Installing CCGX functionality on a raspberry pi

(in progress, everybody reading this page, feel free too add and improve)

The goal of this page is to run as much as possible of the CCGX functionality on a raspberry pi. Just for fun. Below steps have been done on a raspberry pi 2. Much of the source code is not yet available publicly. So to start, I am putting instructions down for all the projects that are available. And for projects that contain private Victron source code I'll.

Cross compile vs compiling locally

There are many differeny ways to make the ccgx functionality run on a rpi. For example do something with Poky/Yocto/Open embedded, our build system. But to keep it simple and also more fun, I will compile all of it compile locally on the rpi.

Optimizing make and gcc (not required!)

Below steps have all been done on my new raspberry pi 2, running this version:

```
pi@raspberrypi ~ $ cat /proc/version
Linux version 3.18.7-v7+ (dc4@dc4-XPS13-9333) (gcc version 4.8.3 20140303
(prerelease) (crosstool-NG linaro-1.13.1+bzr2650 - Linaro GCC 2014.03) )
#755 SMP PREEMPT Thu Feb 12 17:20:48 GMT 2015
pi@raspberrypi ~ $ cat /etc/issue
Raspbian GNU/Linux 7 \n \l
pi@raspberrypi ~ $ gcc --version
gcc (Debian 4.6.3-14+rpi1) 4.6.3
```

This post on the rasberry forums prompted me to try and install a newer gcc version (4.8 instead of 4.6) as well as running make with -j4, to use all 4 cores instead of only 1.

To see if there really was an improvement in compile time, I have timed compiling the git sources. Running make clean before retrying.

With the original gcc version (4.6), it took 12m26s With the original gcc version, and -j4, it took only 4m12 !

Then switched to the new compiler:

```
export CC=gcc-4.8
```

And ran make clean again, and ./configure –prefix=/usr. The output of ./configure shows that it will now use gcc-4.8. And timed the build time again:

With gcc-4.8, and -j4 it took

Steps

1. Install daemontools

Daemontools is a set of tools to run services: it runs an executable, and when it fails it will start it again. To see some commonly used commands, like starting and stopping a service, or seeing status of all services, check the CCGX command-line introduction.

To install daemontools follows the instructions on its install page, with one addition: right before running /package install, there is one change that is necessary to make it compile, without it, you'll get a message “/usr/bin/ld: errno: TLS definition in …”. To fix this, open conf-cc in ./src, and append the following parameter to the gcc call:

```
-include /usr/include/errno.h
```

Daemontools will now be running, to make sure check the processlist:

```
pi@raspberrypi ~ $ ps ax | grep svscan
 2171 ?        Ss     0:00 /bin/sh /command/svscanboot
 2173 ?        S      0:00 svscan /service
 2260 pts/0    S+     0:00 grep --color=auto svscan
```

But is not doing anything yet, as there are no services setup in /service yet.

2. Install the dbus-cli

dbus-cli is a command line interface to dbus. Very useful to play with and debug all the different CCGX services. To find what D-Bus is, read the CCGX introduction and look on Google. See CCGX commandline intro for basic usage.

First, get the code from above, and put it somewhere in the path.

Then, get some dependencies, first install pip, the python installer:

```
sudo apt-get install python-pip
```

Then install lxml, which is needed by dbus-cli:

```
sudo apt-get install libxml2-dev libxslt1-dev python-dev
sudo pip install lxml
```

Done! If all works, you'll be able to run dbus -y from the commandline:

```
pi@raspberrypi ~ $ dbus -y
org.freedesktop.DBus
```

3. Install dbus-modbustcp

dbus-modbustcp is a modbustcp server, that allows for example PLCs to get data (battery voltage, etc.) from the D-Bus.

First get the sources:

```
cd
mkdir dev
cd dev
git clone git@github.com:victronenergy/dbus_modbustcp.git
```

Install the qt libraries:

```
sudo apt-get install libqt4-dev
```

Run qmake to create the makefile (set MACHINE to ccgx to make dbus_modbustcp use the system D-Bus):

```
export MACHINE=ccgx && qmake
```

Run make to compile the sources:

```
make
```

Done! Run the binary to see that you were successful:

```
~/dev/dbus_modbustcp $ ./dbus_modbustcp
INFO 2015-02-27T21:04:03.752 dbus_modbustcp v0.6.3 started (v0.6.3)
INFO 2015-02-27T21:04:03.752 Built with Qt 4.8.2 running on 4.8.2
INFO 2015-02-27T21:04:03.753 Built on Feb 27 2015 at 21:03:50
ERROR 2015-02-27T21:04:03.784 "[Server] QTcpServer error: The address is
protected"
INFO 2015-02-27T21:04:03.787 [Mappings] Add ("com.victronenergy.vebus",
"/Ac/ActiveIn/L1/V", "d", "V AC", "3", "uint16", "10", "R")
INFO 2015-02-27T21:04:03.788 [Mappings] Add ("com.victronenergy.vebus",
"/Ac/ActiveIn/L2/V", "d", "V AC", "4", "uint16", "10", "R")
```

Note that I have not yet looked into that error!

4. Install dbus_qwacs

dbus_qwacs reads data from the Wireless AC Sensors and makes it available on D-Bus, for all the other processes.

To compile, follow the same steps as dbus_modbustcp.

When done, output will look like this:

```
~/dev/dbus_qwacs $ ./dbus_qwacs
INFO 2015-02-27T21:19:24.723 dbus_qwacs v1.0.1 started (v1.0.1)
INFO 2015-02-27T21:19:24.724 Built with Qt 4.8.2 running on 4.8.2
INFO 2015-02-27T21:19:24.724 Built on Feb 27 2015 at 21:15:59
INFO 2015-02-27T21:19:24.728 Wait for local setting on DBus...
INFO 2015-02-27T21:19:26.733 Wait...
INFO 2015-02-27T21:19:28.735 Wait...
```

And that goes on forever, since the local settings process is not yet installed, lets do that first!

5. Local settings

localsettings is a D-Bus settings manager that interfaces between xml file on disk(/data/conf/settings.xml) and D-Bus.

Steps to install (probably not as it should be done with regards to rights etc, welcome to correct this! Also it is perhaps better to make a install script for localsettings. Instead of just cloning the whole thing and adding some files, same thing: if you can improve, please do):

```
cd /opt
sudo mkdir color-control
cd ./color-control
chmod a+w .
git clone --recursive git@github.com:victronenergy/localsettings.git
cd localsettings

sudo apt-get install python-gobject
```

Create the folder where localsettings will store settings.xml

```
sudo mkdir /conf
sudo chmod a+w /conf
```

All done (except for setting this up with daemontools). Run it to see what happens:

```
./localsettings
```

DISQUS

~~DISQUS~~