



di Boaretto Claudio srl

IDSL INITIALIZATION MANUAL

V1.0 - May 2023

A. Annunziato¹, D.A. Galliano², E. Capelli¹, E. Sabbatino³

1. Società Italiana Componenti Elettronici

2 – Joint Research Centre of the European Commission

3 – Piksel S.r.l.

UNESCO Contract – EN- 4500484180



IDSL INITIALIZATION MANUAL

Table of Contents

1	Introduction	4
2	IDSL Initialization	6
2.1	Download a prepared image from JRC	6
2.2	Download a new image from scratch from the Raspberry site.....	6
2.2.1	Download and install.....	6
2.2.2	Integrated tool.....	7
2.2.3	Power up	10
2.2.4	Configure LAN for static IP.....	10
2.2.5	Enable ssh.....	12
2.3	Installation of other needed software	13
2.3.1	Update the system.....	13
2.3.2	Enable time service (OPTIONAL STEP)	14
2.3.3	Install WiringPi.....	14
2.3.4	Install ftp and telnet services.....	14
2.3.5	sshpass	15
2.3.6	serial port support.....	15
2.3.7	MiniCom	16
2.3.8	Bluetooth	16
2.3.9	I2c support.....	16
2.3.10	ssl support	17
2.3.11	Turn off the SWAP space.....	18
2.3.12	Assign hostname to the device	19
2.4	VPN installation	19
2.4.1	Create a userid in the LogMeIn Hamachi	19
3	Installation of IDSL specific software.....	23
3.1	C version or TAD	24
3.1.1	Using the precompiled version.....	24
3.1.2	Recompiling the source code.....	25
3.2	Python version or pyTAD	25
3.2.1	pyTAD download and install.....	25
3.2.2	configuration of the device	26
3.3	C# version or RIO	30

3.3.1	Runtime installation.....	30
3.3.2	RIO installation	30
3.3.3	RIO initialization.....	33
4	Other settings	40
4.1	SMS commands for RPI.....	40
4.2	Update crontab	41
5	Saving the prepared image	43
6	Webcam initialization	44
6.1	Download the image of the basic operating system of the Raspberry W	44
6.1.1	Download a prepared image from JRC	44
6.1.2	Download a new image from scratch from the Raspberry website	44
6.2	Copy the software and prepare it for the execution	47
6.3	Connect the RPI and the ZERO W.....	47
6.4	Install the VPN.....	48
7	First switch on of an IDSL	49
7.1	Check the config file.....	49
7.2	Verify that the software is running	49
7.3	Verify that the sensor is providing data	49
7.4	Verify that the solar panel is working.....	50
8	IDSL Remote Verification.....	51
8.1	Verify that the software is running	51
8.2	Verify that the disk is not full	51
8.3	Verify that the sensor is providing data	51
8.4	the webcam cannot be reached	51
9	Conclusions	52
10	Appendix A – Bash commands Alias for RIO	53

1 INTRODUCTION

This document describes the activities to be performed at the receipt of a new IDSL in order to make it operational. The IDSL Installation Guide describes the physical installation of the IDSL: [IDSL Installation Guide](#) ¹

The IDSL contains a Raspberry PI device on which the data collection software is installed. It allows reading the sensors: level, temperature, voltage of the battery, CPU and ambient temperature, air pressure, but this last only in some recent models.

The software sends the measurements to a server through Internet (using mobile/broadband/satellite connection). Three implementations are available, in C, C# and Python. The server, located at the JRC,

- Contains the definition of the device
- Collects the recorded data
- Visualizes the data
- Allows users retrieving selections of the data.

It is possible to connect remotely with the devices having access through a VPN that allows to perform SSH commands via Internet. A double encryption protects the connection, using both an encrypted VPN and the `ssh` protocol to connect with the device.

To initialize an IDSL, it is necessary to:

- Download the image of the basic operating system of the Raspberry PI 2 model 2 and up
- Copy the software and prepare it for the execution
- Initialize the device
- Activate the VPN to allow a remote connection

If a webcam is also included in the installation kit:

- Download the image of the basic operating system of the Raspberry PI Zero W
- Copy the software and prepare it for the execution
- Initialize the device

IMPORTANT NOTES:

- 1) As soon as it is possible, do not give the commands copying from this document but use copy/paste from this document to the raspberry terminal. This will avoid to introduce wrong commands in the configuration files.**
- 2) Print this document and mark every step with your signature to be sure you did exactly all the steps**

All the files and scripts described in this note can be found in this web page:

¹ [https://idslarchive.z6.web.core.windows.net/IDSL Installation Guide v2.pdf](https://idslarchive.z6.web.core.windows.net/IDSL%20Installation%20Guide%20v2.pdf)

<https://idslarchive.z6.web.core.windows.net/>



IDSL Archive

Space, Security and Migration Directorate - JRC Ispra Site

Raspberry file system images

[IDSL](#)
[Webcam](#)

Precompiled software

[C Version](#)
[RIO Version](#)
[python Version](#)

Additional contents

[Scripts](#)
[Image writing software](#)

Resource Code Repositories

[C Version](#)
[RIO Version](#)
[python Version](#)

Documents

[Installation Guide](#)
[Initialization Guide](#)

2 IDSL INITIALIZATION

Note:

If the device arrives with a preinstalled image, this step and the following are not needed. Skip them and proceed to paragraph 2.3.

To create a new SD card it is necessary to have:

- the image of the IDSL
- an image writer software
- a SD card reader (it could be external to a laptop or part of a laptop itself)

The initialization of the SD card can be performed in two ways:

- Starting from a prepared image from JRC (this is faster)
- Downloading a basic image from the Raspberry site (this could be necessary either if the JRC site is not available or if a latest version of Raspberry requires a new image)

2.1 DOWNLOAD A PREPARED IMAGE FROM JRC

The image can be retrieved from JRC and used with an image writer, such as Win32DiskImager (<https://win32diskimager.org/>) or any equivalent software.

At this web site the latest IDSL image can be found, together with the software to write the image on the SD card:

<https://idslarchive.z6.web.core.windows.net/IDSL.OS.zip>

Using the disk image writer software and a SD card reader it is possible to transfer the image on the SD card. Then you should perform these steps:

- Enable ssh (see 2.2.5)

With this, you can already access the raspberry at address 192.168.1.101 and the device is already transmitting the data as IDSL-00 to the server. More details could be set up but the system should already work with a basic configuration

- Adapt the VPN parameters, from section **Error! Reference source not found.**

The go to section 3 o identify which software version you want to use.

2.2 DOWNLOAD A NEW IMAGE FROM SCRATCH FROM THE RASPBERRY SITE

Alternatively, it is possible to download the latest image from the Raspberry official web site and perform a longer series of preli

- Download and write the image on the SD card
- Install other needed software
- Install the VPN

2.2.1 DOWNLOAD AND INSTALL

Download the software PIIImager that allows to prepare the SD card from the official website:

<https://www.raspberrypi.org/downloads/raspbian/>

Choose the version for the type of computer you are using (Windows, Mac or Ubuntu). Then launch and select the right version of the OS to install the **32 bit Lite** (see 2.2.2). Go to step 2.2.2.

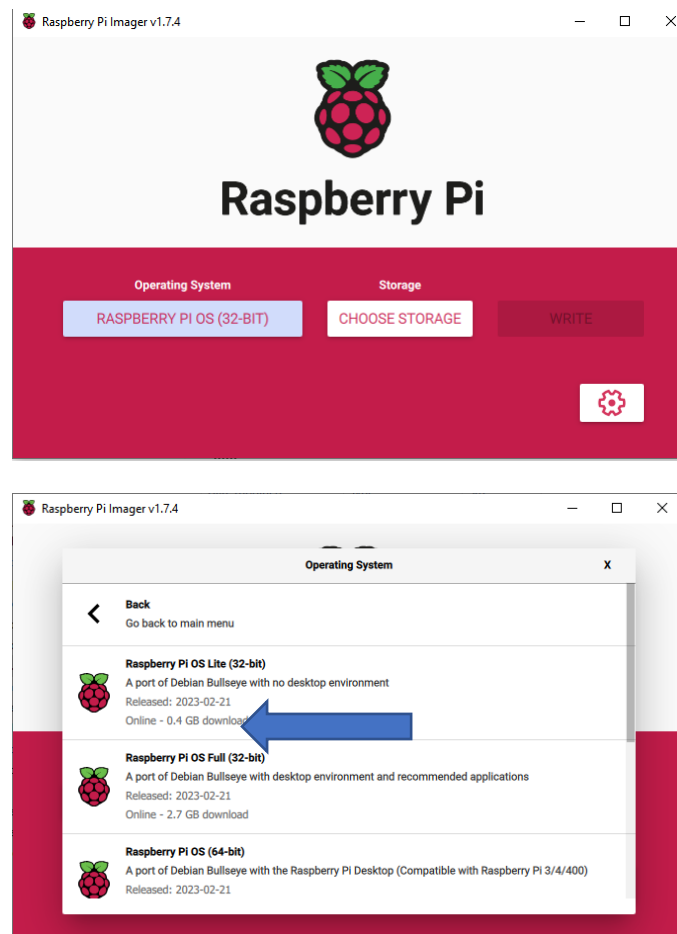
As an alternative you can download the image and use a software tool to flash the SD card. Again chose the image corresponding to a 32 bit Lite version. Two versions are available, with or without Desktop: since it is not needed, chose the one without Desktop, e.g.

https://downloads.raspberrypi.org/raspios_lite_armhf/images/raspios_lite_armhf-2023-02-22/2023-02-21-raspios-bullseye-armhf-lite.img.xz).

If the image was downloaded manually, write it on the microSD using Etcher (<https://etcher.io/>). This is the recommended tool from raspbian team, and it will burn and validate the written image in a simple interface.

2.2.2 INTEGRATED TOOL

The dedicated software from Raspberry to write the image on the card can also be used: this is its interface.



Select the operating system and choose the option **32 bit Lite**.



THIS IS A VERY IMPORTANT POINT. CHOOSE THE RIGHT IMAGE (32 bit Lite) OTHERWISE ALL THE REST DOES NOT WORK CORRECTLY.

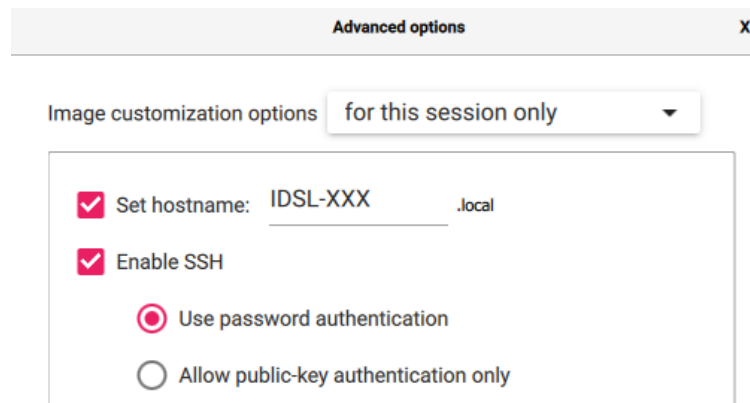
Set Advanced Options

Click on the Gear Button



Select the flag **“Set hostname”** and insert the name of the IDSL (prefix should be IDSL-). If you are testing the system use **IDSL-00** as hostname.

Select the flag **“Enable SSH”** and select **“Use password authentication”**



Set the flag **“Set Username and Password”**

Advanced options X

Set username and password

Username:

Password:

The following step is not necessary but is important if you are installing the system for the webcam. So if you are installing the raspberry for the IDSL go to 2.2.3

Set the Wi-Fi Network (if needed) and **SAVE** it the advanced options

Advanced options X

Configure wireless LAN

SSID:

Hidden SSID

Password:

Show password

Wireless LAN country:

Set locale settings

SAVE

Then press on **WRITE** to transfer the image on the SD card

It is possible to create manually the Wi-Fi config file (wpa_supplicant.conf)

The file can be created in windows environment and copy on the on the SD boot partition

wpa_supplicant.conf

```
country=it
update_config=1
ctrl_interface=/var/run/wpa_supplicant
network={
    scan_ssid=1
    ssid="WiFi"
    psk="WifiPass"
}
```

The boot partition already contain some of the following files :

- bootcode.bin
- loader.bin

- start.elf
- kernel.img
- cmdline.txt

The file will be moved to /etc/wpa_supplicant folder after the first boot of the RPI

2.2.3 POWER UP

The next steps will be performed after having turned on the Raspberry with the new SD card inserted into it, with a keyboard connected to the USB port and a monitor, connected to the HDMI port. To power up the raspberry, use a power cable connected to the microUSB connector.

At the first switch-on the system will ask to include a few details like the language and the time zone.

It will also ask to define a user and password. For the user specify 'pi' and for the password use the standard one 'raspberry.' Do not forget to change it, after completing the configuration.

2.2.4 CONFIGURE LAN FOR STATIC IP

Edit `dhcpcd.conf` with

```
sudo nano /etc/dhcpcd.conf
```

Update the static IP settings in the file, normally commented (with # symbols). Remove the # as first character of the following lines and update their content.

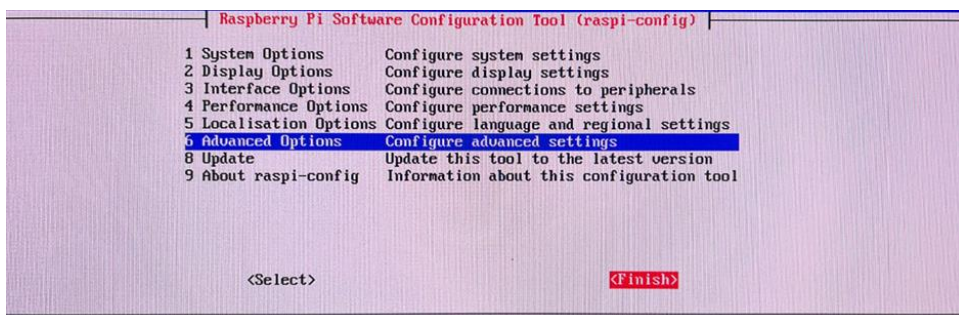
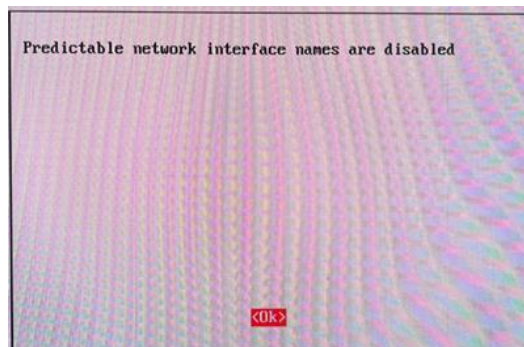
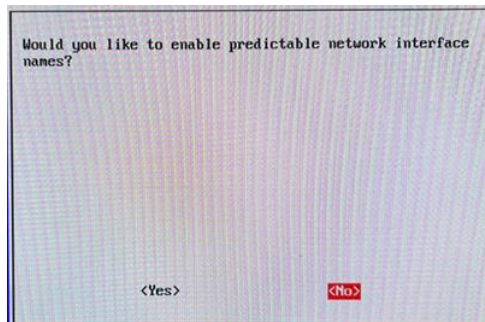
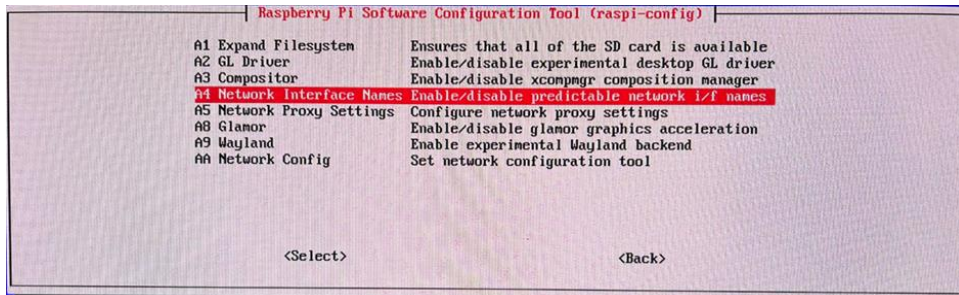
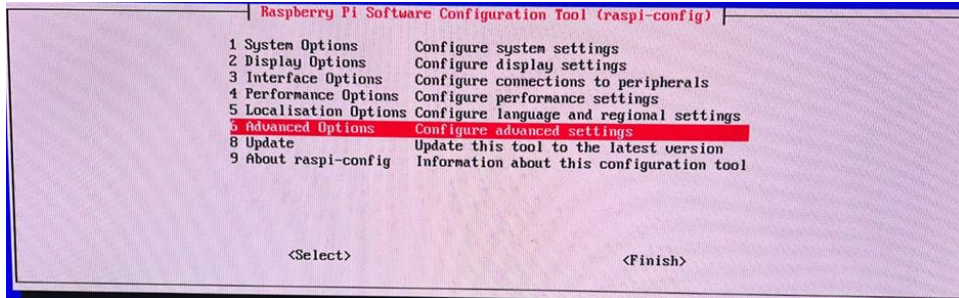
```
#static IP configuration
interface eth0
static ip_address=192.168.1.101    ##  this will be your IP
static routers=192.168.1.1
static domain_name_servers=192.168.1.1
```

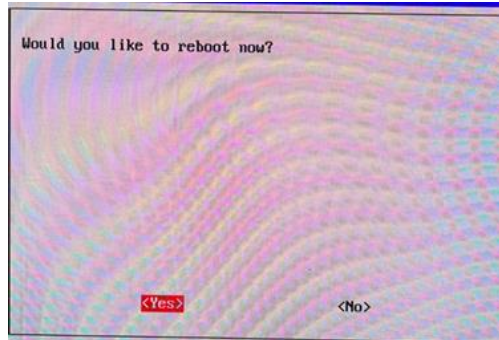
Setting a static IP in the *stretch* release of Raspbian has changed, because the eth0 is replaced by the new enx format.

To disable the predictable network names of the interfaces you can run the command below. However if the system will not ask to reboot it means that this step was not necessary because the system was already set as needed.

sudo raspi-config

then select **6 Advanced Option > A4 Network Interface Names > No > Back – Finish** and Reboot **YES**





<https://www.raspberrypi.com/documentation/computers/configuration.html#network-interface-names>

2.2.5 ENABLE SSH

Issue the following command if during the initialization with PI Imager the ssh was not already enabled:

```
sudo raspi-config
```

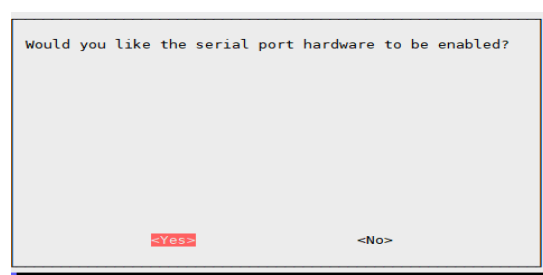
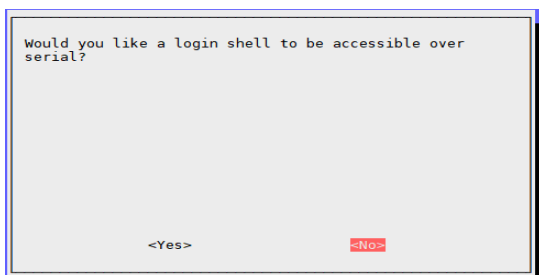
Select "3 Interface Options"

```
enable ssh
```

(To perform this step automatically, include an empty file named ssh in the boot section of the SD card, at the end of the SD card writing procedure and before inserting it into the raspberry.)

Select `Serial port` and disable login shell and enable serial port for data.

```
I1 Legacy Camera Enable/disable legacy camera support
I2 SSH           Enable/disable remote command line access using SSH
I3 VNC          Enable/disable graphical remote access using RealVNC
I4 SPI          Enable/disable automatic loading of SPI kernel module
I5 I2C          Enable/disable automatic loading of I2C kernel module
I6 Serial Port  Enable/disable shell messages on the serial connection
I7 1-Wire       Enable/disable one-wire interface
I8 Remote GPIO  Enable/disable remote access to GPIO pins
```



Now reboot the raspberry either by power-cycle or with the command `sudo reboot`.

Now you should be able to access your Raspberry from a laptop on the same LAN, using an ssh connection software like **putty** or **MobaXterm** by accessing

```
Address: 192.168.1.101
User: pi
Pwd: raspberry
```

In order to connect with the raspberry your laptop must be connected with the Teltonika wifi network . To do that, connect with the wifi (normally SSID=Teltonika or TeltonikaEcml with password Ecml2011 or 1102lmcE).



From this moment on, the external monitor and the keyboard are no longer needed: the system can be accessed using a network connection. In any case, from now on the system needs to access Internet, either provided by the laptop via Ethernet port or with a direct connection, e.g. Wi-Fi.

2.3 INSTALLATION OF OTHER NEEDED SOFTWARE

2.3.1 UPDATE THE SYSTEM

Before starting, please check that you can go in internet with the raspberry. This can be done issuing one of the two commands:

```
ping 8.8.8.8
```

or

```
wget www.google.com
```

Issue the following command:

```
sudo apt-get update
```

Check the system version:

```
uname -a
```

It should reply with something like:

```
"Linux raspberrypi 5.15.84-v7+ #1613 SMP Thu Jan 5 11:59:48 GMT 2023
armv7l GNU/Linux"
```

Set local time to UTC

```
sudo dpkg-reconfigure tzdata
```

when requested indicate "None of the above" and then "UTC"

```
"Current default time zone: 'Etc/UTC'
```

```
Local time is now: Tue Mar 28 10:26:29 UTC 2023.
```

Universal Time is now: Tue Mar 28 10:26:29 UTC 2023.”

2.3.2 ENABLE TIME SERVICE (OPTIONAL STEP)

Only if you want to introduce your own time service, you can follow this

<https://wiki.archlinux.org/index.php/Systemd-timesyncd>

```
sudo timedatectl set-ntp true
```

Modify the ntp servers file including other time servers you would like to include.

```
sudo nano /etc/systemd/timesyncd.conf
```

2.3.3 INSTALL WIRINGPI

Use the following commands to install the library needed to interact with devices connected to the RPi IO:

```
cd /tmp
wget https://unicorn.drogon.net/wiringpi-2.46-1.deb
sudo dpkg -i wiringpi-2.46-1.deb
```

Test the installation running this command:

```
gpio -v
```

The output should look like

```
Copyright (c) 2012-2018 Gordon Henderson
This is free software with ABSOLUTELY NO WARRANTY.
For details type: gpio -warranty
```

Raspberry Pi Details:

Tyls

lspe: Pi 3, Revision: 02, Memory: 1024MB, Maker: Sony

** Device tree is enabled.*

**--> Raspberry Pi 3 Model B Rev 1.2*

** This Raspberry Pi supports user-level GPIO access.*

```
gpio readall (to verify the configuration of all the embedded 40 PINS)
```

2.3.4 INSTALL FTP AND TELNET SERVICES

Install ftp (<https://packages.debian.org/stretch/ftp>)

```
sudo apt-get install ftp
```

Install telnet (<https://packages.debian.org/stretch/telnet>)

```
sudo apt-get install telnet
```

2.3.5 SSHPASS

To install sshpass, <https://packages.debian.org/stretch/sshpass>, issue this command:

```
sudo apt-get install sshpass
```

2.3.6 SERIAL PORT SUPPORT

<https://spellfoundry.com/2016/05/29/configuring-gpio-serial-port-raspbian-jessie-including-pi-3/>

With the previous steps, this should not be necessary. Check however the following

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

If it is not present, add the line (at the bottom):

```
enable_uart=1
```

If it was not present, disable serial service

```
sudo systemctl stop serial-getty@ttyAMA0.service
```

```
sudo systemctl disable serial-getty@ttyAMA0.service
```

In case you are installing on a RPI 3, use this syntax:

```
sudo systemctl stop serial-getty@ttyS0.service
sudo systemctl disable serial-getty@ttyS0.service
```

You also need to remove the console from the cmdline.txt. If this file exists, edit it using this command:

```
sudo nano /boot/cmdline.txt
```

If it contains something like:

```
console=serial0,115200
console=tty1
root=PARTUUID=9298bd00-02
rootfstype=ext4
fsck.repair=yes
rootwait
```

Remove the line:

```
console=serial0,115200
```

Save the file.

Reboot the device with

```
sudo reboot
```

2.3.7 MINICOM

To check the sensor, use a software to connect to the serial port, such as

```
sudo apt-get install minicom
```

In this case, use the following command to see the output from the sea level device

```
Sudo minicom -b 9600 -D /dev/ttyAMA0
```

2.3.8 BLUETOOTH

Disable Bluetooth and set /dev/ttyAM0 to real UART (as before).

Edit the file /boot/config.txt and add the following line at the end:

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

Add at the end the following line

```
dtoverlay=pi3-disable-bt
```

Save the file and exit.

Then stop Bluetooth service with

```
sudo systemctl disable hciuart
```

Reboot the device with

```
sudo reboot
```

2.3.9 I2C SUPPORT

I2C is the serial bus used to connect many peripherals, and it is used for the distance sensor.

```
sudo apt-get install -y python3-smbus
```

<https://packages.debian.org/stretch/python-smbus>

The previous package depends on the i2c-tools package that should be installed automatically. Check it with

```
dpkg -l i2c*
```

It should reply with


```
pi@raspberrypi:~$ dpkg -l i2c*
Desired=Unknown/Install/Remove/Purge/Hold
| Status=Not/Inst/Conf-files/Unpacked/halF-conf/Half-inst/trig-aWait/Trig-pend
|/ Err?=(none)/Reinst-required (Status,Err: uppercase=bad)
||/ Name          Version          Architecture Description
-----+-----
ii i2c-tools       4.2-1+b1        arm64         heterogeneous set of I2C tools for Linux
```

Otherwise, install it manually:

```
sudo apt-get install -y i2c-tools
```

Then edit the modules file

```
sudo nano /etc/modules
```

To add these lines in the file:

```
i2c-bcm2708
i2c-dev
```

Check the file `/etc/modprobe.d/raspi-blacklist.conf`

If the file exists and contains these lines:

```
blacklist spi-bcm2708
blacklist i2c-bcm2708
```

Edit them to be

```
# blacklist spi-bcm2708
# blacklist i2c-bcm2708
```

If the file does not exist or does not contain those lines, forget this point.

Recent version of the operating system use a kernel version 3.18 or higher. It can be checked using `uname -a`.

For these versions, it is also needed to update the `/boot/config.txt` file.

Edit it with `sudo nano /boot/config.txt` and add the text

```
dtparam=i2c1=on
dtparam=i2c_arm=on
```

then

```
sudo apt-get install libi2c-dev
```

<https://packages.debian.org/stretch/libi2c-dev>

2.3.10 SSL SUPPORT

The ssl support must be present. To install it, use

```
sudo apt-get install libssl-dev
```

<https://packages.debian.org/stretch/libssl-dev>

To use it when accessing websites, check if libcurl is up to date:

```
sudo apt-get install libcurl4-openssl-dev
```

<https://packages.debian.org/stretch/libcurl4-openssl-dev>

In case of failure, look here <https://packages.debian.org/stretch/armhf/libcurl4-openssl-dev/download>

Then install bc

```
sudo apt-get install bc
```

<https://packages.debian.org/stretch/bc>

To complete the procedure edit the environment file

```
sudo nano /etc/environment
```

adding the line

```
LANG=en_US
```

2.3.11 TURN OFF THE SWAP SPACE

The swap space is an area of the mass storage used to simulate an additional (virtual) memory of the device. In this case, it is not needed and would age quickly the SD card with no benefit.

To turn off swap use the following command:

```
sudo systemctl disable dphys-swapfile
```

then reboot

```
sudo reboot now
```

This option might not be permanent, to verify if the swap file is created you can use the command

```
sudo swapon -show
```

and double check with

```
free -h
```

if in the output of the second command you see a line with Swap is still present. If Raspbian continues to create a swap file after being rebooted, there is always the option to remove the package that manages it.

```
sudo apt-get remove dphys-swapfile
sudo apt-get purge dphys-swapfile
```

2.3.12 ASSIGN HOSTNAME TO THE DEVICE

If you have not assigned the name at the beginning with PIIImager, this will be the name that will also be sent to the server to store the data

```
sudo raspi-config
```

Select `System options` and then `Hostname`.

This name must also be present in the file `/etc/hosts`. If not, edit the file with:

```
sudo nano /etc/hosts
```

Include

```
127.0.0.1 <Name chosen for the hostname>
```

Save and reboot.

2.4 VPN INSTALLATION

If the ISP is providing the device with a fixed external IP, it will be possible to access the device remotely. In general this type of connection is more expensive or not available.

Other software, mainly using http tunnelling methods are required to establish with the device a Virtual Private Network (VPN). Examples of these software are Logmein Hamachi, Remote.it, TeamViewer and other. JRC used the first one throughout the whole implementation of the project.

For TeamViewer a good explanation on how to setup is provided here:

<https://pimylifeup.com/raspberry-pi-teamviewer/>

Anyway, it did not prove to be 100% reliable.

2.4.1 CREATE A USERID IN THE LOGMEIN HAMACHI

Go in <https://vpn.net/> and select “sign up” at the top right of the page to create a userid. The service is free to connect up to 5 devices: since one controlling station such as a PC must be present in the network, only 4 devices can be added. Subscriptions are available to connect a larger number of devices, the least expensive allowing up to 32 devices.

Once logged in, the network must be created. This is identified by 3 numbers like:

ID	Name	Type	Description
395-712-xxx	AlessandroNET	Mesh	

Name	Client ID	Tag	Details
AA\Laptop [Guest]	218-134-744		Edit
IDSL-CLONE-CAM [Guest]	271-301-298		Edit

ID	Name	Type	Description
----	------	------	-------------

395-712-xxx

(xxx is masked in this case)

This 3 digit number is needed to join the network. This operation can be protected by a password.

Check the last version here <https://www.vpn.net/linux>

Select the link corresponding to the latest armhf.deb, such as

```
wget https://www.vpn.net/installers/logmein-hamachi\_2.1.0.203-1\_armhf.deb --no-check-certificate
```

Then, issue the following commands

```
sudo dpkg -i logmein-hamachi_2.1.0.203-1_armhf.deb
sudo Hamachi
```

**** IF AT THIS MOMENT YOU GET:**

Illegal Instruction,

try the following:

Purge the previous installation

```
dpkg -P logmein-hamachi
```

Download the 'el' version

```
wget https://www.vpn.net/installers/logmein-hamachi\_2.1.0.203-1\_armel.deb
```

Install the el version forcing the architecture:

```
sudo dpkg --force-architecture --force-depends -i logmein-hamachi_2.1.0.203-1_armel.deb
```

After installing launch the following

```
sudo hamachi login
```

```
sudo hamachi attach xxx@xxx.xx
```

Specify here the LogMeIn Hamachi account, then

- connect on LogMeIn website <https://secure.logmein.com/central/Central.aspx>
- accept the pending request in the website of logmein (sometimes it is necessary to log off and login again to see the pending request)
- configure the network in “non-members”
- edit the client to select the right network
- on the device perform the following commands:

```
sudo hamachi set-nick "yyyyy" (example: IDSL-401)
```

```
sudo hamachi do-join 382-886-xxx (the ID of the network)
```

```
sudo nano /var/lib/logmein-hamachi/h2-engine.cfg
```

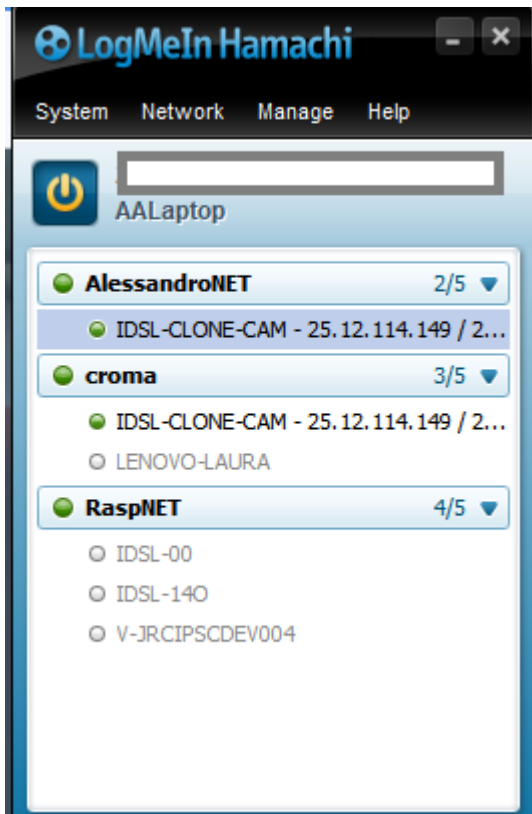
- search and change the keyword LoginOnLaunch

```
set LoginOnLaunch=1
```

On the controlling PC or laptop, the LogMeIn Hamachi client must be installed as well from:

<https://vpn.net/>

The tool allows monitoring the network and identifying the devices addresses to connect with.



From this moment on, the Raspberry can be reached from everywhere. It is also possible to join more than one network. This can be useful for instance to delegate the maintenance of all the devices to the same organization; while all other organizations can access only their own device.

3 INSTALLATION OF IDSL SPECIFIC SOFTWARE

There are 3 possible versions of the software to install:

- The c version (tad)
- The python version (pyTAD)
- The C# version (RIO)

The c version is the first that was developed and runs on all the raspberry versions while the python and C# require at least an ARMv7 CPU.

The python version is relatively simple to install and has the same configuration as the c version. The software is contained in a GitHub space.

The RIO version can be installed on IDSL or TAD panels just by modifying its configuration file (that however is a bit different from the python version but with similar keys).

All the initial IDSLs use the c version while the more recent ones the python version. The TAD panels all the RIO version. The software modes can be interchanged but only one can be made working.

Before starting, download this file and unzip it:

```
wget https://github.com/annunal/pyTAD/archive/refs/heads/main.zip
unzip main.zip
rm main.zip
```

Create the script folder and copy the files there:

```
sudo mkdir /home/script
sudo cp /home/pi/py-main/scripts/* /home/script
```

make all the files with extension sh executable:

```
sudo chmod +x /home/script/*.sh
```

Activating the crontab statements included in the script directory, the acquisition with one or the other software should start. The activation of the crontab is described later.

The choice of the software to be used is done by changing one line in the file SetVars.sh in the `script` folder, the mode is set by modifying the following line:

```
#####
export modeTAD=pyTAD      #   TAD or pyTAD   or RIO
#####
```

3.1 C VERSION OR TAD

The python version of the software can be downloaded here:

<https://github.com/annunal/TAD>

```
wget https://github.com/annunal/TAD/archive/refs/heads/main.zip

unzip main.zip
```

This will create a directory TAD-main. You have 2 possibilities:

- Use the precompiled version
- Recompile the source code

3.1.1 USING THE PRECOMPILED VERSION

To use the precompiled version, copy the folder TAD0 in the folder /home/pi/programs (if it does not exist, create it):

```
mkdir /home/pi/programs

cp -r /home/pi/TAD-main/TAD0 /home/pi/programs

chmod +x /home/pi/programs/TAD0/*.sh

chmod +x /home/pi/programs/TAD0/tad*
```

To test the system, copy the files in a temporary directory on /tmp and launch the program:

```
mkdir /tmp/TAD

cp /home/pi/programs/TAD0/* /tmp/TAD

cp /home/pi/programs/TAD0/periodic/* /tmp/TAD

sudo /tmp/TAD/tad
```

The program should start collecting the data:

```
Tmax30, Tmax300=300.000000,3000.000000
n300=300
n30=30
threshold=0.200000
ratioRMS=5.000000
AddRMS=0.200000
backFactor=0
multFact=-1.000000
addFact=4.817000
Reading file: buffer.txt
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
   Dload  Upload  Total   Spent    Left     Speed
100  270    100  270    0    0    3698    0  --:--:--  --:--:--  --:--:--    3698
17/05/2023 03:38:12, 3.542839, 3.543079, 3.543157, 0.000059, 0.000000, 0.000078, 0, 12, 14.083144, -5.500000, -1.000000, 23.871521, T
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
   Dload  Upload  Total   Spent    Left     Speed
100  270    100  270    0    0    5400    0  --:--:--  --:--:--  --:--:--    5400
17/05/2023 03:38:23, 3.542771, 3.542983, 3.543145, 0.000091, 0.000000, 0.000163, 0, 11, 13.979110, -5.500000, -1.000000, 23.932198, T
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
   Dload  Upload  Total   Spent    Left     Speed
100  270    100  270    0    0    5869    0  --:--:--  --:--:--  --:--:--    6000
17/05/2023 03:38:34, 3.542803, 3.543135, 3.543135, 0.000109, 0.000000, 0.000226, 0, 11, 14.017751, -5.500000, -1.000000, 23.862411, T
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
```

3.1.2 RECOMPILING THE SOURCE CODE

Identify and move to src_2018 folder:

```
cd /home/pi/TAD-main/src_2018
make
```

This should create a fresh new tad program.

Follow all the steps described in 3.1.1. At the end, delete the file `tad` and replace it with the one just created (you could move the file, but this is to be sure that you are using the newly created executable).

```
rm /home/pi/programs/TAD0/tad; rm /home/pi/programs/TAD0/tad1;
rm /home/pi/programs/TAD0/tad-retry
rm /tmp/TAD/tad*
cp /home/pi/TAD-main/src_2018/tad /home/pi/programs/TAD0/tad
cp /home/pi/TAD-main/src_2018/tad /home/pi/programs/TAD0/tad1
cp /home/pi/TAD-main/src_2018/tad /home/pi/programs/TAD0/tad-retry
cp /home/pi/TAD-main/src_2018/tad /tmp/TAD
```

To check that the files are identical you can use the `diff` command:

```
diff /home/pi/TAD-main/src_2018/tad /tmp/TAD/tad
```

You can test the execution as in the previous case launching the command:

```
sudo /tmp/TAD/tad
```

[If all works fine, you should see the log of the acquired data such as the image below and to check if the data are properly updated verify the page on JRC site:](#)

https://webcritech.jrc.ec.europa.eu/TAD_server/Device/IDSL-00

If in the `setVars.sh` script, as described at the beginning of chapter 3 you have indicated TAD the c version of the programme will start automatically (after having copied all the scripts).

3.2 PYTHON VERSION OR PYTAD

3.2.1 PYTAD DOWNLOAD AND INSTALL

The python version of the software can be downloaded here:

<https://github.com/annunal/pyTAD/tree/main/prog>

```
wget https://github.com/annunal/pyTAD/archive/refs/heads/main.zip
unzip main.zip
```

```
rm main.zip
```

This will create a directory `pyTAD-main`. The program will run from the directory `prog` inside this directory.

Create a folder under the `/home/pi` directory named `programs/pyTAD` and copy here all the files contained in the URL (Uniform Resource Locator) indicated before.

```
mkdir /home/pi/programs
mkdir /home/pi/programs/pyTAD
cp /home/pi/pyTAD-main/prog/* /home/pi/programs/pyTAD
```

To run the software, it is necessary to install the following packets, if `python3` is not installed:

```
sudo apt-get install python3
sudo apt-get install python3-pip
sudo pip install psutil requests numpy paho-mqtt pyserial
sudo pip install wiringpi2
sudo pip3 install smbus
sudo pip install pycountry-convert
```

Please note that this will take a few minutes to install.

If the pressure sensor is also installed:

```
sudo pip3 install adafruit-circuitpython-bmp3xx
sudo apt-get install libatlas-base-dev
sudo pip install pybind11
```

To check if the software is correctly configured, run the script, even if the configuration file (`config.txt`) should be modified and updated: this can be done later. Run the commands:

```
cd /home/pi/programs/pyTAD_
sudo python tad.py -c ./
```

[If all works fine, you should see the log of the acquired data such as the image below and to check if the data are properly updated verify the page on JRC site:](#)

https://webcritech.jrc.ec.europa.eu/TAD_server/Device/IDSL-00

If the C or the python version are used, a configuration file, named config.txt must be present in the folder, where the software runs.

An example of configuration file is provided here:

<https://github.com/annunal/pyTAD/blob/main/other%20files/config.txt>

All configuration keywords are explained here (lines starting with * and # are comments and are disregarded by the software):

```

*****
**  Generals
*****
title           = name of the device
location        = location of the device
position        = latitude,longitude of the device, separated by comma
IDdevice        = ID of the device as is in the web repository. To use the name of the device, use $HOSTNAME
SaveURL         = URL to save data
#SaveURLb       = Second URL to save data
AlertURL        = URL to alert

*****
** Photo shot commands
*****
PhotoCMD        = Internal URL to shot a photo
PhotoTimeInterval = Time interval in minutes to shot a photo after an alert
PhotoAlertLevel = Alert level to shot an image

*****
** Alerts parameters
*****
AlertLevel      = Alert is issued if Alert Level is larger than this value

** Email parameters
EmailTo         = list of email addresses, comma separated
EmailURL        = URL of the service to send out emails
EmailTemplate   = Template file containing the email body
EmailSubject    = Template file containing the email subject
AlertTimeInterval = Time interval after which, if still in alert mode, another alert is sent

** SMS (Short Message Service) parameters
SMSowners       = List of SMS recipient names (for reference)
SMSlist         = List of comma separated recipient phone numbers complete with country codes
SMSURL          = URL of the service to send out SMS
SMSuser         = userid to send out SMS messages (** ASK JRC)
SMSpwd          = password to send out SMS messages (** ASK JRC)
SMSTemplate     = Template file for the SMS messages

*****
** Periodic messages
*****
TemplatePeriodic_SMS_Msg = Template file for a periodic test SMS message
SMSlistPeriodic         = List of phone numbers, comma separated, that will receive a periodic SMS
SMSowners_ADM           = List of the names associated to administrative phone numbers for reference
SMSlistPeriodic_ADM     = List of SMS numbers, comma separated, that will receive a daily Admin SMS

EmailToPeriodic         = List of email addresses, comma separate that will receive a periodic email
EmailToPeriodic_ADM    = List of email addresses, comma separate that will receive a daily periodic
admin email

TemplatePeriodic_EMAIL_Body = File containing the email body template for periodic email
TemplatePeriodic_EMAIL_Subj = File containing the email subject template for periodic email

** 4=Wednesday -1=daily

```

Periodic_Day = The week day for periodic emails (4=Wednesday -1=daily)
 Periodic_Day_ADM = The week day for periodic admin email (4=Wednesday -1=daily)
 Periodic_hour = The time (UTC) of the day at which the periodic email are sent

* Analysis parameters

* Ref: <https://www.mdpi.com/1892362>

Interval = Acquisition interval (seconds)
 n300 = Number of points for Long Term Forecast
 n30 = Number of points for Short Term Forecast
 threshold = Threshold for activation of calculation (m), see Ref
 ratioRMS = RMS ratio (see ref)
 AddRMS = Addition to LTF-STF difference (m, see ref)
 backFactor = not used, obsolete
 methodInterp = not used, obsolete
 servo = not used, obsolete

* The level is provided as:

* level= sensorMultFac * MEASLEVEL + sensorAddFac

* MEASLEVEL is the distance between the sensor and the water

* you can correct the offset

sensorMultFac = multiplication factor for the level

sensorAddFac = addition factor for the level

The level is computed as follows:

$L = \text{sensorMultFact} * \text{measLevel} + \text{sensAddFac}$

MeasLevel is the value provided by the sensor, that is the distance between the sensor and the water surface. So the sensorMultFact must be -1 in order to invert the measurement and the sensAddFac will add a constant that should represent the difference between the elevation of the sensor and the bottom of the water below the sensor. In such a way the level will represent the water level below the sensor.

SonarMinLevel = Minimum measurable level of the Level sensor
 SonarMaxLevel = Maximum measurable level for level sensor
 SonarMaxDifference = Maximum difference above which the measure is considered an outlier

Serial = Address of the serial port (usually, /dev/ttyAMA0)
 BaudRate = Baud rate for the Serial port (9600)
 batteryPin = Pin for reading battery value (5, fixed)
 batteryMultiplier = Battery value multiplier (you can calibrate the voltage)

panelMultiplier=not used, obsolete

panelPin = not used, obsolete

sonarTempPin = Pin for reading temperature

* Temperature is provided as: $1/\text{Temp} = \text{SonarTempMultiplier} * \text{MEASTE MP} + \text{SonarTempAddConst}$

* you can adjust those quantities if not correct

SonarTempMultiplier = Multiplication factor for the temperature sensor

SonarTempAddConst = Addition factor for the temperature sensor

```
*****  
SaveAllData = All data are saved (1=yes, 0=no)  
simSonar    = Simulation of acquisition (1=yes, 0=no)  
voltageInterval = not used, obsolete
```

The operational location of the files is under `/home/pi/programs/`

When using the python version (pyTAD), create a directory pyTAD and copy there all the files that were contained in the installation directory:

```
cp /home/pi/pyTAD_main/prog/* /home/pi/programs/pyTAD
```

because the scripts assume that the starting location is **`/home/pi/programs/pyTAD`**

3.3 C# VERSION OR RIO

On the hardware platform developed for the IDSL, the JRC developed RIO (Remote InterOperability), a guest operating system to develop, operate and maintain a network of devices.

The software is based on .Net, and requires a simple set-up. Additional information are available at this page: [The Remote InterOperability Platform](#).²

.Net requires ARMv7 architecture devices as minimum: therefore, it is not compatible with Raspberry Pi Zero, 1 and 2. Please, refer to the Specifications table in [Raspberry Pi - Wikipedia](#) to know which are the compatible models.

3.3.1 RUNTIME INSTALLATION

To minimize the number of files to be deployed with every version of the software, the RIO version of the software is distributed as framework-dependent (for reference, see [Deploy .NET apps to ARM single-board computers | Microsoft Learn](#)³).

To set up the framework, this command will download and install the latest version available:

```
curl -sSL https://dot.net/v1/dotnet-install.sh | bash /dev/stdin --channel STS
```

To make the runtime available more easily, run the following commands:

```
echo 'export DOTNET_ROOT=$HOME/.dotnet' >> ~/.bashrc
echo 'export PATH=$PATH:$HOME/.dotnet' >> ~/.bashrc
source ~/.bashrc
```

To test the .Net installation, verify it with this command:

```
dotnet --version
```

3.3.2 RIO INSTALLATION

The IDSL scripts assume RIO to be installed in `/home/pi/programs/RIO`

It is a safe habit to deploy the software in a folder with a meaningful name, such as RIO.3.2.0 from its version number, and link to it the RIO name:

```
ln -s /home/pi/programs/RIO.3.2.0 /home/pi/programs/RIO
```

This way it will be easier to maintain a history of the versions and refer to them, in case of problems with a newer version.

² <https://www.codeproject.com/Articles/5331192/The-Remote-InterOperability-Platform>

³ <https://learn.microsoft.com/en-us/dotnet/iot/deployment>

The software can be:

- either built, using the Open Source version published on [ec-jrc/RIO: Remote InterOperability, an IoT solution \(github.com\)](https://github.com/ec-jrc/RIO)⁴
- or download a precompiled version from [RIO-IDSL.zip](https://idslarchive.z6.web.core.windows.net/RIO-IDSL.zip)⁵

Building the software

To build the software, it is necessary to

- set up a development machine
- acquire the sources
- build the software
- transfer the software on the device

The current version of RIO is tested and validated against .Net 3.1. As soon as validated against .Net 6, the new version will be available from the same repository.

The software must be prepared using another machine, either Windows or Linux, to download and build the software, since the development kit is not available for the Raspberry platform.

To build from a command line environment, acquire the Software Development Kit from

<https://dotnet.microsoft.com/en-us/download/dotnet/3.1>

and download the version related to the OS on which you are preparing the files (Windows or Linux).

Acquire the source code as a [zip file](#), or using [git](#) (if you do not have you should install it from <https://git-scm.com/downloads>). Then use the command:

```
git clone https://github.com/ec-jrc/RIO.git
```

In the source root, compile with command:

```
cd RIO\TAD

dotnet publish -r linux-arm -c Release .\TAD.csproj
```

The command will create a complete distribution of the software under `RIO/TAD/bin/Release/net6.0/linux-arm` or similar.

It is easier and faster compressing the folder and transfer the compressed file to the device instead of transferring all files one by one. So compress the folder linux-arm into a file named RIO-IDSL.zip

From this moment on, the procedure is the same of precompiled software.

Using compiled software

⁴ <https://github.com/ec-jrc/RIO>

⁵ <https://idslarchive.z6.web.core.windows.net/RIO-IDSL.zip>

Transfer the compressed file, e.g. RIO-IDSL.zip, to /home/pi/programs

Unpack the software: `unzip RIO-IDSL.zip`

Assign a meaningful name to the unpacked folder, e.g.

```
mv linux-arm RIO.3.2.0
```

In case the RIO name is already in use, delete it:

```
rm RIO
```

Assign the RIO name to the new version of the software:

```
ln -s RIO.3.2.0 RIO
```

The final step requires the program to acquire the executable status, since it is transferred as an ordinary file:

```
cd RIO
```

```
chmod a+x TAD
```

Launch for the first time the TAD programme, so that an initial Settings.json file is created that can then be fine tuned later.

```
sudo ./TAD
```

If all works fine, you should see the log of the acquired data such as the image below and to check if the data are properly updated verify the page on JRC site:

https://webcritech.jrc.ec.europa.eu/TAD_server/Device/IDSL-00

The elapsed time should be within few seconds.


```

04:39:00 INF telemetry:
Timestamp      2023-05-17T04:39:00.022
DeviceId      IDSL-00
FeatureId     IDSL
Value         5.04673
ShortForecast  5.04673
LongForecast  5.04673
Rms           0
AlertSignal   0
AlertLevel    0
Measures      30
Swh           0

04:39:05 INF telemetry:
Timestamp      2023-05-17T04:39:05.128
DeviceId      IDSL-00
FeatureId     IDSL
Value         5.04668
ShortForecast  5.04668
LongForecast  5.04668
Rms           0
AlertSignal   0
AlertLevel    0
Measures      31
Swh           0

```

3.3.3 RIO INITIALIZATION

The RIO platform is a modular system that uses plugins to extend its base functionality. When the system starts, plugins are loaded and configured. In case a configuration is not found, a default one is created. This allows users to have a starting point to configure the required functionality.

The configuration is stored in `Settings.json`, a file located where the software is running. If necessary, it will be created when the software runs for the first time. The best practice for a completely new device, is to start RIO and wait for the initialization to complete, then stop it and customize the settings.

The configuration contains general settings for the RIO system and specific settings for each plugin. It is possible to create different instances of the same functionality: for instance, it is possible to equip a RIO device with more sensors of the same type and configure each one differently using the same plugin more times. The plugins are searched in all the libraries (the files with names ending in `.dll`) with a names starting with `JRC.` or `RIO.` or `TAD.`

The IDSL functionality is provided by the library `JRC.IDSL.dll`.

To ease the user's experience, a few commands [\(to be launched in the bash shell of the raspberry\)](#) are available to manage the RIO system:

```

kr:          stops any running RIO instance

lr:          shows continuously the log created by the RIO system

rioset:     starts an editor to edit the Settings file

```

`riotel:` opens an interactive session with the local RIO system

In order to have these commands in your system, please follow the instructions provided in Appendix A.

Keep in mind that editing the Settings file does not change the behaviour of the running instance: to let the new settings be used, save the file and stops the RIO. It will start in a minute or less using the new configuration.

Using `rioset`, start an editor session. The Settings file will list both the general settings and at least one settings section for each functionality. The best part of these sections will include

```
"enabled": false
```

This means the functionality, though configured, is not used.

The main component of an IDSL based on RIO, but there are additional components to read the device status, to acquire additional readings, and to perform scheduled activities.

General settings

At the beginning of the file, usually there is the Id of the system. It is common practice to use the hostname, but it is not mandatory.

The Id will be given the initial value `RIO-Uninitialized_device`, change it to needed string.

At the bottom of the file, there is the rest of the RIO system configuration: every time RIO saves the Settings, it will use again this order: Id at the top, followed by all specific settings for the installed features, then the general settings, that are explained below.

- **Queue:** this is the connection string to a REDIS cache system used to queue the messages to and from RIO; using `ssl` is strongly advised. Both the queue and the credentials should be requested to JRC if the JRC TAD_server is used for storing the data.
- **QueueCredentials:** if needed (best practice), the password protecting the REDIS cache
- **WebAccess:** the URL where to post the telemetry information in case of REDIS failure
- **LocalManagement:** if true, activates a local listener on port 4005 to allow interactive sessions on RIO
- **EnableSlack:** if true, it activates interactions with slack and the RIO will report through it booting, errors and the like; it will also have perform commands with some limitations
- **SlackToken:** the token of the Slack channel to refer to

IDSL settings

Look for the section in the file related to the IDSL module, that looks like this:

```
{
  "Enabled": false,
  "Id": "IDSL",
```

```

    "Type": "Idsl",
    "Properties": {
      "Port": "/dev/ttyAMA0",
      ...
    }
  },

```

Enabled must be true to use the IDSL component of the system.

The Id allows differentiating several instances of the same type. The common practice uses IDSL.

The Type is there to identify which plugin will manage this part of the device and must not change.

The table below shows all settings used by the IDSL module.

Name	Default	Type	Description
Port	COM1	string	Address of the serial port (usually, /dev/ttyAMA0)
Output		string	If it is a number, it will be used as a network port where to accept clients: all readings will be repeated to all connected clients. Otherwise, it is used as a serial port name where to write all data read from the sensor.
Speed	9600	int	Baud rate for the Serial port
ShortWindow	60	int	Number of points for Short Term Forecast
LongWindow	600	int	Number of points for Long Term Forecast
Ratio	4	float	RMS ratio (see ref)
Threshold	0.1	float	Threshold for activation of calculation (m), see Ref
Period	5	float	Acquisition interval (seconds)
AddRMS	0.1	float	Addition to LTF-STF difference (m, see ref)
BackFactor	0	int	Not used, obsolete
SensorMultFac	-1	int	Multiplication factor for the level
SensorAddFac	0	float	Addition factor for the level
SonarMinLevel	0.3	float	Minimum measurable level of the Level sensor
SonarMaxLevel	5	float	Maximum measurable level for level sensor
SonarMaxDifference	0.5	float	Maximum difference above which the measure is considered an outlier
MaxDelay	60	int	This is the number of seconds after which the algorithm is reinitialized, if no data was acquired
SaveAllData	AllData_{0:yyyy-MM-dd}.log	string	If present, this template is used as the name of the file where to store

			all data read from the sensor. It can be a complete path, starting with '/'. If present, this file is used to preserve the status of the algorithm and used to reinitialize it when starting. If older than MaxDelay seconds, it is not used and deleted. It will always be located under /tmp.
DumpBuffer	buffer.txt	string	

Additional readings

Additional information are retrieved from the device using another module, `JRC.Power.dll`.

They will send their readings separately, depending on the desired time interval. This is an example of the same module, `AnalogMeasure`, used for two different tasks.

Below, the configurations of the three readings will be described.

The module `AnalogMeasure` is configured to read from an I2C channel every 180 seconds, 3 minutes, and convert the raw value into a Voltage measure of the battery charge status.

```
{
  "Enabled": true,
  "Id": "Battery",
  "Type": "AnalogMeasure",
  "Properties": {
    "Measure": "Voltage",
    "Frequency": "180",
    "Channel": "1699C",
    "Multiplier": "5.5",
    "Offset": "0"
  },
  "Version": "1.1.1"
}
```

In the following case, the same module `AnalogMeasure` is configured to convert the raw value read from another channel every 60 seconds into the temperature external to the device box.

```
{
  "Enabled": true,
  "Id": "AirTemperature",
  "Type": "AnalogMeasure",
  "Properties": {
    "Measure": "Celsius",
    "Frequency": "60",
    "Channel": "16AFC",
    "Multiplier": "-56",
    "Offset": "93.65"
  },
  "Version": "1.1.1"
}
```

The thermometer component is subject to some production variability. It can happen it appears on a different I2C location, for instance. Use `i2cdetect` to find it and put the location in the Channel property as second and third hex digit: "Channel": "1hhFC".

For the same reason, the Offset and the Multiplier can be different. To tune the sensor, use two random values, e.g. $Offset_0 = 10$ and $Multiplier_0 = -20$; then measure twice the temperature with an alternative mean and record the measurements generated by the device. Let the measured temperatures be T_1 and T_2 , and their difference ΔT . Let the measurements from the device be D_1 and D_2 , and their difference ΔD . Let the arbitrarily chosen offset and multiplier be $Offset_0$ and $Multiplier_0$. These equations will provide the operational values for `Offset` and `Multiplier`.

$$Offset = T_2 - \frac{\Delta T}{\Delta D} \cdot \frac{D_2 - Offset_0}{Multiplier_0}$$

$$Multiplier = \frac{\Delta T}{\Delta D}$$

To read the temperature of the CPU of the Raspberry, it is used a different module of the same library: `CpuTemp`, that acquires it from a system generated file.

```
{
  "Enabled": true,
  "Id": "CpuTemperature",
  "Type": "CpuTemp",
  "Properties": {
    "Period": "900",
    "Scale": "0.001",
    "Path": "/sys/class/thermal/thermal_zone0/temp"
  },
  "Version": "1.0.0"
}
```

Scheduled activities

The RIO systems use two schedulers at the time: one is used to analyse and react to messages on the alert channels and is used mainly by alerting devices (e.g. Tsunami alerting panels or sirens) and the other is used to schedule periodic activities.

Ruleset.json is used for the alerting devices, while crontab.json is used for all tasks.

The crontab file is divided in two sections: schedules and commands. In both the configuration and the last readings of all the RIO modules are available.

Schedules

This section allows defining when the operations described in the second section must be performed.

To interpret a line like the following

```
* * * * * 900 Battery_Voltage < 11.5 lowBattery
```

The structure is:

Second: the system time second must match this value(s), 0-59

Minute: the system time minute must match this value(s), 0-59

Hour: the system time hour must match this value(s), 0-23

Weekday: the system time day of the week must match this value(s), Mon-Sun

Day: the system time day of the month must match this value(s), 1-31

Month: the system time month of the year must match this value(s), 1-12

Delay: after a successful match, the schedule must not be evaluated before this number of seconds has passed

Condition: this is optional and allows a complex combinations of conditions (including parenthesis, mathematical expressions and AND and OR operators) to be evaluated together with the schedule: if both matches, the operation is performed. In the example, the last value assigned to the reading Voltage of the module Battery is verified.

Command: the name in the next section of the operation to be performed if the schedule is evaluated successfully

The example checks every 15 minutes (900 seconds) that the Voltage of the Battery is not below the threshold of 11.5 V. if it is, the lowBattery command is executed.

Every quantity of the schedule can be:

- A *, that matches any value
- A value, that is matched by the same quantity
- An interval, that matches all values from the first to the last
- A comma separated list of values, of which one must match
- Two quantities separated by / meaning that either the first value or the same with an arbitrary quantity of the second value will match

This means that 0,15,30,45 for seconds or minutes has the same effect of 0/15.

Commands

Using the same names occurring in a schedule, in this section are listed the operations to be performed, given the module to be used, the name of the command and all the parameters.

The lowBattery command of the above mentioned example will be:

```
"lowBattery": {
  "Target": "SlackManager",
  "Command": "send",
  "Parameters": {
    "channel": "rio",
    "message": "Battery low: $Battery_Voltage",
    "symbol": ":warning:"
  }
}
```

It will request the SlackManager, the module in charge of using Slack to send messages, to perform the send command, using the given parameters.

If the SlackManager is not enabled, the operation will fail. All operations executions are reported in the log file.

4 OTHER SETTINGS

4.1 SMS COMMANDS FOR RPI

To finalize the connection between the Raspberry and the Raspberry Nano and to allow executing commands via SMS, it is possible to launch the following commands; the system will ask several times the Teltonika password:

```

ipTeltonika='192.168.1.1
cd /home/script
sudo echo $HOSTNAME > sms_config.txt
ip=$(ifconfig eth0 | sed -En 's/127.0.0.1//;s/.*/inet (addr:192)?([0-9]*\.)([0-9]*).*\/\2/p')
echo $ip>sms_IP.txt
scp sms_config.txt root@$ipTeltonika:/sbin/sms_config.txt
scp sms_IP.txt root@$ipTeltonika:/sbin/sms_IP.txt

scp sms_command.sh root@192.168.1.254:/sbin/sms_command.sh
ssh root@$ipTeltonika rm /root/.ssh/id_rsa

echo "#! /bin/sh" > sms_init.sh
echo 'mkdir ~/.ssh'>> sms_init.sh
echo "dropbearkey -t rsa -f ~/.ssh/id_rsa">> sms_init.sh
echo "crontab -l>crontab.txt">> sms_init.sh
Echo "echo '*1 * * * * /bin/sh /sbin/sms_command.sh read > /root/sms_log.txt'>>crontab.txt">>sms_init.sh
echo "crontab crontab.txt">>sms_init.sh
scp sms_init.sh root@$ipTeltonika:sms_init.sh
ssh root@$ipTeltonika 'chmod +x sms_init.sh;sh sms_init.sh'
echo 'copy the Teltonika public key in the authorized keys (from ss-rsa... to RUT230.com)'
cd /home/pi/.ssh
echo 'vi /home/pi/.ssh/authorized_keys'

```

Copy the public key of Teltonika in the file `/home/pi/.ssh/authorized_keys`

The public key is: the section printed on the screen between `ssh-rsa...` and `RUT230.com`. For example in the case below is the red section. Copy and paste it in the file

`/home/pi/.ssh/authorized_keys`

```

pi@IDSL-00:/home/script $ echo "#! /bin/sh" > sms_init.sh
echo 'mkdir ~/.ssh'>> sms_init.sh
echo "dropbearkey -t rsa -f ~/.ssh/id_rsa">> sms_init.sh
echo "crontab -l>crontab.txt">> sms_init.sh
echo "echo '*1 * * * * /bin/sh /sbin/sms_command.sh read > /root/sms_log.txt'>>crontab.txt">>sms_init.sh
echo "crontab crontab.txt">>sms_init.sh
scp sms_init.sh root@$ipTeltonika:sms_init.sh
ssh root@$ipTeltonika 'chmod +x sms_init.sh;sh sms_init.sh'
echo 'copy the Teltonika public key in the authorized keys'
cd /home/pi/.ssh
root@192.168.1.254's password:
sms_init.sh                               100% 189    11.4KB/s   00:00
root@192.168.1.254's password:
mkdir: can't create directory '/root/.ssh': File exists
Generating 2048 bit rsa key, this may take a while...
Public key portion is:
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQCEUR5dV8uuSnhqRIHwz8qXPyeZw0L5r15I5JeTANZKGr1Jo0b0U9AUbqDVg+hP/i1KW9MYuq
3tZ9GNZSygS381XlgWkQccjq+uL TWG49+HX/my0+ZZFL VvcUEubDg2MvJStc02tfSmeI207AtfQ00vfLvz7PGYGoI2egC5dcjmsksrtUTf6Pt3W4q
JEsBEDWVQ06ZjLth60w+P3TL4Knougn8MUS2n0NowS4qMenCneqmun4naiL9rP6gprD0Y+7n8YUUK0WPu8xPDNeSnbP3BoalVB1UDV6ukp7Hf64yS
DjrfGcyLQ+HdJBSJKSbRwepHL7T/H3Kh215XPTs2v root@Teltonika-EcmLRUT230.com
Fingerprint: snai!! 86:72:57:88:12:48:40:89:00:00:02:9C:73:00:00:10:67:05:D8:13
copy the Teltonika public key in the authorized keys
pi@IDSL-00:~/.ssh $ vi authorized_keys

```

Give permission 600 to `/home/pi/.ssh/authorized_keys`:

```
chmod 600 /home/pi/.ssh/authorized_keys
```

```
chmod og-w /home/pi
```

Login into Teltonika router from client machine:

```
ssh root@192.168.1.1
```

Check the connection from the Teltonika router:

```
ssh -i /root/.ssh/id_rsa pi@192.168.1.101 ls
```


In case the Raspberry Pi adopts the default configuration; otherwise, edit the IP address accordingly.

Check sending an SMS:

```
wget "http://192.168.1.1/cgi-
bin/sms_send?username=user1&password=user_pass&number=00ccxxxxxxxx
xxx&text=IDSL-19" -O out.txt
```

Where `cc` is your country code and `xxxxxxxxxxx` is the phone number that will receive the SMS.



If the command fails, check the allowed numbers (phone numbers authorized to send/receive SMS in the Teltonika device) in the Teltonika web interface, by sending a SMS through the services offered by the Teltonika software. Open the web site: <http://192.168.1.1> and navigate to the appropriate SMS services. If the SMSs is not sent either, probably this service is not allowed by the Telecom company.

How to use the system:

Send an SMS to the number of the SIM on the device, with one of the available commands:

```
CMDTELT [cmd] to execute commands on Teltonika
```

```
CMDRPI [cmd] to execute commands on Raspberry
```

The result will be sent back by SMS to the sending number.

The numbers that are allowed to issue commands are identified in the file `sms_command.sh` and must be modified before performing the above procedure or need to be updated in the Teltonika router. Search for this piece of code:

```
# number authorized to send commands
```

```
ALLOWED_NUMBER1="+393299662159" # this is an exampe...put your number
```

```
ALLOWED_NUMBER2=""
```

```
REPLYNUMBER="+393299662159"
```

And include up to 2 numbers.

4.2 UPDATE CRONTAB

Finally, the `crontab` should be updated to start the software after the boot and to check that the software is running. An example of `crontab` is included in the gitHub repository in the section 'other files'. It is also available in the folder `pyTAD-main/other files`.

If you have installed the python version you should have the file below. If you did not, just download the gitHub file and unzip the file (first 4 lines of the chapter 3.2).

To load the `crontab` in the system launch the command:

```
sudo crontab '/pi/home/pyTAD-main/other files/crontab.txt'
```

5 SAVING THE PREPARED IMAGE

Once you have created your image of the raspberry or simply if you want to store the image received from JRC and personalized for your specific device, it is possible to do it by saving the image using the same software (Win32DiskImager) that was used to flash the SD card. However the problem is that the size of this image will be related to the whole size of the SD card used (i.e. 16 GB). When you later will try to flash another card with this same image it may happen that if the size of the receiving SF card is not just a bit larger than this one it will not fit in.

For this reason a software can be used that creates a shrink image much smaller than the size of the SD card.

Download the GitHub set of commands:

```
wget https://github.com/seamusdemora/RonR-RPi-image-utils/archive/refs/heads/master.zip
```

Extract the zip:

```
unzip master.zip
```

Insert a USB memory card in the raspberry. The size should be at least 4 GB to get the image of the raspberry. Prepare the raspberry to read the usb by installing the ntfs protocol (<https://raspberrypi.com/mount-usb-drive-raspberry-pi/>):

```
sudo apt install ntfs-3g
```

Create the mount point

```
sudo mkdir /mnt/usb
```

In general the usb should be identified as /dev/sda1. Using this command it is possible to see the name of the USB card:

```
sudo ls -l /dev/disk/by-uuid/
```

If the name is sda1 you can mount the USB

```
sudo mount /dev/sda1 /mnt/usb -o uid=pi,gid=pi
```

Now the USB is present as a file system under /mnt/usb and you can give the command below to produce the image. Please use all the default replies proposed by the system; the command will take several minutes to complete:

```
cd /home/pi/RonR-RPi-image-utils-master
```

```
sudo image-backup
```

```
➔ Image file to create?
```

```
reply with a filename on the /mnt/usb such as :  
/mnt/usb/image_raw_2023-05-18.img
```

6 WEBCAM INITIALIZATION

The webcam contains a Raspberry Zero W; its initialization is like the Raspberry present in the IDSL but it is simpler because not all the steps are necessary.

6.1 DOWNLOAD THE IMAGE OF THE BASIC OPERATING SYSTEM OF THE RASPBERRY W

The SD card initialization can be performed in 2 ways:

- Starting from a prepared image from JRC (this is the fastest way)
- Downloading a basic image from the Raspberry site (this could be necessary either if the JRC site is not available or if a latest version of Raspberry requires a new image).

6.1.1 DOWNLOAD A PREPARED IMAGE FROM JRC

Note:

If the device arrives with a preinstalled image this step and the following one are not needed. Proceed to section 6.2.

To create a new SD card it is necessary to have:

- - The image of the IDSL
- - An image writer software
- - An SD card reader (it could be external to a laptop or part of a laptop itself)

The image can be retrieved from JRC together with an image writer, such as Win32DiskImager (<https://win32diskimager.org/>) or any other equivalent software.

At this web site the latest IDSL image can be found, together with the software for writing the image of the SD card:

<https://idslarchive.z6.web.core.windows.net/Webcam.OS.zip>

Using the disk image writer and a SD card reader it is possible to transfer the image on the SD card and go to section 6.2.

After that, as in the case of the Raspberry, proceed to assign a hostname and to install the VPN.

6.1.2 DOWNLOAD A NEW IMAGE FROM SCRATCH FROM THE RASPBERRY WEBSITE

Alternatively, it is possible to download the latest image from the Raspberry official web site and perform the preliminary operations described below:

- Download and write the image on the SD card
- Install other needed software
- Install the VPN

To download the SD image follow the same procedure described for the Raspberry PI in chapter 2.2.



In this case it is particularly important to correctly setup the WIFI at the startup because this will be the only way that the Raspberry can communicate with the webcam. In this case the WIFI is offered by the Teltonika router, so the SSID will be "Teltonika" or "TeltonikaEcml" and the password will be either 1102lmcE or Ecml2011. For the correct ssid and password try to connect to it with a laptop and mark down ssid and password.

If you do not follow the automatic method offered by the raspberry programme, as described in chapter 2, you can also in this case create a text file named `wpa_supplicant.conf` in the boot section of the SD card (editable in Windows), with the following content:

```
country=us
update_config=1
ctrl_interface=/var/run/wpa_supplicant

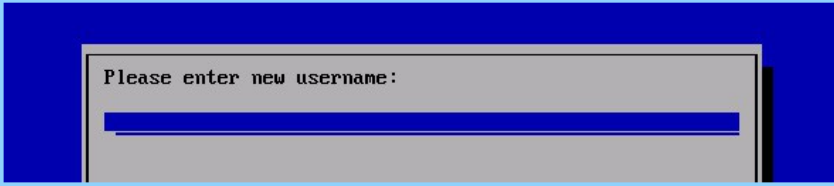
network={
    scan_ssid=1
    ssid="MyNetworkSSID"
    psk="Pa55w0rd1234"
}
```

In `ssid` and `psk` include the `ssid` you marked down and the related password.

Be careful that the EOL method is for Unix (or Linux) and not Windows. This can be done using an editor such as Notepad++ which allows to establish the EOL method.

The first time that the RPI Zero W will boot, it will resize the file-system to the size of the card. With the new OS (Operating System) version (at the moment it is the Raspbian 11 Bullseye 32 bit), it will ask for a username and a password (default user *pi* and password *raspberry*) if not configured through the advanced options in chapter 2.2.

Note: Same thing as with the desktop version, it's now required to create the first username and password on your first boot.



Please enter new username:

The RPI Zero W has a Mini HDMI port, so it is necessary to have an HDMI to mini HDMI cable



To use a keyboard it is necessary to have a micro to USB-A adapter:



Connect the power charger (microusb) to the first port. The power supply should provide 1.2A (see [Power Supply specs](#))



Connect the camera to the CSI camera connector (v1.3 only)



Impose a static IP following the instruction at 2.2.4 and impose 192.168.1.175 as address of the wifi (wlan0) and IP 192.168.1.175.

Edit `dhcpcd.conf` with

```
sudo nano /etc/dhcpcd.conf
```

Add the static IP settings to the end of the file like this

```
#static IP configuration
interface wlan0
static ip_address=192.168.1.175    ## this will be your IP
static routers=192.168.1.1
static domain_name_servers=192.168.1.1
```

Then prepare the SD card to install the needed software:

```
sudo apt-get update
sudo apt-get upgrade
```

Install the software dependencies

```
sudo apt-get install bc
sudo apt-get install imagemagick
sudo apt-get install ftp telnet
```

The camera is already enabled on the latest OS.

Now you should be able to take pictures with the command:

```
libcamera-jpeg -o firstImage.jpg -t 2000
```

The official documentation is available at:

<https://www.raspberrypi.com/documentation/accessories/camera.html>

6.2 COPY THE SOFTWARE AND PREPARE IT FOR THE EXECUTION

The only software to install are a few scripts and the crontab configuration, that allows taking one picture at a fixed interval (1h or 15 min or whatever you prefer). The images are then uploaded on the JRC web site; but it is possible to modify the location for the upload.

Download again the whole package of pyTAD from:

```
wget https://github.com/annunal/pyTAD/archive/refs/heads/main.zip
```

unzip main.zip with the command:

```
unzip main.zip
```

Identify the folder:

```
/home/pi/pyTAD-main/scripts/webcam
```

Create a folder script under /home and copy there all the content of the above folder:

```
mkdir /home/script
cp /home/pi/pyTAD-main/scripts/webcam/* /home/script
chmod +x /home/script/*.sh
```

The crontab is available as one of the files under /home/script. To impose it use the command:

```
sudo crontab /home/pi/pyTAD-main/scripts/webcam/crontab.txt
```

6.3 CONNECT THE RPI AND THE ZERO W

In order to allow the Raspberry PI to perform the script 'scatta.sh' on the Raspberry Nano, it is necessary to exchange public and private keys between the two devices so that a ssh command can be provided to perform the shooting.

To execute a command on Raspberry PI Zero invoked by a script on Raspberry PI B, it is necessary to allow the authentication from the Raspberry PI B to the Raspberry PI Zero with public-key/private-key configuration

- Generate an RSA key pair on the Raspberry PI B with the command

```
ssh-keygen
```

Answer questions with all default values

- Check creation of public key:

```
cat ~/.ssh/id_rsa
```
- Copy the Public Key to Raspberry PI Zero:

```
ssh-copy-id pi@192.168.1.X
```

(where x is the 4th octet of the raspberry PI Zero Address usually .175)

- Connect to the Raspberry PI Zero:

```
ssh pi@192.168.1.x
```

- It is also possible to execute a direct command:

```
ssh pi@192.168.1.x ls (to list the home folder)
```

- In order to store the fingerprint of the webcam Raspi Zero, it is necessary to launch once the following command

```
sudo ssh -i /home/pi/.ssh/id_rsa pi@192.168.1.175 /home/script/scatta.sh
```

6.4 INSTALL THE VPN

Follow the instructions contained in chapter **Error! Reference source not found.**

7 FIRST SWITCH ON OF AN IDSL

The first time that the device is switched on, a few operations are necessary:

- Check the config.txt file
- Verify that all software is running
- Verify that the sensor is providing data

Verify that the solar panel is working

7.1 CHECK THE CONFIG FILE

If the c version is used check that a fine config.txt is present in /home/pi/programs/TAD0

If the pyTAD version is used, open the config.txt file, under /home/pi/programs/pyTAD/progs and verify that the config.txt file is present.

If the RIO version is used, check the Settings.json file in /home/pi/programs/RIO.

If the file is present, verify that the serial port is specified correctly as **/dev/ttyAMA0**

Check that the software is not in simulated mode, otherwise a constant value is sent to the server: **simSonar=0** for TAD and pyTAD. The RIO version does not allow this mode, since it has a simulation module for this use.

Adjust the sensorAddFac value to have a reasonable level. However, this value can also be changed at a later stage if a more precise calibration value is available. The country reference is often used to align all sensors: use this parameter to achieve that.

7.2 VERIFY THAT THE SOFTWARE IS RUNNING

Verify that the software is running with:

- TAD: `ps -efd | grep tad`
- pyTAD: `ps -efd | grep python`
- RIO: `ps -efd | grep TAD.Core`

7.3 VERIFY THAT THE SENSOR IS PROVIDING DATA

To check that the sensor is providing data, check the log files written in the /tmp folder:

- pyTAD writes in /tmp/pyTAD two files:

AllData... current date... .txt and execLog...current date... .txt

If they are present, check that they are periodically updated with the command:

```
tail -f AllData...
```

```
tail -f execLog...
```

- TAD writes in /tmp/TAD the same two files: check with the same commands
- RIO writes a daily log in /tmp/checkLog-`{yyyyMMdd}` .txt, that can be watched with the command `lr`. In the RIO settings, the `SaveAllData` parameter is used to template the name of the related file.

The AllData file should append data every second or less, while the log according to the time interval specified (5s by default). If the files are not updated, use the minicom utility and connect directly to the sensor through the defined serial port. In case data do not arrive correctly on the serial port, refer to the maintenance document.

7.4 VERIFY THAT THE SOLAR PANEL IS WORKING

When an IDSL is installed for the very first time, it is a good practice to check that the solar panel is working. In a very rare case it happened that the solar panel cable was disconnected and it was not charging. The best moment to verify the behaviour of the battery voltage is the early morning.

From the night to the day the solar panel output increases: this guarantees that the solar panel is charging the batteries.

For this reason, it is a good practice to remain by the installation site overnight to verify the correct charging.

8 IDSL REMOTE VERIFICATION

The remote verification is very important and allows to verify that everything is working fine and to prevent problems.

- Verify that the software is running
- Verify that the disk is not full
- Verify that the sensor is providing data
- If the webcam cannot be reached try the switchOffOn command

For these procedure to be performed, it is assumed that a VPN was configured to communicate with the device.

8.1 VERIFY THAT THE SOFTWARE IS RUNNING

Proceed as described in 7.2.

8.2 VERIFY THAT THE DISK IS NOT FULL

In principle nothing is written on the memory card. It is good practice to check periodically the amount of the space left on the temporary partition /tmp and on the /home partition using the command `du -h`

8.3 VERIFY THAT THE SENSOR IS PROVIDING DATA

If data are not uploaded, perform the same checks of 7.3.

8.4 THE WEBCAM CANNOT BE REACHED

Sometimes the webcam does not connect with the Teltonika router. In order to check, try to connect from the RPI using ssh to the IP 192.168.1.175. If it cannot be reached, switch the whole system off and back on by activating the script:

```
/home/script/switchOffOn.sh
```

This script will activate the LAN switch that will power off everything, including the RPI and after 2s will power on the device. Of course, it will not be possible to control the device until it completes the boot sequence. This procedure is rather invasive and should be limited as much as possible, because the device is switched off abruptly.

9 CONCLUSIONS

The document provides all the elements to perform a safe start-up and verification of an IDSL system. Several details are specific of the JRC design; but it is easy to adapt them to a different implementation.

It is better initializing an IDSL system in a laboratory, where the network is more reliable and it is possible, if needed, to download more software or a new image SD card. Once the initialization and the testing are completed, the device can be deployed onsite.

Still, using the remote management, it is possible to correct, modify and restart the system if the VPN is properly established.

10 APPENDIX A – BASH COMMANDS ALIAS FOR RIO

The commands to manage the RIO System need to be configured into the `/etc/bash.bashrc` file. Edit this file with:

```
sudo nano /etc/bash.bashrc
```

and add the following list of commands alias. Then save and reboot the system.

The list of the command is

```
# Mount filesystem in Read Only
alias ro='sudo mount -o remount,ro / ; sudo mount -o remount,ro /boot'
# Mount filesystem in Write
alias rw='sudo mount -o remount,rw / ; sudo mount -o remount,rw /boot'

# setup fancy prompt"
PROMPT_COMMAND=set_bash_prompt
# Edit bash.bashrc
alias ebashrc='sudo nano /etc/bash.bashrc'
# Kill RIO
alias kr='pidof TAD.Core | xargs sudo kill'
# Tail Rio Log
alias lr='date +/tmp/checkLog-%Y%m%d.txt | xargs tail -f'
alias egrep='egrep --color=auto'
alias fgrep='fgrep --color=auto'
alias grep='grep --color=auto'
alias ls='ls --color=auto'
# Uptime
alias upt='uptime -p'
# Edit Rio Settings.json
alias rioset='nano /home/pi/programs/RIO/Settings.json'
# Restart Logmein
alias lmrestart='sudo systemctl restart logmein-hamachi'
# Check Logmein
alias lmcheck='sudo hamachi'
alias ebashrc='sudo nano /etc/bash.bashrc'
alias riotel='telnet localhost 4005'

(echo ' Currently:' | tr "\n" ' ' ; date +"%Y-%m-%d %k:%M:%S" ; echo
' Up Since:' | tr '\n' ' ' ; uptime -s ; echo ' Duration:' | tr '\n'
' ' ; uptime -p)
# Show Banner Alias
alias balias=/etc/update-motd.d/11-banneralias
# Edit Banner Alian
alias          ebalias='sudo          nano          /etc/update-motd.d/11-banneralias'
```

the file `/etc/update-motd.d/11-banneralias` contain the list of the alias, it is showed at the login
The content of the file `/etc/update-motd.d/11-banneralias` is

```
#!/bin/sh
# alias ro='sudo mount -o remount,ro / ; sudo mount -o remount,ro
/boot'
# alias rw='sudo mount -o remount,rw / ; sudo mount -o remount,rw
/boot'
```

```

# alias kr='pidof TAD.Core | xargs sudo kill'
# alias lr='date +/tmp/checkLog-%Y%m%d.txt | xargs tail -f'
# alias upt='uptime -p'
# alias rioreset='nano /home/pi/programs/RIO/Settings.json'
# alias riotel='telnet localhost 4005'
# alias ebashrc='sudo nano /etc/bash.bashrc'
# alias lmrestart='sudo systemctl restart logmein-hamachi'
# alias lmcheck='sudo hamachi'
# alias balias=/etc/update-motd.d/11-banneraliases
# alias ebalias='sudo nano /etc/update-motd.d/11-banneraliases'
# alias ebashrc='sudo nano /etc/bash.bashrc'

```

```
green="\e[32m"
```

```
reset="\e[39m"
```

```
#####
```

```
echo "## ALIAS ##" #
```

```
#####
```

```
echo "${green} # ro : ${reset} To mount filesystem in ReadOnly Mode"
```

```
echo "${green} # rw : ${reset} To mount filesystem in Write Mode"
```

```
echo "${green} # upt : ${reset} To display uptime "
```

```
echo "${green} # kr : ${reset} To kill RIO program "
```

```
echo "${green} # lr : ${reset} To display Current Log of RIO "
```

```
echo "${green} # rioreset : ${reset} To edit the RIO Settings.json "
```

```
echo "${green} # riotel : ${reset} To configure RIO "
```

```
echo "${green} # lmrestart ${reset} To restart logmein-hamachi"
```

```
echo "${green} # lmcheck : ${reset} To check hamachi"
```

```
echo "${green} # ebalias : ${reset} To edit Banner Alias "
```

```
echo "${green} # ebashrc : ${reset} To edit bash.bashrc "
```

```
echo "#####"
```