Remote Control for Telescope

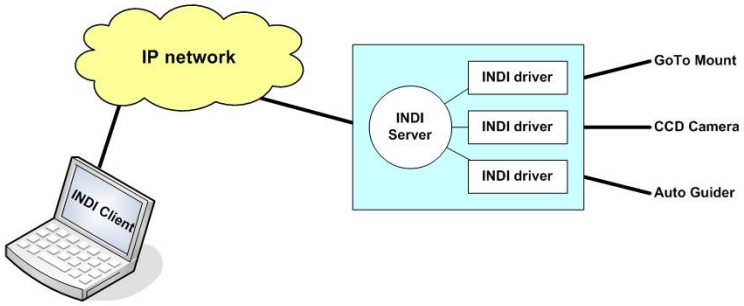
Arjan te Marvelde. November 2016

Optimum observation time is usually during the dark winter and early spring. On higher latitudes these months are also quite cold, and hence observation and astrophotography is less than comfortable. So it is time for a remote control facility, enabling the operation of mount and astrophotography from inside the home.

This article describes my configuration, based on a dual Raspberry Pi 2B (RPi2) configuration and the INDI framework. The description is chopped up in parts which can be used to build other configurations.

From INDI to dual RPi2

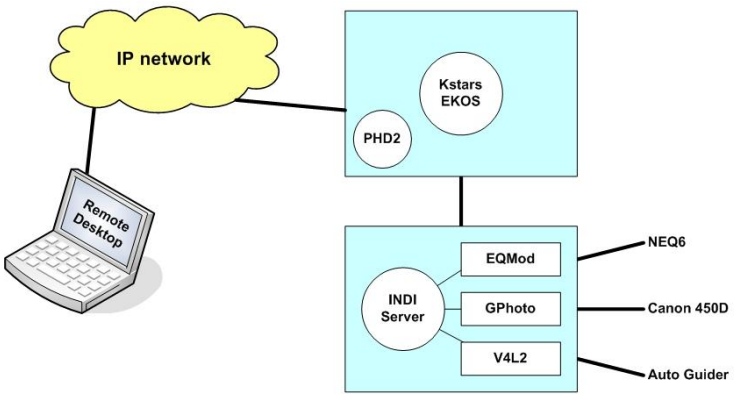
One option for implementing remote control is to route a lot of cables from the observation location into the house. A much more elegant way is to remote all device interfacing on a local embedded computer and have a remote workstation for controlling the setup. This is exactly what the INDI framework offers: a server that provides standardized method to access a wide variety of attached components, such as goto control, camera control and auto guider. The INDI server is connected to INDI compatible clients on the workstation through any IP network, for example the residential (wireless) LAN.



Standard INDI Architecture

When the workstation PC is Windows based, the choice of clients boils down to the planetarium Cartes du Ciel for goto control and the photo capturing software CCD-Ciel. The main alternative running on Linux is KStars / EKOS. This may run in a virtual machine on Windows or MacOS.

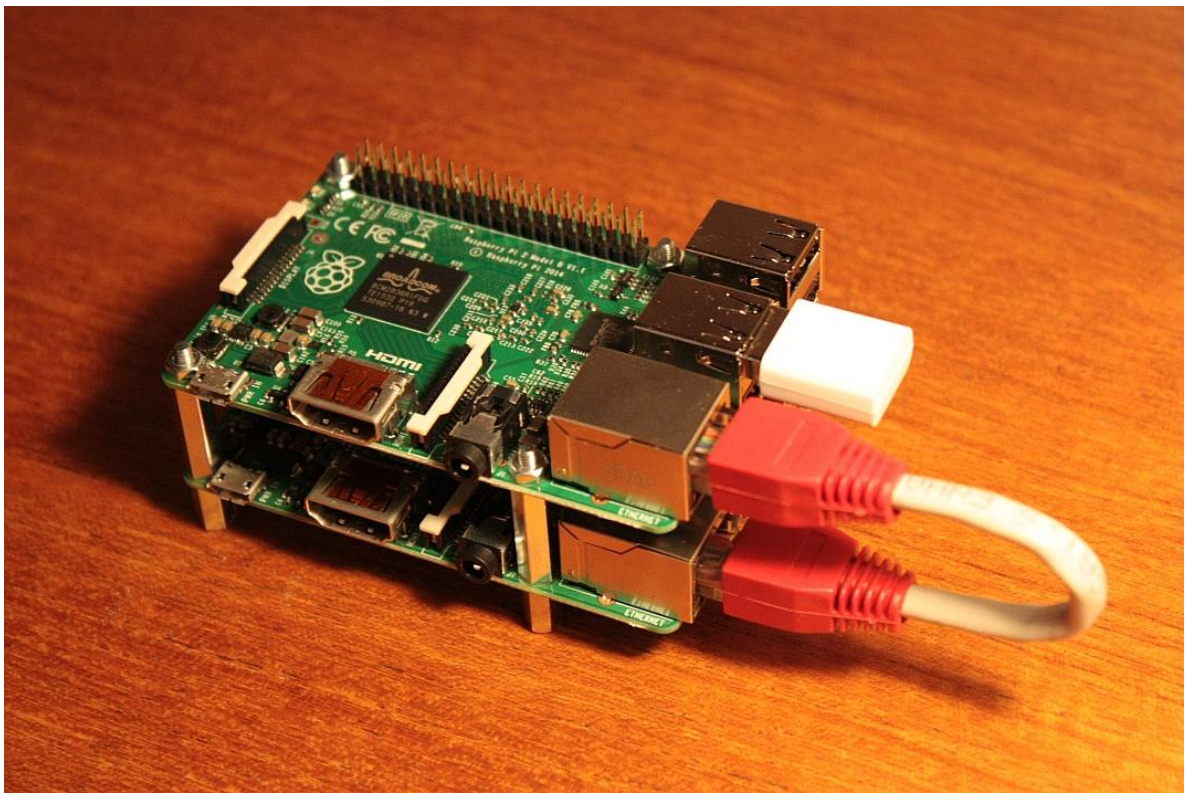
A more elegant and flexible solution is based on dual RPi2, one doing the indiserver stuff and the other running KStars. The user interface can be accessed through a VNC virtual desktop, on the Windows workstation or even on a tablet or a smartphone. Another option is to use X11 natively with an XServer running on the Windows machine[1].



Dual Pi architecture

[1] In the Feb 2016 version of this article I moved control to the Windows laptop, based on CCD-Ciel. However, this has proven to be a fairly tedious solution and so the this article optimizes the KStars on RPi method.

Raspberry Pi configuration



Dual RPi2 boards, upper-pi and lower-pi

The software configuration is divided in several items that can be selected for installation, for each of which the setup is described below. This article is a log of how I did it, but this is ultimately based on what I scrounged together from the web and the INDI forum: see the links at the end of this article. **`Boldface`** used in `code` examples should be customized to own situation.

The steps to be followed:

1. Install Ubuntu Mate on both boards
2. Setup the networking
3. Update Ubuntu packages
4. Install and configure X11 for remote access on Upper Pi
5. Install EKOS/KStars on Upper Pi and INDI server and drivers on Lower Pi

1 Ubuntu installation

Download **Ubuntu Mate 16.04** for Raspberry Pi, unpack it with **7-zip** and for each RPi write it on a fresh Micro-SD (using e.g. **Win32DiskImager**) according to instructions that can be found on the web.

File system

First a user must be created (e.g. **pi**) during initialization. Then open a CLI (**Applications > System Tools > MATE Terminal**) to expand the partition on the SD card:

```
sudo fdisk /dev/mmcblk0
```

Inside **fdisk** remove partition 2 (**d, 2**), and then create it again using the full SD space (**n, p, 2, Enter, Enter**). Finally save the new partition table (**w**). Then reboot the RPi2, on the CLI enter **sudo reboot** or use the appropriate option via the GUI.

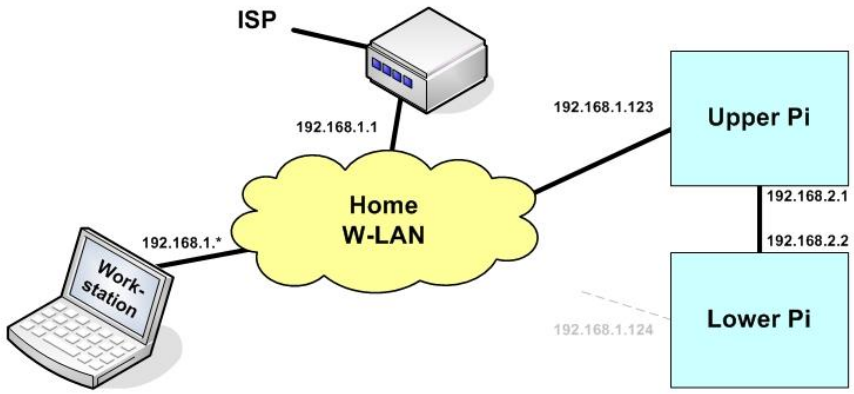
After the user environment has loaded again, open a CLI and then commit the full partition to the filesystem:

```
sudo resize2fs /dev/mmcblk0p2
```

As of 16.04 an alternative method for resizing is to click the raspberry button in the welcome screen.

2 Networking

There are several ways to do this, but I found that this way is working in general. It assumes the presence of a wireless local area network, as typically found in/around peoples' homes these days. This network usually has a gateway that provides access to the internet. The INDI system must be accessible from this residential LAN.



The configuration consists of two RPi2 modules, an Upper and a Lower one since they are mounted on top of each other. The Upper Pi has the WLAN interface, and also connects to the Lower Pi by means of a short Ethernet cable. The Upper Pi will act as a router between the home LAN and the local RPi2 IP subnets. In the configuration of the Lower Pi also an optional WLAN interface may be configured.

An alternative approach, not further discussed in this document, is to configure the Upper Pi as a Wireless Access Point. This may be handy when the setup is used in the field. Note that this requires a WiFi dongle that supports this mode.

Basic network settings:

The addresses used in this section should be adapted to the target situation.

Two Upper-Pi interfaces need to be configured, the Ethernet and the WiFi. First obtain the logical names given by Ubuntu:

```
ifconfig
```

This should give the local loop (`lo`), an ethernet interface name starting with '`e`' and optionally the WLAN interface name starting with '`w`'. On my version it is `enx*` and `wlx*`, where the wildcard is filled in with the MAC address of the interface. In the following the more generic `wlan0` and `eth0` are used as interface names instead.

Both boards will be configured with static IP addresses, so DHCP is not used.

The file `/etc/network/interfaces` needs to be changed to define the static addresses for each interface, and to pass parameters to the WiFi driver.

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

For the Upper-Pi the contents of the file should look like:

```
#The loopback interface
auto lo
iface lo inet loopback

#The WiFi interface
auto wlan0
iface wlan0 inet static
  wpa-ssid YOUR-SSID
  wpa-psk YOUR-PASSKEY
  address 192.168.1.123
  network 192.168.1.0
  netmask 255.255.255.0
  broadcast 192.168.1.255
  dns-nameservers 192.168.1.1 8.8.8.8 8.8.4.4
  post-up route add default gw 192.168.1.1

#The ethernet interface
auto eth0
iface eth0 inet static
  address 192.168.2.1
  network 192.168.2.0
  netmask 255.255.255.0
  broadcast 192.168.2.255
  post-up iptables-restore < /etc/iptables.rules
```

Apart from defining the interfaces, the default gateway (`192.168.1.1`) is added here as an entry in the routing table. Also the forwarding rules are loaded in iptables.

A similar setting is done for the Lower-Pi:

```
#The loopback interface
auto lo
iface lo inet loopback

#The ethernet interface
auto eth0
iface eth0 inet static
  address 192.168.2.2
  network 192.168.2.0
  netmask 255.255.255.0
  broadcast 192.168.2.255
  gateway 192.168.2.1
  dns-nameservers 8.8.8.8 8.8.4.4
```

Here the WiFi interface is not defined, and no routing is required.

Routing:

The Upper-Pi has two network interfaces, and a routing table. Now IP forwarding must be enabled, so that packets will be actually forwarded from one interface to the other.

In the file `/etc/sysctl.conf` the following line must be uncommented, or changed to read:

```
net.ipv4.ip_forward=1
```

Finally commit the sysctl change with:

```
sudo sysctl -p
```

Also the forwarding behaviour must be defined in iptables:

```
sudo iptables -A FORWARD -i eth0 -o wlan0 -j ACCEPT
sudo iptables -A FORWARD -i wlan0 -o eth0 -j ACCEPT
sudo sh -c "iptables-save > /etc/iptables.rules"
```

The line that loads these rules was already added to **`/etc/network/interfaces`**.

Finally, after rebooting the lot, it is a good idea to remove the network manager on both boards, in order to prevent interface management conflicts:

```
sudo apt-get purge network-manager
```

Local names can be saved in `/etc/hosts` for easier resolution, so e.g. add lines:

```
192.168.2.2    lower-pi
192.168.2.1    upper-pi
```

LAN home network:

In order to make the RPi local network accessible and enable internet access from it, the residential LAN router must be configured as well. A static route must be added to the `192.168.2.0` network, where the next hop node is the wireless interface of the Upper-Pi (in the example: `192.168.1.123`).

Testing:

From Upper-Pi as well as Lower-Pi (through e.g. SSH terminal) try to `ping google.com`; if this works routing is okay as well as the name resolution.

3 Ubuntu updates

At this moment both Pi boards are running Ubuntu and can access the internet. It is a good idea to do an update and upgrade of the installed packages after the internet link has been enabled:

```
sudo rpi-update
sudo reboot
sudo apt-get update
sudo apt-get upgrade
```

The updates can take a fairly long time...

Graphical User Interface (or not):

The RPi2 normally acts in graphical console mode and has a GUI, which is accessible through a HDMI monitor, a mouse and a keyboard or via a remote desktop. Presumably this is what has been used to set-up both boards.

For the Lower-Pi this GUI mode can be switched to headless-server mode by executing:

```
sudo graphical disable
sudo reboot
```

The local monitor or a remote login through SSH the give a command line interface (CLI). For SSH from a Windows PC I use **putty**, using the RPi2 IP-address filled in under "Session" and a 30 second keep-alive timeout under "Connection". Best to save this setting for future use...

4 Remote access

Next we need to create a remote GUI on the workstation PC, which is done based on VNC. There are various different candidates, I eventually got it working with the **vinoserver**. This server duplicates the desktop to a conencted VNC client. In fact, whatever you do remotely is copied on the local display and vice versa.

```
sudo apt-get install vino
```

Then update **config.txt** to set the proper display parameters:

```
sudo nano /boot/config.txt
```

by means of the following (uncommented) lines:

```
hdmi_force_hotplug=1
hdmi_group=2
hdmi_mode=18
```

In my case this is a working setting for a standard 1024x768 display, but you may need to experiment.

For some reason automatic start during booting doesn't work, probably because no display is active at that time. A work-around is to have a startup script (e.g. **~/start**):

```
export DISPLAY=:0
/usr/lib/vino/vino-server&
```

and execute this from a SSH session.

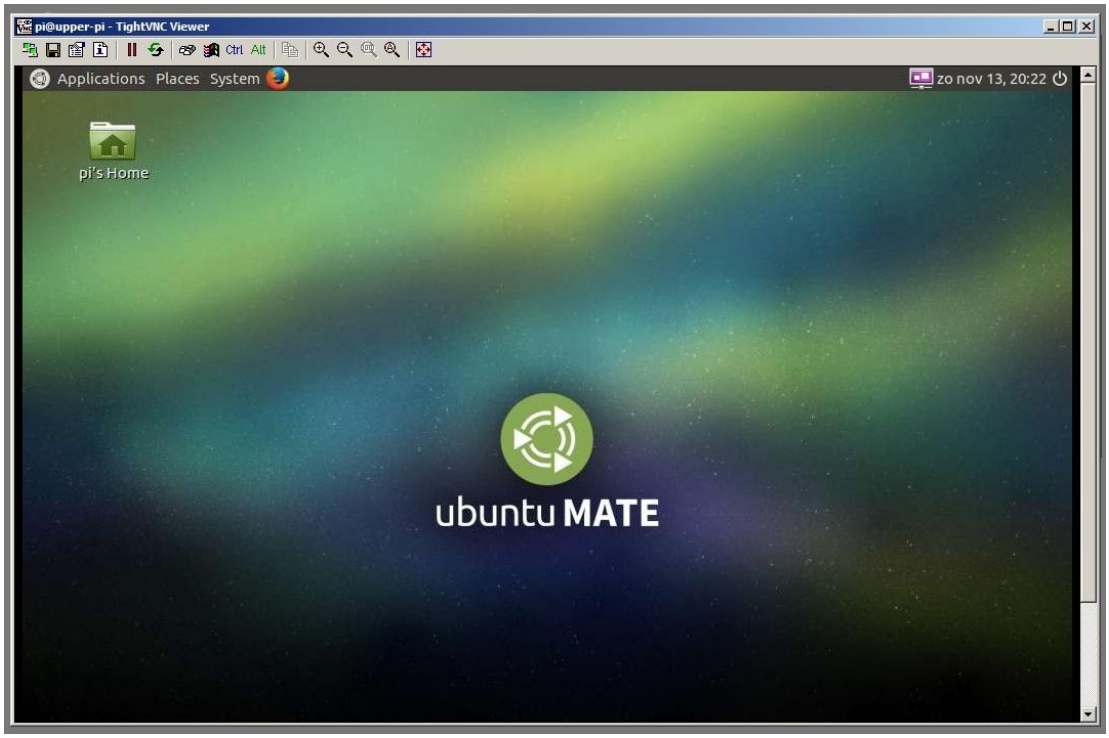
The Upper-Pi should have auto-login enabled. This can be achieved by editing the file **/usr/share/lightdm/lightdm.conf.d/60-lightdm-gtk-greeter.conf** to contain:

```
[Seat:*]
greeter-session=lightdm-gtk-greeter
autologin-user=pi
```

and obviously here the user name is **pi**.

Windows VNC client:

On the PC you can install any VNC client, but I have used TightVNC successfully. All you need to do is fill in the Upper-Pi IP address (WLAN side) without any port. Starting the connection will give something like:



Upper-Pi in TightVNC

From here all works as if connected locally with a screen and keyboard/mouse.

5 KStars/EKOS/INDILib

KStars:

To install the Kstars suite on the Upper-Pi, from a CLI type:

```
sudo apt-add-repository ppa:mutlaqja/ppa
sudo apt-get update
sudo apt-get install kstars-bleeding
```

You can start KStars on the Ubuntu GUI from Applications→Education→Kstars and fill in the parameters for connecting to the INDI server. Before trying to establish the connection, obviously the Lower-Pi that runs the **indiserver** needs to be installed and started up.

INDI-lib:

You can get the latest indy library directly from the PPA like previously registered on Upper-Pi. On the Lower-Pi issue the commands:

```
sudo apt-add-repository ppa:mutlaqja/ppa
sudo apt-get update
sudo apt-get install indi-full
```

In case of errors follow the suggestions given by **apt-get**. Finally issue a reboot on Lower-Pi.

Starting INDI:

The specific **indiserver** settings depend on your situation. In my case, using EQMod and Canon 450D, I would start the server like:

```
indiserver -v -m 200 indi_eqmod_telescope indi_gphoto_ccd
```

The **-m 200** is required, because of queue overflow due to the large image files from the GPhoto device. The **-v** option is not really needed when everything runs smoothly, maybe better to leave out to prevent **syslog** overflow.

To enable serial port write access, the user should be in the **dialout** group. Assuming the user is identified as **pi**:

```
sudo adduser pi dialout
```

Now we need to make sure the indiserver runs automatically after startup.

The Ubuntu service on the Lower-Pi is started as follows. First define a service:

```
sudo nano /lib/systemd/system/indi.service
```

Then copy the following lines (changing parameters where needed):

```
[Unit]
Description=Start INDI server at startup.

[Service]
User=pi
Group=pi
```

```
Type=simple
Restart=always
ExecStart=/usr/bin/indiserver -v -m 200 indi_eqmod_telescope indi_gphoto_ccd

[Install]
WantedBy=graphical.target
WantedBy=multi-user.target
```

And finally register the service:

```
sudo systemctl enable indi.service
sudo systemctl daemon-reload
sudo systemctl restart indi.service
```

To avoid interference of other applications with the **indi_gphoto_ccd** driver the following has to be taken care of.

Remove the following files (if not already done):

```
sudo rm /usr/share/dbus-1/services/org.gtk.Private.GPhoto2VolumeMonitor.service
sudo rm /usr/share/gvfs/mounts/gphoto2.mount
sudo rm /usr/share/gvfs/remote-volume-monitors/gphoto2.monitor
sudo rm /usr/lib/gvfs/gvfs-gphoto2-volume-monitor
```

When in GUI mode sometimes the Canon SD card is mounted to a desktop folder. In order to prevent this, do:

```
gsettings set org.gnome.desktop.media-handling automount false
```

Note: This does not work from a SSH session and also not when graphics mode is disabled.

Chaining indiservers:

The Upper-Pi will host the guider, so here also an indiserver needs to run to make it accessible for EKOS/KStars. In the startup script (**~/start**) , before vino is started, add the line:

```
/usr/bin/indiserver -m 200 indi_v4l2_ccd "EQMod Mount"@lower-pi "GPhoto CCD"@lower-pi &
```

Obviously this represents my case, device references as well as hostname of lower-pi should be changed as required.

Now KStars/EKOS needs to refer to the local indiserver in order to access all devices on both RPi boards.

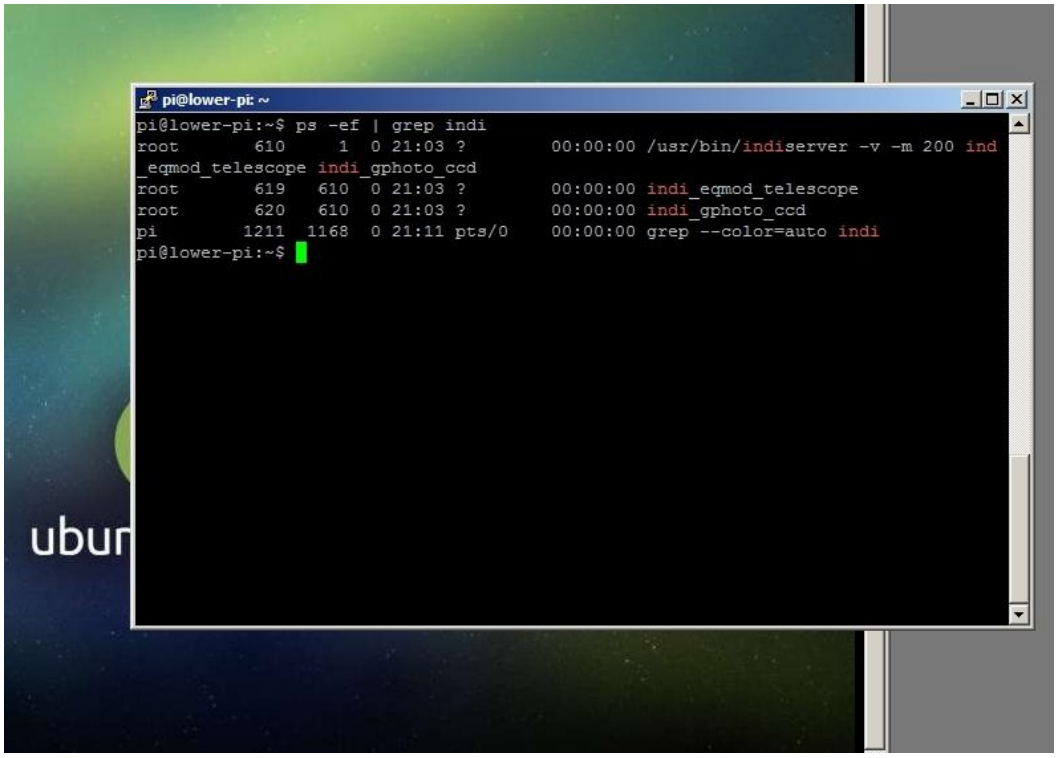
6 Testing

After power cycling the INDI-Box, an SSH session is opened to the Upper-Pi, in order to run the start script to start Vino and indiserver as a background jobs. After this the SSH can be closed:

```
./start
exit
```

Finally, the TightVNC client that runs on teh PC is connected to the Upper-Pi to obtain the remote desktop.

An SSH connection to the Lower-Pi may now be opened to check whether `indiserver` is running:



On the remote desktop now KStars can be started (Applications ¬ Education ¬ KStars).

The settings must be updated to reflect correct coordinates and time as well as to connect to the indiserver.

7 Auto Guiding

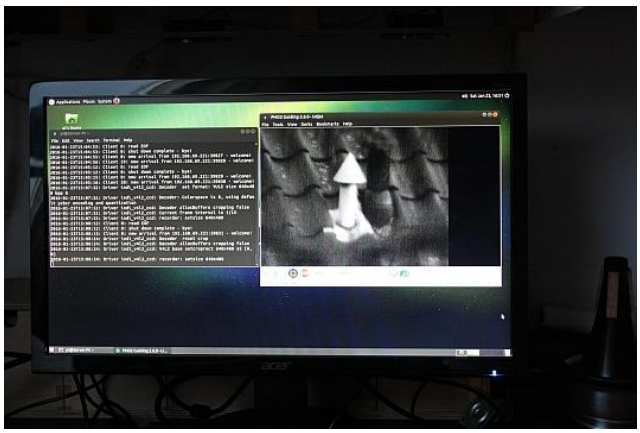
TO BE COMPLETED/TESTED

The PHD2 is a version of OpenPHD that is built for an ARM7 platform, with hardware floating point processor (such as the RPi2). It requires Ubuntu as operating environment.

```
sudo add-apt-repository ppa:pch/phd2
sudo apt-get update
sudo apt-get install phd2
```

This package also requires a GUI to run in, either locally or on a remote desktop. In case it is run on a headless Ubuntu server, it can use X11 to a remote Xserver, for example on the PC.

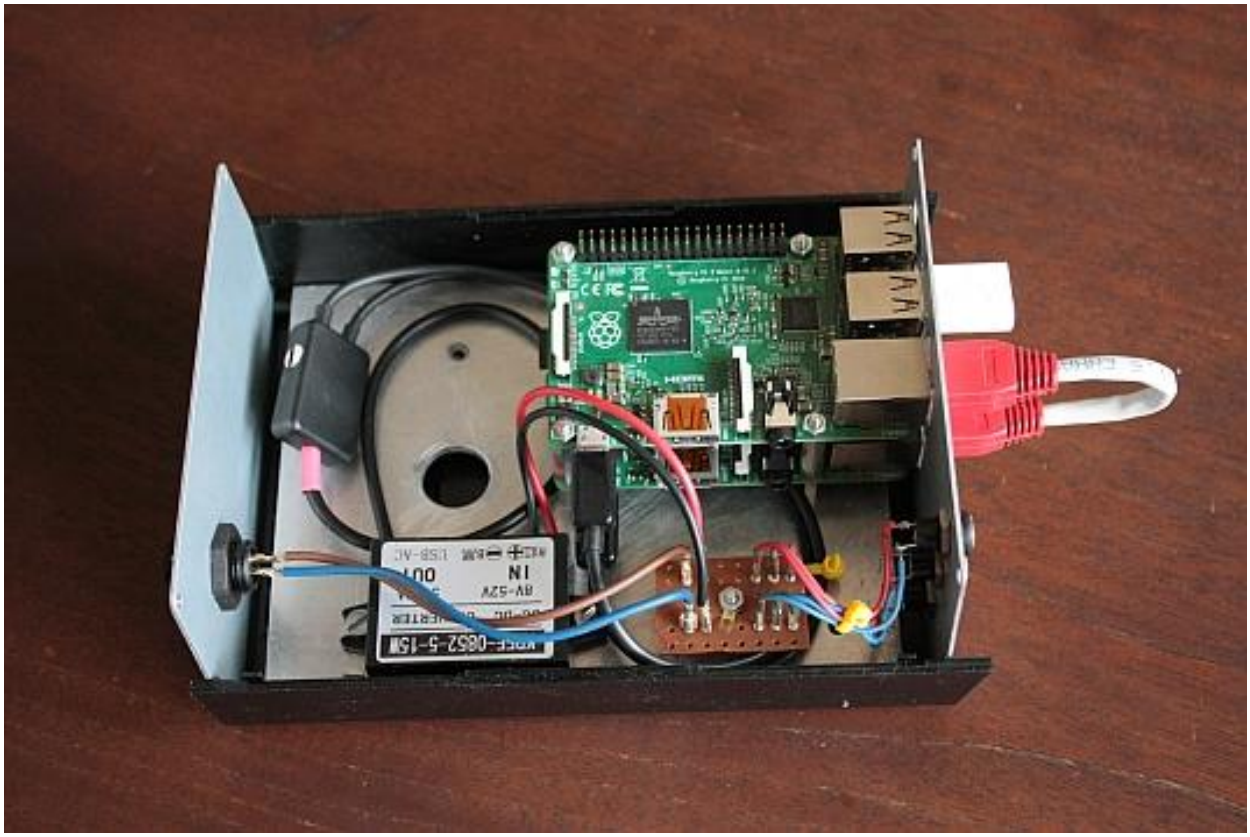
It requires input from a guide camera, and output to the indiserver that runs the EQMod device. In my case the camera input comes from a webcam, through the V4L2 device and a second INDI server.



The image above shows a ZS80 scope with a Philips webcam pointing at a distant roof. PHD2 runs on the upper RPi2, where also the scope and DSLR are conencted through an INDI server. The webcam is connected to the lower RPi2 and accessed through a local INDI server. Hence PHD2 uses information from two different server instances!

The INDIbox

My INDI configuration has been built into a Teko KL22 enclosure, which has plenty space.



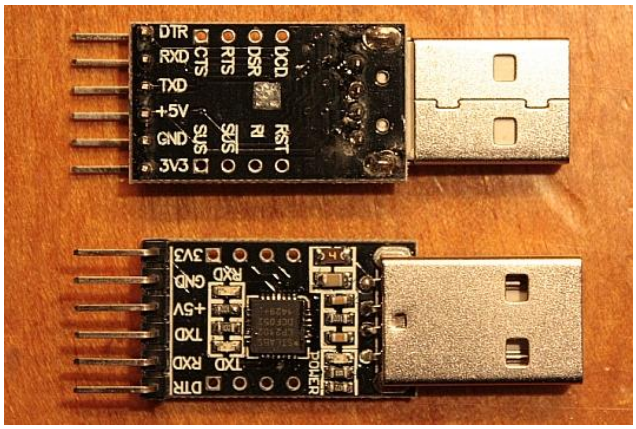
In the picture you can see the dual RPi2, a 3Amp USB power buck converter and a dual micro-USB splitter connected to it. The lot runs on appr. 12V (PSU or battery) which is also put out on the front to supply the NEQ6, the DSLR and a fan. These outlets should really be fused separately, but laziness is a common thing...

The Ethernet is connected between RPi2 boards and a WiFi dongle is already connected. This leaves three USB ports on the Upper-Pi (NEQ6 and DSLR) and four on the Lower-Pi (Webcam).

On the rear side the power input connector is located. It can connect to a power supply (12V/5A) or for example to a 13.8V battery.

NEQ6 - EQMod:

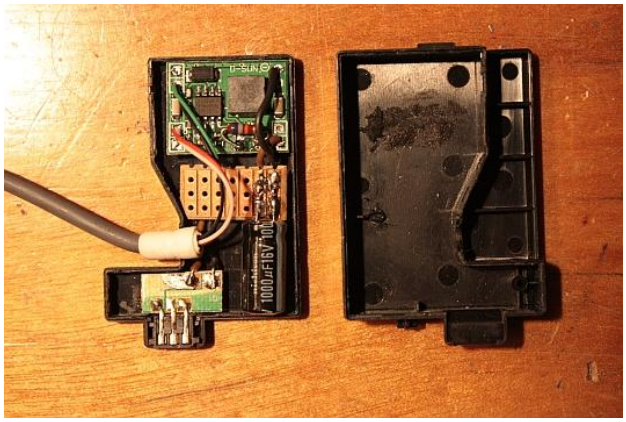
The NEQ6 mount is connected using the INDI EQMod device driver. This device requires a serial interface to be connected via USB. For this purpose, a CP2102 based converter was purchased and built into a small enclosure. Note that the serial side should be at 5V TTL level.



I carefully de-soldered the USB and pinheader connectors, and replaced those with a half USB and a half serial cable. This should work identically to an EQDir USB cable.

DSLR Power:

For powering the Canon 450D directly instead of a battery (which will run empty), I found a cheap plastic adapter on the web. This is nothing more than a battery shaped enclosure that just contains a pair of elco's, which should be powered from an external net adapter.



Since I wanted to connect this directly to the 12V outlet of the INDI box, instead I pried the adapter open and used the empty space to put in a cheap tiny buck converter. The voltage setting potentio meter did not work, and was replaced with a suitable fixed resistor, to yield a 7.6V output voltage. I re-used one of the elco's and added a 100nF capacitor for further filtering purposes.

Summary

To wrap things up a summary of the complete configuration follows here.

In terms of computing platforms I have a:

Laptop running Windows, which is connected to the residential WLAN

RPi2 (**Upper-Pi**) running Ubuntu-Mate 16.04, connected to the WLAN as well as to the local Ethernet segment

RPi2 (**Lower-Pi**) running Ubuntu-Mate 16.04, only connected to the Ethernet segment

The **Lower-Pi** only hosts an INDI-Server which provides access to ASCOM mount, Canon 450D DSLR and optionally to a guide camera. It runs in headless mode.

The **Upper-Pi** hosts EKOS/KStars.

The Windows **Laptop** hosts an X-Window server, enabling GUI to the Upper-Pi applications. This laptop could be replaced with any other device providing X11 or VNC access.

The **Upper-Pi** and **Lower-Pi** are contained in a single enclosure, the **INDIbox**, that provides power and data interfaces for all peripherals.

The 450D DSLR is connected through USB and a dedicated power adapter.

The NEQ6 mount is connected through an EQDir compatible interface and a power cable.

The Philips SPC900 webcam is connected via USB only.

References

The INDI tutorial (“Painless remote control with Ekos/INDI”)

http://indilib.org/support/tutorials/150-painless-remote-control-with-ekos-indi.html

INDI Forum, topic: Mobile astrophotography with RPi2

http://INDIlib.org/forum/general/750-mobile-astrophotography-with-rpi2.html

Raspbian or Ubuntu for the RPi2:

https://www.raspberrypi.org/downloads/

https://ubuntu-mate.org/raspberry-pi/

Terminal emulator, PuTTY:

http://www.putty.org/

Writing a disk image to SD Card, Win32DiskImager:

https://sourceforge.net/projects/win32diskimager/

X11 Windows server for MS-Windows, VcXsrv:

https://sourceforge.net/projects/vcxsrv/

Cartes du Ciel, CCDCiel, PHD2 (Ubuntu/INDI):

http://www.ap-i.net/ccdciel/en/start

http://www.ap-i.net/skychart/nl/start

https://launchpad.net/~pch/+archive/ubuntu/phd2