Part **1** of this document is introduction to the new LLAP+ system
Part **2** of this document is a getting started guide, base around using the new LLAP Launcher

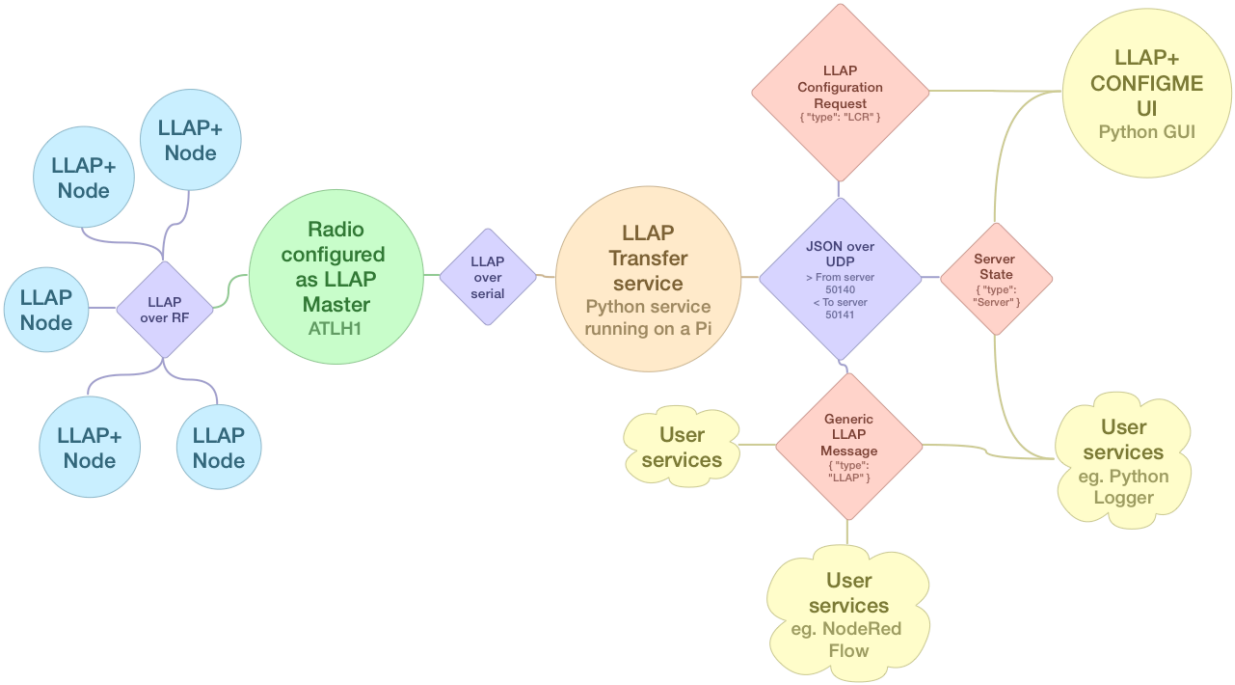
1.1 LLAP+

In an aim to improve the user experience and usability of our LLAP devices we are introducing a new range of devices called LLAP+. Our new LLAP+ devices requiring no soldering or little assembly. Our first three devices will be a temperature sensor, a light sensor and a flood sensor.

As part of our improvements to the UX we have introduced a new method for configuring and reconfiguring devices, this makes it possible to reconfigure sleeping devices, is designed to conserve battery power during the configuration mode, and adds new functionality like the querying a devices settings, the secure setting of encryption keys over the air and unique serial numbers for all devices.

In order to use this new configuration method we are defining a new system overview for use with LLAP devices. It include a Configuration GUI (**1.6**), Transfer Service (**1.5**), JSON over UDP protocol (**1.4**) , 'LLAP Master' mode for RF serial devices (**1.3**) and new extensions to the LLAP Protocol (**1.2**).

The system overview can be seen bellow, hopefully this gives a quick look into how the part all talk to each other



1.2 CONFIGME

We have extended the original LLAP protocol to include a new mode called CONFIGME. Which allows new and sleeping devices to be reconfigured in a low power friendly way.

New LLAP+ device are designed to be configured solely over the air, they have no serial connection and as such firmware updates by the user are no longer possible.

At power up a fresh LLAP+ device with no config will have an id of ?? and after sending

out its STARTED message it will go to sleep, as seen below:
a??STARTED—a??STARTED--a??STARTED--a??STARTED--a??STARTED--a??
STARTED--a??SLEEPING

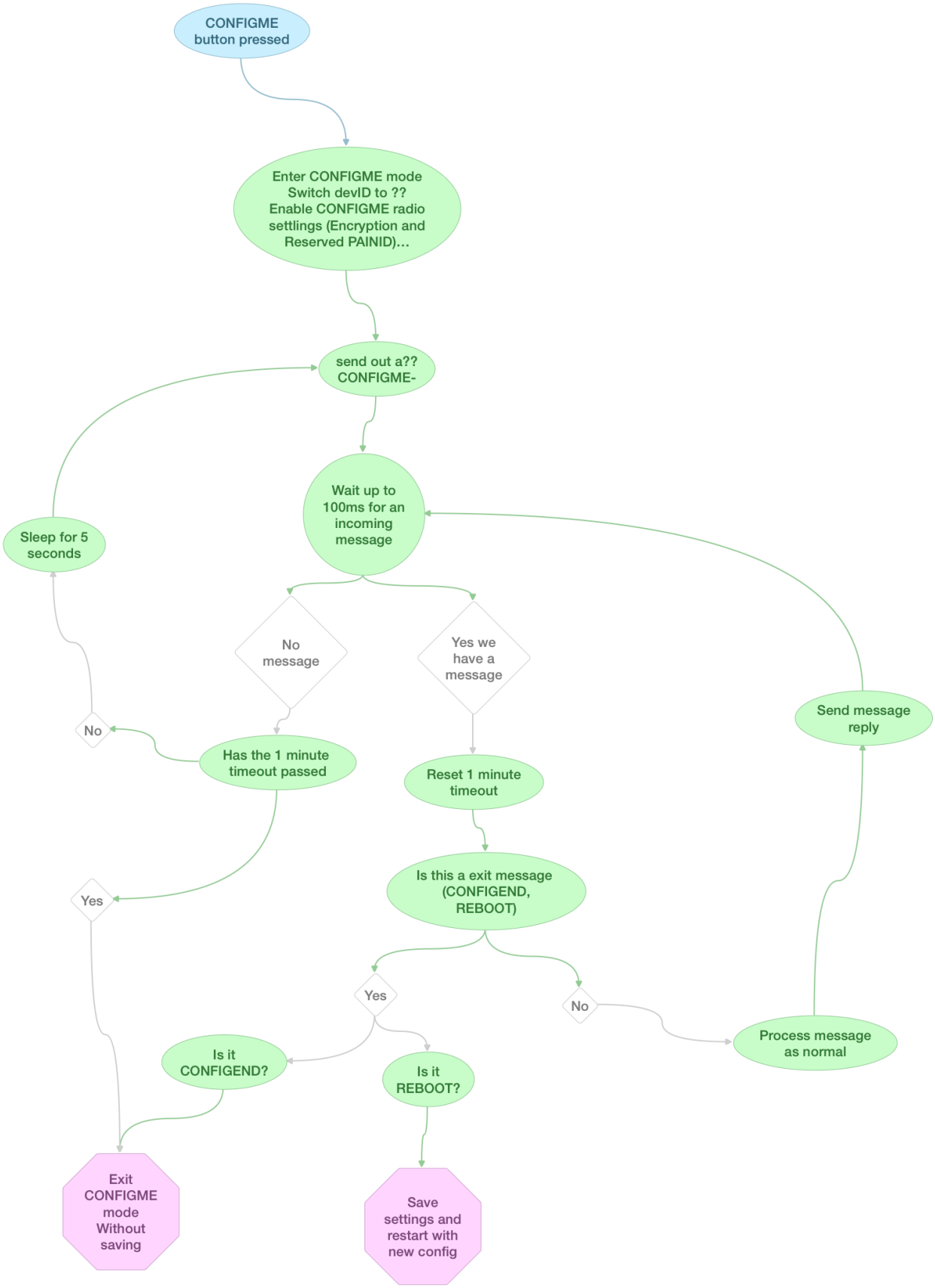
Devices ship from Ciseco may already have the battery connected in which case they will already be in a sleep state.

LLAP+ devices have a button (or similar) which is pressed to enter CONFIGME mode.
To prevent accidental use the button has a 1 second delay, i.e. it needs holding down for 1 second.
All messages in CONFIGME mode are sent using the ?? id, over a special invisible PANID and is encrypted.

Pressing the button will cause a device to enter CONFIGME mode for one minute, during this time the device will enter a loop where it will:-
wake and send a a??CONFIGME- announcement message, wait 100ms for a replay then sleep for 5s before repeating the loop.
If a command is received (as a reply to CONFIGME) within the 100ms window the commands action will be carried out and results returned, after which another 100ms wait cycle will be entered and the 1 minute timeout reset

CONFIGME mode will be exited when one of the following conditions is met:
REBOOT command issued (this will save any changes)
CONFIGEND command is issued (changes will not be saved)
1 minute time out expires (changes will not be saved)

A flow chart of how CONFIGME mode works on a device can be seen below


CONFIGME button pressed
Enter CONFIGME mode Switch devID to ?? Enable CONFIGME radio settlings (Encryption and Reserved PAINID)...
send out a?? CONFIGME-
Wait up to 100ms for an incoming message
Sleep for 5 seconds
No message
Yes we have a message
No
Has the 1 minute timeout passed
Reset 1 minute timeout
Send message reply
Yes
Is this a exit message (CONFIGEND, REBOOT)
Yes
No
Process message as normal
Is it CONFIGEND?
Is it REBOOT?
Exit CONFIGME mode Without saving
Save settings and restart with new config

1.3 LLAP Master

In order to handle the new ?? messages which are sent encrypted and over the special PANID all serial and USB devices have a new mode called LLAP Master.
In this mode the master will transparently handle messages sent or received with a ?? ID
To enable this mode on a serial or USB device use the command ATLH1 followed by ATAC

1.4 JSON over UDP

To allow our new system to work over a network we chose to use JSON over broadcast UDP.

There are three types of JSON packet that are handled by the server: LLAP, LCR, Server.
LLAP is for sending normal LLAP message back and fourth
LCR is use for sending config request back and forth (these are ?? messages)
Server is use for server status and configuration

Bellow is an example of the LLAP type JSON used to send the message aMLHELLO----

```
{
   "type":"LLAP",
   "network":"Serial",
   "id":"MA",
   "data":["TEMP"]
}
```

More examples of the JSON packet format can be found at
Documentation/JSONexamples.txt

1.5 LLAP Transfer service

LLAPServer.py is a python server that will act as a transfer service between our serial radios and a UDP network
On the serial side we talk LLAP. On the UDP network we encode/decode the LLAP messages in a JSON packet

The transfer service can be run on any system with the following requirements
Python 2.7
pySerial
Ciseco Radio with firmware Serial V0.88, USB V0.53 or later

If you are not running on a Raspberry Pi with a Slice of Radio before starting the server you may need to change the serial port setting. This can be found in the LLAPServer.cfg file in the LLAPServer directory

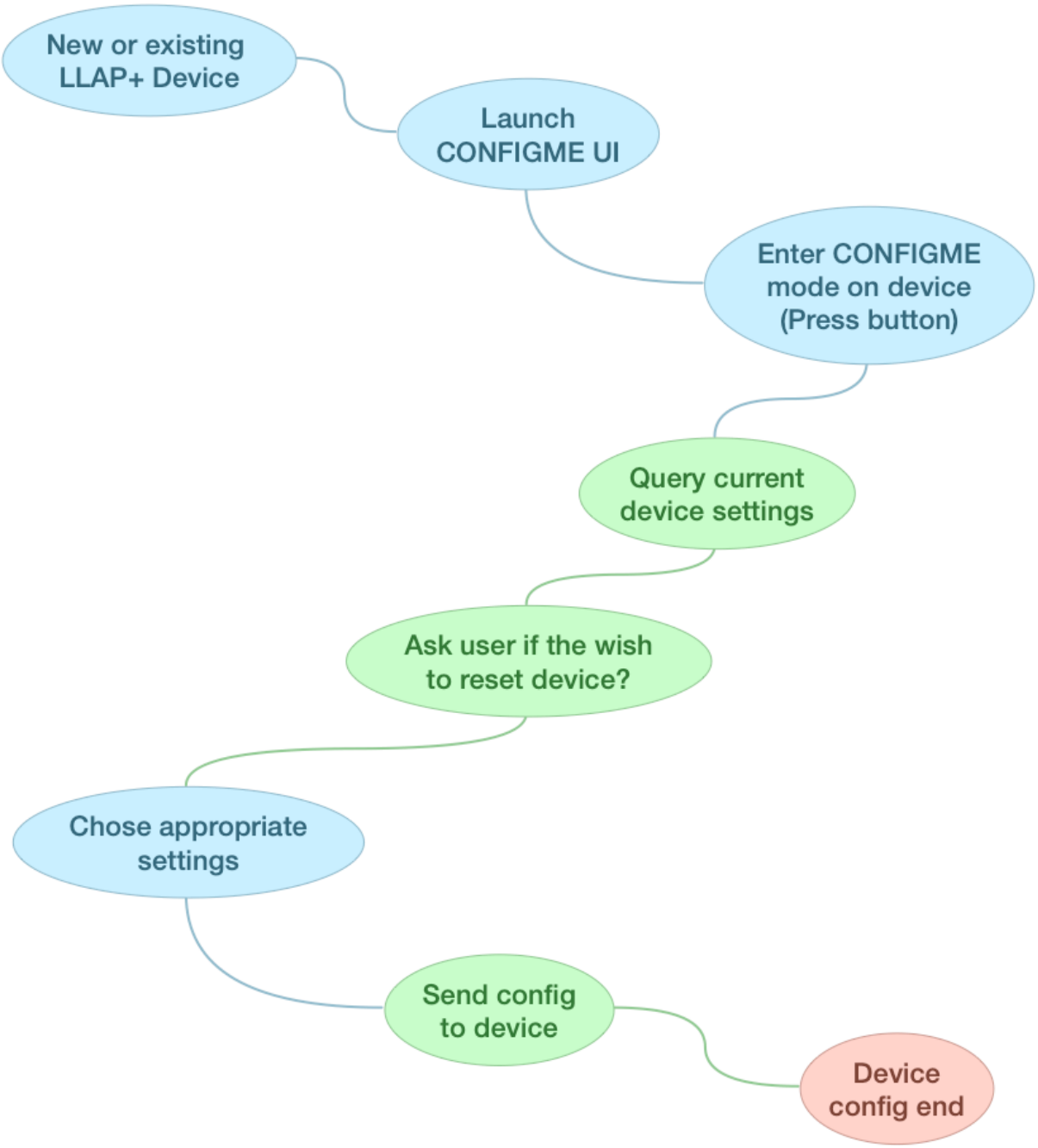
Control of the server is best done using the LLAP Launcher (**1.7**)

1.6 LLAP ConfigMe UI
LLAPConfigMe.py is a python GUI to configure devices using the new CONFIGME mode
The UI allows a user to configure or reconfigure a LLAP+ device by working through a simple wizard
The current version of the UI still requires some knowledge of our LLAP device's, there commands and optimal setup.

The UI only requires Python 2.7 and a network connection to the server
To start the UI use the following command (from inside the LLAPConfigMeUI folder)

```
$ python LLAPConfgime.py
```

The work flow for the UI can be seen in the diagram below



1.7 LLAP Launcher

The Launcher is a GUI to help with the install, running and update of the new collection of LLAP+ software. We intend to distribute all the LLAP+ software as a single downloadable zip file and for the first entry point to be the Launcher (via RunMe.py)
The launcher has the ability to start, stop and restart a LLAP Transfer service, on linux it can install the service to run on boot (windows coming soon).
It can also start the LLAP ConfigMe UI and over time more examples and user services will be added.

1.8 Examples

The Examples folder contains some simple python example of how to send and receive LLAP messages over JSON

SimpleUDPListen:
    This just dumps the raw JSON packet out to the console

CommandUDPSend:
    This takes a LLAP command as an argument and send it out via JSON UDP

SimpleUDPSend:
    This show sending out multiple LLAP message in a single JSON

2.0 Getting started

The first beta release of the ZIP file can be downloaded from
http://files.ciseco.co.uk/LLAP/LLAP_0.03.zip

Include in the zip is all the python software, examples, beta firmwares and documentation we currently have.
Built into the LLAP Launcher is an autoupdate tool that will update these as we release new zip files

2.1 Things you need

R-Pi with access to the graphical desktop (local screen or VNC)
If using a Slice of Pi + XRF or Slice of Radio you need to enable serial port access by disabling the default console (our RasWIK image has this already done)
Python 2.7 and pySerial
A default Raspbian install already has python 2.7 but you may need to install pySerial
Use the following command if needed (our RasWIK image has this already done)
$ sudo apt-get install python-serial

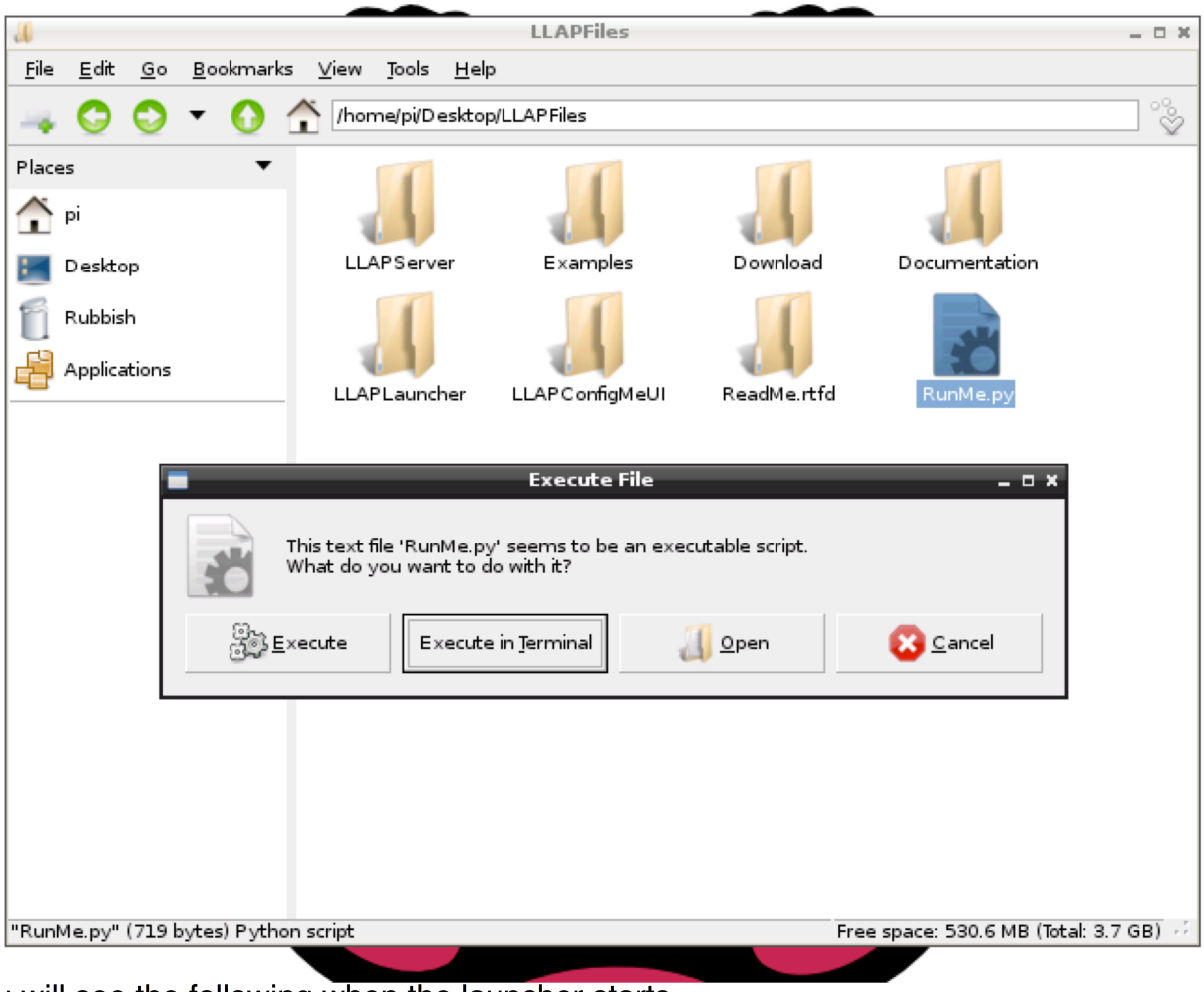
Ciseco radio to act as LLAP Master,
XRF on Slice of Pi, Slice or Radio, SRF-Stick, etc
The Radio needs to be running at least Serial 0.87 or USB 0.52
If you need to update your device the latest version of the our firmware can be found in the Firmware folder
Don't forget USB device can not update over usb and need to be done via serial
The LLAP Master mode (ATLH1) will be autocratically set when the Transfer service is launched

LLAP+ Temperature device
One of these has been given to you for testing
The device is all ready fitted with a coin cell battery.

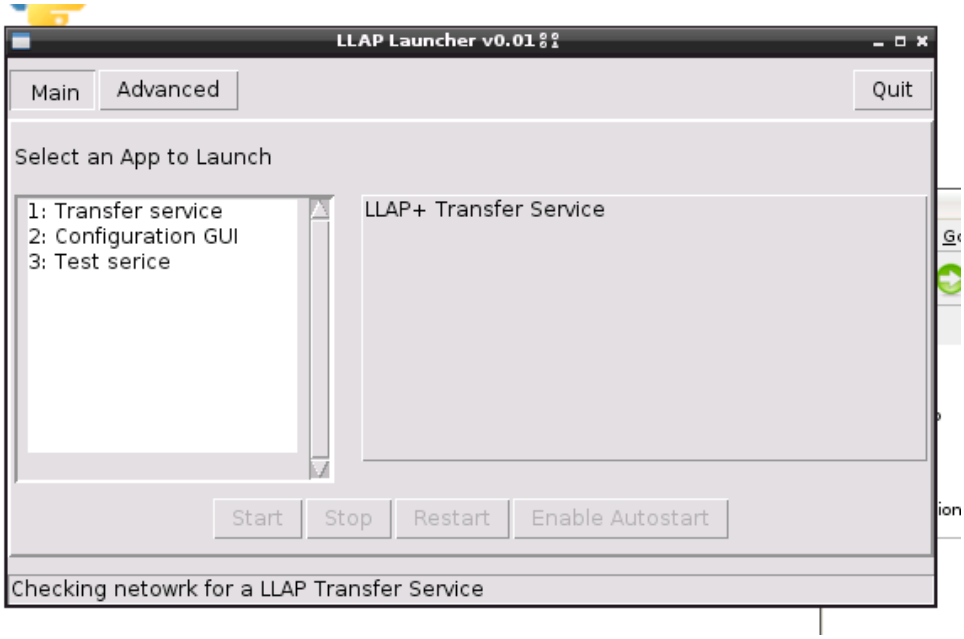
2.2 Setting up the Transfer Service

For easy of use it's recommended that you download the Zip file on the Pi desktop and extract to a folder called LLAPFiles

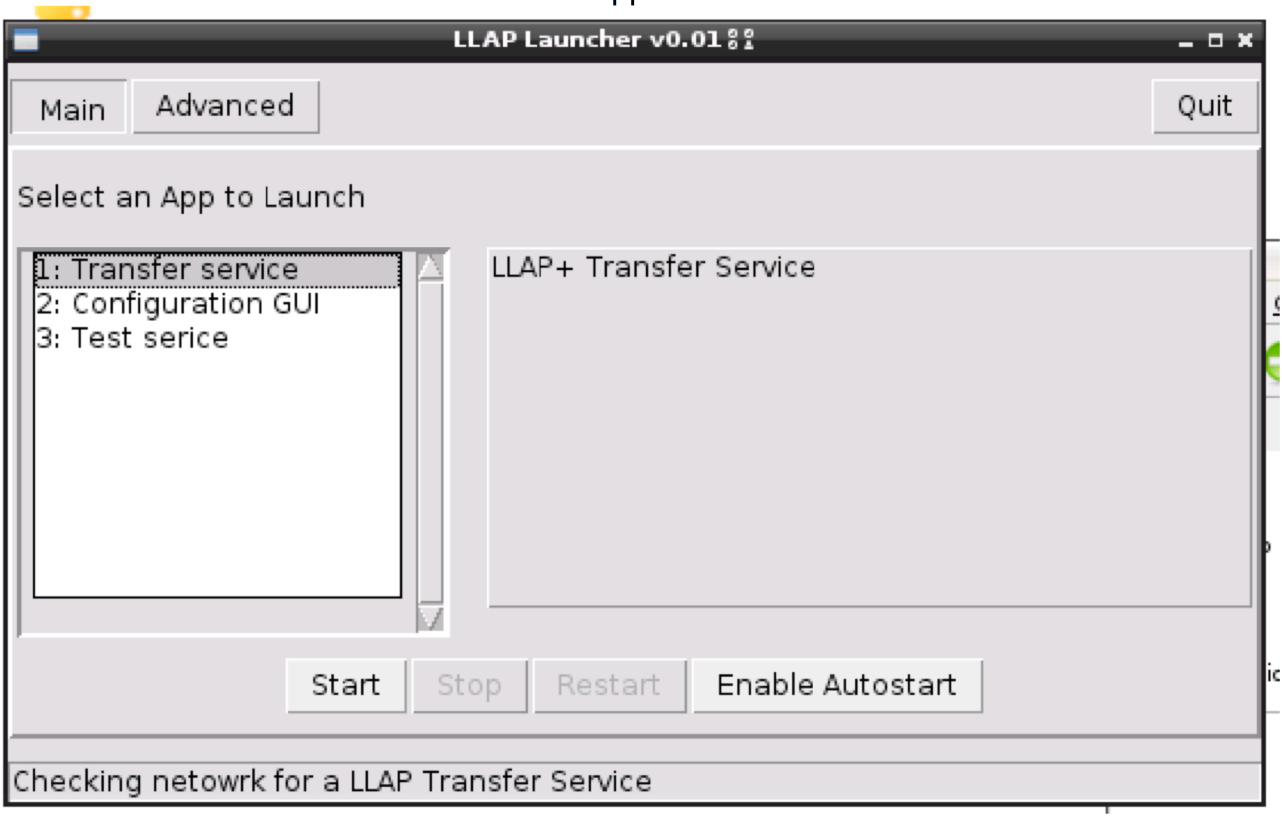
Open the LLAPFiles folder and double click RunMe.py
Click 'Execute' when prompted



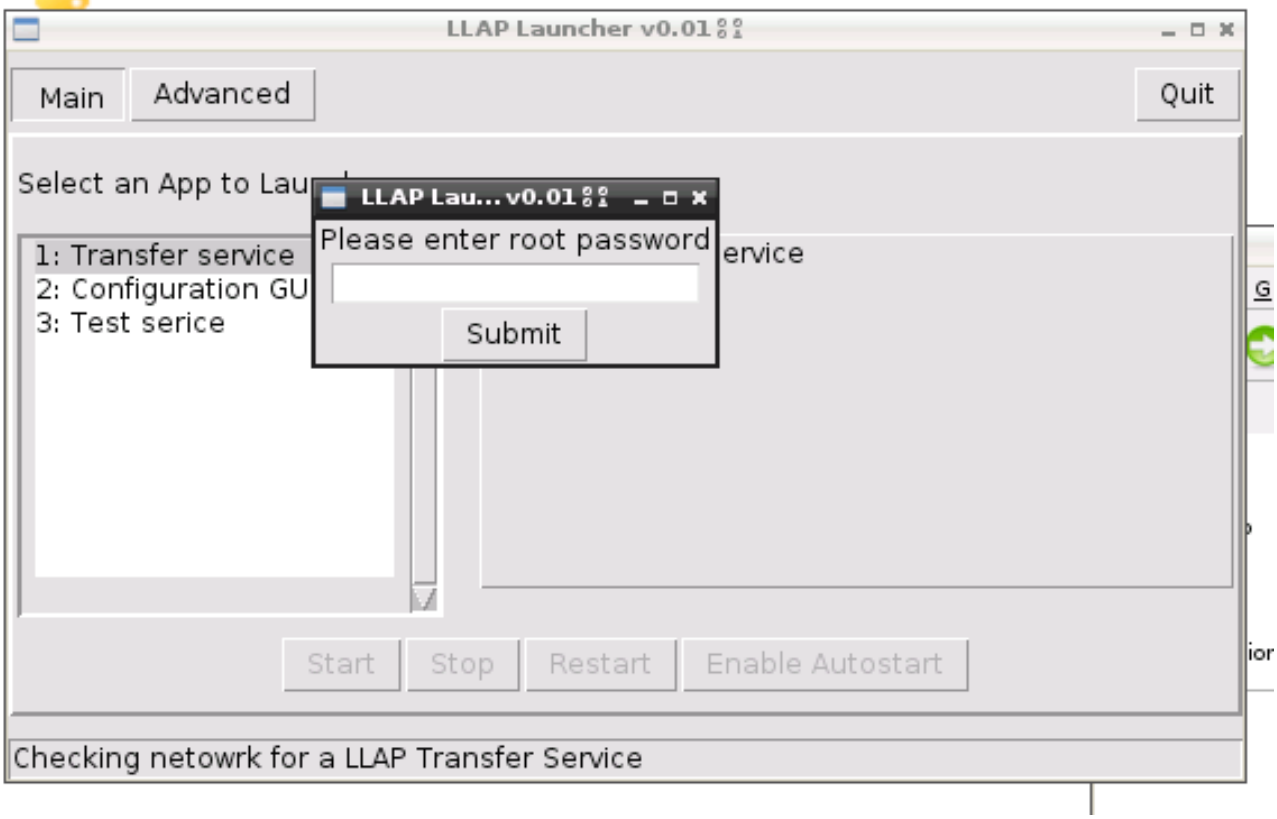
You will see the following when the launcher starts



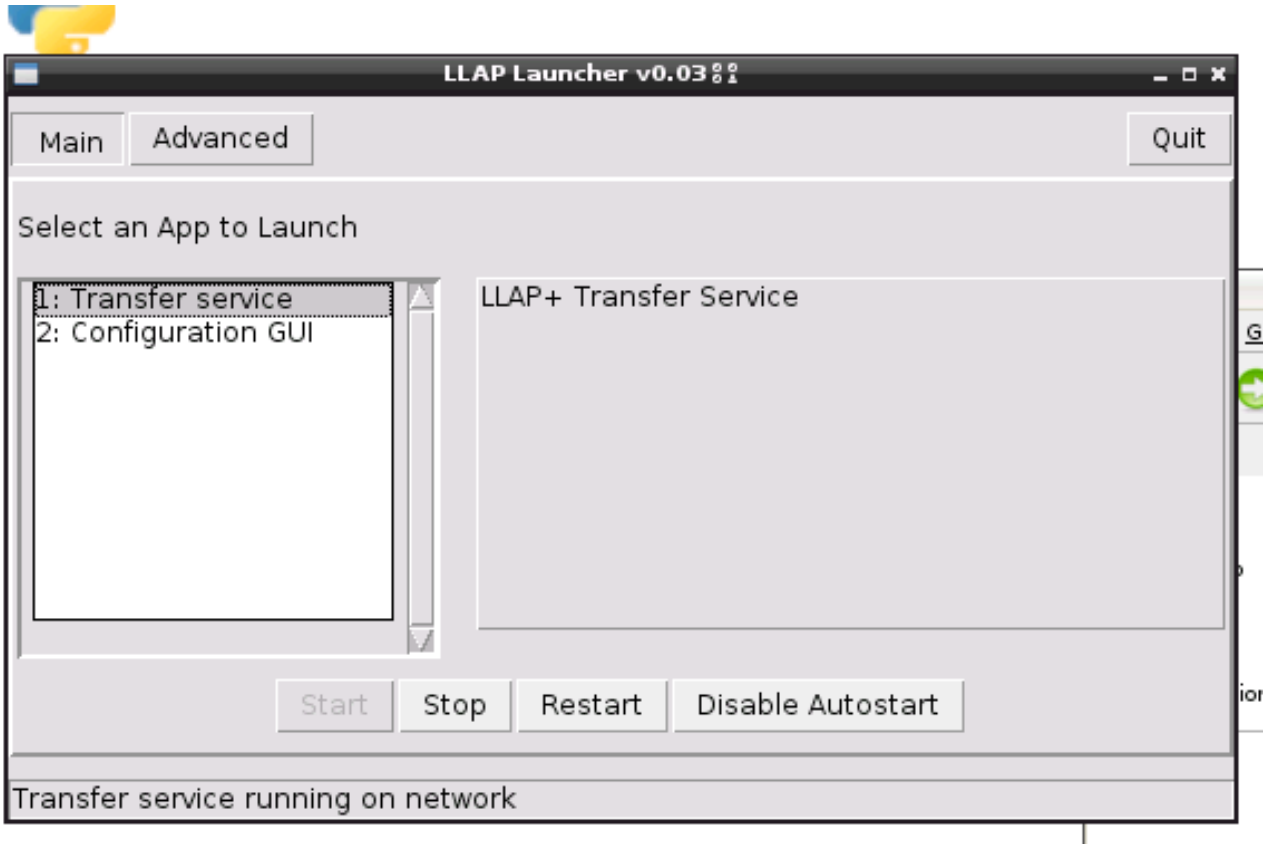
Select the Transfer service form the list of App's



To have the Services start on boot, click the “Enable Autostart” button
You will be prompted for a rood password (for default Raspbian you can just click submit)

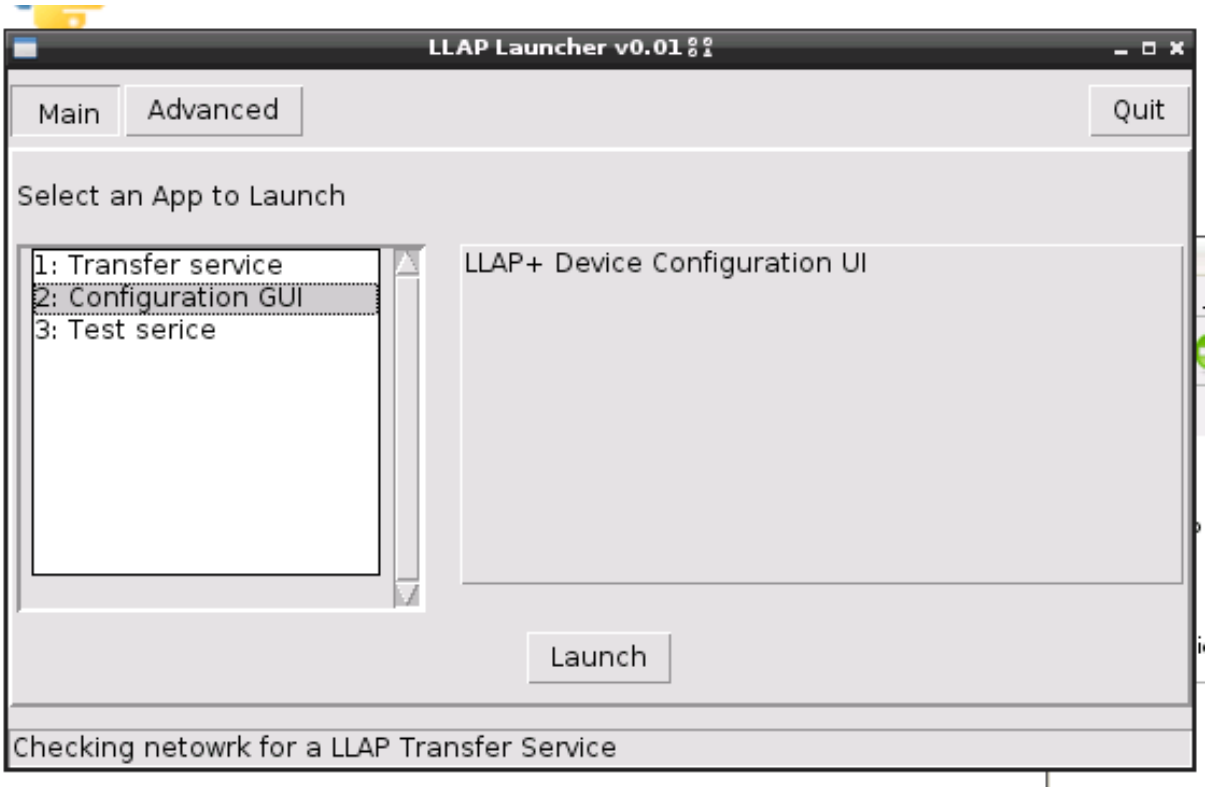


To start the service running click “Start” after a short while the buttons should change and look like the following, Note the status bar has now found a running Transfer service via the network

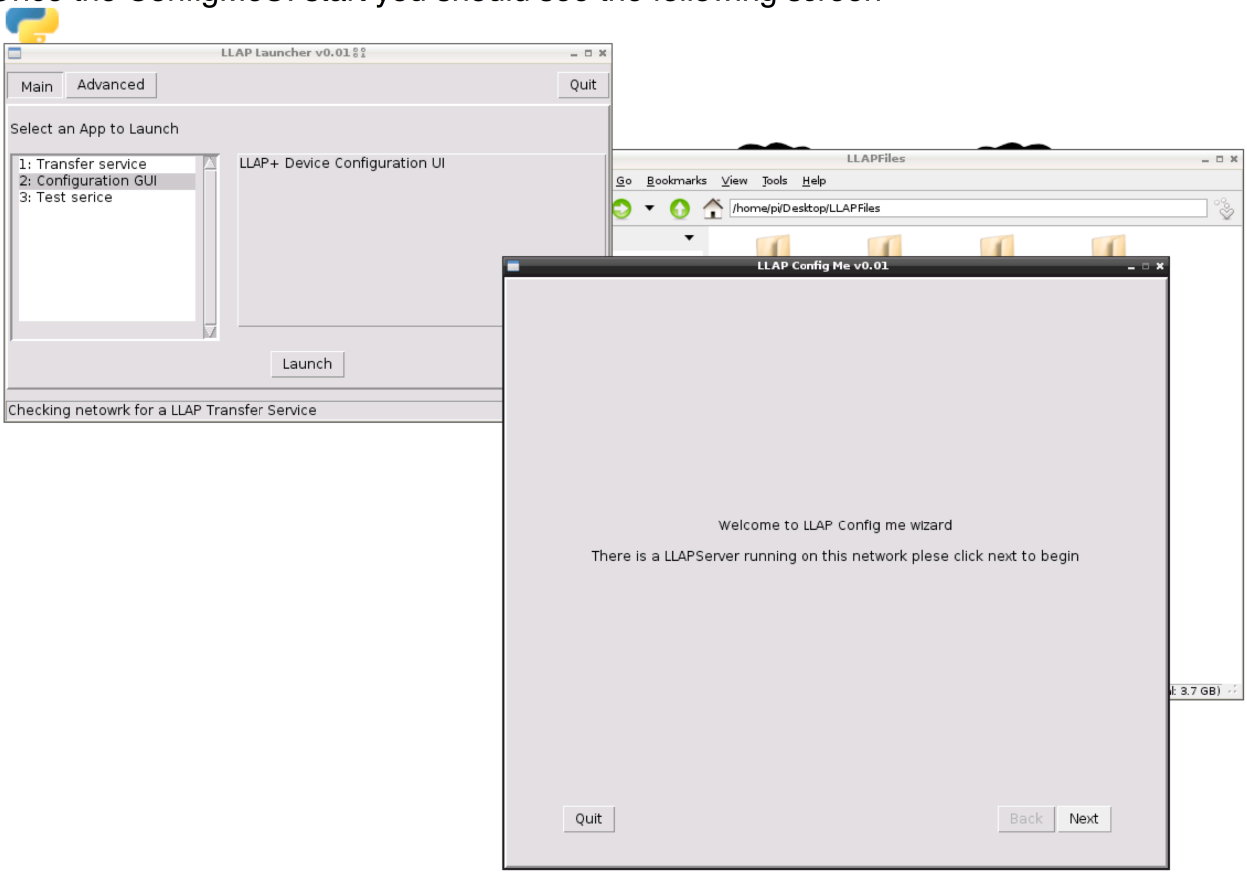


2.3 Configuring a device

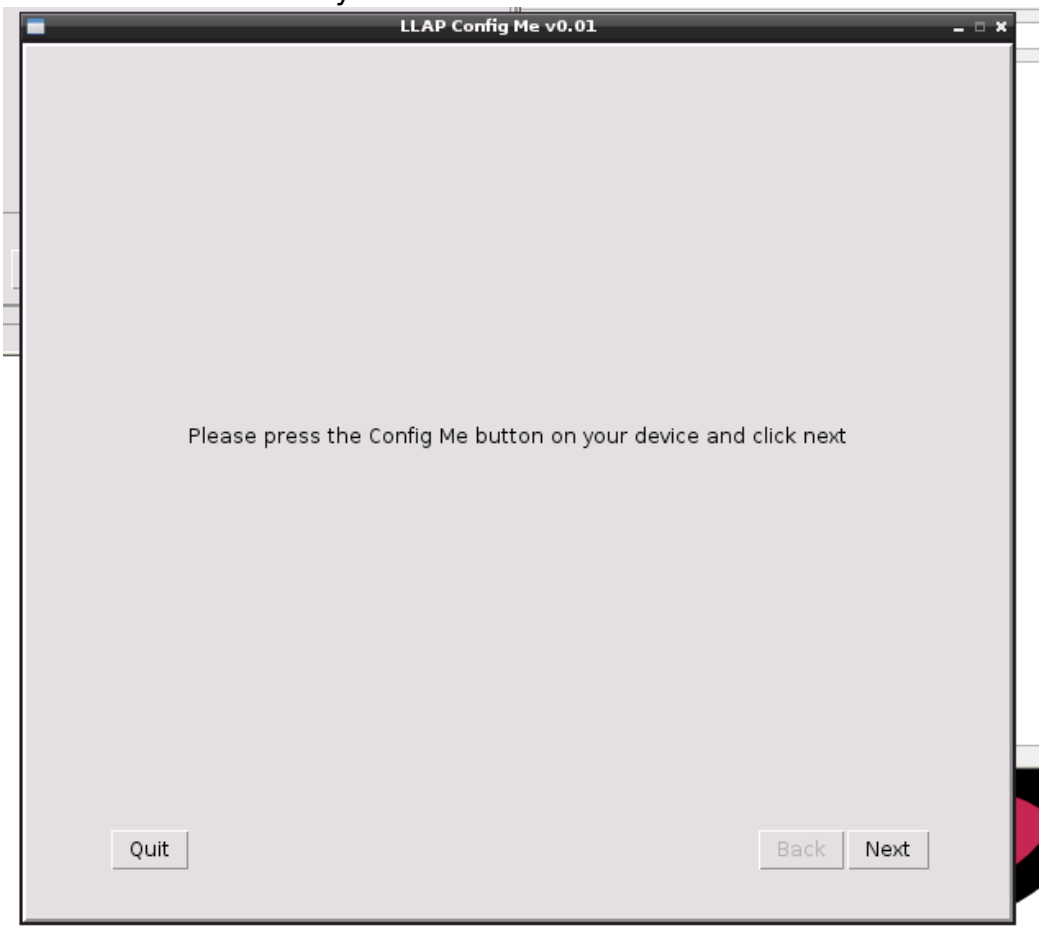
To configure a device you can to start the “Configuration GUI” form the LLAPLauncher and click launch.
(Note this can be done on the Pi the server is running on or another machine with Python installed)



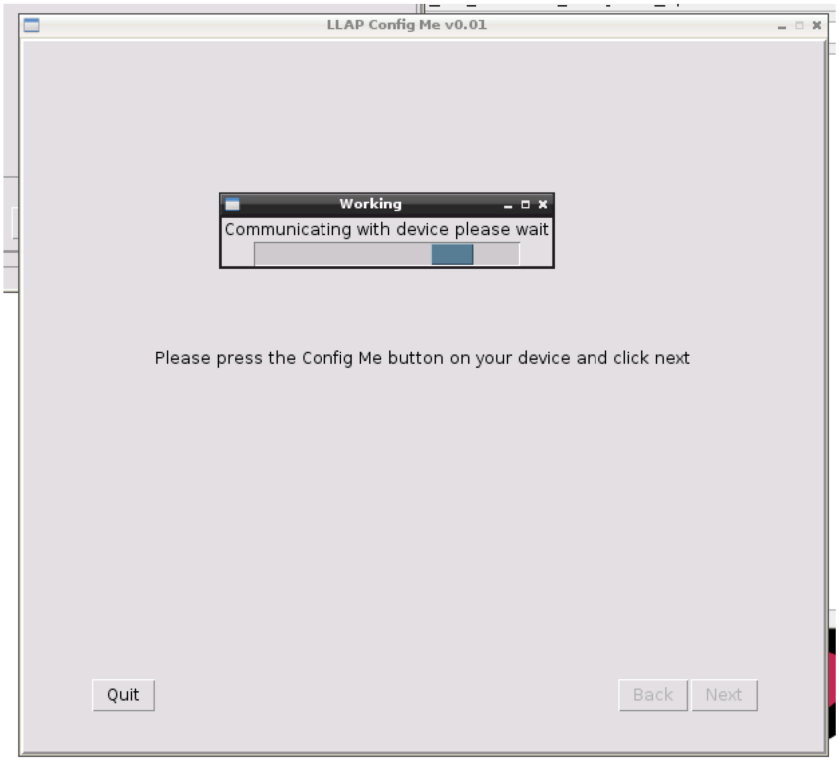
Once the ConfigMeUI start you should see the following screen



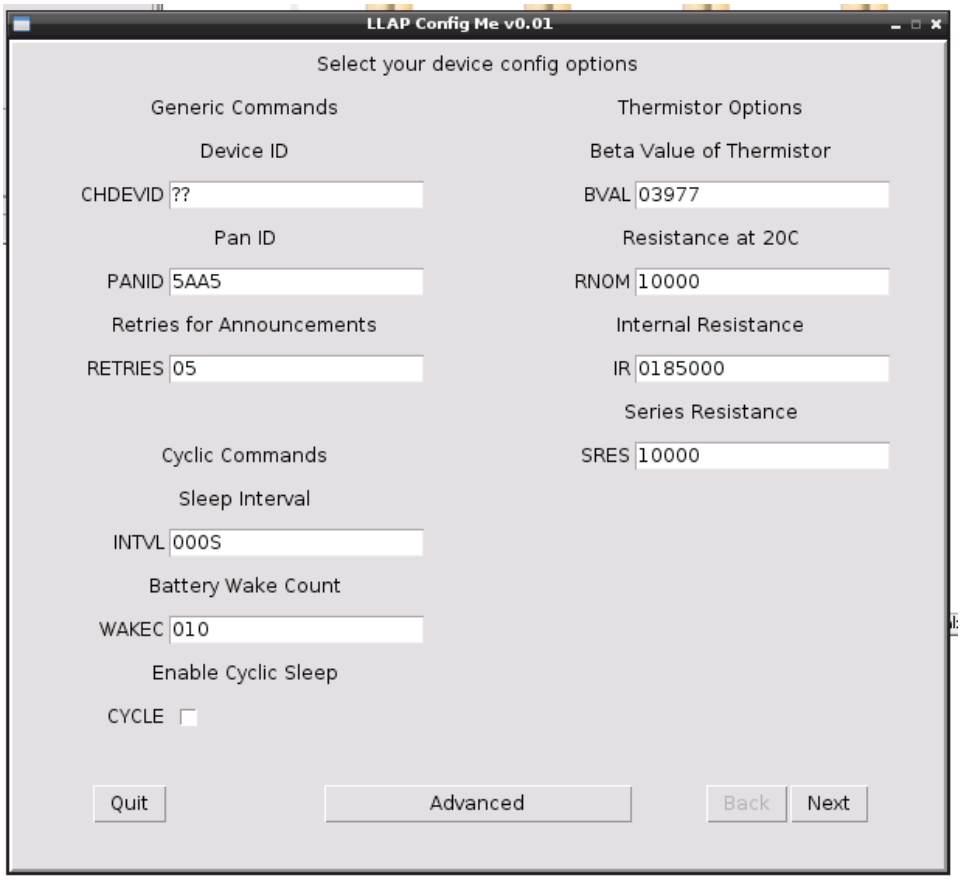
Click Next to being. You will be asked to press the configme button on the device you wish to configure and click next when you have done so



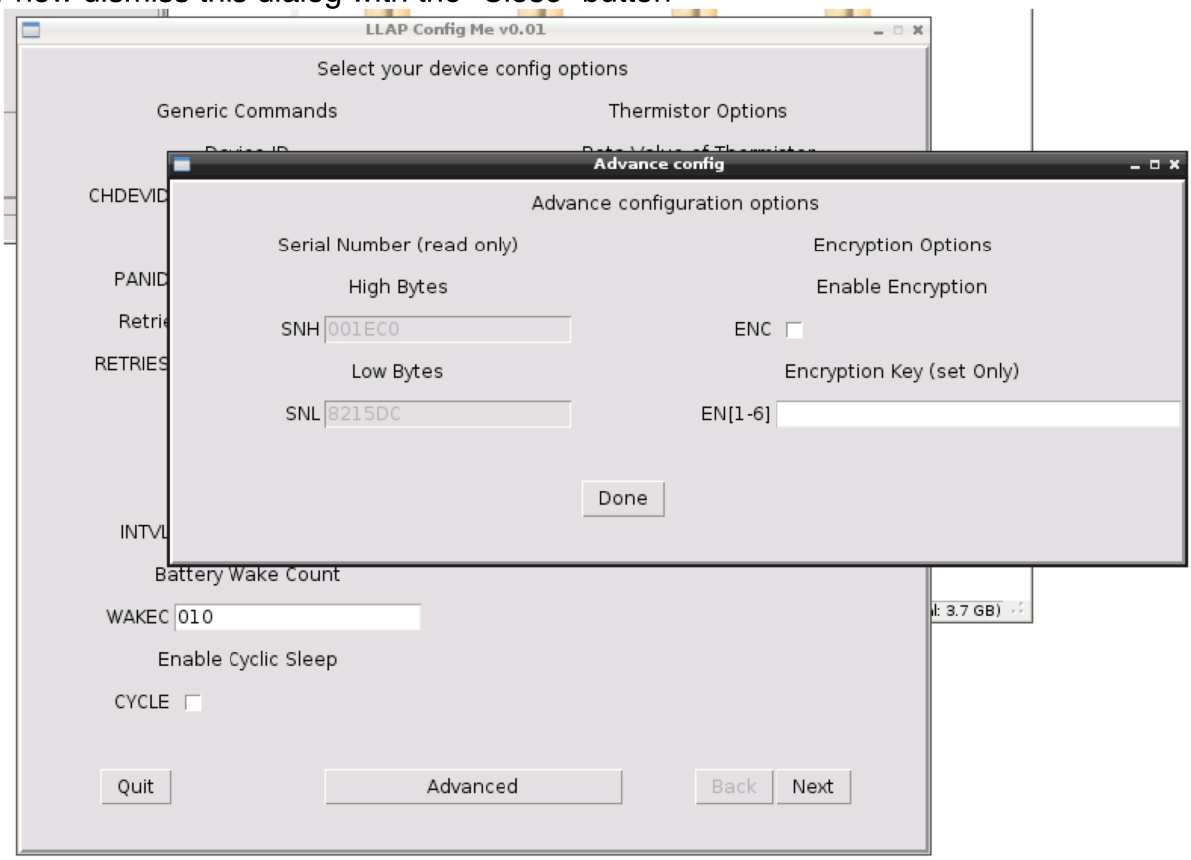
A wait dialog will appear while the UI talks to the Transfer service and get the device current settings.



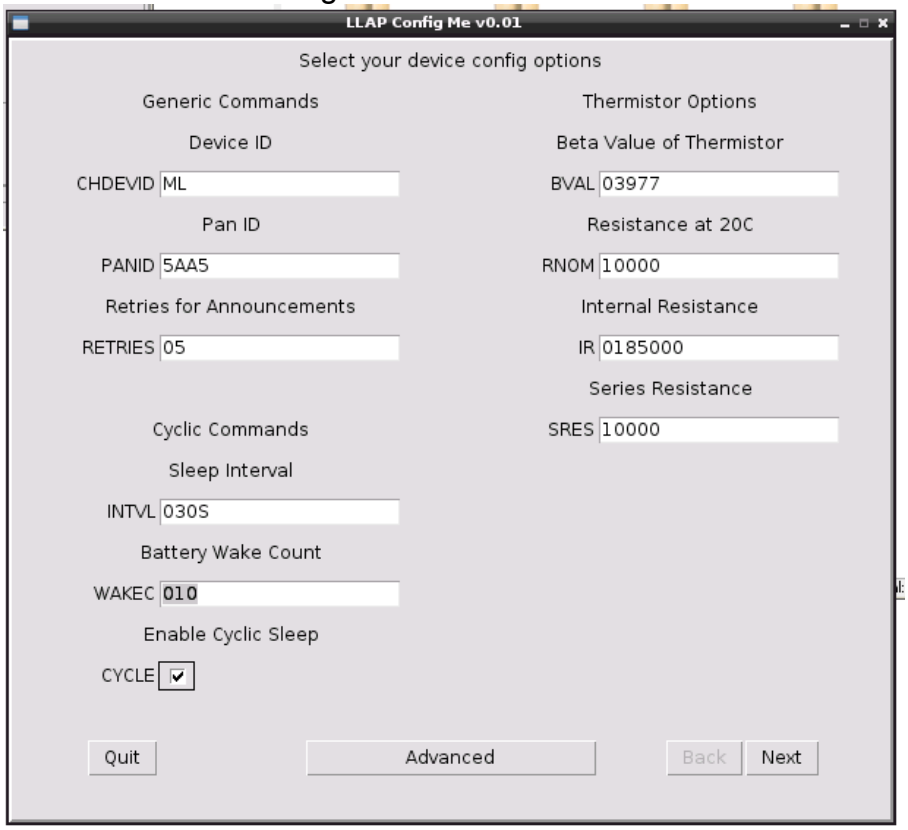
Once the setting have been retrieved from the device you should see something like the following.
Here you can see the standard LLAP commands and setting like Device ID and Sleep interval



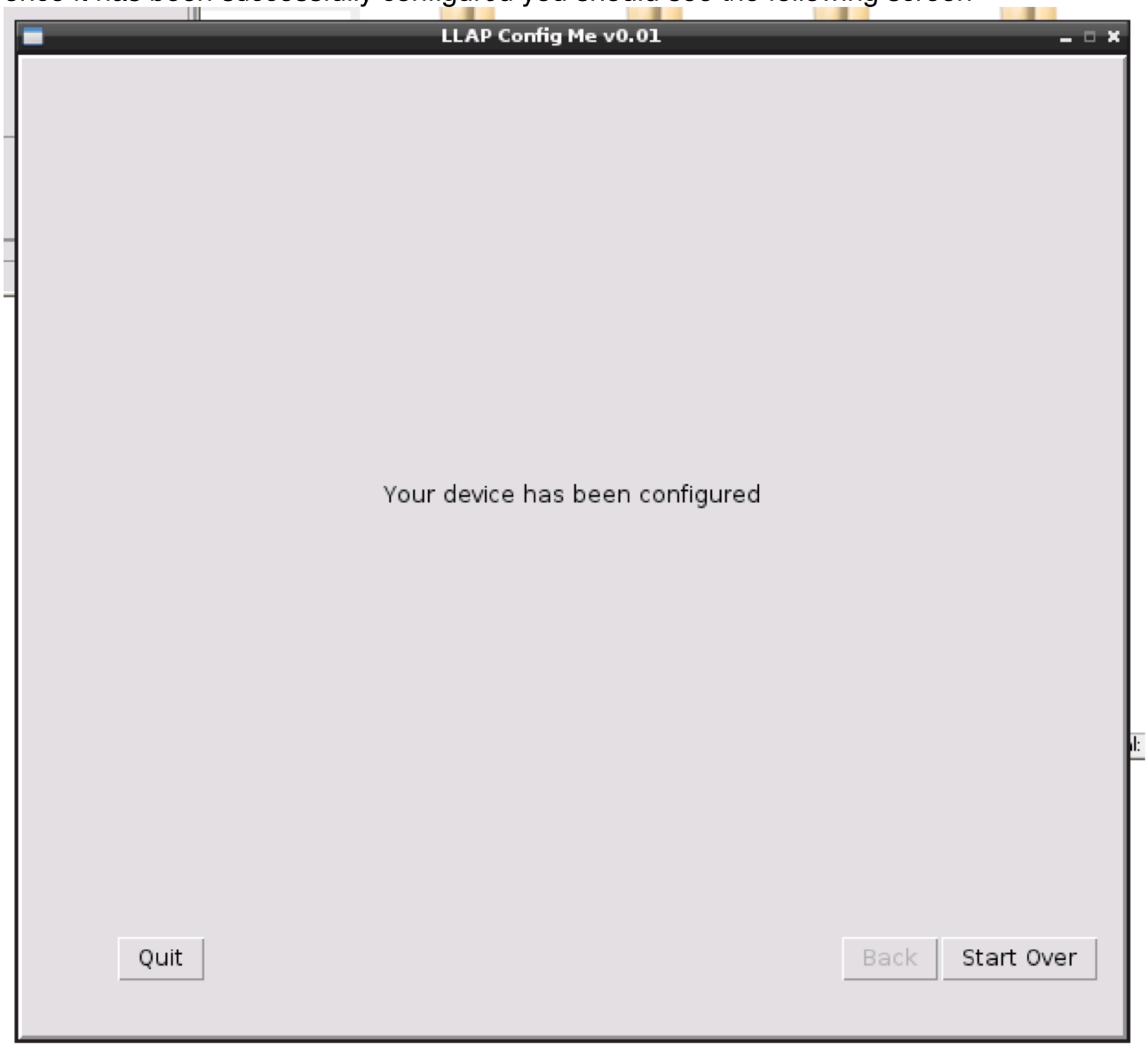
Clicking advance will bring up the following dialog box that allows you to see the devices unique serial number and to set a encryption key for your network.
For now dismiss this dialog with the “Close” button



Lets give our temperature sensor a basic config. At minimum we need to provide a Device ID, and since its a temp we will give it a sleep interval (30 seconds) and enable cyclic sleeping, Enter an ID in the **CHDEVID** box, **030S** in the **INTVL** box and tick the box for **CYCLE.** Click next to send the configuration to the device



You will see another wait dialog while the device is configured by the Transfer service, once it has been successfully configured you should see the following screen

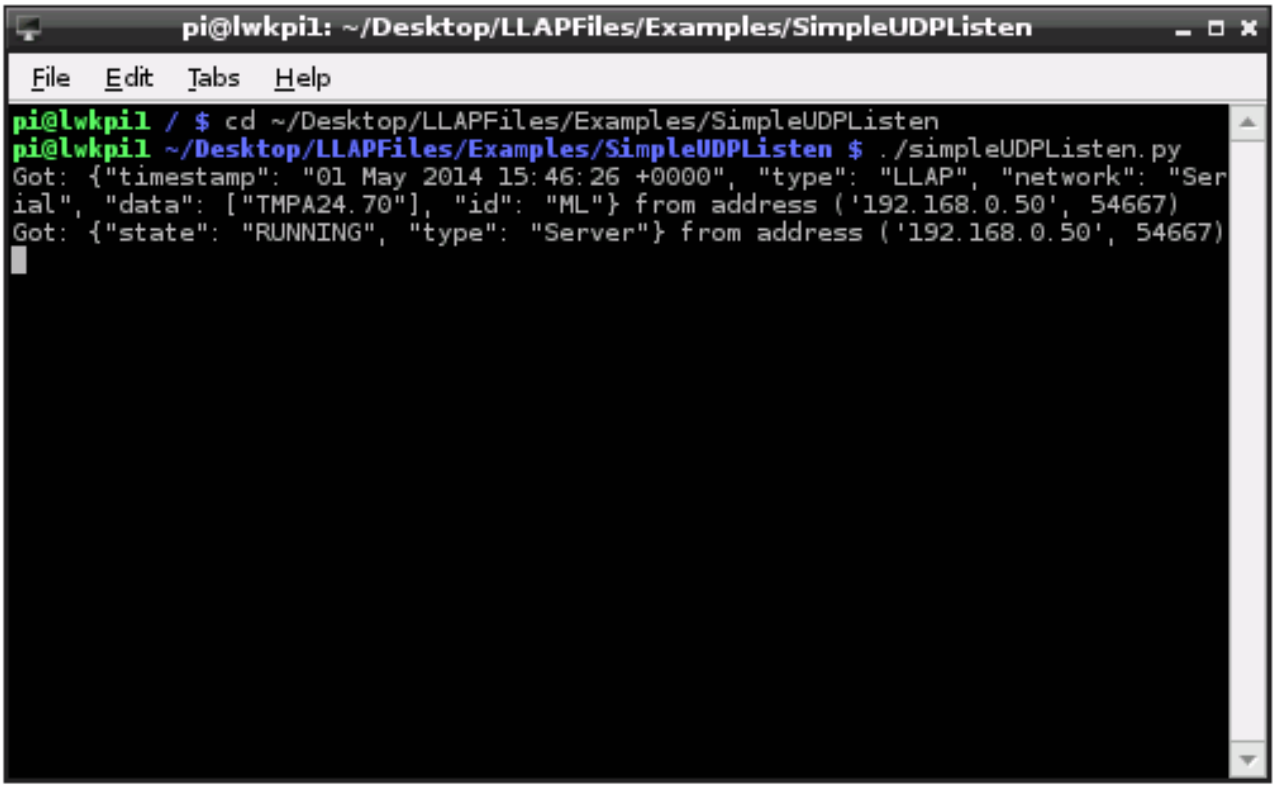


2.4 Example UDP Listen
Now we have a device sending out Temperature reading every 30 seconds we can use the example python program to see the UDP JSON message being sent out via the server.

To do this on the Pi start LXTerminal and navigate to the the
LLAPFiles/Examples/SimpleUDPListen directory
If you extracted the zip to LLAPFiles as suggested then type the following command

$ cd ~/Desktop/LLAPFiles/Examples/SimpleUDPListen

Now we can run the script with the following command and should see an output like the following
$ ./simpleUDPListen.py



You can exit the script with <crtl-C>