Title: Integration of MANET/Mesh networks with qMp in the Guifi·net community

Volume: 1/1

Author: Jorge Luis Florit López

Director: Llorenç Cerdà Alabern

Department: Computer Architecture

Date: January, 2014

ABOUT THE PROJECT

Title:	Integration of MANET/Mesh networks with qMp in the Guifi·net community
Author:	Jorge Luis Florit López
Degree:	Enginyeria Tècnica en Informàtica de Sistemes
Credits:	22.5
Director:	Llorenç Cerdà Alabern
Department:	Computer Architecture

EVALUATION COMITEE *(name and signature)*

President:	Leandro Navarro Moldes
Member:	Jordi Cortadella Fortuny
Secretary:	Llorenç Cerdà Alabern

QUALIFICATION

Numeric qualification:

Descriptive qualification:

Date:

Integration of MANET/Mesh networks with qMp in the Guifi·net community

Jorge Luis Florit López

January, 2014

Acknowledgements

Thanks to:

My parents, *for being always there to give me their support, without which this project could not have been done.*.

Alexandra Fimia, *for having appeared, for the help with the documentation and for being the best*.

Llorenç Cerdà, *for the great help and for the revision of the project*.

Pau Escrich, *for the great support with the whole project specially the qMp related*.

Agustí Moll, *for the technical help and the support to connect to Guifi.net*.

Miquel Martos, *for the help with the Guifi.net website*.

All the Guifi.net community, *for being such a great initiative and project*.

Contents

Chapter 1

Introduction

1.1 Motivation and objectives

This project arises from the need of integrate and standardize the use of Mesh networks with the software from the Quick Mesh Project (qMp) in the Guifi.net community network.

There are many personal motivations to do this project. First of all, when deciding where to focus the work I was clear that I want to do something with a real utility which could have continuity and not only a theoretical thing. Also I like the computer networks themes and I wanted to continue learning about them, and of course, acquiring knowledge and gaining experience on this area.

The main objective of the project is to establish a basis for the use of Mesh networks with qMp in Guifi.net, as well as to promote the use of free software in the community.

Understanding as **Mesh network** the one where all nodes are interconnected using dynamic routing protocols (section 2.1), as **Guifi.net** the free, open and neutral a telecom network made and owned by the users (section 2.2), and as **qMp** the operating system for embedded devices to allow easy deployment of Mesh networks (section 2.3).

To achieve these objectives it is necessary to adapt the Guifi.net website to the real situation of the Mesh netwokrs made with the qMp software. Furthermore is needed to create a new user-friendly web for qMp to allow users to get easily the information about qMp they need.

In addition, tools will be developed to be able to interact between the Guifi.net site and the qMp nodes to configure them, using the 'oneclick' (unsolclic) feature from the Guifi web, which basically consists of facilitating users to configure their devices.

1.2 Organization of this document

This document is structured in 6 chapters.

Chapter 1, contains an introduction with the **motivation** of this project and the **objectives** to achive.

Chapter 2 has explained the **state of art** of the subjects involved in this project.
Firstly, there is an explanation of what is a Mesh network and straightaway is detailed the actual state of the Guifi.net community network, especially regarding to the Mesh networks.
Also is explained the qMp situation, focussing on the operation mode which fits in a community network, and his relation and implication in Guifi.net.

Chapters 3 and 4 are more technical and they refer one in all the **Guifi.net related work**, both the decisions and the development of website functionalities; and the other in all the **qMp work**, both firmware development and new user-friendly website.

Chapter 5 has a detailed a **budget estimation** of what would have been the expenses if this would be a commercial project.

Chapter 6 describes the **conclusions and results** of this project, as well as the continuity and possible improvements to do.

At the end, there are some **Appendices** that contains development related information and source code.

After that, there is a list of **Acronyms** used and then the **Bibliography** consulted.

Chapter 2

State of art

In this chapter is introduced the main concepts related to this project.

Firstly is introduced what is a MANET/Mesh network, that is the base topic around which the project is developed.

After that, is detailed the actual situation (state of art) and the connections between the two subjects involved in this project. They are, the Guifi.net community network[1] and the free software project qMp[2].

2.1 What is a MANET/Mesh network

Usually when we talk about a MANET/Mesh network [16] we do not mean strictly a Mobile Ad-hoc Network (MANET) [8] nor only a mesh network topology [20]. Due to the difficulty to find a proper name, the concept of Mesh network covers a wide range of aspects related like the topology, interconnection and protocols.

The origin of the Mesh networks are the Wireless ad-hoc networks [24], but nowadays some protocols extends them to wired ones.

The main particularity in this networks is that routing protocols used are dynamic [21], but not as the traditional dynamic routing protocols. It means that they can dynamically decide the route of the packets and also they can dynamically link to the neighbours they find in the coverage area, not restricted to static point to point links.

There is no need to configure both sides with the other static routing information, like AS, IP, keys, etcetera; only is needed to set an identifier of the network (usually BSSID [17]).

As the name indicates, the topology of these networks is a mesh. This means that, unlike the typical community network topology, all the nodes can connect to one or more nodes around them, resulting in a meshed organization of links (see figure 2.1).

In this way, all the nodes (participants) are routers and extend the network as they link to each other.

[1]Guifi.net Community Network: http://guifi.net

[2]Quick Mesh Project: http://qmp.cat

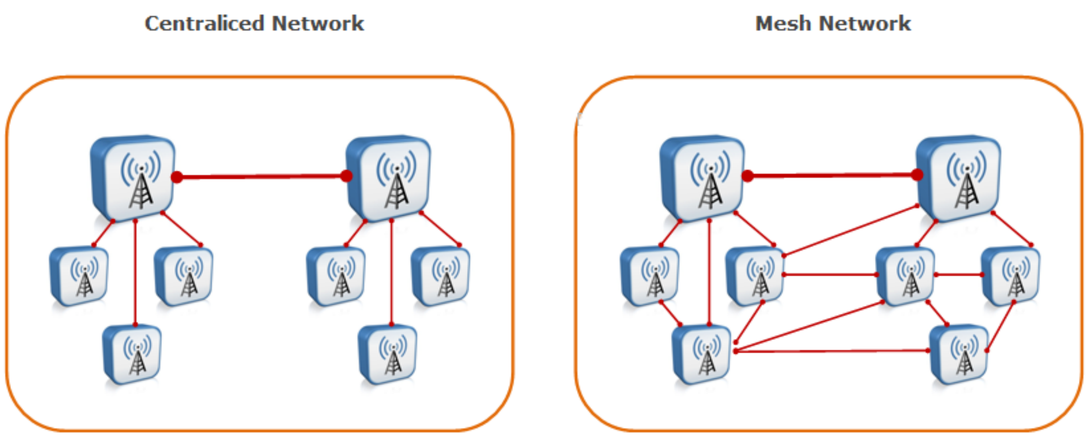


Figure 2.1: Graph comparison of centralized and Mesh networks.

From now on this kind of networks is referred to as "Mesh networks" in this document.

2.2 Guifi.net

Guifi.net is a telecommunications hybrid network and above all is a community network made by persons.

guifi·net

The basics are the "Commons for the Free, Open and Neutral Network"[1]. This is a License for interconnection and can be summarized in:

- Free, because there are no restrictions and the network is made by users and belongs to them. Also you are free to use the network for any purpose as long as you don't harm it and you don't restrict other users' freedom.
- Open, because everyone can join the network, can know how it's build and can contribute to maintain and to extend it; all without depending on an only enterprise.
- Neutral, because there are no limitations in the flowing data through the net. There are no restrictions and no priorities depending on the kind of transmitted data.

2.2.1 Brief history

Guifi.net was born in the spring of the year 2004. In April of that year, people interested in that topic met them to share ideas and to plan the first tests. Next month, in May, due to the success, the first stable and permanent links were established between some towns at northeast of the province of Barcelona in Catalonia.

Guifi.net has received several awards in the years 2004, 2006 and 2007 for being driving of the public open network, for being a pioneering project in new ways of participation and for offering resources to the community and connecting the rural world. Also has the recognition for the social initiatives of using the new information technologies, communication and knowledge.

In July of 2008, was established the Guifi.net Private Foundation for the Free, Open and Neutral Network[3]. It is a platform for the research, development and innovation; and it is also a forum for institutions, organizations and companies to collaborate in a global project for the development of network infrastructures and services. In November of that year, the European Union announces the selection of Guifi.net as a member of the European Network of Living Labs[4].

In March of 2009, the Foundation was registered as a telecommunication operator in the Spanish Telecom Market Commission (now the CNMC) of Spain[5]. And some months later, in November, Guifi.net joined the CATNIX[6]; the Internet Exchange Point of Catalonia.

Now the Foundation is involved in several European research projects[7][8], and the network has grown to more than 20,000 working nodes.

2.2.2 Mesh networks in the community

Traditionally, in Guifi.net, wireless links are set in infrastructure mode instead of ad-hoc. For that reason, the backbone network is made by point to point links, and only there are point to multipoint links in supernodes with AP to connect to client nodes.

In the network layer (L3 in the OSI model [10]), the routing protocols used are usually BGP and OSPF.

However, the Mesh networks are connected in ad-hoc mode if they are wireless. And no matter if they are wired or not, the routing protocols used are the dynamic ones commented above.

The protocols used are commonly BATMAN, BMX and OLSR [21].

[3]Guifi.net Foundation (CAT): http://fundacio.guifi.net/es/01_inf/inf.html

[4]Living Labs member: http://ec.europa.eu/information_society/newsroom/cf/itemlongdetail.cfm?item_id=4537

[5]Registered as a telecom operator (CAT): http://guifi.net/node/25751

[6]Guifi.net joins the IXP: http://www.catnix.net/en/noticia/guifinet_es_connecta_al_catnix/26/

[7]CONFINE project (CAT): http://blogs.guifi.net/fundacio/2011/10/20/projecte-europeu-de-recerca-cofine/

[8]CLOMMUNITY project: http://blogs.guifi.net/fundacio/page/2/

Firmware used

The Mesh networks in Guifi.net began in the GSF community[9]) from the Gràcia district in Barcelona, and after they expanded to other areas, like Guifi Sants, Vallcarca or Poble Nou Sense Fils.

Because of there was no commercial software for routers that allow to use this kind of protocols, the GSF people started to develop a new free software for routers, adding some functionalities and the dynamic protocols to the well known OpenWRT Linux operating system[10]. They called this software '*GSFfirm*'.

You can read how it works in this external document [13].

Web ad-hoc functionalities

To allow users of Mesh networks and make them easier to register their nodes in the Guifi.net website there were developed some functionalities in the Drupal module.

They consist basically in:

- Separate the zones in ad-hoc and infrastructure mode, depending if a Mesh network is deployed on that zone or not.
- Add new types of protocols to be able to select.
- Add new type of IP addresses for each protocol used.
- Restrict the nodes in the zone to disallow them to add more than one wireless device.
- Automatically create only one radio to the device with a new type '*ad-hoc*'.
- Assign automatically to the '*ad-hoc*' radio a new IP address, only with a /32 network mask instead of the /27 usually assigned to the supernodes.

2.3 qMp

The Quick Mesh Project (qMp) is a project that develops a firmware (embedded software) for network devices based on the OpenWRT Linux operating system.

This software provides an easy way to set up Mesh networks no matter they are wired or wireless or a mix of both. It does that with a complete autoconfiguration system at the first boot. And it is also a fast and reliable way to extend an Internet uplink to end-users.

[9]Gràcia Sense Fils: http://graciasensefils.net
[10]OpenWRT: https://openwrt.org

As a layer 3 routing protocol, qMp uses BMX6, a new protocol based in BATMAN experimental but rewritten from zero with native support to IPv6. It uses IPv6 as the main IP protocol and uses IPv4 tunnelled over IPv6 for end-user connections.

You can read more about BMX6 in this document from the protocol's repository [23].

qMp has two main purposes (see figure 2.2):

- **Roaming, for quick deployments**; there's no need to think about network topology, it is only needed to spread the nodes and to connect one to Internet.
- **Community, for the wireless community networks**; it allows to become part of an existing community or to start a new one easily. It is only needed to set some community specific parameters (IP address, Name, BSSID, etc.) to have your device ready.

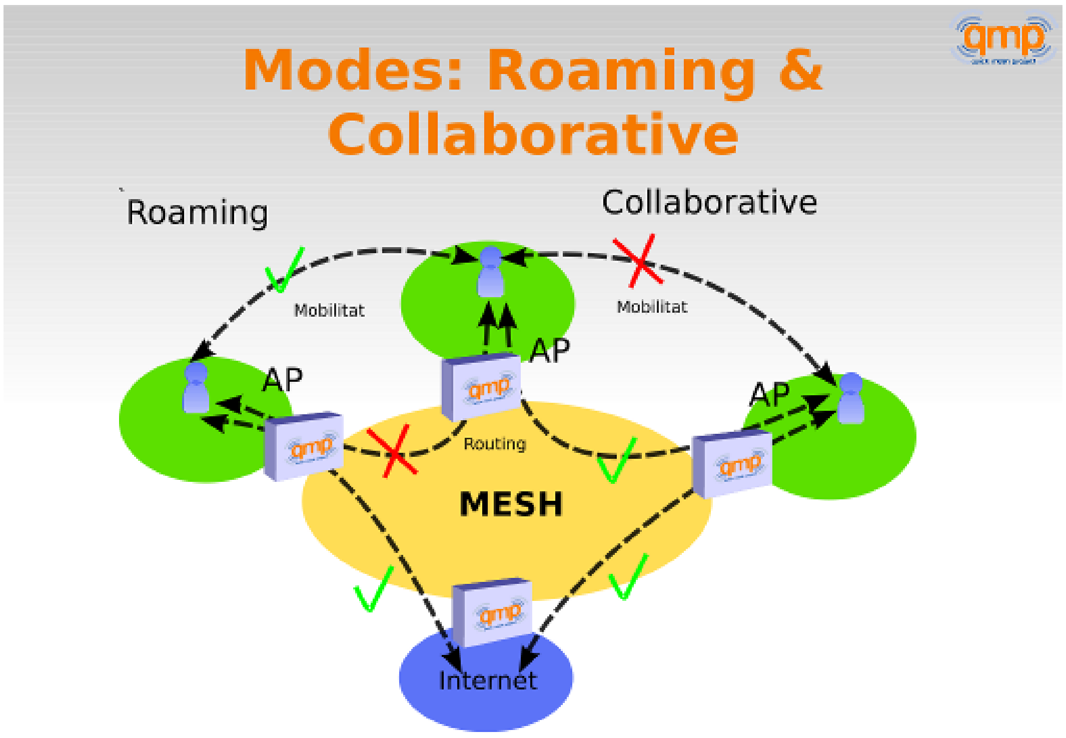


Figure 2.2: qMp modes.

A more detailed information is available in this external document[22], a thesis degree about the qMp project.

2.3.1 History and origins

The Quick Mesh Project (qMp) was drafted by Guifi.net active members during 2010. The idea of this project was based on another project started in 2003 by some network hackers from GSF in the Gràcia district in Barcelona.

It started in 2009 when an organization asked Guifi.net members to cover a big public demonstration which took place in Barcelona. In this scenario the most efficient deployment was a flexible network, so the GSF people adapted their software to cover it.

After that, more people, companies, associations and also political parties were asking for the same kind of deployment to cover their events. However GSF software was not designed to be used in this scenario but to be used in a wireless community, and then some members of GSF and Guifi.net decided to start a new project thinking in this new needs.

2.3.2 qMp in Guifi.net

Before this project, in Guifi.net there was only one Mesh network with qMp software working. It was deployed in the UPC under the research project CONFINE.

The nodes of the UPC network cloud were registered in the Guifi.net website in a new subzone with the zone mode selected to 'ad-hoc'. Then, due to the restrictions commented in 'section 2.2.2', the nodes have to be changed manually their auto-assigned IP addresses to a new one with a /28 network mask range.

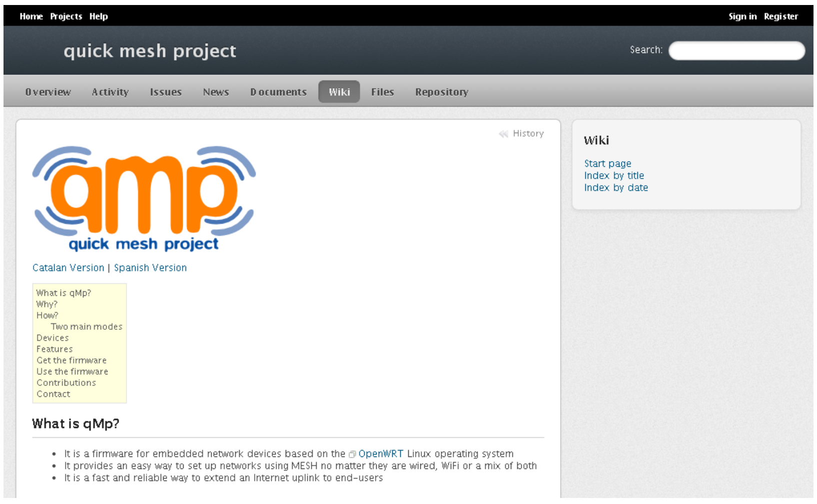
The supernode which acts as frontier between the Mesh and the rest of Guifi network had to be declared in the parent zone, also because the restrictions of zones with ad-hoc mode.

qMp Website

This is not related only to Guifi.net users but every user of any community.

The qMp website (figure 2.3) had been built with a project management web application called Redmine[11]. It has many tools to coordinate and to manage software development projects such as a wiki, an issue track, the integration with several SCM, a calendar, file storage, etcetera.

There is where users, contributors and developers can find information related to the Quick Mesh Project.



What is qMp?

- It is a firmware for embedded network devices based on the OpenWRT Linux operating system
- It provides an easy way to set up networks using MESH no matter they are wired, WiFi or a mix of both
- It is a fast and reliable way to extend an Internet uplink to end-users

Figure 2.3: qMp Redmine website.

[11]Redmine project management web-app: htttp://www.redmine.org

Chapter 3

Guifi.net development

Chapter 3 deals with Guifi.net subjects.

In the section 3.1 are explained the topics to regulate the Mesh networks in the community.

Then, in the second one it is the Drupal module related. Here it is explained the work done in this project with the Guifi Drupal module to adapt it to the specifications commented in the former section.

Furthermore, in the third section is detailed the creation of a new firmware for the 'oneclick' (unsolclic) feature of the Drupal module.

3.1 Regulation of Mesh networks

The reason to regulate the Mesh networks in Guifi.net is to have an organized and structured community; it is a must to have a consensus and a standardization of the use of it.

That is why this section describes the important points regarding the Mesh networks decided in this project. They are the names that nodes should have, the name of the SSID of nodes, and a very important matter to the correct operation of the network; the IP addressing.

It is important to have a good consensus with these points to know what changes have to be performed and how to do them. In the next section 3.2 they are explained.

3.1.1 Nodenames

A consensus of the nodenames was taken by the Guifi.net community to homogenize them [4] and to be more clear.

This scheme is well known to work right and is widely accepted in the community; for that reason is decided to use the same scheme in Mesh networks to be consistent with the existing nodes in Guifi.

Nomenclature to use is:
`[City][Street][Number]`

In case of no street name or to have a name with more local relevance it is accepted to use:
`[City][Localname]`

- [City] is the name of the city or town where the node is.
- [Street] is the name of the street where the node is.
- [Number] is the number of the house or building where the node is.
- [Localname] is a relevant name or information of the node's place.

3.1.2 SSID names

SSID identifies a wireless device in the network, so it is important too to have a consensus with this identifier because it will be the first shown in the wifi scans when new users try to connect. And also SSID differentiates the Guifi nodes from other wifi devices around them.

It is recommended to set the SSID with a location if the node have more than one radio device and/or there are devices in different places.

The SSID should be like:

```
guifi.net/[Nodename]/[Location]
```

- [Nodename] is the name of the node described in 3.1.1.
- [Location] is the physical place of the device.

3.1.3 IP addressing

Nodes

Because of the nodes in Mesh networks really act as traditional supernodes [3], it is to connect to other nodes, they should have assigned an IP address range to give single IPs to clients.

For that reason in the Guifi website the Mesh nodes, unlike the old ad-hoc zone nodes, should have assigned by default a **/27 range**; that means a range of 32 addresses (which allows 29 clients).

Zones

Is understood as zones the areas where nodes are grouped geographically, such as neighbourhoods, towns, cities, etcetera.

Unlike the recommended in Guifi.net for zones, to reserve at least a /24 IP address range, with Mesh nodes it is recommended to use at least a **/21 range** per zone because of each node spends a /27 range, not only a single IP address (/32).

This result in: 64 possible Mesh nodes (with a /27) in zones with /21, instead of 8 nodes (with /27) in zones with /24.

It is also recommended to use a /24 for the qMp roaming mode (explained in chapter 2.3) to allow new unconfigured nodes to be connected to zones and to have their IP routed outside the Mesh. If not, users with a random auto-generated IP of the roaming mode, won't be able to access Guifi.net nodes outside the Mesh.

The decision taken is to reserve the last /24 of the /21 IP range of the zone.

Let's see an example to understand it better:

```
 Zone: 10.1.0.1/21 -> For community: 10.1.0.0 - 10.1.6.255
                   -> For roaming:   10.1.7.0 - 10.1.7.255  (i.e. 10.1.7.0/24)
```

Also it is decided to reserve a wide range of the Guifi one to be used only in the Mesh networks.

This is because of if some Mesh networks grow and see other ones near they will connect together. And to have an IP range only for Mesh would facilitate the IP aggregation in case of the interconnected networks had consecutive IP ranges. So, the frontier nodes could send the prefix of the aggregated range instead of one prefix per range.

Let's see and example with four networks interconnected with consecutive ranges:

```
Network A
   10.1.0.0/21  \
Network B        10.1.0.0/20  \
   10.1.8.0/21  /              \
Network C                       10.1.0.0/19
   10.1.16.0/21 \              /
Network D        10.1.16.0/20 /
   10.1.24.0/21 /
```

In this case the frontier nodes could send the /19 prefix instead of the four /21 prefixes.

Actually in Guifi.net is assigned the range 10.1.0.0/16[1]; that is 32 Mesh zones (with a /21). If needed more, consecutive /16 ranges should be reserved.

3.1.4 Public documentation

The last point, but not the less important, is to share with the community the standardization and all the knowledge of Mesh networks.

For that purpose, after doing this project the changes made has been also documented in the Guifi's wiki to make them available to the community. These articles talk about how to create a Mesh node in the website and how to start a new Mesh zone.

After that, some other Guifi.net members have contributed on it and they have added more information, and also a new 'Mesh' category[2] has been created to group all the topics.

[1]Guifi world IP ranges: http://guifi.net/en/node/3671/view/ipv4

[2]http://ca.wiki.guifi.net/wiki/Mesh

3.2 Drupal module

Guifi.net website is based in the Drupal CMS with the addition of a Guifi specific module for the documentation of the Network.

Drupal is an application written in PHP that allows to manage web contents, to publish and to edit them. All can be done in a centralized interface to make it easy to operate.

This management system uses one or many databases where the contents will be stored, and it is made to isolate the administration of the design from the content.

In addition, the Drupal CMS allows the possibility to create modules[3] which are independent parts that add new features to the system.

Guifi.net community had the need to manage and to store the information about the network. Due to the lack of tools to do it, some members decided to develop a module to add such features to a content management system (Drupal in this case).

The Guifi Drupal module has many features:

- Nodes pages with network related info (Devices, IPs, Firmwares, Geolocation, etc.)
- Zones with lists of nodes to organize them geographically.
- 'Oneclick' (unsolclic) to configure devices easily.
- Maps with the nodes and links painted.
- Management of the IP addresses.
- Management of available devices, firmware and manufacturers.
- And some more.

And also there is an API[4] to develop external apps to interact with the Guifi.net site.

To read more about the Guifi module, read this wiki article [2], and to look or to download the source code go to the Gitorious repository[5].

In addition, go to this external document [6] to read how to set the development environment.

Due to the needs of the specifications for the regulation of the Mesh networks, commented in the previous section 3.1, below are explained the modifications made to the Drupal module to make them possible. Also some of the specifications and modifications have required editing some contents of the database.

[3]Drupal module's guide: https://drupal.org/developing/modules

[4]Guifi.net API: http://guifi.net/sites/all/modules/guifi/contrib/api_doc/index.php

[5]Guifi.net Official repository of the Drupal module: http://gitorious.org/guifi/drupal-guifi/

3.2.1 New ‘qMp’ firmware and supported devices

The Drupal module, as commented above, has a feature to add new firmwares, manufacturers and device models to the database to made them available to be selected by nodes. Also, in each device model it is possible to add the supported firmwares.

In the database there is a manufacturers table which contains them and is connected to another one which contains the device models. This models’ table is related to a firmwares table which has stored the supported models. See figure 3.1 to see the database scheme of the tables (only main ones) implicated in this feature.

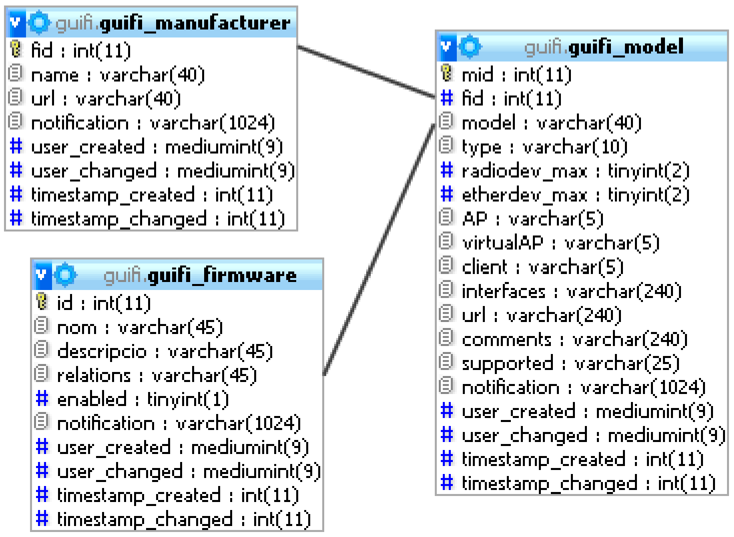

guifi.guifi_manufacturer
fid : int(11)
name : varchar(40)
url : varchar(40)
notification : varchar(1024)
user_created : mediumint(9)
user_changed : mediumint(9)
timestamp_created : int(11)
timestamp_changed : int(11)
guifi.guifi_firmware
id : int(11)
nom : varchar(45)
descripcio : varchar(45)
relations : varchar(45)
enabled : tinyint(1)
notification : varchar(1024)
user_created : mediumint(9)
user_changed : mediumint(9)
timestamp_created : int(11)
timestamp_changed : int(11)
guifi.guifi_model
mid : int(11)
fid : int(11)
model : varchar(40)
type : varchar(10)
radiodev_max : tinyint(2)
etherdev_max : tinyint(2)
AP : varchar(5)
virtualAP : varchar(5)
client : varchar(5)
interfaces : varchar(240)
url : varchar(240)
comments : varchar(240)
supported : varchar(25)
notification : varchar(1024)
user_created : mediumint(9)
user_changed : mediumint(9)
timestamp_created : int(11)
timestamp_changed : int(11)

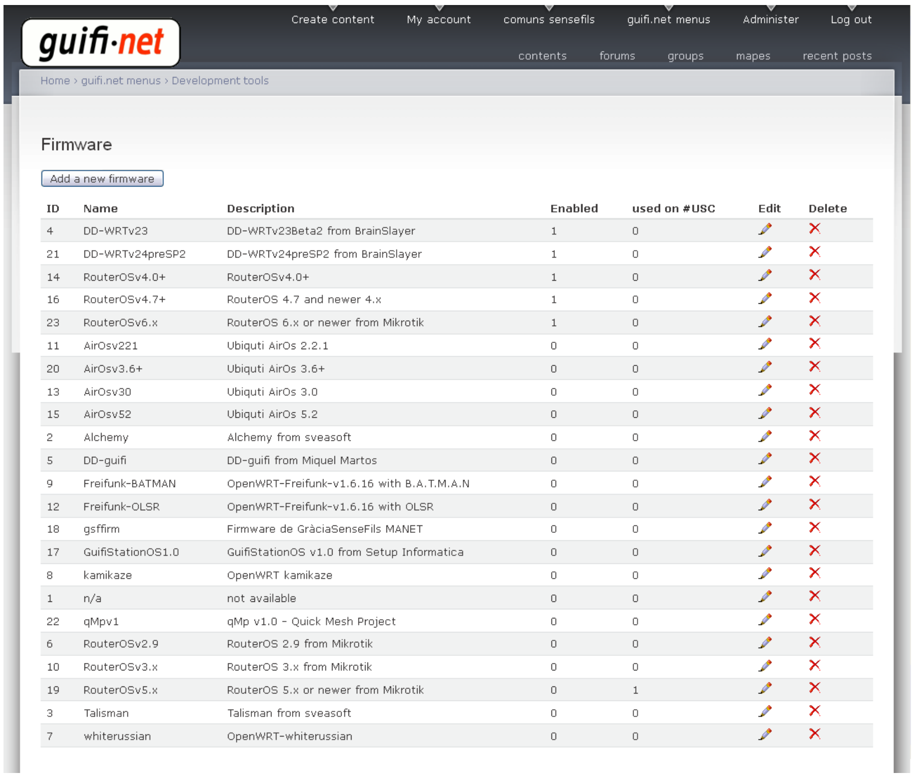
Figure 3.1: Guifi firmwares and devices database scheme.

Then, to allow users to configure their nodes for Mesh, firstly, **qMp** is added as **new firmware** available (see figure 3.2).

After that, the supported devices by qMp[6] are added to the available device models list (see figure 3.3).

Some models are already created and only it is need to add qMp as supported firmware (see figure 3.4) but if not, they have to be created. And the same occurs if the manufacturer is not in the list to (see figure 3.5).

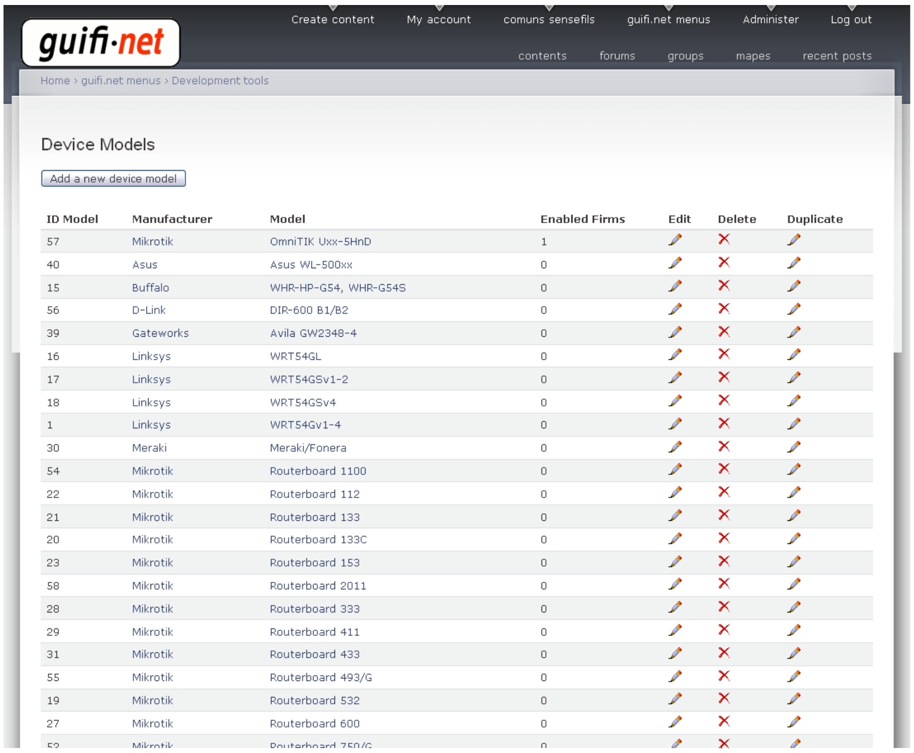
[6]qMp supported devices: http://qmp.cat/Supported_devices



Firmware

ID	Name	Description	Enabled	used on #USC	Edit	Delete
4	DD-WRTv23	DD-WRTv23Beta2 from BrainSlayer	1	0	✎	✗
21	DD-WRTv24preSP2	DD-WRTv24preSP2 from BrainSlayer	1	0	✎	✗
14	RouterOSv4.0+	RouterOSv4.0+	1	0	✎	✗
16	RouterOSv4.7+	RouterOS 4.7 and newer 4.x	1	0	✎	✗
23	RouterOSv6.x	RouterOS 6.x or newer from Mikrotik	1	0	✎	✗
11	AirOsv221	Ubiquti AirOs 2.2.1	0	0	✎	✗
20	AirOsv3.6+	Ubiquti AirOs 3.6+	0	0	✎	✗
13	AirOsv30	Ubiquti AirOs 3.0	0	0	✎	✗
15	AirOsv52	Ubiquti AirOs 5.2	0	0	✎	✗
2	Alchemy	Alchemy from sveasoft	0	0	✎	✗
5	DD-guifi	DD-guifi from Miquel Martos	0	0	✎	✗
9	Freifunk-BATMAN	OpenWRT-Freifunk-v1.6.16 with B.A.T.M.A.N	0	0	✎	✗
12	Freifunk-OLSR	OpenWRT-Freifunk-v1.6.16 with OLSR	0	0	✎	✗
18	gsffirm	Firmware de GràciaSenseFils MANET	0	0	✎	✗
17	GuifiStationOS1.0	GuifiStationOS v1.0 from Setup Informatica	0	0	✎	✗
8	kamikaze	OpenWRT kamikaze	0	0	✎	✗
1	n/a	not available	0	0	✎	✗
22	qMpv1	qMp v1.0 - Quick Mesh Project	0	0	✎	✗
6	RouterOSv2.9	RouterOS 2.9 from Mikrotik	0	0	✎	✗
10	RouterOSv3.x	RouterOS 3.x from Mikrotik	0	0	✎	✗
19	RouterOSv5.x	RouterOS 5.x or newer from Mikrotik	0	1	✎	✗
3	Talisman	Talisman from sveasoft	0	0	✎	✗
7	whiterussian	OpenWRT-whiterussian	0	0	✎	✗

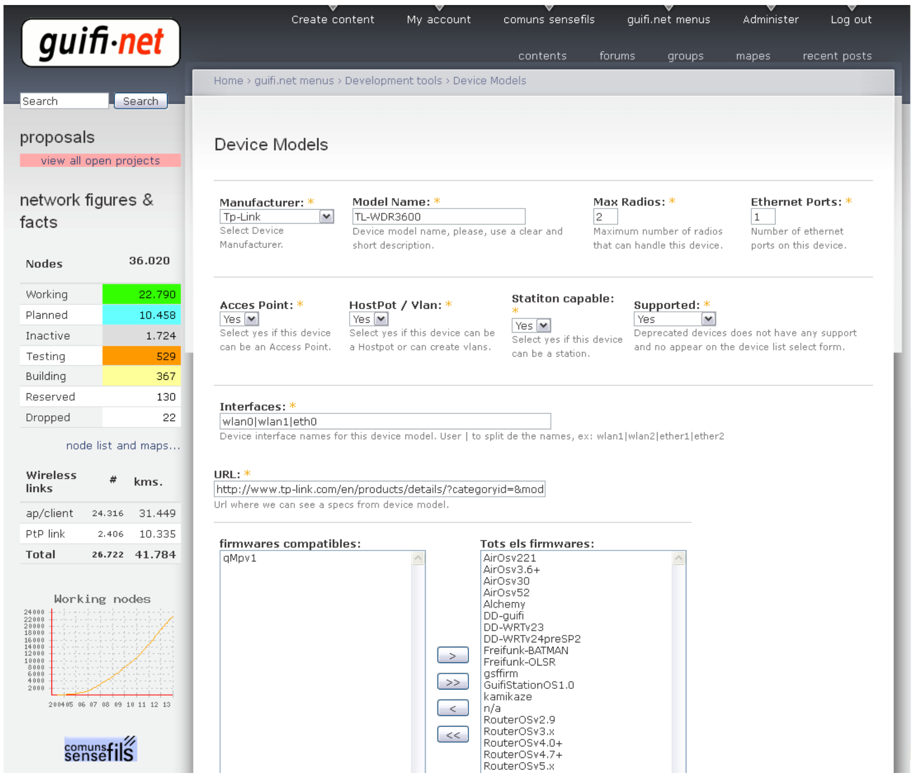
Figure 3.2: Available firmwares.



Device Models

ID Model	Manufacturer	Model	Enabled Firms	Edit	Delete	Duplicate
57	Mikrotik	OmniTIK Uxx-5HnD	1	✎	✗	✎
40	Asus	Asus WL-500xx	0	✎	✗	✎
15	Buffalo	WHR-HP-G54, WHR-G54S	0	✎	✗	✎
56	D-Link	DIR-600 B1/B2	0	✎	✗	✎
39	Gateworks	Avila GW2348-4	0	✎	✗	✎
16	Linksys	WRT54GL	0	✎	✗	✎
17	Linksys	WRT54GSv1-2	0	✎	✗	✎
18	Linksys	WRT54GSv4	0	✎	✗	✎
1	Linksys	WRT54Gv1-4	0	✎	✗	✎
30	Meraki	Meraki/Fonera	0	✎	✗	✎
54	Mikrotik	Routerboard 1100	0	✎	✗	✎
22	Mikrotik	Routerboard 112	0	✎	✗	✎
21	Mikrotik	Routerboard 133	0	✎	✗	✎
20	Mikrotik	Routerboard 133C	0	✎	✗	✎
23	Mikrotik	Routerboard 153	0	✎	✗	✎
58	Mikrotik	Routerboard 2011	0	✎	✗	✎
28	Mikrotik	Routerboard 333	0	✎	✗	✎
29	Mikrotik	Routerboard 411	0	✎	✗	✎
31	Mikrotik	Routerboard 433	0	✎	✗	✎
55	Mikrotik	Routerboard 493/G	0	✎	✗	✎
19	Mikrotik	Routerboard 532	0	✎	✗	✎
27	Mikrotik	Routerboard 600	0	✎	✗	✎
52	Mikrotik	Routerboard 750/G	0	✎	✗	✎

Figure 3.3: Available device models.



Home › guifi.net menus › Development tools › Device Models

Device Models

Manufacturer: *	Tp-Link
	Select Device Manufacturer.
Model Name: *	TL-WDR3600
	Device model name, please, use a clear and short description.
Max Radios: *	2
	Maximum number of radios that can handle this device.
Ethernet Ports: *	1
	Number of ethernet ports on this device.
Acces Point: *	Yes
	Select yes if this device can be an Access Point.
HostPot / Vlan: *	Yes
	Select yes if this device can be a Hostpot or can create vlans.
Statiton capable: *	Yes
	Select yes if this device can be a station.
Supported: *	Yes
	Deprecated devices does not have any support and no appear on the device list select form.
Interfaces: *	wlan0|wlan1|eth0
	Device interface names for this device model. User | to split de the names, ex: wlan1|wlan2|ether1|ether2
URL: *	http://www.tp-link.com/en/products/details/?categoryid=&mod
	Url where we can see a specs from device model.

firmwares compatibles:
qMpv1

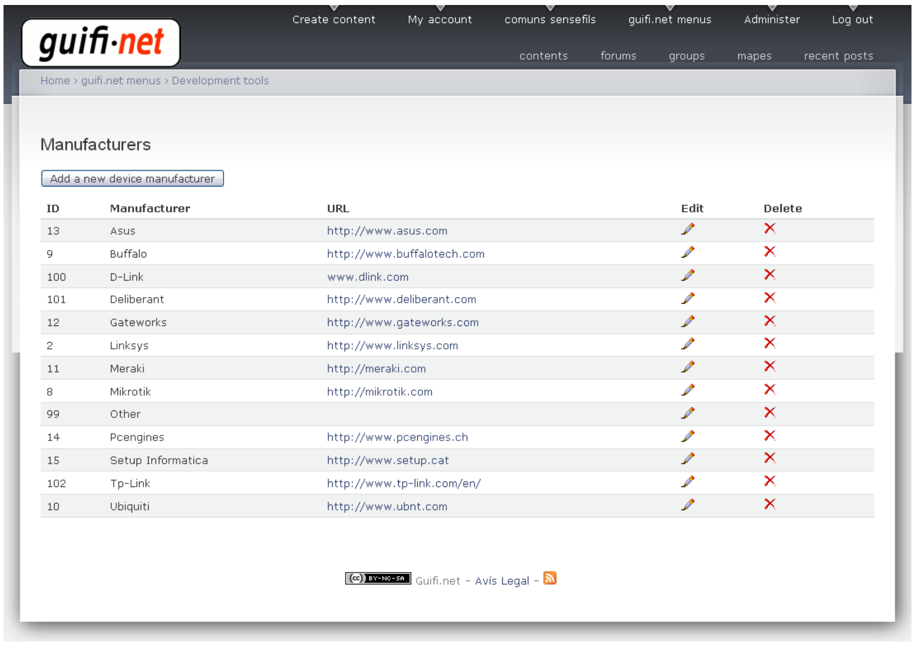
Tots els firmwares:
AirOsv221
AirOsv3.6+
AirOsv30
AirOsv52
Alchemy
DD-guifi
DD-WRTv23
DD-WRTv24preSP2
Freifunk-BATMAN
Freifunk-OLSR
gsffirm
GuifiStationOS1.0
kamikaze
n/a
RouterOSv2.9
RouterOSv3.x
RouterOSv4.0+
RouterOSv4.7+
RouterOSv5.x

network figures & facts

Nodes	36.020
Working	22.790
Planned	10.458
Inactive	1.724
Testing	529
Building	367
Reserved	130
Dropped	22

Wireless links	#	kms.
ap/client	24.316	31.449
PtP link	2.406	10.335
Total	26.722	41.784

Figure 3.4: Edit a device model.



Home › guifi.net menus › Development tools

Manufacturers

ID	Manufacturer	URL	Edit	Delete
13	Asus	http://www.asus.com	✎	✗
9	Buffalo	http://www.buffalotech.com	✎	✗
100	D-Link	www.dlink.com	✎	✗
101	Deliberant	http://www.deliberant.com	✎	✗
12	Gateworks	http://www.gateworks.com	✎	✗
2	Linksys	http://www.linksys.com	✎	✗
11	Meraki	http://meraki.com	✎	✗
8	Mikrotik	http://mikrotik.com	✎	✗
99	Other		✎	✗
14	Pcengines	http://www.pcengines.ch	✎	✗
15	Setup Informatica	http://www.setup.cat	✎	✗
102	Tp-Link	http://www.tp-link.com/en/	✎	✗
10	Ubiquiti	http://www.ubnt.com	✎	✗

Guifi.net - Avís Legal -

Figure 3.5: Available manufacturers list.

3.2.2 Remove 'ad-hoc' zone mode

Having 'ad-hoc' name to distinguish a zone not makes sense because of, as explained in 2.1, Mesh networks are more than wireless ad-hoc networks.

Also, the restrictions imposed in these zones limit and difficult a lot the documentation in the website of a real Mesh network.

For that reason, to adapt the web to the real situation, changes introduced in the Drupal module for the old Mesh (commented in section 2.2.2), are reverted and deleted.

It means that now:

- There are no zone mode types.
- There is no limitations regarding the zone mode.
  - No limit of the number of wireless devices.
  - No limit of the number of radios per device.
- It allows the possibility to create a node with several mixed devices and even a device with multiple mixed radios for Mesh and non Mesh links.
- The IP for mesh does not depends on the zone but the type of radio of the device (see next section 3.2.3).

Source code changes are in the Appendix A.

3.2.3 New IP and radio types 'Mesh'

Due to the need to distinguish a device that is going to connect into a Mesh network, after a lot of thinking, a new radio type is created in the database.

Supernodes have a specific type of radio 'AP' to allow the device to set a point to point link. Client nodes have a specific type too to make a link to a supernode. So, it makes sense to have a new radio type for Mesh devices because they have some particularities when linking.

At the moment in the website the only particularity is that in this new radio type is not allowed to create p2p links and not allowed to accept clients (because in Mesh there are nor clients or supernodes, only Mesh-nodes).

For the same reason, a new type of IP address is created to be assigned in these new radios. And also to have a certain range of IP addresses reserved for Mesh networks, as commented above in section 3.1.3.

Changes introduced in th source code can be found in the Appendix A.

3.2.4 Change IP auto-assignation for Mesh radio

In section 3.1.3 is detailed why a new Mesh node have to have a wide IP range.

That is why the source code has been changed to assign a new /27 IP range of the type 'Mesh' to the devices that have a radio of type 'Mesh'.

Changes can be found in the Appendix A.

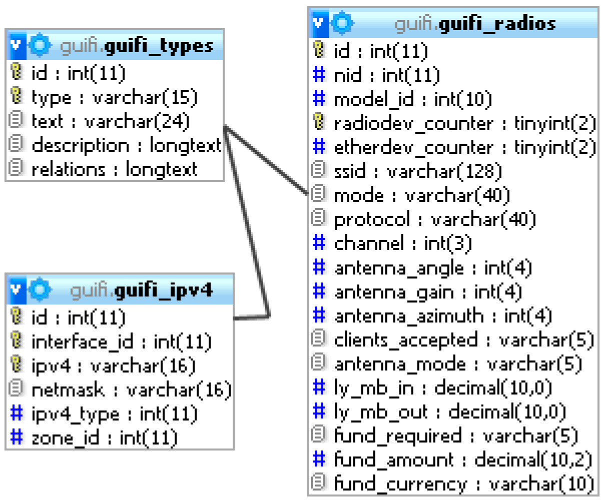


Figure 3.6: Guifi database types relations with ipv4 and radios.

3.3 Oneclick (unsolclic)

'Oneclick' (unsolclic in catalan) is a feature of the Drupal module, but due to the work done on it, it is decided to dedicate this part only for it instead of including it in the previous section which talks about the module.

This Drupal module feature allows to download a configuration file or a configuration script that is automatically generated by the website, based on the firmware selected in the node's device page.

Then, with this file or script, the users can upload or execute into their devices to configure them without efforts in one click (or a few clicks).

This section is focused in the 'oneclick' made in this project specifically for the qMp firmware, for that reason, additional information about the oneclick feature can be found in this external document [9].

3.3.1 For the qMp firmware

First of all to be able the 'oneclick' system to work with a new firmware it should have to be added before, as commented in section 3.2.1.

Because of the need to generate a configuration file for every device supported by qMp without depending on the device, this new 'oneclick' for qMp is generated in the "old way" (see [9]).

It means that the 'oneclick' configuration file is generated by a new firmware specific source code for all the devices supported by qMp. It is done in that way because of doing through the web interface forces you to create one configuration file per model-firmware (i.e. one file for each model that supports qMp), and it makes no sense to repeat many times the same file because it not depends on the device.

The source code is in the Appendix A.1, and below are summarized the steps of the process to generate the ‘oneclick’ file:

1. Check if device has a firmware assigned.
2. Load ‘oneclick’ file of the corresponding firmware.
3. Check the firmware of device to call the function to be executed.
4. It is shown the resulting file.

Example of qMp ‘oneclick’ file:

```
# Generated for:
# qMpv1
#
# qMp Guifi-oneclick v1.0.2
#    __ _  /\/\   _ __
#   / _` |/    \| '_ \
#  | (_| / /\/\ \ |_) |
#   \__, \/    \/ .__/
#      |_|       |_|
#   quick MESH project
#
# Important: You should have 'qmp-guifi' package installed in your node.
#
# To apply this configuration in your node, you can follow this ↩
    instructions:
# http://dev.qmp.cat/projects/qmp/wiki/Guifi_oneclick
#

meshradio='yes'
nodename='BCNTopazi'
devname='BCNTopaziTL'
devmodel='TL-WDR3600'
ipv4='10.1.0.34'
netmask='255.255.255.255'
zoneid='GSF'
```

In the next chapter, in section 4.1, is explained how this values are processed by the qMp system with the new package ‘qmp-guifi’.

Chapter 4

qMp development

In this chapter is detailed all the work done related to qMp.

In the first section 4.1 is explained the new package made which adds some functionalities to qMp for the Guifi.net community. It is explained how the package is built, how it works, and how to get it and update it.
To know more about OpenWRT packages, read this document [5].

The second section 4.2 describes the new website of qMp. It is commented the tools used, the graphic design and the organization in sections of the site.

4.1 qmp-guifi package

One of the main objectives is to integrate qMp in the Guifi.net community, so it is decided for this project to provide diverse features for the Guifi users in Mesh networks with qMp.

Because of qMp is an independent project from Guifi, it is created a new standalone package to be used only in Guifi Mesh networks.

So, for this project it is made this new package called 'qmp-guifi'. It is prepared to contain several features and at the moment is developed one of them. Below is explained it.

The source code is included in the Appendix C

Guifi oneclick

As commented in section 3.3, 'oneclick' is a Guifi.net tool to allow users to configure easily their devices.

Apart from develop the 'oneclick' for the qMp firmware in the Guifi.net site, it is made in this package the option to get this configurations and to apply them in the qMp system.

At the moment of writing this report is released the version 1.0.2 of the package which has implemented this **Guifi oneclick** feature.

This 'qmp-guifi' package feature helps Guifi.net users to configure their qMp devices, via CLI or via LuCI web interface [7], in one (or two) clicks with the Guifi.net site data. It is done with the OpenWRT Unified Configuration Interface (UCI [18]), modifying the qMp configuration file [14].

When a node is configured with **Guifi oneclick** it is set to community mode and it has the IP address, netmask, community ID and SSID changed to the Guifi.net data. It does editing the following configuration options:

- 'qmp.roaming.ignore' is set to 1 to ignore the roaming mode and be in community mode
- 'qmp.networks.publish_lan' is set to 1 to publish the LAN IP into the Mesh.
- 'qmp.networks.disable_lan_dhcp' is set to 0 to enable DHCP for LAN devices.
- 'qmp.networks.lan_address' is set with the got IP address.
- 'qmp.networks.lan_netmask' is set with the got Network mask.
- 'qmp.networks.bmx6_ipv4_address' is set with the IP and the netmask in CIDR format.
- 'qmp.node.community_id' is set with the name of the Device
- 'qmp.@wireless[J].name' all wireless interfaces are set with the SSID formed with Guifi domain + slash + Nodename: "guifi.net/Nodename").
- 'qmp.update.filter' contains the filter to check for updates (explained in section 4.1.3)

4.1.1 Command Line Interface

Configure your node using the command line interface. To do it, there is available a new command: **qmpguifi**.

Below is explained in detail how the command works and also you see the code in the Appendix C.

oneclick

This command option does four options in one to make possible to configure the device with less difficulty and less time.

1. Gets the configuration file.
2. If no errors, checks the downloaded file.
3. If it is a valid 'oneclick' file, prints the values.
4. After user confirmation, configures the node with the previous shown data.

The syntax is:

```
qmpguifi oneclick [URL] <file>
```

- **[URL]**: is the URL of the device. Like: http://guifi.net/guifi/device/"ID"/.
  (Where "ID" is a 5–6 digit number for identify the device in the Guifi.net site database).
- <**file**>: *(optional)* is the temporary file where the 'oneclick' configuration is saved. If it's not specified /tmp/guifi_oneclick is used.

Example:

```
root@qMp6aa2:~# qmpguifi oneclick http://10.139.40.97/guifi/device/53684/

Getting oneclick config:
Done!

Checking oneclick config:
Done!

Showing variables:
 nodename='UPCc6lab104'
 devname='UPCc6lab104-TL'
 devmodel='TL-WDR3600'
 ipv4='10.1.24.33'
 netmask='255.255.255.224'
 zoneid='UPC'

Do you want to configure your node with these settings? [N,y]y

Configuring the node, please wait...

Configuration done!

root@UPCc6lab104-TL6aa2:~#
```

Alternatively it's possible to do the same step by step with individual commands, as you can see below:

get_url

This option calls the URL given and if responses without errors it gets the page into a FILE.

The syntax is:

```
qmpguifi get_url [URL] [FILE]
```

- **[URL]**: is the URL of the device.
- **[FILE]**: is the temporary file where the 'oneclick' configuration is saved.

check

Option that checks if a FILE has the right 'oneclick' configuration file format.

The syntax is:

```
qmpguifi check [FILE]
```

- **[FILE]**: is the temporary file where the 'oneclick' configuration has been previously saved.

print

Option to print the values of the 'oneclick' FILE. *It is recommended to check the file before to avoid errors.*

The syntax is:

```
qmpguifi print [FILE]
```

- **[FILE]**: is the temporary file where the 'oneclick' configuration has been previously saved.

configure

Option to configure the node with the data from the 'oneclick' FILE. *It is* **highly** *recommended to check the file before to avoid errors.*

The syntax is:

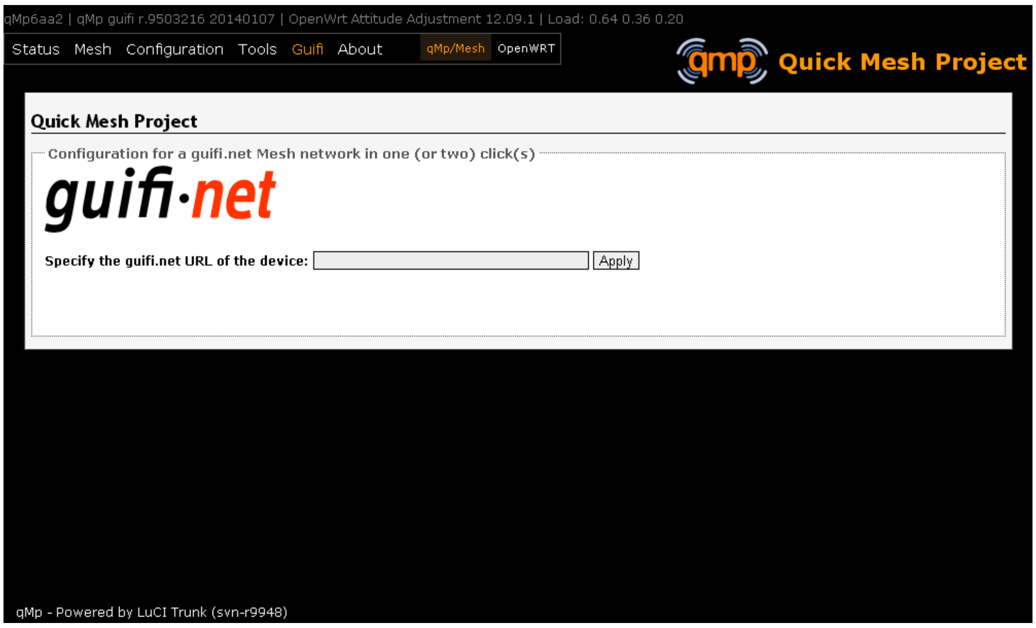
```
qmpguifi configure [FILE]
```

- **[FILE]**: is the temporary file where the 'oneclick' configuration has been previously saved.

4.1.2 LuCI web interface

In addition to the CLI, this package has included a new menu 'Guifi' in the LuCI web interface, with which the device can be configured in really two button clicks.

The interface is very simple and intuitive. It has the Guifi.net logo, a textbox to put the URL of the device and a button to apply the 'oneclick' configuration.


qMp6aa2 | qMp guifi r.9503216 20140107 | OpenWrt Attitude Adjustment 12.09.1 | Load: 0.64 0.36 0.20
Status
Mesh
Configuration
Tools
Guifi
About
qMp/Mesh
OpenWRT
qmp Quick Mesh Project
guifi·net
Apply
qMp - Powered by LuCI Trunk (svn-r9948)

Quick Mesh Project

Configuration for a guifi.net Mesh network in one (or two) click(s)

Specify the guifi.net URL of the device:

Figure 4.1: Guifi 'oneclick' web interface.

Operation

This page is made with a JavaScript code that calls a cgi script, which have the function calls to the main bash script of the package. That is to say, **it uses the same functions as the CLI command** but **through a CGI** called by the JavaScript.

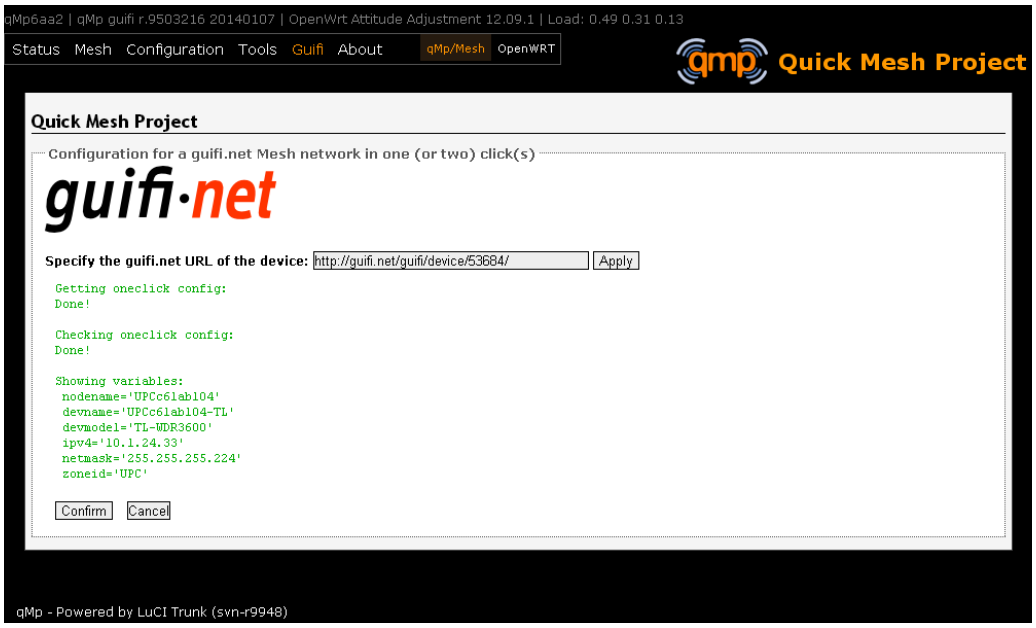
The steps to use it are:

1. Copy the Guifi.net device URL in the textbox.
2. Click "Apply" button.
3. If no errors, values will be printed (see figure 4.2).
4. Click "Confirm" button to configure with the above data, or "Cancel" to stop.
5. If confirmed wait a few minutes until the system will be properly configured.
6. Access to the new IP address.

4.1.3 Install and update of the package

In order to have a qMp node with the package installed it is needed to install it inside a prebuilt image or from a standalone package.

Also it is possible to compile it from your own. Instructions to set the qMp development environment and to compile the system and the package can be read in the Appendix B.


qMp6aa2 | qMp guifi r.9503216 20140107 | OpenWrt Attitude Adjustment 12.09.1 | Load: 0.49 0.31 0.13

Quick Mesh Project

Configuration for a guifi.net Mesh network in one (or two) click(s)

Specify the guifi.net URL of the device: http://guifi.net/guifi/device/53684/ Apply

```
Getting oneclick config:
Done!

Checking oneclick config:
Done!

Showing variables:
 nodename='UPCc6lab104'
 devname='UPCc6lab104-TL'
 devmodel='TL-WDR3600'
 ipv4='10.1.24.33'
 netmask='255.255.255.224'
 zoneid='UPC'
```

qMp - Powered by LuCI Trunk (svn-r9948)

Figure 4.2: Guifi oneclick waiting confirmation.

Installing the package contained in a prebuilt image

The binary images can be found in the qMp firmware images site[1]. Currently only testing branch binaries are released, and they are stored in: */testing/* directory.

The images with the 'qmp-guifi' package included have the label "qMp-Guifi" in the file name. The rest of images (label "qMp") have the standard system without this package.

Once the file is downloaded it is only needed to flash into a supported device and to wait qMp to do the rest.

Note: the flashing method varies depending on the device, each brand have a different way to do it.

Installing the standalone package

It is possible to install only the package in a previous installation of a standard qMp system.

All needed is to download the package from the firmware images site. The package is stored in the packages directory: */testing.generic.ar71xx/packages/*.

The name of the package is **qmp-guifi_X.Y.Z_ar71xx.ipk**, where **X.Y.Z** is the version of the package.

Install it with the OpenWRT opkg utility [11].
For example:

```
opkg install qmp-guifi_1.0.2_ar71xx.ipk
```

[1]qMp firmware: http://fw.qmp.cat

Updating the package and the system

At the moment only manually update is available for the standalone package. It means that it is need to install the new package again as commented above.

On the other hand, the qMp system has an upgrade system to allow users to update their devices preserving all personal configurations.

The ‘qmp-guifi’ package modifies the ‘filter’ parameter from ‘update’ section in the ‘qmp config’ file, which is the variable that the upgrade system uses to check and filter new available images. The filter is set to search only for "Guifi" and "sysupgrade" words in the images file names; using a regular expression:

```
qmp.update.filter=Guifi.*sysupgrade
```

To understand better how the qMp upgrade system works, more detailed info can be read in this article in the qMp wiki [15].

4.2 New user-friendly website

As commented in chapter 2.3.2, the main qMp website is made with the Redmine project manager. This web is a great tool to manage the project but it is mostly development oriented, and also has an unattractive design for common users. That is why some qMp members started to talk about the need of a user-friendly and user oriented new website apart from the development one.

Then, they took the decision to leave the **Redmine site for development** under a new ‘dev’ subdomain[2], and to make a **new website for users** in the root domain[3].

As commented above, the old web was focused on the development but the needs are to introduce qMp into the Guifi.net community, that is why it is decided to create this new user-oriented website in the scope of this project.

In this section are explained the tools used to make the new website. And also the design decisions are detailed, both graphic and content.

4.2.1 CMS: gpEasy

After talking with the qMp members, it is decided to use the gpEasy CMS instead of other CMSs because of its features fits in the needs of the web project; that are mainly to be fast and simple. Below are detailed the gpEasy features.

A CMS (content management system) is a web application to create and to manage multimedia and digital contents, usually through a centralized interface with all the options. Some of the most popular CMS are Drupal, Joomla, Wordpress, among others.

[2]qMp development website: http://dev.qmp.cat
[3]qMp user-friendly website: http://qmp.cat

gpEasy is a CMS that has three main features that makes it the best option to meet the needs of this project; is fast and lightweight, easy and intuitive, it is free.

Is **fast and lightweight** because it do not uses databases to store the information. It is based in PHP, and all configuration and management data are stored in static files, resulting in increased speed by not having to query the database.

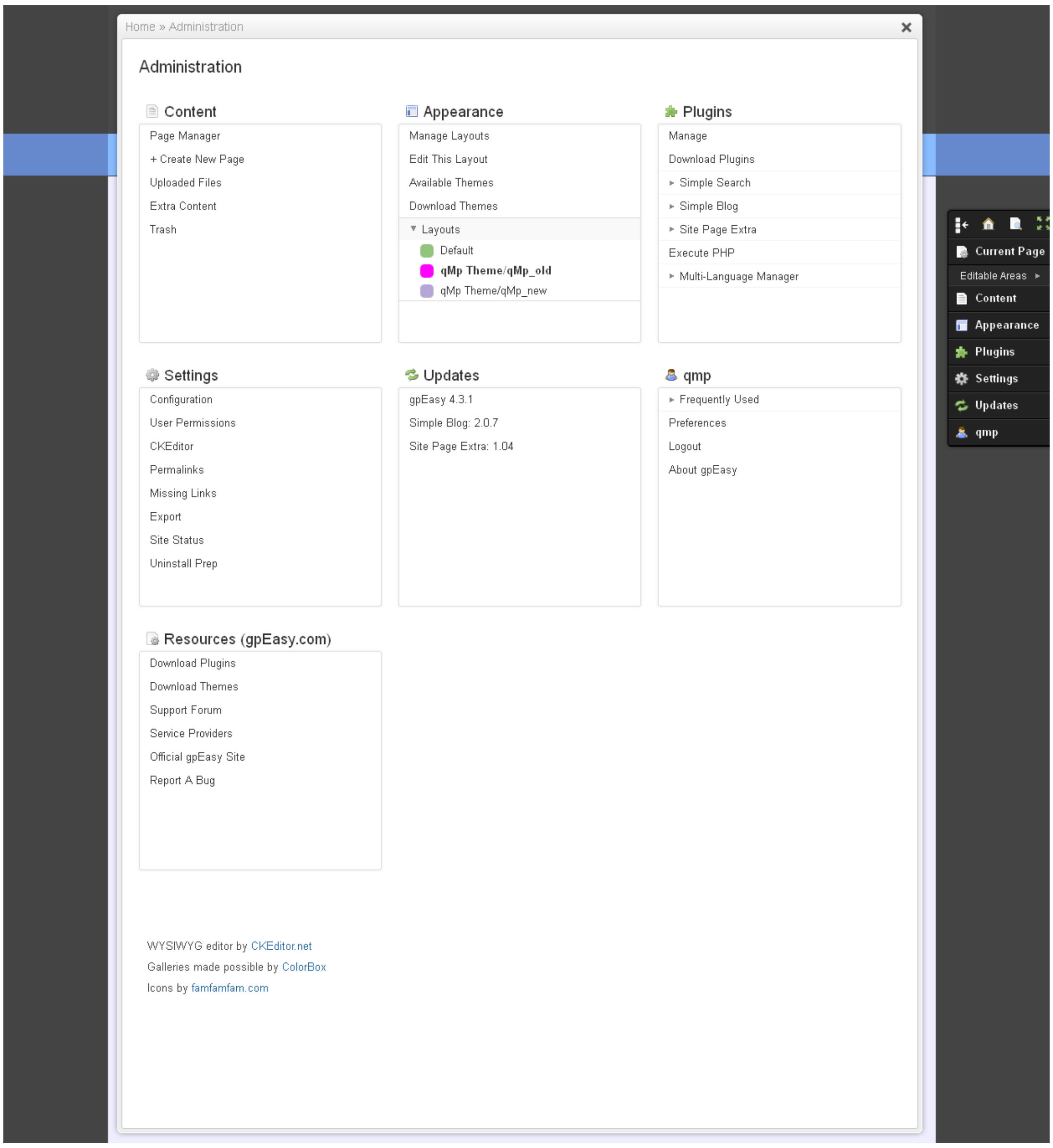
Something very important too is the **easy of use**. From the beginning gpEasy allows to modify the web content in an easy and intuitive way, and with some web layout and development knowledge it is possible to change the appearance of the site. The installation of the CMS is very simple and it does not take much time, among other reasons, because of it is not necessary to install nor to configure any database.

It is **Free Software**, so is in accordance with the objectives of this project.

gpEasy has an administration panel where to manage all the contents, users, tools and settings. As can be seen in the figure 4.3 it has these options:

- **Content**: here are the tools to manage the pages where te contents will be. And there is a file uploader to be able to add files to those pages (for example: images).
- **Appearance**: this section allows to edit the web layout and the themes.
  To add more themes is the option to download new ones, or can be added locally.
- **Plugins**: to manage and to get the plugins, and to configure the installed ones.
- **Settings**: here are the options to manage the CMS, with configuration options, user permissions, site backup and uninstall, etcetera.
- **Updates**: where the new updates are shown when available and to apply them.
- **Logged user**: there is also a section with options to change the password or email.
- **External resources**: at the bottom of the panel there are several links to the gpEasy website to get plugins, get themes, report bugs, etcetera.

Also there is a floating panel that is visible always in all the pages when a user is logged to have on hand the tools of the CMS.


Home » Administration

Administration

Content
- Page Manager
- + Create New Page
- Uploaded Files
- Extra Content
- Trash

Appearance
- Manage Layouts
- Edit This Layout
- Available Themes
- Download Themes
- ▾ Layouts
  - Default
  - **qMp Theme/qMp_old**
  - qMp Theme/qMp_new

Plugins
- Manage
- Download Plugins
- ▸ Simple Search
- ▸ Simple Blog
- ▸ Site Page Extra
- Execute PHP
- ▸ Multi-Language Manager

Settings
- Configuration
- User Permissions
- CKEditor
- Permalinks
- Missing Links
- Export
- Site Status
- Uninstall Prep

Updates
- gpEasy 4.3.1
- Simple Blog: 2.0.7
- Site Page Extra: 1.04

qmp
- ▸ Frequently Used
- Preferences
- Logout
- About gpEasy

Resources (gpEasy.com)
- Download Plugins
- Download Themes
- Support Forum
- Service Providers
- Official gpEasy Site
- Report A Bug

WYSIWYG editor by CKEditor.net
Galleries made possible by ColorBox
Icons by famfamfam.com

Figure 4.3: gpEasy CMS administration page.

4.2.2 Graphic design

As mentioned in the previous section, the CMS used allows the creation of themes. They make possible to do with CSS a template to have the layout and colors style of the website.

Graphic design is inspired in the actual qMp logotype blue and orange colors.
Basically is set with these kinds of blue colors the relevant information of the web, like the Menus and hyperlinks mainly.

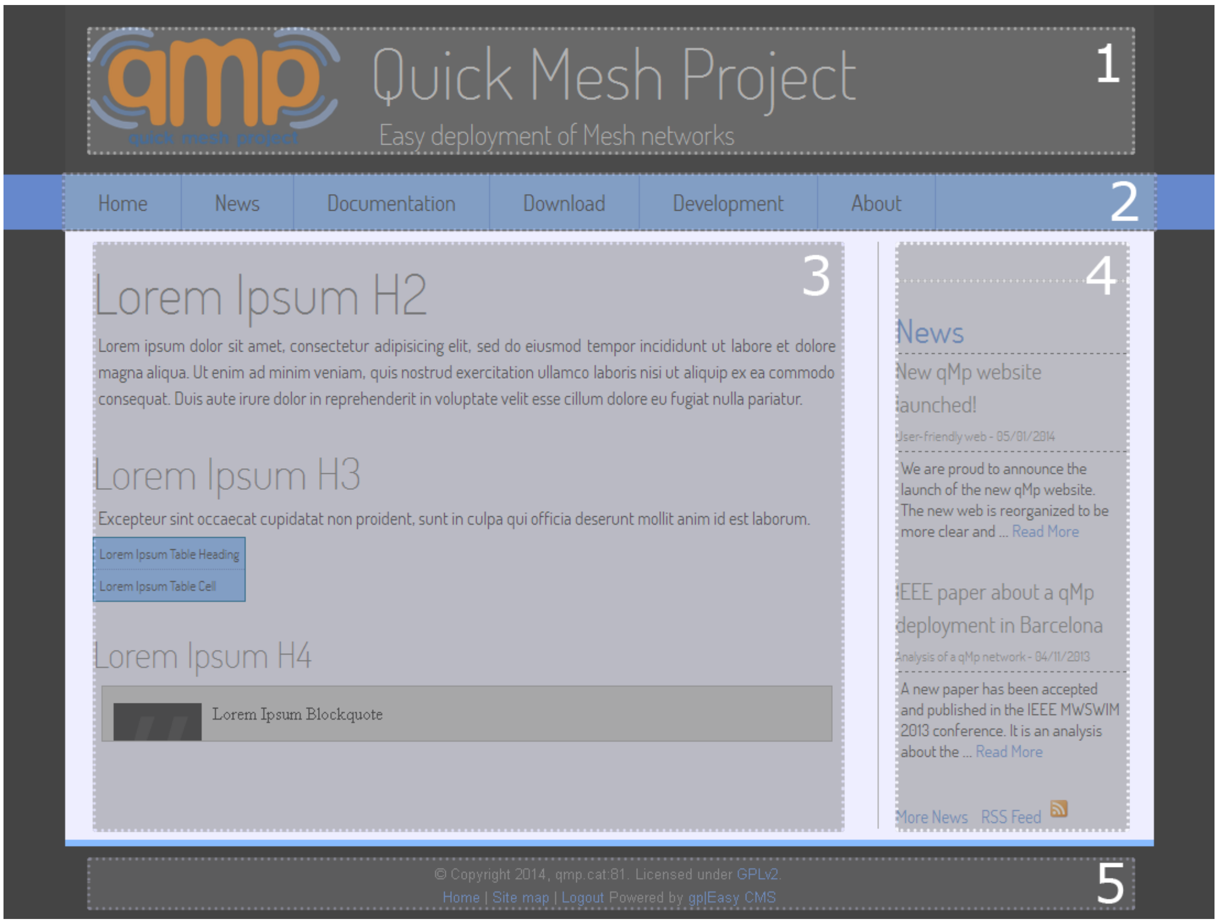
To make the whole page, the structure, the contents, the design and the harmony has been taken into account, that is, the election and distribution of colors and text have been determined with the objective to make a comfortable web for the user.

For the content part, blue color has been elected to maintain a concordance with the whole design, but in a very light blue in contrast with the dark gray of text. For the visual concentration is important to read the screen, that emits light, having light backgrounds with contrasted dark text, because of then the eyes are less forced and it is also good for the visual acuity. To obtain more information about the screens and the eyes, this articles of the prestigious ophthalmological divulgation blog can be read [12] [19].

For the background a medium gray palette of colors has been chosen to complement the blue and orange color, because in the symbology it represents a neutral, elegant and sober color. Also having a gray background, especially with photographies with very dark or very light colors, our eyes observe a better detail of this extreme colors than having a black or white background color.

Layout of the site is divided basically in five parts (see figure 4.4):

1. Header; where is the logotype and title.
2. Menu; contains the section and subsection pages.
3. Main; where the info is the section content.
4. Side menu; column for additional information.
5. Footer; the copyright, site map and login page.



Lorem Ipsum H2

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.

Lorem Ipsum H3

Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Lorem Ipsum Table Heading
Lorem Ipsum Table Cell

Lorem Ipsum H4

Lorem Ipsum Blockquote

News

New qMp website aunched!

ser-friendly web - 05/01/2014

We are proud to announce the launch of the new qMp website. The new web is reorganized to be more clear and ... Read More

EEE paper about a qMp deployment in Barcelona

nalysis of a qMp network - 04/11/2013

A new paper has been accepted and published in the IEEE MWSWIM 2013 conference. It is an analysis about the ... Read More

More News RSS Feed

© Copyright 2014, qmp.cat:81. Licensed under GPLv2.

Home | Site map | Logout Powered by gplEasy CMS

Figure 4.4: qMp web layout.

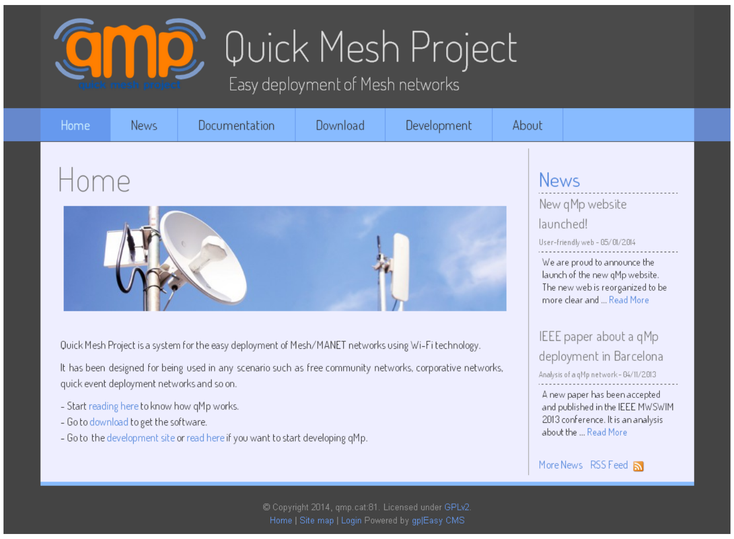
4.2.3 Sections

The content of the new web site is now divided into section and subsection pages to be more organized, to make users easier to find what they are looking for.

Below are detailed the web sections with its contents.

Home

The home page contains a brief explanation of qMp and links to the main sections: Documentation, Download and Development.



Home

Quick Mesh Project is a system for the easy deployment of Mesh/MANET networks using Wi-Fi technology.

It has been designed for being used in any scenario such as free community networks, corporative networks, quick event deployment networks and so on.

- Start reading here to know how qMp works.
- Go to download to get the software.
- Go to the development site or read here if you want to start developing qMp.

News

New qMp website launched!

User-friendly web - 05/01/2014

We are proud to announce the launch of the new qMp website. The new web is reorganized to be more clear and ... Read More

IEEE paper about a qMp deployment in Barcelona

Analysis of a qMp network - 04/11/2013

A new paper has been accepted and published in the IEEE MWSWIM 2013 conference. It is an analysis about the ... Read More

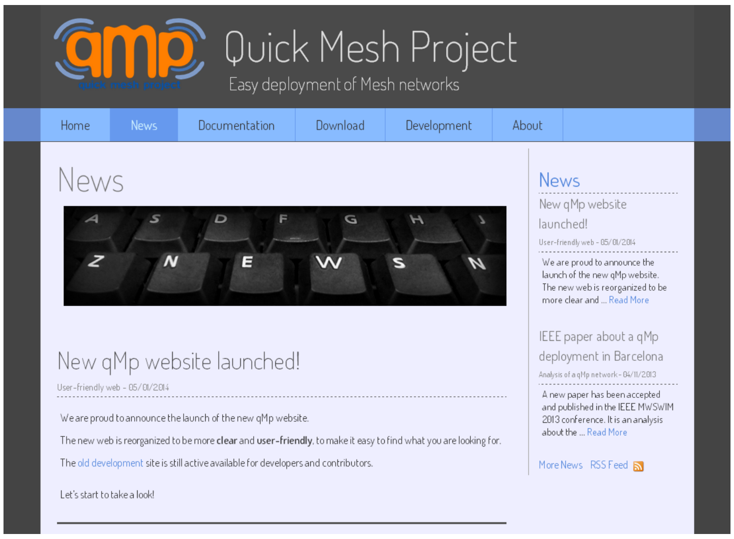
More News RSS Feed

© Copyright 2014, qmp.cat:81. Licensed under GPLv2.
Home | Site map | Login Powered by gp|Easy CMS

News

This section is made with the Simple Blog plugin[4].
It is a nice and easy way to manage and to publish the news related to qMp because of its features:

- Allows to edit posts with a WYIWYG editor.
- Have categories to organize the posts.
- Have a gadget to summarize the last posts (for example in side column).
- And more...



News

New qMp website launched!

User-friendly web - 05/01/2014

We are proud to announce the launch of the new qMp website.

The new web is reorganized to be more **clear** and **user-friendly**, to make it easy to find what you are looking for.

The old development site is still active available for developers and contributors.

Let's start to take a look!

News

New qMp website launched!

User-friendly web - 05/01/2014

We are proud to announce the launch of the new qMp website. The new web is reorganized to be more clear and ... Read More

IEEE paper about a qMp deployment in Barcelona

Analysis of a qMp network - 04/11/2013

A new paper has been accepted and published in the IEEE MWSWIM 2013 conference. It is an analysis about the ... Read More

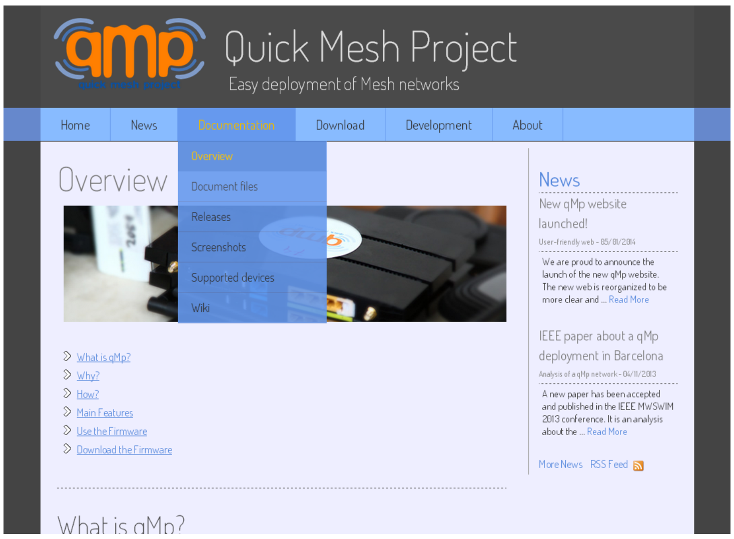
More News RSS Feed

[4]Simple Blog pluggin: http://gpeasy.com/Plugins/17_Simple_Blog

Documentation

The documentation is divided into subsections to organize it better.

- Overview; (is the same page as parent menu) a general explanation of What is qMp, its basics, the main Features and a basic use.
- Document files; links to the development website document section[5].
- Releases; it is a place to revise the released versions of qMp.
- Screenshots; some pictures showing the qMp web management interface.
- Supported devices; a list of the devices supported by qMp.
- Wiki; a link to the wiki of the development site[6].



New qMp website launched!

User-friendly web - 05/01/2014

We are proud to announce the launch of the new qMp website. The new web is reorganized to be more clear and ... Read More

IEEE paper about a qMp deployment in Barcelona

Analysis of a qMp network - 04/11/2013

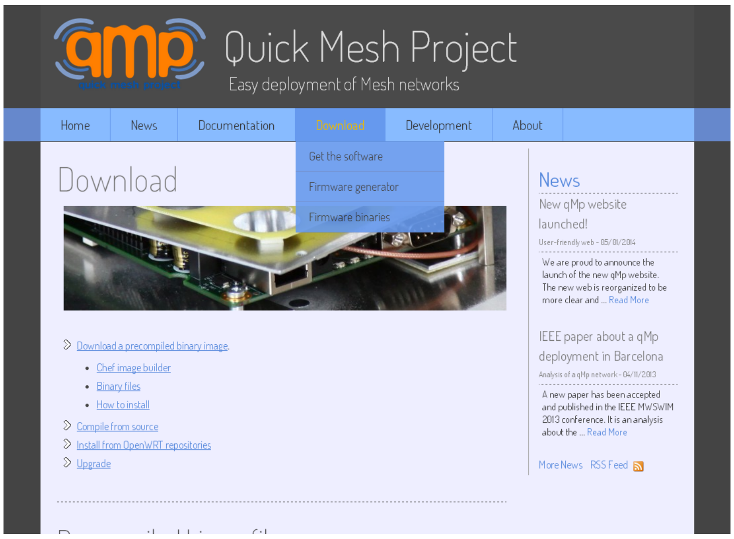
A new paper has been accepted and published in the IEEE MWSWIM 2013 conference. It is an analysis about the ... Read More

Download

The Download page explains the ways to get qMp and how to update it.

The submenus have the links to the qMp firmware images site and to the Chef image builder site.[7]

[5]qMp dev-site documents: http://dev.qmp.cat/projects/qmp/documents
[6]qMp dev-site wiki: http://dev.qmp.cat/projects/qmp/wiki/
[7]Chef image builder: http://chef.qmp.cat


Quick Mesh Project

Easy deployment of Mesh networks

Home | News | Documentation | Download | Development | About

Get the software

Firmware generator

Firmware binaries

Download

- Download a precompiled binary image.
  - Chef image builder
  - Binary files
  - How to install
- Compile from source
- Install from OpenWRT repositories
- Upgrade

News

New qMp website launched!

User-friendly web - 05/01/2014

We are proud to announce the launch of the new qMp website. The new web is reorganized to be more clear and ... Read More

IEEE paper about a qMp deployment in Barcelona

Analysis of a qMp network - 04/11/2013

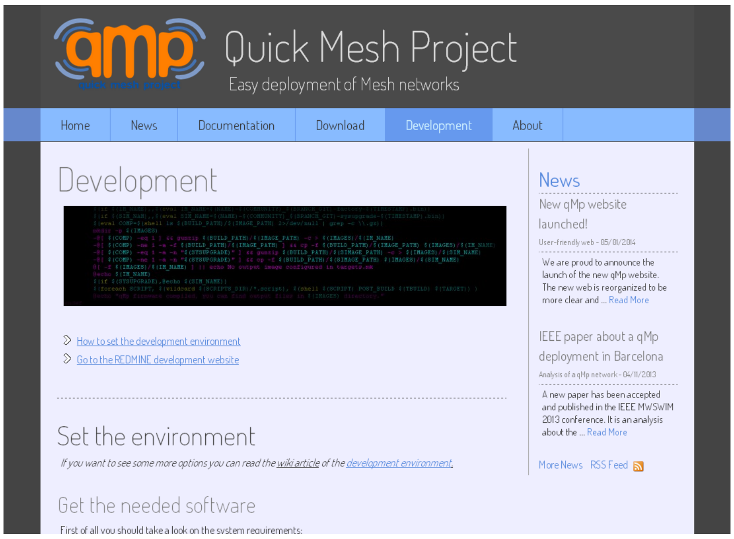
A new paper has been accepted and published in the IEEE MWSWIM 2013 conference. It is an analysis about the ... Read More

More News RSS Feed

Development

Here there are the basic instructions to set the development environment and the instructions to compile the source code.

Also there's a link to the qMp development site (Redmine).


Quick Mesh Project

Easy deployment of Mesh networks

Home | News | Documentation | Download | Development | About

Development

```
$(if $(IM_NAME),,$(eval IM_NAME=$(NAME)-$(COMMUNITY)_$(BRANCH_GIT)-factory-$(TIMESTAMP).bin))
$(if $(SIM_NAM),,$(eval SIM_NAME=$(NAME)-$(COMMUNITY)_$(BRANCH_GIT)-sysupgrade-$(TIMESTAMP).bin))
$(eval COMP=$(shell ls $(BUILD_PATH)/$(IMAGE_PATH) 2>/dev/null | grep -c \\.gz))
mkdir -p $(IMAGES)
-@[ $(COMP) -eq 1 ] && gunzip $(BUILD_PATH)/$(IMAGE_PATH) -c > $(IMAGES)/$(IM_NAME)
-@[ $(COMP) -ne 1 -a -f $(BUILD_PATH)/$(IMAGE_PATH) ] && cp -f $(BUILD_PATH)/$(IMAGE_PATH) $(IMAGES)/$(IM_NAME)
-@[ $(COMP) -eq 1 -a -n "$(SYSUPGRADE)" ] && gunzip $(BUILD_PATH)/$(SIMAGE_PATH) -c > $(IMAGES)/$(SIM_NAME)
-@[ $(COMP) -ne 1 -a -n "$(SYSUPGRADE)" ] && cp -f $(BUILD_PATH)/$(SIMAGE_PATH) $(IMAGES)/$(SIM_NAME)
@[ -f $(IMAGES)/$(IM_NAME) ] || echo No output image configured in targets.mk
@echo $(IM_NAME)
$(if $(SYSUPGRADE),@echo $(SIM_NAME))
$(foreach SCRIPT, $(wildcard $(SCRIPTS_DIR)/*.script), $(shell $(SCRIPT) POST_BUILD $(TBUILD) $(TARGET)) )
@echo "qMp firmware compiled, you can find output files in $(IMAGES) directory."
```

- How to set the development environment
- Go to the REDMINE development website

Set the environment

If you want to see some more options you can read the wiki article of the development environment.

Get the needed software

First of all you should take a look on the system requirements:

News

New qMp website launched!

User-friendly web - 05/01/2014

We are proud to announce the launch of the new qMp website. The new web is reorganized to be more clear and ... Read More

IEEE paper about a qMp deployment in Barcelona

Analysis of a qMp network - 04/11/2013

A new paper has been accepted and published in the IEEE MWSWIM 2013 conference. It is an analysis about the ... Read More

More News RSS Feed

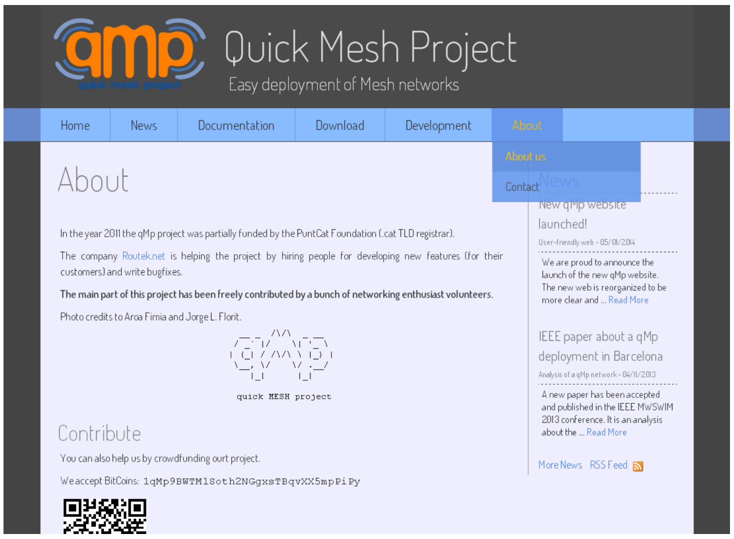
About

The last section contains the information about the project.

It is commented who are the qMp contributors and collaborators, which basically are free volunteers interested in computer networks. Also there is a small company, Routek[8], that helps the project.

Furthermore, there is a Bitcoin[9] wallet to receive donations to support the project.

The contact submenu has the links to the mailing lists and a form to contact directly with the qMp team.



About

In the year 2011 the qMp project was partially funded by the PuntCat Foundation (.cat TLD registrar).

The company Routek.net is helping the project by hiring people for developing new features (for their customers) and write bugfixes.

The main part of this project has been freely contributed by a bunch of networking enthusiast volunteers.

Photo credits to Aroa Fimia and Jorge L. Florit.

quick MESH project

Contribute

You can also help us by crowdfunding ourt project.

We accept BitCoins: `1qMp9BWTM1Soth2NGgxsTBqvXX5mpPiPy`

News

New qMp website launched!

User-friendly web - 05/01/2014

We are proud to announce the launch of the new qMp website. The new web is reorganized to be more clear and ... Read More

IEEE paper about a qMp deployment in Barcelona

Analysis of a qMp network - 04/11/2013

A new paper has been accepted and published in the IEEE MWSWIM 2013 conference. It is an analysis about the ... Read More

More News RSS Feed

[8]Routek: http://routek.net

[9]Bitcoin: http://bitcoin.org

Chapter 5

Budget estimation

Since this is an academic project, the budget below is estimated based on what in a commercial world could be.

We divide the costs into three sections and final total expenses.

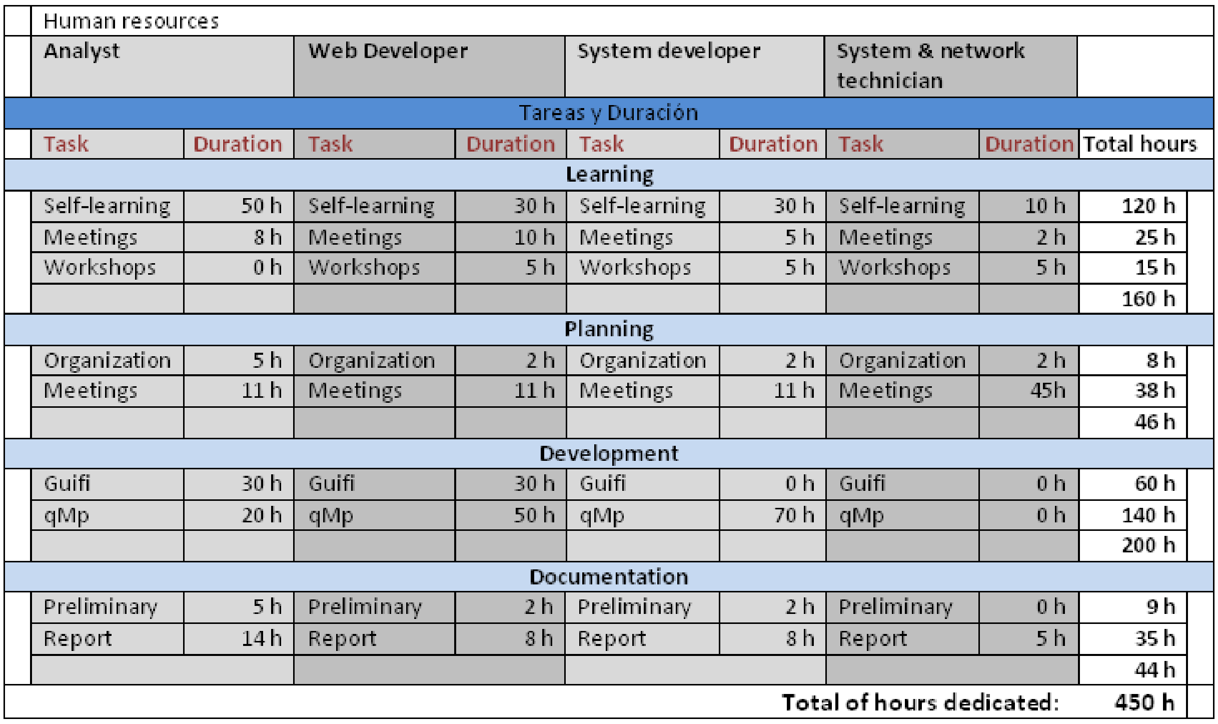
5.1 Human resources

There have been defined four roles, because in a real project it is not made by an only one person and then we have divided the work into different professional profiles.

They are:

- **Analyst**: this is the team leader which does the analysis of the requirements of the project and supervise it, he has more work documenting the project because is who revise it and assemble all the parts of work done. Also does development tasks.
- **Web developer**: is the responsible to do all the web development, both Guifi.net and qMp sites.
- **System developer**: his role is to develop for the qMp system.
- **System and network technician**: who makes the installation of the infrastructures (antenna, cables, etc.).

Below it is a planning table with the tasks and expected duration per role.



Human resources								
Analyst		Web Developer		System developer		System & network technician		
Tareas y Duración								
Task	Duration	Task	Duration	Task	Duration	Task	Duration	Total hours
Learning								
Self-learning	50 h	Self-learning	30 h	Self-learning	30 h	Self-learning	10 h	120 h
Meetings	8 h	Meetings	10 h	Meetings	5 h	Meetings	2 h	25 h
Workshops	0 h	Workshops	5 h	Workshops	5 h	Workshops	5 h	15 h
								160 h
Planning								
Organization	5 h	Organization	2 h	Organization	2 h	Organization	2 h	8 h
Meetings	11 h	Meetings	11 h	Meetings	11 h	Meetings	45h	38 h
								46 h
Development								
Guifi	30 h	Guifi	30 h	Guifi	0 h	Guifi	0 h	60 h
qMp	20 h	qMp	50 h	qMp	70 h	qMp	0 h	140 h
								200 h
Documentation								
Preliminary	5 h	Preliminary	2 h	Preliminary	2 h	Preliminary	0 h	9 h
Report	14 h	Report	8 h	Report	8 h	Report	5 h	35 h
								44 h
							Total of hours dedicated:	450 h

Then is a table with the estimated expenses of the workers with a half working hours (20 h/week).

If the project has to be 450 hours long, it will be divided by the total working hours (80 h/month per person). It means a project estimated duration of 1.4 months with full dedication.

Profile	Cost (€/h)	Monthly salary (20 h/week)	Cost to the company
Analyst	18 €/h	1,440 €	2,016 €
Web developer	14 €/h	1,120 €	1,568 €
System developer	14 €/h	1,120 €	1,568 €
System and network technician	14 €/h	1,120 €	1,568 €

5.2 Hardware

Related to the hardware expenses, in this project has been needed some wireless routers and tools.

Here are detailed the costs of the hardware bought:

Description	**Units**	**Unit price**	**Total**
UBNT Nanostation M5	1	72.46 €	72.46 €
TPlink TL-WDR3600	1	64.49 €	64.49€
UBNT Tough Cable PRO CAT 5e - 304m	1	127.88 €	127.88 €
RJ45 crimping tool for UPT, FTP cables	1	7.55 €	7.55 €
Antenna mast and screws	1	8.43 €	8.43 €
Total:			280.81 €

5.3 Software

All the software used in this project is Free Software without commercial licenses.

So, for this subject there is no economic cost.

5.4 Total

Finally, the total estimated expenses are shown in the table below.

Human resources:	6,720 €
Hardware:	280.81 €
Total:	**7,000.81 €**

Chapter 6

Conclusions

Finally, along the work done, almost all proposed objectives have been achieved.

The main objective of the project was to integrate and to facilitate the deployment of qMp nodes in the Guifi.net community. This goal has been achieved by means of the following actions:

1. Adapting the Guifi.net website to make it possible to add Mesh nodes and zones.
2. Adding qMp as an available firmware in the Guifi.net site.
3. Adding the support to qMp to the ‘oneclick’ feature of the Guifi.net web.
4. Creating the package ‘qmp-guifi’ to provide the qMp system options to the Guifi.net users. At the moment has been created an option to configure the device using the Guifi.net ‘oneclick’ for qMp.
5. It has been designed and created a new user-friendly qMp web.
6. Creating documentation in the Guifi.net and qMp websites. More specifically, the Guifi.net wiki has been provided with two articles; one explaining how to deploy new Mesh nodes with qMp in the existing Mesh networks of Guifi.net, and another one explaining how to start a new Mesh zone in the website.
   Also has been made an article of the ‘qmp-guifi’ package in the qMp wiki, explaining how to use it, how to install it and how to update it.

We can see that Mesh networks have been growing in Guifi.net since the beginning of the project. Nowadays there are more than fifty qMp nodes deployed in the city of Barcelona.

It demonstrates that the changes made in this project have been taken effect into the community because of the people is deploying Mesh networks with qMp more and more. Also some members started to contribute too to the documentation in the Guifi wiki.

Several meetings about qMp and Mesh networks have been realized among Guifi.net users and administrators, qMp developers and foreign people from other Mesh networks around the world. And now it is planned to do more in future and to continue to reinforce this networks.

Is still a lot of work to be done related these topics, since this is a project which is involved into Guifi.net and qMp and they are both projects in continuous growth. And I want to continue contributing them.

Regarding qMp there is a long list of things to do. One of the most important related to this project is to improve the system to allow an easy configuration of the frontier nodes, which are the link between Mesh network and the rest of Guifi.net. This system was partially developed in the Google Summer of Code[1][2] by another student, but it has not been tested yet in production nodes, and it does not have an easy way to install and to configure.

Also there is an interesting feature to add to the 'qmp-guifi' package to allow the qMp system to interact with the Guifi.net API. It will allow qMp users to create their nodes in the Guifi.net website directly from the qMp system, among other things. This subject is already planned to do in another degree project by a student member of Guifi.net.

Other possible improvements to be developed to qMp related to Guifi.net. Some of them are to modify the Guifi.net maps to paint qMp nodes and links, to add a SNMP feature in the qMp system to make the Guifi.net website to interact to them to generate the bandwidth graphics; among others.

The 'oneclick' feature is also open to new improvements to allow more configuration options, such as devices with multiple radios, mixed devices (i.e. Mesh and not mesh links in the same device), support to personalized parameters, etcetera.

This project has been a starting point to enter in the community networks world, with which I have acquired a lot of experience and knowledge in wireless networks and in systems development.

The possibility to join in a great social initiative and big community brings me the possibility to meet other people with the same interests, both from Spain and from other countries. And also it has brought me the opportunity to join to the Quick Mesh Project team, which has been my first experience developing free software.

[1] Google Summer of Code web: http://www.google-melange.com

[2] News of the work done in the GSoC: http://qmp.cat/News/1_Google_Summer_of_Code_and_qMp

Appendices

A Guifi.net drupal module source changes

Below are only the modified lines to add the functionalities commented in chapter 3, section 3.2.

To see the full source code, go to the Guifi Gitorious repository:
http://gitorious.org/guifi/drupal-guifi/

guifi.install

```php
  // --
  // -- IPv4 types
  // --
  db_query("INSERT INTO {guifi_types} (id, type, text, description, relations) VALUES ('1', 'ipv4_types', '1', 'public', '');");
  db_query("INSERT INTO {guifi_types} (id, type, text, description, relations) VALUES ('2', 'ipv4_types', '2', 'backbone', '');");
  db_query("INSERT INTO {guifi_types} (id, type, text, description, relations) VALUES ('3', 'ipv4_types', '3', 'ad-hoc mesh - OLSR', 'kamikaze|freifunk-OLSR');");
  db_query("INSERT INTO {guifi_types} (id, type, text, description, relations) VALUES ('4', 'ipv4_types', '4', 'ad-hoc mesh - BATMAN', 'kamikaze|freifunk-BATMAN');");
  db_query("INSERT INTO {guifi_types} (id, type, text, description, relations) VALUES ('5', 'ipv4_types', '5', 'ad-hoc mesh - BMX', 'kamikaze');");
  db_query("INSERT INTO {guifi_types} (id, type, text, description, relations) VALUES ('6', 'ipv4_types', '6', 'ad-hoc mesh - RouterOS', 'RouterOSv3.x');");
  db_query("INSERT INTO {guifi_types} (id, type, text, description, relations) VALUES ('7', 'ipv4_types', '7', 'mesh', NULL);");
  db_query("INSERT INTO {guifi_types} (id, type, text, description, relations) VALUES ('8', 'ipv4_types', '8', 'reserved', NULL);");

  // --
  // -- radio mode types
  // --
  db_query("INSERT INTO {guifi_types} (id, type, text, description, relations) VALUES ('1', 'mode', 'ap', 'AP or AP with WDS', 'ap|client');");
  db_query("INSERT INTO {guifi_types} (id, type, text, description, relations) VALUES ('2', 'mode', 'client', 'Wireless client', 'ap');");
  //db_query("INSERT INTO {guifi_types} (id, type, text, description, relations) VALUES ('3', 'mode', 'bridge', 'Wireless Bridge', 'bridge');");
  //db_query("INSERT INTO {guifi_types} (id, type, text, description, relations) VALUES ('4', 'mode', 'routedclient', 'Routed client', 'ap');");
  db_query("INSERT INTO {guifi_types} (id, type, text, description, relations) VALUES ('5', 'mode', 'mesh', 'Mesh radio', NULL);");
  //db_query("INSERT INTO {guifi_types} (id, type, text, description, relations) VALUES ('9999', 'mode', 'NAT Client', 'NAT Client', 'ap|client');");
  //db_query("INSERT INTO {guifi_types} (id, type, text, description, relations) VALUES ('8888', 'mode', 'Routed Client', 'Routed Client', 'ap|client');");
  //db_query("INSERT INTO {guifi_types} (id, type, text, description, relations) VALUES ('7777', 'mode', 'Bridged Client', 'Bridged Client', 'ap|client');");
```

```php
function guifi_update_910() {
  $items = array();
  // Remove 'zone_mode' field
  $items[] = update_sql("ALTER TABLE {guifi_zone} DROP `zone_mode`;");
  // Add "Mesh" radio and IPv4 types
  $items = update_sql("INSERT INTO {guifi_types} (id, type, text, description, relations) VALUES ('5', 'mode', 'mesh', 'Mesh radio', NULL);");
  $items = update_sql("INSERT INTO {guifi_types} (id, type, text, description, relations) VALUES ('7', 'ipv4_types', '7', 'mesh', NULL);");
  // Add reserved IPv4 type
  $items = update_sql("INSERT INTO {guifi_types} (id, type, text, description, relations) VALUES ('8', 'ipv4_types', '8', 'reserved', NULL);");
  return $items;
}
```

guifi_api.inc.php

```php
function guifi_api_radio_add($gapi, $parameters) {
  if (!guifi_api_check_fields($gapi, array('mode', 'device_id' ), $parameters)) {
    return FALSE;
  }
```

```php
  $radio = _guifi_radio_prepare_add_radio($device);

  $fields = array('mac', 'antenna_angle', 'antenna_gain', 'antenna_azimuth', 'antenna_mode' );
  if ($parameters['mode'] == 'ap') {
    $fields = array_merge($fields, array('ssid', 'protocol', 'channel', 'clients_accepted' ));
  } else if ($parameters['mode'] == 'mesh') {
    $fields = array_merge($fields, array('ssid', 'protocol', 'channel' ));
  }
```

```php
function guifi_api_radio_update($gapi, $parameters) {
  if (!guifi_api_check_fields($gapi, array('device_id', 'radiodev_counter' ), $parameters)) {
    return FALSE;
  }
```

```php
  $radio = $device['radios'][$radiodev_counter];

  $fields = array('mac', 'antenna_angle', 'antenna_gain', 'antenna_azimuth', 'antenna_mode' );
  if ($radio['mode'] == 'ap') {
    $fields = array_merge($fields, array('ssid', 'protocol', 'channel', 'clients_accepted' ));
  } else if ($radio['mode'] == 'mesh') {
    $fields = array_merge($fields, array('ssid', 'protocol', 'channel' ));
  }
```

guifi_devices.inc.php

```php
function guifi_device_form($form_state, $params = array()) {
  global $user;
```

```php
// DEPRECATED IN FAVOR OF MESH (mesh)
/*
  // if device zone_mode was NULL, get the zone mode (ad-hoc or infrastructure)
  if (is_null($form_state['values']['zone_mode'])) {
    $form_state['values']['zone_mode'] = $zone->zone_mode;

    // That's a new device, because zone_mode was NULL, otherwise would have a value
    // If ad-hoc, add a radio & a public IP (/27)
    if ($zone->zone_mode == 'ad-hoc') {
      if ($form_state['values']['type'] == 'radio') {
        $form_state['values']['newradio_mode'] = $zone->zone_mode;
        guifi_radio_add_radio_submit($form,$form_state);
      } else {
        $intf=array();
        $intf['new']=TRUE;
        $intf['interface_type']='wLan/Lan';
        $ips_allocated=guifi_ipcalc_get_ips('0.0.0.0','0.0.0.0',$edit,1);
        $net = guifi_ipcalc_get_subnet_by_nid($form_state['values']['nid'],'255.255.255.224','public',$ips
          TRUE);
        $i = _ipcalc($net,'255.255.255.224');
        guifi_log(GUIFILOG_TRACE,"IPS allocated: ".count($ips_allocated)." got net: ".$net.'/27',$i);
        $intf['ipv4'][0]=array();
        $intf['ipv4'][0]['new']=TRUE;
        $intf['ipv4'][0]['ipv4_type']=1;
```

```
          $intf['ipv4'][0]['ipv4']=$net;
          guifi_log(GUIFILOG_TRACE,"Assigned IP: ".$intf['ipv4'][0]['ipv4']);
          $intf['ipv4'][0]['netmask']='255.255.255.224';
          $form_state['values']['interfaces'][0] = $intf;

        }
      }

    }
*/
```

```
function guifi_device_create_form($form_state, $node) {

  $types = guifi_types('device');

  $zone = guifi_zone_load($node->zone_id);
-   if ($zone->zone_mode == 'ad-hoc') {
-     $rows = db_result(db_query('SELECT count(*) FROM {guifi_devices} WHERE type="radio" AND nid=%d',$node->nid));
-     if ($rows)
-       unset($types['radio']);
-   }
```

guifi_ipv4.inc.php

```
/**
 * Present the guifi zone editing form.
 */
function guifi_ipv4_form($form_state, $params = array()) {
  guifi_log(GUIFILOG_TRACE,'guifi_ipv4_form()',$params);

  $network_types = array('public'   => t('public - for any device available to everyone'),
                         'backbone' => t("backbone - used for internal management, links..."),
                         'mesh' => t('mesh - for any device in Mesh'),
                         'reserved' => t('reserved - used for reserved addressing'));

  // $network_types = array_merge($network_types,guifi_types('adhoc'));
```

```
function guifi_ipv4_form_validate($form,$form_state) {
```

```
  $zone = guifi_zone_load($form_state['values']['zone']);

/* NO MORE AD-HOC ZONES
  if (($zone->master != 0) and
      ($zone->zone_mode != 'ad-hoc')) {
    if (!in_array($form_state['values']['network_type'],array('public','backbone')))
      form_set_error('network_type',
        t('Only ad-hoc/mesh or root zones can have ad-hoc/mesh ranges assigned'));
  }
  if (($zone->zone_mode == 'ad-hoc') and
     ($form_state['values']['network_type'] == 'backbone'))
    form_set_error('network_type',
      t('You must specify the protocol for backbone ranges on ad-hoc/mesh zones'));
*/
}
```

guifi_radios.inc.php

```
-   if ($edit['zone_mode']=='ad-hoc')
-     $modes_arr=array('ad-hoc' => t('Ad-hoc mode for mesh'));
-   else {
```

```
function guifi_radio_radio_form($radio, $key, &$form_weight = -200) {
```

```
    switch ($radio['mode']) {
      case 'ap':
      case 'mesh':
        $f['s']['ssid'] = array(
          '#type' => 'textfield',
          '#title' => t('SSID'),
          '#parents' => array('radios',$key,'ssid'),
          '#required' => TRUE,
          '#size' => 30,
          '#maxlength' => 30,
          '#default_value' => $radio["ssid"],
          '#description' => t("SSID␣to␣identify␣this␣radio␣signal."),
        );
        $f['s']['protocol'] = array(
          '#type' => 'select',
          '#title' => t("Protocol"),
          '#parents' => array('radios',$key,'protocol'),
          '#default_value' =>  $radio["protocol"],
          '#options' => guifi_types('protocol'),
          '#description' => t('Select␣the␣protocol␣where␣this␣radio␣will␣operate.'),
          '#ahah' => array(
            'path' => 'guifi/js/channel/'.$key,
            'wrapper' => 'select-channel-'.$key,
            'method' => 'replace',
            'effect' => 'fade',
          ),
        );
        $f['s']['channel'] =
          guifi_radio_channel_field(
            $key,
            $radio["channel"],
            $radio['protocol']);
        if ($radio['mode'] == 'ap') {
```

```
function _guifi_radio_prepare_add_radio($edit) {
 // next id
```

```
    case 'mesh':
      $radio['antenna_angle'] = 0;
      $radio['clients_accepted'] = "No";
      $radio['ssid'] = $ssid.t('MESH');
      // first radio, force wlan/Lan bridge and get an IP
      if ($tc == 0) {
        $radio['interfaces'][1] = array();
        $radio['interfaces'][1]['new'] = TRUE;
        $radio['interfaces'][1]['interface_type'] = 'wLan/Lan';
        $ips_allocated = guifi_ipcalc_get_ips('0.0.0.0','0.0.0.0',$edit,1);
        // $net = guifi_ipcalc_get_meship($edit['nid'],$ips_allocated);
        $net = ↩
            guifi_ipcalc_get_subnet_by_nid($edit['nid'],'255.255.255.224','mesh',$ips_allocated,'Yes',
           TRUE);
        guifi_log(GUIFILOG_BASIC,"IPS␣allocated:␣".count($ips_allocated)."␣got␣net:␣↩
           ".$net.'/27');
        $radio['interfaces'][1]['ipv4'][$rc] = array();
        $radio['interfaces'][1]['ipv4'][$rc]['new'] = TRUE;
        $radio['interfaces'][1]['ipv4'][$rc]['ipv4_type'] = 1;
        //$radio['interfaces'][1]['ipv4'][$rc]['ipv4'] = $net;
        $radio['interfaces'][1]['ipv4'][$rc]['ipv4'] = long2ip(ip2long($net)+1);
        guifi_log(GUIFILOG_TRACE,"Assigned␣IP:␣" . ↩
            $radio['interfaces'][1]['ipv4'][$rc]['ipv4']);
        $radio['interfaces'][1]['ipv4'][$rc]['netmask'] = '255.255.255.224';
      }
        $radio['mac'] = '';
      break;
  }
```

```php

  $radio['rc'] = $rc;

  return $radio;
}
```

A.1 Guifi oneclick code

Below there are the files of source code related to 'unsolclic' for qMp firmware from chapter 3, section 3.3

1. guifi_unsolclic.inc.php
2. firmware/qmp.inc.php

1. guifi_unsolclic.inc.php (only a partial code which includes de modifications).

```php
  switch ($dev->variable['firmware']) {
    case 'RouterOSv2.9':
    case 'RouterOSv3.x':
    case 'RouterOSv4.0+':
    case 'RouterOSv4.7+':
    case 'RouterOSv5.x':
      unsolclic_routeros($dev);
      exit;
      break;
    case 'DD-guifi':
    case 'DD-WRTv23':
    case 'Alchemy':
    case 'Talisman':
      unsolclic_wrt($dev);
      exit;
      break;
    case 'AirOsv221':
    case 'AirOsv30':
    case 'AirOsv3.6+':
      unsolclic_airos($dev);
      exit;
      break;
// case 'AirOsv52':
// unsolclic_airos52($dev);
// exit;
// break;
    case 'GuifiStationOS1.0':
      unsolclic_guifistationos($dev);
      exit;
      break;
// case 'qMpv1': // Use a generic one is better
    case preg_match('/^qMp/',$dev->variable['firmware']) == 1:
// print_r(preg_match('/^qMpv1/',$dev->variable['firmware']));
      unsolclic_qmp($dev);
      exit;
      break;

  }
```

2. firmware/qmp.inc.php

```php
<?php

function unsolclic_qmp($dev) {
  $version = "v1.0.2";

```

```
//   sed 's/<br \/>//g'

//   echo "<pre>";
//   _outln_comment("<pre>");
  _outln_comment("<style␣type=\"text/css\">␣x␣{font-family:courier;}␣</style>␣<x>");
  _outln_comment("qMp␣Guifi-oneclick␣".$version);
  _outln_comment("&nbsp;&nbsp;__&nbsp;_&nbsp;&nbsp;/\/\&nbsp;&nbsp;_&nbsp;__");
  _outln_comment("&nbsp;/&nbsp;_`&nbsp;|/&nbsp;&nbsp;&nbsp;&nbsp;\|&nbsp;'_&nbsp;\␣");
  _outln_comment("|&nbsp;(_|&nbsp;/&nbsp;/\/\&nbsp;\&nbsp;|_)&nbsp;|␣");
  _outln_comment("&nbsp;\__,&nbsp;\/&nbsp;&nbsp;&nbsp;&nbsp;\/&nbsp;.__/␣");
  _outln_comment("&nbsp;&nbsp;&nbsp;&nbsp;|_|&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;|_|␣");
  _outln_comment("&nbsp;quick␣MESH␣project␣</x>␣");
  _outln_comment("");
  _outln_comment("<b>Important:</b>␣You␣should␣have␣<b>'qmp-guifi'</b>␣package␣installed␣in␣your␣node.");
  _outln_comment("");
  _outln_comment("To␣apply␣this␣configuration␣in␣your␣node,␣you␣can␣follow␣this␣instructions:␣");
  _outln_comment("<a␣href='http://dev.qmp.cat/projects/qmp/wiki/Guifi_oneclick'␣target='_blank'>http://dev.qmp.cat/projects/qmp/wiki/Guifi_oneclick</a>␣");
  _outln_comment("");

  // ONLY THE FIRST MESH IP FOUND IS GET
  // TO DO: Check if there are more than one MESH radios and/or interfaces
  //
  $mesh="no";
  $ipv4="-";
  $netmask="-";
  $devmodel="-";

  foreach ($dev->radios as $radio) {
    if ($radio['mode'] == 'mesh') {
      $mesh="yes";

      $i=0;
      foreach ($radio['interfaces'] as $interface) {
        // If interface has any IP addresses we get the first one
        if (isset($interface['ipv4'])) {
          $ipv4 = $interface['ipv4'][0]['ipv4'];
          $netmask = $interface['ipv4'][0]['netmask'];
          $maskbits = $interface['ipv4'][0]['maskbits'];
        }
        else { $ipv4="-"; $netmask="-"; $maskbits="-"; }

       }
      break;
    }
  }

  // GET ZONE NICK (MAYBE ID?)
  //
  $node = node_load(array('nid' => $dev->nid));
  $zone = node_load(array('nid' => $node->zone_id));
  $zonename = $zone->nick;


  _outln();
  _outln("meshradio='".$mesh."'");

  if ($mesh == 'yes') {
    _outln("nodename='".$node->nick."'"); // This is the node name
    _outln("devname='".$dev->nick."'");   // This is the device name with mesh radio
    _outln("devmodel='".$dev->model."'");
    _outln("ipv4='".$ipv4."'");
    _outln("netmask='".$netmask."'");
    _outln("zoneid='".$zonename."'");
  }
  else {
    _outln();
    _outln_comment("␣<b>You␣don't␣have␣any␣Mesh␣radio!</b>");
    _outln_comment(" If you want to use Guifi-oneclick, make sure you configure it properly.");
    _outln_comment(" You can follow the instructions in the wiki: <a href='./unsolclic' target='_self'>EN (not yet)</a>, <a href='http://es.wiki.guifi.net/wiki/Mesh#Conectarse_a_una_red_Mesh' target='_blank'>ES</a>, <a href='http://ca.wiki.guifi.net/wiki/Mesh#Connectar-se_a_una_xarxa_Mesh' target='_blank'>CA</a>");
  }

//   var_dump($node->nid['zone_id']);

}

?>
```

B Setting qMp development environment

To be able to develop for qMp

B.1 Getting the needed software

In a Debian based distributions (like Ubuntu), a set of packages are needed to be able to build the system:

```
sudo aptitude install git subversion zlib1g-dev gawk flex unzip ↩
    bzip2 gettext build-essential libncurses5-dev libncursesw5-dev ↩
    binutils cpp psmisc docbook-to-man
```

For 64 bits machines (x86_64) 32 bit development files are needed.

In Debian/Ubuntu:

```
sudo aptitude install gcc-multilib
```

In CentOS/Fedora/RHEL the packets are:

```
gcc.i686, libgcc.i686, and glibc-devel.i686
```

B.2 Getting the code and compile

- [Recommended] Get the qMp firmware generator using git:

  ```
  git clone git://qmp.cat/qmpfw.git qmpfw
  ```

  [Outdated] Get the code using http:

  ```
  wget -c -q -O - ``http://qmp.cat/gitrevision_download?project_id=7&rev=anonymous" | ↩
      tar zxvf -
  ```

- Enter to source directory:

  ```
  cd qmpfw
  ```

- To compile a specific branch of qMp do a checkout specifying it:

  ```
  make .checkout_qmp QMP_GIT_BRANCH=branch_name

  Example for testing branch:
  make .checkout_qmp QMP_GIT_BRANCH=testing
  ```

- Compile it specifying the target:

  ```
  make build T=alix
  ```

You can find available targets by executing:

```
make list_targets
```

And if there are more than one core in your computer you can use J=N:

```
make build T=alix J=4
```

- After that, you will find the images ready to install in your devices inside directory:

  `qmpfw/images/`

C qMp package: 'qmp-guifi'

Here there are the files included in the 'qmp-guifi' package v1.0.2.[3]

1. *Makefile*
   Instructions to compile and package the files.

2. *files/etc/qmp/qmp_guifi.sh*
   Main script file with the functions get_url, check, print, configure, ...

3. *files/usr/bin/qmpguifi*
   Symbolic link to /etc/qmp/qmp_guifi.sh

4. *files/usr/lib/lua/luci/controller/guifi.lua*
   Adds the new Guifi menu in the LuCI web interface.

5. *files/usr/lib/lua/luci/view/qmp/guifi.htm*
   The Guifi oneclick web configuration page.

6. *files/www/cgi-bin/guifi*
   Script to interact between the web page and the main qmp-guifi.sh script

7. *files/www/luci-static/resources/qmp/guifi-logo.png*
   Guifi.net logotype.

1. Makefile

```
#    Copyright (C) 2013 Quick Mesh Project
#
#    This program is free software; you can redistribute it and/or modify
#    it under the terms of the GNU General Public License as published by
#    the Free Software Foundation; either version 2 of the License, or
#    (at your option) any later version.
#
#    This program is distributed in the hope that it will be useful,
#    but WITHOUT ANY WARRANTY; without even the implied warranty of
#    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
#    GNU General Public License for more details.
#
#    You should have received a copy of the GNU General Public License along
#    with this program; if not, write to the Free Software Foundation, Inc.,
#    51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA.
#
#    The full GNU General Public License is included in this distribution in
#    the file called "COPYING".
#
# Contributors:
# Jorge L. Florit
#

include $(TOPDIR)/rules.mk
include $(INCLUDE_DIR)/kernel.mk

PKG_NAME:=qmp-guifi
PKG_RELEASE:=1.0.2
PKG_BUILD_DIR:=$(BUILD_DIR)/$(PKG_NAME)

include $(INCLUDE_DIR)/package.mk

```

[3]qmp-guifi repository: http://dev.qmp.cat/projects/qmp/repository/revisions/testing/show/packages/qmp-guifi

```
define Package/qmp-guifi
  TITLE:=Guifi Oneclick for qMp.
  SECTION:=net
  CATEGORY:=qMp
  URL:=http://guifi.net
  DEPENDS:= \
  +qmp-system
endef


define Package/qmp-guifi/description
 Guifi Oneclick for qMp. Easy configuration for Guifi.net community network Mesh nodes.
endef

define Build/Prepare
 mkdir -p $(PKG_BUILD_DIR)
endef

define Build/Configure
endef

define Build/Compile
endef

define Package/qmp-guifi/install
  $(CP) ./files/* $(1)/
endef

$(eval $(call BuildPackage,qmp-guifi))
```

2. files/etc/qmp/qmp_guifi.sh

```
#!/bin/sh
#     Copyright (C) 2013 Quick Mesh Project
#
#     This program is free software; you can redistribute it and/or modify
#     it under the terms of the GNU General Public License as published by
#     the Free Software Foundation; either version 2 of the License, or
#     (at your option) any later version.
#
#     This program is distributed in the hope that it will be useful,
#     but WITHOUT ANY WARRANTY; without even the implied warranty of
#     MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
#     GNU General Public License for more details.
#
#     You should have received a copy of the GNU General Public License along
#     with this program; if not, write to the Free Software Foundation, Inc.,
#     51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA.
#
#     The full GNU General Public License is included in this distribution in
#     the file called "COPYING".

QMP_PATH="/etc/qmp"

[ -z "$SOURCE_COMMON" ] && . $QMP_PATH/qmp_common.sh
#[ -z "$SOURCE_FUNCTIONS" ] && . $QMP_PATH/qmp_functions.sh

QMP_VERSION="$QMP_PATH/qmp.version"

[ -z $ONECLICK_CGI ] && ONECLICK_CGI=0
ONECLICK_FILE="/tmp/guifi_oneclick"
ONECLICK_PATTERN="qMp_Guifi-oneclick"
ONECLICK_URL="/view/unsolclic"
ONECLICK_URL_BASE="http://guifi.net/guifi/device/"
ONECLICK_VARS="nodename devname devmodel ipv4 netmask zoneid"
#ONECLICK_ZONES="GS='124+' RAV='136' VLLC='108+'"


get_url() {
  echo "Getting␣oneclick␣config:"

  [ -z $1 ] && {
    [ $ONECLICK_CGI -eq 1 ] && { echo "ERROR:␣No␣URL␣specified."; exit 1; } ||
    qmp_error "No␣URL␣specified.␣USE:␣'$0␣get_url␣[${ONECLICK_URL_BASE}#####/]␣[FILE]'"
  } || {
    echo $1 | grep -q "^http://" 2>/dev/null
    ## TO DO CHECK IF ID OR URL IS GIVEN
    [ $? -ne 0 ] && {
      [ $ONECLICK_CGI -eq 1 ] && { echo "ERROR:␣Wrong␣URL␣specified."; exit 1; } ||
      qmp_error "Wrong␣URL␣specified.␣USE:␣'$0␣get_url␣[${ONECLICK_URL_BASE}#####/]␣[FILE]'"
    }
  }
  [ -z $2 ] && {
    [ $ONECLICK_CGI -eq 1 ] && { echo "ERROR:␣No␣temporary␣file␣specified."; exit 1; } ||
    qmp_error "No␣temporary␣file␣specified.␣USE:␣'$0␣get_url␣[${ONECLICK_URL_BASE}#####/]␣↩
        [FILE]'"
  }

  # CHECK IF UNSOLCLIC OR DEVICE URL GIVEN
  local oneclick_url
  echo $1 | grep -q "/view/unsolclic$" 2>/dev/null
  [ $? -ne 0 ] && oneclick_url=$1$ONECLICK_URL || oneclick_url=$1

  wget -q $oneclick_url -O $2 2>/dev/null
  [ $? -ne 0 ] && rm -f $file && {
    [ $ONECLICK_CGI -eq 1 ] && { echo "ERROR:␣Error␣downloading␣$oneclick_url"; exit 1; } ||
    qmp_error "Error␣downloading␣$oneclick_url"
  }

  # REMOVING "<br />" (to change output format edit guifi.net drupal module unsolclic file ↩
      guifi/firmware/qmp.inc.php)
  sed -i 's/^<br \/>//g' $2

  echo "Done!"
  return 0
}

check() {
  echo "Checking␣oneclick␣config:"

  [ -z $1 ] && {
    [ $ONECLICK_CGI -eq 1 ] && { echo "ERROR:␣No␣file␣given."; exit 1; } ||
    qmp_error "No␣file␣given.␣USE:␣$0␣check␣[FILE]␣"
  }
  [ ! -f $1 ] && {
    [ $ONECLICK_CGI -eq 1 ] && { echo "ERROR:␣File␣$1␣not␣found."; exit 1; } ||
    qmp_error "File␣$1␣not␣found."
  }

  # CHECK IF VALID UNSOLCLIC CONFIG
  grep -q "$ONECLICK_PATTERN" $1 2>/dev/null
  [ $? -ne 0 ] && {
    [ $ONECLICK_CGI -eq 1 ] && { echo "ERROR:␣Not␣valid␣unsolclic␣file.␣Check␣file␣or␣↩
        URL."; exit 1; } ||
    qmp_error "Not␣valid␣unsolclic␣file.␣Check␣file␣or␣URL."
  }

  # CHECK IF HAS MESH RADIO
  local meshradio=`grep "meshradio" $1 | awk '{FS="="; print $2}' | tr -d "'"`
  [ "$meshradio" == "no" ] && {
    [ $ONECLICK_CGI -eq 1 ] && { echo "ERROR:␣No␣Mesh␣radio␣found.␣Revise␣your␣device␣↩
        configuration␣in␣the␣guifi.net␣website."; exit 1; } ||
    qmp_error "No␣Mesh␣radio␣found.␣Revise␣your␣device␣configuration␣in␣the␣guifi.net␣↩
        website."
  }

  echo "Done!"
  return 0
}
```

```

print() {
  echo "Showing␣variables:"
  [ -z $1 ] && {
    [ $ONECLICK_CGI -eq 1 ] && { echo "ERROR:␣No␣file␣given"; exit 1; } ||
    qmp_error "No␣file␣given.␣USE:␣'$0␣print␣[FILE]'"
  }
  [ ! -f $1 ] && {
    [ $ONECLICK_CGI -eq 1 ] && { echo "ERROR:␣File␣$1␣not␣found."; exit 1; } ||
    qmp_error "File␣$1␣not␣found."

  }
  local var
  for var in $ONECLICK_VARS; do
    echo "␣$var=`grep␣"$var"␣$1␣|␣awk␣'{FS="=";␣print␣$2}'␣|␣tr␣-d␣"'"␣|␣sed␣'s/\␣/_/g'`"
  done

  return 0
}

configure() {
  echo "Configuring␣the␣node,␣please␣wait..."

  [ -z $1 ] && {
    [ $ONECLICK_CGI -eq 1 ] && { echo "ERROR:␣No␣file␣given"; exit 1; } ||
    qmp_error "No␣file␣given:␣USE:␣'$0␣configure␣[FILE]'"
  }

  # SET COMMUNITY MODE, DHCP AND PUBLISH LAN
  uci set qmp.roaming.ignore=1
  uci set qmp.networks.publish_lan=1
  uci set qmp.networks.disable_lan_dhcp=0

  # SET LAN IP
  local ip="`grep␣"ip"␣$1␣|␣awk␣'{FS="=";␣print␣$2}'␣|␣tr␣-d␣"'"␣|␣sed␣'s/\␣/_/g'`"
  uci set qmp.networks.lan_address="$ip"

  # SET LAN MASK
  local mask="`grep␣"mask"␣$1␣|␣awk␣'{FS="=";␣print␣$2}'␣|␣tr␣-d␣"'"␣|␣sed␣'s/\␣/_/g'`"
  uci set qmp.networks.lan_netmask="$mask"

  # SET BMX IP mask (CIDR)
  local cidrmask=0
  IFS="."
  for dec in $mask; do
    while [ $dec -gt 0 ]; do
      cidrmask=$(($cidrmask+$dec%2))
      dec=$(($dec/2))
    done
  done
  IFS="\␣"
  uci set qmp.networks.bmx6_ipv4_address="$ip/$cidrmask"

  # GET NODE DEVICE NAME - ZONE ID - ZONE CHANNEL
  local zone="`grep␣"zone"␣$1␣|␣awk␣'{FS="=";␣print␣$2}'␣|␣tr␣-d␣"'"␣|␣sed␣'s/\␣/_/g'`"
  local nodename="`grep␣"nodename"␣$1␣|␣awk␣'{FS="=";␣print␣$2}'␣|␣tr␣-d␣"'"␣|␣sed␣'s/\␣↩
      /_/g'`"
  local devname="`grep␣"devname"␣$1␣|␣awk␣'{FS="=";␣print␣$2}'␣|␣tr␣-d␣"'"␣|␣sed␣'s/\␣/_/g'`"
  local ssid="`grep␣"ssid"␣$1␣|␣awk␣'{FS="=";␣print␣$2}'␣|␣tr␣-d␣"'"␣|␣sed␣'s/\␣/_/g'`"

  # SET NODE NAME
  # TO DO: SET ZONE ID IN NAME (OR NOT)
  uci set qmp.node.community_id="$devname-"

        # Select radio mode to set SSID
  local j=0
  local mode ssid
  while qmp_uci_test qmp.@wireless[$j]; do
    mode=$(uci get qmp.@wireless[$j].mode)
    echo $mode | grep -q "adhoc" 2>/dev/null
    [ $? -eq 0 ] && ssid="guifi.net/${nodename}"
    [ "$mode" == "ap" ] && ssid="qMp-AP"
    uci set qmp.@wireless[$j].name=$ssid
    j=$(( $j + 1 ))
  done

  # Set filter to update with image that includes 'qmp-guifi' package
  uci set qmp.update.filter="Guifi.*sysupgrade"

  echo;
  uci commit
  sleep 1
  qmpcontrol configure_network ; qmpcontrol configure_wifi # ; /etc/init.d/bmx6 restart
  return 0
}

oneclick() {
  [ ! -z $1 ] && oneclick_url=$1 || exit 1
  [ ! -z $2 ] && oneclick_file=$2 || oneclick_file=$ONECLICK_FILE

  # GETTING ONECLICK CONFIG
  get_url $oneclick_url $oneclick_file
  [ $? -ne 0 ] && qmp_error "Unexpected error in qmpguifi get_url function"
  echo;

  # CHECKING DOWNLOADED CONFIG
  check $oneclick_file
  [ $? -ne 0 ] && qmp_error "Unexpected error in qmpguifi check function"
  echo;

  # PRINTING CONFIG VARIABLES
  print $oneclick_file
  [ $? -ne 0 ] && qmp_error "Unexpected error in qmpguifi print function"
  echo;

  # CONFIGURING QMP SYSTEM
  read -p "Do you want to configure your node with this settings? [N,y]" a
  echo;
  [ "$a" == "y" ] && {
    configure $oneclick_file
    [ $? -ne 0 ] && qmp_error "Unexpected in qmpguifi configure function"
    echo "Configuration done!"; echo;
    rm -f $oneclick_file
    return 0
  } || {
    echo "Doing nothing."; echo;
    rm -f $oneclick_file
    return 1
  }
}

help() {
  echo "Use: $0 <function> [params]"
  echo ""
  echo "get_url [URL] [FILE]  : Get oneclick file."
  echo "check [FILE]          : Check if valid onelick file."
  echo "print [FILE]          : Print oneclick file values."
  echo "configure [FILE]      : Configure node with oneclick file values (recommended to check file before)."
  echo "-"
  echo "oneclick [URL]        : Do all configuration based on Guifi.net website data."
  echo ""
  exit 1;
}

[ -z "$1" ] && help
$@
```

3. files/usr/bin/qmpguifi

```
lrwxrwxrwx 1 jlflorit jlflorit 21 Jan  9 17:03 qmpguifi -> /etc/qmp/qmp_guifi.sh
```

4. files/usr/lib/lua/luci/controller/guifi.lua

```
--[[
    Copyright (C) 2013 Quick Mesh Project
    Contributors:
       Jorge L. Florit

    This program is free software; you can redistribute it and/or modify
    it under the terms of the GNU General Public License as published by
    the Free Software Foundation; either version 2 of the License, or
    (at your option) any later version.

    This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU General Public License for more details.

    You should have received a copy of the GNU General Public License along
    with this program; if not, write to the Free Software Foundation, Inc.,
    51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA.

    The full GNU General Public License is included in this distribution in
    the file called "COPYING".
--]]

module("luci.controller.guifi", package.seeall)

function index()
  -- setting place from pre-defined value
  local place = {"qmp","Guifi"}

  -- setting position of menu
  local position = "6"

  -- Guifi oneclick menu entry
  entry(place,call("action_guifi"),place[#place],tonumber(position))
  table.remove(place)
end

function action_guifi()
  package.path = package.path .. ";/etc/qmp/?.lua"
  local qmp = require "qmpinfo"
  local key = qmp.get_key()
  luci.template.render("qmp/guifi",{key=key})
end
```

5. files/usr/lib/lua/luci/view/qmp/guifi.htm

```
<%+header%>

<script type="text/javascript">

function option(url,key,option) {
  var output = document.getElementById("output_field");
  if (url) if (output) {
    toggle("confirm",0);    // Hide button
    toggle("cancel",0);     // Hide button
    output.innerHTML ='<%:Collecting data...%>&lt;img ↩
        src="/luci-static/resources/icons/loading.gif" height="20px" alt="Loading" ↩
        style="vertical-align:midle;"/&gt;  ';
    output.style.display = 'block';
    XHR.get("/cgi-bin/guifi",[ key , option , url ] ,
    function(x) {
      var text = new String(x.responseText);
      output.innerHTML="&lt;pre&gt;&lt;font color='#0a0'&gt;"+text+"&lt;/font&gt;&lt;/pre&gt;";
      document.getElementById("guifidiv").style.height = 'auto';
      if ( !(text.indexOf("ERROR") >= 0) ) {
        switch (option) {
        case "apply":
          window.setTimeout('this.toggle("confirm",1)',150);   // Unhide button
          window.setTimeout('this.toggle("cancel",1)',200);    // Unhide button
          break;
        case "cancel":
          output.innerHTML="";
          break;
        case "confirm":
          window.setTimeout('window.location="/luci-static/resources/qmp/wait_long.html"',1000);
          break;
        }
      }
    } )
  }
}
function toggle(id,state){
  var element = document.getElementById(id);
  if (state) element.style.visibility="visible"
  else element.style.visibility="hidden";
  return 0;
}

</script>

<h2>Quick Mesh Project</h2>

<div class="cbi-map">
<form>
<fieldset class="cbi-section">
<legend><%:Configuration for a guifi.net Mesh network in one (or two) click(s)%></legend>
<img src="/luci-static/resources/qmp/guifi-logo.png" alt="Guifi.net"/>

<br /><br />
  <div id="guifidiv" style="clear:both; height:75px; ">
  <strong>Specify the guifi.net URL of the device:</strong>
  <input type="text" id="guifi" name="guifi" />
  <input type="button" id="apply" value="Apply" onclick="option(this.form.guifi.value,'<%=key%>','apply')" style="padding:0 5px;" />
  <span id="output_field" style="display:block;text-align:left;padding:10px;"></span>
  <input type="button" id="confirm" value="Confirm" onclick="option(this.form.guifi.value,'<%=key%>','confirm')" style="visibility:hidden;margin:10px;padding:0 5px;" />
  <input type="button" id="cancel" value="Cancel" onclick="option(this.form.guifi.value,'<%=key%>','cancel')" style="visibility:hidden;" />
  </div>
</fieldset>
</form>
</div>

<%+footer%>
```

6. files/www/cgi-bin/guifi

```
#!/bin/sh

GUIFI_TEMP=/tmp/guifi_oneclick

echo "content-type: text/plain"
echo ""
echo "$QUERY_STRING" >> /tmp/debug
QUERY_KEY="$(echo $QUERY_STRING | cut -d'&' -f1 | cut -d'=' -f2)"
QUERY_TYPE="$(echo $QUERY_STRING | cut -d'&' -f2 | cut -d'=' -f2)"
```

```
QUERY_DATA="$(echo $QUERY_STRING | cut -d'&' -f3 | cut -d'=' -f2 | sed s/'%3A'/':'/g | sed s/'%2F'/'\/'/g)"

echo "$QUERY_TYPE $QUERY_DATA" >> /tmp/debug

KEY_F="$(uci get qmp.node.key)"
[ -z "$KEY_F" ] && KEY_F="/tmp/qmp_key"
KEY="$(cat $KEY_F)"
[ "$KEY" != "$QUERY_KEY" ] && { echo "Invalid key"; exit 1; }


guifi_apply() {
  [ ! -z "$QUERY_DATA" ] && {
    export ONECLICK_CGI=1
    /etc/qmp/qmp_guifi.sh get_url $QUERY_DATA $GUIFI_TEMP
    [ $? -ne 0 ] && exit 1 || echo;
    /etc/qmp/qmp_guifi.sh check $GUIFI_TEMP
    [ $? -ne 0 ] && exit 1 || echo;
    /etc/qmp/qmp_guifi.sh print $GUIFI_TEMP
    [ $? -ne 0 ] && exit 1
    export ONECLICK_CGI=0
  }
}

guifi_cancel() {
  [ -f $GUIFI_TEMP ] && rm -r $GUIFI_TEMP
  export ONECLICK_CGI=0
}

guifi_confirm() {
  [ ! -z "$QUERY_DATA" ] && {
    # /etc/qmp/qmp_guifi.sh get_url $QUERY_DATA $GUIFI_TEMP
    # echo;
    export ONECLICK_CGI=1
    /etc/qmp/qmp_guifi.sh check $GUIFI_TEMP
    [ $? -ne 0 ] && exit 1 || echo;
    /etc/qmp/qmp_guifi.sh configure $GUIFI_TEMP
    [ -f $GUIFI_TEMP ] && rm -r $GUIFI_TEMP
    export ONECLICK_CGI=0
  }
}

[ "$QUERY_TYPE" == "apply" ] && guifi_apply
[ "$QUERY_TYPE" == "cancel" ] && guifi_cancel
[ "$QUERY_TYPE" == "confirm" ] && guifi_confirm

exit 0
```

7. files/www/luci-static/resources/qmp/guifi-logo.png

guifi·net

Acronyms

AP	Access Point
API	Application Programming Interface
BMX6	BATMAN eXperimental 6
BATMAN	Better Approach to Mobile Adhoc Networks
BSSID	Basic Service Set IDentifier
CIDR	Classless Inter-Domain Routing
CGI	Common Gateway Interface
CLI	Command Line Interface
CMS	Content Management System
CONFINE	Community Networks Testbed for the Future Internet
DHCP	Dynamic Host Configuration Protocol
GSF	Gràcia Sense Fils
GSoC	Google Summer of Code
HTML	Hypertext Markup Language
IETF	Internet Engineering Task Force
IP	Internet Protocol
IXP	Internet Exchange Point
JS	JavaScript
LAN	Local Area Network
MANET	Mobile Ad-hoc NETworks
OSI	Open Systems Interconnection
OLSR	Optimized Link State Routing
P9FS	Poblenou Sense Fils
PHP	PHP Hypertext Preprocessor
QMP	Quick Mesh Project
RFC	Request For Comments
SCM	Source Code Management
SNMP	Simple Network Management Protocol
SSID	Service Set IDentifier
UCI	Unified Configuration Interface
UPC	Universitat Politècnica de Catalunya
URL	Uniform Resource Locators

Bibliography

[1] Commons for the Open Free and Neutral Network. http://guifi.net/en/CommonsXOLN.

[2] Guifi.net Drupal module documentation (ES). http://es.wiki.guifi.net/wiki/Módulo_Drupal_Guifi.net.

[3] Guifi.net IPv4 allocation criteria (CAT). http://ca.wiki.guifi.net/wiki/Criteris_d'assignació_d'adreces_IPv4.

[4] Guifi.net name conventions (CAT). http://ca.wiki.guifi.net/wiki/Convencions_de_noms.

[5] How to create OpenWRT packages. http://wiki.openwrt.org/doc/devel/packages.

[6] How to set Guifi development environment (ES). http://es.wiki.guifi.net/wiki/Preparando_el_entorno_de_desarrollo.

[7] LuCI web interface for embedded devices. http://luci.subsignal.org/trac/wiki/Documentation.

[8] MANET documentation and RFCs. http://datatracker.ietf.org/wg/manet/.

[9] Oneclick technical documentation (CAT). http://ca.wiki.guifi.net/wiki/Unsolclic.

[10] Open Systems Interconnection. http://docwiki.cisco.com/wiki/Open_System_Interconnection_Protocols.

[11] OpenWRT OPKG package Manager. http://wiki.openwrt.org/doc/techref/opkg.

[12] Optimizing or visual acuity (ES). http://ocularis.es/blog/?p=483.

[13] Presentation of Mesh networks and GSFfirm. http://guifisants.net/files/manet/presentacio_manet.pdf.

[14] qMp config file. http://dev.qmp.cat/projects/qmp/wiki/Config.

[15] qMp upgrade system. http://dev.qmp.cat/projects/qmp/wiki/Upgrade_system.

[16] Routing packets into Wireless Mesh Networks. http://www.baumann.info/public/wimob07.pdf.

[17] Understanding the Network Terms SSID, BSSID, and ESSID. http://www.juniper.net/techpubs/en_US/junos-space-apps12.3/network-director/topics/concept/wireless-ssid-bssid-essid.html.

[18] Unified Configuration Interface. http://wiki.openwrt.org/doc/techref/uci.

[19] Visual fatigue: computer vision syndrome (ES). http://ocularis.es/blog/?p=75.

[20] Wireless mesh topology. http://www.moskaluk.com/Mesh/wireless_mesh_topology.htm.

[21] Roger Baig. Evaluation of Dynamic Routing Protocols on Realistic Wireless Topologies. http://bmx6.net/documents/20.

[22] Pau Escrich. Quick Mesh Project thesis degree. http://dev.qmp.cat/documents/11.

[23] Axel Neumann. BatMan eXperimental 6. https://github.com/axn/bmx6/blob/master/README.md.

[24] Charles E Perkins et al. *"Ad hoc networking"*, volume 1. Addison-wesley Reading, 2001.