July 19th, 2022. Installation guide for domotica_v47

New in this version are:
- Machine to machine communications now includes ESP8266 and ESP32 MCU's.
- When the web-server runs in a separate thread it no longer locks the console, in interactive sessions under Gforth.
- The UDPport 8899 is used for all UDP messages.
- Parallel ping under Gforth has been replaced bij the use of the arptable.
- Adapted for the Bullseye (Kernel: Linux 5.15.32+)
- Removed the extra queues.

The aims for domotica_v5.2 are:
1.To be able to control devices through a local webserver.
2. The local webservers can communicate to each other to exchange data or to control their connected device(s) over UDP or TCP. MCU's with their webserver can also be used.

This guide was written to get a Raspberry Zero W running headless on a local network.
Warning: I could only test the guide on my own network.
So, you might have to find solutions yourself in case of problems.

The guide expects from you some skills. If you don't know much about Forth see:
https://iforth.nl/sf/sf.html
If you don't know much about IP addresses see:
https://www.youtube.com/watch?v=9FzKxMiZWgg and also the manual of your router.
If you have never used Linux in a bash shell see:
https://www.youtube.com/watch?v=oxuRxtrO2Ag
If you don't know much about HTML see:
https://www.youtube.com/watch?v=v4oN4DuR7YU and the extensive:
https://www.w3schools.com/html/default.asp

In this manual black **bold underlined** text means: there must be an option you can see in the menu, desktop or screen.
$ text means type in a bash shell the **text** after the dollar sign and press <Enter>.

The master-slave part only works on a local network under Gforth on a Raspberry Pi with the Linux and in a BASH-shell. See https://youtu.be/xlzHTPlj8bo for a demonstration.
The Web server light can be used with Win32Forth version 6.15 (compile _demo1.f) or with Gforth using version 0.7.9_20220713 on a Linux PC or Raspberry Pi.

Minor changes in various updates of Gforth or Linux might lead to problems. 10-6-2024: The Bookworm failed me.
It was last tested on Gforth version: **Gforth 0.7.9_20220713** and the Bullseye (Kernel: Linux 5.15.32+)
Your WiFi network should be stable. Speed is less important.
For me the following setting in my router did the trick:
Transmission mode: **11bg mixed** (Try to use as less protocols as possible)
Channel width: **20Mhz**
Channel: **1**

Protection: **WPA2-PSK**
Encryption: **AES**
Beacon interval: **100**
RTS threshold: **576**
DTIM interval: **1**
Connect as many as possible devices to **5G** and set it's RTS threshold to **576**.

If your network connection disappears or hangs, then try the following:
Switch your router off, wait 10 seconds and switch it on again.
The Microsoft Edge often hangs.
Browser such as the Vivaldi of Vivaldi Techonologies or Chrome perform better.

Before you start read the entire guide so you know what to expect.
Start on your PC the CMD prompt and enter: **ipconfig /all** <ENTER>
Determine an IP-addres for your Pi. Use one that ends with a **zero** for the first Pi for an easy master-slave configuration.

Next follows how a Raspberry Zero W can be changed to run on a network
without a keyboard, mouse or monitor (headless)
The parts between [[and]] are optional.
As far as I could test they increase the stability of the Raspberry Pi which is needed
when you run a Pi 24/7. Try Google to find out if these parts or other parts that are unclear.

In order to install or clone a Raspberry Zero W in an easy way the following extra hardware is used:
A Transcend USB 3.0 SDHC/SDXC / microSDHC/SDXC Card Reader, TS-RDF5K
to put files on a SDcard.
A mini HDMI to HDMI adapter to connect the Pi to a monitor.
A microUSB OTG cable.
A SDcard. The SDcards I used till now had a lifespan of about 1 year.
Now I am trying: A SanDisk SDSQXAF-032G-GN6AA. They issue a 30-year
warranty in Germany for this model.
A 5Vdc 1A power unit. (I use an old 5V power supply of a previous smartphone)
A PC with connected to the internet.
A NAS with a shared directory (Only needed when there are multiple Pi's used.)
I use a Linux PC that is connected to my router.

For Installation of the OS on the Pi go in the internet to:
https://www.raspberrypi.org/software/
1. Hit the link **Download for Windows** on your PC.
2. Place the the Transcend USB Device with a SDcard in the PC
3. Install and start the Raspberry Pi Imager.
4 Choose Left your Raspberry Pi Model,
9. Select at **OS**: first **Raspberry PI OS) (Other)**
and then **Raspberry PI OS (Legacy, 32bit) Lite with no desktop**

environment (Bullseye NOT Bookworm)

10. Select next at the **Storage**:
    The **Transcend USB Device** (holding the SDcard)
11. Choose **Next** and finish the wizzard. Overwrite the SDcard.

Hardware setup raspberry pi:
Connect a keyboard to the Pi through the microUSB OTG cable.
Connect the Pi to a monitor with the mini HDMI to HDMI adapter.
Place the SDcard in the Raspberry Pi
Connect the power unit to the Pi. The Pi is very sensitive to power fluctuations.
If you use a power bar then use the socket that is **closest** to the power line.

12. Wait until a reboot
13. Then enter the keyboard layout and
    enter a username (pi) and password

Important: Enable in your **router** the **DHCP server**
Be sure to use always the same determined IP address in your network by adding the PI in your router in **the Reservation list** at the **DHCP server**

Login to the Raspberry pi. Enter:
$ sudo raspi-config
Use the **cursor** keys to select an item then **<tab>** to select choises like **Ok**, **Cancel** or **Finish** and **<enter>** to activate the option.
Choose: **System options | Wireless Lan**
Select your country, fill in the SSD and its password,
Finish

Then in the black part, check the ip address enter:
$ **ifconfig** and see it under **wlan0:** at **inet**
Check the network with:
$ **ping -c3 8.8.8.8**
That should respond 3 times with something like:
64 bytes from 8.8.8.8: icmp_seq=1 ttl=59 time=13.1 ms
$ **pwd**
That should show: (restart when it does not)
$ /home/pi

$ **sudo raspi-config**
Enable SSH with: **Interface Options | SSH**
Enable SPI with: **Interface Options | SPI**
Enable I2C with: **Interface Options | 12C**
Choose: **Localisations Options | Timezone** Set it to the right zone.
Change the hostname at: **System options | hostname**
Change the password hostname at: **System options | Boot / Auto Login | Console**
Finish | reboot

$ sudo shutdown 0 -h
Wait till the powerled is off.
Disconnect the monitor and the keyboard.

The Pi is now able to run headless.

Download on a PC that also is connected to your network
the newest(Update it to avoid errors!) **MSI ('Windows Installer')** of **PuTTy** from:
https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html and install it on your PC.

Put power on the Pi so it starts again and wait 60 seconds.
Then start PuTTY on the PC and enter at the top **the IP address** of the **Pi** and use Port **22** with connection type **SSH.**
Then click on **Open**.
If a screen with a PuttY Security Alert about a fingerprint appears then hit **Accept**.

After you logged on you get into the BASH-shell on your Pi.
Logon and enter **$ nano test.txt** type the word test and save it.
Familiar yourself a bit with this simple editor.
See https://www.youtube.com/watch?v=f_nAA_cbSW4 for a quick tutorial.
The Nano editor is a safe way to change files that end with **.sh**
If you use notepad or another editor that produce a DOS format then these files will not work anymore.

For extra software and updates on the PI or Linux Debian PC:
$ sudo apt-get update
$ sudo apt-get upgrade -y

For ARP:
sudo apt-get install net-tools

To get access to your Pi with the Explorer on a PC:
$ sudo apt-get install samba -y
If a dialog window appears about the use of a DHCP-server then choose **[No]**
$ sudo apt-get install winbind -y

$ sudo nano /etc/samba/smb.conf
Changes in this file:
1) Find the line with
workgroup = **WORKGROUP**
Change WORKGROUP in your used workgroup of your **PC** if they are not the same.

2) Find the line with
read only
at **Share Definitions** under [homes]
Change it into:
read only = no

3) Add at the end of the file the line:
socket options = TCP_NODELAY IPTOS_LOWDELAY SO_RCVBUF=65536 SO_SNDBUF=65536 SO_KEEPALIVE
Then save the file.

Generate a password for pi in samba:
$ sudo smbpasswd -a pi

$ sudo apt-get install git
$ Do you want to continue? [Y/n] Y

Install the wiringpi **only** on the a Raspberry Pi for GPIO .
$ **git clone https://github.com/WiringPi/WiringPi**
$ **cd WiringPi**
$ **./build**
$ **gpio -v**
# Run gpio -v and version 3.6 or better will appear.
# If it does not appear, it means that there is an installation error.

[[Making a RAM disk on a **Raspberry Pi** :
Careful here 1 mistake and you need to repeat the whole installation
$ sudo nano /etc/fstab
Then add the following 6 lines at the end:
tmpfs /tmp tmpfs defaults,noatime,nosuid,size=50m 0 0
tmpfs /var/tmp tmpfs defaults,noatime,nosuid,size=10m 0 0
tmpfs /var/log tmpfs defaults,noatime,nosuid,mode=0755,size=100m 0 0
tmpfs /var/log/samba tmpfs defaults,noatime,nosuid,mode=0755,size=10m 0 0
tmpfs /var/spool/mqueue tmpfs defaults,noatime,nosuid,mode=0700,gid=12,size=30m 0 0
tmpfs /var/run/samba tmpfs defaults,noatime,nosuid,mode=0755,size=20m 0 0
Save the file with added 6 lines above
$ sudo reboot
If you decided that the IP adress of your Pi is 192.168.0.110
Then check in the windows explorer if you can open your
\\192.168.0.110\pi
(Use the pasword of samba)
Create a simple text file as a test.

Disable logging rsyslog
$ sudo systemctl disable rsyslog
See also after a reboot: **ls -l /var/log**

Get improved networking stability at the cost of some power consumption:
$ sudo nano /boot/config.txt
Add the following line at the end:
dtparam=eee=off
dtoverlay=disable-bt # Bluetooth hangs the serial interface
Save the file.

Disable swapping for a faster webserver.
$ sudo dphys-swapfile swapoff
$ sudo dphys-swapfile uninstall
$ sudo update-rc.d dphys-swapfile remove
$ sudo apt purge dphys-swapfile
$ After this operation, 69.6 kB disk space will be freed.
$ Do you want to continue? [Y/n] y

The Pi is going to use simple uni-cast DNS - multi-cast DNS support is not required.
$ sudo systemctl disable avahi-daemon
$ sudo systemctl stop avahi-daemon

]]

Download all the needed files:
$ **git clone https://github.com/Jos-Ven/A-smart-home-in-Forth**
Move the files to the home directory:
$ **mv A-smart-home-in-Forth/*.* .**

Install gforth-0.7.9_20220713for the webserver light:
$ mkdir Downloads
$ sudo chmod 777 Downloads
$ cd Downloads
$ unzip gforth.zip
$ tar xvfJ gforth.tar.xz
To see where the unpacked version is, enter.
$ ls
If the new directory is gforth-0.7.9_20220713 then:
$ cd gforth-0.7.9_20220713
$ sudo apt-get install m4
$ sudo apt-get install libtool-bin
$ sudo apt-get install libffi-dev

Ignore ALL errors of the following 4 commands
It will also take a long time and produce a lot of text.
$ sudo ./install-deps.sh # Ignore the fatal error when shown
$ sudo ./configure
$ sudo make
$ sudo make install
and wait for the line with: === INSTALL SUCCEEDED ===
$ hash -r

Reboot the Pi with: **$ sudo reboot**
Login to the Pi and enter **$ gforth**
As soon as you a line that looks like:
Gforth 0.7.9_20220713, Copyright (C) 1995-2018 Free Software Foundation, Inc.
Then you know the installation is indeed OK
Enter **Bye**

Grant all rights with: **$ sudo chmod 777 *.***
$ sudo gforth
Needs _DemoMaster.fs
remove /tmp/web.log \ If Forth can't create a logfile
ignore other errors
A line like: ' Webserver started at: http://192.168.0.207:8080/home '
should appear.
Go to the indicated page in your browser to see if it works.
<Ctrl+c> Breaks the lock in the console.
Exit forth with:
+f bye

My preferred way is to start Gforth in the back ground of Linux, then it will continue to run even if you logoff. To do this just start enter
$ ./gf.sh

[[More optimizations:

Autostart the webserver in the background , Enter:
$ crontab -e
Choose nano as the editor
Add at the end the line:
@reboot (sleep 10 ; sh /home/pi/auto.sh)
Then save and reboot and check if the webserver shows its home page.
]]

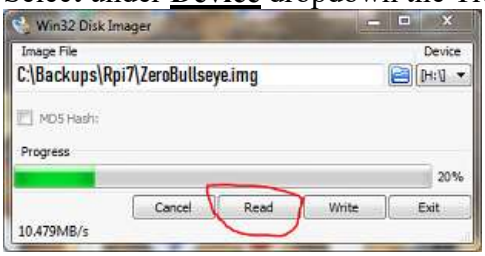
Now I recommend you to backup the entire PI by making an image of the SDcard.
After all a SDcard can de **crash** .....

Note: A restore only works when the new or blank SDcard has at least **the same size** as the SDcard that was used to backup the PI

Update the PI before the backup :
$ sudo apt-get update
$ sudo apt-get upgrade -y

Download, install the Win32DiskImager from https://sourceforge.net/projects/win32diskimager/ on a PC

1. **$ sudo shutdown 0 -h**
2. Disconnect the power cable when the systemled is off.
3. Put the SDcard in the Transcend USB Device and place it in the PC
4. Close all new explorers and do **NOT** format the suggested station.
5. Start the Win32DiskImager
6. Create the file path and name that end with **.img** of the image under **Image File**,
7. Select under **Device** dropdown the Transcend USB Device (Contains the directory: overlays)



8. Click "Read"
9. Wait until ready.
10. Eject the Transcend USB Device in the explorer.

Now, if anything ever goes wrong with your Pi, you can restore your fully-set-up image using the reverse instructions:

1. **$ sudo shutdown 0 -h**
2. Disconnect the power cable when the systemled is off.
3. Insert the SDcard in the Card Reader and place it into your computer.
4. Head to the search of the start menu and type "disk management." Open the disk management program and find your SDcard in the list.
5. Right-click and delete all the partitions on your SDcard. **Be sure** to select the partitions of **SDcard** only!
6. When it's empty, make one new partition of maximal 32 GB
7. Right-click on the new partition it and format it as **FAT32**. **Be sure** to select the partition of **SDcard** only!
8. Open Win32DiskImager again and browse for your image file. Select your device from the **Device** dropdown just as you did before. **Be sure** to select the **SDcard**.
9. This time, click "Write" to write the image to the SDcard.
10. When it finishes, eject the SDcard and re-insert it into your Raspberry Pi. When you boot it

up, it should be in the exact same state it was in when you first backuped the SDcard. Once you've done this, setting up your Pi from scratch will be a whole lot simpler and faster!

In this pack: All files that ends with **.f** or **.fs** can be compiled under Gforth.
Files ending with **.f** can also be compiled with Win32Forth and must be edited with an editor using a DOS format.
Since **.fs** files may also be edited in DOS format I use the win32forthIDE to edit **.f** and **.fs** files.

gf.sh is used to start a Gforth application under Lunix in the background.
See gf.log for errors.
Modify **gf.sh** if you would like to use another application or example.
A TCP or UDP port can only be one time used.
To see all gforth processes enter: **$ ps aux | grep gforth**
Then you could also use: **$ sudo killall gforth-fast** to kill it.
Or you can try: **$ ./kf.sh** to kill Gforth processes.

More details:
First kill all Gforth processes. A secondary start of the webserver at the same time is most likely to fail at the start.
Remove in the file **_DemoMaster.fs** the definition **demo-request** the first '\'
That activates the line with: 2dup. s cr type
Save the file.
Start PuTTY login to the Pi and place the window of PuTTY on the left side of your screen.
Start Gforth with **$ sudo gforth-itc**
Enter in Gforth: **needs _DemoMaster.fs**
Start your internet browser and put the browser on the right side of your screen.
Enter the adress of the webserver in your browser.

In Gforth you will now see what has been received if you choose an option in your browser.
That will be done by the definition **demo-request**.
demo-request is started by the definition **handle-web-packet** in the file
Web-server-light.f at the **Gforth** part and gets a memory buffer and its length of the received packet on the stack. **demo-request** expects a new memory buffer and its length containing a web page back. **IF** the length (lcount) is negative then the web server will not respond.

The last received packet from the Browser is saved. It can be seen with:
&RxPackets /received type

Each line in the definition **demo-request** is used to detect and respond to an incoming request.
ifttt$ will search for the string supplied in **recv-pkt$ cnt**
if found it will drop **recv-pkt$ cnt** and execute the next word.

That word should create a new html-file in a memory buffer.
EG: Enter **home-page .s** and you will see where it has been stored.
You could also try **htmlpage$ lcount .s type** to see what is in there.
And you could debug in it **gforth-itc** by using **dbg home-page** or use **break:** in a definition as usual in gforth-**itc**.
Entire received html packets will be searched by iftt$ definitions from the start to the end. The referrer field may disrupt a correct response. A space is used to avoid that.
Moving a line in upwards a list of " ifttt$ statements will give it a higher priority. EG:
s" /test1 " ifttt$ dotest1
s" /test " ifttt$ **dotest**
then **dotest** will never be executed since ifttt$ will find /test1 if it gets /test to search for.
Place the shortest strings followed by a **space** first in the list to avoid problems.
Also note: **recv-pkt$** and **cnt** are 2 times on the stack so it can be used again in dotest etc. **demo-request** expects the HTML-response + its count back.

Errors should appear in the logging of the console (File gf.log).
Note the times in the logging to be sure that you are not looking at an old logging.
If you crash out of Gforth then the only way out is to use the console and find out what is going wrong. That can happen with an uncaught exception.

The *.f files can be compiled under Win32Forth and Gforth.
The *.fs files can only be compiled on Gforth.
It should be possible to use the Web server light over the internet (not recommended) if you know how to forward incoming requests to the web server.
That should be explained in the manual of your router.

------- If you just use **ONE** Pi then there is no need to continue reading. -------

Cloning a **same Raspberry Pi model** can be used to install a secondary Pi.
You don't need to repeat the entire installation. Here is how it can be done:

Make a backup and restore it on a new SDcard
Place the restored SDcard in the new PI
Power up the new Pi without any other hardware connected to it.
You have already defined a WiFi connection on your first Pi.
Now the new Pi is going to connect to your router and behave totally as the first Pi.
Use **PuTTY** on your PC to logon to the new Pi (use the password of the first Pi)
To avoid 2 exact same Pi's on your network make the following

modifications:
$ **sudo raspi-config**
Change the pasword at: **System options | Password**
Change the hostname at: **System options | hostname**
Finish | reboot

[[Set a new password for samba: **$ sudo smbpasswd pi**
]]

Setting up the NAS:
Copy all the extracted files **EXCEPT gf.sh** from domotica_v4700.zip to the NAS.
Copy **gf.sh** directly to the slave(s).
The master will use **npush.sh** to copy files to the NAS.
The slaves will use **nget.sh** to copy files from the NAS.
These 2 scripts use the file: /home/pi/Documents/pw.sh
$ mkdir Documents
$ sudo chmod 777 Documents
$ cd Documents
Create **pw.sh** with nano containing the following 3 lines:
MOUNTING="//192.168.0.209/homes"
USR="pi"
PW="Password of pi on the machine of the NAS"
Most likely is that you have to change the used **IP address** and **password** between the quotes. The IP address schould be the IP address of the machine of the NAS
save the file, then:
$ sudo chmod 777 pw.sh
$ cd ..

Make a small file named **test.f** in your **/home/pi** directory on the master and enter **$ ./npush.sh**
Then see if test.f to has indeed been copied to the NAS.
On the slave enter **$ ./nget.sh**
$ ls test.f
to see that test.f has been copied to the /home/pi of the slave.
If one of these tests failed than you have to solve the problems before you continue.

Making the Pi a **slave**:
Change in the file **gf.sh** of the slave, so it is not a master when Gforth is started.
Put a **'#'** at the start of the line that contains **_DemoMaster.fs** so it will not start again.
Remove the '**#**' at the start of the line that contains **_down.fs.**
Save the file.
Start Gforth with **./gf.sh**

Disable NTP for slaves:
$ timedatectl | grep synchronized
System clock synchronized: **yes**
$ sudo timedatectl set-ntp False

$ sudo reboot
$ timedatectl | grep synchronized
System clock synchronized: **no**
The master will take care for the right time on the slaves.

An ip table is used for communications.
Specify the number of the used Pi's and other systems in the network on the **master**:
$ ./kf.sh to kill Gforth
In autogen_ip_table.fs there is a line with
10 constant FirstIpRange
Change the **10** into the number of Pi's you are going to use.
Delete the file **ip_table.bin** on the master
Start Gforth with **./gf.sh**

Start one or more slave and go to the home page of the master in your browser.
(Mine is at: http://192.168.0.201:8080/home) and choose:
Administration | **Update settings:** | the button **IP Table** at Update the ip table. |
back to **Administration** | then the button **UpdateAll** |
Then just wait a few seconds and hit **Scan** again. As soon as a new version number appears your slave is up again and ready to use.

The update and synchronize options are script driven.
When activated, Gforth writes a new nupdate.sh
Next Gforth uses **npush.sh** to copy nupdate.sh and the changed *.f files to the NAS.
*.f files that start with a **zero** are excluded. Then a message is send to the slaves.
They will use the following scripts:
1. Run nget.sh to copy all added/changed files from the NAS to their /home/pi directory.
2. Run nupdate.sh to execute more Linux scripts if they are there.
3. Run gf.sh to restart Gforth which will compile the web server and its application .

When you use the option **Update Gforth after download** you must copy a downloaded /gforth.tar.xz to your NAS first! It also assumes that there is an **/tmp** directory on your **RAM disk**.

Now you can also update an application without having to log on to a slave
and starting the Gforth-console manually.
In a network there should be only one Pi that is the master. All the other Pi's should be a slave. Otherwise a secondary master will not receive data from the slaves.
Updating can be done through the link **Administration** on the Master.
If a version number is negative it means an update is in progress or there is an error. See **nget.log** or **nupdate.log** on the slave for errors.
upd_gforth.log is created when Gforth is updated.
Master.fs changes a system into a master.

See the sources _DemoMaster.fs and _down.fs.

That is all.
