Télécommande Bluetooth Android

Une télécommande et son application Android associée, utilisée dans le cadre d'un projet en collaboration avec la BSPP, pour commander une application de VoIP (Mumla).

- Présentation du projet
- Étapes
- Fichiers sources et références

Présentation du projet

Informations

- Nicolas PEUGNET
- nicolas.peugnet@lip6.fr
- LIP6

- 15/01/2024 - Date de fin estimée (ou réelle)

Contexte

Dans le cadre d'un projet en collaboration avec la Brigade des Sapeurs Pompiers de Paris (BSPP), nous avions besoin d'une télécommande avec quelques boutons, nous permettant de piloter une application Android de Voix sur IP (VoIP), afin de se passer de l'écran tactile et de pouvoir l'utiliser sans enlever les gants et sans avoir à regarder l'écran.

Objectifs

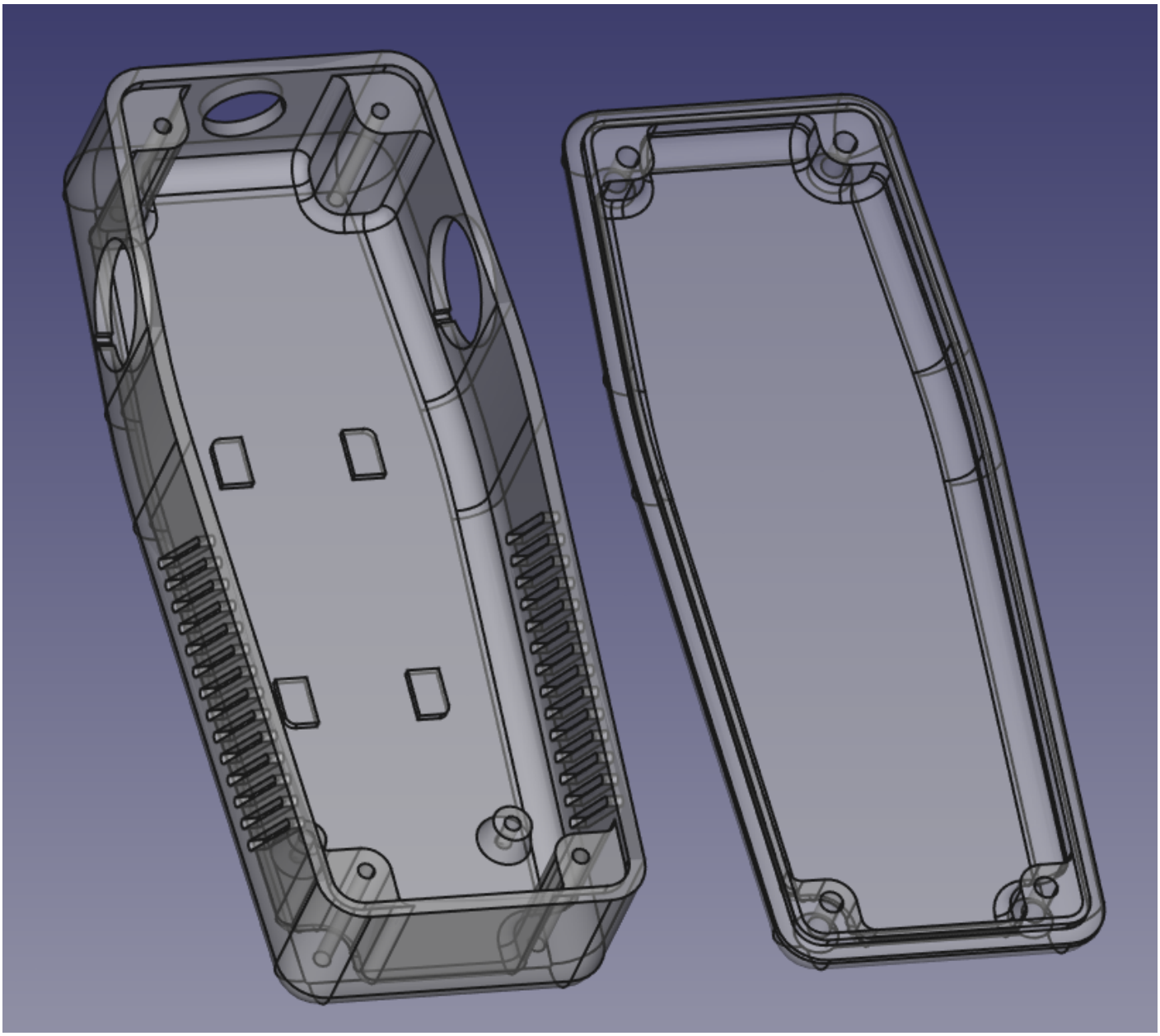
Réaliser un boîtier pour cette télécommande. Comme il s'agit d'un prototype qui ne sera utilisé que pour une expérimentation, un boîtier en impression 3D sera amplement suffisant. Il faudra en revanche le produire en de multiples examplaires, pour que chaque pompier en ait un lors de l'expérimentation.

Description

On veut réaliser une télécommande Bluetooth, basée sur une carte Arduino, permettant de contrôler à distance une application de communication vocale. Cette télécommande comprend trois boutons pouvant servir à déclencher des actions sur l'application Android.

Les données des boutons sont transmises via le "Bluetooth Low Energy" (BLE), grâce à un serveur GATT (voir la spécification) à l'application de communication vocale.

Pour maximiser la modularité du projet, une application Android faisant l'interface entre la télécommande et la ou les applications de communication vocale a été développée. Mais il est aussi possible pour une application de communication vocale de se connecter directement au serveur GATT de la télécommande.



Étapes

Étapes à suivre pour refaire le projet (conception, construction, réalisation, manipulation...)

Étape 1 : Arduino + électronique

Dans cette étape nous allons réaliser le circuit électronique, puis compiler et flasher le code sur l'Arduino.

Prérequis

Les prérequis logiciels et matériels de la première étape.

Logiciels

- Git
- Arduino-cli : https://arduino.github.io/arduino-cli

Matériel

- Arduino nano 33 BLE (simple ou Sense Rev2)
- PC

Circuit électronique

Le circuit est extrêmement simple. Il est représenté dans le README du dépôt de code :

https://gitlab.lip6.fr/ene5ai/arduino-ptt

```
 _|_   |    |      D4----+--|--|--10kohm--+
|   | |    |      D3-------+--|--10kohm--+
| - | |    |      D2----------+--10kohm--+
|   | |    |        |                 |
|   | |    |     GND---------------------+
| + |  +-----GND      |
|_ _|   +----VIN      |
  |     |   |________|
   +-----+
```

Code Arduino

La télécommande est prévue pour fonctionner avec la carte "Nano 33 BLE", ou bien "Nano 33 BLE Sense Rev2", car celles-ci sont de taille réduite et intègrent une puce Bluetooth supportant le BLE. Le code peut donc être compilé pour l'une ou l'autre de ces cartes. S'il s'agit de la deuxième, des fonctionnalités supplémentaires sont activées : les données de trois de ses capteurs (pression atmosphérique, température et humidité) sont également lisibles via le serveur GATT.

```
# Install arduino:mbed_nano core:
arduino-cli core install arduino:mbed_nano

# Install required library:
arduino-cli lib install ArduinoBLE

# Install optional libraries for the sense rev2 board:
arduino-cli lib install Arduino_LPS22HB Arduino_HS300x

# Build for nano33ble board:
make nano33ble

# Upload to nano33ble board:
make upload

# If your board is connected to a different port tha /dev/ttyACM0
# You can list connected boards to find the port, for example here it is /dev/ttyACM0:
#  Port         Protocol Type              Board Name  FQBN            Core
```

```
#  /dev/ttyACM0 serial   Serial Port (USB) Arduino Uno arduino:avr:uno arduino:avr
arduino-cli board list

# Then set the PORT environment variable when uploading
PORT=/dev/ttyACM0 make upload

# On Linux, the serial ports are probably not writable by default.
# On Debian and derivatives, the easiest way to get access to them
# is to add your user to the `dialout` group, then log out/in.
# See: <https://wiki.debian.org/SystemGroups#Other_System_Groups>
sudo adduser $USER dialout
```

Étape 2 : Application Android

L'application android utilisée est assez spécifique à notre cas d'usage, mais il est possible de l'adapter à d'autres usages : https://gitlab.lip6.fr/ene5ai/android-bt

Prérequis

- JDK: >= 11, <= 19
- Android platform: 31
- Android build tools: 30.0.3
- Make

Sur Debian ou un dérivé, il est possible d'installer et configurer tous les prérequis de la manière suivante :

Avec le component `non-free` activé :

```
sudo apt install default-jdk make \
    google-android-platform-31-installer \
    google-android-build-tools-30.0.3-installer \
    google-android-platform-tools-installer \
    google-android-cmdline-tools-9.0-installer
```

Définir ensuite la variable d'environnement suivante :

```
ANDROID_HOME=/usr/lib/android-sdk/
```

Et finalement, accepter les licences Android :

```
sudo sdkmanager --licenses
```

Compilation

Pour compiler l'application, il suffit de lancer la commande `make build` puis `make install` pour l'installer sur un mobile Android.

Étape 3 : Impression et montage du boîtier

Prérequis

La modélisation 3D a été réalisée (tant bien que mal) grâce à FreeCAD, mais il n'est nécessaire que pour modifier le fichier source (ptt-case.FCStd). Le fichier STL peut être utilisé tel quel.

Impression

Paramètres recommandés :

hauteur de couche	0.2mm
épaisseur de coque	2
remplissage	20%
supports	contact uniquement

Fichiers sources et références

Fichiers 3D du boîtier

- ptt-case.FCStd
- ptt-case.FCStd (latest git)
- ptt-case-lowres.stl

Codes sources

- gitlab.lip6.fr/ene5ai/arduino-ptt
- gitlab.lip6.fr/ene5ai/android-bt