Raspberry Pi headless setup + daemon

1. Use instructions from TomsHardware.com
2. Install Raspberry Pi Imager for your workstation
3. Enable SSH. Write an empty text file named "ssh" (no file extension) to the root of the directory of the card. When it sees the "ssh" on its first boot-up, Raspberry Pi OS will automatically enable SSH (Secure Socket Shell), which will allow you to remotely access the Pi command line from your PC.
4. Set up a network connection. To set up a Wi-Fi connection on your headless Raspberry Pi, create a text file called wpa_supplicant.conf, and place it in the root directory of the microSD card. You will need the following text in the file. If you're already on the device, edit /etc/wpa_supplicant/wpa_supplicant.conf.

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

network={
        ssid="YourHomeNetwork"
        psk="YourHomeNetworkPassword"
        key_mgmt=WPA-PSK
        id_str="home"
}

network={
        ssid="RFO"
        psk="d0g!st@r"
        key_mgmt=WPA-PSK
        id_str="rfo"
}
```

   Test the configuration by restarting the network service with the new entries in place in wpa_supplicant.conf:
   > sudo systemctl restart networking
   If you can still connect to the internet, you were successful. A stronger test is to reboot and verify that the configuration comes up as expected.

5. Close Raspberry Pi Imager and insert SD card in Raspberry Pi. Boot it up.

6. Connect via ssh. (user 'pi', see password bank for password)

7. Set up a static IP address. You must do this while inside the target network.
   a. Static IP info for the Power Meter Raspberry Pi (powerpi?):
      i. IP address: 192.168.74.11

ii. Netmask: 255.255.255.0 (aka /24)
iii. Gateway: 192.168.74.1

b. Edit /etc/dhcpcd.conf and add this section:

```
# static ip address at RFO on wireless
interface wlan0
static ip_address=192.168.74.11
static routers=192.168.74.1
static domain_name_servers=192.168.74.1
```

This configuration is in the powerpi already, commented out until ready for installation at the observatory.

c. Full network design for the observatory in this spreadsheet and this diagram.

8. Set up remote access via VPN
   a. Access over the VPN requires that you add a static route to 10.74.74.0/24 via 192.168.74.10 (http://clusterfrak.com/kb/linux_kb/linux_static_route/). Add these lines to /etc/network/interfaces

```
# Static routes and VPN routing for powerpi at RFO
auto wlan0
iface wlan0 inet Static
address 192.168.74.11
netmask 255.255.255.0
gateway 192.168.74.1
up route add -net 10.74.74.0 netmask 255.255.255.0 gw 192.168.74.10
```

   b. Then restart networking:
      > sudo systemctl restart networking.service
   c. Verify the route:
      > sudo route -n

9. Enable VNC.

```
sudo raspi-config
{select Interfacing Options}
{Enable VNC}
```

10. Download and install VNC on your workstation. Connect to the Raspberry Pi using the static IP address you set up above.

11. Set up the watchdog reset.
    https://diode.io/raspberry%20pi/running-forever-with-the-raspberry-pi-hardware-watchdog-20202/

Watchdog is loaded by default in Raspberry Pi 4. No need to add the dtparam to the /boot/config.txt file.

> sudo apt-get install watchdog

That install created the /etc/watchdog.conf file. Edit it.

> sudo vi /etc/watchdog.conf

Remove the leading # (comment) from:
#watchdog-device = /dev/watchdog

After that line add this line:
watchdog-timeout = 15

Remove the comment from:
#max-load-1 = 24

Remove the comment and change the interface name from eth0 to wlan0:
interface = wlan0

Write the file.

Now you can enable, start and check status of the watchdog:
> sudo systemctl enable watchdog
> sudo systemctl start watchdog
> sudo systemctl status watchdog

The last command gives you feedback that it is working:

pi@raspberrypi:~ $ sudo systemctl status watchdog
● watchdog.service - watchdog daemon
   Loaded: loaded (/lib/systemd/system/watchdog.service; enabled; vendor preset: enabled)
   Active: active (running) since Thu 2021-03-11 12:06:58 PST; 7s ago
  Process: 10619 ExecStartPre=/bin/sh -c [-z "${watchdog_module}"] || ["${watchdog_module}" = "none"] || /sbin/modprobe $w
  Process: 10620 ExecStart=/bin/sh -c [$run_watchdog != 1] || exec /usr/sbin/watchdog $watchdog_options (code=exited, status
 Main PID: 10622 (watchdog)
    Tasks: 1 (limit: 4915)
   Memory: 628.0K
   CGroup: /system.slice/watchdog.service
           └─10622 /usr/sbin/watchdog

Mar 11 12:06:58 raspberrypi watchdog[10622]: interface: wlan0
Mar 11 12:06:58 raspberrypi watchdog[10622]: temperature: no sensors to check
Mar 11 12:06:58 raspberrypi systemd[1]: Started watchdog daemon.
Mar 11 12:06:58 raspberrypi watchdog[10622]: no test binary files
Mar 11 12:06:58 raspberrypi watchdog[10622]: no repair binary files

Mar 11 12:06:58 raspberrypi watchdog[10622]: error retry time-out = 60 seconds
Mar 11 12:06:58 raspberrypi watchdog[10622]: repair attempts = 1
Mar 11 12:06:58 raspberrypi watchdog[10622]: alive=/dev/watchdog heartbeat=[none] to=root no_act=no force=no
Mar 11 12:06:58 raspberrypi watchdog[10622]: watchdog now set to 15 seconds
Mar 11 12:06:58 raspberrypi watchdog[10622]: hardware watchdog identity: Broadcom BCM2835 Watchdog timer

You can also test by running a 'fork bomb' in the shell that will exhaust all resources:

> sudo bash -c ':(){ :|:& };:'

12. Test the **rfopower** application. Success criteria:
    a. Connect successfully to adafruit.io
    b. Connect successfully to Omnimeter through USB/RS485 link
    c. Successful data read from Omnimeter
    d. Successful data write to adafruit.io dashboard
    e. Successful logging to /home/pi/Projects/ekm/rfopower.log
    f. Successful write to /run/rfopower.pid file

13. Set up **rfopower** in systemd startup configuration

All startup programs must run as root, no matter which approach you take. The libraries used by **rfopower** are adafruit-io and ekmmeters. When installed with *pip3* they don't always get installed so root can find them when it runs **rfopower.**

Check with pip3:
> pip3 show ekmmeters adafruit-io
> sudo pip3 list
> sudo pip3 show ekmmeters adafruit-io
For me, this showed ekmmeters installed for root but not adafruit-io

> sudo -u root pip3 install adafruit-io
> sudo -u root pip3 show adafruit-io
This results in the adafruit-io library being installed globally for all users.

Daemon program setup

rfopower is setup as a python3 executable by starting the file with the location of python3 in your pi and by chmod +x rfopower from the command line.

Features of rfopower are:
- Sets up log file at /home/pi/Projects/ekm/rfopower.log
- Sets up pid file at /run/rfopower.pid
- Sets up Adafruit.io username and io_key

- Custom rfo_print_log() function
- Process id is written to pid file at start up
- Feeds are checked and set up at adafruit.io as needed
- Connect to Omnimeter once via USB/RS485 port
- Sets CTratio to 100 for our circuit panel
- Main loop: reads data from Omnimeter and writes to adafruit.io

run-rfopower.sh is a bash executable file that is the wrapper function that starts up ***rfopower*** as a background process with STDERR and STDOUT joined. It first checks to see if the process is already running, and exits if it is running already. Create a link (ln -s /home/pi/Projects/ekm/run-rfopower.sh) in the /usr/local/bin folder.

rfopower.service is the configuration file for the rfopower service needed by systemd to make this all work in startup. Key configurations are:

- After=network.target in [Unit] to allow the network to come up before we try to run this service
- StartLimitIntervalSec=0 in [Unit] to let systemd restart forever. Revisit this if you are encountering bad behavior.
- Type=forking in [Service] to tell systemd that the child process is the one to watch and to expect that service process to not stop.
- PIDFile=/run/rfopower.pid in [Service] is required with Type=forking and the service itself MUST put the process id in the pidfile.
- Restart=always in [Service] to not limit the number of restart attempts.
- RestartSec=10 in [Service] to wait 10 seconds before restarting to keep the pi from running hot in permanent failure situations.
- User=root in [Service] because nothing else works (!)
- ExecStart=/usr/local/bin/run-rfopower.sh in [Service]
- As the comments in the file remind you, create a link to this file in the /usr/local/lib/systemd/system folder and in /etc/systemd/system directory.

The startup sequence for a new daemon/service is:

- **sudo systemctl status rfopower** and check the output to show it is inactive.
- **sudo systemctl start rfopower** and check to see that the rfopower process has started and continues to run. You should also check the logs using **journalctl -xe --unit=rfopower**
- You can validate that the services is not enabled by running **sudo systemctl is-enabled rfopower**.
- Enable the new rfopower service with **sudo systemctl enable rfopower**.

If all is working, you will see that after the start command the rfopower process is running and remains running and that it starts up when the pi is rebooted. There are many more interesting options for **journalctl** and **systemctl** that are documented in their respective man pages. I found the best results by running always as sudo, even though I could often successfully run those utilities as the pi user.

14. Troubleshooting

- Aliases set up in the pi user:

- alias rj='journalctl -b --unit=rfopower'
  Shows systemd journal since last boot for rfopower service unit

- alias rlog='tail /home/pi/Projects/ekm/rfopower.log'
  Shows last 20 lines of rfopower log file

- alias rps='ps -ef | grep rfopower |grep -v grep'
  Shows list of processes running as the rfopower daemon

- alias rstart='sudo systemctl start rfopower'
  Starts the rfopower daemon - the daemon checks to see if another instance is already runs and does not run if one exists

- alias rstatus='sudo systemctl status rfopower -n50'
  Shows the systemd status of rfopower service daemon, including the last 50 lines of the journal

- alias rstop='sudo systemctl stop rfopower'
  Stops the rfopower daemon.

- Every startup of the rfopower daemon puts the STARTING RFOPOWER log entry into the rfopower log.
- All scripts, python programs, service files and logs are located in /home/pi/Projects/ekm folder.
- The adafruit.io dashboard is located at https://io.adafruit.com/georgel1/dashboards/rfo-electric-service
- If data flow appears to have stopped coming to the dashboard:
  - Connect to the powerpi
  - Check that the daemon is running (rps)
  - If running:
    - Check the rfopower.log file to see if it is reporting problems connecting to the internet.
    - Check the rfopower.log file to see if the daemon is reporting problems with proper operation of the adafruit.io service (request errors)
    - Check the rfopower.log file to see if the daemon is reporting problems with connecting to the Omnimeter
  - If not running:
    - Check systemd status of daemon (rstatus)
    - Check systemd journal (rj)

15.