Chapitre 1

Contexte

Le Moteino est un ordinateur de très petite taille et peu cher. Il est basé sur l'Arduino, très modulaire.

Le but de ce projet est de prendre en main l'architecture embarquée du Moteino : produire du code C++ à installer sur un Moteino, mais surtout comprendre comment manipuler correctement cette architecture. La production de code C++ demande l'utilisation d'outils particuliers. Arduino propose un IDE minimaliste qui ne convient pas aux projets industriels, nous utilisons donc platformio pour automatiser la gestion des librairies, compiler et envoyer le code sur les chipsets.

Chapitre 2

Liens utiles

le git du projet
Le site de platformio

Chapitre 3

Organisation du projet

3.1 Code source

Le répetoire src/ contient trois répertoires :

src/main contient les sources principaux à installer sur les chipsets. Chaque répertoire contient un module à installer sur un Moteino.

src/lib contient les librairies que nous développons, utilisées dans les main.

src/ext contient le librairies que nous avons importées, après d'éventuelles modifications, depuis d'autres sites.

src/test Contient les tests (pas utilisé pour l'instant)

Le répertoire .tmp/ contient entre autre les dépendances gérées par plateformio. Il peut etre supprimé sans danger.

3.2 Documentation

Le répertoire doc/ contient la documentation de ce projet, en particulier ce fichier et le code permettant de le générer.

3.3 Outils de développement

diffrents outils sont utlisés pour s'assurer du fonctionnement de ce projet.

3.3.1 Gestionnaire de version : git

Présent par défaut sous ubuntu.

Pour télécharger le git du projet à des fins de test il suffit de faire
`git clone https://github.com/glelouet/Moteino.git`

Pour modifier le projet il faut par contre utiliser la connexion git en ssh, par exemple
`git@github.com:glelouet/Moteino.git`

3.3.2 outils en ligne de commande : pio

Platformio met à disposition une commande pour compiler automatiquement le code, pio.

Une fois le code téléchargé, il suffit de faire `pio -t upload` dans un répertoire de main pour déployer les binaires sur les chipsets.

3.3.3 Interface graphique : Atom/Platformio

Utilisant pio pour le déploiement en ligne de commande, il est pratique d'utiliser l'IDE correspondant qui a une intégration native avec les fichiers de configuration. Cependant d'autres IDE peuvent surement etre utilisés.

Éditeur Atom

Atom est un éditeur de texte hautement modulaire. Télécharger atom sur le site officiel

Plug-in Platformio

Le plug-in Plaformio pour Atom permet le déploiement rapide depuis l'éditeur des binaires sur les chipsets.

Pour l'installer suivre le site officiel.

Une fois installé, il faut ouvrir le projet avec `file/open prject flder` et entrer un des répertoires présents dans src/main. Pour installer ce projet il suffit de cliquer à gauche sur la flêche de légende `platformio:upload` . si l'upoad ne fonctionne pas, il faut s'assurer que le `upload_port` du fichier platformio.ini correspond bien au port USB du moteino. En cas de multiple pors USB ceux-ci peuvent varier.

Chapitre 4

Coder le Moteino

Le moteino est mono-thread et basé sur une boucle de gestion. Un code déstiné au Moteino, ou plus généralement à un Arduino, doit avoir deux fonctions :

1. une fonction d'initialisation des élements appelée `init()`.
2. une fonction utilisée périodiquement appelée `loop`

Chaque fichier source dédié à etre injectée dans un moteino doit donc avoir ces deux méthodes. L'utisation de librairies doit donc prendre en compte cette contrainte.

4.1 Principe de code

Les librairies sont des éléments de code partagées. Elles représentent généralment un élément physique du moteino, ou un protocole. Un fichier source peut donc utiliser des librairies, il est responsable de les initialiser durant son init() et de les appeler durant son loop().

Étant donné que le Moteino est mono-thread, il ne faut surtout pas avoir d'appel à sleep() dans le code, que ce soit dans le main ou dans les librairies. En effet un appel à sleep() bloque tout le système, les protocole réseaux auront donc des timeout. Il vaut mieux à la place enregistrer le temps de la prochaine "sortie de veille" et ne rien faire dnas les fonctions loop() tant que ce temps n'est pas atteint.

4.2 Les librairies présentes

Deux types principaux de librairies sont présentes dans le répertoire src/lib. Les librairies de drivers codent l'accès en dur aux élements du moteino. Les librairies de gestion codent les protocoles logiciels. Dans certains cas le driver est fourni dans le meme répertoire que la librairie de gestion.

4.2.1 Moteino

4.2.2 RFM69

4.2.3 ButtonCommand

La librairie Button écoute sur un pin la tension (0 ou 1) présente à chaque loop(). Elle enregistre les variations sous la forme d'une série de durées d'appui. Lorsqu'un relachement suffisamment long (3s) est fait, cette série est mise à disposition pour un autre module.

La librairie ButtonCommand quant à elle traduit les séquences d'appuis en actions à effectuer sur le Moteino, en fonction du nombre et de la durée des appuis.

2 appuis démarrage du mode pairing

3 appuis extinction du mode pairing

un appui court moins de .5s, demande aux éléments du réseau de clignoter

un appui moyen entre .5 et 3s, tente de se connecter à un moteino en mode "pairing"

un appui long entre 3 et 10s acquiert un réseau aléatoire et se met en mode "pairing"