Raspberry

things I tried

L.J.M. Dullaart

1. Headless set-up

1.1 Intro

Every time I deploy a new pi, something has changed. This makes it difficult to create a simple set-up that works every time. It also means that all instructions you find on the internet are outdated. Google is great in finding set-up instructions from back in 2012, but those do not work anymore. And most instructions do not have a date in them, so you're completely lost why it doesn't work.

This instruction made for people that have Linux running.

Version	Date	Raspian	Comment
1	28-6-2017	Jessie	
2	29-12-2017	2017-11-29-raspbian-stretch	
3	29-12-2017	2017-11-29-raspbian-stretch	
4	19-1-2019	2018-11-13-raspbian-stretch.img	

1.2 Burning the image

First get the latest Raspian:

```
wget http://downloads.raspberrypi.org/raspbian_latest
```

What you'll get is a zip-file with the latest raspian-image. Unzip and burn on the SD-card.

Many tutorials go into great length on how to identify your SD-card. In most cases, it is `/dev/mmcblk0` or one of the `/dev/sd*` devices.

```
mv raspbian_latest raspbian_latest.zip
unzip raspbian_latest.zip
sudo if=2018-11-13-raspbian-stretch.img  of=/dev//dev/mmcblk0 status=progress
```

Ofcourse, this takes a long time; that is why the `status=progress` is on the command line. Total is about 3.5G.

Remove the card and plug it back in. Normally, it will be mounted automatically, and you will see:

```
/dev/mmcblk0p1 on /run/media/ljm/boot type vfat
/dev/mmcblk0p2 on /run/media/ljm/5c01c1ce-fe60-428a-8e68-0be0e8ed6b7a type ext4
```

Otherwise, mount by hand.

For raspian-stretch, the root filesystem will be called `rootfs` instead of the big number.

1.3 The networking

Because from Jessie on, it is now using systemd, everything you knew about the configuration of networking is now of no value. In previous releases, networking was done via `/etc/network/interfaces` but now, `dhcpcd` is used. It also means that all tutorials and howto's are now obsolete.

The main configuration file for `dhcpcd` is `/etc/dhcpcd.conf.` For every connection that you want to have a fixed IP address add a block, of course with your own IP addresses:

```
interface eth0

static ip_address=192.168.178.53/24
static routers=192.168.178.1
static domain_name_servers=192.168.178.6

interface wlan0

static ip_address=192.168.178.3/24
static routers=192.168.178.1
static domain_name_servers=192.168.178.6
```

For some dark and unknown reason, you still need to edit `/etc/network/interfaces` to add

```
allow-hotplug eth0
```

Next, setup the wpa-supplicant in `etc/wpa_supplicant/wpa_supplicant.conf` :

```
country=GB
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1

network={
    ssid="ssidforwifi"
    psk="wifipsk"
}
```

And that should be enough for the network.

1.4 Enable ssh

Enabling ssh requires an `ssh` file in the `boot` directory. Normally, you see a directory

```
/dev/mmcblk0p1     /run/media/username/boot
```

if you query all mounts. So do a

```
touch /run/media/username/boot/ssh
```

and ssh will start at boot-time.

But you don’t want to type passwords, so we’ll distibute the keys:

```
cd $piroot/root
mkdir .ssh
chown root.root .ssh
chmod 700 .ssh
cp  ~/.ssh/id_rsa.pub  .ssh/authorized_keys
chmod 600 .ssh/authorized_keys
```

1.5 Connecting and manual actions.

If you do it in this way, everything should run and the pi should be accessible under your WiFi IP address.

Try a `ssh root@192.168.178.3` (use your own IP address) and voila.

There are some manual actions to take before everything works. First, make your users that need to be present on the system. In my case, that is "ljm":

```
adduser ljm
mkdir /home/ljm
cp -r /root/.ssh ~ljm
chown -R ljm.ljm ~ljm/.ssh
```

Next item on the list: raspi-config. Use the menus to set the hostname. But more importantly, under **7 Advanced Options** you will find **A1 Expand Filesystem** which will allow you to use the complete sd card.

Do not reboot after this!

Make `vi` our deault editor:

```
update-alternatives --set editor /usr/bin/vim.tiny
```

You will also need to add the users in the sudoers-file:

```
ljm   ALL=(ALL)       NOPASSWD: ALL
```

And now: reboot

1.6 Security

With this set-up you can add the pi to your local network. Not to the Internet. There are a lot of security implications that we have not considered. One of the most important is that the user `pi` is still present and having his default password. Also the `NOPASSWD` in the `sudoers` is practical, but a bad idea security-wise.

The goal of this part was to get the pi working; not to make it secure.

2. 433Mhz

2.1 Introduction

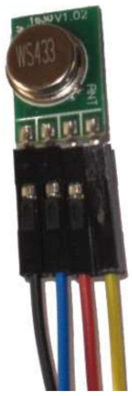
In the Netherlands, and perhaps beyond out borders too, there is a simple system for remote switching of lights, called Klik aan Klik uit (KAKU). Although not the cheapest option available, it is quite affordable, especially if you use the APA3-1500R Starterset.

Although a complete Internet solution is available using "The Cloud", I decided that one of my Pi's would be just as good.

There are a number of steps to be taken to get a working solution. I have tied to keep it as simple as possible. This means that I have not always chosen the cheapest solution.

2.2 Step 1: Transmitter hardware

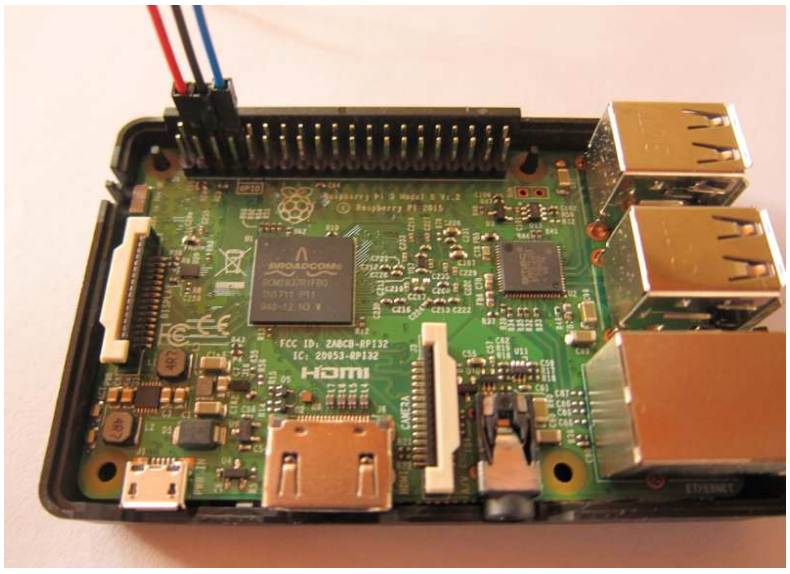
The KAKU works at 433MHz. That means we need a 433MHz transmitter. These things are cheap and it does not really matter which one you choose. I took this one. I use female-female jumper wires to connect the transmitter to the Pi.



Pin	Color	Function	Comment
1	black	Ground	
2	blue	Data	
3	red	Vcc	5V; maximum is 12V
4	yellow	Antenna	marked ANT; should be 16.7 cm

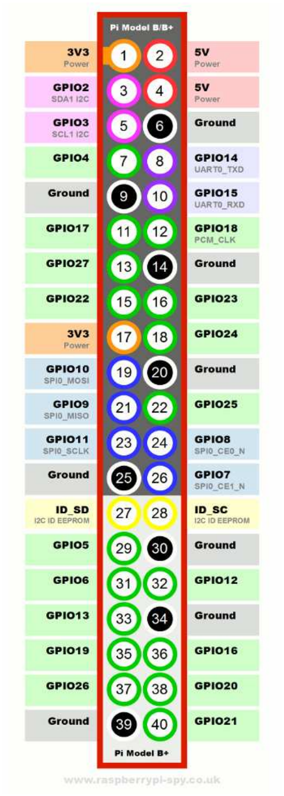
These are connected to the Pi as follows:

IMG_4134.JPG



In practice, this means that GPIO14 is used:

Layer


Pi Model B/B+
3V3 Power 1 2 5V Power
GPIO2 SDA1 I2C 3 4 5V Power
GPIO3 SCL1 I2C 5 6 Ground
GPIO4 7 8 GPIO14 UART0_TXD
Ground 9 10 GPIO15 UART0_RXD
GPIO17 11 12 GPIO18 PCM_CLK
GPIO27 13 14 Ground
GPIO22 15 16 GPIO23
3V3 Power 17 18 GPIO24
GPIO10 SPI0_MOSI 19 20 Ground
GPIO9 SPI0_MISO 21 22 GPIO25
GPIO11 SPI0_SCLK 23 24 GPIO8 SPI0_CE0_N
Ground 25 26 GPIO7 SPI0_CE1_N
ID_SD I2C ID EEPROM 27 28 ID_SC I2C ID EEPROM
GPIO5 29 30 Ground
GPIO6 31 32 GPIO12
GPIO13 33 34 Ground
GPIO19 35 36 GPIO16
GPIO26 37 38 GPIO20
Ground 39 40 GPIO21
Pi Model B+
www.raspberrypi-spy.co.uk

2.3 Step 2: Transmitter software

To test the set-up, we need some software to send control signals. In general, all source software is under `~/src` in my set-up. You might want to follow that. So, `mkdir ~/src` if that directory does not exist. Also, you need GIT, so `sudo apt-get install git` if you haven't done that earlier.

The software set-up comes in two parts:

— the libraries to send the signals

— a front-end to do the switching

First, install the libraries:

```
cd ~/src
git clone git://git.drogon.net/wiringPi
cd wiringPi
./build
```

The will also install the libraries.

Next, a simple CLI front-end:

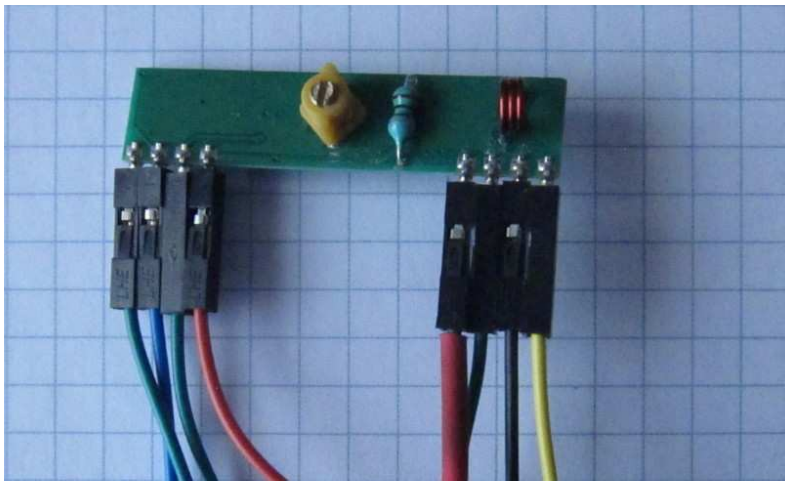
```
cd ~/src
git clone https://github.com/chaanstra/raspKaku
cd raspKaku
g++ -o kaku kaku.cpp -I/usr/local/include -L/usr/local/lib -lwiringPi
```

Get a simple wall-plug from KAKU, plug it in ans do `sudo ./kaku 1 A on` within two seconds. And Lo and Behold: It switches! You can switch it off again with `sudo ./kaku 1 A off`

If you are experimenting with the kaku command you will notice that it folds address space. This is because the original Klik Aan Klik Uit protocol was rather limited in number of devices. In the directory, there is also a `newkaku.cpp` which implements the new KAKU protocol. Apparentlythe new protocol allows 26 bits addresses, which is sufficient for most (anything up to 67108863).

2.4 Receiver

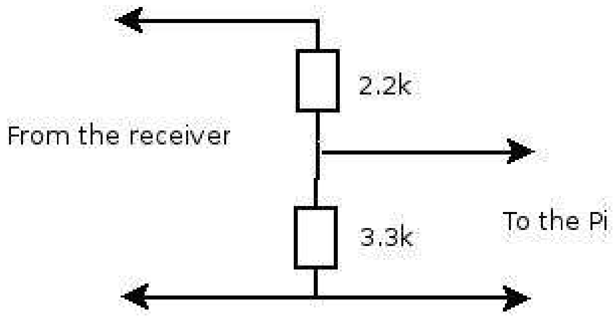
Next step is a receiver. I used this one: https://iprototype.nl/products/components/communications/rf-receiver-434mhz



Before you start connecting, some soldering must be done. The GPIO pins on the Pi will accept 3.3V, but not the 5V that comes out of the receiver. Depending on the source you use, the GPIO-pins will use 0 and 1 when:

logical value	minimum	maximum
0	0V	0.54V
1	2.3V	3.3V

A voltage divider with resistors will do the trick:



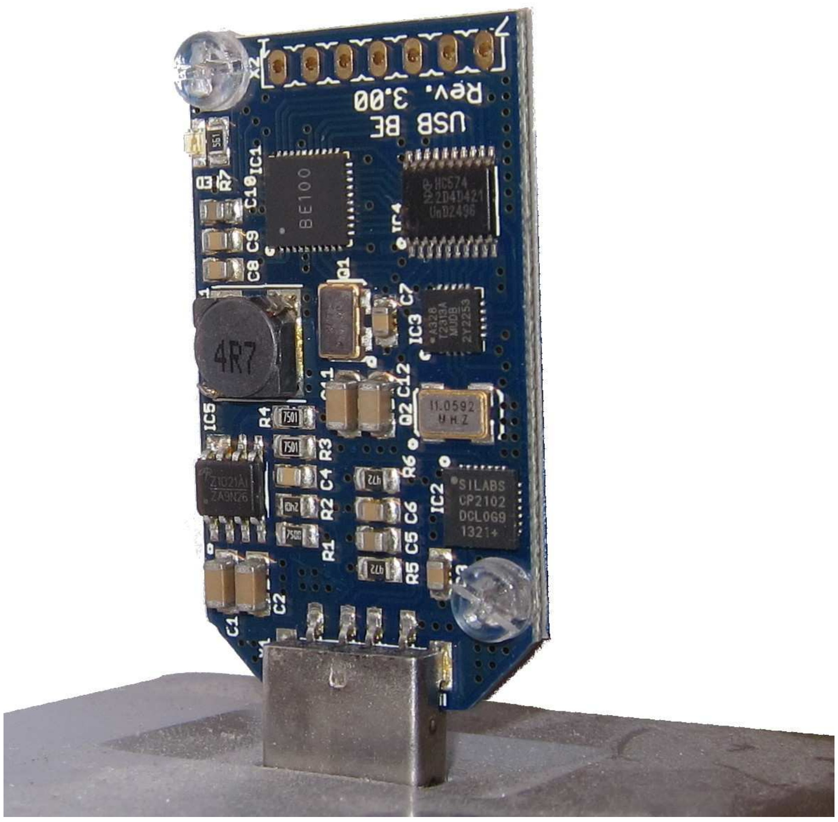
The output of the voltage divider is connected to gpio15.

3. Bitcoin mining with a pi

3.1 Intro

If you don’t know what bitcoins are, or if you expect to get a miner that gets you rich quickly, skip this.

Long time ago, I had a small USB stick for bitcoin mining. It gave me a whooping 300Mh/s, which was about ten times what I got from mining with my display adapter. Due to various reasons, I shelfed the stick.



Now, I recently found it back, and although it is not really a profit-mining, it takes very little power, especially if connected to a raspberry pi. So, mostly for nostalgic reasons, I have re-activated my USB-miner.

The pool that I used in the past is now defunct, so I decided to use slushpool basically because it is as good as any (at least at this stage).

3.2 Setting up the miner

3.2.1 Make an account

You have to make an account, which will verify your e-mail

3.2.2 Install the software

First, install the dependencies:

```
apt-get update
apt-get install autoconf autogen libtool uthash-dev libjansson-dev libcurl4-openssl-dev
libusb-dev libncurses-dev git-core
apt-get install libevent-dev
```

Then, get a git-clone of the `bfgminer`

```
git clone https://github.com/luke-jr/bfgminer.git
cd bfgminer
./autogen.sh
./configure
make
```

And start mining:

```
./cgminer  --userpass  userID.workerID:any-password  --url  stratum+tcp://stratum.slush-
pool.com:3333
```

And see the satoshis roling in:

```
bfgminer  version  5.4.2-7-g9f781b8  -  Started:  [2017-11-29  20:32:52]  -  [   0  days
01:19:06]
[M]anage  devices  [P]ool  management  [S]ettings  [D]isplay  options              [H]elp
[Q]uit
Pool 0: ...m.slushpool.com  Diff:128  +Strtm  LU:[21:51:36]  User:myuserid.worker1
Block #496752: ...db12af37  Diff:1.35T ( 9.64E)  Started: [21:49:36]  I: 1.80nBTC/hr
ST:3  F:1  NB:7  AS:0  BW:[ 47/  3 B/s]  E:2.15  BS:617
1            | 332.8/333.1/221.6Mh/s | A:2 R:0+1( 33%) HW:0/none
---------------------------------------------------------------------------------------------
BES 0:       | 335.9/332.9/221.4Mh/s | A:2 R:0+1( 33%) HW:0/none
---------------------------------------------------------------------------------------------
[2017-11-29 21:46:35] Stratum from pool 0 detected new block
[2017-11-29 21:49:36] Stratum from pool 0 detected new block
```

Some calculations:

1.8	nBTC/hr
43.2	nBTC/day
302.4	nBTC/week
1296	nBTC/month
0.153	mBTC/year
65	years before payout

hmmm.... Lets get another pool...

4. Change to ssl

4.1 Intro

At some point in time, you will want to upgrade your webserver to ssl. This is how I did it.

4.2 Let’s encrypt

The certificate-part is taken from https://www.pestmeester.nl/index.groff#10.0 and adapted to my situation.

Certbot or Let’s Encrypt is not natively compiled in Raspbian. So we will have to install it manually. First, install GIT:

```
sudo apt-get install git
```

Next get a clone of Let’s Encrypt.

```
sudo git clone https://github.com/certbot/certbot /etc/letsencrypt
```

Now we’re going to get those certificates. I run a single domain on my server, ljm.name The first time you apply for a certificate, you’ll get an account. The next time you apply for new certificates, they will just be added to the same account.

For my domain I got 1 certificate: ljm.name

```
sudo  /etc/letsencrypt/certbot-auto  certonly  --agree-tos  --webroot  -w  /links/www  -d
ljm.name
```

Follow the instructions, especially the first time to create the account (by filling out email, password, agree with TOS, etc.).

After succesful validation and installation you should see the message:

```
Congratulations! Your certificate and chain have been
saved at /etc/letsencrypt/live/mysite.com/fullchain.pem.
Your cert will expire on 2017-05-09.
To obtain a new or tweaked version of this certificate in
the future, simply run certbot-auto again. To
non-interactively renew *all* of your certificates,
run "certbot-auto renew"
```

Create a cronjob to automate certificate renewal

```
sudo crontab -e
0 6 * * * /etc/letsencrypt/certbot-auto renew --text >> /etc/letsencrypt/certbot/cert-
bot-cron.log
```

This cron job runs daily, but the certificate is only renewed if it is less than 30 days until expiry.

4.3 Apache

Apache has always been a PITA to configure, and enabling SSL is no exception to that. First, under `/etc/apache2/sites-available` create a copy of `default-ssl` and name it `ssl`.

In `ssl` I set the following:

```
<IfModule mod_ssl.c>
<VirtualHost _default_:443>
        ServerAdmin webmaster@localhost

        DocumentRoot /room/sda1/www/groff
        <Directory />
                Options FollowSymLinks
                AllowOverride None
        </Directory>
        <Directory /var/www/>
                Options Indexes FollowSymLinks MultiViews
                AllowOverride None
                Order allow,deny
                allow from all
        </Directory>
        ErrorLog ${APACHE_LOG_DIR}/error.log

        LogLevel warn

        CustomLog ${APACHE_LOG_DIR}/ssl_access.log combined
        SSLEngine on

        SSLCertificateFile    /etc/letsencrypt/live/ljm.name/cert.pem
        SSLCertificateKeyFile /etc/letsencrypt/live/ljm.name/privkey.pem
        SSLCertificateChainFile /etc/letsencrypt/live/ljm.name/fullchain.pem
        BrowserMatch "MSIE [17-9]" ssl-unclean-shutdown

</VirtualHost>
</IfModule>
```

Make a link in `/etc/apache2/sites-enabled` reload apache and ...

Nothing.

`netstat -an` does not give port 443 as LISTEN. No logrecords to help, starting in debug mode doesn't give additonal information, just nothing. So,...

First thing is to enable the apache ssl-module. The fact that this module is a standard part of the distribution, and that Apache recomends using SSL doesn't mean that the ssl module is enabled by default. Sigh..

```
a2enmod ssl
```

reload apache and ...

Nothing.

Remove `000-default` Areload apache and suddenly it starts working. However, now my http site is gone. `ls -s ../sites-available/default http` solves that problem. Apparently, `000-default` prevents other sites from being enabled.

5. A mailserver

Setting up a mail server has two big problems:

- It is difficult to maintain; especially the security patches are a PITA
- Mail server set-up is complicated. Very complicated.

There are two solutions for that:

- With `fetchmail` you do not need to expose your mail server to the Internet
- With `citadel` setting up the mail server is easy

So, Citadel it is.

5.1 Install Citadel on a Pi

5.1.1 Considerations

You will be wanting to keep quite a lot of mail on-line. The SD card is not the right place to do that. And, although you could use a USB stick, I would recommend spinning rust. You will need to put `/var/lib/citadel` on that disk. You could choose to put `/var` completely on the disk, which will further relieve your SD card from write actions.

I would not state that mail is the most important thing in life, but I would be pretty miffed if I lost my mail. So a backup is required for me.

Installing Citadel is an interactive process. You will be prompted with Package Configuration dialogs, that may change from version to version. I would have loved to install Citadel with Ansible, but I failed to do that consistently.

5.1.2 The installation

The default installation method for Citadel on a Pi is:

```
sudo -s
apt-get update
apt-get upgrade
apt-get install citadel-suite
```

You will see a number of dialog screens:

- Please specify the IP address which the server should be listening to. 0.0.0.0 is OK, because we’re not exposing it to the Internet.
- Authentication method to use: I use Internal, because I do not serve an LDAP or AD
- Citadel administrator username: admin is fine; choose your own password.
- Use internal for webcit, unless you plan to integrate it with Apache
- HTTP port 80, HTTPS 442
- User defined language

which is basically all the defaults. (If that was consistent I could install it via Ansible...)

5.1.3 Configuration

Adding accounts is reasonably well documented, so just read the fine material that Citadel provides.

5.2 Fetchmail

To get the mail in, I use fetchmail. This allows me to have different mail providers and get it all in one mailbox at home.

Installing is, as you'd expect:

```
sudo apt-get install fetchmail
```

Next, make for every user a in their home directory. The file should look like this:

```
poll pop.provider1.nl with proto POP3
    user "username"     , with password "secret"    , is my_name here warnings 3600
    user "second_user"  , with password "hemlighet" , is my_name here warnings 3600
       user "third_user"   , with password "tajomstvo" , is my_name here warnings 3600
poll pop.provider2.nl with proto POP3
       user "mailbox"      , with password "geheim"    , is my_name here warnings 3600
```

And you might put in the `crontab` for the user:

```
0,15,30,45 *  *   *   *    /usr/bin/fetchmail -v > /tmp/user.last_mail_fetch 2>/tmp/us-
er.last_mail_error
```

to get mail every 15 minutes.

5.3 A backup server

Making a backup-server is basically the same as the primary server. The main difference is that you do not enable fetchmail yet.

Having a backup server is good, but you must transfer your mail to that server. Once again, it is time for some condiderations.

- If your server crashes, how much mail would be acceptable to loose?
- What kind of problems will you protect yourself against?

For me, the acceptable loss will be from the backup that night to the crash and the main problem is a disk crash on the primary server or a complete loss of that server. In case of file corruption that is propagated, a longer loss of mail would be acceptable.

This leads to the following strategy:

- At night, when I'm asleep, bring down both Citadels (primary and backup)
- Make a copy of `/var/lib/citadel` and `/etc/citadel` to a backup server
- do an `scp` of those directories from the primary to the backup server
- bring the Citadels back on-line.

Due to the speed of the Pis, size of the mails and the network performance in my home, this backup takes about an hour.

A problem where I ran into was that the backup server was of a newer version than the primary server. That meant that I had both a new and a legacy configuration on the backup server. Citadel did not know whether to use the legacy or the new and refused to start to avoid corruption. I removed the new configuration, did a new backup and everything worked.

As with every backup, you need to check from time to time that it went well. Because Citadel has a web-server, you do not need a mail client to do that check.

CONTENTS