Documentation de la suite logicielle GBMProject3A

Guillaume Gibert

08/04/2020

# Introduction

Ce document décrit les différents éléments de la suite logicielle GBMProject3AInstallation Android Studio

## Côté PC

Le code fourni est écrit en C++ avec comme unique dépendance l'API Qt.

Il se compose d'un ensemble de classes qui permet :

* de communiquer avec un système Arduino que ce soit en réception ou en émission;
* de traiter des signaux (filtrage, FFT);
* d'afficher des signaux;
* de faire des requêtes sur une base de données MySQL.

## Côté Arduino

Un code de test est fourni qui permet d'envoyer sous forme d’une chaîne de caractères standardisée les valeurs lues sur les 5 canaux analogiques de la carte Arduino via le port série et via Bluetooth (module HC-05 par exemple).

Une fonction de lecture d'évènements est également mise en place pour lire, en parallèle, les données sur le port série provenant du PC.

Une série d'exemples permet d'implémenter chaque fonctionnalité. Ils vont être décrits dans la suite du document.

# Installation

## Qt

Pour utiliser le code fourni dans ce projet, il faut installer l'API Qt.

Pour ce faire, il faut :

* aller sur le site de Qt <https://www.qt.io/>
* télécharger l'installateur pour la version Open Source (Onglet "Download. Try .Buy", puis "Go for Open Source", enfin "Download the Qt Online Installer");
* lancer l'installateur (vous devrez créer un compte Qt si vous n'en possédez pas) et choisir la version Open Source;
* choisir une version de Qt pour un compilateur donné (par exemple MinGW 64-bit)
* finaliser l'installation en suivant les instructions.

Une fois l'installation terminée, vous trouverez l'outil qtcreator.exe dans **QTDIR**/Tools/QtCreator/bin où **QTDIR** est le répertoire où vous avez choisi d'installer Qt.

## Arduino

Pour pouvoir utiliser la carte Arduino et téléverser vos programmes dans celle-ci, il faut installer l'IDE Arduino.

Pour ce faire, il faut :

* aller sur le site d'Arduino <https://www.arduino.cc/>
* télécharger l'installateur (Onglet "Software" puis "Downloads");
* lancer l'installateur;
* finaliser l'installation en suivant les instructions.

## Github

Pour récupérer le code de ce projet, deux options sont possibles :

1. Télécharger les sources
   * aller sur le site <https://github.com/GuillaumeGibert/GBMProject3A> ;
   * cliquer sur "Clone or download";
   * cliquer sur "Download ZIP".
2. Cloner les sources
   * aller sur le site <https://desktop.github.com/>
   * cliquer sur "Download for Windows";
   * installer Github Desktop;
   * cloner le projet :
     + ouvrir une commande Git bash ;
     + taper : git clone <https://github.com/GuillaumeGibert/GBMProject3A.git>

## GBMProject3A

### Arduino

Il faut charger dans la carte Arduino le projet testArduino.ino qui se trouve dans **GBMProject3ADIR**/testArduino où **GBMProject3ADIR** est le répertoire dans lequel vous avez téléchargé les sources du projet.

Un programme Arduino est composé de 2 fonctions au minimum :

* une fonction setup() dans laquelle les variables sont initialisées et qui est appelée une seule fois au début du programme;
* une fonction loop() qui est appelée régulièrement dès qu'elle est terminée.

### PC

Il faut ouvrir le fichier .pro correspondant à l’exemple choisi à l'aide du logiciel qtcreator.exe.

# Code Arduino

## Variables globales

Le code Arduino est composé de variables globales :

* \_nbPinIn : nombre de pins analogiques connectés ;
* \_portBaud : taux de transfert sur le port série ;
* \_fps : fréquence de rappel de la fonction loop().

## setup()

La fonction setup() initialise les port série et Bluetooth avec le taux de transfert spécifié dans la variable \_portBaud.

## loop()

La fonction **loop()** lit les valeurs sur les ports analogiques de la carte grâce à la commande **analogRead(int**). Le temps de lecture des données et d’envoi sur le port série est mesuré et est stocké dans la variable **elapsedTime**. En fonction de la valeur du taux de rafraîchissement **\_fps**, la fonction attend un délai correspondant à la période de rafraîchissement moins la valeur de **elapsedTime**.

Elle crée une chaîne de caractères de la forme :

Sensor:|<input1>|<input2>|<input3>|<input4>|<input5>|\n

Puis elle envoie cette chaîne de caractère sur les ports série et Bluetooth grâce à la fonction Serial.println(String) et virtualSerial.println(String).

## serialEvent()

La fonction serialEvent() est lancée entre deux appels à la fonction loop() si des données sont reçues sur le port série.

Cela permet de récupérer sur la carte Arduino des informations en provenance du PC.

# Exemple 1 : Récupérer des données depuis un Arduino (liaison filaire)

La carte Arduino est connectée via un câble USB au PC.

La fonction **loop()** est appelée régulièrement à la fréquence **\_fps**. Elle lit les données sur les ports analogiques grâce à la fonction **analogRead(l\_pinIn)**, puis envoie ces valeurs sur le port série grâce à la fonction **Serial.print(analogRead(l\_pinIn))**.

Côté PC, deux objets sont instanciés : un de type SerialPortManager et un de type Spy.

Un objet spm de type SerialPortManager récupère les données envoyées par l'Arduino sur le port série.

Un ensemble de valeurs doit être fournies à l’objet spm. Attention, il faut vérifier que le taux de transmission (baud rate) est identique à celui spécifié dans le code Arduino. Ensuite, la méthode findSerialPort(« Arduino ») cherche si la carte Arduino est détectée sur le PC. Enfin, la méthode openSerialPort() ouvre le port série et commence à récupérer les données envoyées par la carte Arduino.

Un objet spy de type Spy est créé.

L’objet spm et spy sont connectés via un signal/slot Qt. Les données reçues par spy sont affichées à l'écran.

# Exemple 2 : Récupérer des données depuis un Arduino (liaison Bluetooth)

Cet exemple est similaire au précédent. La seule différence réside dans le type de connexion entre l'Arduino et le PC. Dans l'exemple précédent, L'Arduino était connecté avec un câble USB au PC alors que dans cet exemple, il est connecté en Bluetooth via un module HC-05 par exemple.

Côté Arduino, il faut connecter le module HC-05 de la façon suivante :

* la borne TX du module HC-05 sur l’entrée Rx de l’Arduino ;
* la borne RX du module HC-05 sur l’entrée Tx de l’Arduino ;
* la borne VCC du module HC-05 sur 5V ;
* la borne GND du module HC-05 sur GND.

La bibliothèque SoftwareSerial est utilisée pour créer un port série virtuel pour la communication Bluetooth. Il est utilisé de la même façon que l’objet Serial.

Côté PC, il faut activer le Bluetooth. Il faut ensuite se connecter au module HC-05. Le mot de passe par défaut est 1234. Le code est très similaire à l’exemple précédent à l’exception de deux points :

* spm.findSerialPort("Bluetooth"); : on recherche un port COM nommé Bluetooth ;
* //spm.setFirstPortFound(false); : Windows crée deux ports COM pour la communication Bluetooth ; par défaut le premier sert à la communication et cette ligne doit rester commenter. Parfois, les ports sont inversés et le second sert à la communication. Si la communication n’est pas établie, il faudra décommenter cette ligne.

# Exemple 3 : Récupérer des données depuis un générateur de signaux

Cet exemple montre comment utiliser un générateur de signaux quand aucune carte Arduino n’est disponible.

Un objet de type SineGenerator est créé. On lui spécifie le nombre de signaux et leurs caractéristiques (amplitude, phase, fréquence) ainsi que la fréquence d’échantillonnage. Ce générateur va envoyer les amplitudes des signaux à intervalles réguliers comme le fait l’objet de type SerialPortManager.

Un objet de type Spy est également crée.

Ces deux objets sont connectés via un signal/slot Qt. Les valeurs reçues par l’objet de type Spy sont affichées à l’écran.

# Exemple 4 : Afficher des signaux temporels

Dans cet exemple, un générateur de signaux envoie les valeurs d’amplitude des sinusoïdes générées à intervalles réguliers.

Un objet de type MainWindowEx4 est créé. Cette classe hérite de la classe MainWindow qui permet d’afficher des signaux temporels ou bufferisés. Dans cet exemple, un objet de type TemporalSignalDisplay est créé. Les caractéristiques des signaux (nombre, amplitude min/max, fréquence d’échantillonnage, etc.) à afficher sont renseignées. Il permet d’afficher les signaux temporels au fur et à mesure de la réception des échantillons.

Les objets signalGenerator et window sont connectés via un signal/slot Qt. Dès que de nouveaux échantillons sont envoyés par signalGenerator, ils sont affichés graphiquement par window.

# Exemple 5 : Filtrer des signaux temporels

Dans cet exemple, un générateur de signaux envoie de façon régulière les échantillons à un objet de type SignalProcessing. Cet objet va appliquer un filtre IIR dont les coefficients sont spécifiés dans le constructeur à tous les nouveaux échantillons avant de les renvoyer grâce à un signal Qt.

Un script matlab/octave intitulé generateLowPassFilterCoefficients.m est disponible dans le répertoire octave du dépôt git. Il permet de générer les valeurs des coefficients d’un filtre de type Butterworth et de les sauvegarder dans un fichier texte pour pouvoir les recopier facilement.

L’objet window de type MainWindowEx5 récupère les signaux bruts et traités pour les afficher.

# Exemple 6 : Calculer une transformée de Fourier rapide (FFT)

Dans cet exemple, un générateur de signaux envoie de façon régulière les échantillons à un objet de type Buffering qui va créer des buffers de données. La durée du buffer ainsi que le décalage entre les deux buffers consécutifs doivent être renseignés.

Chaque buffer est ensuite envoyé via un signal/slot Qt à un objet de type FFT qui va calculer la transformée de Fourier rapide.

Un objet de type MainWindowEx6 récupère les signaux temporels ainsi que l’amplitude de la transformée de Fourier afin de les afficher grâce à des objets de type TemporalSignalDisplay et BufferedSignalDisplay respectivement.

# Exemple 7 : Envoyer une chaîne de caractères à l'Arduino

Dans cet exemple, la communication va se faire dans la sens PC vers carte Arduino.

Comme pour l’exemple 1, un objet de type SerialPortManager est créé avec les mêmes caractéristiques. Une fois la connexion établie et le port série ouvert, la carte Arduino rentre dans la boucle loop() régulièrement et envoie des données sur le port série. La fonction serialEvent() est appelée entre deux appels à la fonction loop(). Elle permet de récupérer des informations en provenance du PC sur le port série. La chaîne de caractères inputString va contenir ce qui a été envoyé par le PC. Dans la fonction loop(), si cette chaine de caractère est égale à « Start », la LED interne de la carte Arduino est allumée, si elle est égale à « Stop », elle est éteinte grâce à la fonction digitalWrite.

Une interface de type MainWindowEx7 est créé. Elle comporte 2 QPushButton qui permettent d’envoyer soit la chaîne de caractères « Start » soit « Stop ».

# Exemple 8 : Gérer une base de données "patient" depuis une interface graphique

Il faut installer wamp. Attention il faut installer la même version que le compilateur de Qt (soit 32, soit 64bits).

Une fois installé, il faut compiler le plugin mysql qui n'est pas présent par défaut dans Qt.

Il faut ouvrir une invite de commande, puis taper:

cd %QTDIR%\qtbase\src\plugins\sqldrivers

qmake -- MYSQL\_INCDIR="<WAMP\_DIR>\bin\mysql\mysql5.7.28/include"

MYSQL\_LIBDIR="<WAMP\_DIR>\bin\mysql\mysql5.7.28/lib"

nmake sub-mysql

nmake install

Si vous n'utilisez pas le compilateur de Visual Studio, il faudra remplacer nmake par mingw32-make