L.I.S.A Documentation

Version 0.1

Julien Syx

15 October 2014

Table des matières

CHAPITRE 1

Introduction à L.I.S.A

1.1 Le résumé en 30 secondes

L.I.S.A c'est :

— Un système intelligent afin de vous aider dans votre vie de tout les jours.
— Un framework qui interconnecte les objets entre-eux et expose l'API HTTP

Elle à été développée en ayant une interface similaire à "J.A.R.V.I.S" (Iron Man).

Le nom de L.I.S.A est tiré de "Weird Science" une série télévisée. Deux intellos essaie de créer la femme parfaite à l'aide de leur ordinateur et une nuit elle vient à la vie. En France, cette série télévisée est appelée "Code LISA".

C'est pour cela que ce projet porte ce nom.

1.2 S'appuyant sur une technologie approuvée.

L.I.S.A profite d'un certains nombres de technologies et techniques. La couche réseau est construite avec l'excellente bibliothèque réseau 'Twisted'.

L.I.S.A utilise des clés publiques lors de l'authentification de l'identité des clients autorisés à se connecter.

L'interface web est construit au-dessus de 'Django' qui est un framework pour python.

1.3 Communauté de L.I.S.A

Rejoigniez L.I.S.A !

Il existe une multitude de possibilités de participer au projet et communiquer avec la communauté de L.I.S.A.

L.I.S.A dispose d'un canal IRC actif et d'un forum.

1.4 Forum

Rejoindre le 'forum L.I.S.A'_. C'est le meilleur endroit pour poser des questions concernant L.I.S.A et voir le développement de L.I.S.A ! C'est ouvert aux nouveaux membres.

1.5 IRC

Le canal IRC "#lisa-project" est hébergé par le populaire réseau 'Freenode'. Vous pouvez voir le 'client du tchat Freenode' à droite de votre navigateur.

1.6 Suivre sur Github

Le code de L.I.S.A est développé via Github. Suivre L.I.S.A pour les nombreuses mises à jours et voir le développement de L.I.S.A :

— 'Serveur L.I.S.A. <http://www.giyhub.com/Seraf/L.I.S.A>'

Vous pouvez suivre la feuille de route sur Waffle :

— 'Liste des tâches de L.I.S.A. <https://waffle.io/seraf/lisa>'

Et vérifiez le statut de construction sur Travis :

— 'Construction de L.I.S.A. <https://travis-ci.org/Seraf/LISA>'

1.7 Liens des autres communautés

— 'Projet L.I.S.A. <http://www.lisa-project.net>'

— 'Google+ <https://plus.google.com/u/0/communities/109665724480163795371>'

1.8 Bidouiller le code

Si vous voulez vous impliquer dans le développement du code source ou dans la documentation de L.I.S.A, veuillez regarder : :doc :'la section bidouillage <development/hacking>' !

CHAPITRE 2

A propos de L.I.S.A et son auteur

CHAPITRE 3

Installation

Voir aussi :

Contribuez à L.I.S.A

3.1 Installation rapide

```
# Installing Debian dependencies
sudo apt-get install mongodb python-setuptools libxslt1-dev libxslt1.1 libxml2-dev libffi-dev build-e
# Installing pip
sudo easy_install pip
# Installing tools to create a python virtual environment
sudo pip install virtualenv virtualenvwrapper
# Create the lisa user (you can choose another username)
sudo useradd -s /bin/bash -m alivelisa
# Login with this user
sudo su - alivelisa
# Set environment variable to use correctly the new virtual environment
export WORKON_HOME=$HOME/.virtualenvs
echo "export WORKON_HOME=$HOME/.virtualenvs" >> ~/.bashrc
export PROJECT_HOME=$HOME/
echo "export PROJECT_HOME=$HOME/" >> ~/.bashrc
source /usr/local/bin/virtualenvwrapper.sh
echo "source /usr/local/bin/virtualenvwrapper.sh" >> ~/.bashrc
# Create the virtual environment named "lisa"
mkproject lisa
# This command let you enter your environment and load all libraries. Now Lisa is usable
workon lisa
# Install twisted then the LISA server
pip install twisted
pip install lisa-server
# Create your user
lisa-cli createsuperuser
```

3.2 Instructions d'installation pour plateformes spécifiques

Ces guides detaillent l'installation de L.I.S.A sur une plateforme donnée.

3.2.1 Installation sur Arch Linux

3.2.2 Installation sur Debian Linux

3.2.3 Installation sur Fedora Linux

3.2.4 Installation sur FreeBSD

3.2.5 Installation sur Gentoo Linux

3.2.6 Installation sur Mac OSX

3.2.7 Installation sur RedHat Linux

3.2.8 Installation sur Solaris

3.2.9 Installation sur un NAS Synology

3.2.10 Installation sur Windows

3.2.11 Installation sur Suse Linux

3.3 Dépendances

L.I.S.A devrait fonctionner sur n'importe quelle plateforme du type Unix du moment que les dépendances sont respectés.

```
mongoengine >= 0.8.7
Django >= 1.6.2
Sphinx >= 1.2.2
Twisted >= 13.2.0
autobahn >= 0.8.7
pymongo >= 2.7
requests >= 2.2.1
django-tastypie >= 0.11.0
django-tastypie-mongoengine >= 0.4.5
django-tastypie-swagger >= 0.1.2
pytz >= 2014.2
pyOpenSSL == 0.13
lisa-plugin-ChatterBot
PyWit >= 0.2.0
```

3.4 Mise à niveau de L.I.S.A

Pour manuellement mettre à niveau L.I.S.A, vous pouvez mettre à jour le paquet python (dans le virtualenv) :

```
pip install --upgrade lisa-server
```

CHAPITRE 4

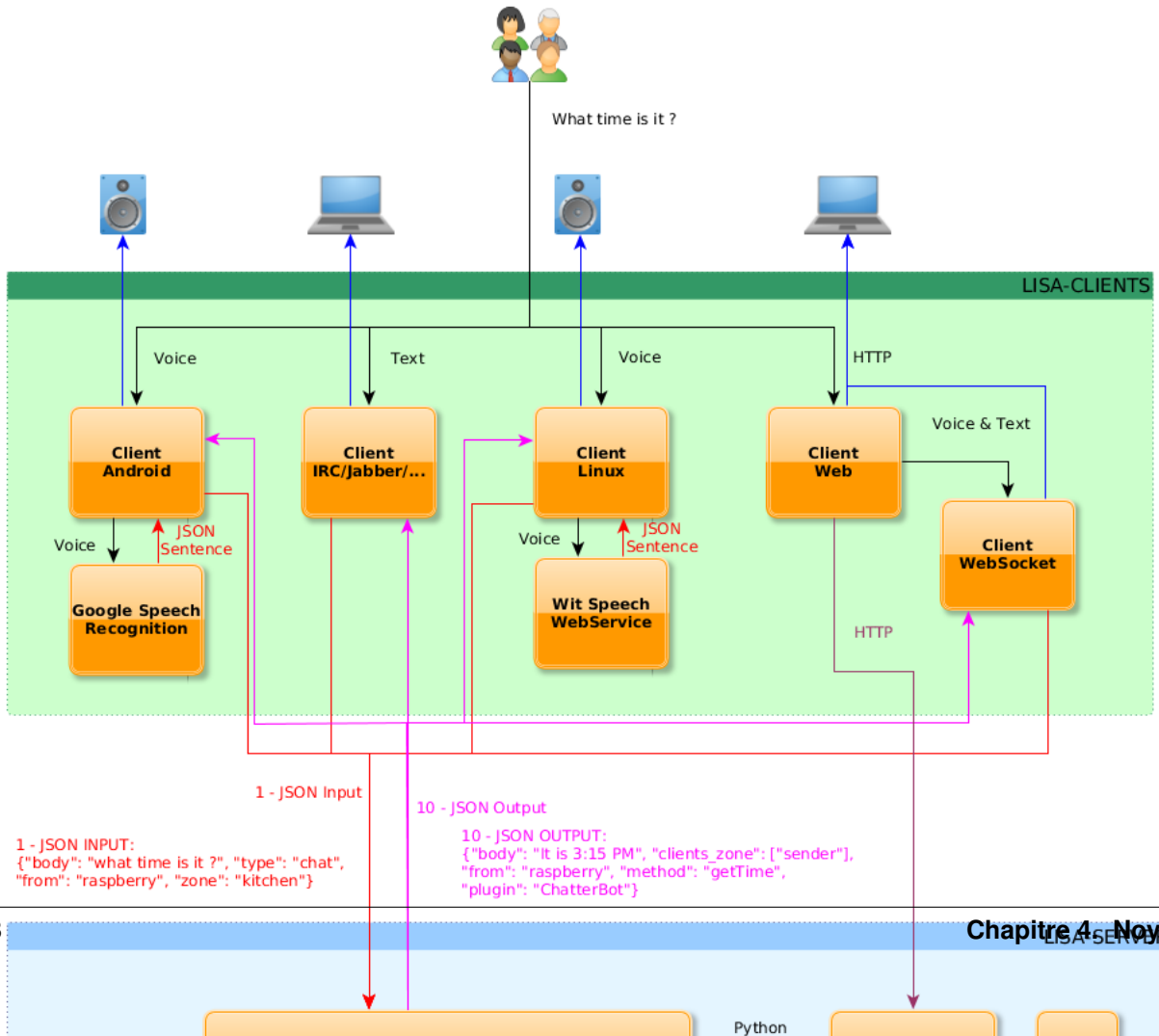
Noyau

4.1 Moteur

4.2 Gestionnaire de plugins

4.3 Serveur Web

4.4 Workflow



CHAPITRE 5

Plugins

5.1 Créer un plugin pour L.I.S.A

Créer un plugin est très facile, merci à l'outil cli.

Pour créer votre plugin, type (dans votre environnement virtuel) :

```
lisa-cli plugin PLUGINNAME --create
```

Ensuite, répondez aux questions, et vous trouverez le plugin dans votre environnement virtuel.

```
/home/alivelisa/.virtualenvs/lisa/local/lib/python2.7/site-packages/lisa/plugins/PLUGINNAME
```

Pour créer un nouveau plugin vous ne devez pas l'installer c'est automatique. Ecrivez juste le code que vous voulez dans le plugin, puis l'installer afin de l'utiliser. N'oubliez pas de l'installer en mode développeur (en utilisant –dev)

Voir aussi :

:doc :'Distribuez votre plugin </topics/plugins/distribute>'

5.2 Distribuez votre plugin

5.2.1 Paquet Python

Une fois votre plugin créé, vous peut-être aimeriez-vous le partager.

LISA utilise le paquet index de python pour gérer la dépendance des modules complémentaires et l'installation/désinstallation. Avec ce système, les modules complémentaires peuvent être utilisés comme une bibliothèque dans un autre module complémentaire tant que c'est un paquet de python.

L' *option création* ne crée pas l'architecture pour le deployer.

Pour créer une architecture d'installation, vous pouvez regarder comment les plugins sont actuellement fait : Chatter-Bot Plugin

5.2.2 Ajoutez le au Plugin Store

Pas encore disponible. Veuillez me contacter afin de l'ajouter à la liste des plugins.

5.3 Gestion des Plugins

La gestion de plugins est utile afin de gérer rapidement les plugins à partir de l'"interface en ligne de commande. Cela utilise la même fonction que l'interface web est à donc le même comportement.

Lister les fonctionnalités

```
$ lisa-cli plugin --help
Usage: /home/alivelisa/.virtualenvs/lisa/bin/lisa-cli plugin [options] <plugin_name>

Manage the plugins

Options:
  --list                List all plugins and show their status
  --create              Create a template plugin with a given name
  --enable              Enable a plugin
  --disable             Disable a plugin
  --install             Install a plugin
  --dev                 Dev mode
  --uninstall           Uninstall a plugin
  --upgrade             Upgrade a plugin
  -h, --help            show this help message and exit
```

list Cette option n'a pas besoin d'être appellé par un nom de plugin.

```
$ lisa-cli plugin --list
ChatterBot => [Installed] [Enabled]
BBox => [Installed] [Enabled]
ProgrammeTV => [Installed] [Enabled]
Cinema => [Not installed] [Not enabled]
Meteo => [Not installed] [Not enabled]
SNCF => [Not installed] [Not enabled]
Shopping => [Not installed] [Not enabled]
Wifiledlamps => [Not installed] [Not enabled]
```

create Cette option va créer un plugin dans votre répertoire des plugins. Quelques questions vous seront demandés afin de configurer automatiquement le plugin.

```
$ lisa-cli plugin --create PLUGINNAME
What is your full name ? : Lisa
What is your email ? : lisa@lisa-project.net
[OK] Plugin created
```

enable / disable Cette option va activé ou désactivé un plugin. Cela signifie que le plugin est toujours présent dans la base de donnée and sur le système de fichier mais ne sera pas chargé par le gestionnaire de plugin.

```
$ lisa-cli plugin --enable PLUGINNAME
[OK] Plugin enabled

$ lisa-cli plugin --disable PLUGINNAME
[OK] Plugin disabled
```

dev Cette option vous permet de spécifier ce que vous écrivez dans l'extension. Cela ne le télécharge pas dans Python Package Index mais utilise plutôt l'extension local. Elle ne fait rien dans les fichiers système mais essentiellement dans la base de données. Utilisée principalement pour installer et désinstaller une fonction. Quand elle est utilisée pour l'installation, elle actualise tout les champs dans la base de données selon vos fichiers d'extension local. Si vous l'utilisez afin de désinstaller avec le mode développeur, elle ne désinstalle pas le fichier json, mais seulement les enregistrements de votre base de données.

install Cette option installe l'extension. Par défaut elle télécharge le paquet depuis Python Package Index puis lis le fichier json et installe tout les composants (règles, crons, intension, configuration de l'extension) dans la base de données.

```
$ lisa-cli plugin --install PLUGINNAME
[OK] Plugin installed
```

uninstall Cette option désinstalle l'extension. Par défaut cela supprime le paquet et tout les fichiers relatif à l'extension et supprime les entrées relatif à l'extension dans la base de données.

```
$ lisa-cli plugin --uninstall PLUGINNAME
[OK] Plugin uninstalled
```

upgrade Cette option n'est pas encore implémentée.

```
$ lisa-cli plugin --upgrade PLUGINNAME
[OK] Plugin upgraded
```

5.4 Structure d'un Plugin

CHAPITRE 6

Comment Wit fonctionne ?

CHAPITRE 7

Tutoriels

CHAPITRE 8

Résolution de problèmes

L'objectif de la section résolution de problèmes est de présenter les solutions aux problèmes fréquemment rencontrés pour les utilisateurs ainsi que les outils disponible pour aider le debug du code.

8.1 Résolution de problèmes sur le Serveur

Si votre serveur rencontre des problèmes tels que le non retournement de données, faible rapidité d'exécution, ou autres : :doc :'Résolution des problèmes serveurs </topics/troubleshooting/server>'

8.1.1 Exécution au premier plan

Un grand nombre d'informations sont disponibles via le journal d'activités du débogage, si vous rencontrez des problèmes, il suffit de démarrer le serveur au premier plan afin d'afficher les activités dans la console :

```
twistd -n lisa-server
```

8.2 Quels ports doivent être ouverts sur le Serveur et les Clients ?

Aucun port ne doit être ouvert du coté des clients. Pour la partie serveur, les ports TCP 10042, 10043 et 8000 (ou 80) doivent être ouverts. Si vous ne pouvez accédez à votre serveur ainsi qu'à l'interface web, cela peut être un problème de parefeu.

Vous pouvez vérifier que le port est bien ouvert avec la commande nc :

```
nc -v -z lisa.server 10042
nc -v -z lisa.server 8000
```

CHAPITRE 9

En developpement

9.1 Contribuer

There is a great need for contributions to L.I.S.A and patches are welcome ! The goal here is to make contributions clear, make sure there is a trail for where the code has come from, and most importantly, to give credit where credit is due !

Il y a plusieurs façons de contribuer au développement de L.I.S.A.

9.1.1 Envoyer un sondage sur GitHub

This is the preferred method for contributions. Simply create a GitHub fork, commit changes to the fork, and then open up a pull request.

The following is an example (from Open Comparison Contributing Docs) of an efficient workflow for forking, cloning, branching, committing, and sending a pull request for a GitHub repository.

First, make a local clone of your GitHub fork of the L.I.S.A GitHub repo and make edits and changes locally.

Ensuite, créer une nouvelle branche sur votre clone en entrant les commandes suivantes :

```
git checkout -b fixed-broken-thing
```

```
Switched to a new branch 'fixed-broken-thing'
```

Choisissez un nom à votre branche qui décrit son but.

Maintenant commentez vos changements sur cette branche avec la commande suivante :

```
git commit -am 'description of my fixes for the broken thing'
```

Note : Using `git commit -am`, followed by a quoted string, both stages and commits all modified files in a single command. Depending on the nature of your changes, you may wish to stage and commit them separately. Also, note that if you wish to add newly-tracked files as part of your commit, they will not be caught using `git commit -am` and will need to be added using `git add` before committing.

Poussez vos changement fait en local vers GitHub :

```
git push --set-upstream origin fixed-broken-thing
```

Now go look at your fork of the L.I.S.A repo on the GitHub website. The new branch will now be listed under the "Source" tab where it says "Switch Branches". Select the new branch from this list, and then click the "Pull request" button.

Put in a descriptive comment, and include links to any project issues related to the pull request.

The repo managers will be notified of your pull request and it will be reviewed. If a reviewer asks for changes, just make the changes locally in the same local feature branch, push them to GitHub, then add a comment to the discussion section of the pull request.

Note : Travis

Whenever you make a pull request against the main L.I.S.A repository your changes will be tested. On average these tests take few minutes to run and once they are complete a PASS/FAIL message will be added to your pull request. This message contains a link to https://www.travis-ci.org where you can review the test results. This message will also generate an email which will be sent to the email address associated with your GitHub account informing you of these results. It should be noted that a test failure does not necessarily mean there is an issue in the associated pull request as the entire development branch is tested.

Note : Versions mineures

Minor releases normally contain bug fixes.

When submitting a pull-request which should be considered for a minor release, please note in the comments that it should be reviewed for inclusion.

Pull requests that are accepted to L.I.S.A but not merged into a minor release will always be available in the next major release.

9.1.2 Maintenir votre Fork de L.I.S.A en synchronisation

L.I.S.A is advancing quickly. It is therefore critical to pull upstream changes from master into forks on a regular basis. Nothing is worse than putting in a days of hard work into a pull request only to have it rejected because it has diverged too far from master.

Pour enlever les modifications en amont :

```
# For ssh github
git remote add upstream git@github.com:Seraf/LISA.git
git fetch upstream

# For https github
git remote add upstream https://github.com/Seraf/LISA.git
git fetch upstream
```

To check the log to be sure that you actually want the changes, run the following before merging :

```
git log upstream/develop
```

Ensuite, pour accepter les modifications et fusionner avec la branche actuelle :

```
git merge upstream/develop
```

Pour plus d'informations, voir 'Guide de dépôt Fork sur GitHub' ou 'Ouvrir le document de comparaison des contributions'

9.1.3 Approuver votre demande de sondage

When making pull requests, please sign-off the pull request in a pull request comment. You can simply make a comment something like :

```
Signed-off-by: John Doe <john.doe@hisdomain.com>
```

By signing-off you indicate that you are accepting the Developer Certificate Of Origin. For now, we are using them same DCO as Linux kernel developers are using.

9.2 Bidouiller L.I.S.A

9.3 Exécution des tests

Comme L.I.S.A est écrit avec Twisted il utilise le test du système intégré s'appelant "trial". Pour le lancer ;

```
trial lisa/server
```

9.4 Tests d'écriture

9.5 Politique de git

L'équipe de L.I.S.A suit la politique de git afin de maintenir la stabilité et la cohérence avec le référentiel.

La politique de git à été développé afin d'encourager les contributions et vous faciliter dans votre contribution à L.I.S.A. Les contributeurs du projet L.I.S.A N'ONT PAS BESOIN DE LIRE CE DOCUMENT, car tous proviennent d'une voie unique.

La première règle du management de git pour L.I.S.A est de faciliter les contributeurs et développeurs pour l'envoie de code. La simplicité est toujours notre but !

9.6 Conventions

CHAPITRE 10

Notes de sortie

10.1 L.I.S.A 0.2 Release Notes

release 2014-09-07

0.2 has arrived ! This release come with new features !

10.1.1 Major Features

- New interface written in AngularJS
- Everything use API
- Upgrade plugins and core from Interface
- Use sockjs for Websocket

10.1.2 Compatibility

You will have to delete your user from Mongo Database.

```
-seraf@tagada-laptop ~/Sandbox/LISA ‹lisa› ‹master*›
- mongo

MongoDB shell version: 2.4.9
connecting to: test
> use lisa
switched to db lisa
> db.user.remove()

lisa-cli createsuperuser
```

10.2 Archive

10.2.1 L.I.S.A 0.2 Release Notes

release 2014-09-07

0.2 has arrived ! This release come with new features !

Major Features

- New interface written in AngularJS
- Everything use API
- Upgrade plugins and core from Interface
- Use sockjs for Websocket

Compatibility

You will have to delete your user from Mongo Database.

```
-seraf@tagada-laptop ~/Sandbox/LISA ‹lisa› ‹master*›
- mongo

MongoDB shell version: 2.4.9
connecting to: test
> use lisa
switched to db lisa
> db.user.remove()

lisa-cli createsuperuser
```

CHAPITRE 11

Comment contribuer à la documentation ?

CHAPITRE 12

Foire Aux Questions

FAQ

12.1 Est-ce que LISA est open-source ?

Oui. LISA est 100% opensource, idem pour toutes les API.

Et à l'avenir, mais si je crée une entreprise avec LISA, le projet restera à code ouvert et libre.

12.2 Puis je en faire un fork ?

Oui vous pouvez en faire un fork si vous voulez. Tout sera à code ouvert, même le "appstore". Mais évidemment j'encourage tout le monde à contribuer au projet au lieu de diluer nos talents.

12.3 C'est LISA est en mode local ou faut-il une connexion internet ?

LISA à actuellement besoin d'une connexion internet cependant je travaille dessus afin de fournir quelque chose qui fonctionne même quand la connexion se coupe.

LISA à au moins besoin d'internet pour son client (puisque la reconnaissance vocale est en ligne) et pour le moteur Wit sur la partie serveur. Wit fonctionne également sur un serveur local pour Wit. Une fois disponible, il sera retro-porté sur LISA le plus rapidement possible.

12.4 Quel moteur de synthèse vocal utilisez vous ?

Même si elle une partie les plus importante du projet, je n'ai pas envie de me focaliser dessus. Le client Linux actuel utilise pocketsphinx afin de détecter keyword (le nom du robot). C'est le déclencheur pour lancer la reconnaissance vocale par voix. LISA écouter et diffuse chaque fragment sonore vers l'API Wit Speech. C'est un service web en ligne et ils utilisent plusieurs moteurs vocaux pour la reconnaissance vocale. Ils ont l'intention de créer des modèles acoustiques gratuits et utiliser (dans le futur) essentiellement un moteur vocal opensource. Ils sont partenaire de CMUSphinx.

Pas convaincu ? Vous ne voulez pas voir votre voix aller en dehors de votre maison ?

Eh bien, LISA a été construit afin que l'utilisateur puisse choisir son moteur de reconnaissance vocale. Vous pouvez créer un robot client sur irc, un robot hangout, et jabber, le client voix peut utiliser Dragon Natural Speech, Windows Grammar, Android Speech Engine ou autres. C'est vous qui décider, il suffit d'envoyer son choix à LISA.

N'oubliez pas que le plus petit est le périphérique sur lequel est hébergé le client, Moins il y aura de capacité de calcul pour la reconnaissance en mode local. Mais pas de panique, je souhaite pour moi quelque chose en local et fonctionnent sans connexion internet.

12.5 Pourquoi LISA utilise mongodb comme base de données ?

Je suis un DevOps, donc quand je code, je reflechi toujours à comment mon code va fonctionner en production. Est il évolutif ? Mongodb permet à LISA d'être compatible et de s'adapter à la perte d'un serveur. La base de données peut être distribué à travers les clients par example. La technologie NoSQL permet aux développeurs de créer facilement des plugins.

Vu que les plugins et la configuration est stocké dans la base mongodb, vous pouvez facilement mêttre à échelles le serveur LISA avec une répartition de charge.

Le seul problème est le manque de support de la plateforme ARM. Mais il y à un ticket d'ouvert sur leur traqeur de bogues JIRA.

12.6 Can I install LISA on a Raspberry Pi ?

Quick answer : Yes.

Long answer : RPI is an ARM architecture and actually Mongodb is unsupported officially. So it isn't packaged for raspbian for example. But you should be able to compile it (take ~7hours).

There's a ticket already opened on their JIRA for ARM support, and they are working on it.

By now, I suggest to run mongodb (and the LISA server) in a virtual machine or a dedicated x86 server.

12.7 Quels ports dois-je ouvrir sur mon pare-feu ?

LISA utilise par défaut le port 10042 pour communiquer entre le client et le serveur, et le port 8000 pour le serveur web.

12.8 Puis je faire de la domotique avec ce projet ?

Réponse rapide : Non. LISA vous permet de créer un pont entre vos objets connectés et votre boitier domotique. LISA peut également gérer quelques "intelligence" à exécuter des ordres sur votre boite, mais cela ne replacera pas un boitier domotique.

12.9 Pourquoi LISA est écrit en Python ?

Premièrement, je cherche à apprendre ce langage depuis longtemps. Le code peut ne pas être parfait alors n'hésitez pas à envoyer un correctif. J'aime Python car beaucoup d'API sont écrit avec ce langage et comme LISA est construit afin de d'interconnecter les API, c'est donc logique langage facile d'apprentissage et facile à utiliser.