Light Exemple C

Table des matières

[1. Librairie (lib) 2](#_Toc106978969)

[2. Fichier config (etc/cfg) 2](#_Toc106978970)

[3. Projet principal (src) 3](#_Toc106978971)

[a. Makefile 3](#_Toc106978972)

[b. Main du projet 3](#_Toc106978973)

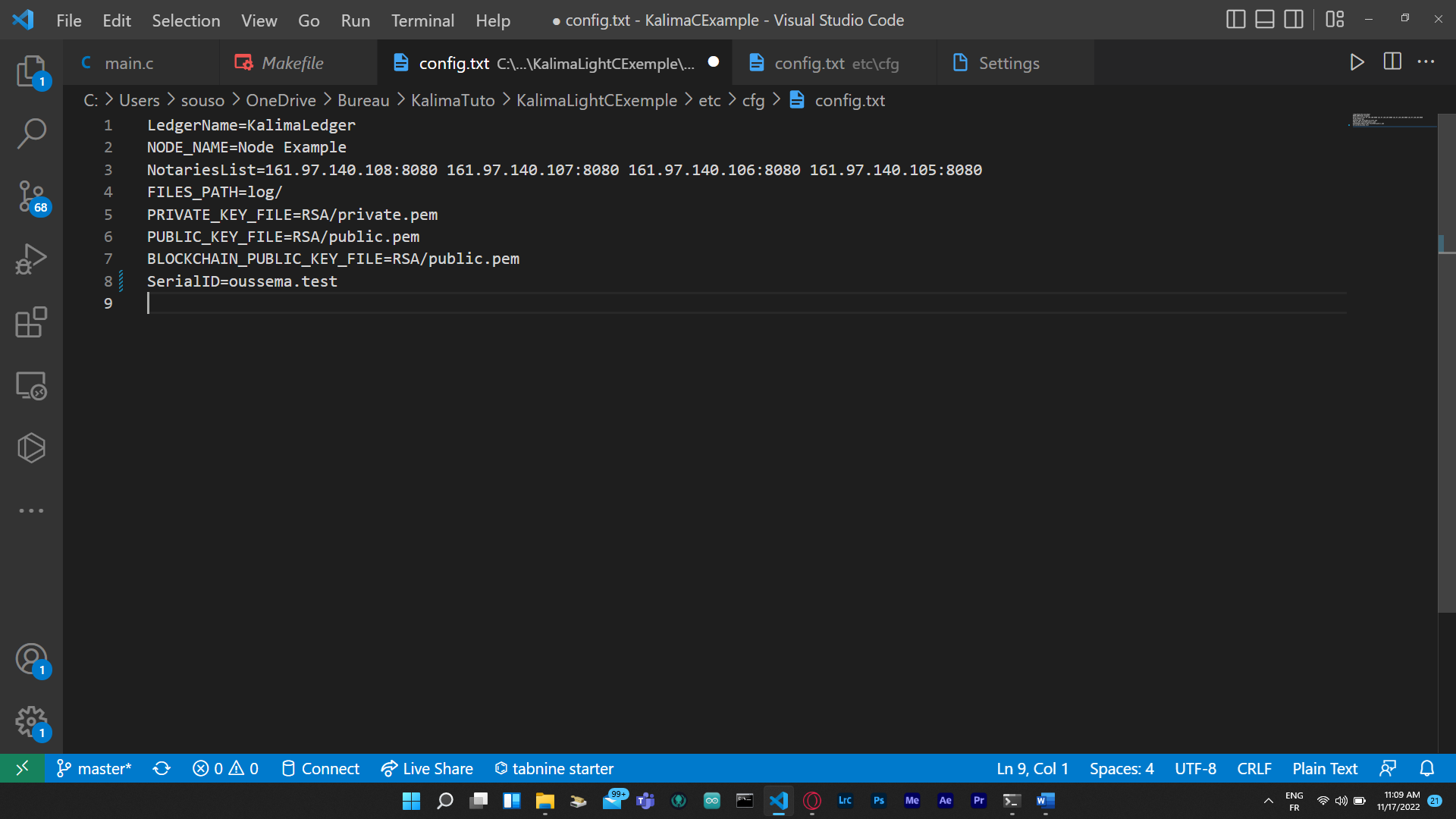
[4. Exécution du programme 5](#_Toc106978974)

[5. Problèmes possibles 6](#_Toc106978975)

# Librairie (libs)

Cette librairie contient tous les headers de la librairie, ainsi que l’archive statique libKalimaMQC.a aui contient nos objets crées depuis notre code source. Cette archive nous servira lors de la création de l’exécutable dans le projet (src). Les headers nous serviront à appeler les méthodes qui nous seront utiles dans notre exemple.

# Fichier config (etc/cfg)

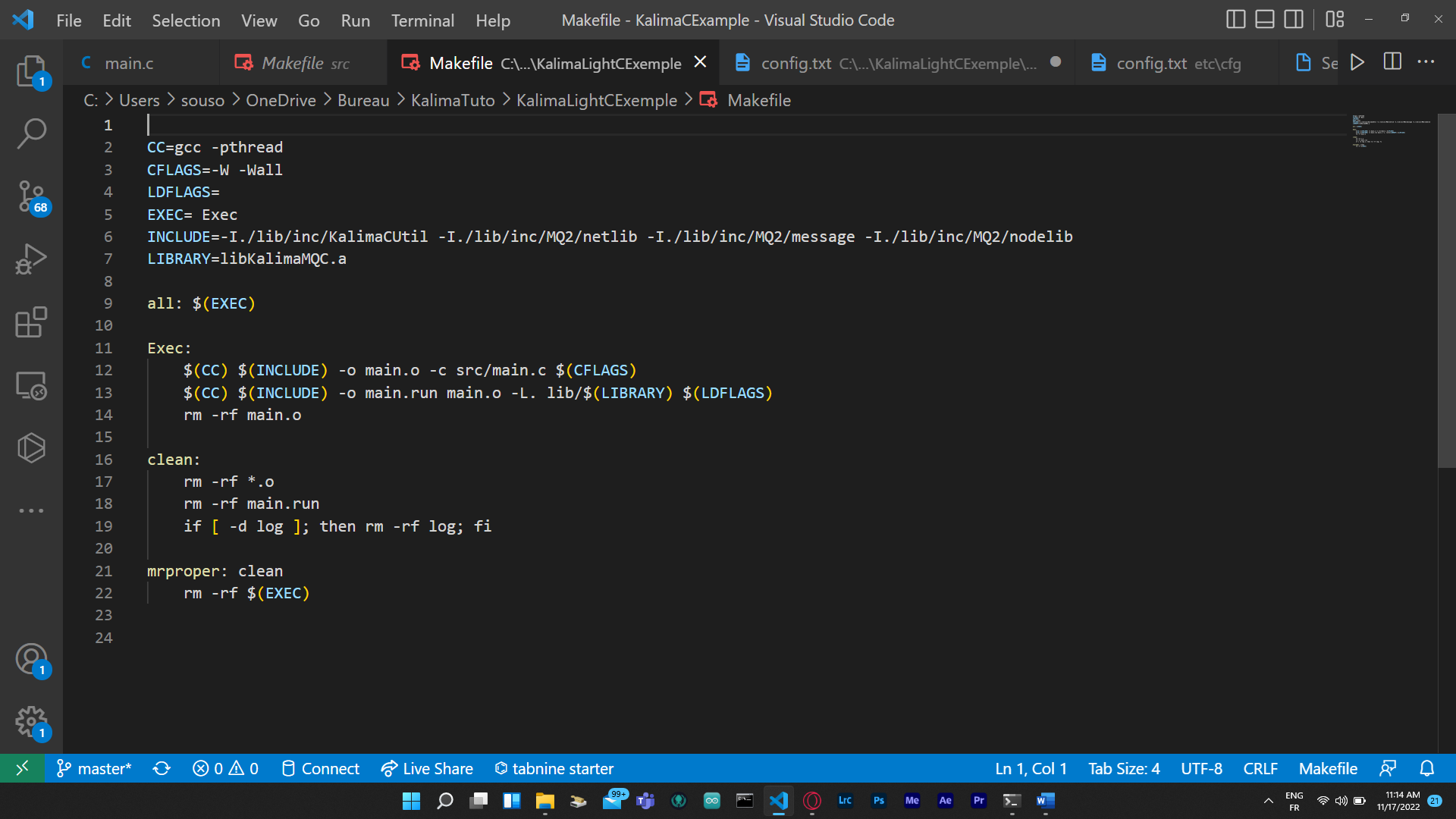


Voyons l’utilité des différentes configurations :

* LedgerName -> C’est le nom du Ledger auquel nous allons nous connecter. Pour l’exemple ce n’est pas très utile.
* NODE\_NAME -> C’est le nom du node que nous créons. Ce n’est pas non plus très utile pour notre exemple
* NotariesList -> C’est la liste des nodes auxquels nous devons nous connecter pour communiquer avec la blockchain tuto. Si vous souhaitez vous connecter à une autre blockchain, il suffit de changer les notaries ici. Entre chaque node, il faut mettre un espace.
* FILES\_PATH -> C’est le directory dans lequel on pourra trouver les fichers de logs. Dans notre exemple, le dossier log se créera au lancement dans le directory
* KEY\_FILES -> Ce sont les chemins des différents fichiers de cryptage RSA. Ces fichiers sont nécessaires pour communiquer avec la Blockchain. Ils seront également créés automatiquement au lancement du projet.
* SerialID -> ID permettant l’autorisation de votre nœud sur la blockchain que vous pouvez obtenir ici : <https://inscription.tuto.kalimadb.com/airdrop> (vous disposez de 10 serialId reçus par mail).

# Projet principal (src)

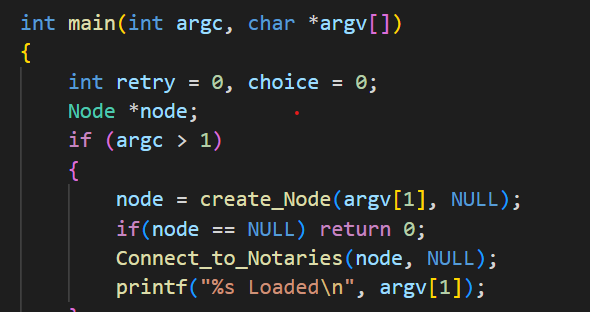
## Makefile



Le makefile nous permet de créer un objet pour notre main. Ensuite, à partir de cet objet et de l’archive de la librairie, nous pouvons créer notre exécutable « main.run » en indiquant qu’on utilisera les threads et l’emplacement des includes.

* Pour générer le main.run tapez : make
* Pour nettoyer le projet tapez : make clean

Main du projet



Ici, nous créons notre node. Par défaut, le node utilisera le fichier « config.txt » comme fichier de config. Ici, on doit préciser l’emplacement du fichier de config (dans notre exemple on tape : **./main.run etc/cfg/config.txt** sinon il prendra l’emplacement par défaut prédéfinit.

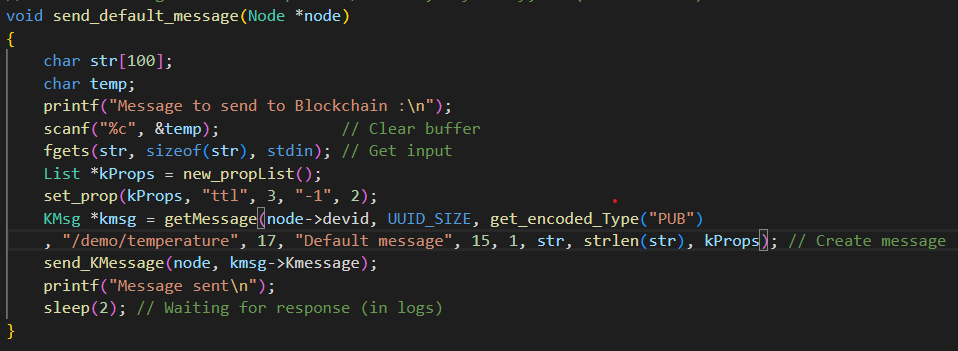
Lors de la création du node, nous allons également créer un deviceID aléatoire qui sera crypté et écrit dans le fichier « DeviceID » qui sera créé dans le dossier src. Si ce fichier crypté existe déjà, il ne sera pas recréé. Le node va juste décrypter le fichier et utiliser le deviceID récupéré. Ce deviceID, en lien avec le SerialID, permettra à la blockchain d’identifier notre node et de nous permettre d’écrire dans celle-ci. Un dossier RSA sera également créé contenant une clé publique et une clé privée servant au cryptage des communications avec la blockchain.

Les paramètres « NULL » que l’on peut voir ici correspondent au fait que l’on peut permettre au node d’utiliser des scipts LuaJIT avec un callback. Vu que dans cet exemple nous n’en utilisons pas, il faut mettre NULL.

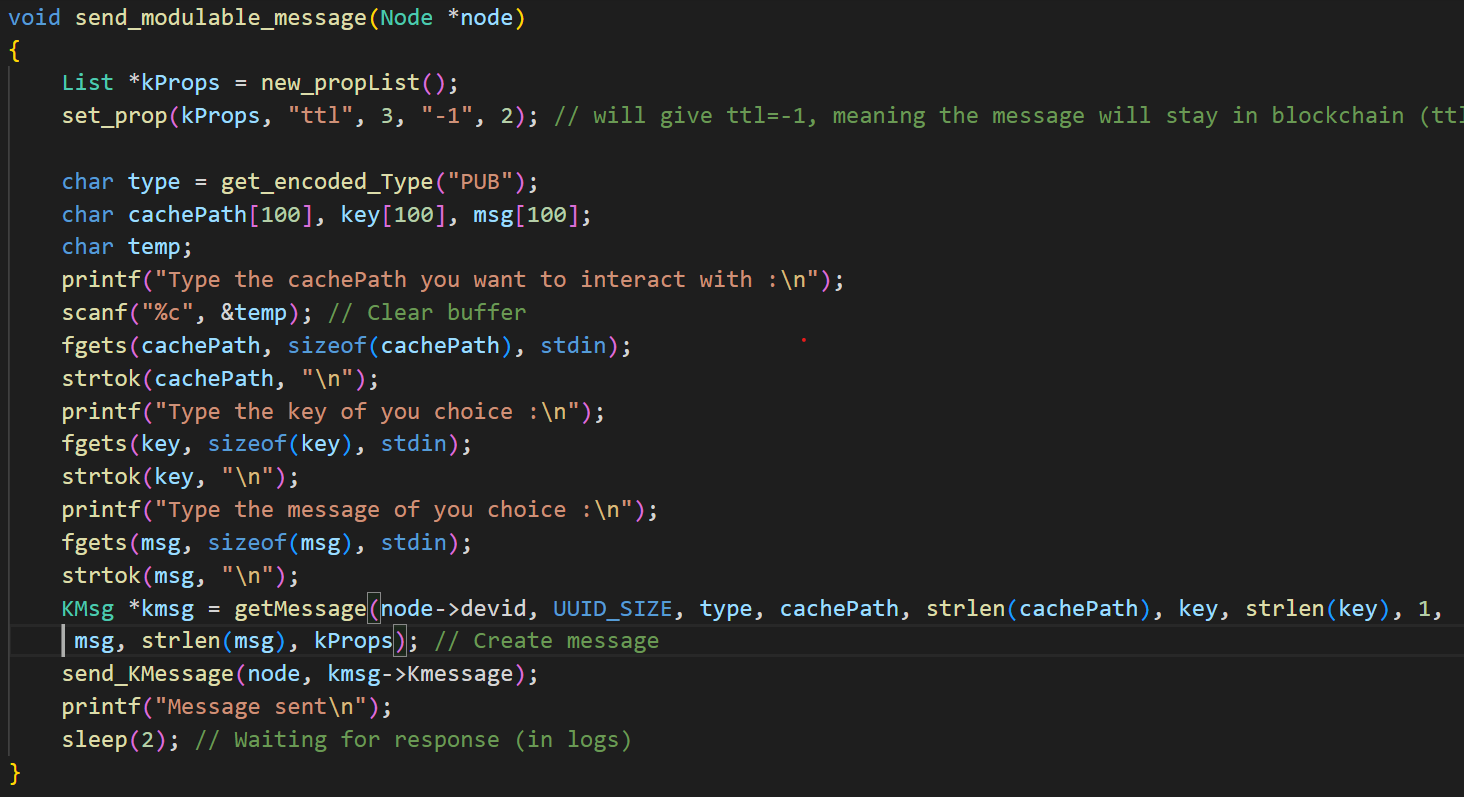
Enfin, il se connectera directement aux notaries indiqués dans le fichier config.

Après la création du node, l’exemple propose deux choix à l’utilisateur :

* Envoi de message par défaut



Ce choix envoie un message que l’utilisateur va saisir à la blockchain sur l’adresse « /demo/temperature ». le message sera permanent dans la blockchain parceque on a mis le ttl a -1 sinon le message aura un temps bien précis avant qu’il soit supprimer de la blockchain.

Message configurable :

Ici nous construisons nous même le message.

On propose tout d’abord à l’utilisateur de choisir le cache path (exemple /demo/tmp). Ensuite le key (exemple température). Enfin la valeur (exemple 35 C) après on envoie le message. Comme l’exemple précédent, le message restera ici indéfiniment.

# Exécution du programme

Pour exécuter l’exemple, il suffit d’ouvrir un terminal et de se placer dans le dossier « src ».

Il suffit ensuite simplement de taper « make» pour lancer le makefile vu précédemment. Cette commande créera le fichier exécutable « main.run » qui nous permettra de lancer le programme (ainsi qu’un fichier main.o qui peut être supprimé, le dossier DevID et le dossier RSA).

Enfin, il faut écrire « ./main.run » pour lancer le programme.

# Problèmes possibles

Si votre message n’apparaît pas dans la blockchain, voici quelques possibles erreurs :

* Vérifiez que le SerialID dans le fichier config est le même que celui entré dans la blockchain.
* Si vous supprimez un fichier DeviceID ou RSA, le nouveau calculé sera différent. Il faudra donc refaire le SerialID car le DeviceID associé ne sera plus le même.
* Si vous avez fait des modifications sur le main qui ne sont pas prises en compte, faîtes un « make clean » avant de refaire « make » pour que toutes les modifications soient bien prises en compte.
* Si vous avez d’autres problèmes, vous pouvez regarder dans les logs quand le programme a des problèmes et nous contacter.