ICARUS

Power Monitor

Software Reference Manual

This document describes the functionalities of the board, based on Aria G25 SMD Module. On the board is installed an embedded Linux system, generated with Buildroot tool. It needs about 16 seconds to boot. To prevent micro SD corruption when switch off the board, the files system is mounted in read only mode.

To communicate with the board the power monitor server use the WebSocket protocol (RFC 6455) on port 4444. A WebSocket library for LabView is provided with a simple example to communicate with the board over Ethernet, at link: http://www.pd.infn.it/~caste/pub/WebSockets.zip. On the board is running a Web server (lighttpd) with a web page (see figure 1,2,3,4) to control and monitor the power supply, also a Secure Shell server (sshd) is running on the system.

The board control the solid state relay to switch on/off the power supply and monitor the voltage on the output of the four supplies (not the voltage on the load, also if sense to compensate cable drop voltage is connected), the current and the temperature. The board acquire about 95/105 sample per seconds of the voltage and current and store the average, min and max value for a total of 1024 points (one per second) to display history of the last 17minutes. It also store all sampled voltage on four histograms that can be used to monitor the voltage noise on the output. Note that output voltage change with sense connected if change the load current, because sense compensate the cable drop voltage.

All monitored parameter can be read without authentication, instead power on/off and fan setting can be changed only after authentication.

Getting Started

Ethernet Configuration:

By default, the Ethernet is configured with DHCP enabled, configuration can be changed by touchscreen display or using console debug connector (mini USB connector, USB to RS232. FTDI FT230X) and edit file /etc/network/interfaces. The files system is in read only mode, use command /usr/sbin/rw to switch from read only to read write mode and /usr/sbin/ro to switch back.

WARNING the Ethernet MAC address is generated with microSD serial number.

USB device connector:

The board can be used directly connected to PC via USB device connector (USB typeB), it provides a USB Ethernet Gadget configured with static IP 10.42.0.10 , configure the PC side with the same subnet address, for example 10.42.0.1. . Configuration can be changed by touchscreen display or can be changed by edit file /etc/network/interfaces, logging using console debug connector. Warning the Ethernet Gadget at boot generate random MAC address, so on the linux PC (for

example Ubuntu with Network Manager) the network must be reconfigured or you must create a special UDEV rule to automatically configure it. This problem is not present on Windows 7 OS (see Appendix A for driver installation).

Below a simple script to create udev rule for usb0 on Linux PC, run the scrip only one time and with root privilege. This script work on ubuntu12.04 and scientific Linux 6.6. If your Linux system detect the device with different name from usb0 the script must be modified accordingly.

```
#!/bin/sh
MAC=02:11:22:33:44:55
echo "[keyfile]" >>/etc/NetworkManager/NetworkManager.conf
echo "unmanaged-devices=mac:$MAC" >>/etc/NetworkManager/NetworkManager.conf
echo "#!/bin/sh" >/etc/udev/usb0.sh
echo "/sbin/ifconfig usb0 down" >>/etc/udev/usb0.sh
echo "/sbin/ifconfig usb0 hw ether $MAC" >>/etc/udev/usb0.sh
echo "/sbin/ifconfig usb0 10.42.0.1" >>/etc/udev/usb0.sh
echo "/sbin/ifconfig usb0 up" >>/etc/udev/usb0.sh
echo 'ATTRS{idVendor}=="0525" ATTRS{idProduct}=="a4a2", RUN+="/bin/sh /etc/udev/usb0.sh"'
>>/etc/udev/rules.d/70-persistent-net.rules
```

The commands accepted by server are:

Status command: return the status of the four supply. The voltage is measured on the output of the supply .

Syntax: **Status?**

Returns: the status in JSON format, example:

```
{ "arr": [{ "volt": 11.548000, "curr": 0.012023, "temp": 27.937000 }, {
"volt": 9.456476, "curr": 0.014617, "temp": 27.750000 }, { "volt": 5.028000,
"curr": 0.012695, "temp": 27.750000 }, { "volt": 3.495564, "curr": 0.023248,
"temp": 27.062000 }], "local": 0, "power": 1, "fan": 0, "time": 1245.345666}
```

Fan value: 0=high speed, 1=middle speed, 2=low speed. Value -1.00 on voltage and current or value -300.00 on temperature indicate a read error on I2C bus, temperature unit is always °C. Time is the time in seconds from processor boot up.

Version command: return the software version.

Syntax: **Version?**

Returns: **V2.4-1-gfe2b8b0**

Set Power command: set power supply on/off

Syntax: **Set:Power <0/1>**

Returns: **OK**

Set Fan command: set fan velocity

Syntax: **Set:Fan <0/1/2>**

Argument: Fan value: 0=high speed, 1=middle speed, 2=low speed.

Returns: **OK**

Authenticate command: This command is used to obtain the information necessary for the authentication handshake. The nonce value expire after 60 seconds.

Syntax: **Authenticate?**

Returns: `{ "realm": "authorized only", "nonce": "bb7a2bc19db7495606c57750f90ba775"}`

Authorization command: This command in conjunction with Authenticate command must be used to enable the connection to accept SET commands. User and password can be added or changed using console with linux command htdigest. Example:

```
~$ htdigest /etc/wspasswd "authorized only" operator
```

/etc/wspasswd is the password file, see configuration file, "authorized only" is realm string obtained by **Authenticate** command and operator is the username. By default users can authenticate with user "operator" and password "icarus".

Syntax: **Authorization:<user>:<realm>:<nonce>:<response>**

Arguments: <user> specified the username.
<realm> specified the realm string obtained by Authenticate command.
<nonce> specified the nonce string obtained by Authenticate command.
<response> must be calculate using MD5 hash by the following string:
ha1=MD5("<user>:<realm>:<password>");
response=MD5("<ha1>:<nonce>");

Returns: `OK`

Authentication handshake example with user "operator" and password "icarus":

Client `-> Authenticate?`

Server `-> {"realm": "authorized only", "nonce": "93482f2f0719e2b8ed2b5ad54f7e9150"}`

Client `-> Authorization:operator:authorized only:93482f2f0719e2b8ed2b5ad54f7e9150:d6995fa640f1ad4dafd009199422490a`

Server `-> OK`

Records command: return the history of the monitoring data. The board acquire about 95/105 samples per seconds and store the average, min and max samples every second for a total of 1024 points. The history data can be obtained in binary format, see binary commands

Syntax: **Records:Range? <time>,<maxsize>**

Arguments: <time> specified the time in seconds. First time use time=0 to obtain the first data available, then use the last time+0.1, received from a previous call, to obtain successive data.
<maxsize> is the maximum number of record to return and must be less or equals than 64 to prevent processor overload when the processor convert binary data to JSON format.

Returns: return the stored data in JSON format, example:

```
{"T0" : [[15.5,24.31],[16.5,24.31],[17.5,24.31],[18.5,24.31],[19.5,24.31]],
"T1" : [[15.5,23.94],[16.5,23.94],[17.5,23.94],[18.5,23.94],[19.5,23.94]],
"T2" : [[15.5,23.94],[16.5,23.94],[17.5,24.00],[18.5,24.00],[19.5,24.00]],
"T3" : [[15.5,24.12],[16.5,24.12],[17.5,24.19],[18.5,24.19],[19.5,24.19]],
"V0" : [[15.5,12.00],[16.5,12.00],[17.5,12.00],[18.5,12.00],[19.5,12.00]],
"V1" : [[15.5,9.00],[16.5,9.00],[17.5,9.00],[18.5,9.00],[19.5,9.00]],
"V2" : [[15.5,-5.00],[16.5,-5.00],[17.5,-5.00],[18.5,-5.00],[19.5,-5.00]],
"V3" : [[15.5,3.30],[16.5,3.30],[17.5,3.30],[18.5,3.30],[19.5,3.30]],
"I0" : [[15.5,1.00],[16.5,1.00],[17.5,1.00],[18.5,1.00],[19.5,1.00]],
"I1" : [[15.5,1.00],[16.5,1.00],[17.5,1.00],[18.5,1.00],[19.5,1.00]],
"I2" : [[15.5,1.00],[16.5,1.00],[17.5,1.00],[18.5,1.00],[19.5,1.00]],
"I3" : [[15.5,1.00],[16.5,1.00],[17.5,1.00],[18.5,1.00],[19.5,1.00]],
"V0min" : [[15.5,12.00],[16.5,12.00],[17.5,12.00],[18.5,12.00],[19.5,12.00]],
"V1min" : [[15.5,9.00],[16.5,9.00],[17.5,9.00],[18.5,9.00],[19.5,9.00]],
"V2min" : [[15.5,-5.00],[16.5,-5.00],[17.5,-5.00],[18.5,-5.00],[19.5,-5.00]],
"V3min" : [[15.5,3.30],[16.5,3.30],[17.5,3.30],[18.5,3.30],[19.5,3.30]],
"I0min" : [[15.5,1.00],[16.5,1.00],[17.5,1.00],[18.5,1.00],[19.5,1.00]],
"I1min" : [[15.5,1.00],[16.5,1.00],[17.5,1.00],[18.5,1.00],[19.5,1.00]],
"I2min" : [[15.5,1.00],[16.5,1.00],[17.5,1.00],[18.5,1.00],[19.5,1.00]],
"I3min" : [[15.5,1.00],[16.5,1.00],[17.5,1.00],[18.5,1.00],[19.5,1.00]],
"V0max" : [[15.5,12.00],[16.5,12.00],[17.5,12.00],[18.5,12.00],[19.5,12.00]],
"V1max" : [[15.5,9.00],[16.5,9.00],[17.5,9.00],[18.5,9.00],[19.5,9.00]],
"V2max" : [[15.5,-5.00],[16.5,-5.00],[17.5,-5.00],[18.5,-5.00],[19.5,-5.00]],
"V3max" : [[15.5,3.30],[16.5,3.30],[17.5,3.30],[18.5,3.30],[19.5,3.30]],
"I0max" : [[15.5,1.00],[16.5,1.00],[17.5,1.00],[18.5,1.00],[19.5,1.00]],
"I1max" : [[15.5,1.00],[16.5,1.00],[17.5,1.00],[18.5,1.00],[19.5,1.00]],
"I2max" : [[15.5,1.00],[16.5,1.00],[17.5,1.00],[18.5,1.00],[19.5,1.00]],
"I3max" : [[15.5,1.00],[16.5,1.00],[17.5,1.00],[18.5,1.00],[19.5,1.00]],
"maxrecord": 1024}
```


Value -1.00 on voltage and current or value -300.00 on temperature indicate a read error on I2C bus, temperature unit is always °C. To convert the time in local time, read the current time of the processor by command “Status” and as soon as receive the return data by the command, read the PC local time and calculate the offset=PC_local_time - processor_current_time , then local_time=time+offset.

Histograms command: return the histograms of the accumulated voltage sample , about 95/105 sample per seconds. In the histograms are accumulated also the values during power on/off. Use Clear Histograms command to perform noise measure in a specific elapsed time.

Syntax: **Histograms?**

Returns: the noise histograms in JSON format, example:


```
{"clrtime": 3240.671261,"time": 4054.382413,
"histogs":[[[12.21,11922],[12.22,41975]],[[8.19,43740],[8.20,10157]],[[-
5.20,53097],[-5.21,800]],[[3.31,53793],[3.32,104]]]}.
```


To convert the `clrtime` and `time` in local time read the PC local time as soon as receive the return data, then calculate the offset=PC_local_time - time , then clear_local_time=clrtime+offset and local_time=time+offset .

Clear Histograms command: clear all histograms.

Syntax: **ClrHistograms**

Returns: **OK**

App Log command: return the application log.

Syntax: **Log?**

Returns: the application log text.

Returns Error message Syntax: **ERROR:<number>,<message>**

Binary commands accepted by server are:

History data, Command 0x01: return the history of the monitoring data. The board acquire about 95/105 samples per seconds and store the average, min and max samples every second.

Syntax: 0x01,[<double time>]

Arguments: <time> optional argument, specified the time in seconds. First time use time=0 to obtain the first data available, then use the last time+0.1, received from a previous call, to obtain successive data.

Returns: 0x02, <uint32 Totalsize>,

<uint32 Namesize>,<Cstring>,

<uint32 Datasize>,<Data>,

<uint32 Namesize>,<Cstring>,

<uint32 Datasize>,<Data>,

…

…

<uint32 Maxrecord>

Totalsize: is the size in byte of data transferred, tag 0x02 included, in little Endian format.

Namesize: is the size in byte of the string name that identified the data, end string included (char=0) , in little Endian format.

Cstring: is the byte array contained the name of data .

Datasize: is the data size in byte.

Data: is the vector array. Vector_0 [x, y],Vector_1[x,y]…..

x and y are in double float precision (8 byte) and little Endian format.

The x value is the time in second from processor boot.

Maxrecord: is the maximum number of vector return in data.

The array name returned are: "T0", "T1", "T2", "T3", "V0", "V1", "V2", "V3", "I0", "I1", "I2", "I3","V0min", "V1min", "V2min", "V3min", "I0min", "I1min", "I2min", "I3min", "V0max", "V1max", "V2max", "V3max", "I0max", "I1max", "I2max","I3max".

Figure 1

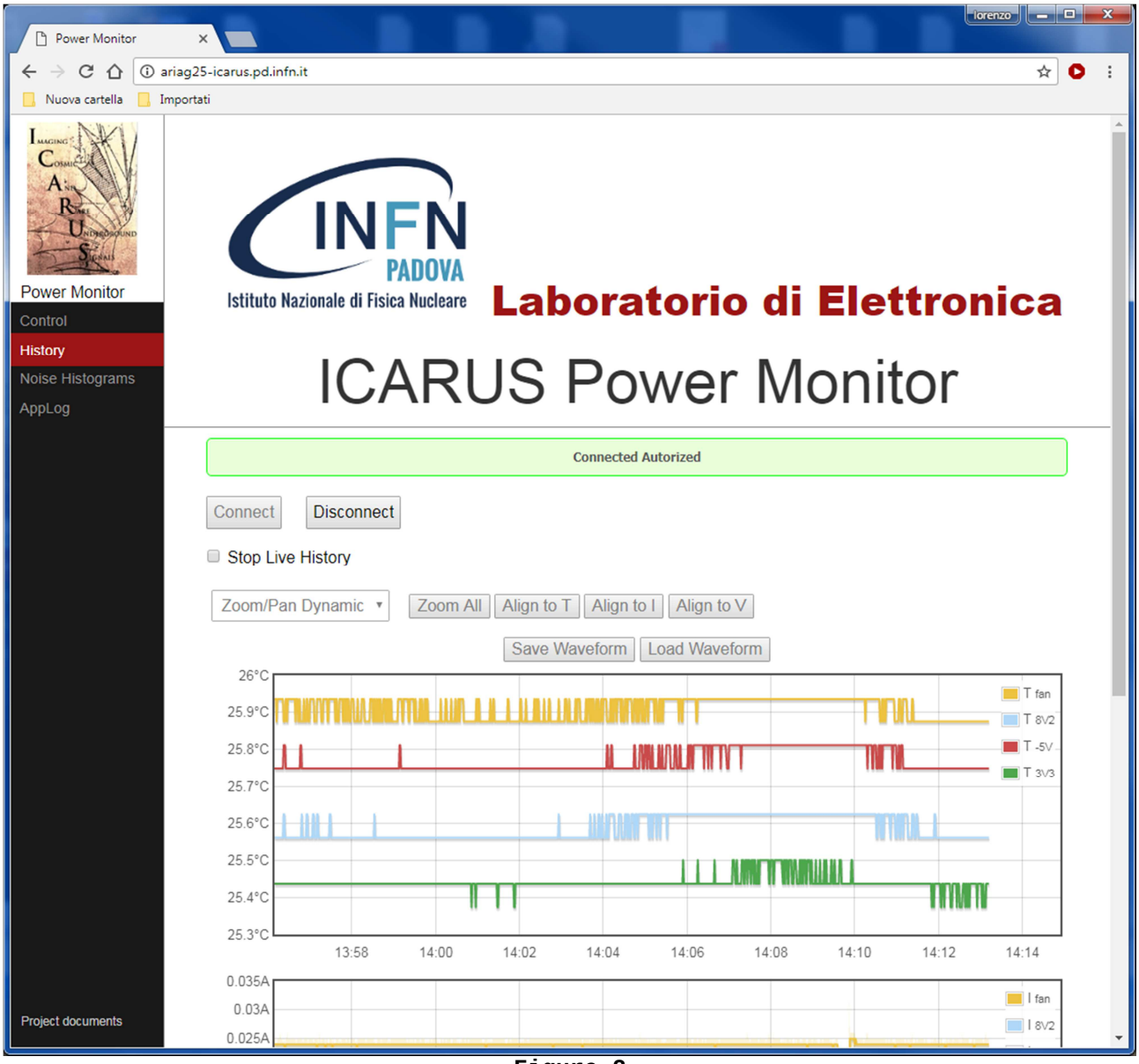

Power Monitor
ariag25-icarus.pd.infn.it
Nuova cartella
Importati
Power Monitor
Control
History
Noise Histograms
AppLog
Project documents
INFN
PADOVA
Istituto Nazionale di Fisica Nucleare
Laboratorio di Elettronica
ICARUS Power Monitor
Connected Autorized
Connect
Disconnect
Stop Live History
Zoom/Pan Dynamic
Zoom All
Align to T
Align to I
Align to V
Save Waveform
Load Waveform
26°C
25.9°C
25.8°C
25.7°C
25.6°C
25.5°C
25.4°C
25.3°C
13:58
14:00
14:02
14:04
14:06
14:08
14:10
14:12
14:14
T fan
T 8V2
T -5V
T 3V3
0.035A
0.03A
0.025A
I fan
I 8V2

Figure 2

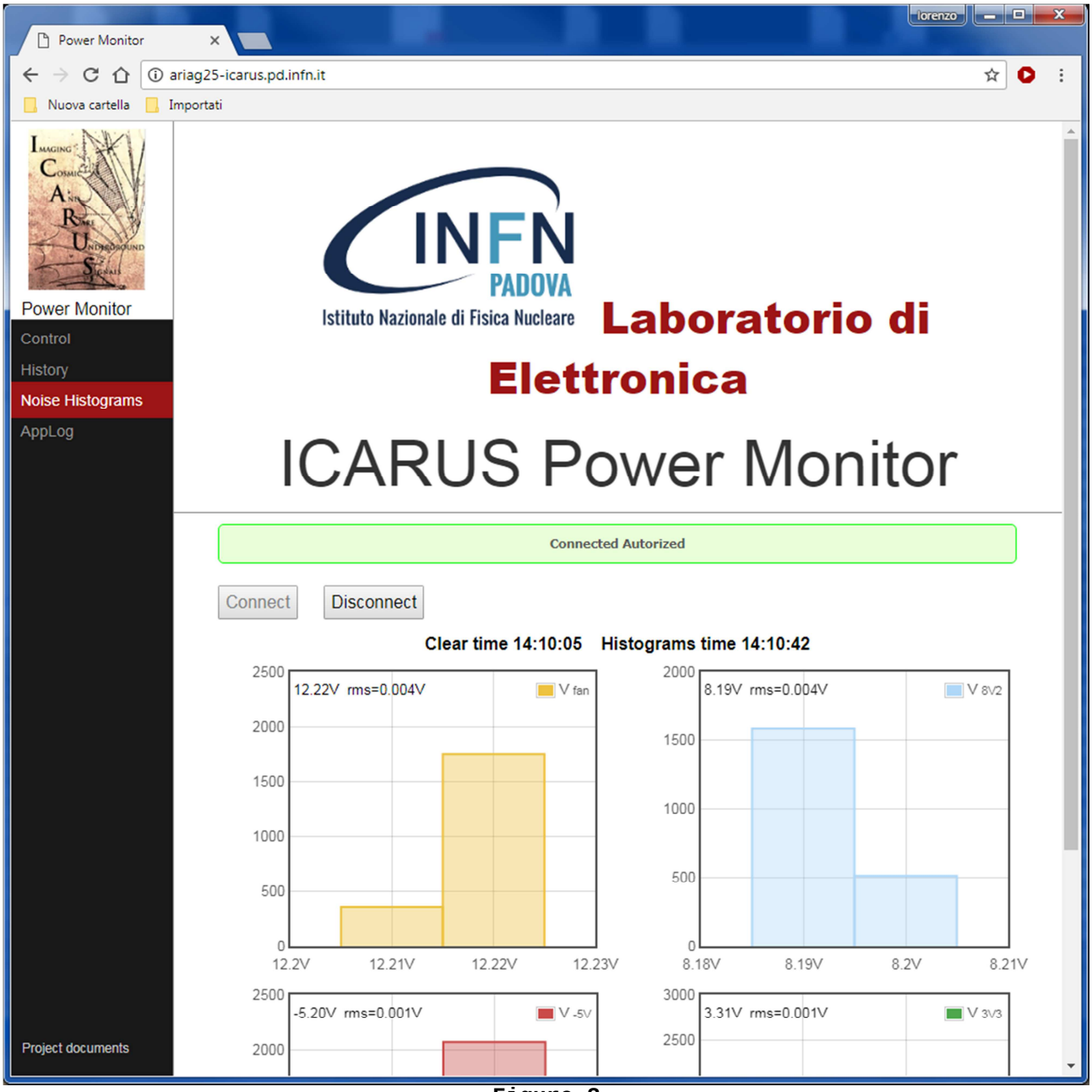

Power Monitor
ariag25-icarus.pd.infn.it
Nuova cartella
Importati
Power Monitor
Control
History
Noise Histograms
AppLog
Project documents
INFN
PADOVA
Istituto Nazionale di Fisica Nucleare
Laboratorio di Elettronica
ICARUS Power Monitor
Connected Autorized
Connect
Disconnect
Clear time 14:10:05 Histograms time 14:10:42
12.22V rms=0.004V
V fan
8.19V rms=0.004V
V 8V2
-5.20V rms=0.001V
V -5V
3.31V rms=0.001V
V 3V3

Figure 3

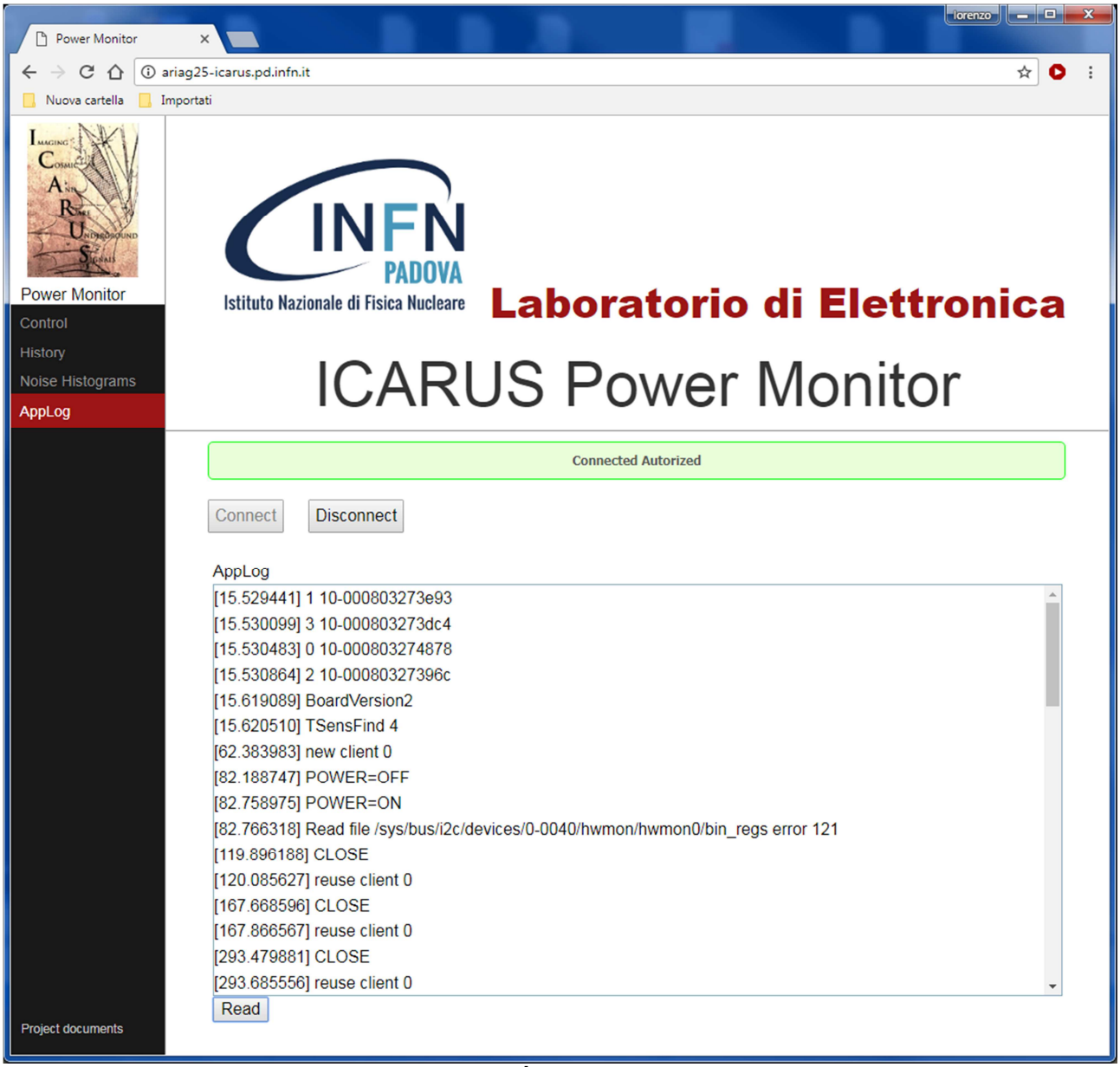

Power Monitor
ariag25-icarus.pd.infn.it
Nuova cartella
Importati
Power Monitor
Control
History
Noise Histograms
AppLog
Project documents
INFN
PADOVA
Istituto Nazionale di Fisica Nucleare
Laboratorio di Elettronica
ICARUS Power Monitor
Connected Autorized
Connect
Disconnect
AppLog
[15.529441] 1 10-000803273e93
[15.530099] 3 10-000803273dc4
[15.530483] 0 10-000803274878
[15.530864] 2 10-00080327396c
[15.619089] BoardVersion2
[15.620510] TSensFind 4
[62.383983] new client 0
[82.188747] POWER=OFF
[82.758975] POWER=ON
[82.766318] Read file /sys/bus/i2c/devices/0-0040/hwmon/hwmon0/bin_regs error 121
[119.896188] CLOSE
[120.085627] reuse client 0
[167.668596] CLOSE
[167.866567] reuse client 0
[293.479881] CLOSE
[293.685556] reuse client 0
Read

Figure 4

Configuration file /etc/IcaPwrMon.conf:

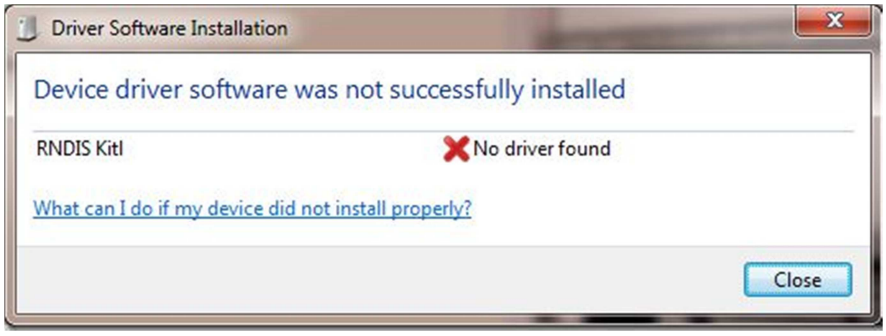
```
#
# Temperature sensors assignment
# syntax:
# alias <temperature ID> <Power Supply ID>
#

alias 10-0008006e80d5   1
alias 10-00080048a21e   3
alias 10-0008006ea336   0
alias 10-0008006ebc6e   2

#
# Temperature unit
# C = Celsius
# F = Fahrenheit
T-Unit C

#
# BoarVersion
#
BoardVersion   2

#
# Password file
#
passwordfile /etc/wspasswd

#
# Current Calibration
# syntax:
# Imon<Power Supply ID> <m> <q>
#
# I=Iread*m+q
#
Imon0 0.99586 -0.02582
Imon1 0.988377 -0.010092
Imon2 0.98981 -0.01502
Imon3 0.93496 0.002679

#
# Voltage Calibration
# syntax:
# Vmon<Power Supply ID> <m> <q>
#
# V=Vread - (I*m+q)
# (I*m+q) is shunt partition (about shunt/3)
#
Vmon1 0.008839 0.007395
Vmon3 0.01766 0.004025

#
# Noise monitor
# syntax:
# Monitor<Power Supply ID>  <flash time ms> <threshold>
#

Monitor0 3000 0.1
Monitor1 3000 0.1
Monitor2 3000 0.1
Monitor3 3000 0.1
```

```
#
# SaveScreen
# SaveScreenTime unit ms
#
SaveScreenEn   false
SaveScreenTime 300000
```

Appendix A

Windows 7 RNDIS driver installation

1. After the device is connected to the PC, OS will automatically search for the RNDIS driver. After it fails to find the driver, the following message will be shown.

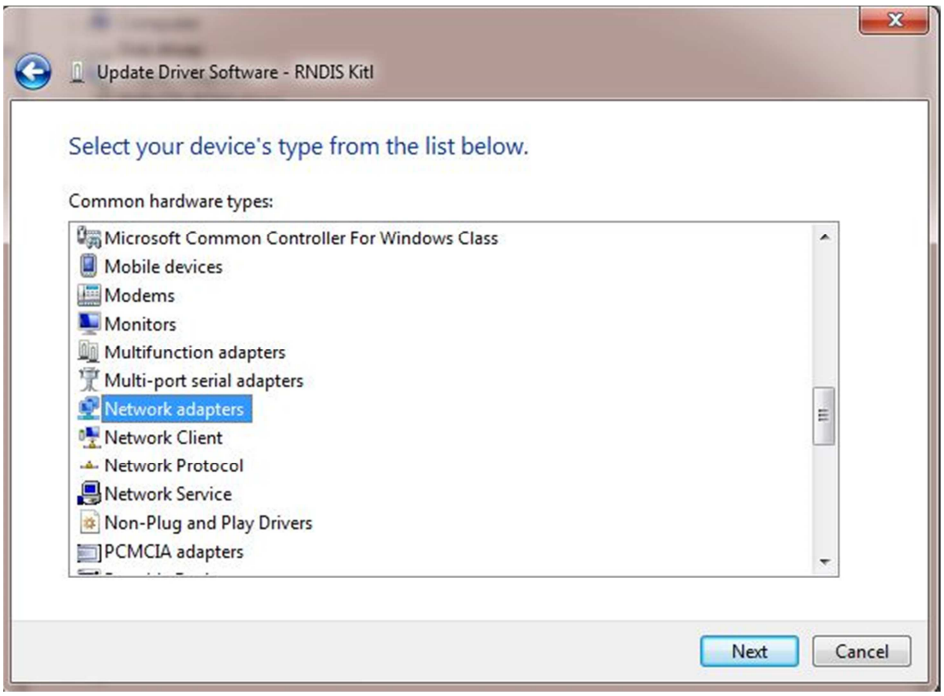


2. Right click on **Computer** and select **Manage**. From **System Tools**, select **Device Manager**. It will show a list of devices currently connected with the development PC. In the list, RNDIS Kitl can be seen with an exclamation mark implying that driver has not been installed.

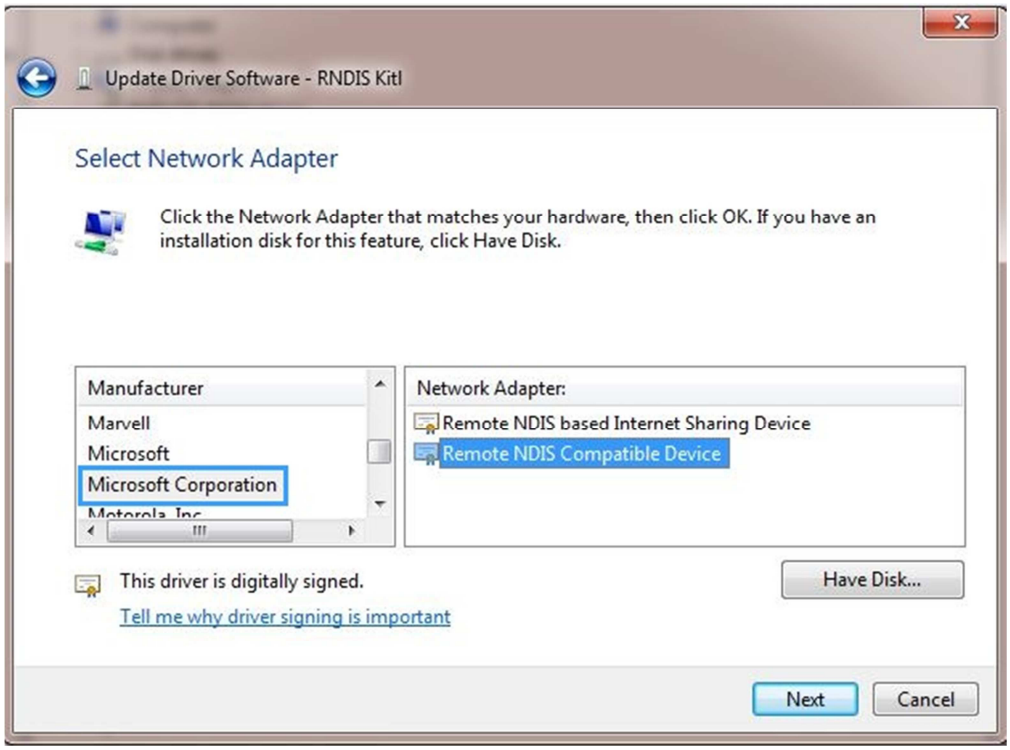
3. Right click on it and select **Update Driver Software...** When prompted to choose how to search for device driver software, choose **Browse my computer for driver software**.

4. **Browse for driver software on your computer** will come up. Select **Let me pick from a list of device drivers on my computer.**

5. A window will come up asking to select the device type. Select Network adapters, as RNDIS emulates a network connection.



6. In the Select Network Adapter window, select Microsoft Corporation from the **Manufacturer** list. Under the list of **Network Adapter:**, select **Remote NDIS compatible device**.



7. The RNDIS Kitl device is now installed and ready for use.