodulationen vorwärts, rückwärts, nach links oder rechts gelenkt werden können. An die CSI-Schnittstelle ist das Kamera Modul von RasPi angeschlossen, die drahtlos das Bild auf einen Webbrowser von Smartphone, Tablet oder Desktop-PC liefert. Zusätzlich kann der GPIO ROBOT über den Browser ferngesteuert werden. Der Roboter kann auch in einen Automodus versetzt werden. Mithilfe eines Ultraschallsensors kann das Gefährt Hindernisse erkennen und diesen durch programmierte Fahrmanöver ausweichen.

4.5.1 Beschreibung Modul 5

Bauplan

Herzstück des GPIO ROBOT ist der RasPi Modell B. Die Stromversorgung des RasPi´s übernimmt ein mobiler 5V-Powerakku mit 2Ah, der den RasPi über einen Micro-USB-Anschluss versorgt. Die beiden Motoren im Chassis werden durch 4 x AA Batterien mit insgesamt 6V mit Strom versorgt. Die Batterien befinden sich in einem Batteriehalter, der im unteren Bereich des Chassis befestigt ist. Am Chassis des Roboters befindet sich ein Schalter, mit dem die Motoren ein-/ausgeschaltet werden können. Die Kommunikation zwischen den Motoren und den GPIO-Pins übernimmt eine L298N H-Brücke, die über Jumperkabel verbunden sind. (siehe Anhang Schaltplan Motorsteuerung). Die Verbindung zwischen GPIO ROBOT und anderen Geräten erfolgt über WLAN.

MotoMama L298N H-Brücke

Die zwei Räder manövrieren bei der Lenkung nach links bzw. rechts mit unterschiedlicher Geschwindigkeit und einer gegenläufigen Bewegung. Dazu werden die beiden Gleichstrommotoren mittels der Pulsweitenmodulation (PWM) in der Geschwindigkeit reguliert. Da der Pi nur ein Hardware-PWM-Pin besitzt, übernimmt die Pulsweitenmodulation die H-Brücke problemlos. Und bei Bedarf lässt sich die Geschwindigkeit der Motoren im Programm jederzeit einstellen. Die Motorensteuerung übernimmt die eingebaute L298N H-Brücke, die von RasPi die Befehle für die Motoren bekommt. Die Steuersignale werden mittels der Pulsweitenmodulation in die H-Brücke eingespeist. Dabei gibt es fünf verschiedene Fahrmanöver (sieh Motorsteuerungsbefehle). Durch verschiedene Einwirkungen kann es passieren, dass ein Motor zu wenig Spannung bekommt und die beiden Motoren nicht synchron beim geradeaus- bzw. rückwärtsfahren sind. Durch das Verändern der Tastwerte kann die Synchronisation der beiden Motoren korrigiert werden.

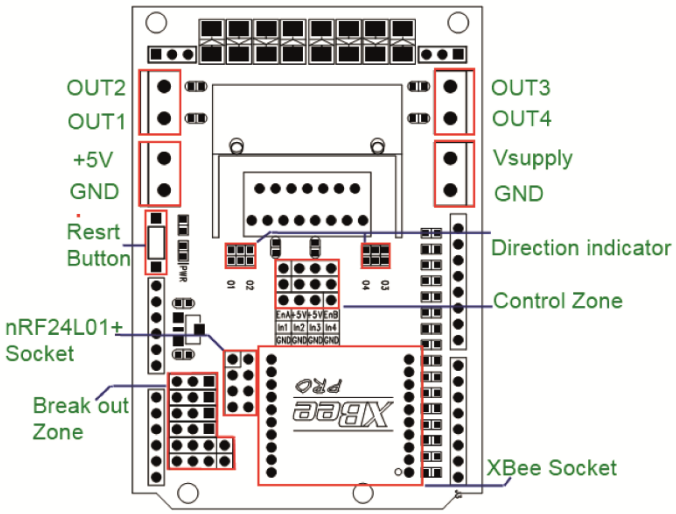


Abb27. MotoMama L298N H-Brücke Schemata

Um die zwei Gleichstrommotoren gleichzeitig aber unabhängig voneinander steuern zu können ist der L298 Treiber auf dem Board montiert. Dies geschieht mit Hilfe der Transistor-Transistor-Logik (TTL). An OUT1 und OUT2 wird der rechte Motor und an OUT1 und OUT2 der linke Motor montiert. Die vier OUT-Anschlüsse sind baugleich. Die Motorbrücke benötigt mindestens 5V für ein störungsfreies Arbeiten. Ein Batteriehalter mit 4 AA Batterien mit jeweils 1,5V, ist an den „Vsupply“ und GND angeschlossen. An das Plus Kabel von dem Batteriehalter wurde ein Kippschalter montiert, der es erlaubt, den Stromfluss zu den Motoren jederzeit zu unterbrechen. Die Befehle für die Steuerung der Motoren übergeben die Input-Ports. Input-Port A besitzt 6 Pins (befindet sich in der Control Zone der Abb27.), EnA (zwei Pins die für DC-Motoren überbrückt sind), IN1 und IN2 und 2x GND. IN1 und IN2 sind zwei digitale Pins die über den L298N-Chip die Befehle (Tastverhältnis) an die OUT1 und OUT2 übergeben. Nach der gleichen Methode funktionieren OUT3 und OUT4 zusammen mit dem Port B.

Motorsteuerungsbefehle

Die Fahrmanöver werden über die Pulsweitenmodulation an die Motoren übergeben. „*Vorteil dieser Ansteuerung ist, dass weniger Leistung verbraucht wird, da nicht permanent eine Eingangsspannung anliegt, die von einer Elektronik auf die gewünschte Motorspannung heruntergeregelt wird, sondern der Motor durch die Breite der Schaltimpulse gesteuert wird.*“[28]

Geschwindigkeit: Tastverhältnis für beide Motoren synchron

Unterschiedliche Tastverhältnis-Werte (in %) für die Geschwindigkeit

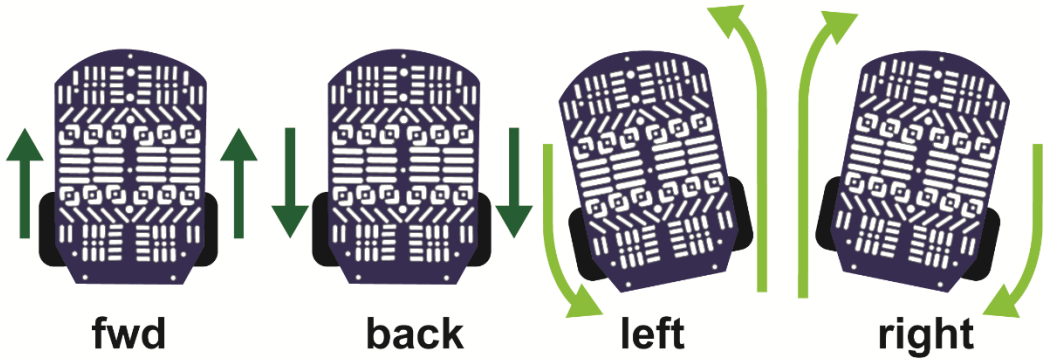


Abb28. Die vier Fahrmanöver, die über die Schaltimpulse gesteuert werden.

Die Fahrtrichtungen werden wie gefolgt deklariert:

fwd: GPIO ROBOT fährt vorwärts: Motor links und Motor rechts => Tastverhältnis von je 50% (L_FWD und R_FWD)

back: GPIO ROBOT fährt rückwärts: Motor links und Motor rechts => Tastverhältnis von je 50% (L_BWD und R_BWD)

left: GPIO ROBOT links: Der Motor links (L_BWD) dreht sich langsamer rückwärts und Motor rechts (R_FWD) dreht sich schneller vorwärts. Damit erreicht man eine geschmeidige Linksdrehung.

right: GPIO ROBOT rechts: Der Motor rechts (R_BWD) dreht sich langsamer rückwärts und der Motor links (L_FWD) dreht schneller vorwärts. Damit erreicht man eine geschmeidige Linksdrehung.

stop: GPIO ROBOT bleibt stehen: Beide Motoren werden auf 0 gesetzt.

Entfernungssensor / Ultraschalsensor HC-SR04

Der Automodus wird über einen Entfernungssensor gesteuert. Der Sensor misst die Distanz. Beträgt die Entfernung weniger als 50 cm so entscheidet der Zufall (Random-Funktion) in welche Richtung sich der GPIO ROBOT drehen soll. Der Entfernungssensor wird an einen 5V Pin von RasPi angeschlossen und nutzt eine Stromaufnahme von 15mA. Dabei muss der Echo Pin mit zwei 1kOhm (siehe Abb28.) geschützt werden, weil dieser ein 5V Spannungssignal an den GPIO-Pin sendet. Der Öffnungswinkel des Ultraschallsensors ist 15 Grad und die Reichweite beträgt 450cm bei einem

[28] http://rn-wissen.de/wiki/index.php/Pulsweitenmodulation

Anfangsmesswert von 2cm. Die Messgenauigkeit beträgt +/- 0,3cm und es können bis zu 50 Messungen in einer Sekunde durchgeführt werden[29]. Der Entfernungssensor hat 4 Pins, Vcc-, Trig-, Echo und ein GND-Pin. Der Vcc-Pin wird an den 5V Pin und der GND-Pin an den Masse-Pin von Raspberry angeschlossen. Der Trig wird als GPIO-OUT und der Echo-Pin als GPIO-IN deklariert.

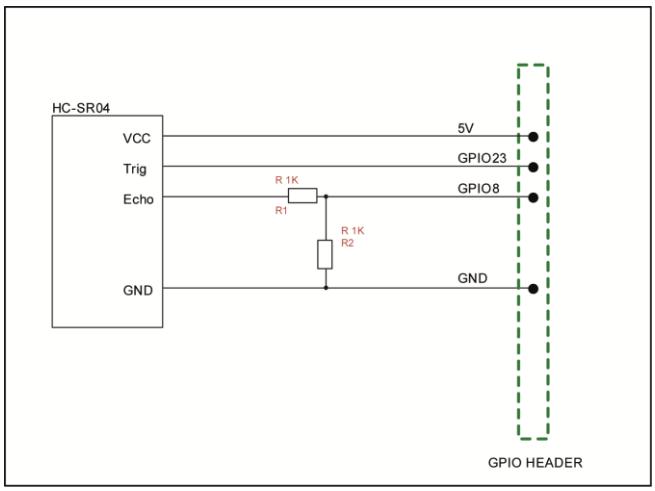


Abb29. Schaltplan: Ultraschalsensor angeschlossen am GPIO-Port

Funktionsweise des Sensors:

Der Raspberry Pi sendet ein 10 µs HIGH-Pegel Impuls an den Trigger-Pin, anschließend sendet dieser einen achtfachen 40kHz Ultraschall-Impuls (Burst-Signal) für die Dauer von 200µs. Danach wird der Echo-Pin aktiviert, der auf das Echo des 40kHz Ultraschall-Impulses wartet. Wenn das Echo für 200ms auf HIGH-Pegel verweilt, war die Messung erfolglos. Wird der Burst empfangen, so übermittelt der Echo-Pin die Informationen anschließend an den Raspberry. Das Echo sendet an den Pi ein Impuls, der je nach Dauer, die Entfernung zum Gegenstand bedeutet.

[29] ftp://imall.iteadstudio.com/Modules/IM120628012_HC_SR04/DS_IM120628012_HC_SR04.pdf

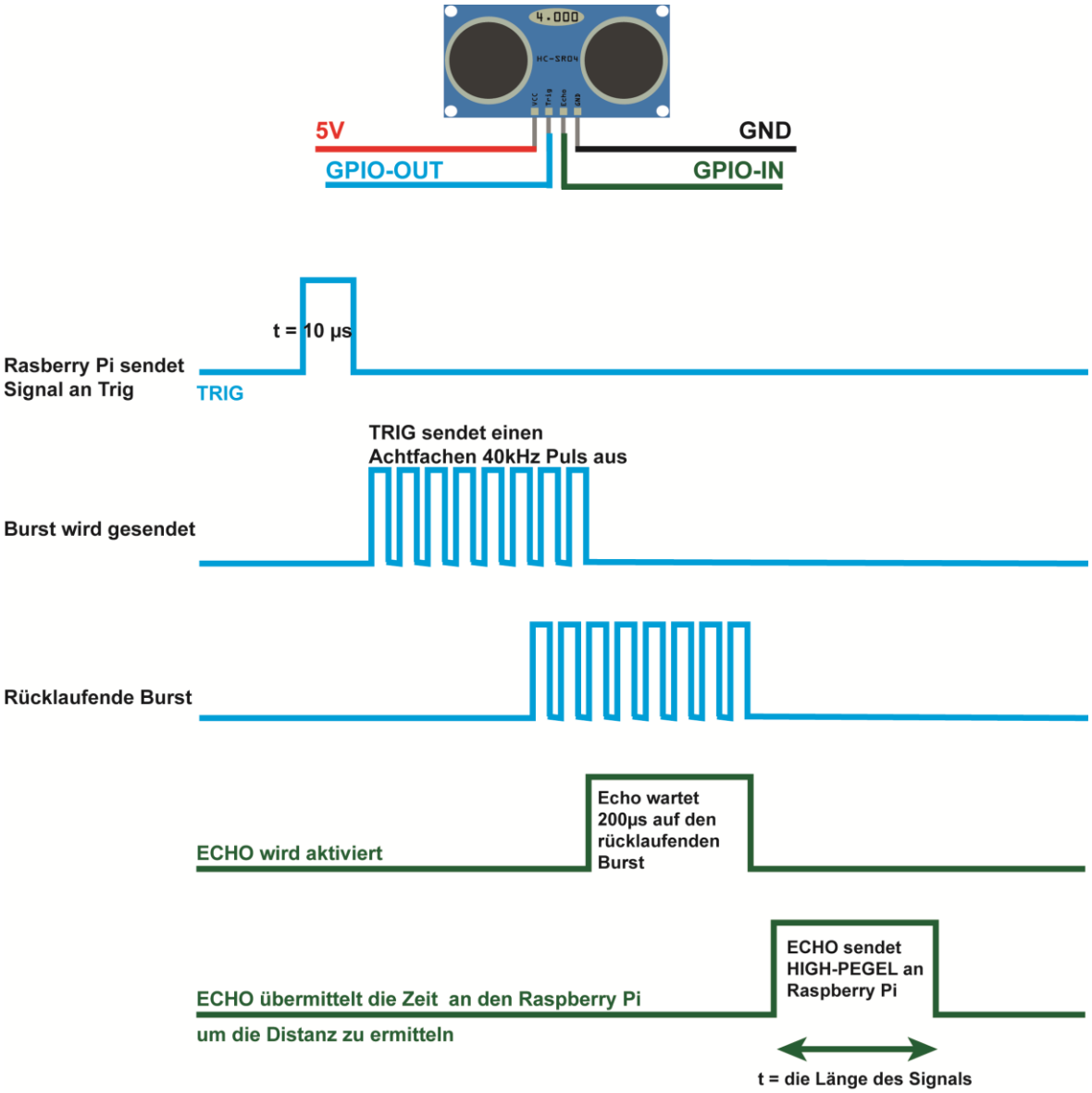


Abb30. Die Funktionalität des HC-SR04 Sensors

Um die Entfernung zu berechnen wird die Strecke des 40kHz Impuls vom Sender zum Gegenstand und zum Empfänger zurück mit einbezogen. Dabei beträgt die Schallgeschwindigkeit ca. 340m/s (wenn weitere Faktoren wie Temperatur ignoriert werden). Dadurch ergibt sich folgende Formel:

$$v = 2 * s/t$$

Diese Formel wird nach der Distanz umgestellt:

$$s = 170\frac{m}{s} * t$$

Software

In der Software werden die fünf Fahrmanöver implementiert, die an die H-Brücke übermittelt werden. Jede Fahrtrichtung hat seine eigene Funktion, die über das Terminal oder Weboberfläche an die Motoren weitergeleitet werden. Bei der Steuerung über den Browser erfolgt die Implementierung über den Webiopi-Server und dem in Python geschriebenen Programm. Im Automodus weicht der GPIO ROBOT den Hindernissen selbstständig aus.

Fernsteuerung durch den Webbrowser

Auf dem Raspberry Pi ist „Webiopi“ installiert, das über einen eigenen Webserver verfügt. Dieser übergibt die Befehle über die Clients von Python an die Java Clients von WebiOpi. Anschließend werden diese Befehle für die Motoren von der H-Brücke übersetzt und ausgeführt. Die auf den anderen Geräten installierten Webbrowser dienen als Steuerkonsole. Die Steuerkonsole wird über die *http://Domainname:Port* aufgerufen. Die angezeigte Webseite besteht aus sechs Elementen. Aus fünf Steuer-Buttons (Forward, Back, Left, Right und Stop) die im linken Bildschirmrand untereinander aufgelistet sind. Den größten Ausschnitt der Webseite nimmt das Bild der RasPi-Cam ein, die über den „mjpg-streamer“ in Echtzeit streamt. Die fünf Buttons nehmen die Steuerkommandos des Anwenders entgegen und übergeben diese für die Motorsteuerung weiter an die H-Brücke. Beim Drücken eines Knopfes wird ein Java-Code ausgeführt, der den Befehl für die Fahrtrichtung an das in Python geschriebene Programm weitergibt. In der Software wird das Manöver ausgeführt und an die Motoren übermittelt. Die Fahrmanöver sind fest einprogrammiert (siehe Motorsteuerung). Um eine flüssige Live Übertagung der Kamera zu garantieren, wurde die Qualität der Übertragung auf 50 Prozent gestellt und die Auflösung auf 800 x 600 Pixel mit 25 Bildern/s gesetzt.

Automodus

Im Automodus berechnet die Software die Entfernung zum Gegenstand und führt bei bestimmter Entfernung ein kurzes Fahrmanöver nach rechts bzw. nach links aus. Wenn sich der Roboter auf ein Hindernis zubewegt und die Distanz > 50cm wird, löst die Software eine Random-Funktion aus (Fahrmanöver links oder rechts) bei der die Fahrrichtung geändert wird. Ist die Entfernung größer als 50cm zum Gegenstand, fährt der GPIO ROBOT weiter gerade aus. Bleibt das Gefährt an einem Gegenstand hängen, signalisiert der Sensor nach kurzer Zeit ein Time Out und löst die Random-Funktion aus um die Fahrtrichtung zu ändern. Die Distanz zum Gegenstand wird mit der physikalischen Formel für die Strecke (siehe Funktionsweise des Sensors) berechnet und übermittelt die Werte an den Bildschirm (optional). Über einen internetfähigen Browser wird das Live-Bild der Kamera mittels „mjpg-streamer“ auf einem anderen Gerät angezeigt.

4.5.2 Entwicklung

Die Entwicklung dieses Moduls war viel anspruchsvoller, als alle anderen Systeme. Es musste ein Weg gefunden werden um einen Roboter einfach zu konstruieren, diesen verständlich zu dokumentieren und die Software nachvollziehbar zur programmieren.

Das Modul wird in vier Abschnitte unterteilt, der Bau eines Roboters, die Überprüfung der fehlerfreien Funktion der Treiber und Sensoren, die Steuerung des Roboters über die Weboberfläche und die automatische Bewegung des GPIO-ROBOTERS mit der Nutzung des Ultraschallsensors HC-SR04.

Der erste Abschnitt beschreibt den groben Aufbau des GPIO-ROBOT, die genaue Installation der Gleichstrommotoren von der H-Brücke bis zum GPIO-Port und den Zusammenbau der Schaltung für den Ultraschallsensor. Um den fehlerfreien Betrieb der Motorsteuerung und des HC-SR04 Sensors zu garantieren, werden diese im Abschnitt zwei mit einer speziell dafür entwickelten Software getestet. Im dritten Abschnitt werden die RasPi-Cam und der „mjpg-streamer“ aktiviert und mit der Steuerungssoftware „webiopi“ gestartet. Gesteuert wird der Roboter über eine HTML-Seite, die sich auf dem Raspberry Pi befindet via einem Smartphone, Tablet oder Computer. Im letzten Teil des Moduls wird der mobile GPIO-ROBOT in einem Autonomiemodus gestartet, in dem er sich ohne externe Steuerung fortbewegt.

4.6 Modul 6: Trellis Finger Tipp Reaktionsspiel

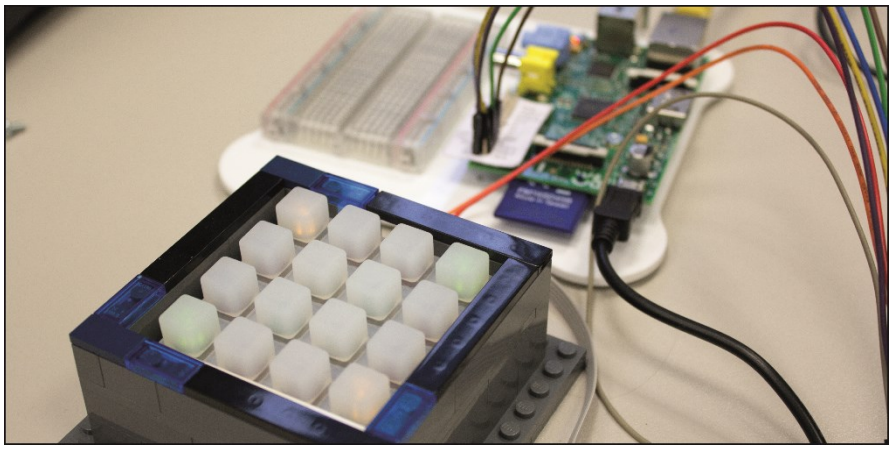


Abb31. Das Finger Tipp Reaktionsspiel – Gehäuse aus Legobausteinen konstruiert

In diesem Modul wurde ein Reaktionsspiel entwickelt. Ziel des Spiels ist es, nacheinander aufleuchtende Muster so schnell wie möglich auszuschalten. Die Figuren sind in einem Feld (Array) gespeichert und werden per Zufall auf das Trellis-Pad geladen. .

Als Vorlage für das Finger Tipp Reaktionsspiel diente das von „Tiger Electronics“ entwickelte „Lights Out“ aus dem Jahr 1995. Bei „Lights Out“ werden durch Zufall eine Anzahl von Lichtern eingeschaltet, die mit wenigen Knopfdrücken ausschalten werden müssen. Je weniger Versucher desto mehr Punkte.[30]

4.6.1 Beschreibung Modul 5

Die Trellis Platine ist über 5 Jumpe-Kabel mit dem den Raspberry Pi verbunden:

- **5V** Kabel für die Stromversorgung der Platine wird am 5V Pin angeschlossen.
- Die Masse (**GND**) wird an einer beliebiegen GND-Pin angeschlossen
- **SDA** wird an dem I²C-Data-Pin (GPIO2) angeschlossen
- **SCL** wird an den I²C-Clock-Pin (GPIO3) angeschlossen
- Der Interrupt (**INT**) wird mit einen beibiegen Input/Output Pin verbunden

Die 16-LEDs und Buttons der Platine werden über den I²C-Bus (SDA, SCL) gesteuert. Dabei ist zu beachten, dass der Interrupt-Pin in diesem Modul keine Rolle spielt.

Vorbereitung:

Standardmäßig ist der I²C-Bus auf dem Raspberry Pi deaktiviert. Um es zu aktivieren werden zwei Tools (`I2C-tools` und `python-smbus`) auf dem Pi installiert. Nach der Installation müssen Systemdateien editiert werden. In der Datei „modules“ (Pfad: `/etc/modules`), werden zwei Kommandos eingetragen (`i2c-bcm2708 ` und `i2c-dev`), die I²C-Bus aktivieren. Im nächsten Schritt wird der SPI- und I²C-Bus aus der Blackliste der SoCs aktiviert. Die „Blacklist“ ist unter `/etc/modprobe.d/raspi-blacklist.conf` zu finden. Um die Schnittstellen zu aktivieren, werden der SPI- und I²C-Bus mit der „#“-Raute auskommentiert.

Das Unternehmen Adafruit stellt für seine Produkte extra zugeschnittene Bibliotheken für verschiedene Programmiersprachen bereit. Im Falle von Trellis Keypad werden die Module für

[30] http://en.wikipedia.org/wiki/Lights_Out_%28game%29

Python heruntergeladen und installiert. Damit garantiert Adafruit eine reibungslose Funktionalität der eigenen Produkte.

Software:

Das Fingertipp Reaktionsspiel funktioniert nur, wenn sich die "Adafruit_I2C.py" Datei in demselben Ordner, wie das Spiel befindet. Die Software des Reaktionsspiels wählt per Random-Funktion aus einem Array ein Muster. Die Eingabe des Array wird in eine temporäre Variable abgespeichert und auf die Trellis-Platine eingespeist. Drückt der Anwender die eingeschaltete LED auf der Platine aus, so wird das geladene Array mit der temporären Variable abgeglichen. Die Zahl für die ausgeschaltete LED wird anschließend aus der temporären Variable gelöscht. Sind alle LEDs des Musters aus, lädt die Software per Zufall die nächste Figur aus dem Array. Die Funktionalität der LEDs und der Buttons, die nicht zum Muster gehören, bleiben während der geladenen Figur deaktiviert.

4.6.2 Entwicklung

Das Modul soll die Funktionalität eines I²C-Buses aufzeigen. Mit der Schnittstelle lassen sich problemlos die LEDs und Buttons des Boards ansteuern. Die LEDs mussten an die Platine gelötet werden und ein Gehäuse aus Lego wurde konstruiert. Während der Software Entwicklung entstand ein kleines Problem mit dem Array. Zwar wurde ein Muster aus dem Array geladen aber nachdem es ausgeschaltet wurde, konnte die Software kein weiteres laden. Das Problem wurde mit einer der temporären Variable gelöst. Das Fingertipp Reaktionsspiel ist so aufgebaut, dass der Anwender die Software ausbauen kann (z.B. neue Arrays, Timer usw.).

4.7 Modul 7: BrickPi Robot

Dank der hohen Prozessor-Leistung und dem 512 MB großen Speicher kann der Raspberry Pi Modell B in Verbindung mit BrickPi bis zu 4 Motoren und 4 Sensoren anschließen. Lego, RasPi und BrickPi bieten zusammen viele Kreativitätsmöglichkeiten. Ob Roboter, Maschinen oder nur ferngesteuerte Autos, vieles ist möglich. In diesen Modul wird ein Roboter gebaut, der in einen Automodus versetzt wird. Für die Umgebungserkennung wird ein Ultraschallsensor genutzt.

Der Vorteil gegenüber der normalen GPIO-Steuerung ist, die Konstruktionen des Lego-Roboters ist, er ist schnell veränderbar. Jeder Motor und Sensor benötigt nur ein Verbindungskabel, das an den BrickPi angeschlossen wird.

Das BrickPi-Board wird auf die Pins des GPIO-Ports aufgesetzt. Der BrickPi auf dem Raspberry Pi Modell B fungiert dabei als Kommunikationsschnittstelle zwischen den Lego-Sensoren / Motoren und den GPIO-Pins. Der BrickPi und der Raspberry werden mit 8x AA Batterien mit insgesamt 12 V betrieben (9V-12V empfiehlt der Hersteller). Die Batterien befinden sich in einem Batteriehalter, der mit Lego über dem Gehäuse von Raspberry Pi und BrickPi befestigt werden kann. Die Sensoren und Motoren werden über die Lego NXT-Anschlüsse mit einem Kabel verbunden, dass Ähnlichkeit mit RJ-45 Stecker hat. Der Berührungssensor und der Ultraschallsensor nutzen dank einem Regulator auf dem BrickPi-Board 5V. Diese Sensoren werden über den Mikrokontroller gelesen, dabei spielt es keine Rolle ob der Sensor Analog oder Digital ist. Ein Motortreiber der Marke „Texas Instruments SN754410“ steuert die Motoren des Gefährts. Die Geschwindigkeit der Motoren wird über die sog. Pulsweitenmodulation geregelt und die Spannungsversorgung ist mit 9V datiert.

4.7.1 Entwicklung

BrickPi nutzt alle Bus-Schnittstelen des Raspberry Pis. Diese werden von den verschiedenen Lego Sensoren benötigt. „Dexter Industries“ stellt eine eigene Raspbian Distribution dar (s.o), die auf das BrickPi Board zugeschnitten ist um dem Anwender stundenlange Konfiguration zu ersparen. Das Modul 7 baut auf dieser Distribution auf.

Im ersten Schritt wurde ein Roboter aus Lego gebaut, der drei Motoren nutzt, zwei Berührungssensoren und einen Ultraschallsensor. Der Roboter musste so konstruiert werden, dass der BrickPi leicht zugänglich ist, um weitere Bauelemente anzuschließen. Ein weiteres Kriterium ist, dass der Batteriehalter einfach zugänglich ist, damit die Batterien ohne Probleme ausgewechselt werden können.

Die Präsentationssoftware wurde in Python geschrieben, die alle sechs elektrischen Lego-Bauelemente nutzt.

5. Schlussfolgerung

Die Entwicklung des Baukastensystems war ein langwieriger Prozess. Zuerst mussten verschiedene Themen für die einzelnen Module gefunden und praktisch umgesetzt werden. Die Suche nach geeigneten Themen wurde durch Brainstorming, recherchieren im Internet, in Magazinen und Fachliteratur durchgeführt. Des Weiteren wurde der Bekanntenkreis befragt und einige Foren nach Interessen der Mitglieder durchsucht. Dank der Vielseitigkeit des Raspberry Pi und den Recherchen konnten für die Master Thesis einige interessante Ansätze gefunden werden. Aus diesen Ansätzen konnten dann fertige Themenbereiche für die Module, wie Mediacenter, GPIO Grundlagen, GPIO Robot usw. erstellt werden. Der Hauptschwerpunkt bei der Entwicklung der Module wurde auf die einfache und verständliche Sprache/Durchführung gelegt. *„Der Unterschied zu Windows oder Macintosh ist, dass man sich unter den Linux-Distributionen viel mit der Kommandozeile auseinander setzen muss.“* [31] Wegen der Umstellung vieler Anwender von Windows OSx auf Linux wurde das Modul 1 so ausgelegt, dass es dem Nutzer die Grundlagen des Betriebssystems vermittelt. Die Anwender lernen dabei wichtige Befehle, die Struktur des Systems und einige Einstellungsmöglichkeiten kennen (s.o.). Im nächsten Schritt musste der Übergang von einem Modul auf das Nächste gefunden werden. Nach einigen Überlegungen wurden die Module so aufgebaut, dass jedes Modul auf dem Nächsten aufbaut und den Schwierigkeitsgrad erhöht. Die Ergebnisse der Entwicklung des Baukastensystems sind positiv zu bewerten. Eine Ausnahmestellt das Modul RasPi ROBOT, Spycam und der BrickPi Lego Roboter dar, da diese gegenwertig nicht das System updatet und ungeradet werden sollte. Nach der Aktualisierung des Systems konnte bei den Robotern und der Spycam die RasPi-Cam nicht mehr gestartet werden oder der MJPG-Streamer konnte nicht mehr ausgeführt werden. Das Problem liegt an den Änderungen des Systems von Raspbian und den Programmen der Drittanbieter. Es stellt das ganze Baukastensystem nicht vor große Probleme, denn ohne die Systemaktualisierung, funktionieren alle Module einwandfrei. „Never change the running System“ lautet auch im Baukastensystem die Devise. Fehler die entstehen könnten, sind zum größten Teil auf falsch angeschlossene oder defekte Hardware zurückzuführen.

Alle Module wurden drei bis vier Mal mit verschiedenen Raspberry Pi´s einem Probelauf unterzogen. Der Testlauf ist ohne jegliche Funktionsstörungen erfolgreich durchgeführt worden. Das Modul 1

---

[31] Michael Kos, Verschiedene Projekte mit Raspberry Pi, S.42

wurde auch in einer Vorlesung von Studenten durchgeführt. Die Studenten wurden in 3 Gruppen mit jeweils zwei Personen aufgeteilt und sollten das Modul 1 Schritt für Schritt durchführen. Alle drei Gruppen haben das Modul erfolgreich abgeschlossen. Eine der Gruppen führte sogar das Modul fehlerfrei durch. Dabei sei zu beachten, dass diese zwei Probanden den Raspberry Pi zum ersten Mal bedient haben. Die anderen zwei Gruppen, die einen eigenen Raspberry Pi besitzen oder einen genutzt hatten, standen während der Durchführung vor kleinen Problemen. Diese Probleme sind aber darauf zurückzuführen, dass diese Probanden die Anleitung des Moduls nicht ausführlich gelesen bzw. befolgt und dabei einige Details übersehen haben. Das Resultat dieses Versuchs ist, wird die Anleitung jedes Moduls „Step by Step“ befolgt, so funktionieren die Systeme einwandfrei ohne Schwierigkeiten.

Leider konnte der am 2. Februar 2015 rausgebrachte Raspberry Pi 2 keine große Rolle in der Master Thesis spielen. Das ist darauf zurückzuführen, dass die ersten Bestellungen längere Lieferzeiten mit sich brachten. Die Module „Raspbian Grundlagen“, „OpenELEC“ und „GPIO-Grundlagen“ können aber ohne Probleme auf den Raspberry Pi 2 übertragen werden, denn bei diesen Modulen funktioniert die Implementierung ohne weiteres. Der Grund dafür ist, dass die Distribution Raspbian und OpenELEC für den Pi 2 extra von den Entwicklern zugeschnitten wurde. Die GPIO-Grundlagen wurden zusätzlich von einer dritten Person auf der neuen Hardware getestet. Die folgende Tabelle, zeigt welche der Module auf dem RasPi 2 funktionieren und welche verändert werden müssen.

Modul	Implementierbar	Grund
Modul 1	Ja	Die zu installierende Pakete funktionieren ohne Probleme.
Modul 2	Ja	Eine neue Distribution mit denselben Funktionen wurde von den Entwicklern herausgebracht und funktioniert ohne Probleme.
Modul 3	Ja	Die in Python geschriebene Software funktioniert mit der GPIO-Schnittstelle einwandfrei. Raspberry Pi 2 unterstützt alle elektrischen Bauelemente.
Modul 4	Ja	Die in Python geschriebene Software funktioniert mit der GPIO-Schnittstelle einwandfrei. Raspberry Pi 2 funktioniert fehlerfrei mit der RasPi-Cam.
Modul 5	Nicht vollständig	Den Roboter über das Terminal steuern funktioniert fehlerfrei. Der Automodus ist voll funktionsfähig. Der MJPG-Streamer wurde geändert und funktioniert nicht mehr nach demselben Aufbauprinzip. Die Funktion der Kamera müsste umgebaut werden. WebiOpi konnte aus Zeitgründen nicht mehr getestet werden.
Modul 6	Möglich	Der I²C-Bus hat sich im Raspbian für den Pi 2 verändert. Beim Umbau auf den Raspberry Pi 2 müsste die neue Struktur und Tools herausgefunden werden.
Modul 7	Möglich	Für BrickPi auf dem Raspberry Pi 2 stand noch keine Distribution von den Entwicklern bereit. Lösung: Erstellung einer eigenen Distribution mit der Anleitung von „Dexters Industries“ - Dies nimmt viel Zeit in Anspruch.

Tabelle 12 Modulimplementierung auf Raspberry Pi 2

Das Schöne an diesem Baukastensystem ist, jedes Modul baut auf dem Anderen auf. Durch diesen Baukastenmodus wird das Wissen, die Erfahrungen in Linux, im Programmieren und in der Hardwareanwendung nach jedem Modul ausgebaut. Das Ziel der Master-Thesis war eine Art Baukastensystem zu entwickeln, dass mit wenigen Schritten aufgebaut wird und nach Belieben

erweiterbar ist. Es soll dem Anwender in leicht verständlicher Sprache Linux Python und die Kombination zwischen Hard- und Software vermitteln. Dieser Mix aus software- und hardwareorientierter Schulungsanleitung soll dem Anwender schnell das wichtigste „Knowhow“ vermitteln und das Interesse am informationstechnischen Bereich wecken. Nach der vollständigen Durchführung des Baukastensystems müssten die Anwender in der Lage sein, eigene Projekte zu starten oder die Module weiter auszubauen.

Das Baukastensystem ist ein guter Einstieg in die Welt der Informationstechnik. Die Module lassen sich schnell durch die kurzen Texte und bildlichen Darstellungen aufbauen und sind einfach zu verstehen. Das Baukastensystem könnten schon Schüler mit zwölf Jahren oder jüngeren Alters mit ein wenig Computererfahrung durchführen.

Das Baukastensystem hätte noch mehr Platz für weitere Module gehabt, doch leider konnte durch die lange Entwicklung der einzelnen Module, einige Themen keinen Platz finden. Ein sehr interessantes Thema ist z.B. die Haussteuerung mit dem Raspberry Pi. Nach gründlichen Recherchen und Ausprobieren, musste das Thema aufgegeben werden. Denn die Fachkenntnisse haben nicht ausgereicht, um dieses Vorhaben durchzusetzen. Die Möglichkeit bestand, eine Hausteuerung mit WLAN-gesteuerter-Hardware durchzuführen, aber dafür wären Lichtschalter und Steckdosen einer sehr preishohen Marke von Nöten gewesen. Eine andere Variante bestand darin, einen Schaltkasten zu simulieren, der in einem Haus die gesamte Elektronik steuert. In diesem Fall fehlten leider die Programmierkenntnisse.

Anhand dieser Beispiele lässt sich erkennen, dass das Baukastensystem noch um weitere Module erweiterbar ist, denn der Raspberry Pi lässt in seiner „Offenheit“ fast keine Grenzen. An diesem Punkt kann weiter angesetzt werden und vielleicht bald auch ein Haus konstruiert werden, das nur durch den Raspberry Pi gesteuert wird. Dank der neuen Generation vom Raspberry Pi und des Baukastensystem werden sich vielleicht bald mehr junge Menschen für Informationstechnik interessieren. Denn die Zukunft gehört der Informationstechnik.

6. Verzeichnisse

Sekundärliteratur

Bernd Klein, Einführung in Python 3, Carl Hanser Verlag München 2013

Hacks für Raspberry Pi, R. Suehle & T. Callaway 2014

Kofler Michael, Linux – Das umfassende Handbuch, 1 Auflage Galileo Press, Bonn 2014

Bartmann Erik, Durchstarten mit Raspberry Pi; 2012 by O'Reilly Verlag GmbH & Co. KG, 1. Auflage 2012

Michael Kos, Verschiedene Projekte mit Raspberry Pi, Belegarbeit Forschungs- und Entwicklungsmodul 2014

Richardson Matt & Wallace Shawn, Getting Started with Raspberry Pi, 2012 O'Reilly Media, Inc.

Magazine

2013: Raspberry Pi GEEK; Heft 05/2013

2013: Raspberry Pi GEEK; Heft 06/2013

2014: Raspberry Pi GEEK; Heft 01/2014

2013: Das ultimative Raspberry Pi Handbuch 02/2014

Internet

Die Erfolgsgeschichte des Scheckkartencomputers Raspberry Pi

http://www.gruenderszene.de/allgemein/raspberry-pi-eben-upton [Stand: 20.04.2015]

Compute Module ist lieferbar

http://www.golem.de/news/raspberry-pi-compute-module-ab-sofort-lieferbar-1406-106550.html [Stand: 20.04.2015]

Downloads OS-Images

http://www.raspberrypi.org/downloads/ [Stand: 20.04.2015]

Raspberry Pi 2 ausprobiert

http://www.golem.de/news/raspberry-pi-2-schnell-rechnen-langsam-speichern-1502-112107.html [Stand: 20.04.2015]

Raspberry Pi 2 ausprobiert

http://www.golem.de/news/raspberry-pi-2-schnell-rechnen-langsam-speichern-1502-112107-3.html [Stand: 20.04.2015]

Banana Pi im Test

http://www.golem.de/news/banana-pi-im-test-bananen-sind-keine-himbeeren-1408-108314.html [Stand: 20.04.2015]

BeagleBoard Wiki

http://elinux.org/Beagleboard:BeagleBoneBlack [Stand: 20.04.2015]

Arduino Yun

http://arduino.cc/en/Main/ArduinoBoardYun [Stand: 20.04.2015]

Intel Edison ausprobiert

http://www.golem.de/news/intel-edison-ausprobiert-ich-seh-dich-das-mona-lisa-projekt-1411-110577.html

[Stand: 20.04.2015]

Mobile DRAM K4P4G324EB

http://www.samsung.com/global/business/semiconductor/product/mobile-dram/detail?productId=7609

[Stand: 20.04.2015]

Die Python Geschichte

http://www.python-kurs.eu/entstehung_python.php [Stand: 20.04.2015]

Ubuntuusers Verzeichnisstruktur - Wiki

http://wiki.ubuntuusers.de/Verzeichnisstruktur [Stand: 20.04.2015]

Passiver-Infrarot-Bewegungsmelder

http://reglomat.bircher.com/uploads/media/PIR_d_01.pdf [Stand: 20.04.2015]

PIR-Passiv-Infrarot-Sensoren

http://rn-wissen.de/wiki/index.php/Sensorarten#PIR_Passiv_Infrarot_Sensoren [Stand: 20.04.2015]

Pulsweitenmodulation

http://rn-wissen.de/wiki/index.php/Pulsweitenmodulation [Stand: 20.04.2015]

HC-SR04

ftp://imall.iteadstudio.com/Modules/IM120628012_HC_SR04/DS_IM120628012_HC_SR04.pdf [Stand: 20.04.2015]

Lights Out - Wiki

http://en.wikipedia.org/wiki/Lights_Out_%28game%29 [Stand: 20.04.2015]

Raspberry Pi 2

https://www.raspberrypi.org/raspberry-pi-2-on-sale/ [Stand: 20.04.2015]

XBMC to Kodi
http://www.heise.de/newsticker/meldung/Aus-XBMC-wird-das-Kodi-Entertainment-Center-2282292.html [Stand: 20.04.2015]

Weitere Links

http://raspberrypiguide.de/stuff/bcm2835-arm-peripherals.pdf [Stand: 20.04.2015]

http://www.raspberrypi.org/raspberry-pi-compute-module-new-product/ [Stand: 20.04.2015]

http://www.lemaker.org/resources/9-38/image_files.html *[Stand: 20.04.2015]*

http://beagleboard.org/latest-images *[Stand: 20.04.2015]*

https://communities.intel.com/docs/DOC-23242 *[Stand: 20.04.2015]*

https://www.adafruit.com [Stand: 20.04.2015]

www.dexterindustries.com *[Stand: 20.04.2015]*

http://elinux.org/RPi_Low-level_peripherals [Stand: 20.04.2015]

http://elinux.org/RPi_USB_Wi-Fi_Adapters [Stand: 20.04.2015]

http://dlnmh9ip6v2uc.cloudfront.net/datasheets/Dev/RaspberryPi/BrickPi%201.7.3.pdf
[Stand: 20.04.2015]

http://www.dexterindustries.com/BrickPi/getting-started/pi-prep/ *[Stand: 20.04.2015]*

https://learn.adafruit.com/adafruit-trellis-diy-open-source-led-keypad *[Stand: 20.04.2015]*

https://learn.adafruit.com/trellis-python-library *[Stand: 20.04.2015]*

https://github.com/raspberrypi/noobs/blob/master/README.md [Stand: 20.04.2015]

https://code.google.com/p/webiopi/ *[Stand: 20.04.2015]*

https://us.pycon.org/2015/[Stand: 20.04.2015]

https://wiki.python.org/moin/OrganizationsUsingPython [Stand: 20.04.2015]

http://de.wikibooks.org/wiki/Linux-Praxisbuch:_Verzeichnisse_unter_Linux [Stand: 20.04.2015]

http://wiki.ubuntuusers.de/Verzeichnisstruktur [Stand: 20.04.2015]

http://en.wikipedia.org/wiki/Filesystem_Hierarchy_Standard [Stand: 20.04.2015]

http://wiki.ubuntuusers.de/Benutzer_und_Gruppen [Stand: 20.04.2015]

http://openbook.galileo-press.de/linux/linux_kap13_002.html#dodtp202e5804-54bd-499e-8ef0-c16a0416d741 [Stand: 20.04.2015]

<2015>

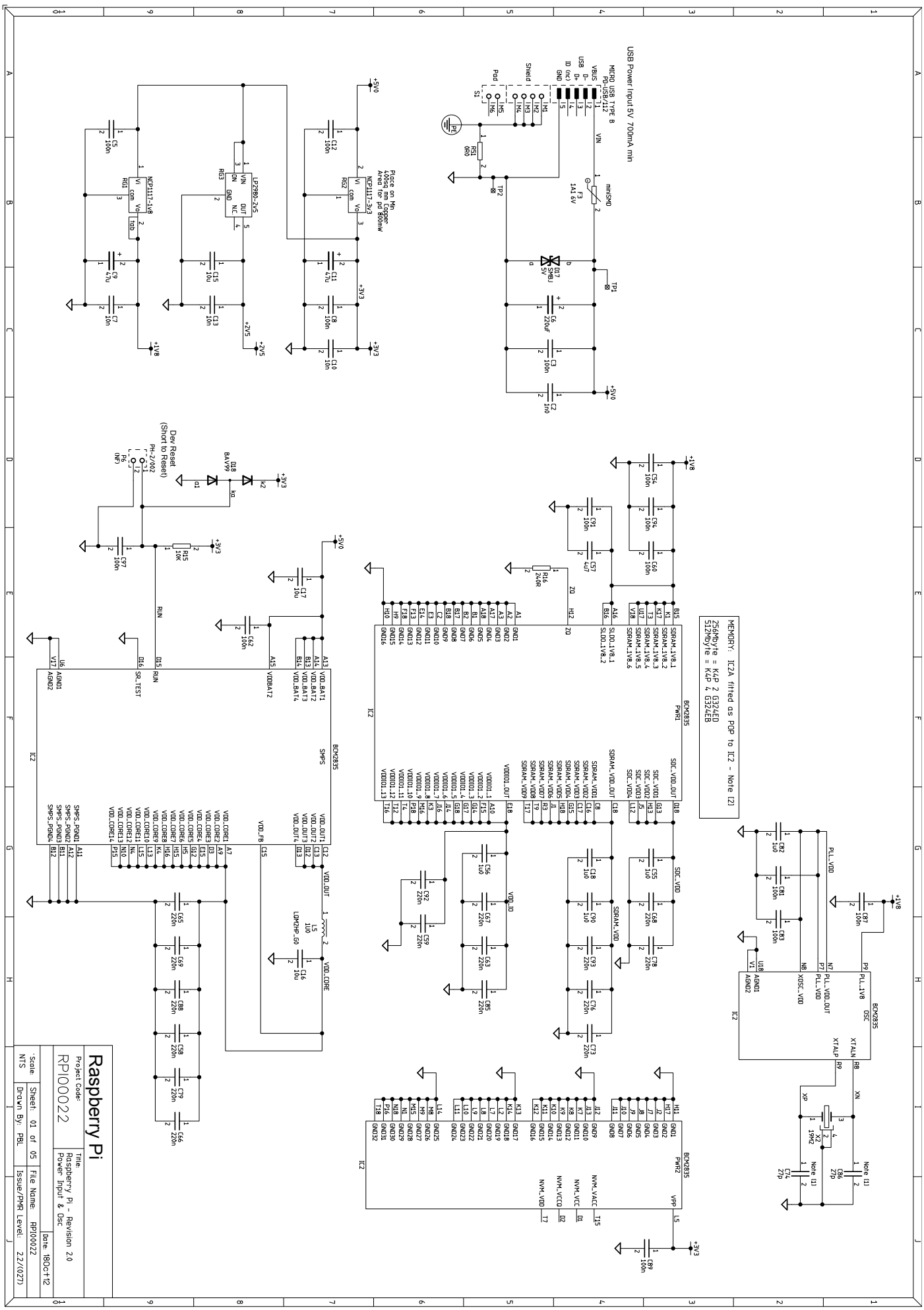
Eigenständigkeitserklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und nur unter Verwendung der angegebenen Literatur und Hilfsmittel angefertigt habe. Stellen, die wörtlich oder sinngemäß aus Quellen entnommen wurden, sind als solche kenntlich gemacht. Diese Arbeit wurde in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegt.

Mittweida 28.04.2015

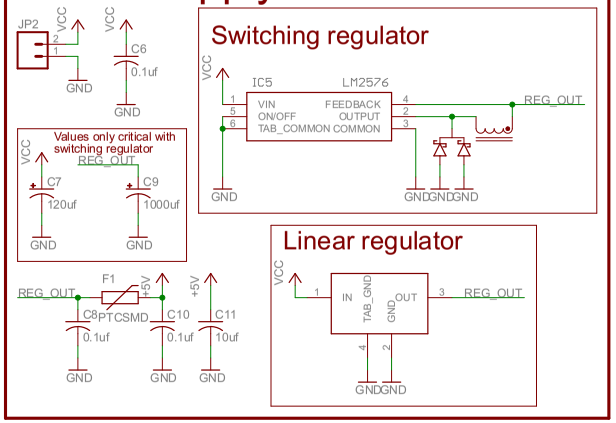
Ort, Datum Vorname, Nachname

ANHANG: Raspberry Pi

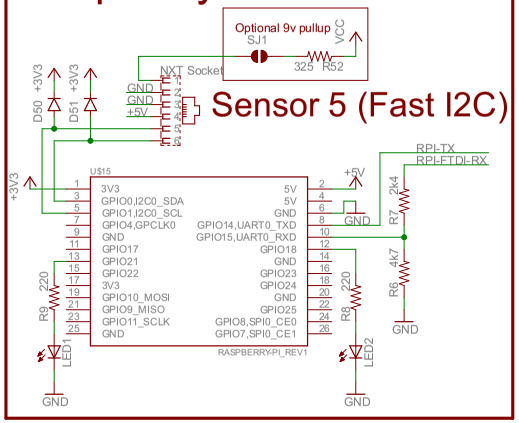


ANHANG: BrickPi

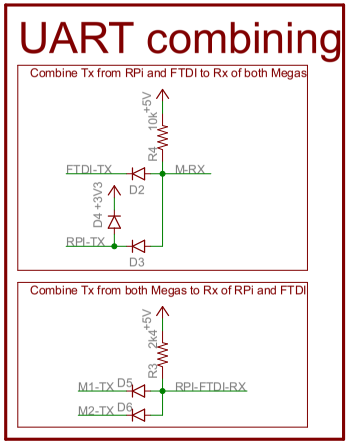
Power supply



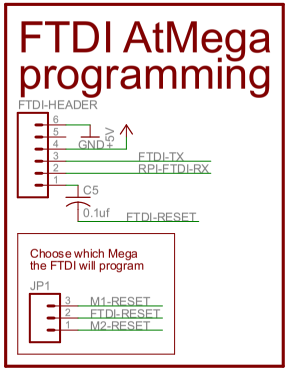
Raspberry Pi interface



UART combining

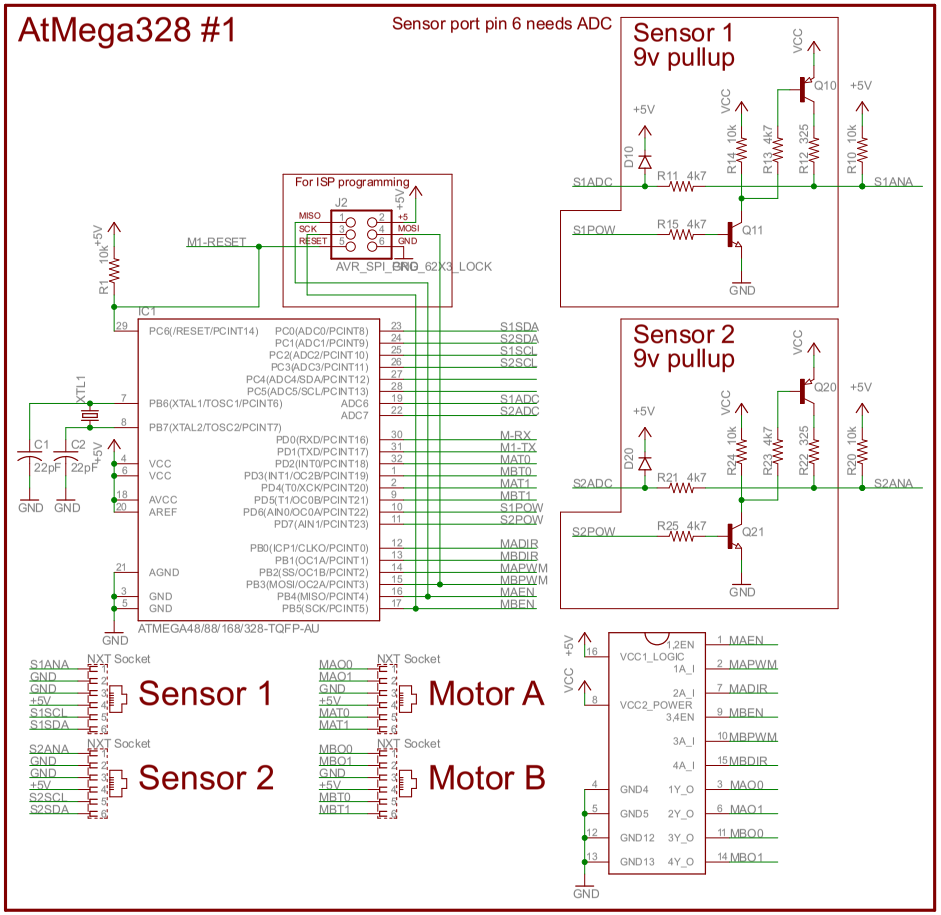


FTDI AtMega programming



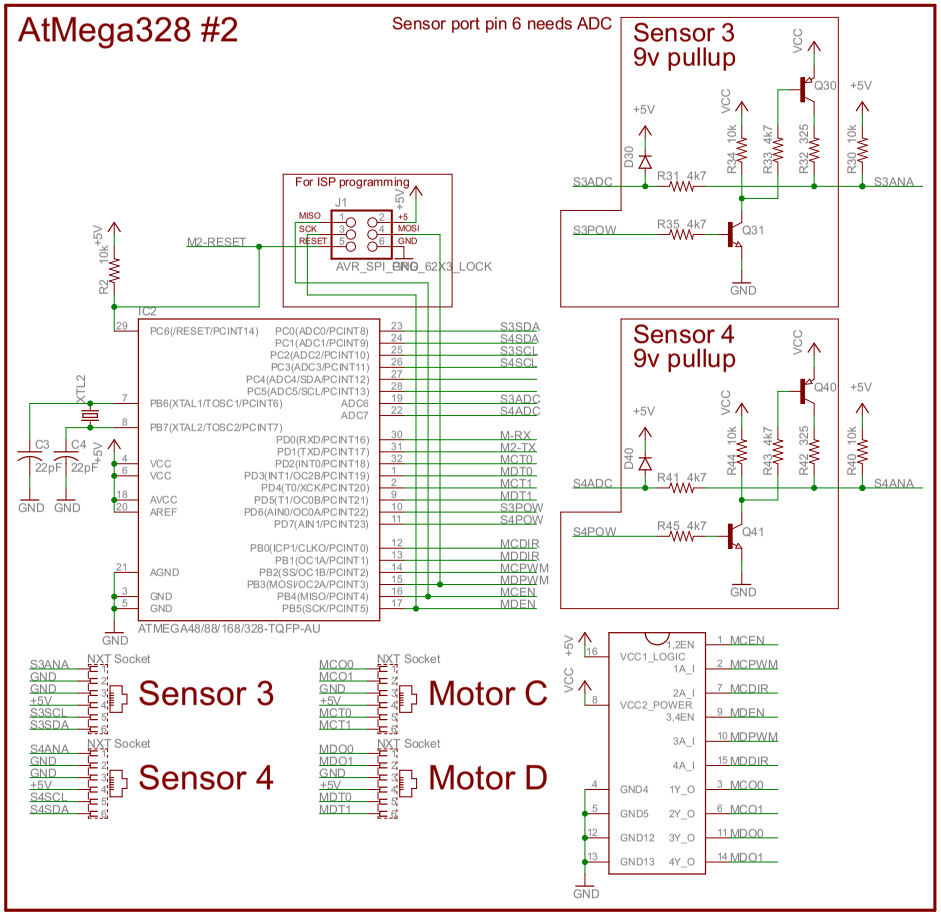
AtMega328 #1

Sensor port pin 6 needs ADC

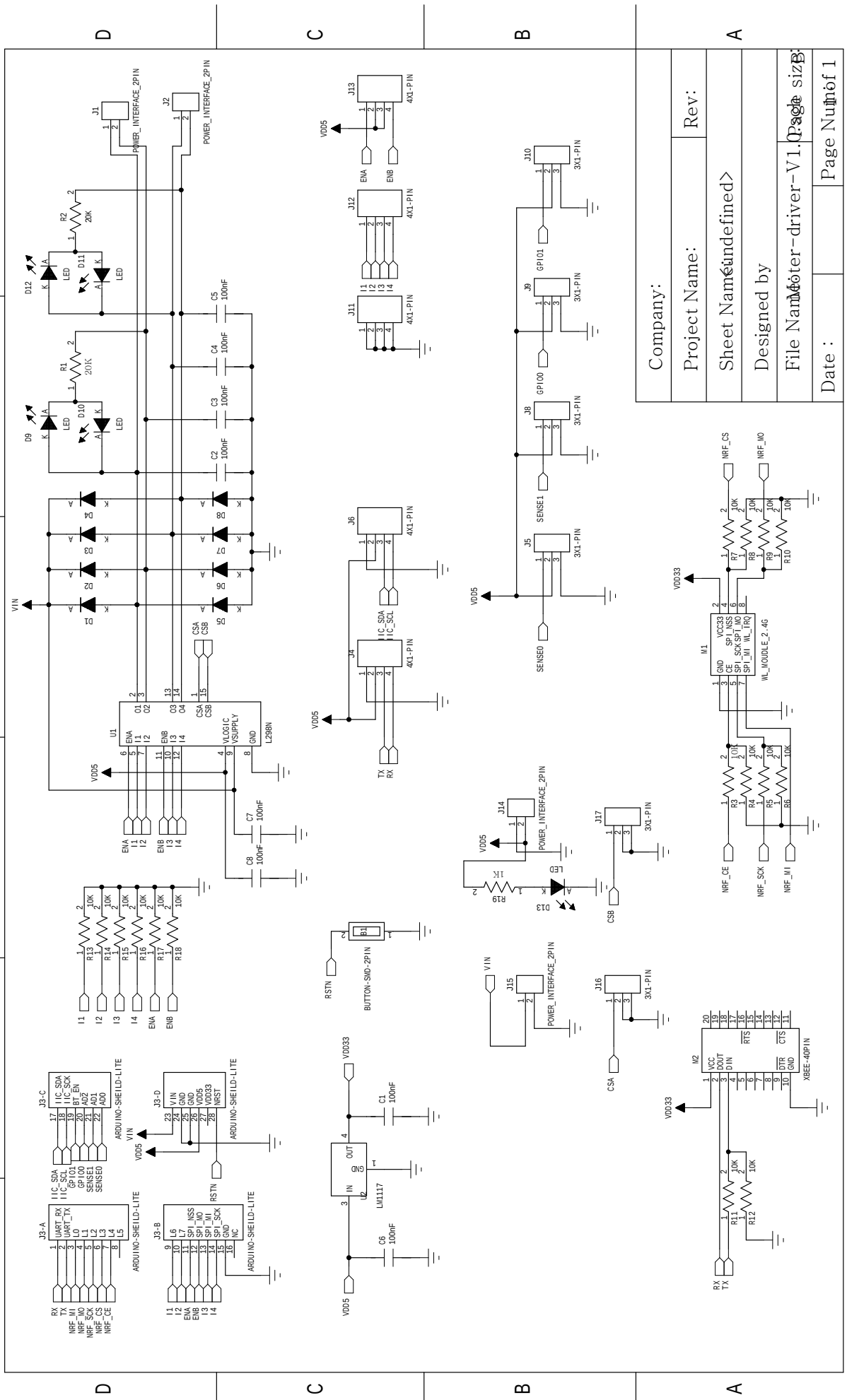


AtMega328 #2

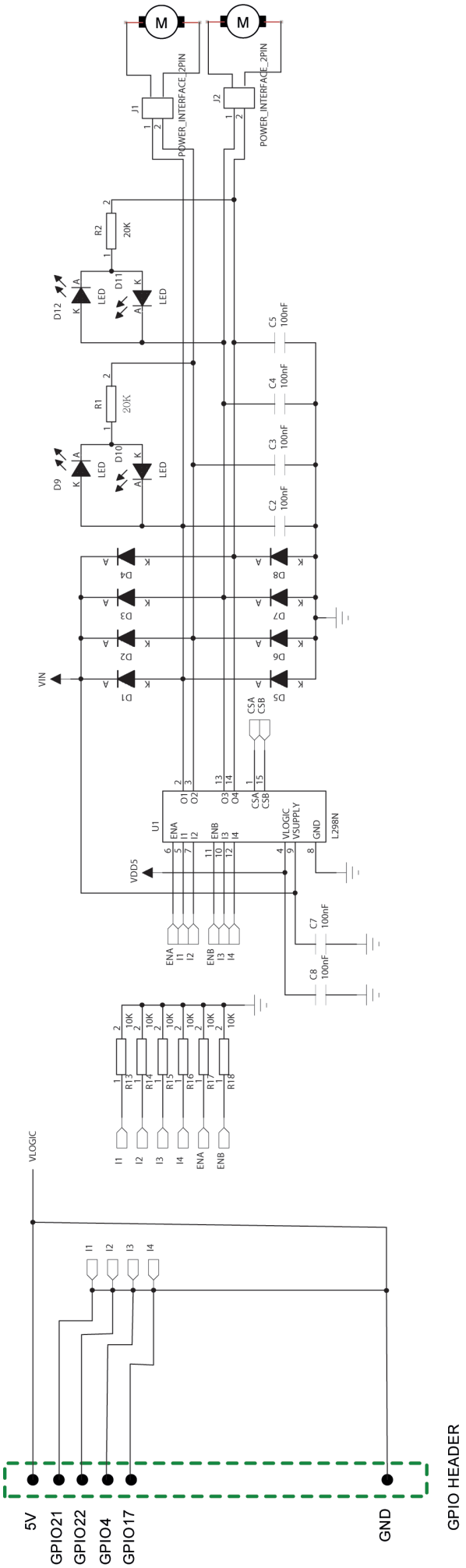
Sensor port pin 6 needs ADC



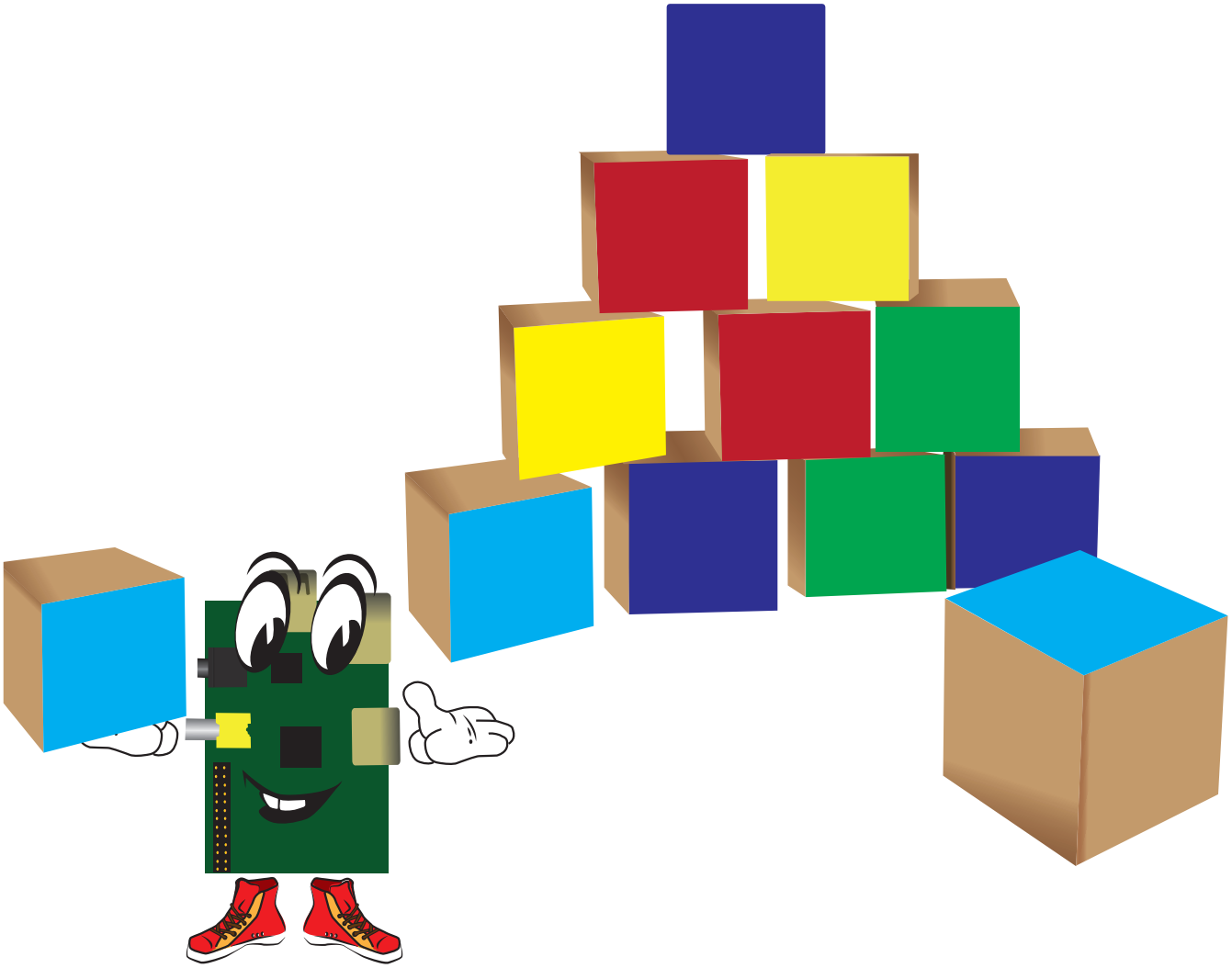
ANHANG: MotorMama H-Brücke



ANHANG: MotorMama angeschlossen am Raspberry Pi



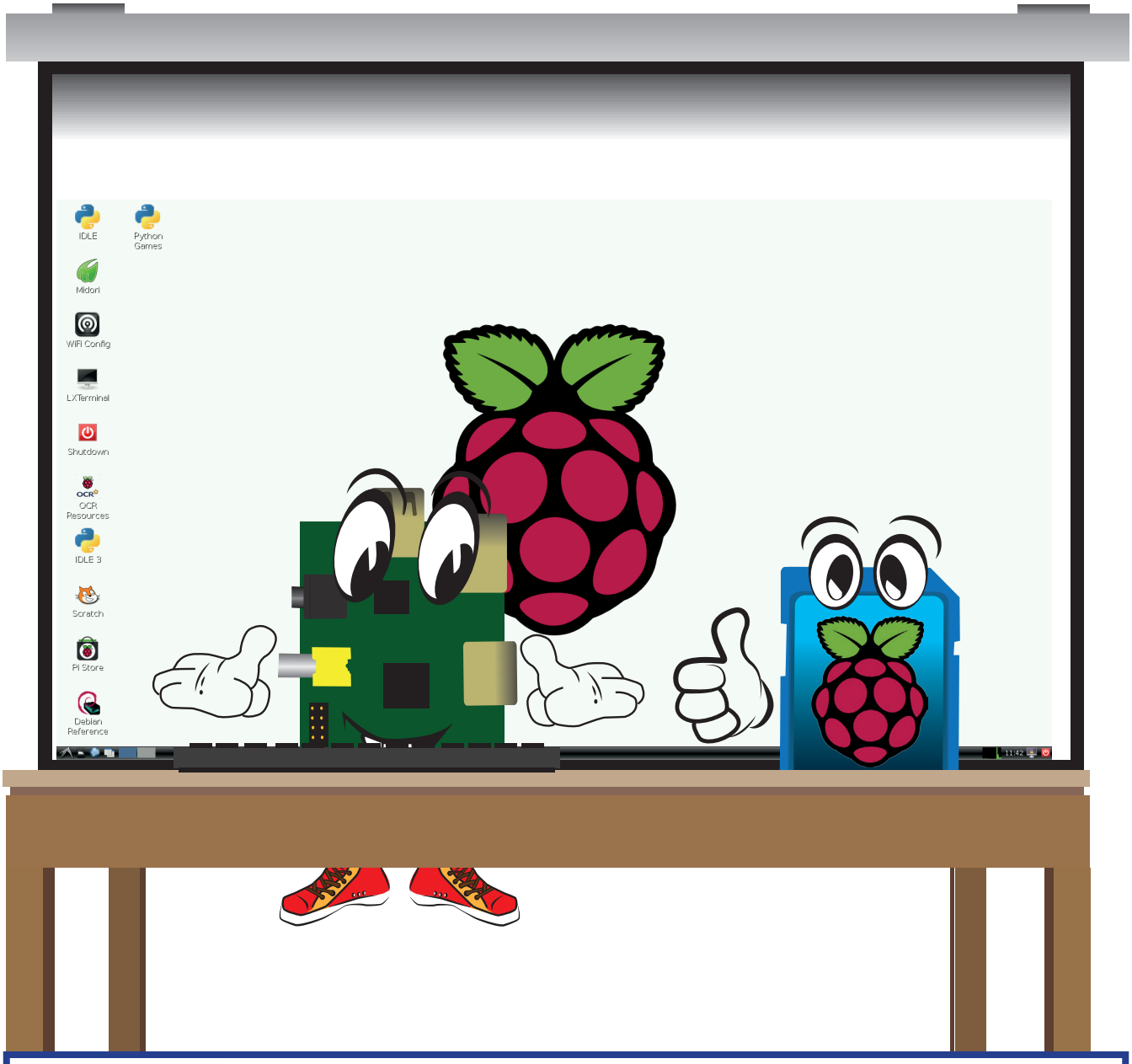
Baukastensystem



Baukasten

Modul 1 RasPi und Raspbian
Modul 2 RasPi als Mediencenter
Modul 3 GPIO Grundlagen
Modul 4 RasPi als Spycam
Modul 5 GPIO Robot
Modul 6 Finger Tipp Reaktionsspiel
Modul 7 BrickPi & RasPi - Der Legoroboter

Das erste Treffen mit Linux



Modul 1

RasPi zusammenbauen
RasPi installieren
RasPi konfigurieren
RasPi updaten & upgraden
RasPi Applikationen installieren
RasPI WLAN konfigurieren

RasPI what´s your name?
RasPi & Samba
RasPi Kopfloss (mit Putty)

Inhalt

STEPS

- 1. Die Vorbereitung
- 2. Raspbian konfigurieren
- 3. Neue Software installieren
- 4. Netzwerkkonfiguration(WLAN)
- 5.Fernzugriff auf den Pi

Lern Ergebnisse

- Das System installieren und konfigurieren
- Umgang mit Linux (mit der grafischen Oberfläche und der Kommandozeile)
- Raspberry Pi bedienen können
- Eine kopflose Verbindung auf den Pi

Software

Windows:

SD-Formatter
Putty
Bonjour
Winrar

NOOBS mit Raspbian

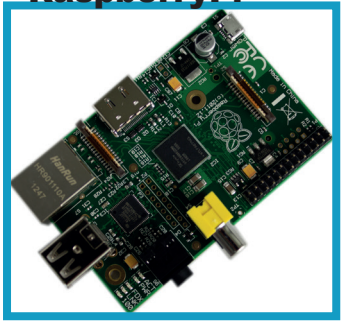
Raspbian:

Samba Server
Avahi-Daemon

Raspberry Pi und Raspbian

Hardware

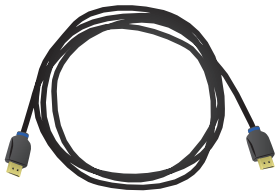
RaspberryPi



SD-Karte

Netzteil

HDMI-Kabel



Monitor

Tastatur/Maus

LAN-Kabel

WLAN-Adapter

Router

Die Vorbereitung (wird im Versuch nicht durchgeführt)

NOOBS auf SD-Karte kopieren

Diese Anleitung zur Installation von NOOBS auf die SD-Karte beschreibt die Durchführung auf einen Windows-Rechner:

Öffne einen Browser deiner Wahl und lade dir NOOBS mit der neusten Distribution von Raspbian herunter: *http://www.raspberrypi.org/downloads/*
Als nächstes brauchst du noch den SD-Formatter: *https://www.sdcard.org/downloads/formatter_4/*

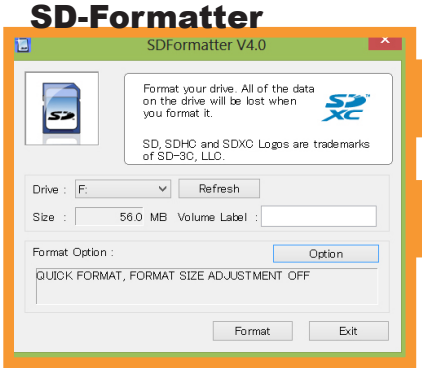
Download NOOBS

- Geh auf die Seite *http://www.raspberrypi.org/downloads/*
- Klicke auf den **Download Zip Button** von **NOOBS online and offline** und speichere die Datei in einem Ordner deiner Wahl ab.
- Entpacke die Dateien mit Winrar auf der Festplatte

SD-Karte formatieren

Bevor du NOOBS auf die SD-Karte kopieren kannst musst du die SD-Karte formatieren.

SD-Formatter



Wenn du dir den SD-Formatter heruntergeladen hast musst du ihn noch installieren:

Folge den Installationsanweisungen.

Stecke die SD-Karte jetzt in dein Kartenlesegerät ein und kontrolliere ob das Laufwerk stimmt.

Wähle den richtigen Laufwerkbuchstaben in SD-Formatter und drücke auf **Format.**

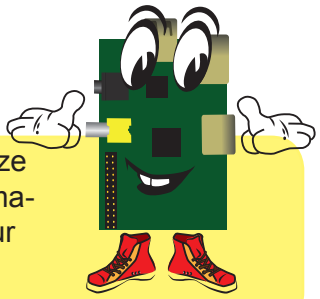
NOOBS kopieren

- Kopiere jetzt alle Dateien, die du aus der NOOBS.ZIP entpackt hast, auf die SD-Karte
- Wenn der Kopierprozess zu Ende ist, entferne die SD-Karte aus dem Kartenlesegerät.

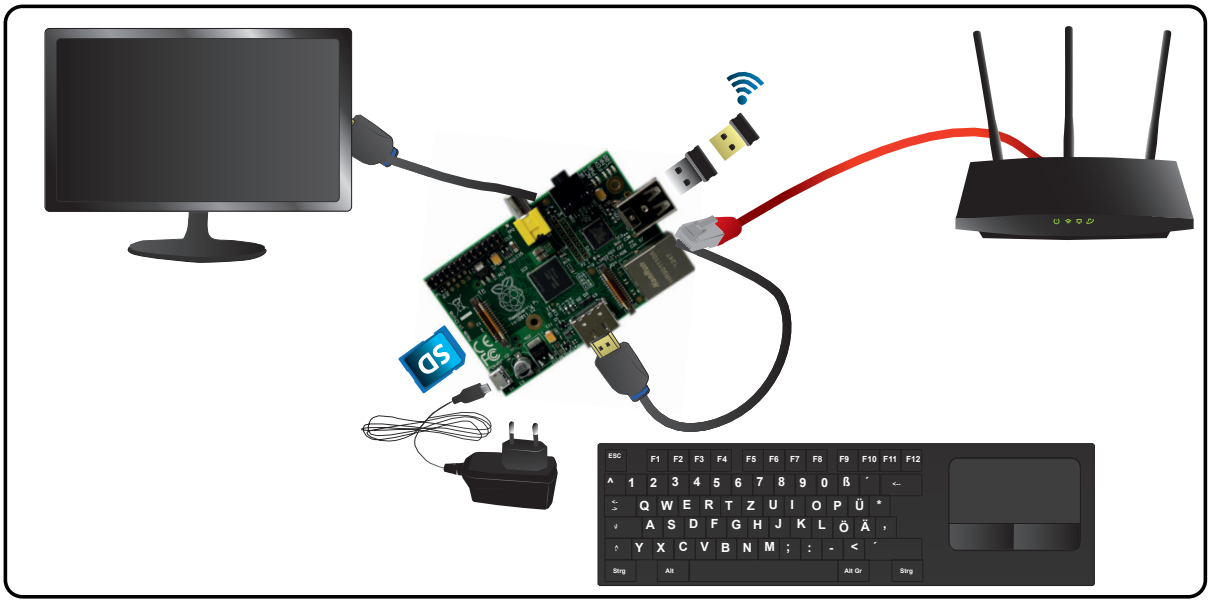
Die Vorbereitung

Hardware Zusammenstecken

Jetzt geht es los! Baue die Komponenten rum um den Raspberry Pi so wie in der Skizze auf.



Stecke auf keinen Fall das Netzteil in die Steckdose, bevor du nicht die ganze Hardware an den RasPi angeschlossen hast! Das System wird z.B. nichts machen, wenn die SD-Karte nicht im Slot ist oder wenn du den Dongle (Tastatur oder WLAN) einsteckst wird der Pi neustarten.



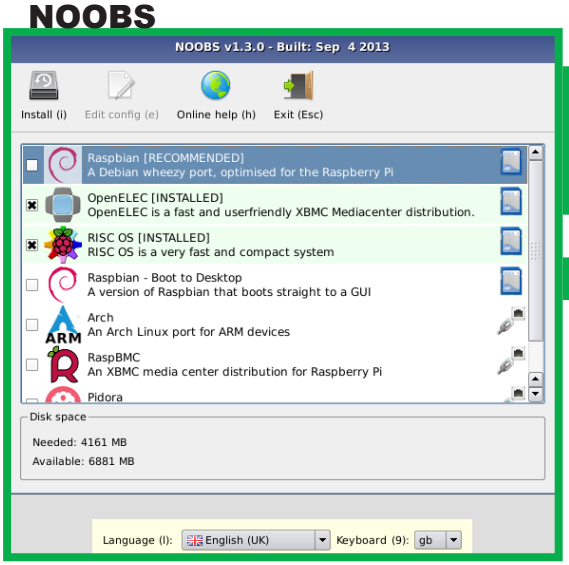
Jetzt kannst du das Netzteil mit der Steckdose verbinden und der Raspberry bootet.

Raspbian konfigurieren

NOOBS auf SD-Karte kopieren

Der Raspberry Pi lädt das NOOBS Installationsmenü, wo eine Auswahl an Distributionen zur Verfügung stehen.

NOOBS



Unten kannst du die Sprache und das Tastatur-Layout ändern. Mit der **Taste 9** kannst du das Layout auf **de** setzen und mit der **l Taste** stellst du die Sprache auf Deutsch um.

Mache ein **x** bei Raspbian durch anklicken der Box.

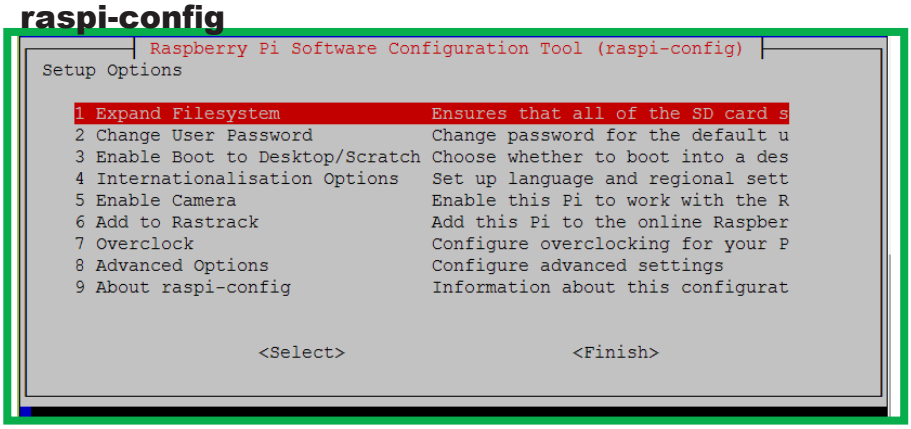
Klicke auf **Install** um Raspbian zu installieren

Nach einer Weile ist Raspbian installiert und der RasPi startet `raspi-config`.

Raspi-Config

Beim ersten Start von Raspbian, wird ein Einstellungs-Tool namens `rasp-config` gestartet. In diesem Konfigurations-Tool müssen wir einige Änderungen vornehmen. Ich werde die nützlichsten Punkte kurz erläutern und dir sagen was du machen musst.

raspi-config



Expand Filesystem

Normalerweise nutzt Raspbian nur ein Bruchteil der SD-Karte. NOOBS hat für uns schon die Arbeit erledigt und am Anfang den vorgegebenen physikalischen Platz der SD-Karte eingestellt.

Raspbian konfigurieren

Change User Password

Standardmäßig ist der Benutzer: **pi** und das Passwort: **raspberry** eingestellt. Willst du das Passwort ändern, darfst du es nur nicht vergessen. Wie man den Benutzernamen ändert zeig ich dir später.

Internationalisation Options

Da wir die Sprache und Tastatur-Layout im NOOBS geändert haben, müssen wir nur noch die Zeit und das Land ändern.

I2 Change Timezone

Wähle **Europa** als Kontinent aus und **Berlin** für Die Zeitzone aus.

Advanced Options

In diesem Untermenü müssen wir noch noch zwei Sachen erledigen und dann kann es losgehen.

A2 Hostname

Hier kannst du dem RasPi einen Namen vergeben um ihn in deinem Netzwerk leichter zu finden. Ein cooler Name ist RasPi ;-)

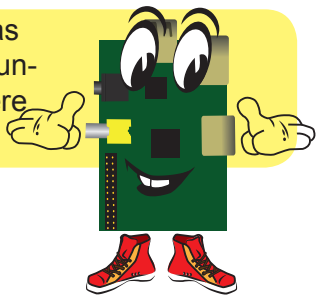
A4 SSH

Um eine kopflose Verbindung zu deinem Raspberry über einen anderen Computer herzustellen um ihn von dort fernzusteuern musst du die Funktion auf **Enable** setzen.

RasPi neu starten

Hast du alle Einstellungen angepasst, musst du nur noch den Pi neustarten, einfach auf **Finish** klicken und RasPi fragt nach einem Neustart. Bestätige und warte bis die kleine Platine neu gebootet hat.

Nach dem Neustart fragt der Pi nach einem Benutzer und Passwort. Hast das Passwort nicht geändert so lautet es **raspberry** und der Benutzer **pi**. Und wundere dich nicht, wenn du beim Passwort eintippen kein Sternchen oder andere Zeichen siehst. Du siehst nichts aber du tippst sie ein.



Um Raspbian nicht nur von der Textseite kennenzulernen wechseln wir jetzt mal in die Desktopoberfläche. Dazu musst du in der Konsole folgenden Befehl eintippen und mit RETURN bestätigen.

```
startx
```

Neue Software installieren

Der Befehl sudo

Bevor wir mit der Desktopoberfläche ein wenig rumspielen, muss ich dir noch etwas erklären. Es geht um die Nutzerrechte bei Raspbian und Linux. Der RasPi ist so eingerichtet, dass der Benutzer pi in seinem Ordner `/home/pi/` alles machen kann was er will, d.h. neue Verzeichnisse und Dateien erstellen, sie editieren und löschen etc. Willst du dies außerhalb des `/home/pi/` Verzeichnisses machen, sagt dir Raspbian, dass du die Rechte dafür nicht besitzt. Damit schützt Linux das System, um keinen Unsinn mit den Systemdateien anzurichten. Aber keine Angst, es gibt ein Befehl der dir da weiter hilft, falls du doch mal ein Programm oder eine Datei öffnen musst. Der Befehl lautet:

```
sudo
```

RasPi updaten & upgraden (wird im Versuch nicht durchgeführt)

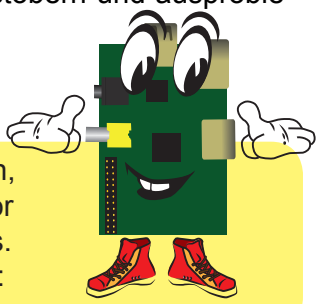
Auf den Desktop befindet sich ein Icon mit dem Namen LXTerminal. Klicke dieses an und warte bis es sich öffnet. Als nächstes wollen wir den RasPi auf den neusten Stand bringen. Zuerst einmal wollen wir die Datenbank mit den Downloadlisten der Programme aktualisieren.
Tippe dazu in dem LXTerminal diesen Befehl und bestätige ihn mit RETURN:

```
sudo apt-get update
```

Nach kurzer Zeit ist die Liste aktualisiert.
Als nächstes müssen wir die Programme und das System aktualisieren:

```
sudo apt-get upgrade -y
```

Das `-y` steht für „Yes“, damit du die `upgrades` nicht bestätigen muss.
Wenn Raspbian einige Aktualisierungen gefunden hat musst du jetzt länger warten. In der Zeit kannst du dich ein wenig mit der grafischen Oberfläche auseinandersetzen, herumstöbern und ausprobieren.



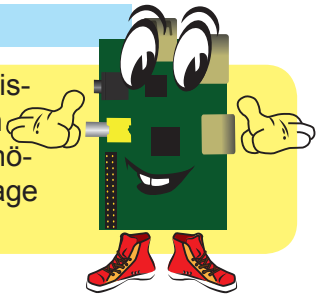
Der Befehl `sudo` erlaubt dir Programme auszuführen, Dateien zu bearbeiten, Installationen, Deinstallationen usw. durchzuführen, die nur der Administrator ausführen darf. Du bekommst also damit die super Rechte des Super-Users. Probiere doch mal aus, wenn du den Befehl `upgrade` ohne sudo schreibst.:

Neue Software installieren

Installieren mit apt-get

Ich zeige dir jetzt mal, wie du neue Software in Raspbian installieren kannst. Dafür lernst du einen neuen Befehl:

```
apt-get
```



Die Programme werden unter Linux auch Pakete genannt, diese werden meistens aus dem Internet gezogen. Beim Installieren von avahi-daemon werden auch andere Pakete installiert, die avahi für reibungsloses Funktionieren benötigt. Diese Aufgabe übernimmt das Verwaltungssystem APT (Advance Package Tool) = `apt-get`

Pakete suchen

Lass uns mal schauen, wie man ein Paket sucht, installiert und anschließend deinstalliert.
Den Paket-Namen den wir suchen, heißt shutter. Shutter ist ein Screenshot Programm. Würden wir es über die Suchfunktion von APT nach Screenshot suchen, würden wir Shutter nicht finden. Also empfehle ich erstmal eine Internetsuche Durchzuführen, um das richtige Programm zu finden. Jetzt müssen wir schauen ob die Lokale Datenbank das Paket auch da hat.

```
sudo apt-cache search shutter
```

Da du die Datenbank geupdatet hast (sudo apt-get update), erhälst du auch ein positives Ergebnis. In einer kleinen Liste befindet sich Shutter.

Pakete installieren

```
sudo apt-get install shutter -y
```

Das Programm wird jetzt mit allen abhängigen Dateien installiert.

Pakete deinstallieren

Wenn du das Paket nicht mehr brauchst, kannst du es mit dem Befehl entfernen:

```
sudo apt-get remove shutter
```

Damit bleiben aber noch die Konfigurationsdateien übrig, falls du die Einstellungen beim nächsten installieren brauchst. Willst du aber die Software komplett löschen, so tippe folgendes ein:

```
sudo apt-get purge shutter
```

Komplett Reinigung

Wie jedes System, hinterlässt auch Raspbian veraltete Pakete, von alten Installationen liegen oder Dateien die keiner mehr benötigt. Da kann die Putzfrau helfen:

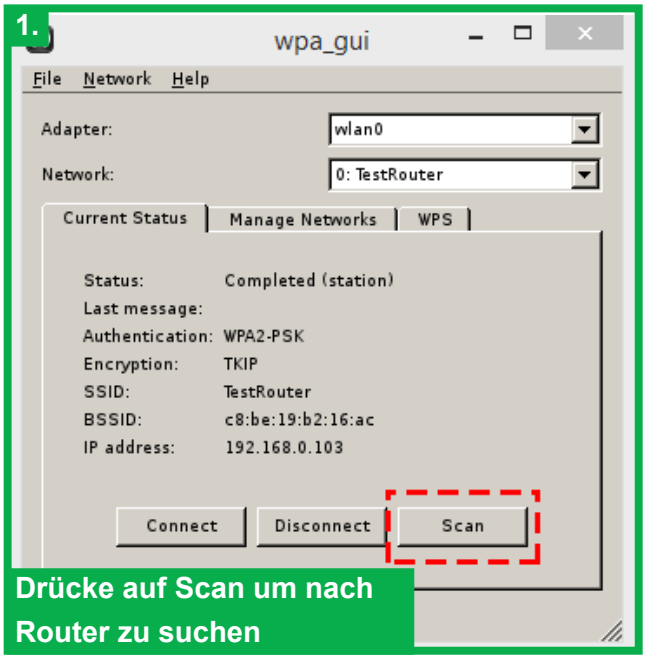
```
sudo apt-get autoremove
sudo apt-get autoclean
```

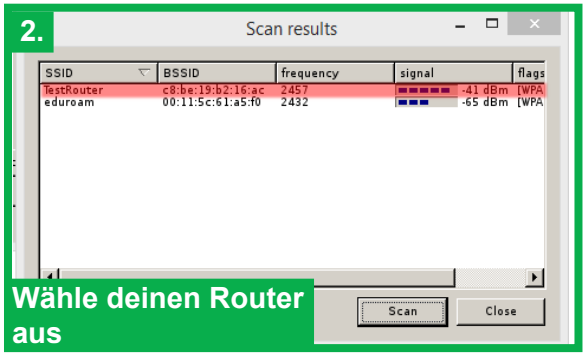
Netzwerkkonfiguration (WLAN)

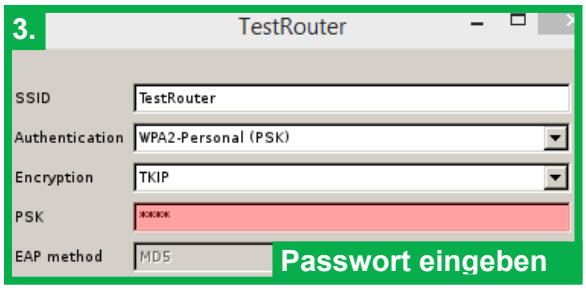
WLAN Konfiguration

Unter diesem Icon verbiergt sich die WIFI-Konfiguration. Ich zeige dir nun, wie man den WLAN-Dongle, den du am Anfang an den RasPi angeschlossen hast, nutzen kannst. Standardmäßig, findest du die WLAN-Einstellungen unter dem **Startmenü / Einstellungen**.

Nachdem sich das Einstellungsmenü geöffnet hat, musst du nur noch nach deinem Router suchen, Passwort eingeben und abspeichern. Ich zeige dir, wie es geht, am einen Beispiel:







1. Drücke auf den Scan-Button um nach dem Router zu suchen.
2. Wähle den richtigen Router aus und öffne ihn mit einem Doppelklick.
3. In der Zeile SSID sieht du dein WLAN-Namen deines Routers.

- Alle Einstellungen müssten vom Router richtig übernommmen worden sein. Nur in der Spalte PSK musst du das im Router eingestellte Passwort eingetragen.
- Mit Save werden die Einstellungen gespeichert und wpa_gui verbindet automatisch (notfalls auf Conect drücken) mit dem Router. Der Router übergibt dem Raspberry eine IP-Adresse.
- Sobald die WLAN-Verbindung besteht musst du die Einstellungen unter File / Save Configuration dauerhaft speichern.

Jetzt brauchst du das LAN-Kabel nicht mehr.

Fernzugriff auf den Pi

Eine kopflose Vernindung zum RasPi

Was ist eine kopflose Verbindung? Jeder Computer braucht eine Maus, Tastaur und ein Monitor um ihn zu bedienen. In weiteren Verlauf des Baukastensystems wirst du ein Roboter bauen, nur mit Monitor, Tastatur ist es schwer mobil zu werden. Also biettet dir der RasPi die Möglichkeit, sich über einen anderen Computer mit ihm zu verbinden und ihn zu bedienen. Deshalb wird eine kopflose Verbindung über Secure Shell (SSH) zum Raspi hergestellt. Die SSH bietet eine sichere, verschlüsselte Verbindung zum RasPi, bei der man sich mit dem Name und Passwort von Raspbian anmelden muss. SSH hast du ja in der raspi-config eingestellt.

Ich zeige dir aber noch weitere Varianten.

IP-Adresse herausfinden

Bevor du den RasPi mit

```
sudo halt
```

ausschaltest und die Tastatur/Maus, den Monitor von RasPi entfernst, brauchst noch die IP-Adresse deines Pi´s. Du hast jetzt drei Möglichkeiten die IP-Adresse herauszufinden schau in deinem Router nach oder in die Wifi-Config bzw. über das Terminal mit dem Befehl:

```
ifconfig
```

```
wlan0     Link encap:Ethernet  Hardware Adresse
          inet Adresse:192.168.0.103  Bcast:192.168.0.255  Maske:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metrik:1
          RX packets:131 errors:0 dropped:0 overruns:0 frame:0
          TX packets:76 errors:0 dropped:0 overruns:0 carrier:0
          Kollisionen:0 Sendewarteschlangenlänge:1000
          RX bytes:17069 (16.6 KiB)  TX bytes:11785 (11.5 KiB)
```

Putty für Windows herunterladen

Während du mit OSx bzw. Linux ganz einfach über das Terminal mit dem RasPi Kontakt aufnehmen kannst:

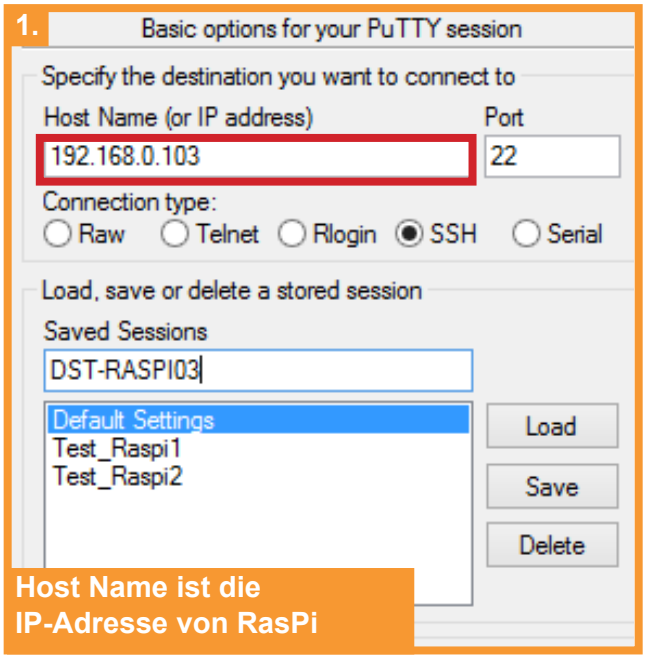
```
ssh user@IP-Adresse
```

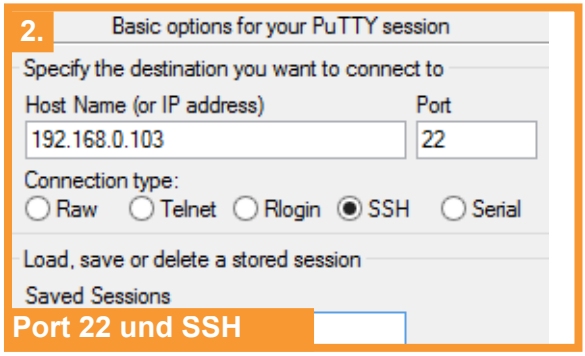
benötigst du unter Windows ein kleines Tool namens Putty, denn Windows hat keinSSH (Secure Shell) intergriert. Putty für Windowsrechner findest du auf der Seite http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html. Putty arbeitet nur auf der Konsolenebene.

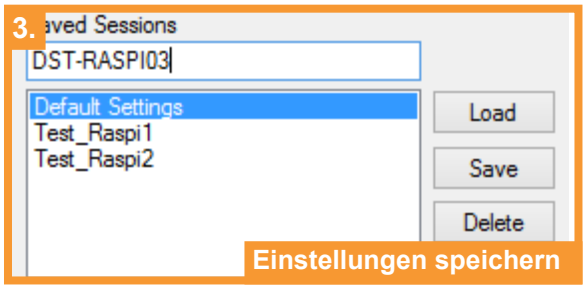
Fernzugriff auf den Pi

Putty Konfiguration

Hast du die putt.exe heruntergeladen, kann es mit der Konfiguration losgehen.
Öffne Putty:







1. Unter Host Name gibst du die IP-Adresse des RasPi´s, die du vorher rausgefunden hast.
2. Kontroliere ob der Port auf 22 steht und SSH als Verbindung ausgewählt ist.
3. Unter Saved Session kannst du deiner Sitzung einen Name vergeben.
   Mit Save speichert Putty die Einstellungen.
   Klicke auf Open und das dir bekannte Terminalfenster öffnet sich. In diesem Fenster fragt dich RasPi nach dem Benutzernamen und Passwort.

Weitere Varianten

In letzten Abschnitt des Raspberry Pi und Raspbian Moduls zeige ich dir, was es noch für Möglichkeiten gibt, um auf den Desktop von Raspbian zuzugreifen. Ich zeige dir, wie du statt der IP-Adresse, sich mit dem Hostnamen mit dem RasPi verbindest und auf die Dateien von Pi über den Explorer zugreifen kannst. Und somit hast du dir einen Netzwerkfähigen kleinen Computer gebaut.

Fernzugriff auf den Pi

Avahi-Daemon

Es ist nervig, wenn sich der Router entscheidet manchmal dem RasPi eine neue IP-Adresse zu vergeben, jedes mal musst du nach der IP-Adresse im Router oder über ifconfig nachschauen. Mit Avahi-Daemon und Bonjour unter Windows, kannst du dem RasPi eine Domain vergeben, auf der Basis von dem Hostnamen. D.h. statt der IP-Adresse gibst nur noch „*raspi.local*“.

Avahi-Daemon installieren

Im Terminal von Raspbian installiere avahi-daemon mit folgenden Befehl:

```
sudo apt-get install avahi-daemon -y
```

Als nächstes musst du RasPi sagen, dass er bei jedem botten starten soll:

```
sudo update-rc.d avahi-daemon defaults
```

Unter rasp-config, hast den Namen schon definiert. Wenn du dich nicht mehr erinnern kannst, rufe die Raspi-config auf:

```
sudo raspi-config
```

In Advanced Options findest du unter dem Hostname, wie er heißt.

Bonjour unter Windows

Jetzt könntest du in Windows über Putty es ausprobieren, vorausgesezt du hast den Apple Dienst Bonjour auf dem Rechner. Ist es nicht der Fall kannst du ihn hier downloaden:
http://support.apple.com/kb/DL999?viewlocale=de_DE
Jetzt kannst du in Putty, statt der IP, den Hostnamen eingeben z.B.*“raspi.local“.*

Remote Desktop installieren

Hast du das Terminal satt, würdest du lieber die Desktopoberfläche von Raspbian benutzen, dann müsst du nur xrdp auf dem RasPi intallieren:

```
sudo apt-get install xrdp -y
```

Unter Windows im Suchfenster gibst du nur noch „*Remotedesktopverbindung*“ ein. Dort musst du unter Computer die IP-Adresse oder die Domain (z.B. raspi.local) des Raspberry Pi´s angeben. Die Zugangsdaten entsprechen denen, die du in SSH (Terminal/Putty) verwendest.

Es kommt schon mal vor, dass das –y am Ende des Befehls nicht funktioniert. Sollte das der Fall sein musst du die Installation ohne –y ausführen. Der Befehl vereinfacht die Installation, denn dann musst du den Download nicht mit „j“ bestätigen.

Fernzugriff auf den Pi

Samba Server installieren

Du willst doch bestimmt auch auf Dateien und Ordner von RasPi über deinen Windowsrechner zugreifen. Dafür gehst du in Explorer unter Netzwerk und dann siehst du schon den RasPi. Dafür musst du auf dem RasPi den Samba Server installieren:

```
sudo apt-get install samba samba-common-bin -y
```

Jetzt legst du mittels des folgenden Kommandos ein Samba Benutzer für den Benutzer pi an.

```
sudo smbpasswd -a pi
```

Jetzt wird es Zeit eine Freigabe einzurichten, das erledigst du in der Konfigurationsdatei vom Samba. Dazu trägst du folgenden Block am Ende der smb.conf Datei ein. Hierbei gibst du in den eckigen Klammern den Namen [RASPI01] für die Freigabe .Als nächste kommt der Pfad für den freigegebenen Ordner /home/pi. Im nächsten Schritt definierst du die Nutzerrechte writeable = yes, d.h. dass der Nutzer alle Dateien und Ordner verändern bzw. löschen darf. Mit der Einstellung guest ok = no definierst du, dass nur authentifizierte Benutzer auf den Ordner zugreifen dürfen.
Das alles trägst du am Ende der smb.conf ein. Mit folgenden Befehl rufst sie auf:

```
sudo nano /etc/samba/smb.conf
```

Trage es jetzt so ein:

```
[RASPI01]
path = /
writeable = yes
guest ok = no
```

Mit STRG+X verlässt du den Editor. Dabei wirst du gefragt, ob du die Datei speichern willst, bestätige mit j und drücke RETURN. Jetzt nur noch den Samba Server neu starten. Dann lässt es sich auf deinem Windowsrechner ausprobieren.

```
sudo /etc/init.d/samba restart
```

Dazu gehe in den Explorer, klicke klicke in die Adresszeile und tippe die Adresse vom Raspi folgendermaßen ein:

```
\\IP-ADRESSE VON RASPI
```

Entertaiment Pur

Modul 2

OpenELEC auf RasPi installieren
OpenELEC konfigurieren
Medien abspielen
OpenELEC mit NAS verbinden
OpenELEC über Smartphone
oder Tablet steuern

Inhalt

STEPS

1. Die Vorbereitung
2. OpenELEC installieren und konfigurieren
3. Addons installieren und konfigurieren
4. Mit OpenELEC auf NAS zugreiffen
5. OpenELEC über Smartphone steuern

Lern Ergebnisse

Mediacenter installieren und konfigurieren

Umgang mit OpenELEC

OpenELEC mit anderen Geräten steuern

Software

Windows:

Firefox

NOOBS mit OpenELEC

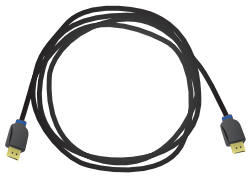
Smartphone:

Yatse
XBMC

OpenELEC Hardware

Hardware

RaspberryPi

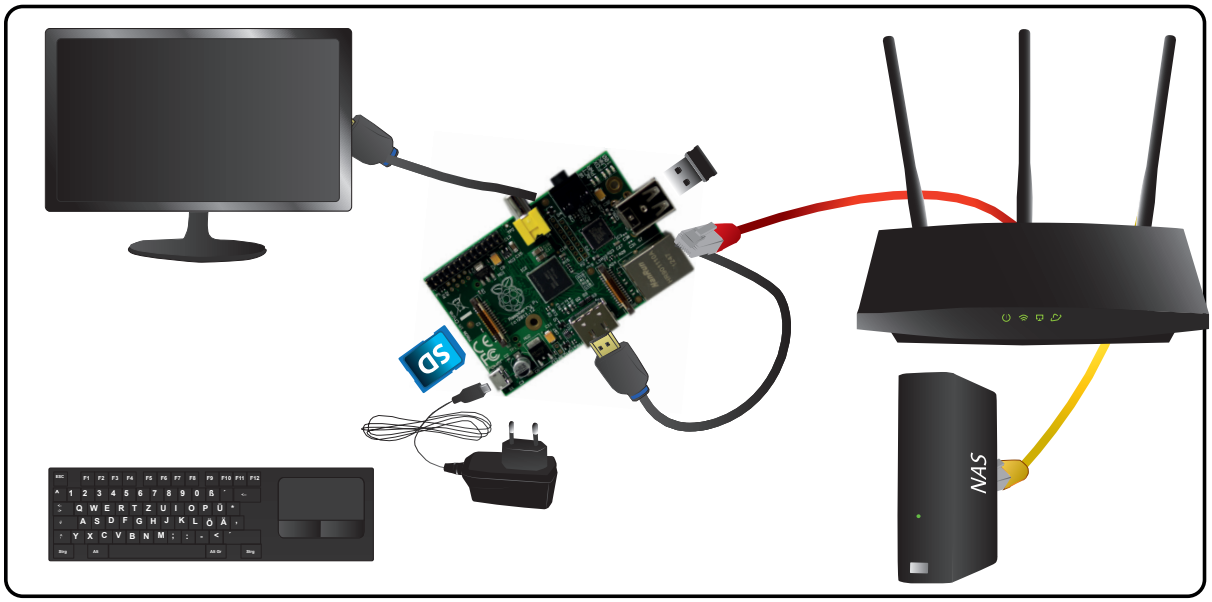


SD-Karte

Netzteil

HDMI-Kabel

Hardware Zusammenstecken

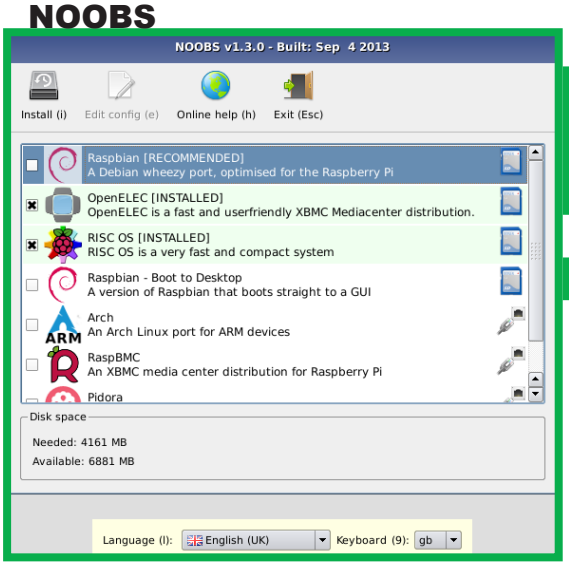


OpenELEC installieren

NOOBS auf SD-Karte kopieren

Der Raspberry Pi lädt das NOOBS Installationsmenü, wo eine Auswahl an Distributionen zur Verfügung stehen.

NOOBS



Unten kannst du die Sprache und das Tastatur-Layout ändern. Mit der **Taste 9** kannst du das Layout auf **de** setzen und mit der **l Taste** stellst du die Sprache auf Deutsch um.

Mache ein **x** bei OpenELEC durch anklicken der Box.

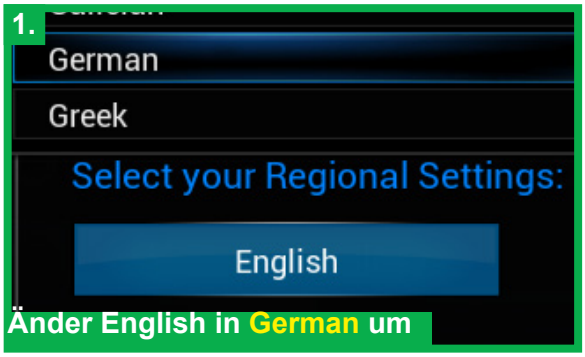
Klicke auf **Install** um OpenELEC zu installieren

Nach kurzer Zeit ist OpenELEC installiert und der RasPi startet es.

OpenELEC konfigurieren

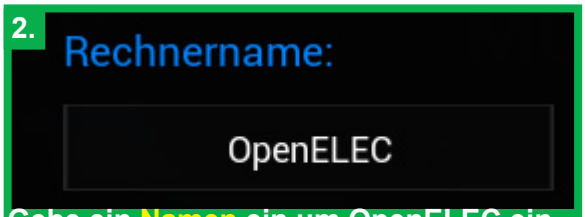
Wilkommensbildschirm

1.



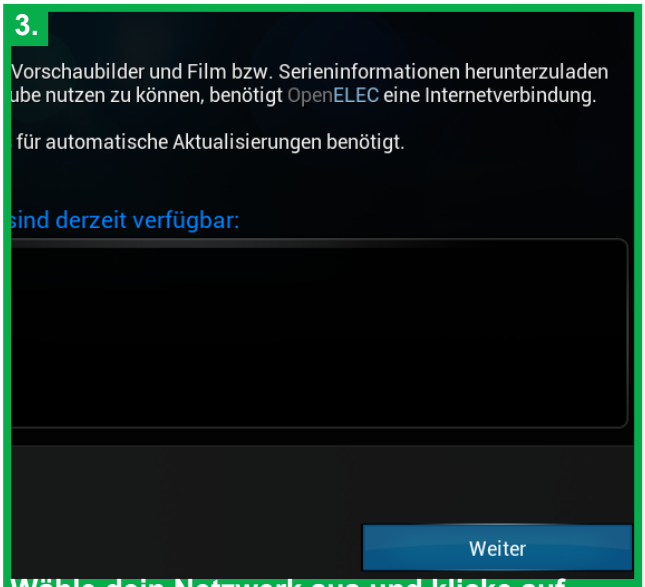
Änder English in German um

2.



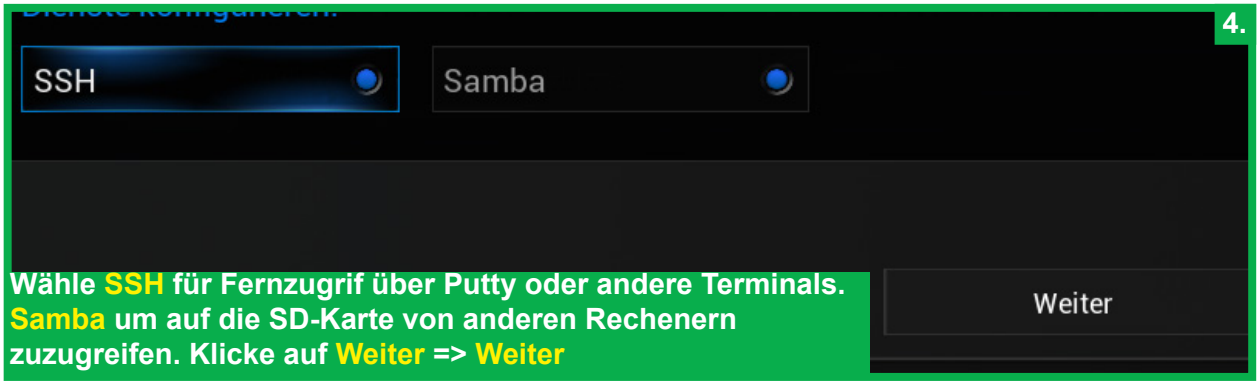
Gebe ein Namen ein um OpenELEC einfacher im Netzwerk zu finden

3.



Wähle dein Netzwerk aus und klicke auf Weiter

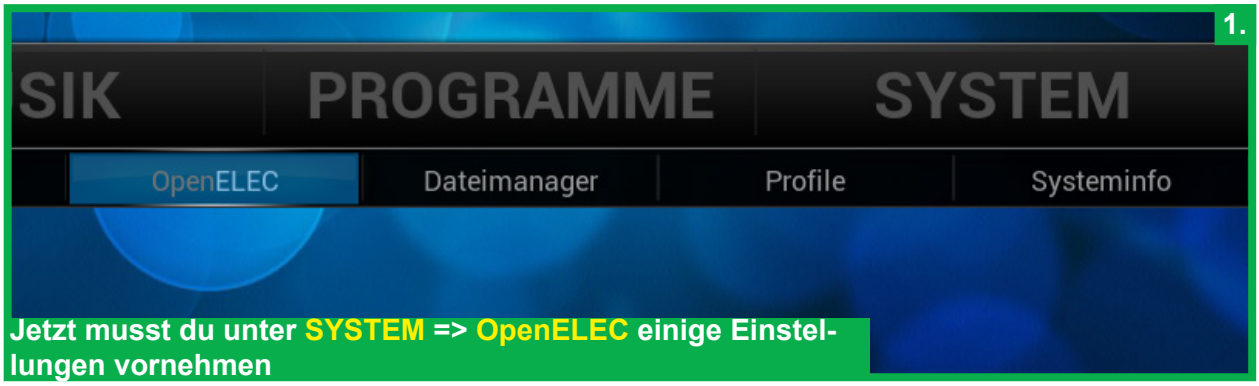
4.



Wähle SSH für Fernzugrif über Putty oder andere Terminals. Samba um auf die SD-Karte von anderen Rechenern zuzugreifen. Klicke auf Weiter => Weiter

SYSTEM => OpenELEC

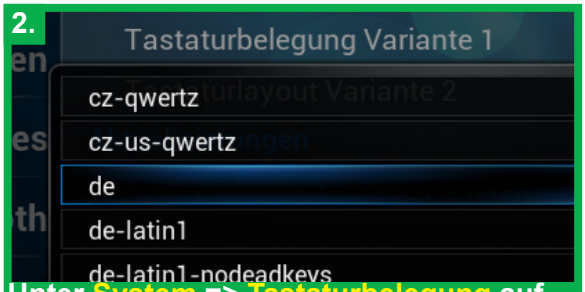
1.



Jetzt musst du unter SYSTEM => OpenELEC einige Einstellungen vornehmen

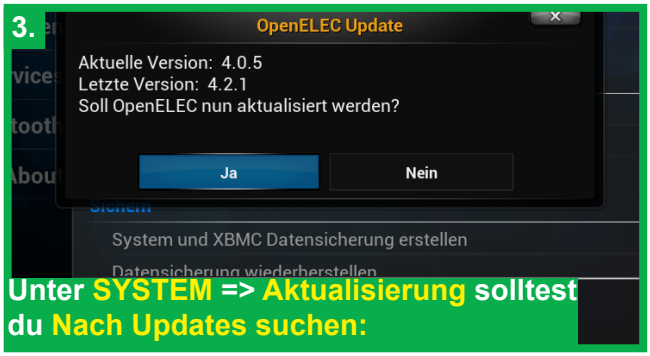
OpenELEC konfigurieren

2.



Unter **System** => **Tastaturbelegung** auf **de** ändern

3.



Unter **SYSTEM** => **Aktualisierung** solltest du **Nach Updates suchen:**

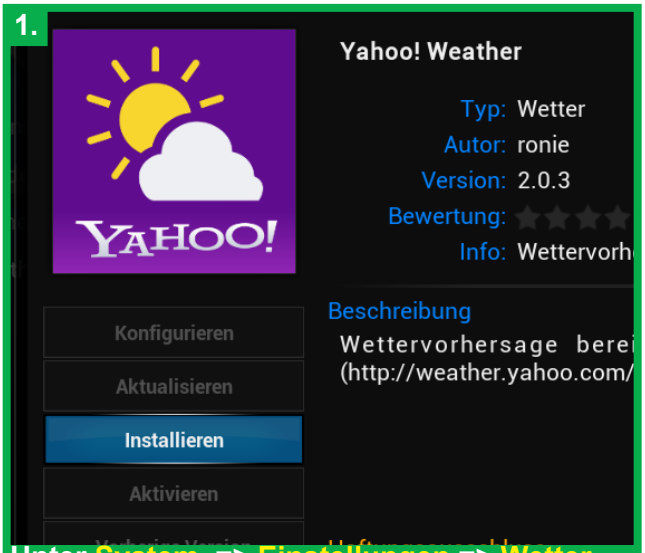
Für letzten wichtigen Einstellungen musst du unter SYSTEM => Einstellungen gehen

Darstellung				Die Einstellungen für das Aussehen und Sprache von OpenELEC werden hier vorgenommen
	Skin	RSS-Newsfeeds deaktivieren		Um nicht ständig News unten im Bildschirm zu bekommen
	Sprache & Region	Region	Deutschland	
		Zeitzonen-Region	Germany	Für die korrekte Zeitangabe
Video				In dieser Rubrik, kannst du das Abspielen der Videos konfigurieren
Live TV				Wenn du einen DVB-T Stick hast kannst du die Einstellungen für Fernsehbild konfigurieren
Musik				Hier kannst du Einstellungen für Musik vornehmen
Bilder				In diesem Menü kannst du Einstellungen vornehmen, wenn du dir deine geschossenen Bilder in einer Diashow über den RasPi anschauen willst
Wetter				Siehe unter Wetter installieren
Addons				Siehe unter YouTube installieren
Dienste				In diesen Menü kannst du einige Netzwerkprotokolle konfigurieren (siehe Webserver einstellen)
System				In dieser Sparte werden Einstellungen für Bildschirmauflösung, die Audioausgabe und weiter technische Konfigurationen vorgenommen
	Video-Hardware	Auflösung		Passe die Auflösung deinem Bildschirm bzw. Fernseher an
		Bildwiderholungs-rate		Es ist die Bildwiederholungsrate für die Oberfläche
	Audio-Ausgabe	Audio Ausgabegerät	HDMI/Analog	Willst du über Lautsprecherbox hören stellst du es auf Analog um

Addons installieren

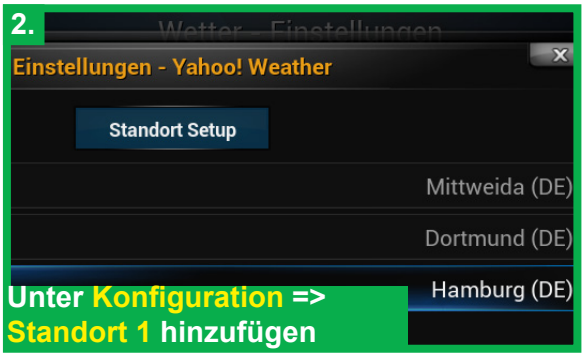
Wetter Addon

1.



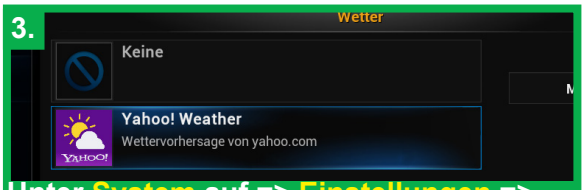
Unter System => Einstellungen => Wetter => Wetterservice => Mehr.. => Yahoo! Weather => Installieren

2.



Unter Konfiguration => Standort 1 hinzufügen

3.

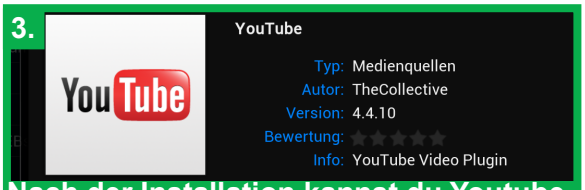


Unter System auf => Einstellungen => Wetterservice => Yahoo! Weather auswählen

Youtube Addon

Unter VIDEOS => Addons => Mehr... findest du Youtube als Addon => installiere es

3.



Nach der Installation kannst du Youtube konfigurieren => Konfigurieren

Eine Zusammenfassung der Einstellungen, die du für Youtube Addon ändern kannst

Allgemeines	In diesen Menü kannst dich bei Youtube mit deinen Benutzerdaten einloggen, falls du ein Login besitzt.
	Du kannst auch die Qualität der Videos einstellen, den Ordner für die Downloads der Videos und ob du Untertitel aktivieren willst.
Fortgeschritten	In dieser Rubrik kannst du den Anzeigemodus für Videoliste konfigurieren. Dabei kannst du einige wichtige Servereinstellungen vornehmen.
Ordnernamen	In Ordnernamen kannst du auswählen, welche Ordner die in dem Addon angezeigt werden sollen.
Entdecke Youtube	Welche Links von Youtube in Addon angezeigt werden sollen

Jetzt kannst du unter => **VIDEOS** => **Addons** => **Youtube** dort nach herzenslust Videos aus Kategorien anschauen sowie nach Videos suchen.

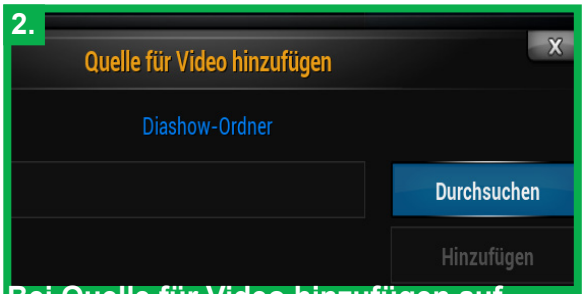
Mit NAS-Server verbinden

NAS in Openelec einbinden

Wie du ja sicher weißt hat die SD-Karte nicht so viel Platz um viele Filme darauf zu kopieren. Du kannst natürlich ein USB-Stick an den RasPi anschließen und die Videos über => VIDEOS => Dateien, den USB-Stick auswählen und Film abspielen. Eine andere Alternative zur dieser Lösung ist ein NAS-Server, auf dem sich alle deine Dateien befinden und du auf sie jederzeit über den RasPi zugreifen kannst. Aber es kommt noch besser, hast du mehrere RasPi´s mit dem Mediacenter zu Hause verteilt, können alle auf die Filme jederzeit zugreifen und das ohne Kabelsalat.

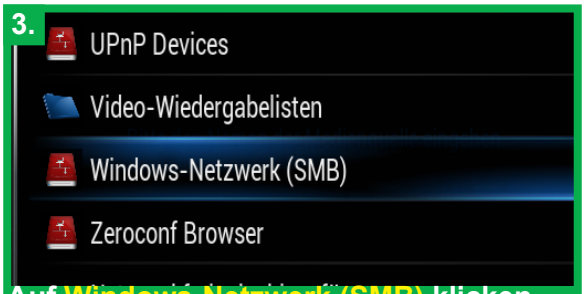
1. Damit sich die Videos vom Nas-Server anschauen lassen gehe zu:
VIDEOS => Dateien => Videos hinzufügen...

2.



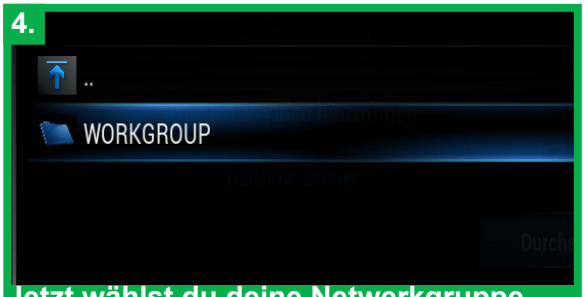
Bei Quelle für Video hinzufügen auf Durchsuchen klicken

3.



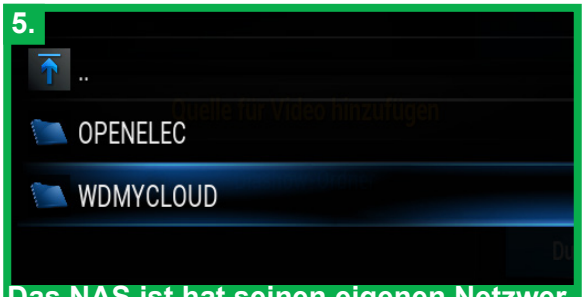
Auf Windows-Netzwerk (SMB) klicken

4.



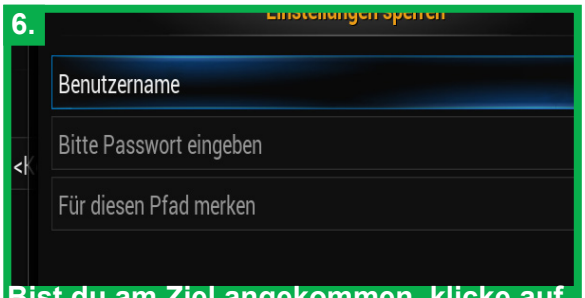
Jetzt wählst du deine Netwerkgruppe aus. In dem Beispiel ist es WORKGROUP

5.



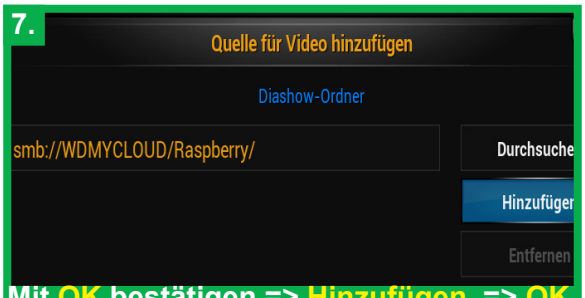
Das NAS ist hat seinen eigenen Netzwerknamen im Besipiel: WDMYCLOUD

6.



Bist du am Ziel angekommen, klicke auf OK und gebe die Logindaten ein

7.



Mit OK bestätigen => Hinzufügen => OK
Der Ordner steht jetzt zur Verfügung

8. Unter **VIDEOS** => **Dateien** findest du jetzt den NAS-Ordner mit den Filmen und kannst dir sie anschauen

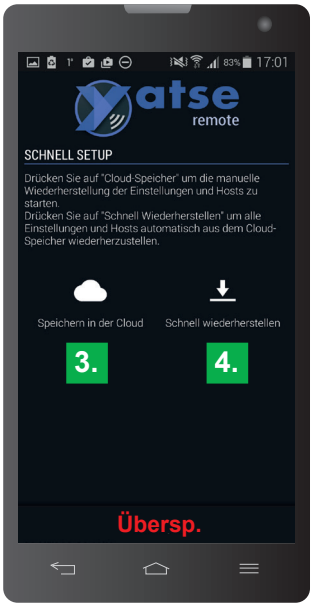
Fernsteuerung

Openelec mit Smartphone

Jetzt wird es Zeit OpenELEC über dein Smartphone zu steuern. Dazu musst du dich natürlich in demselben Netzwerk befinden, wie OpenELEC. Der Fernzugriff erfolgt über ein Android Smartphone mit einer App namens Yatse. Über IPhone funktioniert es so ähnlich, musst nur die passende App für dich finden.
Zuerst musst du den Fernzugriff aktivieren:

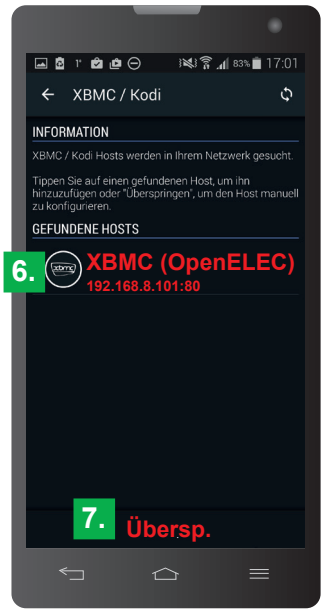
1. Gehe zu **Einstellungen** => **Dienste** => **Fernsteuerung** => und aktiviere **Steuerung über entfernte Programme zulassen**

2. Yatse ist eine der besten XBMC Apps auf den Markt, sie ist kostenlos und beinhaltet viele Funktionen. Es ist eine Super Fernbedienung für dein OpenELEC.
Im Play Store musst du nach Yatse suchen und dann installieren.



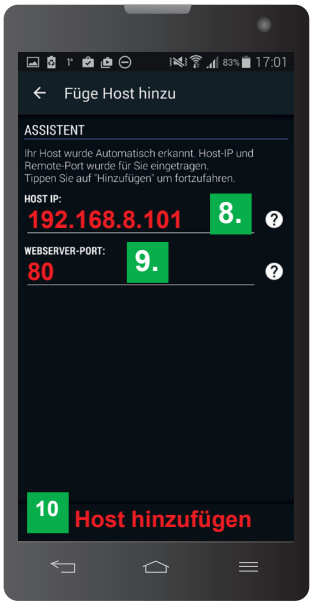
Starte jetzt Yatse. Mit **3.** **Speichern in der Cloud** und **4.** **Schnell wiederherstellen**, kann du abgespeicherte Einstellungen aus der Cloud laden. Da du keine hast, drücke auf **5.** **Überspr.**

Fernsteuerung



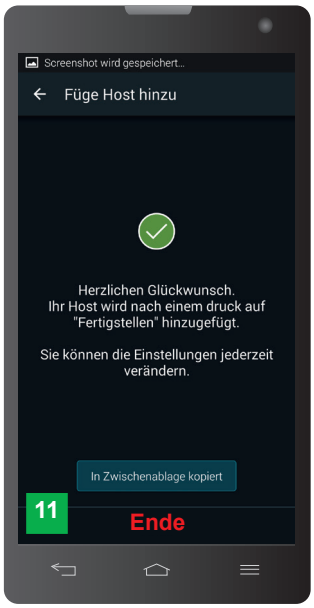
Yatse sucht jetzt nach OpenELEC, nach sehr kurzer Zeit müsste er das Mediacenter finden => klicke darauf. **6.**

7. Sollte es doch nicht der Fall sein, musst du diese Angaben manuell eingeben => Drücke dafür Überspr.



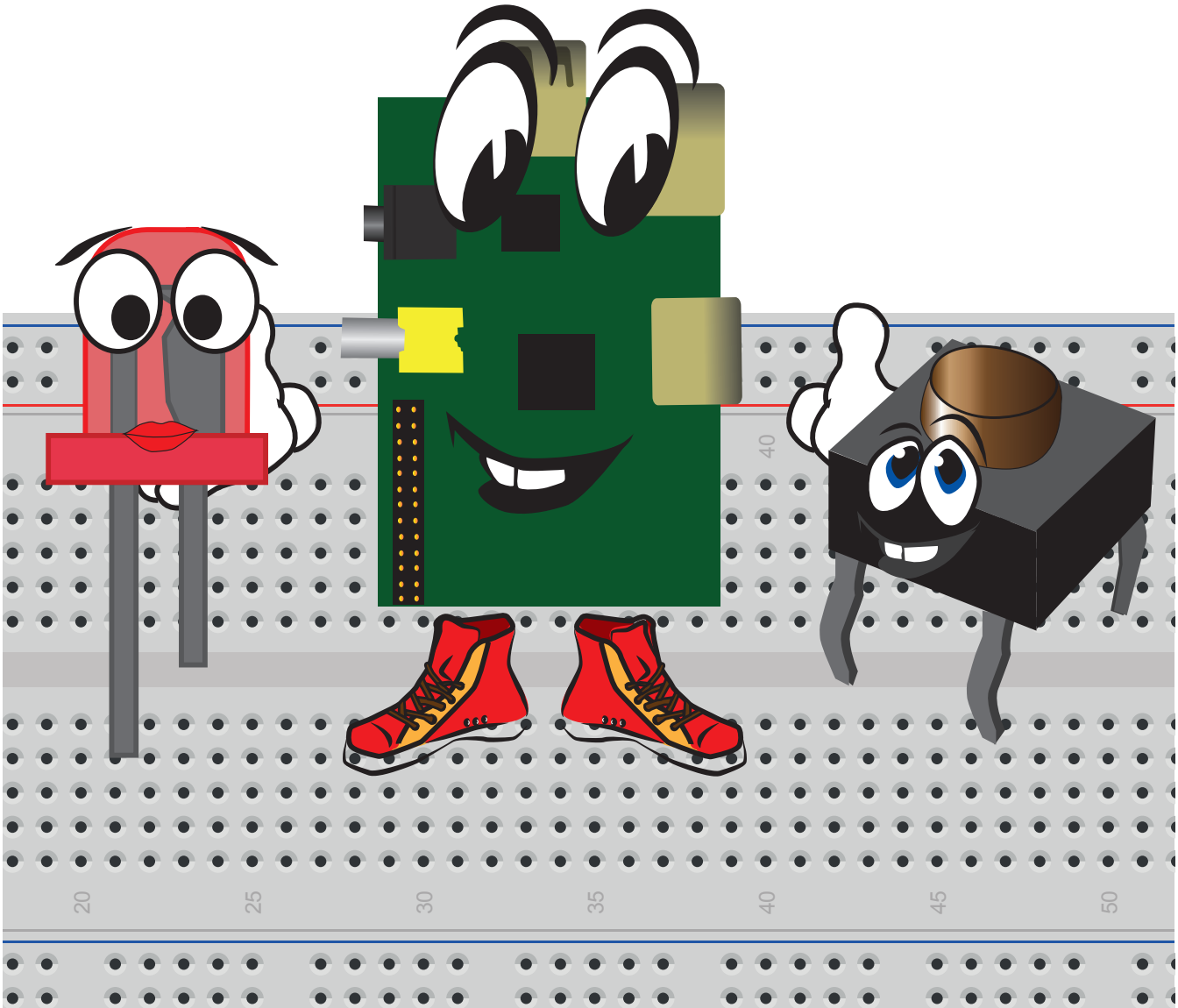
Wenn die Angaben für **8. HOST-IP** und **9. WEBSERVER-PORT** stimmen musst du nur noch auf **10 Host hinzufügen** klicken.

Sollte es nicht der Fall sein kannst du das in OpenELEC unter Systeminfo überprüfen. Den Port für Wedbserver findest unter **Einstellungen** => **Dienste**



Die Konfiguration ist jetzt fertig.
Jetzt nur noch ein Druck auf **11 Ende** und du kannst jetzt OpenELEC mit dem Smartphone steuern. Probiere es aus!

Hardwareprogrammierung



Modul 3

Grundlagen: elekektronische Bauelemente
GPIO Pins
Spannungsversorgung der Pins
Elektronische Bauelemente mit dem RasPi verbinden
Programmieren mit Python, das andere „Hello World“

GPIO Grundlagen

STEPS

- 1. Grundlagen
- 2. Aufbau nach Schaltplan
- 3. Python-Datei erstellen, Programmieren, Programm ausführen

Lern Ergebnisse

- Einstieg in die Elektronik
- Erfahrung und Umgang mit GPIO Pins
- Einstieg in die Python Programmierung

Software

NOOBS mit Raspbian

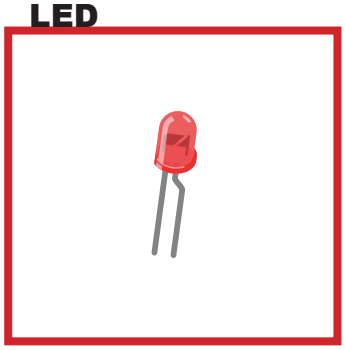
Raspbian:

Nano (Texteditor)
Python

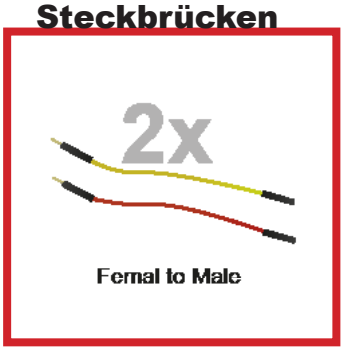
GPIO Grundlagen

Hardware A1

LED



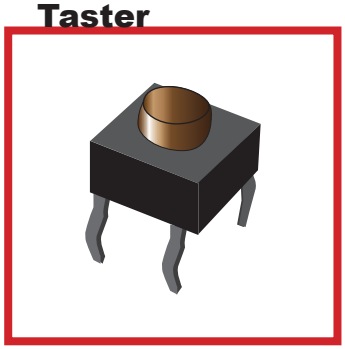
Steckbrücken



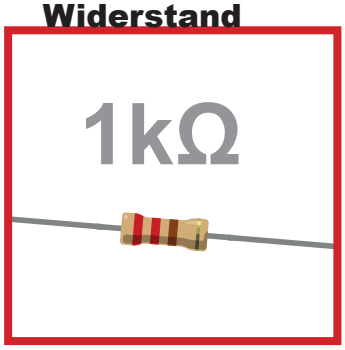
Widerstand

Hardware A1 + A2

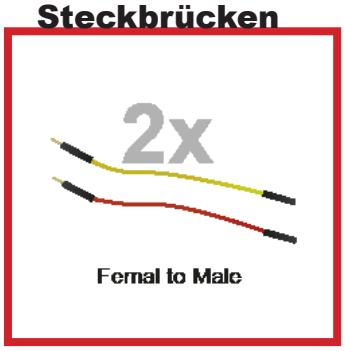
Taster



Widerstand

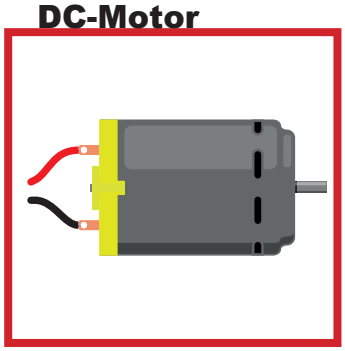


Steckbrücken

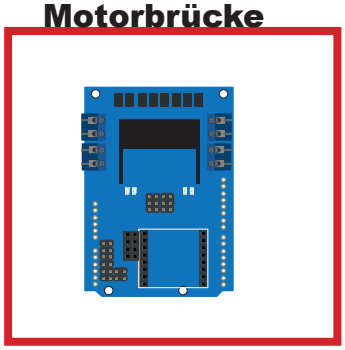


Hardware A1 + A2 + A3

DC-Motor



Motorbrücke

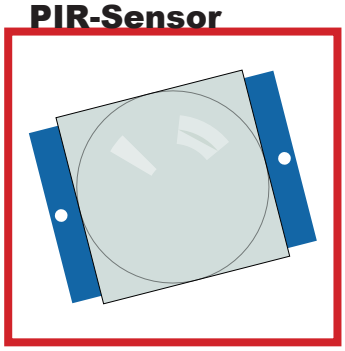


Bateriehalter

GPIO Grundlagen

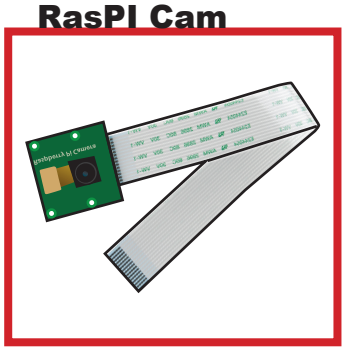
Hardware A1 + A4

PIR-Sensor



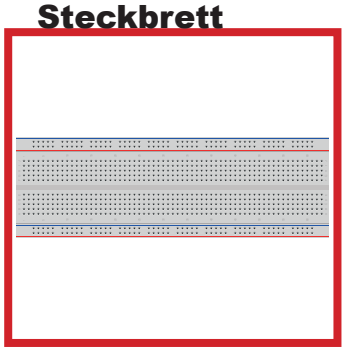
Hardware A1 + A4 + A5

RasPI Cam

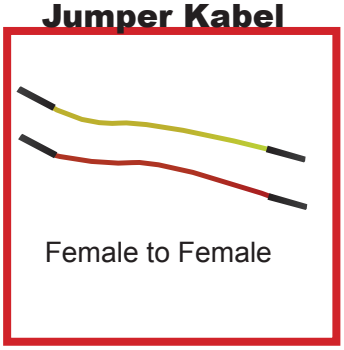


Für jeden GPIO-Aufbau

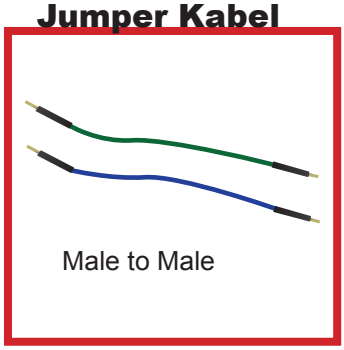
Steckbrett



Jumper Kabel



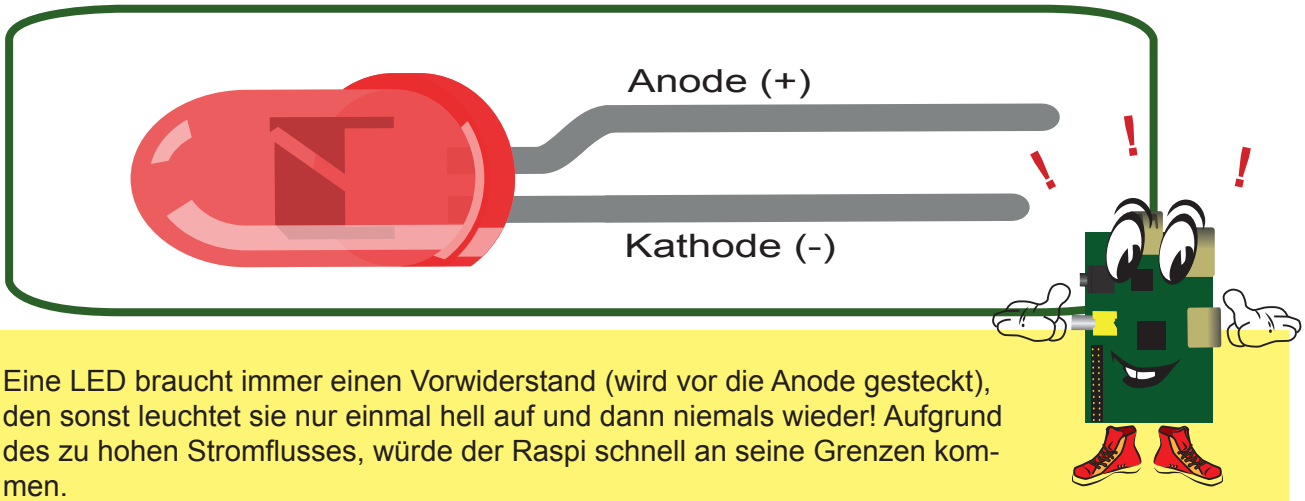
Jumper Kabel



Grundlagen

Die LED

Die LED gehört zu den Halbleitern. Sie leuchtet nur, wenn die Richtung des Stormflusses stimmt. Schließt man die LED falschrum an, wird diese nicht leuchten. Bei Betrachtung der LED stellt man fest, dass das eine Beinchen kürzer ist als das andere. Das längere Beinchen ist die Anode (+) und das kürzere Beinchen ist die Kathode (-).



Eine LED braucht immer einen Vorwiderstand (wird vor die Anode gesteckt), den sonst leuchtet sie nur einmal hell auf und dann niemals wieder! Aufgrund des zu hohen Stromflusses, würde der Raspi schnell an seine Grenzen kommen.

Der Widerstand

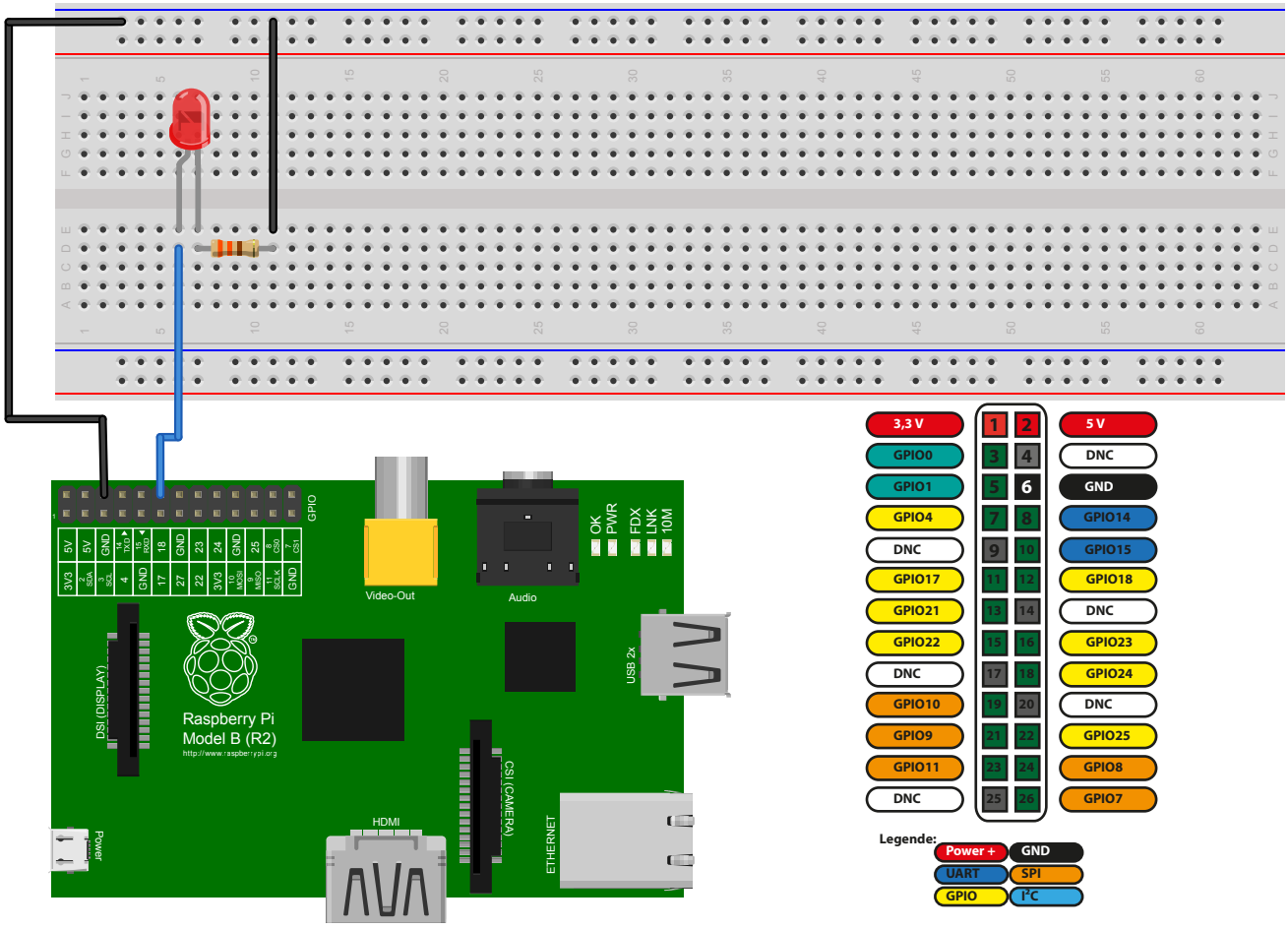
Damit der Raspberry Pi und die LED nicht kaputt gehen, wird ein Widerstand benötigt, der den Stromfluss reduziert. Dazu sollten einige Angaben über die Pins und der LED beachtet werden. Die Pins, die später die Programmierung entgegen nehmen, haben eine maximale Spannung von 3,3 V. Die rote LED hat eine Durchflussspannung von UF= 2V, d.h. der Widerstand muss noch die restlichen 1,3V von dem Pin kompensieren. Der Ganze Schaltkreis wird mit 4mA Strom versorgt. Durch diese Angaben lässt sich der Widerstand berechnen.

$$R = \frac{U_{ges} - U_F}{I} = \frac{3{,}3\ V - 1{,}3V}{0{,}004A}$$

Das Ergebnis liefert dir 325 Ohm, weil es so einen Widerstand nicht gibt, muss der nächst höhere Widerstand verwendet werden. In diesen Fall ist es ein 330 Ohm Widerstand.

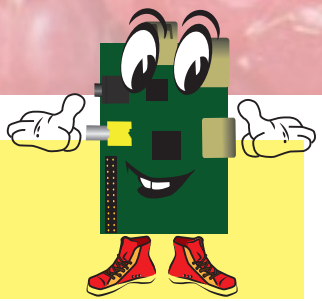
Aufbau nach Schaltplan

Bevor der Raspberry Pi eingeschaltet wird musst du zuerst die Schaltung aufbauen.
In diesen kleinen Versuch wird eine LED mit der Kathode an Pin 12 (GPIO18) angeschlossen der eine Versorgungspannung von 3,3V hat. An die Anode wird ein Wiederstand mit 330 Ohm angeschlossen um den Stromfluss zu reduzieren. Das andere Ende von dem Widerstand wird an den GND-Pin (in deinen Fall Pin 6) angeschlossen.



Elemente	Farbe	Anschluss	Farbe	RasPi
LED		Kathode		PIN 12 (GPIO18)
LED		Anode		
330 Ohm Widerstand		An die Anode der LED		
330 Ohm Widerstand		An GND des RasPi´s		PIN 6 (GND)

Programmieren mit Python



Um sich auf den Pi einzuloggen, lautet der Standardname: **pi** und das Passwort: **raspberry**

Schritt 1: Python Datei erstellen

Lege mit dem folgenden Befehl in der Kommandozeile eine leere Python Datei an:

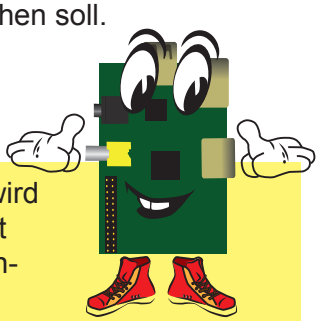
```
sudo nano 01_LED.py
```

Schritt 2: Der Python Code

Für die Steuerung der LED wird die Programmiersprache Python verwendet, die die LED einschaltet und nach einiger Zeit wieder ausschaltet.

Wenn der Pin 12 unter Spannung gesetzt wird, benutzt Python dafür den Befehl `HIGH` und `LOW` für keine Spannung.
Als erstes muss der Pin 12 als output deklariert werden, damit weiß Python, dass es sich um ein Ausgabeelement handelt.

Um die blinkende LED zu programmieren muss es in eine `while True` Schleife gepackt werden und mit dem Befehl `time.sleep` sagt Pyhton, wann die LED aus bzw. an gehen soll.

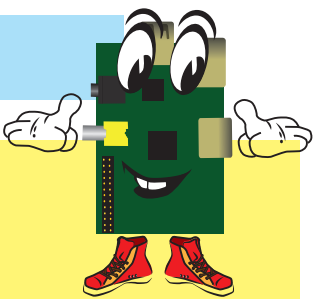


Bei der `while True` Schleife handelt es sich um eine Endlosschleife. Sie wird so lange durchgeführt, so lange die Bedinung erfüllt ist. In diesen Beispiel ist sie immer erfüllt. Das Programm kann dann nur mit der Taste STRG+C beendet werden.

Schritt 3: Ausführen

Mit der STRG+O speicherst du das Pyhtonprogramm ab und mit STRG+X kannst du Nano jetzt verlassen.
Als nächstes muss die Datei ausführbar gemacht werden und dann kannst du sie starten:

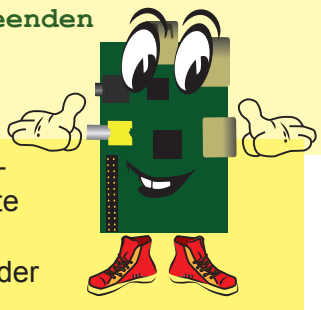
```
sudo chmod + x 01_LED.py
sudo ./01_LED.py
```



Sollte beim Ausführen des Codes folgender Fehler auftauchen:
`RuntimeError: No access to /dev/mem` bedeutet es, dass du den Befehl sudo davor vergessen hast zu schreiben.
Nur mit dem „Super-User“ können die GPIO-Pins gesteuert werden.

Python Code: 01_LED.py

```
#Mit dem Kommentar sagst du dem Interpreter, dass es sich um ein Python 2.x
#Programm handelt
#!/usr/bin/python

#Die wichtigen Bibliotheken(Module) werden eingefügt
import RPi.GPIO as GPIO     #Imortiert das GPIO-Modul für die Steuerung
import time                 #Importiert den Zeit-Modul
import os                   #Imortiert die Befehle des Systems-Modul

#Gibt an welcher Nummerierung des GPIO Ports verwenden werden soll.
GPIO.setmode(GPIO.BOARD)#Es wird von oben links nach unten rechts beginnent
                        # mit  Pin 1 (3,3V)

#Schaltet Warnungen aus, die für dich nicht wichtig sind
GPIO.setwarnings(False)

#Variablen
#An GPIO Pin 12 wird die LED angeschlossen, dieser wird als LED deklariert.
LED = 12

#Setze den Pin als Output (Ausgabe) / LED steht jetzt für Pin 12
GPIO.setup(LED, GPIO.OUT)

os.system ("clear") #Loescht den Bildschirm, es ist ein Terminalbefehl

while True: #Solange die Bedingung wahr ist, wird die while-Schleife
              ausgefuehrt
      try:
            print "LED AN" #Schreibt diesen Text auf den Bildschirm
            GPIO.output(LED, GPIO.HIGH) #GPIO-Pin wird auf High gesetzt
            time.sleep(1) #Wartet die angegebene Zeit ab

            print"Licht AUS"
            GPIO.output(LED, GPIO.LOW) #GPIO-Pin wird auf Low gesetzt
            time.sleep(1)

      except KeyboardInterrupt: #Mit STRG-C => Programm beenden
            print"Ende"
            GPIO.cleanup() #Resetet die GPIO-Pins
            quit()      #Beendet sauber den Skript
```



Die Raute sagt Python, dass es sich um ein Kommentar handelt. Das Programm ignoriert dahinter stehenden Text. Ausgeschlossen ist davon die erste Zeile: `#!/usr/bin/python`

Damit Python das Programm korrekt ausführen kann müssen die Blöcke in der Schleife erkannt werden, dies wird durch einrücken erfüllt (TAB-Taste)

Programmieren mit Python

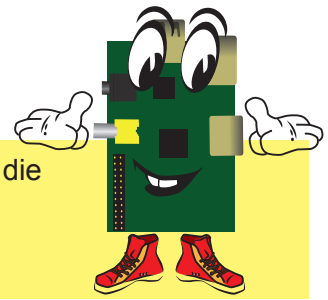
Schritt 1: Python Datei erstellen

Lege mit dem folgenden Befehl in der Kommandozeile eine leere Python Datei an:

```
sudo nano 02_LED_Terminal.py
```

Schritt 2: Der Python Code

In diesem Skript wird die LED mittels der Konsole gesteuert.
In einen kleinen Menü kann die LED EIN- bzw. AUS-geschaltet werden.
Wenn du die Taste 1 an deiner Tastatur drückst und mit RETURN bestätigst, wird der Pin 12 auf HIGH gesetzt und die LED geht AN.
Drückst du die Taste 2, wird der Pin 12 auf LOW gesetzt und die LED geht AUS.
Mit der Taste 3 kannst du das Programm verlassen.



`if wahl==1:` führt nur den Block der Programmierung aus, wenn wirklich die Taste 1 gedrückt wird, ansonsten wird dieser Block nicht ausgeführt.

Schritt 3: Ausführen

Mit der STRG+O speicherst du das Pyhtonprogramm ab und mit STRG+X kannst du den Texteditor verlassen.
Eine weiter Möglichkeit um die Datei auszuführen, sieht so aus:

```
sudo python 02_LED_Terminal.py
```

Python Code: 02_LED_Terminal.py

```
#!/usr/bin/python
import RPi.GPIO as GPIO
import time
import os

GPIO.setmode(GPIO.BOARD)#BCM wäre die ander Variante, dort werden die Pins nach
GPIO.setwarnings(False) #der Nummerierung des GPIO-Boards verwendet

#Variablen
LED = 12
wahl = 0    #Eine Variable, um die Tasten aus dem Terminal zu lesen

GPIO.setup(LED, GPIO.OUT)
while True:
      try:
            # Kleines Auswahlmenu wird erstellt
            os.system("clear")
            print "AN = 1"          # wahl == 1
            print "Aus =2"          # wahl == 2
            print "Beenden = 3"     # wahl == 3

            #Die Eingabe der Variablen wahl wird ueber die Kommandozeile uebergeben
            wahl = input("Schalte Licht an oder Aus:")

            #Wenn die folgende Zahlen gedrueckt werden:
            # 1: LED wird eingeschaltet
            # 2: LED wird ausgeschaltet
            # 3: Programm wird beendet
            if wahl == 1:
                  print"Licht AN"
                  GPIO.output(LED, GPIO.HIGH)
                  time.sleep(0.3)

            if wahl == 2:
                  print"Licht AUS"
                  GPIO.output(LED, GPIO.LOW)
                  time.sleep(0.3)
            if wahl == 3:
                  GPIO.cleanup()
                  quit()


      except KeyboardInterrupt:
            GPIO.cleanup()
            quit()
```

Python Code:03_LED_PWM.py

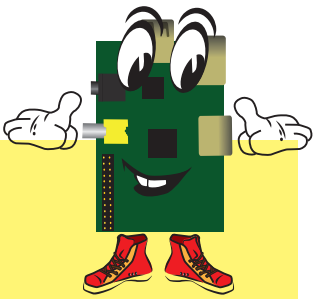
Schritt 1: Python Datei erstellen

Lege eine neue Python Datei an:

```
sudo nano 03_LED_PWM.py
```

Schritt 2: Der Python Code

Das Programm lässt die LED mit der Pulsweitenmodulation langsam angehen und ausgehen. Dabei wird die Spannung langsam erhöht und die die LED erhellt oder erlischt langsam.



Die `for i in range (100)` Schleife wird so lange ausgeführt bis Sie das Ergebnis 0 erreicht und springt folglich in die zweite `for` Schleife

Schritt 3: Ausführen

```
sudo python 03_LED_PWM.py
```

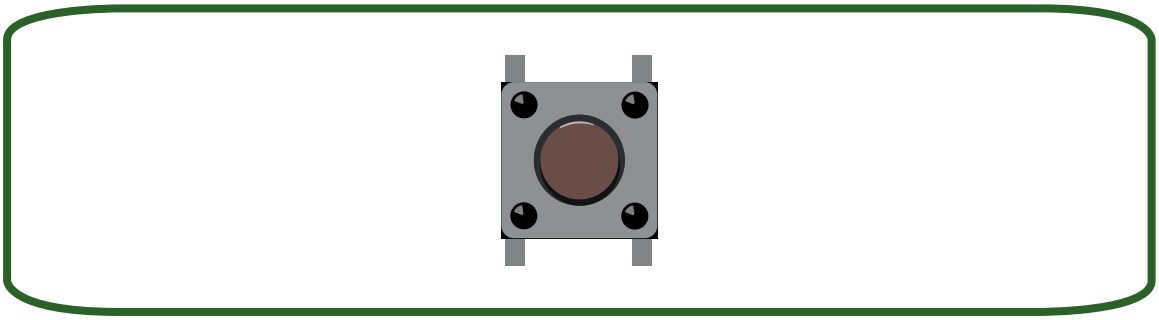
Python Code:03_LED_PWM.py

```
#!/usr/bin/python
import RPi.GPIO as GPIO
import time
import os

GPIO.setmode(GPIO.BOARD)
GPIO.setwarnings(False)

#Variablen
LED = 12

GPIO.setup(LED, GPIO.OUT)

LED_PWM = GPIO.PWM(LED, 50) #Der LED Pin wird auf 50 Prozent gesetzt
LED_PWM.start(0) #Der Anfangswert von PWM wird auf 0 gesetzt

os.system ("clear")


while True: #Solange die Bedingung wahr ist wird die while-Schleife ausgefuehrt
      try:
            for i in range (100):
                        #GPIO-Pin wird von 0 auf 100 Prozent in angegebener Zeit
                        #gesetzt
                        LED_PWM.ChangeDutyCycle(i)    #Tastverhältnis ist i

                        time.sleep(0.02) #Wartet die angegebene Zeit ab
            for i in range (100):
#GPIO-Pin wird von 100 auf 0 Prozent in angegebener Zeit gesetzt
#Tastverhältnis 100 - 1 bis 0, dann springt er wieder in die obere for Schleife
                        LED_PWM.ChangeDutyCycle(100-i)
                        time.sleep(0.02)


      except KeyboardInterrupt: #Mit STRG-C wir das Programm unterbrochen
            print"Ende"
            LED_PWM.stop() #Beendet die Modulation
            GPIO.cleanup() #Resetet die GPIO-Pins
            quit()       #Beendet sauber das Skript
```

Grundlagen

Der Taster

Ein Taster kann z.B. eine LED ein- oder ausschalten. In unberührten Zustand (offen) ist der Stromkreis unterbrochen, in gedrückten Zustand (geschlossen) ist die Verbindung hergestellt. Wird der Taster gedrückt, ist der Stromkreis geschlossen und die LED leuchtet. Wird der Taster losgelassen, ist der Stromkreis unterbrochen und die LED schaltet sich aus.



Output

In der vorherigen Schaltung hast du gelernt, dass eine LED ein OUTPUT Element ist. Dabei gibt es zwei Zustände:

Logischer (physikalischer) Zustand	Bedeutung in Volt
0 (LOW)	0V
1 (HIGH)	3,3V

INPUT (Taster)

Im nächsten Schaltplan, wird ein Taster angeschlossen. Wie du dir vielleicht schon denken kannst, ist der Taster ein INPUT Element. Drückst du die Taste, werden Informationen aus dem Python Code übermittelt, der Stromkreis ist geschlossen und die LED leuchtet auf. Lässt du ihn los, werden keine Informationen übermittelt und die LED geht aus.
Der Unterschied beim INPUT ist, dass trotzdem Strom fließt:

Bedeutung in Volt	Logischer (physikalischer) Zustand
ungefähr 0-1,2V	0 (LOW)
1,3-3,3V	1 (HIGH)

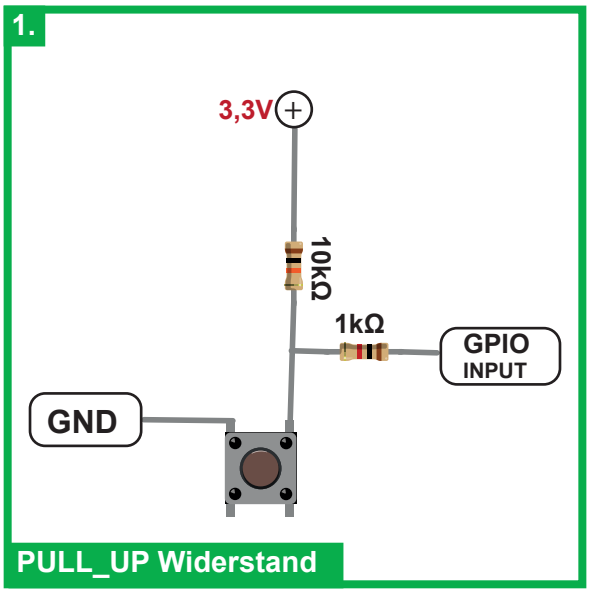
PULL UP/DOWN Widerstand

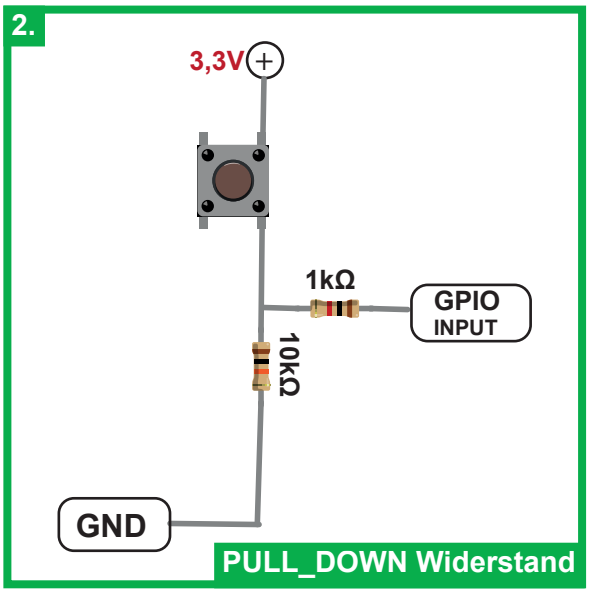
Für diesen Grund brauchst du noch ein PULL UP (=Spannung hoch ziehen)
PULL DOWN (=Spannung runter ziehen) Widerstand der im RasPi integriert ist, um Unregelmäßigkeiten in der Spannung zu verhindern. Du ziehst also mit dem PULL UP die Spannung des Pins auf 3,3V und mit PULL DOWN ziehst du sie auf GND.

Grundlagen

PULL UP/DOWN Widerstand

Es gibt zwei Möglichkeiten den Taster mit dem RasPi zu verbinden. Willst du, dass der INPUT Pin physikalisch LOW erhält, wenn die Taste gedrückt wird, so wird der Widerstand als PULL_UP genutzt und muss folgendermaßen aufgebaut werden (BILD 1). Dabei liegt der Kontakt zwischen dem INPUT des GPIO Pins und GND.

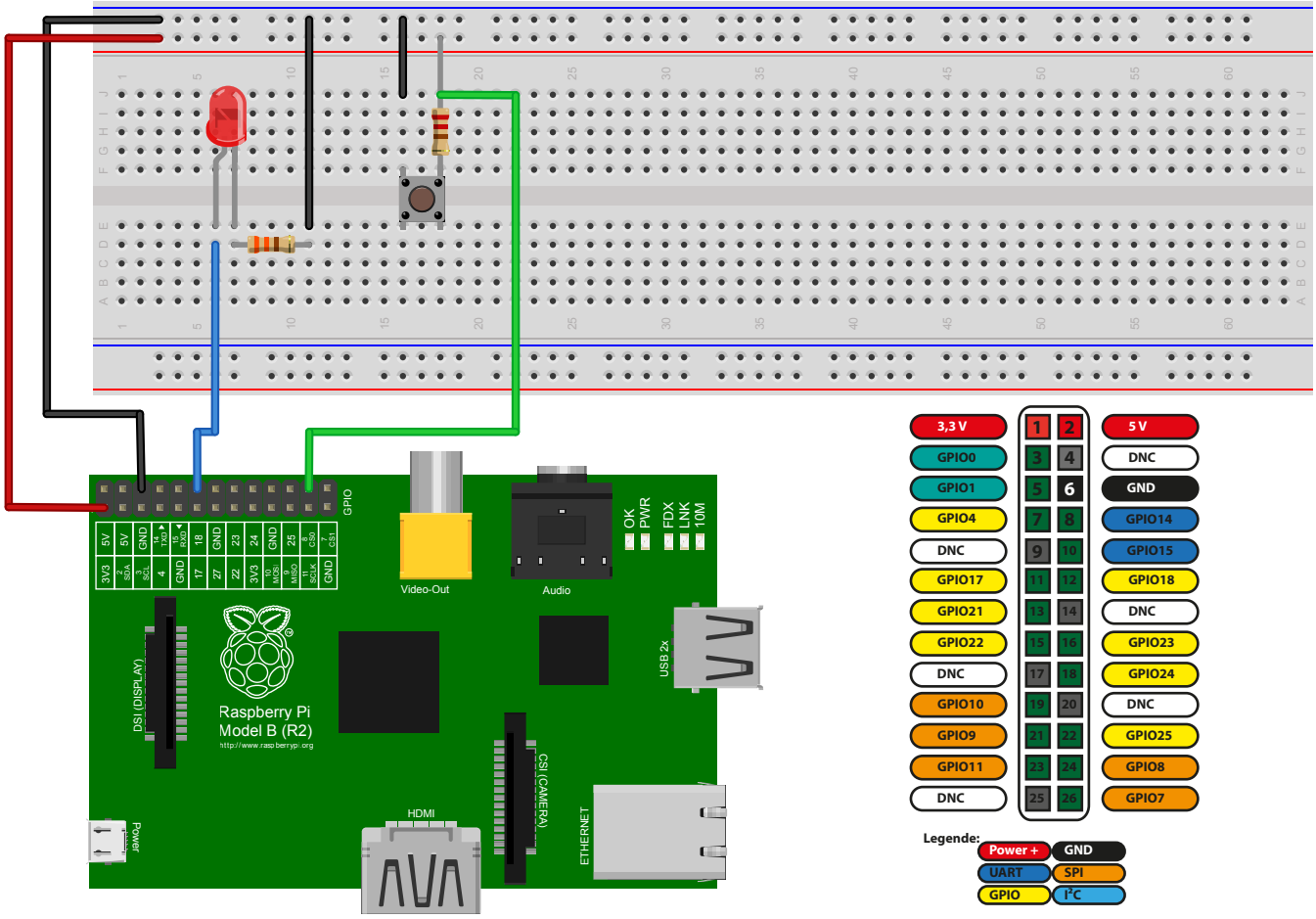




Willst du, dass der INPUT physikalisch HIGH erhält, wenn die Taste gedrückt wird, so wird der Widerstand als PULL_DOWN geschaltet. Der Kontakt liegt zwischen dem INPUT und den 3,3V Pin. Der PULL_DOWN -Widerstand liegt zwischen dem INPUT und GND. Beim Öffnen des Kontaktes zieht der PULL_DOWN-Widerstand die Spannung am INPUT hinunter auf GND, was den physikalischen Zustand LOW entspricht (BILD 2).
Zum Schutz wird noch 1kOhm Widerstand für den GPIO Pin eingebaut.
.

Aufbau nach Schaltplan

Jetzt kannst du die Schaltung um die LED mit einen Taster erweitern. Stecke den Taster auf das Streckbrett und baue die Schaltung nach dem Schaltplan auf. Dabei wird das linke Beinchen des Tasters mit dem GND-Pin (Pin 6) und das rechte Beinchen mit dem Pin 24 (GPIO8) verbunden. An das linke Beinchen wird noch der 1kOhm Widerstand gesteckt, der mit dem zweiten Ende an den Pin 1 (3,3V) verbunden wird.



Elemente	Farbe	Anschluss	Farbe	RasPi
Taster	■	linkes Beinchen	■	PIN 6 (GND)
Taster		rechtes Beinchen	■	PIN 24 (GPIO8)
1kOhm Widerstand		An rechtes Beinchen des Tasters		
1kOhm Widerstand		An den 3,3V Pin von RasPi	■	PIN 1 (3,3V)
LED		Kathode	■	PIN 12 (GPIO18)
LED		Anode		
330 Ohm Widerstand		An die Anode der LED		
330 Ohm Widerstand		An GND des RasPi´s	■	PIN 6 (GND)

Python Code:04_Input_Taster.py

Schritt 1: Python Datei erstellen

```
sudo nano 04_Input_Taster.py
```

Schritt 2: Der Python Code

In dieser Schaltung wird die LED mit einen Taster eingeschaltet -> Drückst du den Knopf, geht die LED an, lässt du den Knopf los, geht die LED aus.

Der Taster wird im Code mit dem PULL_UP_DOWN Widerstand in Python eingestellt. Drückst du den Taster, wird der PIN 24 (GPIO8) auf LOW gesetzt und die LED am PIN 12 (GPIO18) wird durch die Programmierung auf HIGH gesetzt. Lässt du los geht die LED aus.

Als erstes muss der Pin 24 als Input deklariert werden, damit weiß Python, dass es sich um ein Eingabeelement handelt.

Die Tasterabfrage wird in eine `while True` Schleife gepackt und mit `if` sagst du dem Taster, was passieren soll, wenn er gedrückt ist ansonsten `else` wenn er nicht gedrückt ist.

Python Code:04_Input_Taster.py

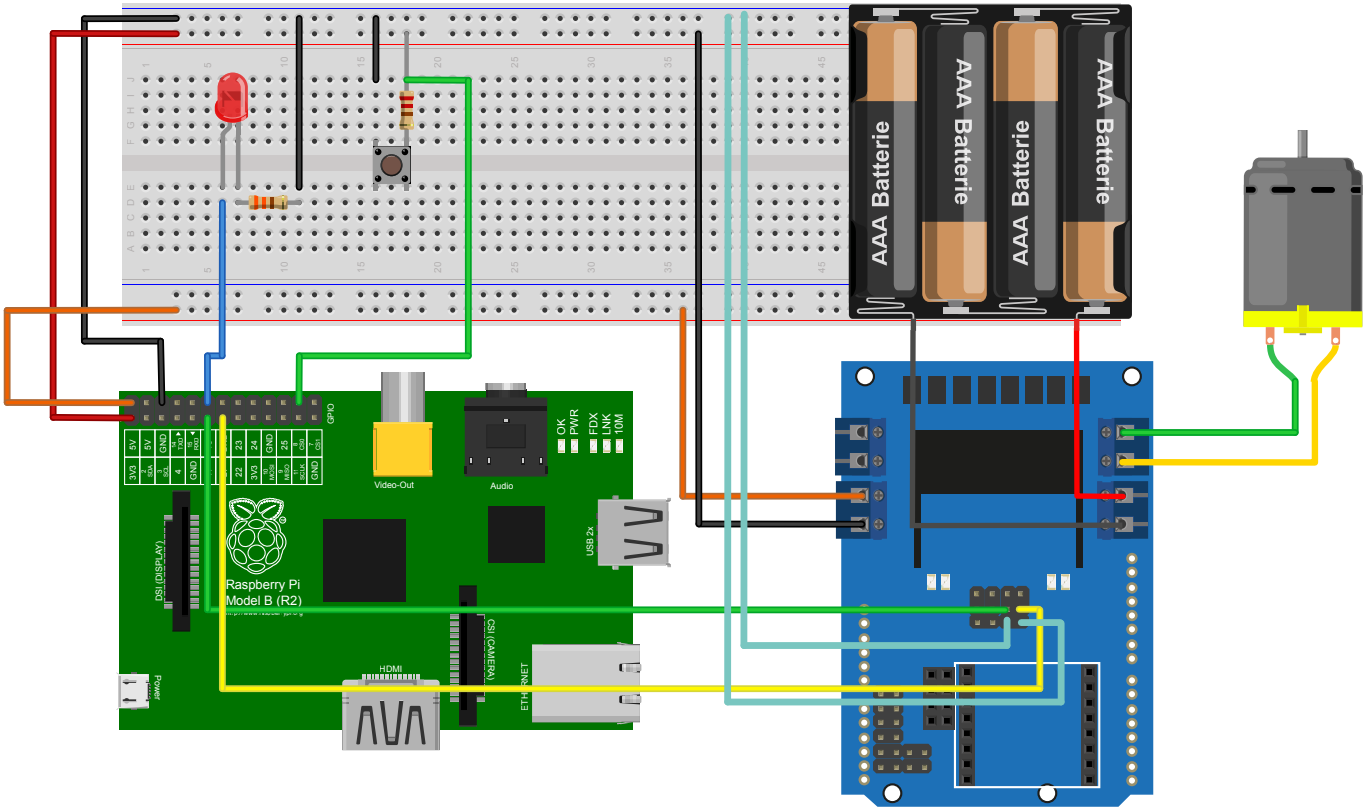
```
#!/usr/bin/python
import RPi.GPIO as GPIO
import time
import os

GPIO.setmode(GPIO.BOARD)

GPIO.setwarnings(False)

#Variablen
LED = 12
Taster = 24

#Setzt den Pin LED als Output
GPIO.setup(LED, GPIO.OUT)

#Der Taster wird auf Input gesetzt und mit dem PULL-UP Widerstand ausgestattet
GPIO.setup(Taster, GPIO.IN, pull_up_down=GPIO.PUD_UP)

os.system ("clear")

#Solange die Bedingung auf True (wahr) gesetzt wird :
#D.h. der Taster ist gedrueckt (if), leuchtet die LED
#(else)Ansonsten ist sie aus
while True:
      try:
            if GPIO.input(Taster) == True: #Taster ist gedrueckt
                  os.system ("clear")
                  print "LED AN"
                  GPIO.output(LED, GPIO.HIGH) #LED leuchtet
                  time.sleep(0.3) #Wartet die angegebene Zeit ab

            else:
                  os.system ("clear")
                  print "LED AUS"
                  GPIO.output(LED, GPIO.LOW) #LED ist aus
                  time.sleep(0.3)


      except KeyboardInterrupt:
            GPIO.cleanup()
            quit()
```

Python Code:05_Interrupt_Taster.py

```
#Der Anfangscode von letztem Beispiel, ist der Gleiche, der einzige Unterschied
ist, die Variablen wurden um Status = 0 erweitert. Du kannst den Code bis zu while
True aus 04_Input_Taster.py übernehmen.

#Variablen
LED = 12
Taster = 24
Status = 0
#An dieser Stelle werden Funktionen definiert, die im Code eingesetzt werden
#Licht_AN setzt den LED-Pin auf HIGH und übergibt den Text LED AN
#Licht_AUS setzt den LED-Pin auf LOW und übergibt den Text LED AUS
#ende resetet die PINs und schließt das Programm sauber ab.
def Licht_AN():
        os.system(„clear")
        print(„LED AN")
        GPIO.output(LED, True)
        time.sleep(0.3)

def Licht_AUS():
        os.system(„clear")
        print(„LED AUS")
        GPIO.output(LED, False)
        time.sleep(0.3)

def ende ():
            GPIO.cleanup()
            quit()
while True:
      try:
            #Erst nach dem druecken der Taste wird die Schleife ausgeführt
            GPIO.wait_for_edge(Taster, GPIO.RISING) #=>Interrupt Taster, so lange
            #die Taste nicht gedrückt wird, startet die Schleife nicht.


            #Der Status des Tasters wird überprüft und dementsprechend gesetzt
            if GPIO.input(Taster) == True:
                  if (Status == 1):
                        Status = 0
                  elif (Status == 0):
                        Status = 1
            #Wenn der Status des Tasters = 1 LED an, Status des Tasters = 0 LED aus
            if (Status == 1):
                  Licht_AN() #Die Funktion Licht_AN wird aufgerufen
            elif (Status == 0):
                  Licht_AUS()#Die Funktion Licht_AUS wird ausgeführt
      except KeyboardInterrupt:
            ende()
```

Aufbau nach Schaltplan

In diesen Teil des Moduls wird deine Schaltung um ein DC-Motor erweitert. Dazu musst du die Bauelemente nach dem Schaltplan zusammenstecken. Für die Bastelarbeit brauchst du ein Schraubenzieher, um die Kabel in der Motor H-Brücke festzumachen. Wo jedes Kabel festgeklemmt oder gesteckt wird (siehe Tabelle unten).



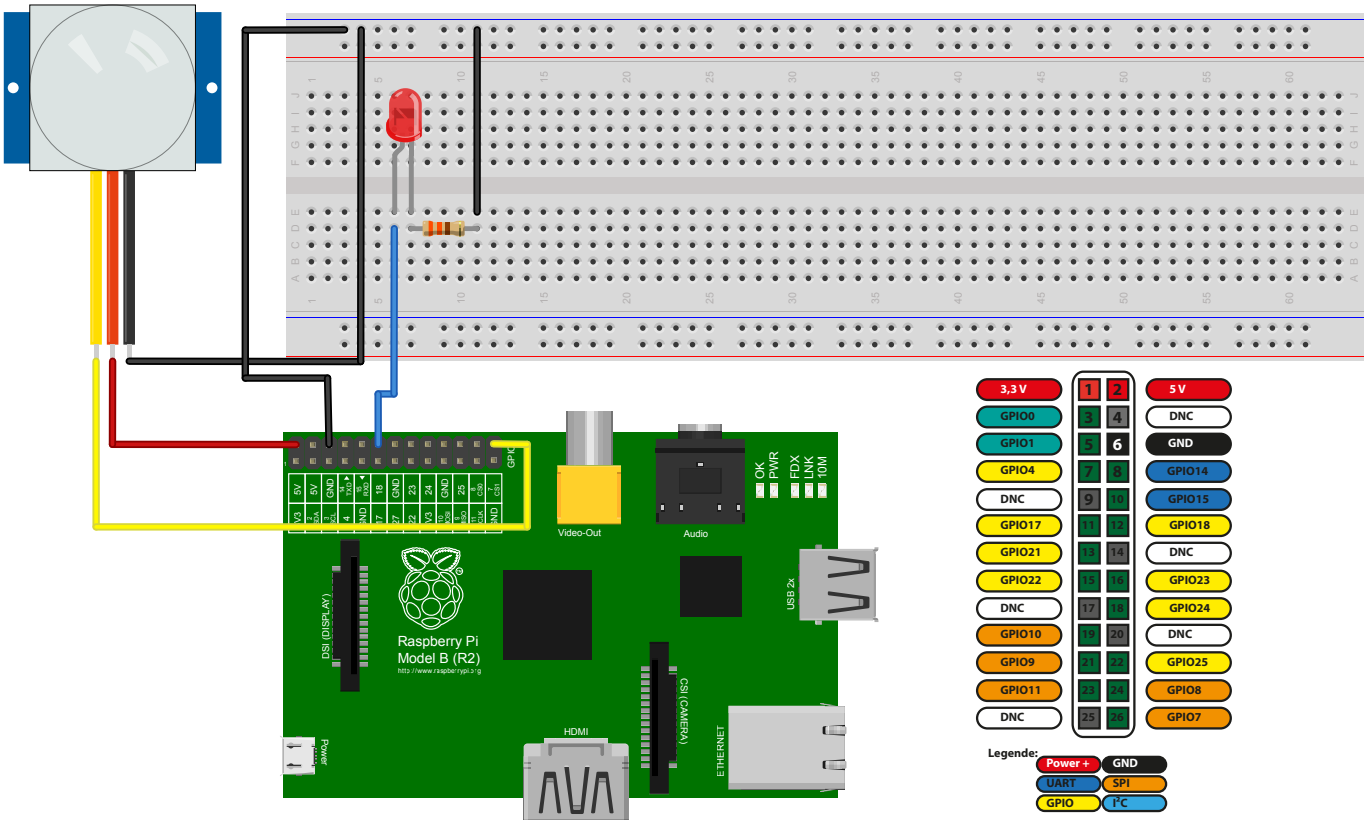
Elemente	Farbe	L298N H-Brücke	Farbe	RasPi
Motor		OUT 3		
Motor		OUT 4		
Batteriehalter		Vsupply		
Batteriehalter		GND		
RasPi		5V		5V
RasPi		GND		GND
L298N (Motor)		IN3		PIN 11 (GPIO17)
L298N (Motor)		GND (IN3)		GND
L298N (Motor)		IN4		PIN 13 (GPIO21)
L298N (Motor)		GND (IN4)		GND

Python Code:06_Motor.py

```
#!/usr/bin/python
import RPi.GPIO as GPIO
import time
import os

GPIO.setmode(GPIO.BOARD)
GPIO.setwarnings(False)

#Variablen
LED = 12
TSensor = 24
MVOR = 11 #Der Pin wird für Drehung des Motors auf Vorwärts deklariert.
MZUR = 13 #Der Pin wird für Drehung des Motors auf Rückwärts deklariert.

#Output GPIO
GPIO.setup(MVOR,GPIO.OUT)
GPIO.setup(MZUR,GPIO.OUT)
GPIO.setup(LED,GPIO.OUT)

#Input GPIO
GPIO.setup(TSensor,GPIO.IN)
os.system(„clear")

#Funktion fuer Vorwaerts fahren
def fwd():
        GPIO.output(MVOR, True)      #PIN MVOR wird auf HIGH gesetzt
        GPIO.output(MZUR, False)     #PIN MZUR wird auf LOW gesetzt
        GPIO.output(LED, False)      #PIN LED ist aus
#Funktion fuer Rueckwaerts fahren
def back():
            print('Rueckwaerts')
            GPIO.output(MVOR, False)
            GPIO.output(MZUR, True)
            GPIO.output(LED, True)
            time.sleep(3)
def ende ():
            GPIO.cleanup()
            quit()
while True:
      #Drucktaster (Tsensor) wird als DTaster deklariert
      DTaster = GPIO.input(TSensor)
      #Wenn der Taster nicht gedrueckt wird dreht sich das Rad nach vorne - Funktion fwd()
      try:
            if DTaster == True:
                  fwd ()
            #ansonsten wird die Funktion back() rueckwaerts aufgerufen
            else:
                  back ()
       except KeyboardInterrupt:
            ende() #Funktion ende wird aufgerufen
```

Aufbau nach Schaltplan

Entferne aus dem Steckbrett alle elektrischen Bauelemente ab und schließe den Bewegungssensor mit seinen drei Kabeln an den RasPi an. Dabei steckst du den GND-Kabel in PIN 6 (GND), den VCC-Anschluss an den 5V Pin und den Output an den Pin 26 (GPIO7).



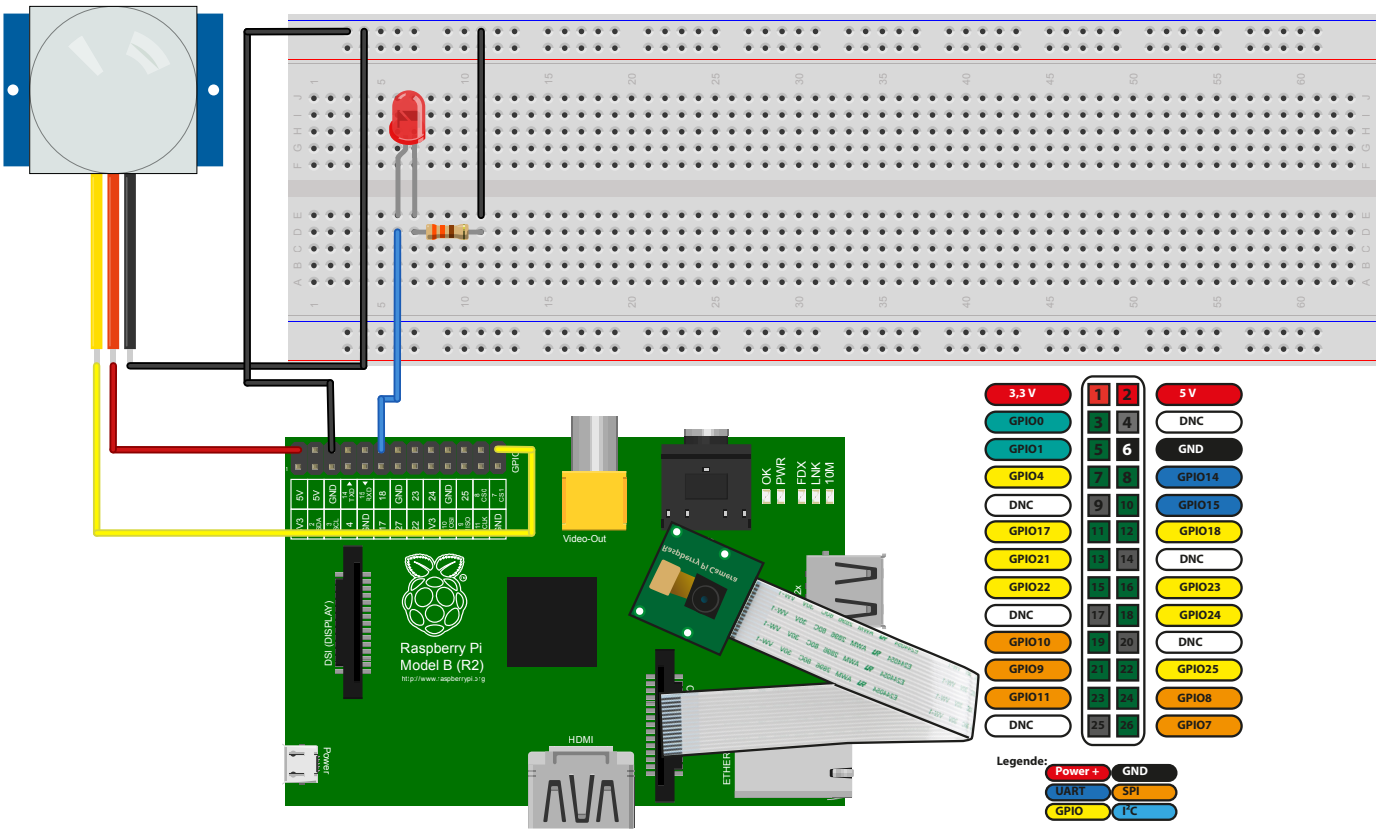
Elemente	Farbe	Anschluss	Farbe	RasPi
PIR-Sensor		OUT		PIN 26 (GPIO7)
PIR-Sensor		VCC		PIN 2 (5V)
PIR-Sensor		GND		PIN 6 (GND)
LED		Anode		
LED		Kathode		PIN 12 (GPIO18)
330 Ohm Widerstand		An die Anode der LED		
330 Ohm Widerstand		An GND des RasPi´s		PIN 11 (GND)

Python Code:07_Bewegungssensor.py

```
#!/usr/bin/python
import RPi.GPIO as GPIO
import time
import os

GPIO.setmode(GPIO.BOARD)
GPIO.setwarnings(False)

#Variablen
LED = 12
beweg = 26

#Output GPIO
GPIO.setup(LED,GPIO.OUT)

#Input GPIO
GPIO.setup(beweg,GPIO.IN)

#Zustand-Variablen
Strom_State = 0
Vor_State = 0

os.system(„clear")

def ende ():
            GPIO.cleanup()
            quit()
try:
      print "Warte auf Bewegung"
      while GPIO.input(beweg) == 1: #Pin 26 wird auf Wahr gesetzt
            Strom_State = 0
      print "Ferig"

      while True:
      #Lese Bewegungsstatus
            Strom_State = GPIO.input(beweg)
            if Strom_State == 1 and Vor_State == 0:
                  print "Bewegung erkannt"
                  GPIO.output(LED, GPIO.HIGH) #GPIO-Pin wird auf High gesetzt
                  time.sleep(2) #Wartet die angegebene Zeit ab

                  GPIO.output(LED, GPIO.LOW) #Schaltet die LED aus
                  time.sleep(0.3)
            elif Strom_State == 0 and Vor_State == 1:
                  #Springt zurueck zum Status Fertig
                  print "Fertig"
                  Vor_State = 0
                  time.sleep (0,01)
except KeyboardInterrupt: #Mit STRG-C wir das Programm unterbrochen
            ende()
```

Aufbau nach Schaltplan

Im letzten Beispiel musst du die RasPi Cam in das CSI einstecken. Dabei ziehst du den Verschluss von CSI hoch und steckst das Flexkabel rein, dass die blau Seite zur SD-Karte schaut. Nach dem Start von Raspbian musst du nur noch unter Raspi-config die Kamera auf Enable stellen.

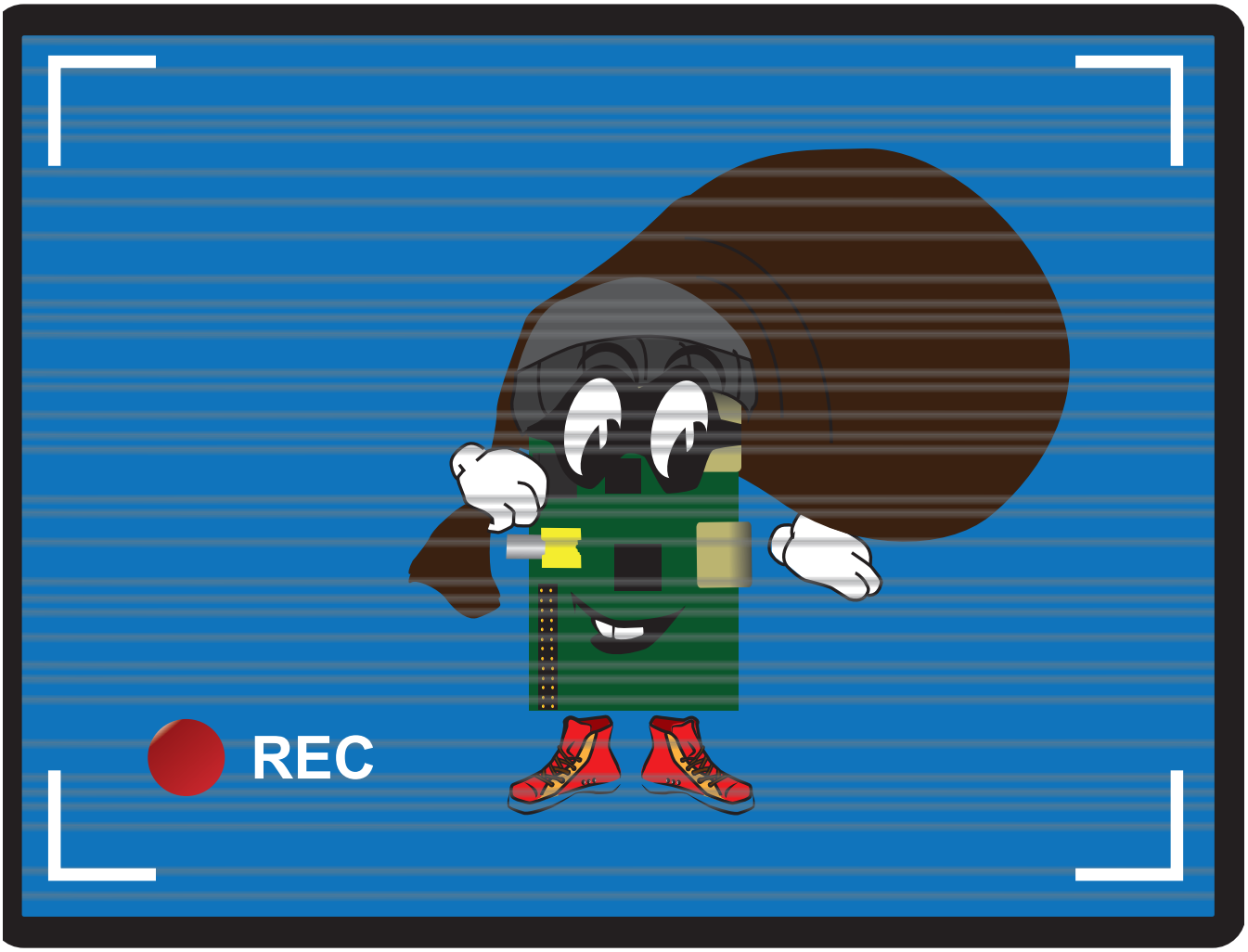


Elemente	Farbe	Anschluss	Farbe	RasPi
PIR-Sensor		OUT		PIN 26 (GPIO7)
PIR-Sensor		VCC		PIN 2 (5V)
PIR-Sensor		GND		PIN 6 (GND)
LED		Anode		
LED		Kathode		PIN 12 (GPIO18)
330 Ohm Widerstand		An die Anode der LED		
330 Ohm Widerstand		An GND des RasPi´s		PIN 11 (GND)
RasPi-Cam		Flexkabel		CSI-Schnittstelle

Programmieren mit Python

```
#!/usr/bin/python
import RPi.GPIO as GPIO
import time
import os
import picamera #Importiert das Modul fuer die Kamera
import datetime #Modul fuer Datum und Urzeit

GPIO.setmode(GPIO.BOARD)
GPIO.setwarnings(False)

#Variablen
bewegS = 26
cam = picamera.PiCamera ()#der Code picamera.PiCamera wird in cam umbenannt

#Input GPIO
GPIO.setup(bewegS, GPIO.IN, pull_up_down = GPIO.PUD_DOWN)

#Zustand-Variablen
vor_State = False
bew_State = False

cam = picamera.PiCamera ()
os.system("clear")

#Funktion um Dateinamen zu generieren
def getFileName () :
      return datetime.datetime.now() .strftime ("%Y-%m-%d_%H.%M.%S.h264")

while True: #Solange die Bedingung wahr ist wird die while-Schleife ausgefuehrt
      time.sleep(0.1)
      vor_State = bew_State
      bew_State = GPIO.input(bewegS)
      if bew_State != vor_State: #Bewegungsstatus des Sensors wird angezeigt
            neuer_State = "HIGH" if bew_State else "LOW"
            print "GPIO pin %s is %s" % (bewegS, neuer_State)
            if bew_State:     #Wenn Bewegung erkannt = Aufnahme = Datei erstellen
                  fileName = getFileName () #Datei wird erstellt
                  cam.start_preview()     #Vorschaubild an
                  cam.start_recording(fileName)  #Aufnahme startet
                  time.sleep(5)     #5Sekunden aufnahme
                  os.system ("mv *.h264 Motion/")#Aufnahme in Motionorder verschieb
            else: #Wenn keine Bewegung = Kamera stop
                  cam.stop_preview()
                  cam.stop_recording()
#Bevor du den Pythoncode ausführst musst du noch ein Ordenr anlegen:
mkdir Motion
in diesen Ordner werden die Videos abgelegt.Die Videos kannst du dir so anschauen:
omxplayer DATEINAME.h264
```

Überwachungskamera



Modul 4

PI-Cam anschließen
Konfigurationsdatei editieren
Überwachung aktivieren

Inhalt

STEPS

- 1. Pi-Cam und Bewegungssensor anschließen
- 2. NAS-Server in Raspbian einbinden
- 3. Die Konfigurationsdatei editieren
- 4. Spycam starten

Lern Ergebnisse

- Eine Konfigurationsdatei für Pythonskript erstellen
- Mehrere Programme in Python kombinieren
- NAS-Laufwerk mounten

Software

NOOBS mit GPIO-Raspbian

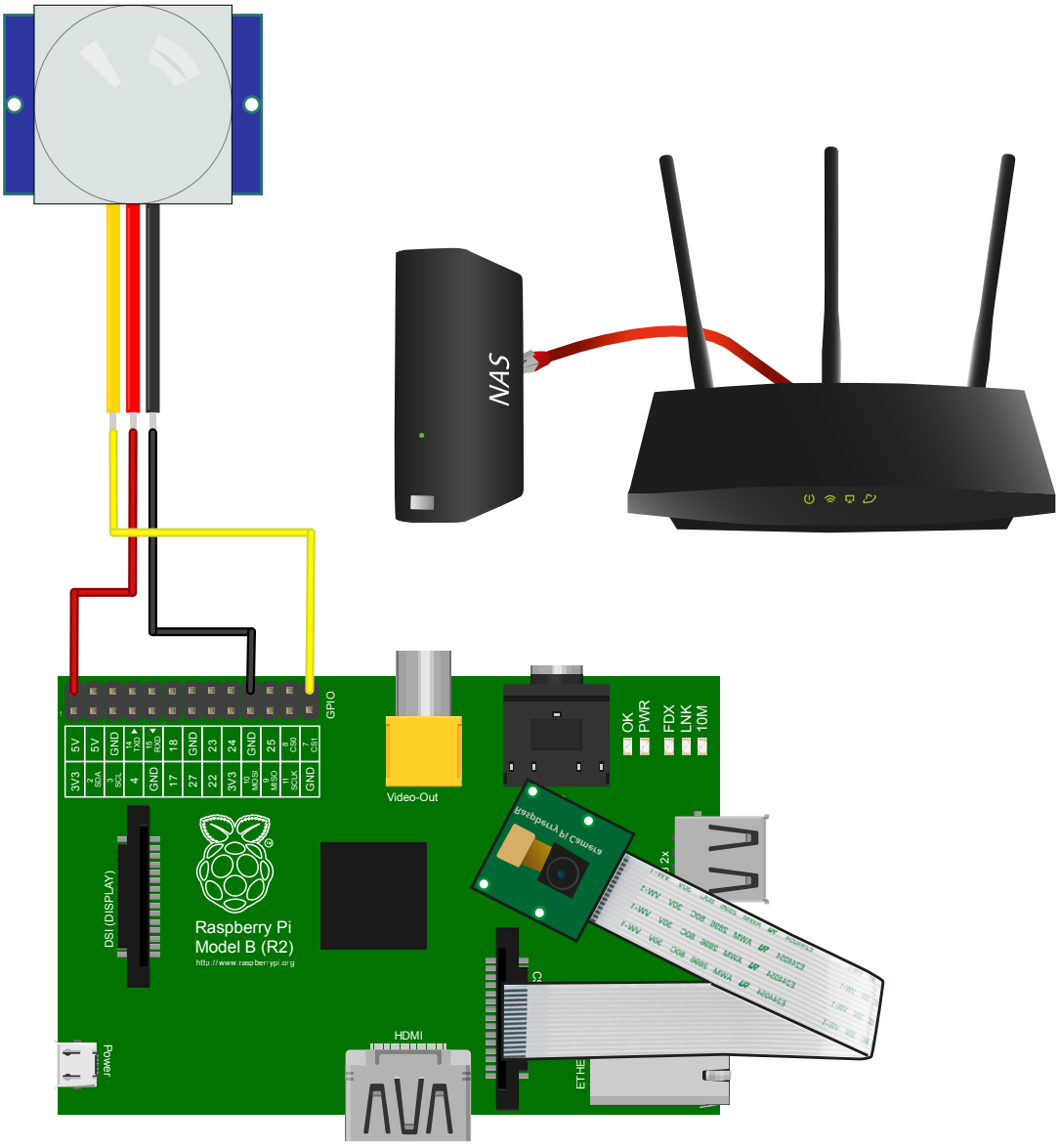
Raspbian:

Nano (Texteditor)
Python und config.txt

Beschreibung

Bei diesem Modul 4 handelt es sich um ein Überwachungssystem, dass durch Bewegungserkennung aktiviert wird. Das System zeichnet je nach Einstellung x-Sekunden ein Video und speichert es auf einem NAS-Server. Du wirst anschließend über eine E-Mail informiert, dass sich etwas vor der Kamera bewegt hat.

Zusammenbau



Farbe	PIR-Sensor	Farbe	RasPi
	OUT		PIN 26 (GPIO7)
	VCC		PIN 2 (5V)
	GND		PIN 9 (GND)

NAS-Server mounten

mount_nas konfigurieren

Als erstes musst du das NAS-Laufwerk in Raspbian einbinden. Dazu wechsle in das Verzeichnis von Spycam:

```
cd Spycam
```

In diesem Ordner befindet sich ein Skript, das du editieren musst:

```
nano mount_nas
```

Die Datei enthält den Pfad zum NAS und die Login-Daten zum Verzeichnis in dem die Videos kopiert werden.

```
#!/bin/bash
sudo mount -t cifs //1IP_ADRESSE_VOM_NAS/ORDNERNAMEN -o username=BENUTZERNAME,pass-
word=PASSWORT /media/PICAMREC

#Für das IP_ADRESSE_VOM_NAS musst du die IP-Adresse des NAS-Servers eintragen,
#gefolgt von ORDNERNAMEN des Ordners, wo die Videos gespeichert werden.  Als
#nächstes musst du den BENUTZERNAME eingeben und PASSWORT. Hinter den Login steht
#der Zielordner, in dem das NAS eingebunden wird: /media/PICAMREC
```

Die IP-Adresse oder die Domäne des NAS-Servers findest du im Benutzerhandbuch, wenn du dich in den Router einloggst oder auf den beiliegenden Zettel.

Jetzt kannst du die Datei abspeichern, und ausführen:

```
sudo ./mount_nas
```

Hat der Bildschirm keine Fehlermeldung angezeigt hat es geklappt.
Wechsle in das Verzeichnis, in dem das NAS gemountet wurde:

```
cd /media/PICAMREC
```

Der letzte Schritt ist, dass Raspbian die NAS-Festplatte immer beim Neustart in das System einbindet. Dazu musst du in die Autostartdatei crontab:

```
crontab –e
```

In dieser Datei musst du nur noch die „#“ entfernen und speichern. Beim nächsten Neustart ist das NAS dauerhaft eingebunden.

Die config.txt

Die config.txt hat einige wichtige Einstellungen für die Überwachungskamera, die dann das ausgeführte Python Skript übernimmt und durchführt. Ich zeige dir hier nur die wichtigsten Einstellungen, die du unbedingt wissen oder verändern musst damit das Python Programm reibungslos läuft. Wechsle zuerst in den Spycam Ordner und öffne die config.txt:

```
nano config.txt
```

```
#An dieser Stelle konfigurierst du, wo die Aufgezeichneten Videos kopiert werden
#soll.

[Datei1]    #Es ist eine Navigation für Python, damit er weiß, wo er die
            #Einstellungen suchen sol, für das NAS.

#In Verschieben werden die erstellten Dateien in einen vorgegebenen Ordner auf dem
#Nas kopiert. Der Ordner muss auf dem NAS notfalls erstellt werden.

Verschieben= sudo mv /home/pi/Spycam/*.h264 /media/PICAMREC/PICAM/
ende= sudo rm /home/pi/spycam/*.h264 -f

#ende löscht die Dateien auf der SD-Karte
```

```
#*********************************************************************************
[Picam] #Navigation für Python für die Kameraeinstellungen

#Schaltet die LED der Kamera EIN/AUS (True = AN, False = Aus)
LED = False

#Die Auflösung kann hier eingestellt werden (min.640 x 480, max. 1920 x 1080)
#Screensize (1920, 1080)
width= 1280
height= 720

#Bei einer Auflösung von 1920 x 1080 kann fps nur von 1 bis 30 Bilder/Sekunde
#eingestellt werden
#Bei einer Auflösung von 1280 x 720 kann fps bis von 1 bis 60 Bilder/Sekunde
#eingestellt werden
fps= 30

#Schaltet den Vorschaubildschirm in True = Volbildmodus
#Schaltet den Vorschaubildschirm in False = Normalmodus
fullscreen= True
#*********************************************************************************
```

Die config.txt

```
#*******************************************************************************
[Email]     #Die Navigation für Python für dn E-Mailversand.

#An wenn soll die Email versendet werden
#Da du die Email an dich gehen soll, musst du deine E-Mail und Login eintragen
#In diesem Beispiel ist das E-Mail-Konto für GMail vorkonfiguriert. Wenn du ein
#anderes Konto benutzen willst, musst du den Host und den Port deines

#E-Mail-Anbieters eintragen.
send_to = EMAILDOMAIN@gmail.com

#Emailkonto des Benutzer von dem die Email versendet werden soll
user = EMAILDOMAIN@gmail.com

#Passwort des Emailkontos
pwd = PASSWORT

#Der Hostname für den Emailserver und der Port (Voreingestell GMail)
host = smtp.gmail.com
port = 587

#Betreffzeile
sub = ALARM 919 !!!

#Der Inhalt der Email
msgtxt = Bewegung im Überwachten Raum wurde erkannt!

#*******************************************************************************
```

Die config.txt

Kamera aktivieren

Hast du die Einstellungen angepasst, müssen wir nur noch die Kamera aktivieren und das System neu-starten:

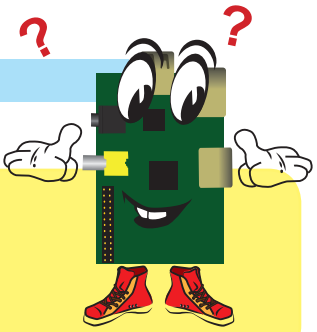
```
sudo raspi-config
```

Gehe dort auf **5 Enable Camera** => **Enable** und starte RasPi neu.

SPY-CAM ausführen

Wechsle in den Ordner von Spycam und Starte die Datei Spycam.py:

```
sudo python Spycam.py
```



Wieso funktioniert das Programm nicht korrekt? Irgendwo in der config.txt hast du entweder in der NAS-Konfiguration oder bei der E-Mail einen Fehler gemacht. Überprüfe alles genau. Machst du irgendwo in der config.txt einen Fehler, überspringt Python diese Stelle mit einer Fehlermeldung und springt zur nächsten Stelle im Code.

Video ansehen

Hast du die Einstellungen angepasst, müssen wir nur noch die Kamera aktivieren und das System neu-starten:

```
sudo raspi-config
```

Gehe zu 5 Enable Camera => Enable und anschließend auf Finish. Starte RasPi neu!
Nach dem einloggen, wechsle in den Ordner von Spycam und starte das Programm:

```
cd Spycam
sudo python Spycam.py
```

Python Code:Spycam.py

```
#!/usr/bin/python
import RPi.GPIO as GPIO
import time
import os
import picamera #Importiert das Modul für die Kamera
import datetime #Modul für Datum und Uhrzeit
import ConfigParser #Dient zur Erstellung und zum auslesen von Konfigurationsdateien
import smtplib #Bibliothek um Emails zu verschicken

GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)

#Variablen
bewegS = 7
cam = picamera.PiCamera ()
cfg = ConfigParser.ConfigParser()
cfg.read('config.txt')

#Input GPIO
GPIO.setup(bewegS, GPIO.IN, pull_up_down = GPIO.PUD_DOWN)

#Zustand-Variablen
vor_State = False
bew_State = False

os.system("clear")

#Funktion um die Video-Dateinamen zu generieren
def getFileName () :
      return datetime.datetime.now() .strftime ("%Y-%m-%d_%H.%M.%S.h264")

#Funktion für das kopieren der %Y-%m-%d_%H.%M.%S.h264 Datei
def copy_file():
      return cfg.get('Datei1','verschieben') #Ziel wird in der config.txt definiert

#Funktion zur Löschung der %Y-%m-%d_%H.%M.%S.h264 Datei auf der SD-Karte
def del_file():
  return cfg.get('Datei1','ende')   #Löscht alle *.h264 aus dem Ordner
```

Python Code:Spycam.py

```
#Kameraeinstellungen [Picam] in der config.txt werden ausgelesen
def cam_properties():
    cam.resolution = (cfg.getint('Picam','width'), cfg.getint('Picam','height'))
    cam.framerate = (cfg.getint('Picam','fps'))
    cam.preview_fullscreen = (cfg.getboolean('Picam','fullscreen'))
    cam.preview_alpha = (cfg.getint('Picam','alpha'))
    cam.raw_format = (cfg.get('Picam','format'))
    cam.annotate_text = (cfg.get('Picam','text'))
    cam.led = (cfg.getboolean('Picam','LED'))
    cam.sharpness = (cfg.getint('Picam','sharpness'))
    cam.contrast = (cfg.getint('Picam','contrast'))
    cam.brightness = (cfg.getint('Picam','brightness'))
    cam.saturation = (cfg.getint('Picam','saturation'))
    cam.ISO = (cfg.getint('Picam','saturation'))
    cam.video_stabilization = (cfg.getboolean('Picam','stabilization'))
    cam.meter_mode = (cfg.get('Picam','meter_mode'))
    cam.image_effect = (cfg.get('Picam','image_effect'))
    cam.rotation = (cfg.getint('Picam','rotation'))
    cam.hflip = (cfg.getboolean('Picam','hflip'))
    cam.vflip = (cfg.getboolean('Picam','vflip'))

#Einstellungen die in der [Email] in der config.txt werden ausgelesen
def send_mail ():
  to = (cfg.get('Email','send_to'))
  gmail_user = (cfg.get('Email','user'))
  gmail_pwd = (cfg.get('Email','pwd'))
  smtpserver = smtplib.SMTP((cfg.get('Email','host')), (cfg.getint('Email','port')))
  smtpserver.starttls()
  smtpserver.ehlo() # zusätzliche Zeichen zu bearbeiten erlauben
  smtpserver.login(gmail_user, gmail_pwd)
  subj = (cfg.get('Email','sub'))
  header = 'To:' + to + '\n' + 'From: ' + gmail_user + '\n' + 'Subject:%s\n'
%(subj)
  print header
  msg = header + (cfg.get('Email','msgtxt'))
  smtpserver.sendmail(gmail_user, to, msg)
  print 'done!'
  smtpserver.close()
```

Python Code:Spycam.py

```
while True: #Solange die Bedingung wahr ist wird die while-Schleife ausgefuehrt
      try:
            time.sleep(0.1)
            vor_State = bew_State
            bew_State = GPIO.input(bewegS)
            if bew_State != vor_State:
                  neuer_State = „HIGH“ if bew_State else „LOW“
                  if bew_State: #Bewegung erkannt Aufnahme starten
                        fileName = getFileName () #Videodatei wird erstellt
                        cam_properties()  #Kamereinstellungen werden übenohmen
                        cam.start_preview()      #Vorschaubild aktivieren
                        cam.start_recording(fileName)  #Aufnahme starten
                        send_mail ()       #Email wird verschickt
                        time.sleep(5)      #5 Sekunden aufzeichnen
                  else:                    #Ansonsten wenn keine Bewegung
                        cam.stop_preview()       #Vorschaubild deaktivieren
                        cam.stop_recording()     #Aufnahme beenden
                        os.system (copy_file())  #Funktion Videodatei kopieren
                        os.system (del_file())   #Alle *.h264 Dateien löschen
                                                 #Siehe config.txt

      except KeyboardInterrupt:
                  GPIO.cleanup()
                  cam.stop_preview()
                  quit()
```

Hardwareprogrammierung

Modul 5

GPIO ROBOT zusammenbauen
Motorbrückensteuerung
Entfernungssensor
Varianten der Steuerung
-Über Terminal, Web
Automodus

Inhalt

STEPS

- 1. GPIO ROBOT zusammenbauen
- 2. Motorsteuerung testen
- 3. Entfernungssensor testen
- 4. Steuerung über Internetbrowser
- 5. Automodus

Lern Ergebnisse

- Fortführender Einstieg in die Elektronik
- Mehrere Programme mit einander kombinieren
- Basis Wissen in Python Programmierung ausbauen

Software

NOOBS mit GPIO-Raspbian

Raspbian:

Nano (Texteditor)
Python
Webiopi
mjpg-streamer

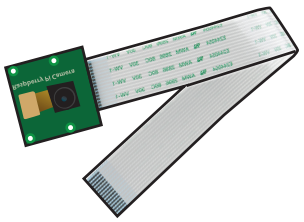
Beschreibung

GPIO ROBOT ist ein kleiner mobiler Roboter der auf dem RasPi aufbaut. Gesteuert wird er durch zwei Motoren, die durch Hilfe der Pulsweitenmodulationen vorwärts, rückwärts, nach links oder rechts gelenkt werden können. An die CSI-Schnittstelle ist das Kamera Modul von RasPi angeschlossen, die drahtlos das Bild auf einen Webbrowser von Smartphone, Tablet oder Desktop-PC liefert. Zusätzlich wird der GPIO ROBOT über den Browser und Java-Buttons ferngesteuert. Der Roboter kann auch in einen Automodus versetzt werden. Mithilfe eine Entfernungssensors erkennt das Gefährt Hindernisse ab bestimmter Entfernung und weicht diesen selbstständig durch programmierte Fahrmanöver aus.

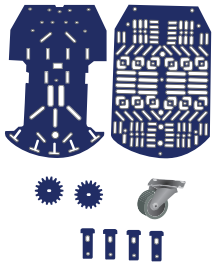
Hardware

Hardware

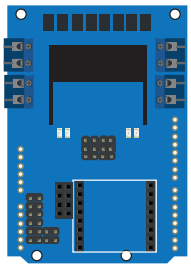
Raspi Cam



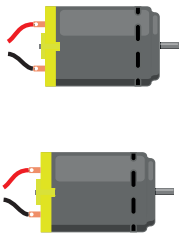
Robot Chassis



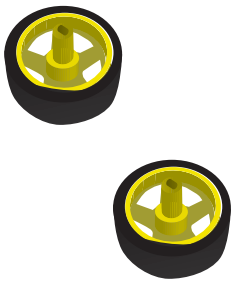
Motorbrücke



DC-Motoren



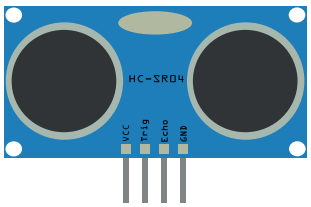
Räder



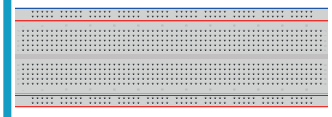
Batteriehalter

Power-Akku

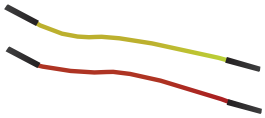
Entfernungssensor



Steckbrett

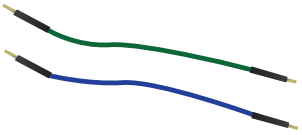


Jumperkabel



Female to Female

Jumperkabel



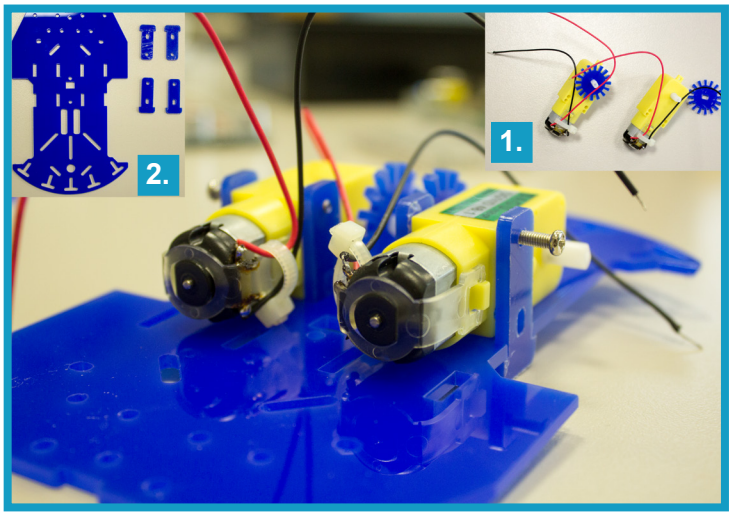
Male to Male

Weitere Teile

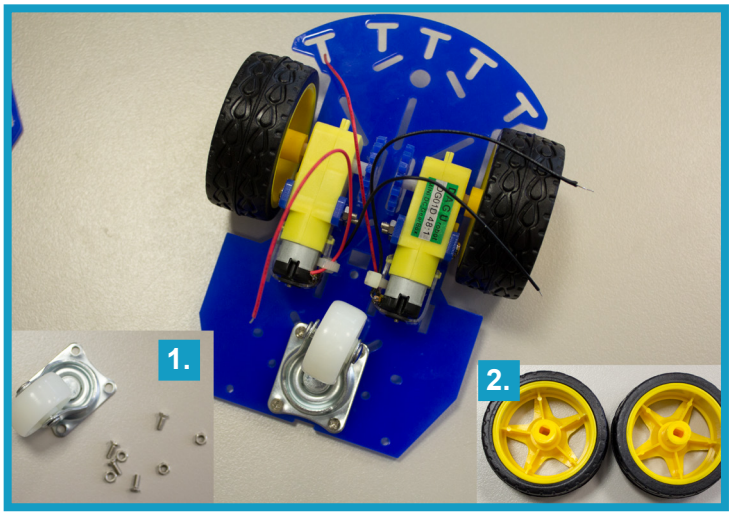
Schrauben
Muttern
Unterlegscheiben

Zusammenbau

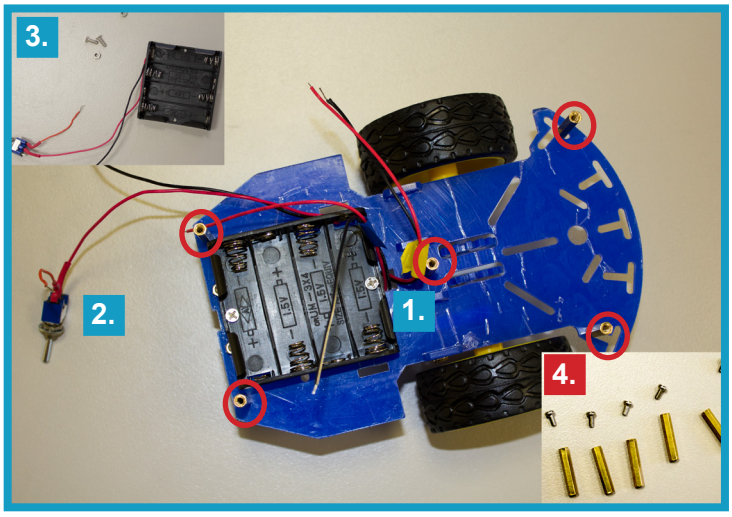
Beschreibung



1. Die beiden Motoren musst du auf der unteren **2.** Chassis-Platte mit den T-förmigen Teilen festmachen.
Die Motoren werden mit Schrauben und Müttern befestigt.



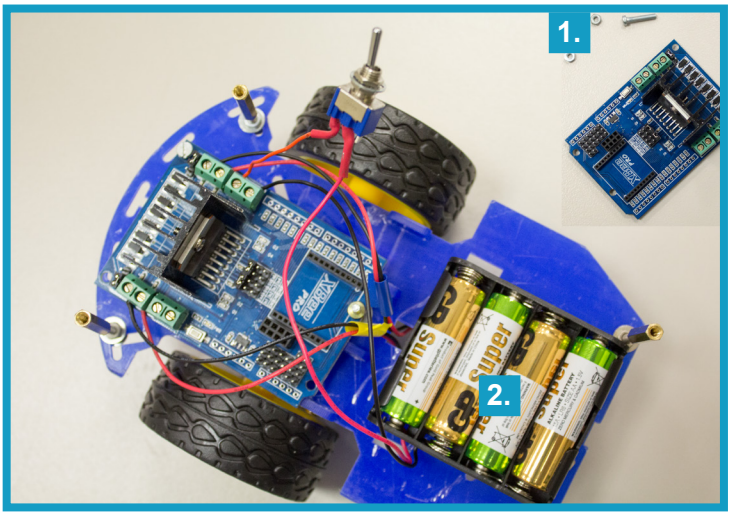
1. Befästige das Gelenkrad mit den vier Schrauben und Müttern vorne an die passenden Löcher.
2. An die Motoren montierst du die gelben Räder.



1. Ziehe die vier Kabel der Motoren durch das mittlere Loch im Chasis durch. Ein Tipp um die Kabel nicht zu verwechseln klebe verschdene Farben von Isolierband auf das jewalige Kabel.
2. An das rote Kabel(+) von Baterriehalter wird ein Schalter gelötet um die Motorsteuerung während dem Betrieb EIN/ AUS-zuschalten.
Befestige den **3.** Bateriehalter und die **4.** Gewindebuchsen, so wie im Bild.

Zusammenbau

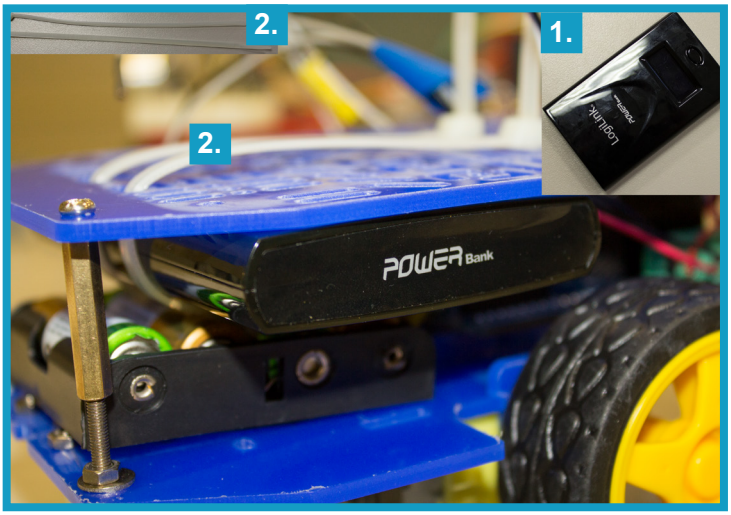
Beschreibung



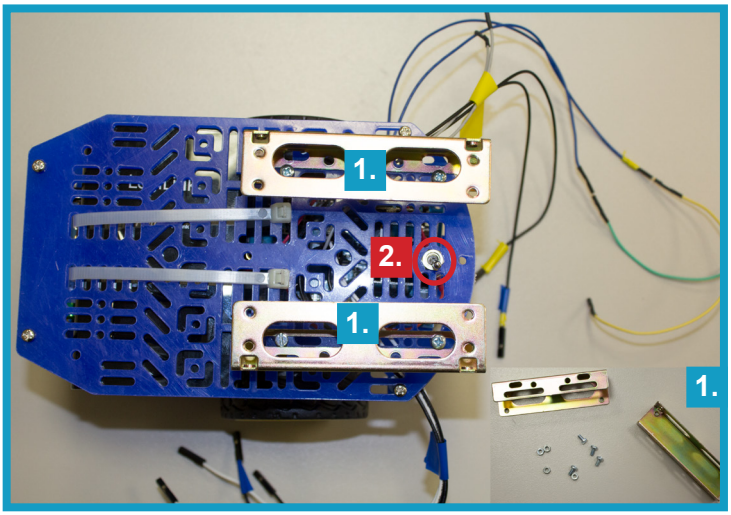
1. Die Motorbrücke musst du so positionieren, dass beim Fahren die Kabel nicht gegen die Räder reiben.

2. Die Batterien, kannst du auch schon in den Batteriehalter einstecken. Achte nur darauf, dass der Schalte auf AUS ist.

Die Kabel vom Batteriehalter und den Rädern werden an der H-Brücke festgeschraubt. Schaue in **Schaltung 1** nach an welche Klemmen sie festgeschraut werden.



2. Nehme zwei Kabelbinder und befästige damit, über den Batteriehalter, den
1. Powerakku.
Ziehe die zwei Kabelbinder (Siehe nächstes Bild) nicht allzu fest, damit der Akku rausnehmbar bleibt.

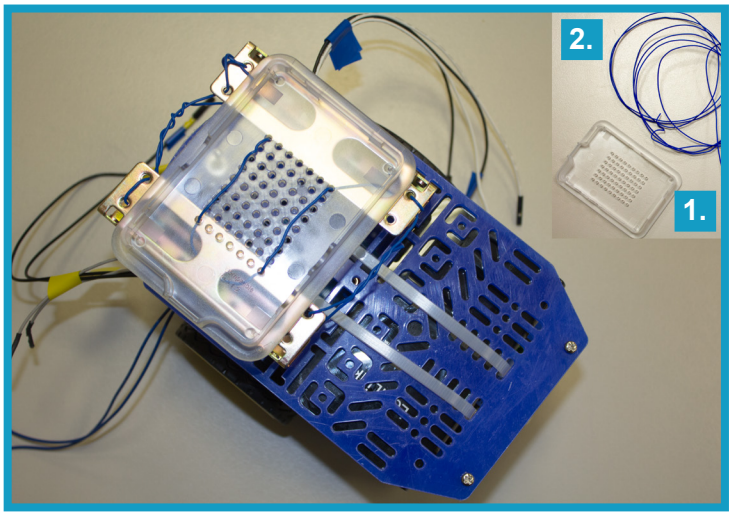


1. Als nächstes befästige die alten Fesplattenhalter mit Schrauben und Schraubenmüttern fest.

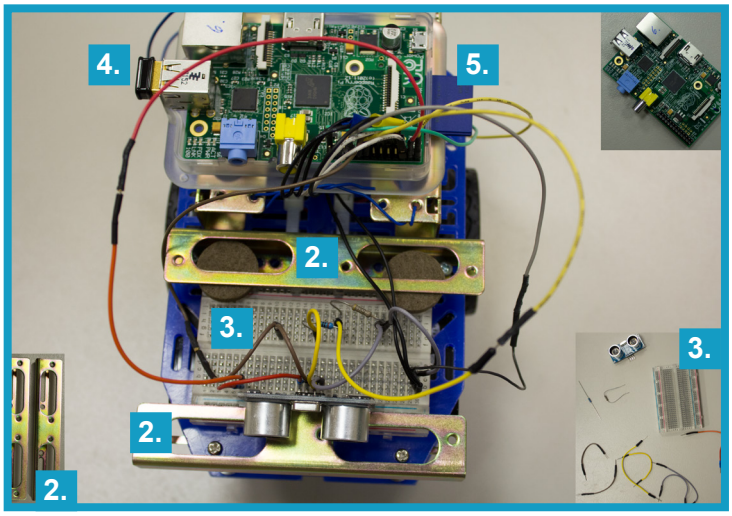
Jetzt kannst du das obere Chassisteil endgültig festschrauben.
2. Dabei kannst du den Schalter für den Baterriehalter an obere Chassis festschrauben.

Zusammenbau

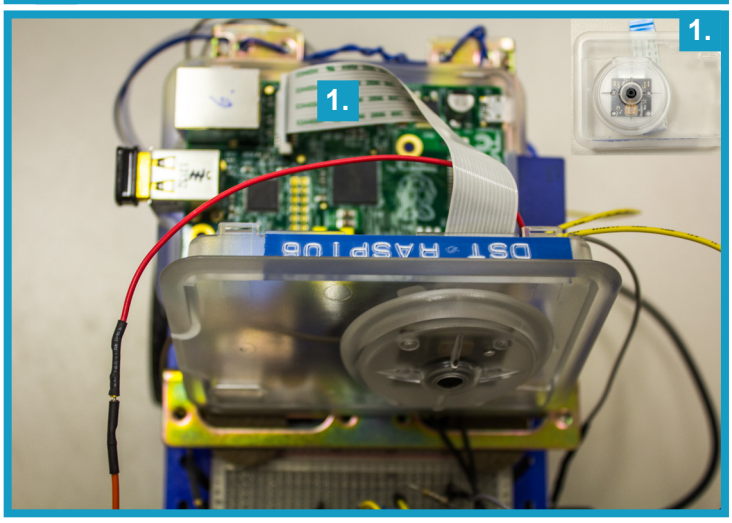
Beschreibung



Du hast verschiedene Möglichkeiten den RasPi oben zu befästigen. in diesen Fall wurde das untere Gehäuse **1.** von RasPi mit einen Draht **2.** an den alten Festplatten-halterungen festgebunden.



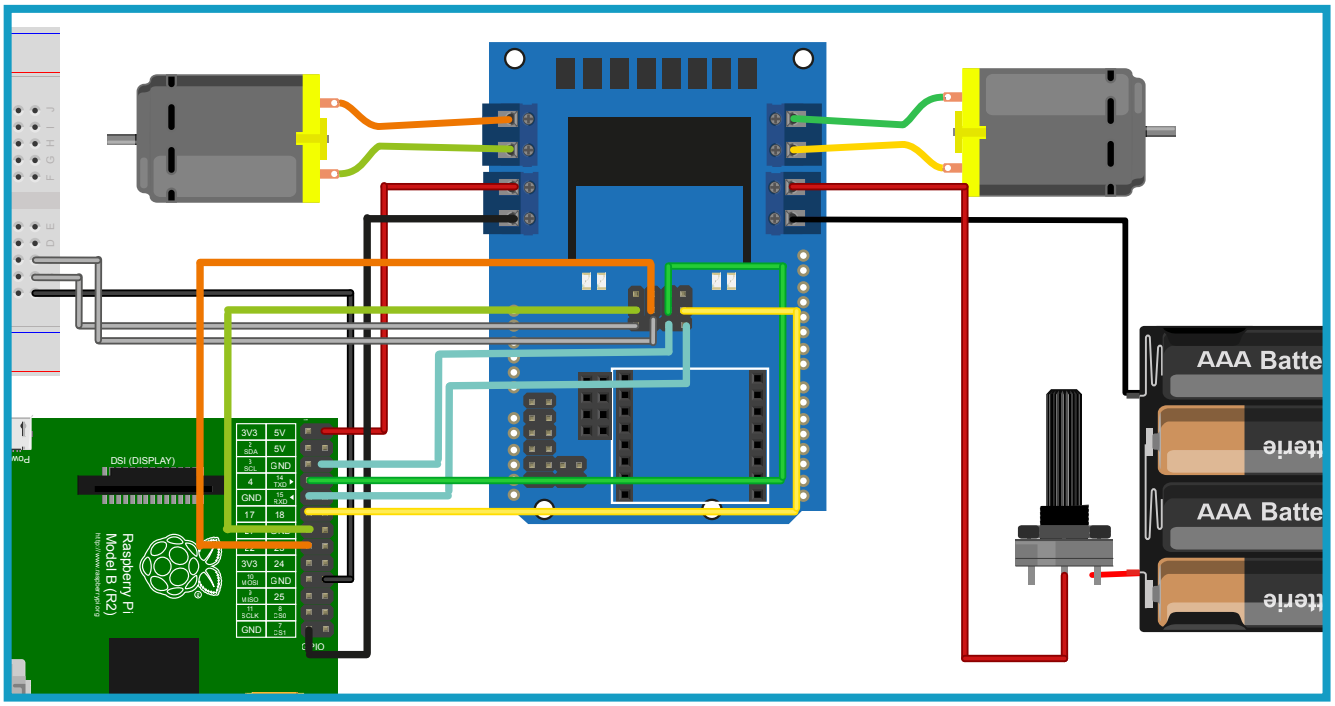
Bevor du den **1.** RasPi in das Gehäuse reinsteckst, musst du die **2.** Festplattenhalterungen vorne montieren.
Diese müssen so montiert sein, dass das **3.** Steckbrett zusammen mit der **Schaltung 2** sich hineinschieben lässt.
Jetzt stecke den **4.** WLAN-Adapter und die **4.** SD-Karte in den RasPi ein.



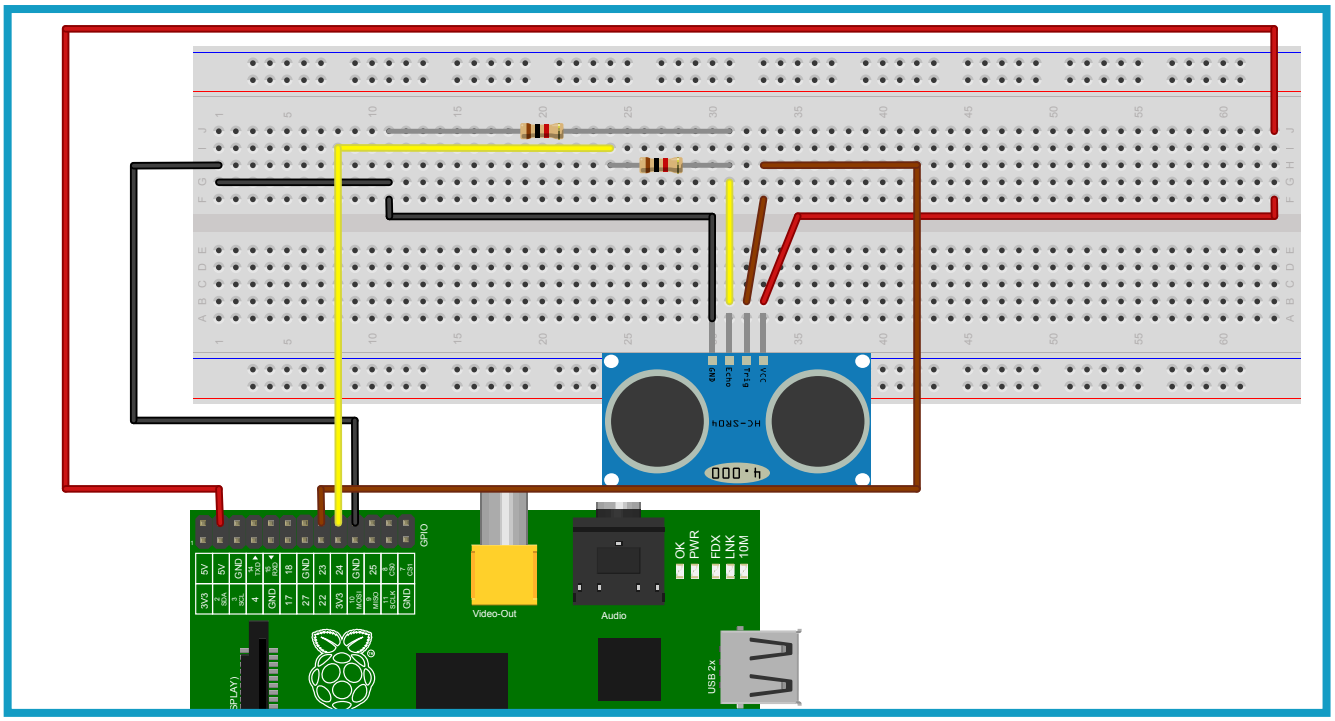
1. Im letzten Schritt wird die Pi-Kamera an die CSI-Schnittstelle gesteckt und festge-klemmt.

Zusammenbau

Schaltung 1



Schaltung 2



Zusammenbau

Zur Schaltung 1

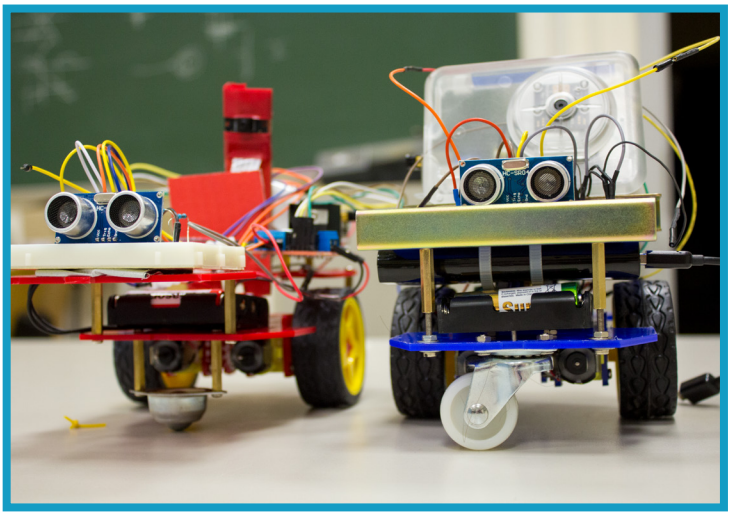
Elemente	Farbe	L298N H-Brücke	Farbe	RasPi
Motor Links		OUT 3		
Motor Links		OUT 4		
Motor Rechts		OUT 1		
Motor Rechts		OUT 2		
Batteriehalter		Vsupply		
Batteriehalter		GND		
RasPi		5V		5V
RasPi		GND		GND
L298N (Motor) L		IN3		PIN 7 (GPIO4)
L298N (Motor) L		GND (IN3)		GND
L298N (Motor) L		IN4		PIN 11 (GPIO1)
L298N (Motor) L		GND (IN4)		GND
L298N (Motor) R		IN1		PIN 13 (GPIO21)
L298N (Motor) R		GND (IN1)		
L298N (Motor) R		IN2		PIN 15 (GPIO22)
L298N (Motor) R		GND (IN2)		

Zur Schaltung 2

Elemente	Farbe	HC-SR04	Farbe	RasPi
HC-SR04		VCC		5V
HC-SR04		Trig		PIN 15 (GPIO23)
HC-SR04		Echo an 1kΩ Widerstand		PIN 18 (GPIO24)
HC-SR04		Echo an 1kΩ Widerstand mit GND verbunden		
HC-SR04		GND an 1kΩ Widerstand von Echo		GND

Zusammenbau

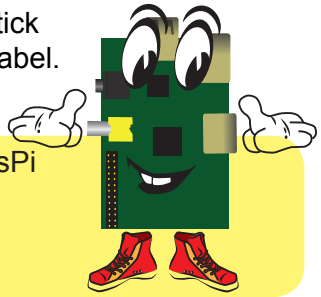
Beschreibung



Fertig! Du kannst deiner Kreativität freien Lauf lassen. Der GPIO ROBOT kann später noch weiter ausbaut werden z.B. mit weiteren Sensoren und elektrische Komponenten. Bevor du das machst, teste erstmal ob die Motoren richtig funktionieren und der Sensor Entfernungswerte übermitelt.

Motoren und Sensor testen

Schließe zuerst den RasPi an den Router mit einem LAN-Kabel und verbinde dich mit ihm über Remotedesktopverbindung vom Windows-Rechner. Konfiguriere dein WLAN-Stick (siehe Modul 1 WLAN Konfiguration), speichere es ab und entferne das LAN-Kabel.



Solltest du jetzt nicht mehr mit dem RasPi verbinden können, starte den RasPi neu. Ausprobieren kannst du es über Putty. Die Domain war **„raspi.local“**

Motoren test:

Auf dem Pi befinden sich zwei Python-Testdateien.
Öffne Putty , verbinde dich mit dem RasPi und wechsle in das Verzeichnis, wo sich die Dateien befinden:

```
cd GPIO/GPIO_Robotik
```

Jetzt wollen wir die Motoren überprüfen. Stelle am besten den GPIO ROBOT auf den Boden, nicht das er beim Testen runter fällt. Führe motor.py aus:

```
sudo python motor.py
```

Mit den Tasten [W, S, A, D und X] + RETURN kannst den Roboter steuern. Probiere es aus!

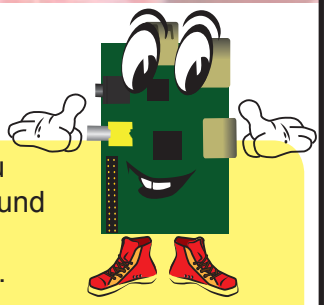
Entfernungssensor test:

Um den Entfernungssensor zu testen, musst du folgende Datei ausführen:

```
sudo python ultrasonic.py
```

Der Sensor berechnet die Entfernung zum nächsten Gegenstand in seiner Richtung.

GPIO ROBOT steuern



Sind die Räder nicht Synchron in den nächsten beiden Beispielen, musst du ein wenig nachhelfen. Du musst mit nano die jeweilige Python Datei öffnen und die Pulsweitenmodulation in den Rädereinstellungen ändern.
Probier es so lange aus bis der GPIO ROBOT auch wirklich geradeaus bzw. rückwärts synchron rückwärts fährt.

Nicht vergessen! Du musst in der Raspi-Config die Kamera aktivieren.
Zur Erinnerung sudo `raspi-config` und dann `Enable Camera`.

GPIO ROBOT über Browser steuern

Jetzt geht es richtig los! Um den GPIO ROBOT über den Browser deines Rechners zu steuern, starte die webcar.py Datei. Aber bevor du das tust, schalte die Pi-Kamera noch ein, damit du siehst wohin du fährst:

```
sudo service mjpg-streamer start
```

Übrigens mit `stop` statt `start` kannst du den Stream wieder ausschalten.
Starte auf deinen Windowsrechner Firefox oder ein anderen Browser und gib folgendes ein:

```
http://raspi.local:8080
```

Du wirst nach einem Namen und Passwort gefragt:

```
Name: pi
Passwort: raspberry
```

Funktioniert die Pi-Kamera einwandfrei, geht's jetzt zur Steuerung über den Browser.

Schließe die Seite in Browser und führe in Putty `webcar.py` aus:

```
sudo python webcar.py
```

Jetzt kannst du im Browser noch einmal die Domain eingeben, nur mit einen anderen Port.

```
http://raspi.local:8000
```

Schließe die Seite in Browser und führe in Putty `webcar.py` aus:

```
Name: webiopi
Passwort: raspberry
```

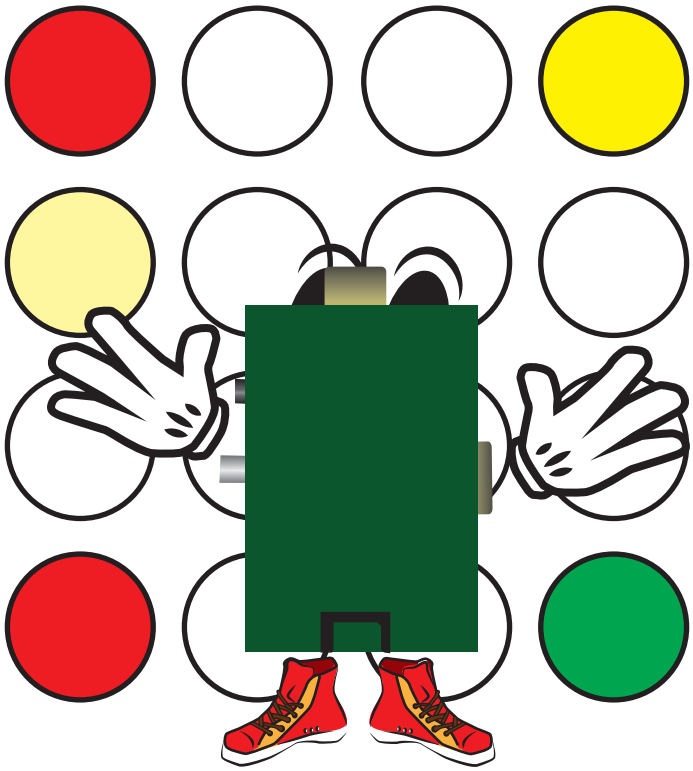
Viel Spaß beim Fahren!

Der Automodus

Hast du keine Lust selbst den GPIO ROBOT zu steuern? Willst du vielleicht, dass er allein in der Gegend rumfährt. Schau dir über den MJPG-Streamer wohin ihn der Weg verschlägt.
Dazu öffne im Browser zuerst den Stream und starte über Putty den Automodus:

```
sudo python automode.py
```

Reaktionsspiel für die Finger



Modul 6

Trellis anschliessen

Das Spiel

Neue Figuren erzeugen

Finger Tipp Reaktion

Inhalt

STEPS

- 1. Trellis an RasPi anschliessen
- 2. Das Spiel
- 3. Neue Figuren hinzufügen

Lern Ergebnisse

- Hardwareprogrammierung ohne direkte GPIO ansteuerung
- Python: Arrays erstellen
- Python: Random

Software

NOOBS mit GPIO-Raspbian

Raspbian:

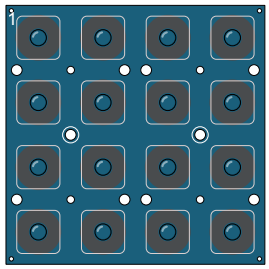
Nano (Texteditor)
Python
Adafruit_I2C.py

Beschreibung

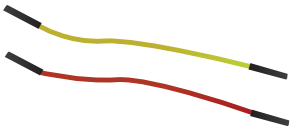
Finger Tipp Reaktion ist ein kleines Reaktionsspiel. In diesem Spiel, werden per Zufall (Random) verschiedene Muster aus einer Anzahl von Mustern (Array) geladen, die du per Knopfdruck auf dem Pad ausschalten musst.

Hardware

Trellis

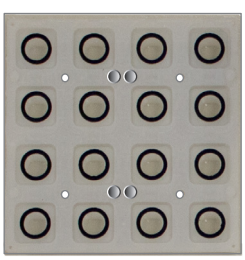


Steckbrücken

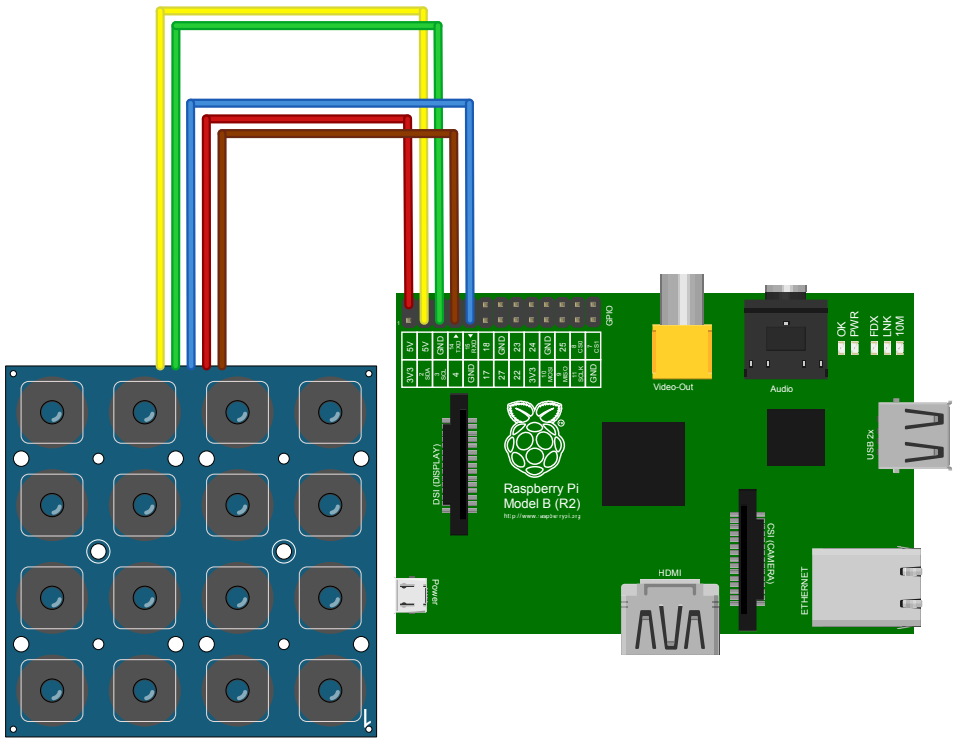


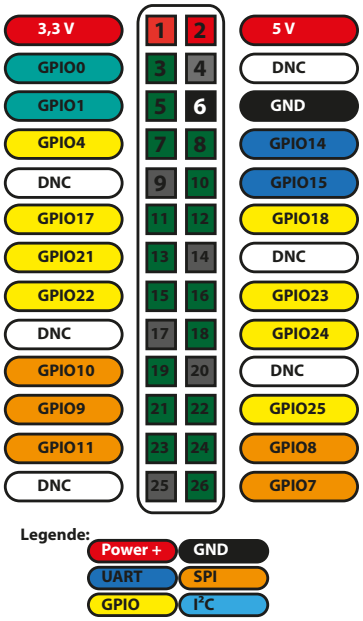
Female to Female

Widerstand



Zusammenbau





Farbe	Trellis	Farbe	RasPi
	SDA		PIN 3 (2 SDA)
	SCL		PIN 5 (3 SCL)
	GND		PIN 9 (GND)
	5V		PIN 2 (5V)
	INT		PIN 7 (GPIO4)

Python Code:Finger_Tipp.py

```
#!/usr/bin/python
import Adafruit_Trellis
import numpy as np
import time
import array
import random

#Damit die Python-Programme von Trellis funktionieren muss sich immer
#die Datei „Adafruit_I2C.py" im selben Ordner wie der Pythom Code befinden

#Man kann mehrere Trellis aneinander anschliessen, hier werden die Konfiguration
#vorgenommmen matrix0 = ein Trellis
matrix0 = Adafruit_Trellis.Adafruit_Trellis()
trellis = Adafruit_Trellis.Adafruit_TrellisSet(matrix0)
I2C_BUS = 1

#Der INT Input wird konfiguriert: An welcher Schnittstelle des I2C_Bus das Trellis
#ausgeführt wird
trellis.begin((0x70, I2C_BUS))

#Arrays die die Bilder zum Ausschalten bilden:
a = np.array([(8,12,13,14), (4,5,6,7),(1,2,4,5,6,7), (1,2,4,5,6,7,9,10),
(1,2,4,7,8,11,13,14), (0,5,10,15), (0,3,5,6,9,10,12,15), (5,6,9,10), (0,3,12,15),
(0,3,4,7,8,11,12,15), (0,3,4,7,8,9,10,11,12,15), (1,2,8,11,13,14), (3,4,6,9)])

liLedList = [] #Aktive LED-Liste definieren

#Diese Funktion lädt den ausgesuchten Array in Zwischenspeicher
def getLEDList(a,t):
      liTemp = []                    #Der Zwischenspeicher wird erstellt
      for i in range(len(a[t])):     #Der Array wird ausgewählt
            liTemp.append(a[t][i])   #und in die liTempgespeichert

      return liTemp                  #Anschließend an getLEDList(a,t) übergeben
```

Python Code:Finger_Tipp.py

```
#Per Zufall wird ein Bild im Array ausgelesen und ausgegeben
while True:
      trellis.clear()                 #Die LEDs werden alle ausgeschaltet (resetet)
      t = random.randint (0,12)       #Zufallszahl zwischen 0 und 12 für Bild
      liLedList = getLEDList(a,t)     #Das ausgewählte Array wird in liLEDList
                                      #gespeichert


      print a[t]                      #Druckt das Array auf dem Bildschirm aus
#Die LEDs die sich im ausgewählten Array befinden gehen an:
      for i in range(len(a[t])):
            c = a[t][i]
            trellis.setLED(c)         #Setzt die LEDs aus c am Trellis auf HIGH
            trellis.writeDisplay()    #LEDs am Trellis gehen an
            time.sleep(0.01)
      time.sleep(1)
      while len(liLedList)>0:       #Überprüft das List so lang bis keine LED an ist
            time.sleep(0.01)

            if trellis.readSwitches():    #List den Satus des LED-Knopfs aus
                  for i, c in enumerate(liLedList):
                        if trellis.justReleased(c):   #Wenn eine LED ausgeschaltet
                              print ,^{0}`.format(c)  #wird, wird in c eine 0
                              trellis.clrLED(c)       #geschrieben, wird ausge
                              liLedList.remove(c)     #schaltet und aus der
                              print liLedList         #liLedList entfernt

                  trellis.writeDisplay()
      print „Geloest: Neues Bild"
```

Python Code:Finger_Tipp.py

Neue Figuren

In das `np. array` kannst du weitere Figuren hinzufügen oder alte löschen. Die in Klammern gespeicherten Zahlen 1. `(8,12,13,14)` ist eine Liste für die LEDs die am Trellis angehen soll. Nehm dir am besten ein Blatt Papier um neue Figuren zu erstellen. Male ein 2. Quadrat mit 4x4 kleine Quadrate im inneren. Eins worauf du achten musst, das Array für Trellis fängt nicht ab 1 an sondern bei 0. D.h. du fängst nicht mit 1 zu zählen sondern mit 0. Erstelle neue Arrays 3. und Trage diese in `a = np.array([(...),....])` ein.

0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

1.

0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

2.

0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

3.

Damit die neuen Figuren aus dem Array gelesen werden können, musst du ale Figuren in `np.array` zählen. Anschließend trägst du die neue Zahl in `t = random.randint (0,12)` ein.

Hast du z.B. 20 Figuren, musst du die Zahl um – 1 reduzieren, also statt 20 trägst dort 19 ein.
Du weißt ja die 0 ist in diesem Fall deine neue 1.