RAPPORT DE SYNTHÈSE - GSA : Gestion de Stock Alimentaire

RIVIÈRE – ROIG – DYE

I. INTRODUCTION

En début de semestre, il nous a été imposer un projet en Fondamentaux de l'Architecture et Systèmes d'exploitation. Parmi un large panel de sujets, nous avons choisi celui qui demandait un objet d'une cuisine connectée.
Nous avons pensé à toutes sortes de choses, mais sommes rapidement arrivés à un gestionnaire de stock de légumes et de fruits.

Le but de ce projet est de concevoir une caméra connectée, reliée à un raspberry pi permettant de faire l'inventaire des fruits et légumes disponibles dans la cuisine de l'utilisateur (gestionnaire de stocks). L'utilisateur prend en photo l'aliment à enregistrer. Pour chaque victuaille, une caméra détecte la couleur dominante et propose une liste de fruits et légumes présentant cette couleur pour faciliter la recherche de la denrée à l'utilisateur. Cette interface se présente sur un petit écran tactile. L'utilisateur n'a plus qu'à choisir l'aliment correspondant parmis une liste de 5 aliments, ou à cliquer sur le boutton "Autres" pour voir la liste complète.

Le Raspberry enregistre l'aliment dans sa base de donnée et propose à l'utilisateur de consommer tel légume plutôt qu'un autre à un jour t donné en se basant sur les temps de péremption moyens de ceux-ci. L'utilisateur peut enfin récupérer ses aliments et les ranger dans son réfrigérateur.

On veillera également à permettre à l'utilisateur de retirer les aliments utilisés pour qu'il puisse tenir son stock à jour.

II. MODE D'EMPLOI

Le fonctionnement est on ne peut plus simple : l'utilisateur doit brancher la caméra sur un port Camera Serial Interface (CSI) du Raspberry - entre les ports HDMI et Ethernet. Il faut ensuite autoriser le Raspberry à utiliser la caméra dans le BIOS du nano-ordinateur.

Pour l'écran, il faut commencer par connecter l'alimentation. Pour cela les câbles noir et rouge doivent être branchés tel quels : Rouge=5V sur la pin 2 du Raspberry et Noir=GND sur la pin 6 du Raspberry. Il faut ensuite brancher la nappe aux connecteurs du Raspberry et de l'écran, en utilisant les ergots pour bloquer la nappe.
Il faut ensuite lancer l'application, en se plaçant dans le dossier GSA et en entrant la commande :

```
mkdir bin *
mkdir obj *
make
make start.
```

* : nécessaire car git ne stocke pas des dossiers vides. Or ceux-ci sont vides avant que l'on entre la commande make et n'existent donc pas.

Il existe d'autres commandes plus spécifique comme la mise à jour : `make maj`, ou encore la recompilation complète du projet `make forceMaj.`

Enfin, l'utilisateur peut compiler la documentation à l'aide de la commande `make doxygen` (pourvu qu'il ait doxygen installé)

III. MOYENS MATÉRIELS

Pour le fonctionnement de ce projet, nous avons eu besoin :

- D'un Raspberry Pi 2, Modèle B, 1GB RAM
  Pas vraiment de justification à donner, nous avons travaillé avec le Raspberry basique donné par les professeurs.
- D'une caméra basique pour Raspberry Pi (Camera board)
  Là encore, nous n'avons pas cherché la complication, il nous fallait de quoi prendre un photo de qualité suffisante pour pouvoir être analysée.
- D'un écran 7" tactile pour Raspberry Pi (7"" Touch screen)
  L'écran est l'élément qui nous permet d'organiser le lien entre l'utilisateur et la base de donnée, il affiche l'interface, montre les photos prises etc.

Il est à noter que, initialement, nous voulions connecter au Raspberry une balance (ou plus exactement un capteur de force) afin de pouvoir peser les aliments. Cette fonctionnalité a été abandonnée, par manque de capteur et de temps. De plus, il nous a été conseillé de nous concentrer sur la partie reconnaissance visuelle et cela nous a pris beaucoup de temps.

IV. MOYENS HUMAINS

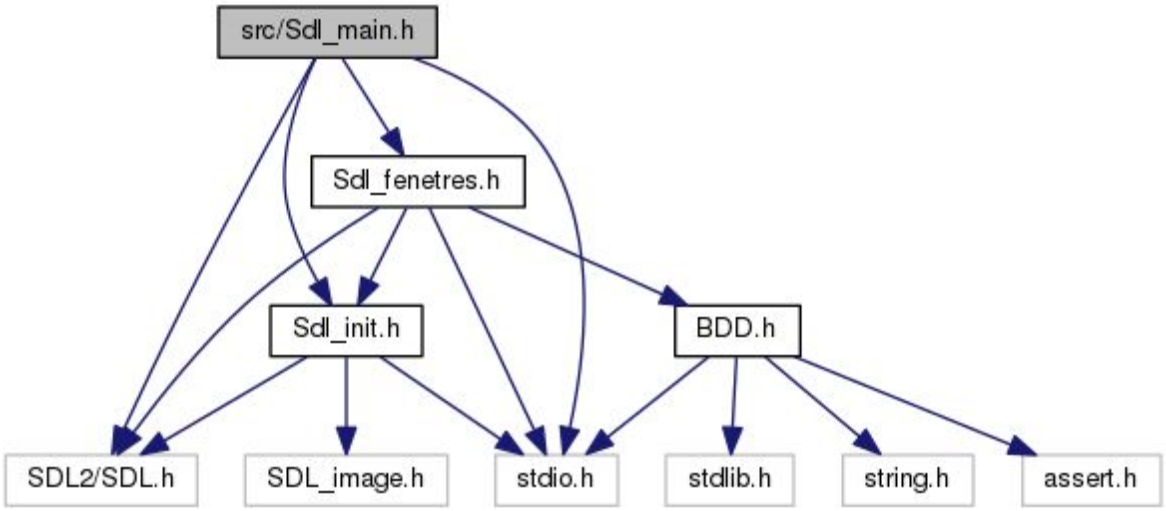
Au niveau des moyens humains dans ce projets, nous étions en trinôme : Clément ROIG, Tristan RIVIERE et Matthieu DYE.
Clément et Tristan avaient déjà l'habitude de travailler avec ce genre de technologies, alors que Matthieu beaucoup moins. Néanmoins, toute l'organisation du projet ainsi que sa documentation a été gérée par l'ensemble du trinôme.
Ainsi, Clément s'est surtout occupé de la partie base de données et de l'écriture des fonctions **back-end** (conception d'une base de données, reconnaissance photo, lecture d'image BMP).
De son côté, Tristan s'est occupé de toute la partie maintenance et outils du projet (installation des librairies, création d'un makefile et la Doxygen) ainsi que de l'interface utilisateur (**front-end**).
Enfin, Matthieu a aidé autant en front qu'en back-end, en fonction des demandes de Tristan et Clément.

V. ARCHITECTURE DU PROJET

- data : Dossier où sont stockées les données
- src : Dossier des fichier C et des en-têtes

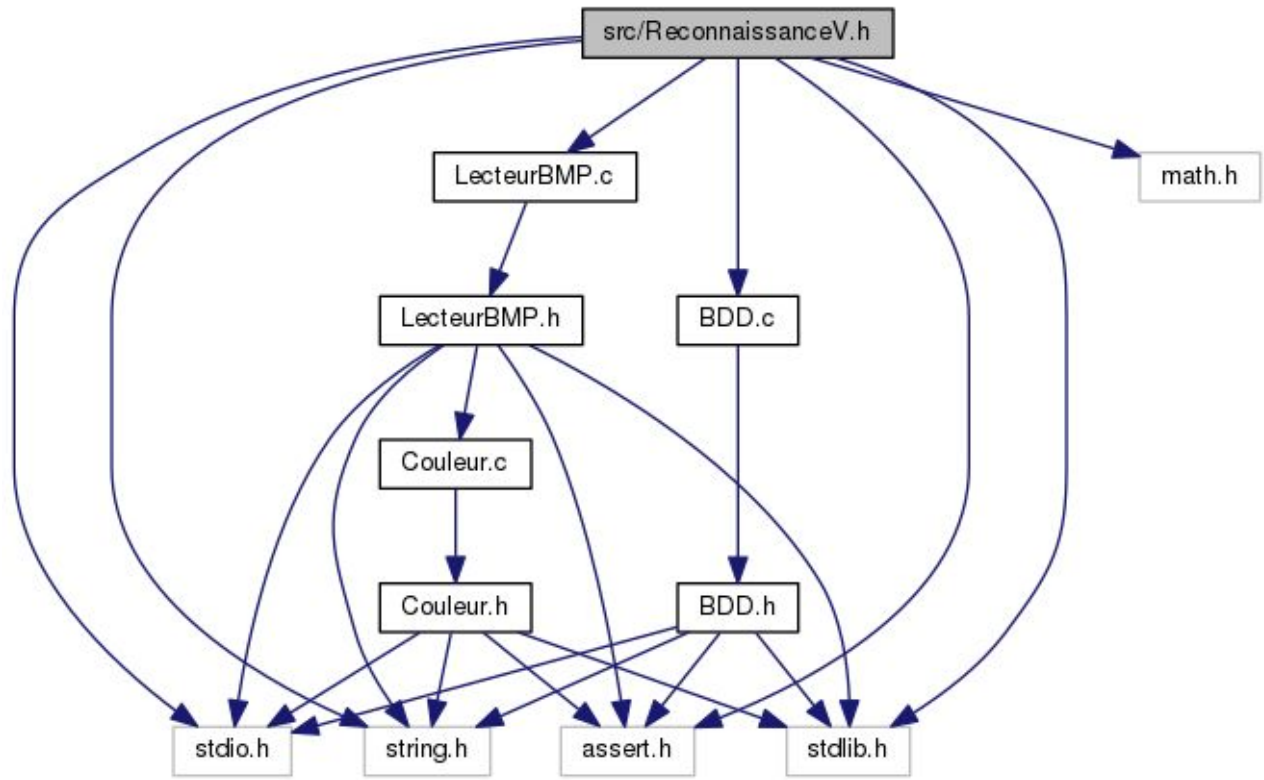
- obj : Dossier des fichier compilés .o
- bin : Dossier comportant l'exécutable du projet
- doc : Dossier comportant la documentation
- doc/html : Documentation doxygen (voir index.php)
-

Dépendances des fichiers



:

eV.h:

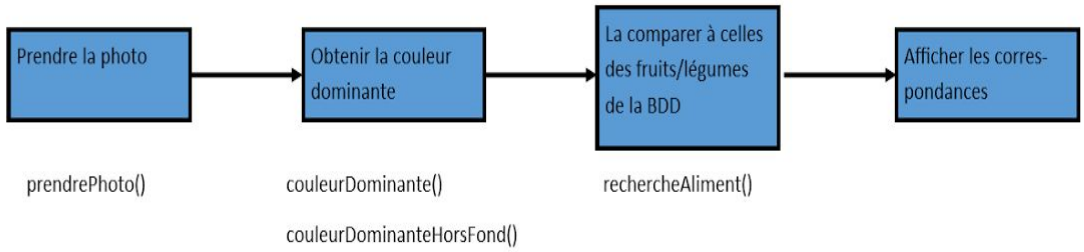


VI. CODE

Nous n'avons utilisé qu'une seule libraire (et ses différents modules) dans le but de gérer l'interface graphique : il s'agit de **SDL2**. Pour tout le reste, nous avons codé toutes les

fonctionnalités dont nous avions besoin : lecture d'une image au format BMP, base de données (basée sur le format CSV) et reconnaissance photo de couleur.
Pour simplifier l'inventaire de ces fonctions, plutôt que de toutes les noter ici, nous vous redirigeons vers la Doxygen que nous avons élaborée. Pour cela, il suffit de télécharger le dossier html dans le dossier doc et d'ouvrir le fichier index.html dans votre navigateur Internet.

Pour résumer, notre application fonctionne selon ce modèle très simple :



prendrePhoto() : la caméra prend une photo du fond dans un premier temps puis de l'aliment sur le fond.

couleurDominante() : en ayant codé l'image comme une image BMP, on peut récupérer les composantes R,G, et B (grâce à des fonction telles que getRCoul(), getGCoul(), getBCoul()), et en déduire une couleur dominante calculée comme une moyenne.

couleurDominanteHorsFond() : une fois la couleur du fond extrait de la première photo, cette fonction soustrait cette couleur à la deuxième photo comprenant le fond et l'aliment. On obtient ainsi la couleur de l'aliment (pour peu que le fond ne soit pas de la même couleur...).

getIdAlimentParCouleur() : on parcourt la BDD en comparant la couleur obtenue précédemment en paramètre à celles de la BDD. Différentes variables de "précision" permettent de paramétrer à partir de quelles valeurs une couleur est proche d'une autre.

VII. PERSPECTIVES

Notre système est aujourd'hui fonctionnel. Bien sûr, avec plus de temps, nous pourrions ajouter plus de fonctionnalités, comme par exemple de proposer des recettes qui utiliseraient les fruits/légumes présents dans la base de données ou ajouter ses propres aliments à la base de donnée via une interface graphique (actuellement il faut éditer un fichier texte).

Le prototype de notre système est néanmoins prêt à la commercialisation. Le marché sur lequel il se vendrait - la domotique informatisée - est en pleine expansion. Les "maisons intelligentes" se démocratisent de plus en plus et ce, dans de nombreux domaines (systèmes de régulation de température, de luminosité, pièces intelligentes qui s'adaptent en fonction de la personne qui l'occupe etc.).

Nous avons aujourd'hui fait notre premier pas dans ce domaine en présentant notre gestionnaire de stock à reconnaissance visuel. Au niveau du coût, pour produire notre projet, il faudra acheter les composants suivants : un écran à 80€, une caméra raspberry officiel à 30€,un raspberry (3 de préférence de part la masse de calcul de notre programme) à 40€, ce qui fait un total moyen de 150€ pour le matériel). Bien entendu, nous étions loin d'utiliser le Raspberry au maximum de ses capacités et il va de soit que le matériel de notre projet (Raspberry + caméra + écran tactile) pourrait être utilisée en parallèle par bien d'autres programmes chargés dans le nano-ordinateur.

VIII. Notes et remarques

Matthieu, clément et moi même sommes tous trois issues de formations bien différentes : Post paces, IUT et Peip (option informatique à Lyon). Nous avons donc tous les trois suivies des enseignement bien différents et qui se sont vues complémentaire lors de ce projet. La ou Clément avait suivi un cours sur la reconnaissance de couleur, et Matthieu à imaginer un produit, j'avais pour ma part suivi des cours de C standard 98 ou l'on nous à appris comment gérer des projets de large ampleurs en équipe (utilisation de doxygen, de git, ou encore d'outils de recherche de fuites mémoire). Ce projet fas aura donc vraiment été enrichissant pour chacun d'entre nous car il nous aura permis à tous les trois d'élargir nos connaissance dans un domaines que nous aimons tous trois.