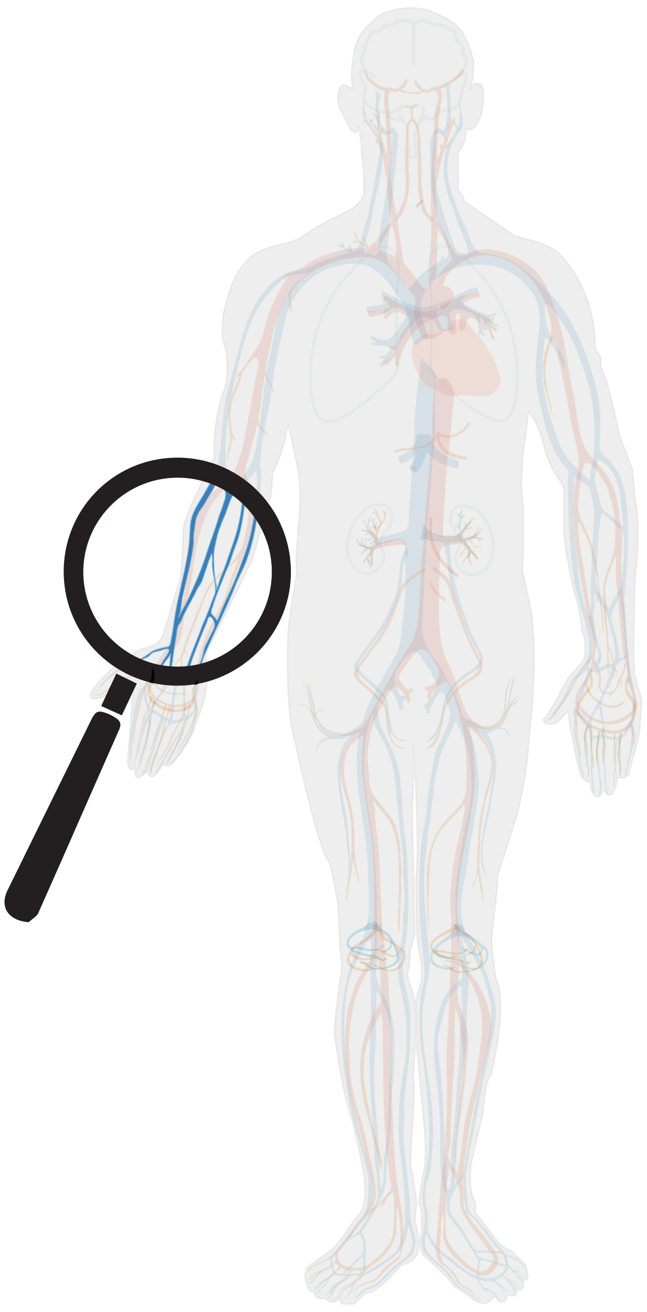
Venenfinder – a DIY medical assistive device.

This tutorial describes how to build the „Venenfinder“ (vein-finder) from scratch. It is a medical assistive device that was developed by us to make it easier for medical stuff as well as people with chronic illnesses to get vein access.



The veins are illuminated with IR light (950nm) and the back scattering is captured by the Raspberry Camera (the one without the IR-filter). You can use old analogue film tape as a filter to block visible light and let only pass IR-light. The camera picture is processed in several stages to get an improved distribution of light and dark parts of the image (histogram equalization). The reason to use near IR illumination lies in the optical properties of human skin and in the absorbance spectrum of haemoglobin.

The device was developed by us (code, illumination, 3d-files as well as numerous tests of prototypes and real-world tests in a hospital). In this tutorial we reference to the following blogs who helped us developing this mobile version of the "Venenfinder":

Pyimagesearch:
Here you can find tons of information and well describes tutorials how to start with computer vision and the Raspberry Pi from scratch. We cite some steps from Adrian's blog on how to install openCV on the Raspberry Pi from scratch.
https://www.pyimagesearch.com

Pi as a Access point

If you want to stream the calculated image to your smartphone, TV or tablet, you either need to integrate the Raspberry into your local Wifi network – or just start a new one. We don't want the user to have to deal with editing wifi settings on a terminal session,

so we just decided to turn the Pi into a Hotspot. Here we followed Phil Martin's blog on how to use the Raspberry as a Wifi Accesspoint:
https://frillip.com/using-your-raspberry-pi-3-as-a-wifi-access-point-with-hostapd/

Rotary encoder:
Since you need a way to change the setting of the image enhancement, we decided to use rotary encoders. These are basically just 2 switches and they sequence they close and open tells you the direction the knob was turned.
We soldered 3 rotary encoders to a little board and created a Raspberry HAT on our own. For the code we used:
http://www.bobrathbone.com/raspberrypi/Raspberry%20Rotary%20Encoders.pdf

Streaming Server in Python:
We used some code from Igor Maculan – he programmed a Simple Python Motion Jpeg (mjpeg) Server using a webcam, and we changed it to Picam, added the encoder and display of parameters. Original Code:
https://gist.github.com/n3wtron/4624820

Install / Compile Softwarecomponents

1. download Jessie (full) from Raspberri.org (Version 2017-04-10)
2. unzip file and write the image to an SD-Card (e.g. using Apple Pi-Baker or other tools; see e.g. https://www.raspberrypi.org/documentation/installation/installing-images/)
3. start the system with the new sd-card and configure it according to... http://www.pyimagesearch.com/2016/04/18/install-guide-raspberry-pi-3-raspbian-jessie-opencv-3/
4. Make sure network connections work (WiFi or LAN) – you need to download software packets

Steps to install the Software, OS, Network

Follow the steps in Adrian's tutorial http://www.pyimagesearch.com/2015/10/26/how-to-install-opencv-3-on-raspbian-jessie/

1. expand filesystem (automatic)
2. updates: (sudo apt-get update & sudo apt-get upgrade)
3. install the packages as described, but download opencv3.2 and additions for 3.2

Install packages:

developer tools:
```
sudo apt-get install build-essential git cmake pkg-config
```

image I/O packages
```
sudo apt-get install libjpeg-dev libtiff5-dev libjasper-dev libpng12-dev
```

video I/O packages:
```
sudo apt-get install libavcodec-dev libavformat-dev libswscale-dev libv4l-
dev
sudo apt-get install libxvidcore-dev libx264-dev
```

display images, gui etc:
```
sudo apt-get install libgtk2.0-dev
```

extra optimation for matrix operations:
```
sudo apt-get install libatlas-base-dev gfortran
```

Python 2.7 und 3 header files:
```
sudo apt-get install python2.7-dev python3-dev
```

Source-Files:
```
wget -O opencv.zip https://github.com/Itseez/opencv/archive/3.2.0.zip
unzip opencv.zip

wget -O opencv_contrib.zip https://github.com/Itseez/opencv_contrib/archive
/3.2.0.zip
unzip opencv_contrib.zip
```

python package manager
```
wget https://bootstrap.pypa.io/get-pip.py
sudo python get-pip.py
```

install numpy (numerical python)
```
pip install numpy
```

compile opencv
```
cd ~/opencv-3.2.0/
mkdir build
cd build
cmake -D CMAKE_BUILD_TYPE=RELEASE \
    -D CMAKE_INSTALL_PREFIX=/usr/local \
    -D INSTALL_PYTHON_EXAMPLES=ON \
    -D OPENCV_EXTRA_MODULES_PATH=~/opencv_contrib-3.2.0/modules \
    -D BUILD_EXAMPLES=ON ..

make -j4
```
(This takes a good 45-60min).

Now openCV should be installed with all the necessary additions. Please test it by re-booting and starting Python 2:
```
sudo shutdown -r now
```

After reboot and login start Python interpreter either by choosing IDLE-Python 2 from program menu or by typing into terminal:
```
python

>>>import cv2
```
(should produce no errors)

Hardware, 3D-files etc.

For the hardware you simply need to print the stl-files according to your 3D-Printer's software. The case is designed to fit the Raspberry Pi3 with the three encoders attached, but you can adjust it to you needs.
The Encoders attach to GND and to the GPIOS 20,21 / 18, 23 and 24,25. The IR-LEDs used have a peak at 940 or 950 nm and require a 12 Ohm Resistor, it you connect three of the LEDs in series. Connect 3 series of the LEDs with a resistor parallel and you have an array of 3x3 LEDs, which will fit into the casing designed for the reflectors.

That one fits into the main casing:

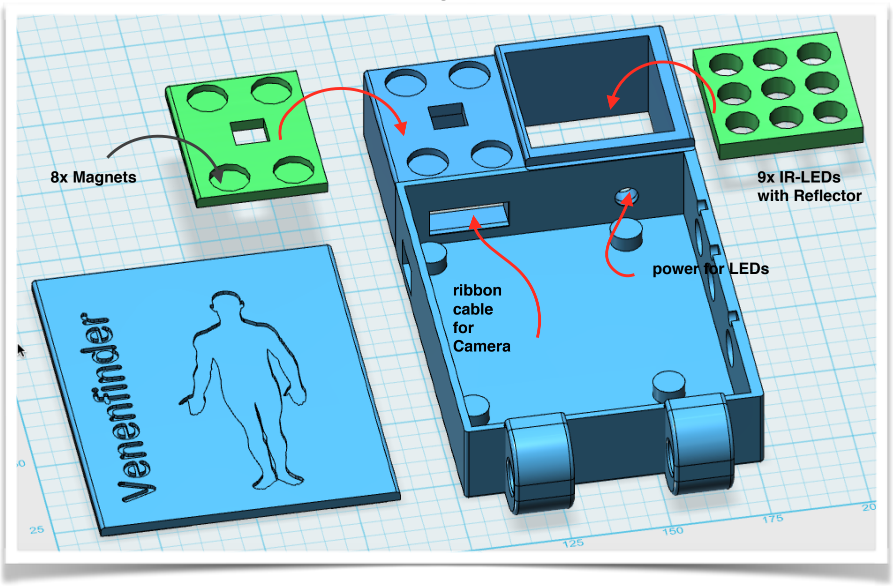
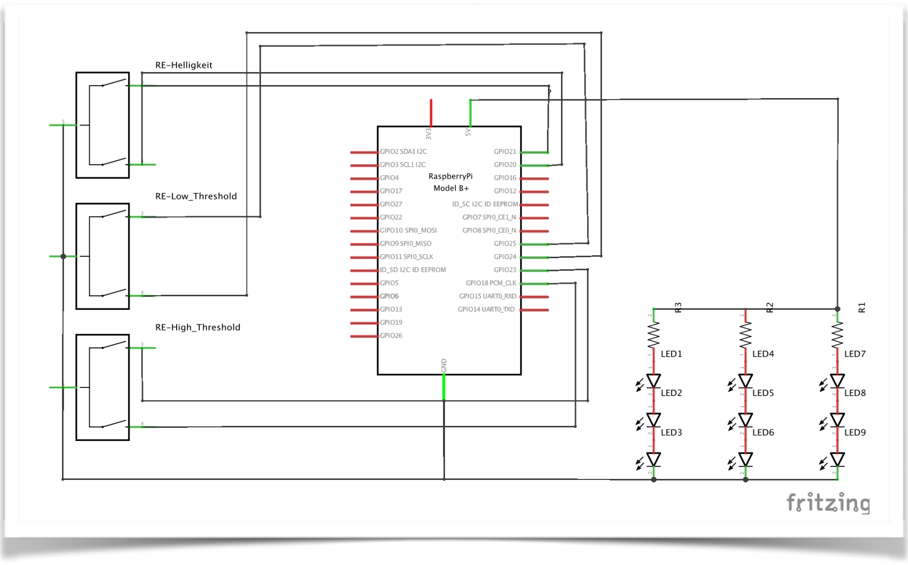


Image: 3D-printed parts and assembly

Print the cover and the main casing with support; the adapter for the filter (analogue film tape) and IR reflector matrix won't need support.

The circuit diagram for connecting LEDs and encoders:



For our mobile version of the Venenfinder we used a piece of strip board, but if you want to build your own circuit board please keep in mind you may need to change the case to fit yours in…

Putting it all together: Wifi Access-Point, Autostart etc.

You can find the latest version of Venenfinder on Myrijam's github page, including the code (Python script) and the 3d-files. All you need to do is to clone the git to your raspberry pi (or your regular computer to work with the 3d-files).
https://github.com/Myrijam/Venenfinder

Server-Installation (Python)

Python already includes a simple http server you can use. It is just a little line of code to import the corresponding libraries into your python code (don't run this in terminal):

```
from BaseHTTPServer import BaseHTTPRequestHandler,HTTPServer
from SocketServer import ThreadingMixIn
```

Wifi-Configuration (

These steps are mainly a short excerpt from Phil Martin's blog:

Install hostapd (which will give you the ability to run an Accesspoint with building Wifi) and dnsmasq (combined DNS and DHCP Server):

```
sudo apt-get install dnsmasq hostapd
```

Edit the WLAN0-interface (build-in Wifi) preferences by using an editor on the commandline:

```
sudo nano /etc/dhcpcd.conf
```

add this line at the bottom, before any instructions starting with "interface" (if there are any)

```
denyinterfaces wlan0
```

Accesspoint needs a fixed (static) IP, so that the QR-Codes can work on the back of the device. Open this file and edit the section starting with WLAN0:

```
sudo nano /etc/network/interfaces
```

It should look like:

```
allow-hotplug wlan0
iface wlan0 inet static
    address 192.168.42.1
    netmask 255.255.255.0
```

Now you can restart the dhcp-service and reload the changed configuration:

```
sudo service dhcpcd restart
sudo ifdown wlan0; sudo ifup wlan0
```

Now the hostapd needs to be configured:

Create/edit the file by this command:
```
sudo nano /etc/hostapd/hostapd.conf
```

This is the content of hostapd.conf

```
# This is the name of the WiFi interface we configured above
interface=wlan0
# Use the nl80211 driver with the brcmfmac driver
driver=nl80211
# This is the name of the network
ssid=Venenfinder
# Use the 2.4GHz band
hw_mode=g
# Use channel 6
channel=6
# Enable 802.11n
ieee80211n=1
# Enable WMM
wmm_enabled=1
# Enable 40MHz channels with 20ns guard interval
ht_capab=[HT40][SHORT-GI-20][DSSS_CCK-40]
# Accept all MAC addresses
macaddr_acl=0
# Use WPA authentication
auth_algs=1
# Require clients to know the network name
ignore_broadcast_ssid=0
# Use WPA2
wpa=2
# Use a pre-shared key
wpa_key_mgmt=WPA-PSK
# The network passphrase
wpa_passphrase=vene1181
# Use AES, instead of TKIP
rsn_pairwise=CCMP
```

Please pay attention to the statements in bold – these are SSID and PW of the device. It is not an elegant solution to hard-code any password in, but in the rush for the final we could not find an easier configuration –so please change them!

Now tell the hostapd where to look for the config file:
```
sudo nano /etc/default/hostapd
```

change the line from `#DAEMON_CONF=""` to `DAEMON_CONF="/etc/hostapd/hostapd.conf"`

Next step is to change the config of DNSMASQ. Phil Martin suggests to copy the pre-configured file and keep it that way before creating a new one:

```
sudo mv /etc/dnsmasq.conf /etc/dnsmasq.conf.orig
```

And then create a new config file:
```
sudo nano /etc/dnsmasq.conf
```

Paste the following content:
```
interface=wlan0              # Use interface wlan0
listen-address=192.168.42.1  # Address to listen on
bind-interfaces              # Bind to the interface
server=8.8.8.8               # Forward DNS requests to Google DNS
```

```
domain-needed                    # Don't forward short names
bogus-priv                       # Never forward non-routed address spaces.
dhcp-range=192.168.42.10,192.168.42.150,12h # Clients IP range and time
```

Now it is time to set the rules for managing traffic on the interfaces:

```
sudo nano /etc/sysctl.conf
```

Search fort he line containing `net.ipv4.ip_forward=1` and remove the `#` from the beginning of the line. Enable this rule by:

```
sudo sh -c "echo 1 > /proc/sys/net/ipv4/ip_forward"
```

The following commands define the traffic routing between Wifi and Ethernet; it is probably not necessary for this special use case:

```
sudo iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
sudo iptables -A FORWARD -i eth0 -o wlan0 -m state --state RELAT-
ED,ESTABLISHED -j ACCEPT
sudo iptables -A FORWARD -i wlan0 -o eth0 -j ACCEPT
```

These rules need to run every time at bootup, so this saves the rules for later use:

```
sudo sh -c "iptables-save > /etc/iptables.ipv4.nat"
```

Now run the following command to enable restoring these rules at bootup:

```
sudo nano /etc/rc.local
```

Just before the line `exit 0` enter the following command:

```
iptables-restore < /etc/iptables.ipv4.nat
```

Now you just need to restart the corresponding services:

```
sudo service hostapd start
sudo service dnsmasq start
```

Now the Raspberry 3 should work as an access point, providing a network with the specified SSID (“Venenfinder”) and the PSK (“vene1181”).
To make sure the device runs the Venenfinder-Python code at reboot, please refer to the following section.

Autostart

Use the following command to add autostart to the Python Code from Venenfinder :

```
sudo nano /etc/profile
```

And put a last line to the script:

```
python /home/pi/Venenfinder/Venenfinder.py
```

Rebooting the Raspberry will start the Venenfinder now :-)

Connecting…

You can scan these QR-Codes to connect to Venenfinder, given you used the combination of SSID and PSK provided in this tutorial:

QR-Code for Wifi-Network

SSID: Venenfinder PSK: vene1181

QR-Code for Live-Streaming (Android, Computer):

http://192.168.42.1:8080/cam.mjpg