



Stage Ouvrier

Réalisé par : Alibi Mourad -G11

Encadré par : Mr ANIS Ellouze
CHEZ LA SOCIETE [PIVA Software](#)

Année universitaire 2015-2016

PIVA Software



Tél (+216) 74 611 029
Fax: (+216) 74 612 729

Adresse
Route Kaid Mhamed Km 4,5
Sfax 3062 Tunisie

Site web : <http://www.pivasoftware.com>
E-mail: sales@pivasoftware.com

Présentation de la société

Présentation générale de la société

PIVA Software est une société de services logiciels et solution créée en 2006 et spécialisée dans le développement de logiciels embarqués, assurance de la qualité et le développement mobile.

Produit et services

PRESTATIONS DE SERVICE :

Embarque DEVELOPPEMENT LOGICIEL

Linux, VxWorks, UC Linux, Android
pile IP, BSP, Pilotes,
GUI, l'espace utilisateur Applications,
Modules spatiaux Kernel

INTERNET DES OBJETS

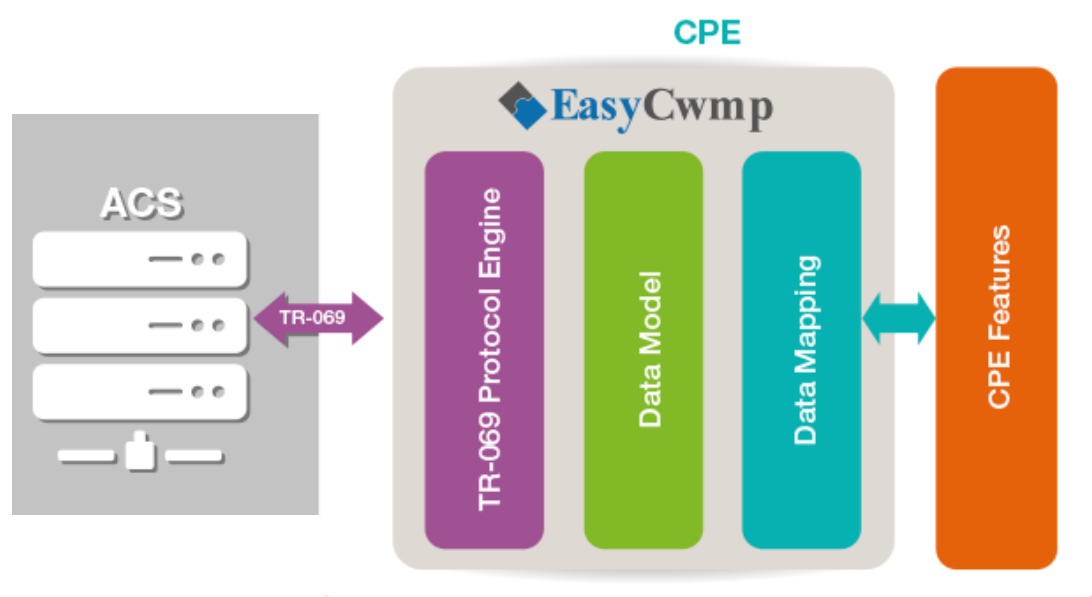
IOS, Android, Windows Applications
Peer-to-peer / application des réseaux client-serveur,
système intégré de développement pour les appareils intelligents,
applications mobiles natives

ASSURANCE QUALITÉ

Linux, Windows
Spécification d'essai, l'automatisation de test,
test mise en œuvre de lit
analyse Crash, Correction, soutien et Followu.

SOLUTION EasyCwmp UN CLIENT TR-069 OPENSOURCE

PIVA Software introduit EasyCwmp, un client open source TR-069, une plate-forme de gestion unique qui peut supporter une gamme de dispositifs de réseau de Home CPE, points d'accès, VoIP et décodeurs basés sur différents modèles de données et les spécifications....



Tâches réalisées

Introduction

Durant la période du stage, on essayera de développer EasyCwmp UN CLIENT **TR-181** OPENSOURCE.

Presentation OpenWRT - EasyCwmp

OpenWRT-CC [«Chaos Calmer»]



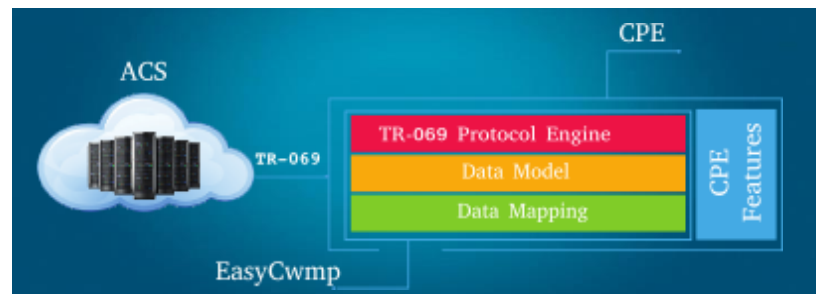
OpenWrt est une distribution hautement extensible GNU / Linux pour les appareils embarqués (généralement des routeurs sans fil). Contrairement à beaucoup d'autres distributions pour ces routeurs, OpenWrt est construit à partir du sol pour être un système d'exploitation complet, facilement modifiable pour votre routeur.

Le projet OpenWrt a commencé en Janvier 2004. Les premières versions OpenWrt étaient basées sur Linksys GPL sources pour WRT54G . OpenWrt est passé par de nombreuses révisions , le version stable de OpenWrt (nom de code « Chaos Calmer ») fonctionne sur des centaines de modèles de routeur.

EasyCwmp :

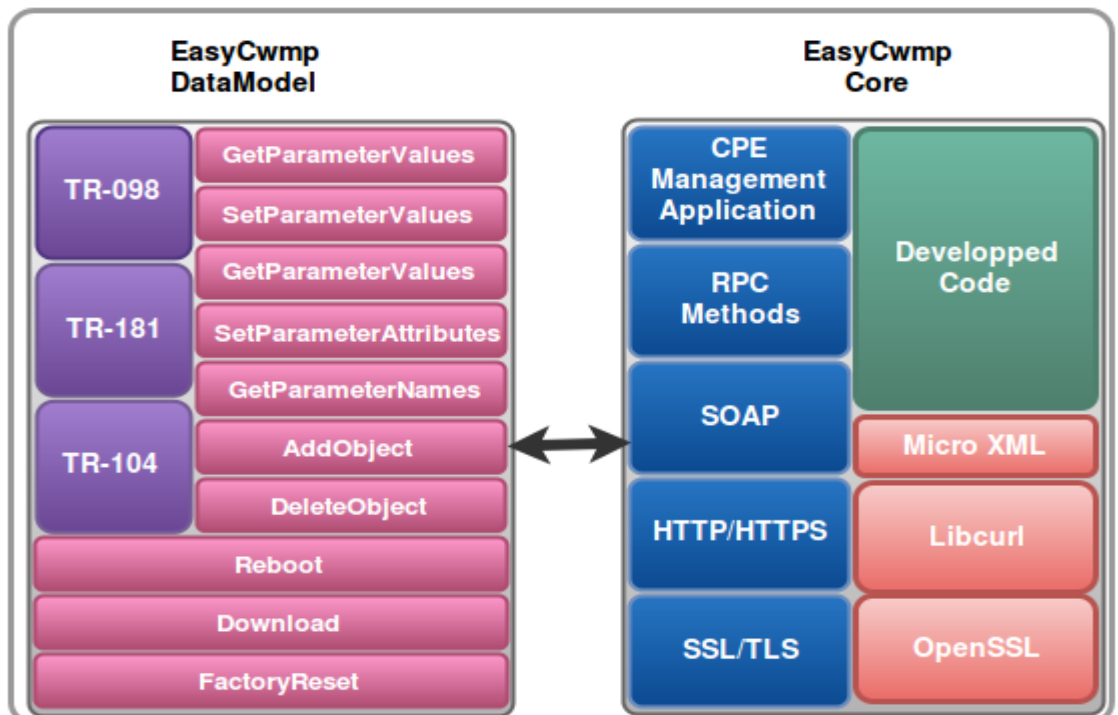


EasyCwmp est une GPLv2 implémentation open source du [TR069CWMP](#) standard. EasyCwmp est développé par [PIVA Software](#) et il est dérivé du projet freecwmp. Le but de ce projet est d'être entièrement conforme à la norme TR069CWMP.



Conformité aux normes

- **TR-069** : Protocole de gestion CPE WAN v1.1
- **TR-098** : version Internet Gateway Device 1 (modèle de données pour TR-069)
- **TR-181** : Version de l'appareil 2.
- **TR-104** : Paramètres Provisioning pour VoIP CPE la version 2
- **TR-106** : Devices Modèle de données pour TR-069-Enabled
- **TR-111** : Application TR-069 pour la gestion à distance des périphériques réseau Accueil



Développeurs

Les mainteneurs du projet EasyCwmp sont :

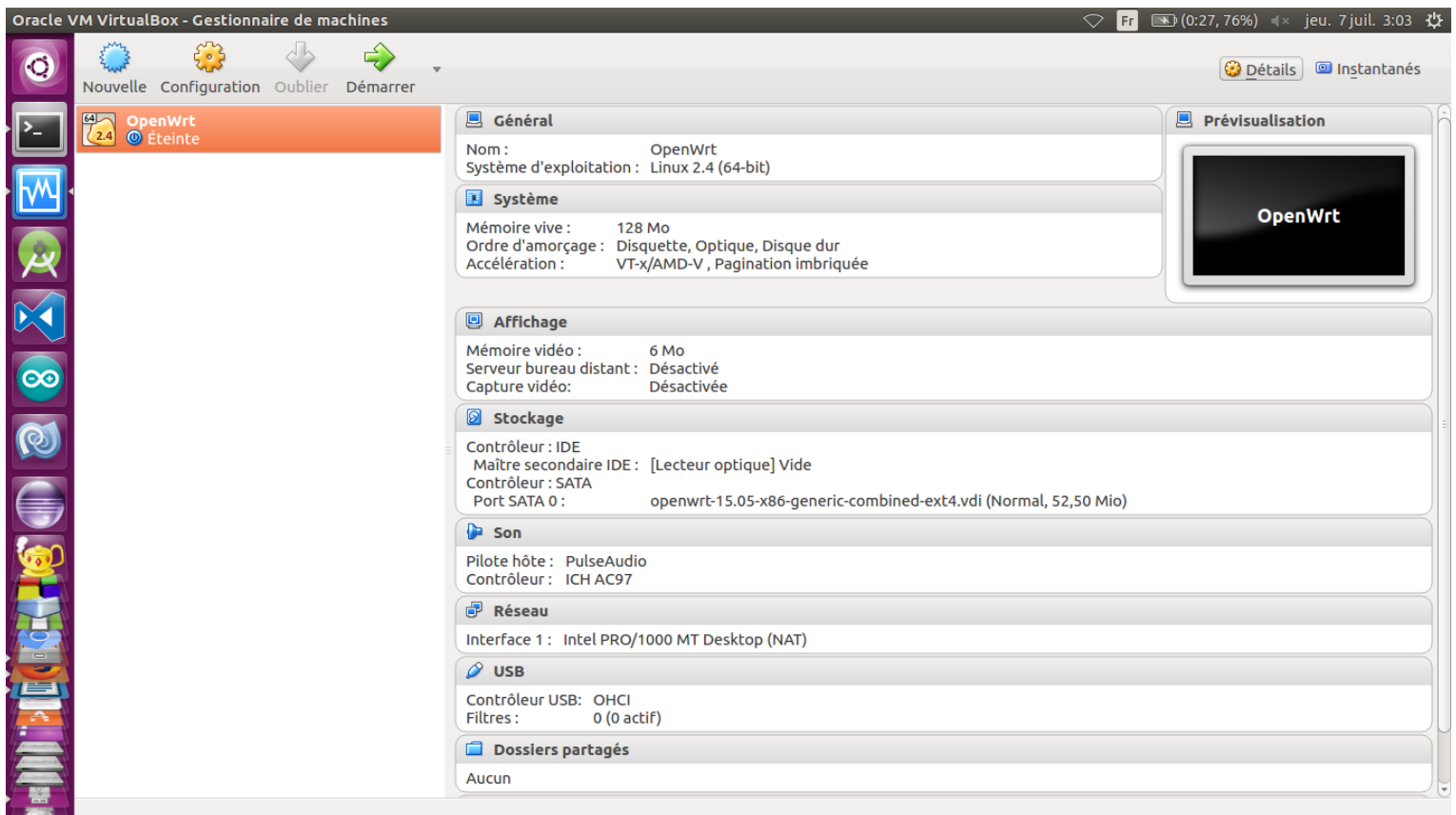
- **MOHAMED Kallel** : conduit le développement et la documentation.
- **ANIS Ellouze** : le développement et le test.

Les missions principales

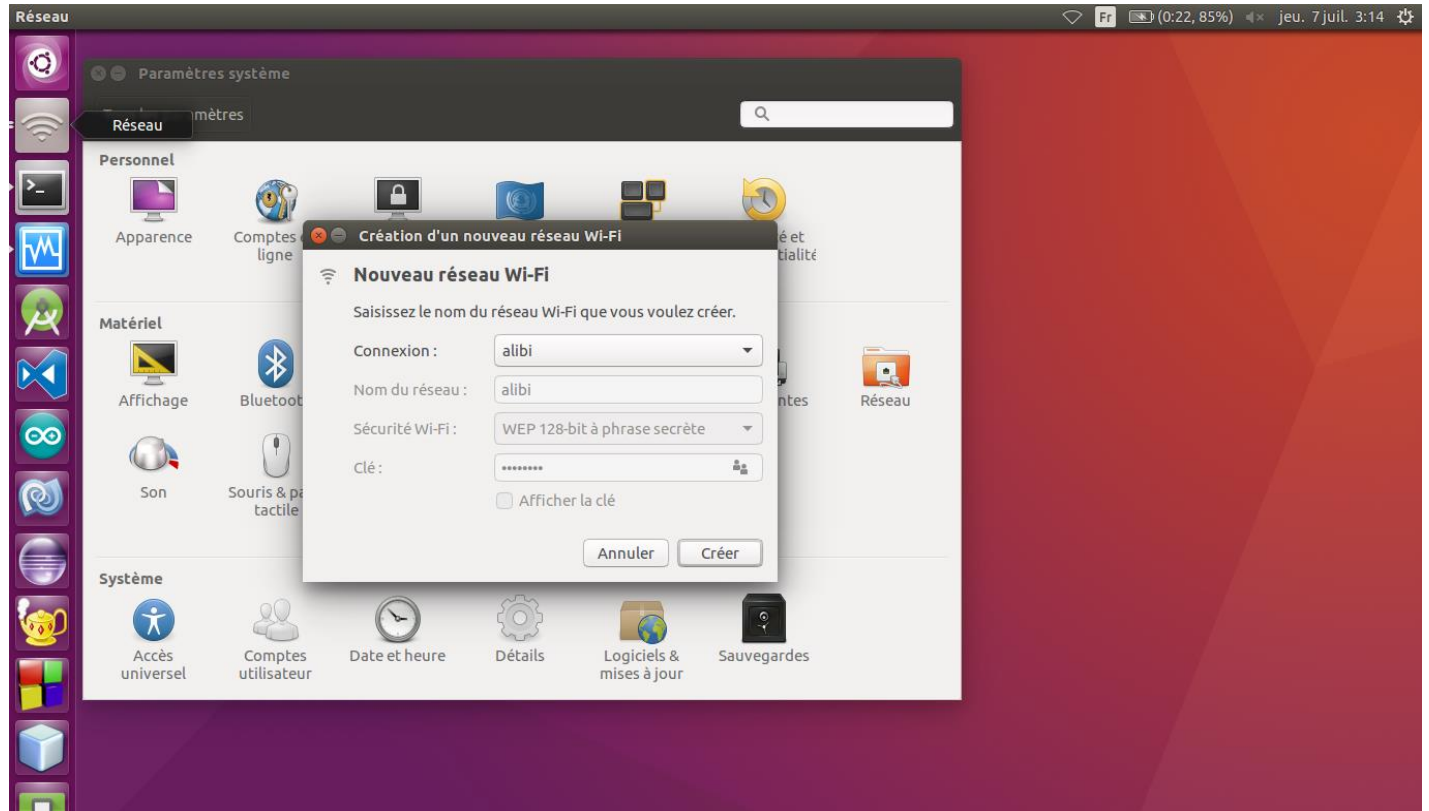
Mission 1 : Installer OpenWrt-CC

Ma première mission a été Installer OpenWrt-CC (Chaos Calmer) sur un routeur par des étape :

- Télécharger "ubuntu-16.04-desktop-amd64.iso" et avec "LinuxLive USB Creator.exe" ou bien "Universal-USB-Installer.exe" on crée LIVE-USB sur un clé USB.
- Installer "ubuntu-16.04 " en DUAL BOOT avec Windows10 et on Configure le grub.
- On installer "VM Virtual Box " à l'aide « Logithèque Ubuntu »
- Télécharger OpenWrt-CC: "https://downloads.openwrt.org/chaos_calmer/15.05.1/"
- Convertir fichier .IMG vers. VDI pour Virtual Box avec la commande « VBoxManage convertdd file.img file.vdi »
- Et en fin, installer OpenWrt.vdi.



- Création un réseau privé :

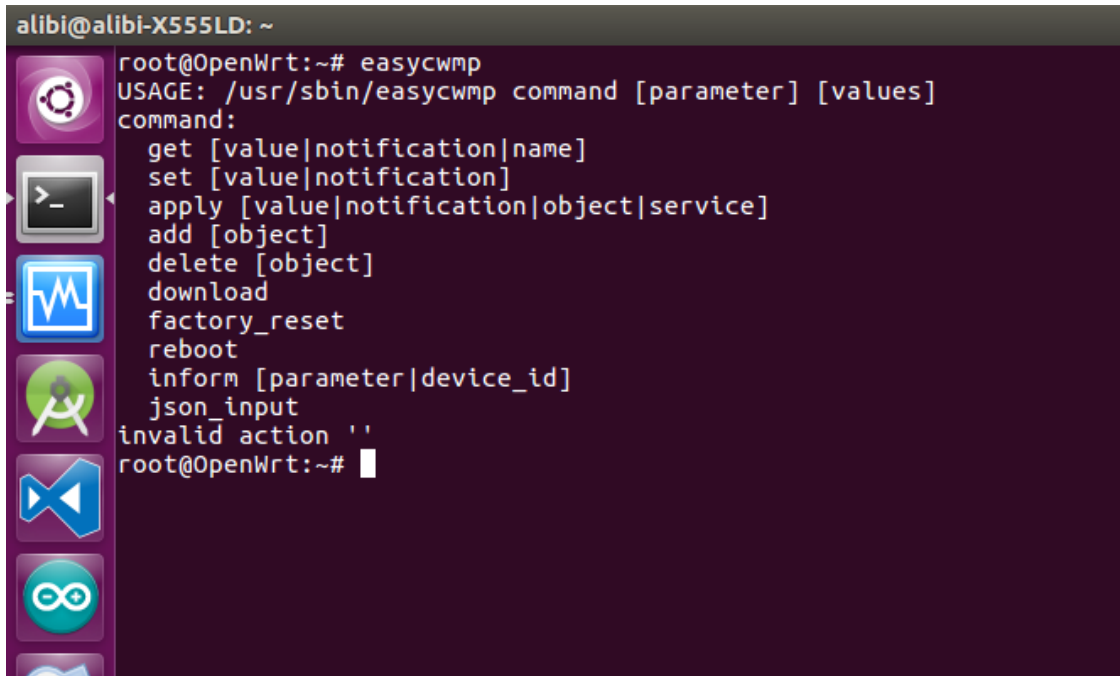


- Démarrer OpenWrt et le donne un adresse IP fixe par exemple : 10.42.0.2 avec la commande « ifconfig »

```
root@OpenWrt:/# ifconfig eth0 10.42.0.2
[ 549.924835] 8021q: adding VLAN 0 to HW filter on device eth0
[ 549.925496] IPv6: ADDRCONF(NETDEV_UP): eth0: link is not ready
root@OpenWrt:/# [ 549.930865] e1000: eth0 NIC Link is Up 1000 Mbps Full Duplex,
Flow Control: RX
[ 549.932881] IPv6: ADDRCONF(NETDEV_CHANGE): eth0: link becomes ready
root@OpenWrt:/# _
```

- Modifier le mot de passe du root par "root" avec "passwd " : (juste on taper pqsszrt)

```
root@OpenWrt:/# passwd
Changing password for root
New password:
Bad password: too short
Retype password:
Password for root changed by root
root@OpenWrt:/# _
```


Mission 4: tester easycwmp:


```
alibi@alibi-X555LD: ~
root@OpenWrt:~# easycwmp
USAGE: /usr/sbin/easycwmp command [parameter] [values]
command:
  get [value|notification|name]
  set [value|notification]
  apply [value|notification|object|service]
  add [object]
  delete [object]
  download
  factory_reset
  reboot
  inform [parameter|device_id]
  json_input
  invalid action ''
root@OpenWrt:~#
```

- Vérifier que bien installer : "root@OpenWrt:~# easycwmp get value " :

Vous devriez voir la sortie comme ceci:

```
{ "Paramètre": "InternetGatewayDevice.DeviceInfo.Manufacturer", "FAULT_CODE": "",
"valeur": "easycwmp", "type": "xsd: string" }
{ "paramètre": "InternetGatewayDevice.DeviceInfo.ManufacturerOUI", "FAULT_CODE": "",
"valeur": "FFFFFF", "type": "xsd: string" }
{ "paramètre": "InternetGatewayDevice.DeviceInfo.ProductClass", "FAULT_CODE": "",
"valeur": "easycwmp", "type": "xsd: string" }
{ "paramètre": "InternetGatewayDevice.DeviceInfo.SerialNumber", "FAULT_CODE": "",
"valeur": "FFFFFF123456", "type": "xsd: string" }
{ " paramètre ":" InternetGatewayDevice.DeviceInfo.HardwareVersion "," FAULT_CODE ":"
"," valeur ":" example_hw_version "," type ":" xsd: string " }
{ " paramètre ":" InternetGatewayDevice.DeviceInfo. softwareversion "," FAULT_CODE ":" ,
"value ":" example_sw_version "," type ":" xsd: string " }
{ " paramètre ":" InternetGatewayDevice.DeviceInfo.UpTime "," FAULT_CODE ":" valeur " ,
":" 429120 "," type ":" xsd: string " }
...
```

Développer DATA MODEL TR 181 (avec Shell)(dans tous les restes des jours) :

Le projet EasyCwmp - TR181 client CWMP est développé avec Shell comme solution libre et C comme solution commerciale.

*On trouve le cahier des charges sur le site de la société Broadband-Forum :
<https://www.broadband-forum.org/cwmp#tr-181-2-10-0.xml>*



Ma mission est alors développée data-model TR181, donc je fais développer les parties suivantes :

- 1) *Device.DeviceInfo.MemoryStatus.*
- 2) *Device.DeviceInfo.ProcessStatus.(PID)*
- 3) *Device.Ethernet.*
- 4) *Device.Ethernet.Interface. {I}.*
- 5) *Device.WiFi.Radio. {I}.*
- 6) *Device.WiFi.SSID. {I}.*
- 7) *Device.IP.Interface. {I}.*
- 8) *Device.IP.Interface. {I} .IPv4Address. {I}.*
- 9) *Device.IP.Interface. {I} .IPv6Address. {I}.*

Tous les détails se trouvent sur le site .

<https://www.broadband-forum.org/cwmp#tr-181-2-10-0.xml>

Puis, envoyer les fichiers ".sh" vers le routeur avec :

- `scp -v /home/alibi/wifi root@10.42.0.2:/usr/share/easycwmp/functions/`
- `scp -v /home/alibi/device_info root@10.42.0.2:/usr/share/easycwmp/functions/`
- `scp -v /home/alibi/ethernet root@10.42.0.2:/usr/share/easycwmp/functions`
- `scp -v /home/alibi/easycwmp root@10.42.0.2:/usr/sbin/`

- `scp -v /home/alibi/ProcessStatus_Process root@10.42.0.2:/usr/share/easycwmp/functions/`
- `scp -v /home/alibi/ip root@10.42.0.2:/usr/share/easycwmp/functions`
- `scp -v /home/alibi/network root@10.42.0.2:/etc/config/network`

N.B : à la partie « wifi », on ne peut pas tester sur un VM, alors on utilise un vrai routeur : juste on le branche avec un câble réseau, et connecter en ssh `root@192,168,0,201` si non `192.168.1.1` selon l'adresse IP par défaut du routeur, tester la get et set :

Par exemple :

✓ Get les interfaces Ethernet :

```
alibi@alibi-X555LD: ~
root@OpenWrt:~# easycwmp get Device.Ethernet.Interface.
{ "parameter": "Device.Ethernet.Interface.1.Enable", "value": "" }
{ "parameter": "Device.Ethernet.Interface.1.Status", "value": "1", "type": "xsd:string " }
{ "parameter": "Device.Ethernet.Interface.1.Name", "value": "eth0", "type": "xsd:string " }
{ "parameter": "Device.Ethernet.Interface.1.MACAddress", "value": "08:00:27:2A:9D:05", "type": "xsd:string " }
{ "parameter": "Device.Ethernet.Interface.1.MaxBitRate", "value": "[ 9.029134] e1000: eth0 NIC Link is Up 1000 Mbps Full Duplex, Flow Control: RX", "type": "xsd:int " }
{ "parameter": "Device.Ethernet.Interface.1.DuplexMode", "value": "[ 9.029134] e1000: eth0 NIC Link is Up 1000 Mbps Full Duplex, Flow Control: RX", "type": "xsd:string " }
{ "parameter": "Device.Ethernet.Interface.2.Enable", "value": "" }
{ "parameter": "Device.Ethernet.Interface.2.Status", "value": "1", "type": "xsd:string " }
{ "parameter": "Device.Ethernet.Interface.2.Name", "value": "eth1", "type": "xsd:string " }
{ "parameter": "Device.Ethernet.Interface.2.MACAddress", "value": "08:00:27:2A:9D:05", "type": "xsd:string " }
{ "parameter": "Device.Ethernet.Interface.2.MaxBitRate", "value": "[ 89.890289] e1000: eth1 NIC Link is Up 1000 Mbps Full Duplex, Flow Control: RX", "type": "xsd:int " }
{ "parameter": "Device.Ethernet.Interface.2.DuplexMode", "value": "[ 89.890289] e1000: eth1 NIC Link is Up 1000 Mbps Full Duplex, Flow Control: RX", "type": "xsd:string " }
{ "parameter": "Device.Ethernet.Interface.3.Enable", "value": "" }
{ "parameter": "Device.Ethernet.Interface.3.Status", "value": "1", "type": "xsd:string " }
{ "parameter": "Device.Ethernet.Interface.3.Name", "value": "eth2", "type": "xsd:string " }
{ "parameter": "Device.Ethernet.Interface.3.MACAddress", "value": "08:00:27:2A:9D:05", "type": "xsd:string " }
{ "parameter": "Device.Ethernet.Interface.3.MaxBitRate", "value": "[ 101.330286] e1000: eth2 NIC Link is Up 1000 Mbps Full Duplex, Flow Control: RX", "type": "xsd:int " }
{ "parameter": "Device.Ethernet.Interface.3.DuplexMode", "value": "[ 101.330286] e1000: eth2 NIC Link is Up 1000 Mbps Full Duplex, Flow Control: RX", "type": "xsd:string " }
{ "parameter": "Device.Ethernet.Interface.4.Enable", "value": "" }
{ "parameter": "Device.Ethernet.Interface.4.Status", "value": "1", "type": "xsd:string " }
{ "parameter": "Device.Ethernet.Interface.4.Name", "value": "eth3", "type": "xsd:string " }
{ "parameter": "Device.Ethernet.Interface.4.MACAddress", "value": "08:00:27:2A:9D:05", "type": "xsd:string " }
{ "parameter": "Device.Ethernet.Interface.4.MaxBitRate", "value": "[ 107.404299] e1000: eth3 NIC Link is Up 1000 Mbps Full Duplex, Flow Control: RX", "type": "xsd:int " }
{ "parameter": "Device.Ethernet.Interface.4.DuplexMode", "value": "[ 107.404299] e1000: eth3 NIC Link is Up 1000 Mbps Full Duplex, Flow Control: RX", "type": "xsd:string " }
root@OpenWrt:~#
```

✓ Get IP interface:

```

alibi@alibi-X555LD: ~
root@OpenWrt:~# easywmp get Device.IP.
{ "parameter": "Device.IP.Interface.1.Enable", "value": "", "type": "xsd:boolean" }
{ "parameter": "Device.IP.Interface.1.Name", "value": "loopback", "type": "xsd:string" }
{ "parameter": "Device.IP.Interface.1.MaxMTUSize", "value": "" }
{ "parameter": "Device.IP.Interface.1.Type", "value": "Loopback", "type": "xsd:string" }
{ "parameter": "Device.IP.Interface.1.IPv4Address.1.IPAddress", "value": "127.0.0.1", "type": "xsd:string" }
{ "parameter": "Device.IP.Interface.1.IPv4Address.1.AddressingType", "value": "static", "type": "xsd:string" }
{ "parameter": "Device.IP.Interface.2.Enable", "value": "", "type": "xsd:boolean" }
{ "parameter": "Device.IP.Interface.2.Name", "value": "lan", "type": "xsd:string" }
{ "parameter": "Device.IP.Interface.2.MaxMTUSize", "value": "" }
{ "parameter": "Device.IP.Interface.2.Type", "value": "Normal", "type": "xsd:string" }
{ "parameter": "Device.IP.Interface.2.IPv4Address.1.IPAddress", "value": "192.168.0.201", "type": "xsd:string" }
{ "parameter": "Device.IP.Interface.2.IPv4Address.1.AddressingType", "value": "static", "type": "xsd:string" }
{ "parameter": "Device.IP.Interface.3.Enable", "value": "", "type": "xsd:boolean" }
{ "parameter": "Device.IP.Interface.3.Name", "value": "wan", "type": "xsd:string" }
{ "parameter": "Device.IP.Interface.3.MaxMTUSize", "value": "" }
{ "parameter": "Device.IP.Interface.3.Type", "value": "Normal", "type": "xsd:string" }
{ "parameter": "Device.IP.Interface.3.IPv4Address.1.IPAddress", "value": "192.168.5.201", "type": "xsd:string" }
{ "parameter": "Device.IP.Interface.3.IPv4Address.1.AddressingType", "value": "static", "type": "xsd:string" }
{ "parameter": "Device.IP.Interface.4.Enable", "value": "", "type": "xsd:boolean" }
{ "parameter": "Device.IP.Interface.4.Name", "value": "wan6", "type": "xsd:string" }
{ "parameter": "Device.IP.Interface.4.MaxMTUSize", "value": "" }
{ "parameter": "Device.IP.Interface.4.Type", "value": "Normal", "type": "xsd:string" }
{ "parameter": "Device.IP.Interface.4.IPv6Address.1.IPAddress", "value": "0::0:0:0", "type": "xsd:string" }
{ "parameter": "Device.IP.Interface.4.IPv6Address.1.Origin", "value": "dhcpv6", "type": "xsd:string" }
root@OpenWrt:~#

```

✓ Set IP, Interface (loopbak) n°:1 adresse n°1: IPV4: 127.0.0.1

```

alibi@alibi-X555LD: ~
root@OpenWrt:~# easywmp get value Device.IP.Interface.1.IPv4Address.1.IPAddress
{ "parameter": "Device.IP.Interface.1.IPv4Address.1.IPAddress", "value": "127.0.0.1", "type": "xsd:string" }
root@OpenWrt:~# easywmp set Device.IP.Interface.1.IPv4Address.1.IPAddress 237.1.1.2
root@OpenWrt:~# easywmp apply
{ "status": "1" }
root@OpenWrt:~# easywmp get value Device.IP.Interface.1.IPv4Address.1.IPAddress
{ "parameter": "Device.IP.Interface.1.IPv4Address.1.IPAddress", "value": "237.1.1.2", "type": "xsd:string" }
root@OpenWrt:~#

```

Les difficultés du stage et les solutions apportées

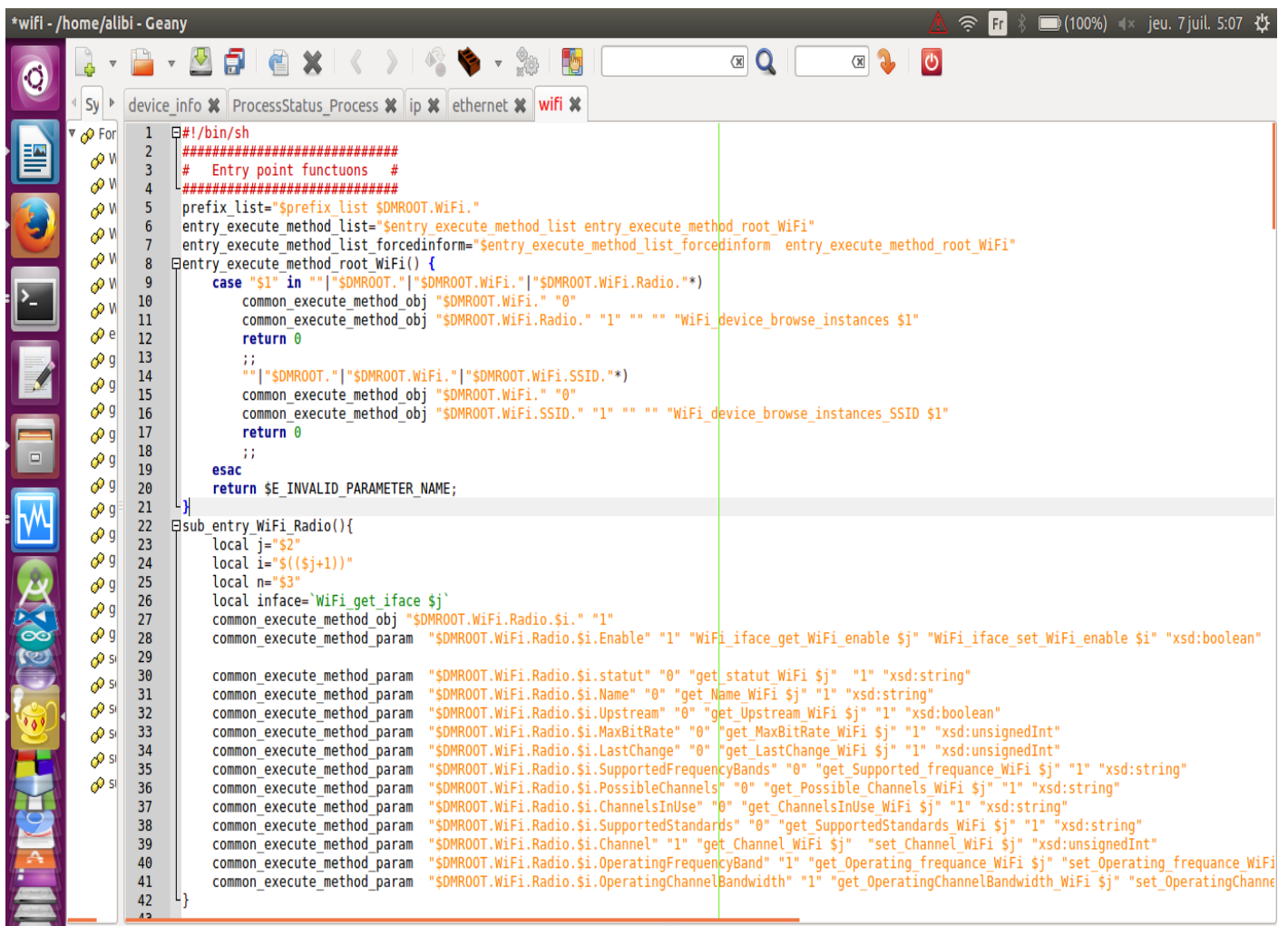
A – Les difficultés rencontrées :

J'ai rencontré des difficultés lors début de mon stage. Mais après un effort pour comprendre les quelles notions, j'ai très rapidement su gérer les missions confiées.

B – Les solutions apportées à ces difficultés :

Toutefois, j'ai trouvé des solutions aux obstacles avec l'aide de mon maître de stage et d'autre par moi-même. Afin d'illustrer mon propos, je vais vous donner un exemple.

❖ Exemple du code Shell : « wifi.sh »



```

1 #!/bin/sh
2 #####
3 # Entry point fonctions #
4 #####
5 prefix_list="$prefix_list $DMROOT.WiFi."
6 entry_execute_method_list="$entry_execute_method_list entry_execute_method_root_WiFi"
7 entry_execute_method_list_forcedinform="$entry_execute_method_list_forcedinform entry_execute_method_root_WiFi"
8 entry_execute_method_root_WiFi() {
9     case "$1" in
10         "$DMROOT."|" $DMROOT.WiFi."|" $DMROOT.WiFi.Radio.*")
11             common_execute_method_obj "$DMROOT.WiFi." "0"
12             common_execute_method_obj "$DMROOT.WiFi.Radio." "1" "" "" "WiFi_device_browse_instances $1"
13             return 0
14             ;;
15         "$DMROOT."|" $DMROOT.WiFi."|" $DMROOT.WiFi.SSID.*)
16             common_execute_method_obj "$DMROOT.WiFi." "0"
17             common_execute_method_obj "$DMROOT.WiFi.SSID." "1" "" "" "WiFi_device_browse_instances_SSID $1"
18             return 0
19             ;;
20         *)
21             return $E_INVALID_PARAMETER_NAME;
22     esac
23 }
24 sub_entry_WiFi_Radio(){
25     local j="$2"
26     local i=$((j+1))
27     local n="$3"
28     local iface="WiFi_get_iface $j"
29     common_execute_method_obj "$DMROOT.WiFi.Radio.$i." "1"
30     common_execute_method_param "$DMROOT.WiFi.Radio.$i.Enable" "1" "WiFi_iface_get_WiFi_enable $j" "WiFi_iface_set_WiFi_enable $i" "xsd:boolean"
31
32     common_execute_method_param "$DMROOT.WiFi.Radio.$i.statut" "0" "get_statut_WiFi $j" "1" "xsd:string"
33     common_execute_method_param "$DMROOT.WiFi.Radio.$i.Name" "0" "get_Name_WiFi $j" "1" "xsd:string"
34     common_execute_method_param "$DMROOT.WiFi.Radio.$i.Upstream" "0" "get_Upstream_WiFi $j" "1" "xsd:boolean"
35     common_execute_method_param "$DMROOT.WiFi.Radio.$i.MaxBitRate" "0" "get_MaxBitRate_WiFi $j" "1" "xsd:unsignedInt"
36     common_execute_method_param "$DMROOT.WiFi.Radio.$i.LastChange" "0" "get_LastChange_WiFi $j" "1" "xsd:unsignedInt"
37     common_execute_method_param "$DMROOT.WiFi.Radio.$i.SupportedFrequencyBands" "0" "get_Supported_frequance_WiFi $j" "1" "xsd:string"
38     common_execute_method_param "$DMROOT.WiFi.Radio.$i.PossibleChannels" "0" "get_Possible_Channels_WiFi $j" "1" "xsd:string"
39     common_execute_method_param "$DMROOT.WiFi.Radio.$i.ChannelsInUse" "0" "get_ChannelsInUse_WiFi $j" "1" "xsd:string"
40     common_execute_method_param "$DMROOT.WiFi.Radio.$i.SupportedStandards" "0" "get_SupportedStandards_WiFi $j" "1" "xsd:string"
41     common_execute_method_param "$DMROOT.WiFi.Radio.$i.Channel" "1" "get_Channel_WiFi $j" "set_Channel_WiFi $j" "xsd:unsignedInt"
42     common_execute_method_param "$DMROOT.WiFi.Radio.$i.OperatingFrequencyBand" "1" "get_Operating_frequance_WiFi $j" "set_Operating_frequance_WiFi $j" "xsd:unsignedInt"
43     common_execute_method_param "$DMROOT.WiFi.Radio.$i.OperatingChannelBandwidth" "1" "get_OperatingChannelBandwidth_WiFi $j" "set_OperatingChannelBandwidth_WiFi $j" "xsd:unsignedInt"
44 }

```



```

*wifi - /home/alibi - Geany
device_info ProcessStatus_Process ip ethernet wifi
#####
# Data model browse instances #
#####
WiFi_device_browse_instances(){
    local n=`WiFi_Radio_get_nbr_Radio`
    local j=0
    while test $j != $n
    do
        sub_entry_WiFi_Radio "$1" "$j" "$n"
        j=$((j+1))
    done
}

WiFi_device_browse_instances_SSID(){
    local n=`WiFi_Radio_get_nbr_Radio`
    local j=0
    while test $j != $n
    do
        sub_entry_WiFi_SSID "$1" "$j" "$n"
        j=$((j+1))
    done
}

WiFiv4Address_device_browse_instances(){
    local j=$1
    local n=`WiFiv4_Radio_get_nbr_Radio $j`
    local i=1
    while test $i != $((n+1))
    do
        sub_entry_WiFiv4_Radio "$1" "$i" "$j"
        i=$((i+1))
    done
}

WiFiv6Address_device_browse_instances(){
    local j=$1
    local n=`WiFiv6_Radio_get_nbr_Radio $j`
    local i=1
    while test $i != $((n+1))
    do
        sub_entry_WiFiv6_Radio "$1" "$i" "$j"
        i=$((i+1))
    done
}

ligne: 90 / 316 col: 14 sel: 0 INS TAB MOD mode: LF codage: UTF-8 type de fichier: Sh portée: WiFiv6Address_device_browse_instances

```

```

*wifi - /home/alibi - Geany
device_info ProcessStatus_Process ip ethernet wifi
#####
# get Supported Standards #
#####
get_SupportedStandards_WiFi(){
    $UCI_GET wireless.radio$1.hwmode | sed 's/[0-9]//g'
}

#####
# set Possible Channels #
#####
get_Possible_Channels_WiFi(){
    local p=`pwd`
    local path=`$UCI_GET wireless.radio$1.path`
    cd /sys/devices/
    cd $path
    cat local_cpulist
    cd $p
    common_delay_service_restart_in_apply_service "network" "1"
}

#####
# get Name SSID #
#####
get_Name_ssid(){
    $UCI_GET wireless.@wifi-iface$1.ssid
}

#####
# get MAC address SSID #
#####
get_MACAddress_ssid(){
    local p=`pwd`
    local path=`$UCI_GET wireless.radio$1.path`
    cd /sys/devices/
    cd $path
    cd ieee80211
    cat `find -name macaddress`
}

```

Conclusions

Enfin ; je remercie toute l'équipe de Piva Software pour les conseils et les soutiens qu'ils m'ont prodigués tout au long de ce stage.

A la fin de ce stage l'élevé ingénieur apprendre beaucoup des choses techniques, par contre dans l'école la formation est presque basées sur la théorie, au cours de cette formation technique qui nous offrir la société je n'ai pas bien pratiqué c'est que j'ai vus dans l'ENIS, mais j'ai la chance de Travail avec un groupe, de partager l'information, d'améliorer mes compétences d'avoir un esprit de travail collaboratif, esprit de la groupe et d'ingénieur.

Malgré le fait que ce stage ne s'est pas déroulé dans des bonnes conditions avec notamment un problème de transport, J'ai réussi à atteindre les objectifs demandés. Je suis content pour cette expérience et nouveaux contacts.

Ce stage me conforte dans mon choix de carrière professionnelle. Il confirme mon souhait de me spécialiser dans informatique embarquée.

Informations de contact



Nom : **ELLOUZE ANIS**

Titre : **Embedded System engineer**

Tél : **22 778 784**

anis.ellouze@pivasoftware.com

Specialize: Embedded software development: C/C++, VxWorks, Linux -C embarqué - Software Development-Telecommunications-SQL-Java-Linux-C++-Integration-XML-C-Project Management-JavaScript-Software Project Management-Microsoft SQL Server-Product Management-Business Analysis-HTML-MySQL

->Hardware - G.SHDSL :

* CPU : MPC8272 PowerQUICC II

* Chipset G.SHDSL : GLOBSPAN, INFINEON

-->ADSL : * CPU : Centillium Palladia P300 MIPS32

* Chipset ADSL : INFINEON, Amazon

-->VDSL : * CPU : MPC8323 PowerQUICC II Pro

* Chipset VDSL2 : ikanos CPE5

-->FastEthernet : * CPU : MPC8323 PowerQUICC II Pro

* Chipset FastEthernet : MARVELL, MICREL

-->DVB : * CPU : STBx25xx (powerpc)

-Automatic test : shell, TCL, TK, Expect, Perl

-GUI development: html, Ajax, javascript., PHP, css

-Network and protocols: TR069, WiFi, SNMP, ATM, XDSL, IPsec, SIP, FTP ...

Informations sur l'entreprise

PIVA Software

Adresse

Route Kaid Mhamed Km 4,5

Sfax 3062 Tunisie

Tél (+216) 74 611 029

Télécopie (+216) 74 612 729

Site web :<http://www.pivasoftware.com>



LES PARTENAIRES



Sitographie

<http://www.pivasoftware.com/>

<https://wiki.openwrt.org>

<http://www.easycwmp.org/>

<https://www.broadband-forum.org/cwmp#tr-181-2-10-0.xml>