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 It also means that almost all instructions you find on the internet are outdated.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 is made for people that have Linux running. It is updated after each time I (re-)install a Pi.

Version	Date	Raspian	Comment
1	2017-06-21	Jessie	
2	2017-12-29	2017-11-29-raspbian-stretch	
3	2017-12-29	2017-11-29-raspbian-stretch	
4	2019-01-19	2018-11-13-raspbian-stretch.img	
5	2020-03-18	2020-02-13-raspbian-buster	
6	2021-02-23	2020-02-13-raspbian-buster.img	installed psi
7	2021-08-12	2020-02-13-raspbian-buster.img	installed blueberry

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 dd if=2020-02-13-raspbian-buster.img  of=/dev/mmcblk0 status=progress
```

Of course, this takes a long time; that is why the `status=progress` is on the command line. Total is about 3.6G.

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 and buster, the root file system 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 sometimes 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"
}
```

You will need to copy the wpa_supplicant.conf file for your particular wireless network in the boot folder, and when the Pi first boots, it will copy that file into the correct location in the Linux root file system and use those settings to start up wireless networking. After the Pi is connected to power, make sure to wait a few (up to 5) minutes for it to boot up and register on the network. The Pi's IP address will not be visible immediately after power on, so this step is crucial to connect to it headlessly. Depending on the OS and editor you are creating this on, the file could have incorrect newlines or the wrong file extension so make sure you use an editor that accounts for this.

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 distribute 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 host name. But more importantly, under **7 Advanced Options** you will find **A1 Expand File system** which will allow you to use the complete sd card.

Under Buster, you will need to set under

```
    4 Localisation Options
```

the Wifi country

```
I4 Change Wi-fi Country
```

Do not reboot after this!

Make `vi` our default editor:

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

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

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

If you want to manage your pi via Ansible, you may want to

```
sudo apt-get install -y aptitude
```

And to be up-to-date, do:

```
apt-get update
apt-get upgrade
```

This may take a *very* long time. You will may see a lot of lines

```
Removing 'diversion of /boot/bootcode.bin to /usr/share/rpikernelhack/bootcode.bin by rpikernelhack'
Removing 'diversion of /boot/start4.elf to /usr/share/rpikernelhack/start4.elf by rpikernelhack'
Removing 'diversion of /boot/start4cd.elf to /usr/share/rpikernelhack/start4cd.elf by rpikernelhack'
Removing 'diversion of /boot/start4db.elf to /usr/share/rpikernelhack/start4db.elf by rpikernelhack'
Removing 'diversion of /boot/start4x.elf to /usr/share/rpikernelhack/start4x.elf by rpikernelhack'
```

which take a while to complete. When I did this, it took about two hours.

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.

1.7 note

Raspberry Pi is a trademark of the Raspberry Pi Foundation.

2. 433MHz

2.1 Introduction

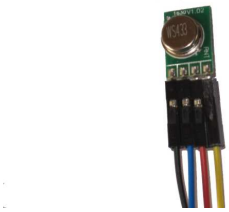
In the Netherlands, and perhaps beyond our borders too, there is a simple system for remote switching of lights, called `Klik aan Klik uit (KAKU)`. Although not the cheapest option available, it is quite affordable, especially if you use the `APA3-1500R Starterset`.

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.

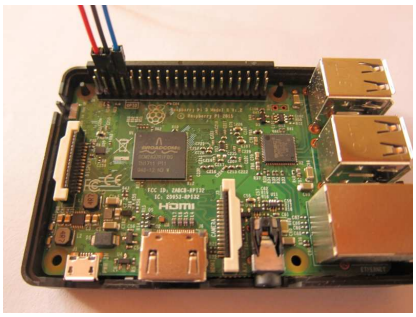
2.2 Step 1: Transmitter hardware

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.

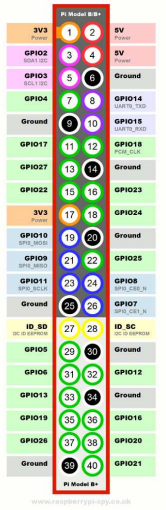


Pin	Color	Function	Comment
1	black	Ground	
2	blue	Data	
3	red	Vcc	5V; maximum is 12V
4	yellow	Antenna	marked ANT; should be 16.7 cm

These are connected to the Pi as follows:



In practice, this means that GPIO14 is used:



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. In buster, this should be part of the distribution, but it doesn't hurt to make sure it is installed.

```
sudo apt-get install wiringpi
```

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 ./newkaku 1 A on` within two seconds. And Lo and Behold: It switches! You can switch it off again with `sudo ./newkaku 1 A off`

Install `newkaku` in `/usr/local/bin` and make it set-uid:

```
sudo cp newkaku /usr/local/bin
sudo chmod u+s /usr/local/bin/newkaku
```

2.4 A website to switch

CLI is for real geeks; if you want to show your familiy and friends that it works, you will need a simple website.

2.4.1 Install a webserver

First that means that you should install a web server. My choice is Apache. I also dislike installing by hand. I therefore use an Ansible playbook.

`tasks/main:`

```
---
- name: install apache
  become: yes
  apt:
    name:
      - apache2

- name: enabled mod_rewrite
  apache2_module: name=rewrite state=present
  become: yes
  notify:
    - restart apache2

- name: enable cgi
  become: yes
  apache2_module:
      state: present
      name: cgid
  notify:
    - restart apache2


- name: copy default site
  become: yes
  copy:
      src: 000-default.conf
      dest: /etc/apache2/sites-available
  notify:
    - restart apache2
```

`handlers/main.yml`

```
---
- name: restart apache2
  become: yes
  service: name=apache2 state=restarted
```

`files/000-default.conf`

```
LoadModule cgid_module modules/mod_cgid.so
<VirtualHost *:80>

    ServerAdmin webmaster@localhost
    DocumentRoot /var/www/html
    ScriptAlias "/cgi-bin/" "/var/www/cgi-bin/"


    ErrorLog ${APACHE_LOG_DIR}/error.log
    CustomLog ${APACHE_LOG_DIR}/access.log combined

</VirtualHost>
```

If you are not using Ansible, use the `000-default.conf` and put it in `sites_enabled`

2.4.2 The website

I made a directory `/var/www/data/kaku` where all the data for my CGI script is stored. In that directory, there are two files:

- `data`
- `codes`

In the codes file, the codes for switching are placed:

```
0;all;11110001,11110002,11110003,11110004
101;switch 1;11110001
102;switch 2;11110002
103;switch 3;11110003
104;switch 4;11110004
```

The `data` file is used for a simple timer interface to the cronjob.

The scripts are in
`CGI script` and `cronjob` (a workable version) or at `"https://github.com/ljmdullaart/kakuweb"` (the most recent version).

2.5 note

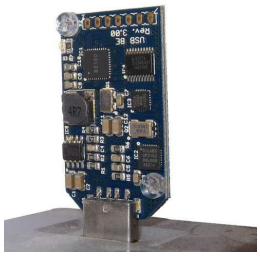
Raspberry Pi is a trademark of the Raspberry Pi Foundation.

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 shelved 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
make
```

And start mining:

```
```

And see the satoshis rolling 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...

3.3 note

Raspberry Pi is a trademark of the Raspberry Pi Foundation.

4. Change to SSL

4.1 Intro

At some point in time, you will want to upgrade your web server to SSL. This is how I did it.

4.2 Let's encrypt

The certificate-part is taken from "https://www.pestmeester.nl/index.html#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 successful 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/certbot-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/html
        <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 log records to help, starting in debug mode doesn't give additional 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 recommends using SSL doesn't mean that the SSL module is enabled by default. Sigh..

```
a2enmod ssl
```

reload apache and ...

Nothing.

Remove `000-default` Reload 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.

4.4 note

Raspberry Pi is a trademark of the Raspberry Pi Foundation.

5. A mail server

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

5.1.3 Via apt-get

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 443

— User defined language

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

5.1.4 Easy install

The problem with Easy Install is, that it is neither easy nor quick. The difficult part of the "easy" install is the dependencies. On my raspian, I had to install a number of development packages before the compiles succeeded.

```
sudo -s
cd /root
apt-get install libical-dev
apt-get install libldb-dev  libldap2-dev
apt-get install gettext
apt-get install autoconf
apt-get install libcurl4-openssl-dev
curl http://easyinstall.citadel.org/install | sh
```

Answer the questions that after a while appear, and you're in business. Login as admin and create your own user-id under the **Administration** button. For testing purposes, I also created a test-user, aptly named `test`.

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 `.fetchmailrc` 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/user.last_mail_error
```

to get mail every 15 minutes.

5.3 A backup server

A backup server is just another instance of Citadel. Every night, I make a backup of the relevant files to the backup server. I do a full backup; that means that I shutdown the primary mail server before the backup.

When the backup server is a newer version you cannot just copy the files to the backup server. Citadel will refuse to start, complaining about having both new and legacy configuration. apparently, citadel converts much of the legacy to the new when it starts up.

So, for me, the backup script is as follows. Note that `sigma` is the name of the backup server.

```
logger "BACKUP:citadel:START"
/etc/init.d/citadel stop
ssh sigma /etc/init.d/webcit stop
ssh sigma /etc/init.d/citadel stop
sleep 60
logger "BACKUP:citadel:STOPPED"
ssh sigma rm -rf /var/lib/citadel.old/*
ssh sigma rm -rf /etc/citadel.old/*
ssh sigma mv /var/lib/citadel/* /var/lib/citadel.old
ssh sigma mv /etc/citadel/* /etc/citadel.old

scp -r /var/lib/citadel/* sigma:/var/lib/citadel
scp -r /etc/citadel/* sigma:/var/lib/citadel/etc
scp ~ljm/.fetchmailrc sigma:/var/lib/citadel/fetchmailrc
logger "BACKUP:citadel:COPIED to sigma"

ssh sigma /etc/init.d/citadel start
ssh sigma /etc/init.d/webcit start
/etc/init.d/citadel start
logger "BACKUP:citadel:RESTARTED"
logger "BACKUP:citadel:END"
```

As a bonus, there is an extra copy of the mail (1 day older) that is saved.

5.4 note

Raspberry Pi is a trademark of the Raspberry Pi Foundation.

6. home.pl

CGI script for a simple home-webserver

```
#!/usr/bin/perl
#REMOTE@ chi /var/www/cgi-bin/home.pl
use strict;
use CGI;
use POSIX qw(strftime);

my $datafile='/var/www/data/kaku/data';
my $codefile='/var/www/data/kaku/codes';

my %crons;
my $display='switch';

if (open (my $DATA, "<", $datafile)){
	while (<$DATA>){
		chomp;
		(my $key,my $value)=split ',';
		$crons{$key}=$value;
	}
	close $DATA;
}
my $query = CGI->new();
print $query->header( "text/html" );
print $query->start_html(
	-title   => "Home",
	-style   => "http://chi.home/kaku.css",
	-meta    => {
		viewport => 'width=device-width,initial-scale=1,user-scalable=yes'
	}
);

my $debugout;

my $action='none';
my @lines ;

my %codes;
my %codelist;
if (open (my $CODES, "<", $codefile)){
	while (<$CODES>){
		chomp;
		s/#.*//;
		my @a=/;/g;
		if ( scalar @a == 2){

			(my $key,my $name,my $list)=split ';';
			$codes{$key}=$name;
			$codelist{$key}=$list;
		}

	}
	close $CODES;
}
else {
	print "<p>No code file</p>";
}

# read form data

my %params=$query->Vars;

if ($params{"display"} ne ''){
		$display=$params{"display"};
}
if ($params{"clock"} ne ''){
		$display='clock';
}
if ($params{"switch"} ne ''){
		$display='switch';
}

for my $k (keys %codes){
	if ($params{"aan$k"} ne ''){
			$action="aan$k";
			$display='switch';
	}
	if ($params{"uit$k"} ne ''){
			$action="uit$k";
```

```
                $display='switch';
        }
        for (my $i=0; $i<24; $i++){
                if ($params{"c$i$k"} ne ''){
                        $action="c$i$k";
                        $display='clock';
                }
        }
}
#
if ($action =~ /aan(..*)/) {
        my $key=$1;
        my @list=split(',',$codelist{$key});
        for (@list){
                system ("logger KAKU  $_ 1 on");
                system ("newkaku $_ 1 on | logger 2>&1");
        }
}
elsif ($action =~ /uit(..*)/) {
        my $key=$1;
        my @list=split(',',$codelist{$key});
        for (@list){
                system ("logger KAKU  $_ 1 off");
                system ("newkaku $_ 1 off | logger 2>&1");
        }
}
elsif ($action =~/^c/){
        for my $k (sort (keys %codes)){
                my $hr;
                if (exists ($crons{$k})){
                        if ($action =~ /c([0-9]*)$k/){
                                $hr=$1;
                                if ($crons{$k}=~/on$hr;/){
                                        $crons{$k}=~s/on$hr;/off$hr;/;
                                }
                                elsif ($crons{$k}=~/off$hr;/){
                                        $crons{$k}=~s/off$hr;//;
                                }
                                else {
                                        $crons{$k}="$crons{$k}on$hr;";
                                }
                        }
                }
        }
}


if (open (my $DATA, ">", $datafile)){
        for my $k (sort (keys %codes)){
                print $DATA "$k,$crons{$k}\n";
        }
        close $DATA;
}

#
# provide the output
#
#
#
#
print $query->h1( "Klik Aan, Klik Uit" );
print "\n";
print $query->hr;
print $query->start_form;
print $query->submit(
        -name    => "switch",
        -value   => 'switch',
        -class   => 'tab',
);
print $query->submit(
        -name    => "clock",
        -value   => 'clock',
        -class   => 'tab',
);
print $query->hidden(
        -name    => 'display',
        -value   => $display,
);
if ($display eq 'switch'){
        print "<table class='switchtable'>";
        print "\n";
```

```
	for my $k (sort (keys %codes)){
		print "    <tr>\n        <td class='swname'>$codes{$k}</td>\n        <td class='swon'>";
		print $query->submit(
			-name    => "aan$k",
			-value   => 'aan',
			-class   => 'onoff',
		);
		print "</td>\n        <td class='swoff'>";
		print $query->submit(
			-name    => "uit$k",
			-value   => 'uit',
			-class   => 'onoff',
		);
		print "</td>\n    </tr>\n";
	}
	print "</table>";
	#
}
elsif ($display eq 'clock'){
	print "<table>";
	print "\n";
	for my $k (sort (keys %codes)){
		print "    <tr>\n        <td>$codes{$k}</td>\n        <td>";
		for (my $i=0; $i<24;$i++){
			print "</td>\n        <td>";
			my $style;
			$style='cron';
			if ($crons{$k}=~/on$i;/){
				$style='cronon';
			}
			if ($crons{$k}=~/off$i;/){
				$style='cronoff';
			}
			print $query->submit(
				-name    => "c$i$k",
				-value   => "$i",
				-class   => $style,
			);
		}
		print "</td>\n    </tr>\n";
	}
	print "</table>";
	#


}
print $debugout;
print $query->end_html;
```

7. Cronjob for 433Mhz swiching

```
#!/bin/bash
#REMOTE@ chi /usr/local/bin/kakucronjob

VERBOSE=false
LOG=/tmp/kakucron.log
logsize=$(ls -s $LOG | sed 's/ .*//')
if [ $logsize -ge 500 ] ; then
    mv $LOG.1 $LOG.2
    mv $LOG $LOG.1
    touch $LOG
fi

date >>$LOG

if [ "$1" = "-v" ] ; then
       VERBOSE=true
fi

debug(){
       if $VERBOSE ; then
              echo $*
       fi
}

hour=$(date +%H | sed 's/^0//')
debug hour=$hour
NOW=$(date)
if [ -f codes ] ; then
       CODEFILE=codes
else
       CODEFILE=/var/www/data/kaku/codes
fi
debug CODEFILE=$CODEFILE
declare -A codemap

codefile=$(sed 's/ /_/g;s/#.*//' $CODEFILE)
for line in $codefile ; do
       debug line=$line
       key=${line%*}
       codes=${line##*;}
       codemap[$key]=$codes
       debug codemap filling $key=$codes ${codemap[$key]}
done

debug test ${codemap[test]}

if [ -f data ] ; then
       DATAFILE=data
       echo "NO GLOBAL DATA FILE">>$LOG
else
       DATAFILE=/var/www/data/kaku/data
fi
debug DATAFILE=$DATAFILE

if [ -x /usr/local/bin/newkaku ] ; then
       KAKU=/usr/local/bin/newkaku
else
       KAKU=/bin/echo
       echo "NO KAKU COMMAND">>$LOG
fi
debug KAKU=$KAKU

kakucode(){
       debug entering kakucode
       code="$1"
       sub="$2"
       state="$3"
       codes=${codemap[$code]}
       debug code=$code sub=$sub state=$state
       debug codes=$codes
       while [ "$codes" != "" ] ; do
              c=${codes%%,*}
              codes=${codes#*,}
              if [ "$codes" = "$c" ] ; then
                     codes=''
              fi
              echo "$c $sub $state">>$LOG
              $KAKU $c $sub $state | logger 2>&1
```

```
            sleep 1
            $KAKU $c $sub $state | logger 2>&1
      done
}

if $VERBOSE ; then
      echo -n "on: "
      cat $DATAFILE | grep "on$hour;" | sed 's/,.*//'
fi
cat $DATAFILE | grep "on$hour;" | sed 's/,.*//' |
while read code ; do
        logger "KAKU CRON ON $NOW - $hour: $code"
        debug kakucode $code 1 on
        kakucode $code 1 on
        sleep 1
        kakucode $code 1 on
        sleep 1
done

if $VERBOSE ; then
      echo ""
      echo -n "off: "
      cat $DATAFILE | grep "off$hour;" | sed 's/,.*//'
fi
cat $DATAFILE | grep "off$hour;" | sed 's/,.*//' |
while read code ; do
        logger "KAKU CRON OFF $NOW - $hour: $code"
        debug kakucode $code 1 off
        kakucode $code 1 off
        sleep 1
        kakucode $code 1 off
        sleep 1
done

if $VERBOSE ; then
      echo ""
fi
```

CONTENTS

1. Headless set-up 1
   1.1 Intro 1
   1.2 Burning the image 1
   1.3 The networking 2
   1.4 Enable ssh 2
   1.5 Connecting and manual actions. 3
   1.6 Security 4
   1.7 note 4
2. 433MHz 5
   2.1 Introduction 5
   2.2 Step 1: Transmitter hardware 5
   2.3 Step 2: Transmitter software 6
   2.4 A website to switch 7
   2.5 note 8
3. Bitcoin mining with a pi 9
   3.1 Intro 9
   3.2 Setting up the miner 9
   3.3 note 10
4. Change to SSL 11
   4.1 Intro 11
   4.2 Let's encrypt 11
   4.3 Apache 12
   4.4 note 12
5. A mail server 13
   5.1 Install Citadel on a Pi 13
   5.2 Fetchmail 14
   5.3 A backup server 14
   5.4 note 15
6. home.pl 16
7. Cronjob for 433Mhz swiching 19