Projet LabView

Programmation & Instrumentation Visuel

COFFRE- FORT

Réalisé par : Groupe TP 3

Encadré par : M .Leroy

Sommaire

Introduction

Dans le cadre du projet Labview du cours d'Instrumentation et Programmation visuelle, nous devions décider du sujet que notre projet allait s'orienter. Après plusieurs *Brainstorming* en petit groupes, plusieurs projets ont émergé. Nous avons décidé suite à un vote de faire *Coffre Fort avec une Reconnaissance Fiscale*. Pour pouvoir rendre un prototype viable dans un temps limité qui sont les séances de cours, nous avons défini les limites:

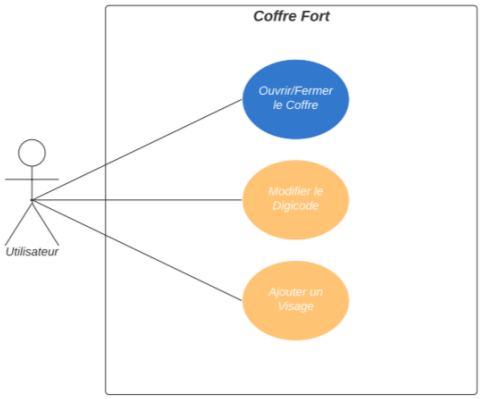
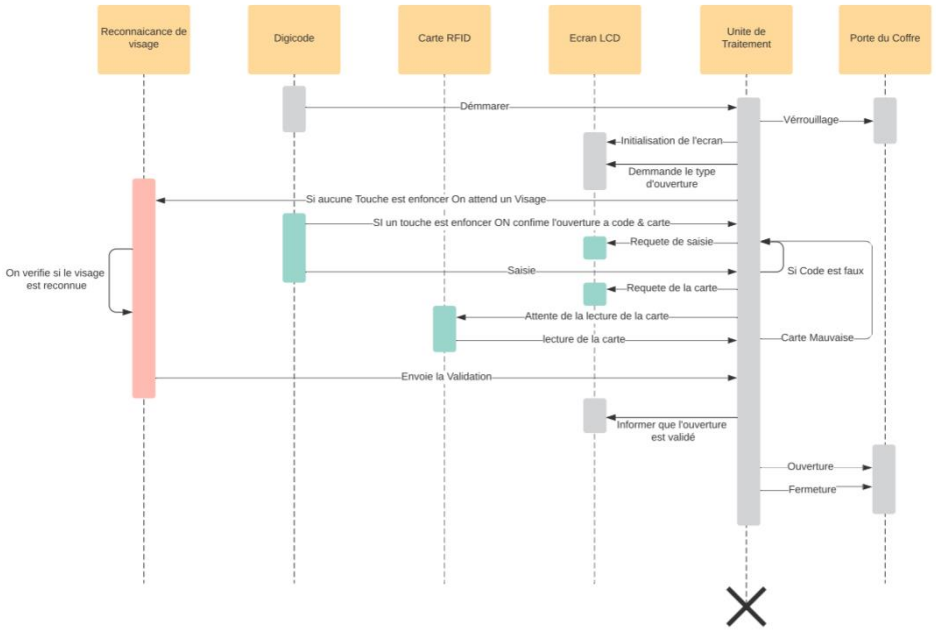


Diagramme de Cas d'utilisation

Schéma explicatifs de l'interaction des composants



Thématique du Projet

Notre projet *Coffre Fort* réunit des thématiques par la programmation des différents composants:

- Programmation d'une Raspberry PI sous LabView.
- Programmation événementielle.
- Variable local globale et partager.

Management des équipes

La répartition faite lors de ce projet a été fait en découpant notre coffre fort en composants, chaque sous-groupe devait s'occuper de l'une de ces parties :

Ecran

AZOUAY Mouad (Responsable d'équipe)
KFIFAT Abir
ZINSOU Françoise

Digicode

CAFFIAUX Matis (Responsable d'équipe & Chef de projet)
UYTTERSPROT Valentin
BONNET Quentin
LOUARD Abir

Lecteur RFID

HAOUDER Oussama (Responsable d'équipe)
KORIS Abdellah
SALHI Omar

Reconnaissance faciale

NGUYEN Martin (Responsable d'équipe)
BENALI Jamal
OUAHID Mohammed
BELMIR Aymane
BENBAZZA Mohammed

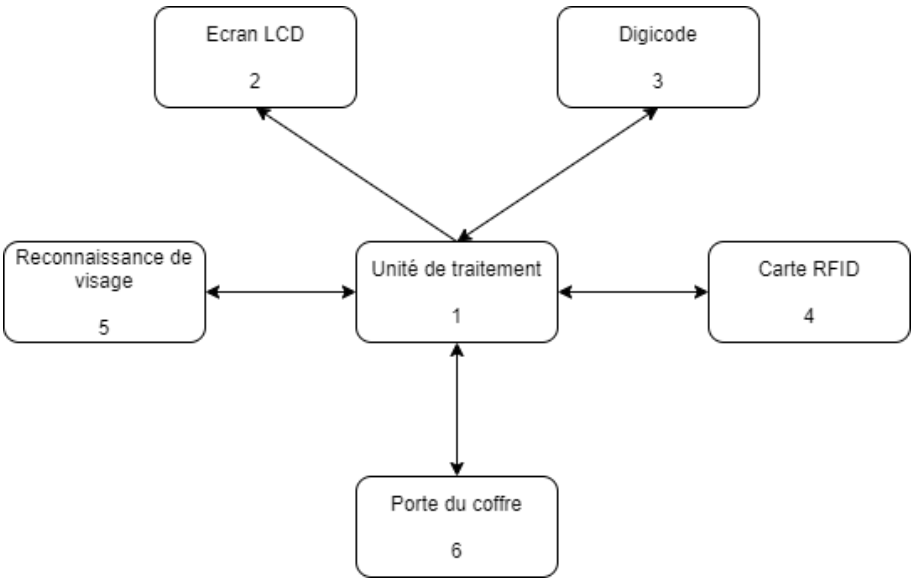
Unité de traitement

BALLO Prince (Responsable d'équipe)
MBOUP Mbayang
ADISSA Maria

Chaque responsable d'équipe gère son pôle et son composant de façon autonome, la seule condition qui est imposée sur le groupe et de suivre le cahier des charges fonctionnels.

Cahier des Charges

Schéma du système :



Nomenclature des Charges Fonctionnelles :

1	1.1 Verrouillage de vérification de la porte (**6**) et sortie du mode veille par la pression d'une touche du digicode (récupérer une Saisie simple du Digicode)

	1.2 Initialise l'écran à la sortie du mode veille par un message qui est envoyé à l'écran (Exemple : “Bonjour”)
	1.3 Si une Saisie renvoie un code “1”, l'unité de traitement lance la procédure de déverrouillage manuelle. (étape 1 : Code | étape 2 : Carte RFID)
	1.4 Si une Saisie renvoie un code “2”, l'unité de traitement lance la procédure de déverrouillage automatique. (Reconnaissance Faciale)
	1.5 Si une option de déverrouillage aboutit à un échec, on réexécute la procédure de déverrouillage à zéro (demande du choix du type d’ouverture) en avertissant l'utilisateur de l'échec d'authentification.
	1.6 Envoie des requêtes aux différents composants pour déclencher leurs utilisations. Les retours des composants doivent être logiques (True or False) pour **5** et **6**, et numériques pour **3** et **4** .
	1.7 Avertir l’utilisateur de toutes les étapes pertinentes durant le cycle de vie.
2	2.1 Affiche correctement la chaîne de caractères envoyée par l'unité de traitement (**1**)
	2.2 S’allume et s'éteint quand l’unité de traitement (**1**) envoie un code spécifique (à déterminer entre les groupes du composant **1** et **2**)
3	3.1 Entrée et sortie du mode veille lors de l’envoie d’un résultat ou de la réception d’une commande de l’unité de traitement (**1**)
	3.2 Choix du type de saisie : Saisie unitaire (un seul caractère sans manipulation possible) ou multiple (chaîne de caractères de taille 6 avec manipulation cf. **3.3** et **3.4**), gérée par une des commandes d’entrées de l’unité de traitement (**1**)
	3.3 Avoir une touche de suppression de caractère sur une saisie multiple
	3.4 Avoir une touche d’annulation sur une saisie multiple
4	4.1 Démarre la lecture de la carte quand l’unité de traitement (**1**) envoie une commande (à déterminer le type de commande)
	4.2 Vérifier que l’acquisition des données de la carte est complète
	4.3 Renvoie les données utiles de la carte à l’unité de traitement (**1**) (cf **1.1**)
5	5.1 Scanner un visage à l’aide d’un dispositif photographique
	5.2 Comparer le visage scanné aux classes de(s) personne(s) autorisée(s)
	5.3 Envoie le résultat de la comparaison du visage scanné à l’unité de

	traitement (**1**) avec un booléen (cf **1.1**)
	5.4 S'éteint quand la tâche est effectuée
6	6.1 Se verrouille à la sortie de la veille (cf **1.1**)
	6.2 S'ouvre ou se ferme sur une demande de l'unité de traitement
7 Système	7.1 Économiser l'énergie électrique au maximum en n'alimentant pas les connections électriques des composants à l'exception du digicode (**3**) et de l'unité de traitement (**1**)

Composants

Ecran

I- Introduction

Dans cette partie, on a créé au début une interface utilisateur sur LabView qui interagit avec les choix de l'utilisateur.
Quand l'utilisateur choisit une option, l'unité de contrôle le traite son choix et renvoie des résultats sous forme de code qui seront traduit en des messages affichés sur l'interface.

- Entrée:

Entier: qui contient le code du message d'erreur à afficher.

- Traitement:

On crée un fichier texte contenant les différents codes qui sont mappés avec des messages. Lorsque l'on reçoit, après envoi de l'unité de traitement, un code en entrée, on vérifie s' il existe dans notre fichier et on affiche le message correspondant.

- Sortie:

Une chaîne de caractère qui va être affichée à l'écran.

```
100     1-Entrez votre mot de passe :
110     Mot de passe valide
200     2-Placez votre visage
210     Detection OK
300     3-Passez votre carte
310     Carte valide
401     Mot de passe invalide
402     Carte invalide
403     Erreur détection
505     Bienvenue
```

Figure 1 : fichier code - message

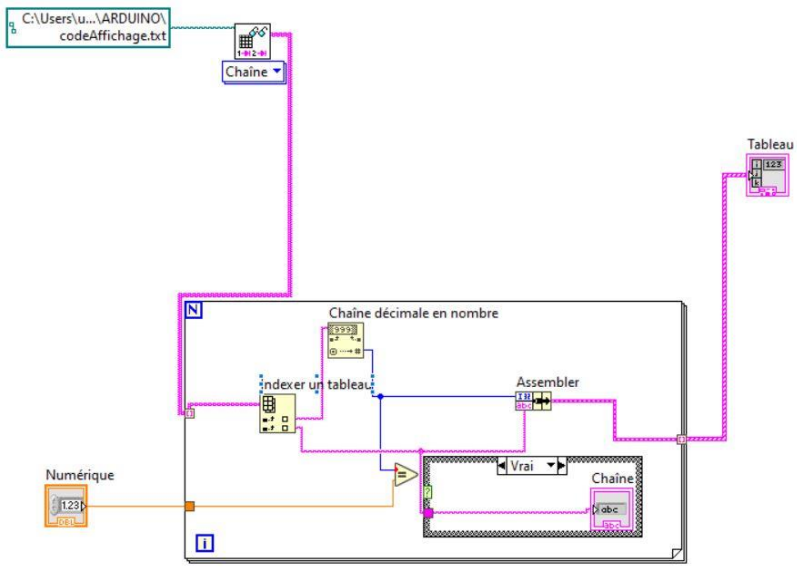


Figure 2 : vi de recherche du code sur un fichier

Au début, il existe un fichier contenant différent code, chaque code correspond à un message. Un input de type control lit le code saisi par l'utilisateur après dans une boucle "while" on balaye le fichier ligne par ligne à la recherche du message approprié au code saisie .

II- Partie Interfaces :

Figure 3: interface d'accueil

Au début la page d'accueil affiche deux options pour s'authentifier lorsque l'utilisateur clique sur un bouton, il sera directement dirigé vers la page approprié à la méthode d'authentification choisie.

Figure 4 : interface d'option mot de passe

III- Problème rencontres :

Une difficulté concernant la dynamisation des interfaces au moment de la gestion des événements générés par l'utilisateur (clic sur un bouton et affichage d'une autre page) . Problème d'intégration de l'interface avec l'unité de traitement.

IV- Bibliographie:

https://www.ni.com/fr-fr/support/documentation/supplemental/08/building-labview-user-interfaces.html
https://www.youtube.com/watch?v=fSsarSxeQRw&ab_channel=Sixclear

Digicode

I- Introduction

Avec notre équipe nous étions en charge de l'élaboration du digicode. Pour cela nous devions respecter les différents points de notre cahier des charges qui sont :

- Entrée et sortie du mode veille lors de l'envoie d'un résultat ou de la réception d'une commande de l'unité de traitement.
- Choix du type de saisie : Saisie unitaire (un seul caractère sans manipulation possible) ou multiple (chaîne de caractères de taille 6 avec manipulation), gérée par une des commandes d'entrées de l'unité de traitement.
- Avoir une touche de suppression de caractère sur une saisie multiple.
- Avoir une touche d'annulation sur une saisie multiple.

L'objectif de notre travail sera de produire un VI complet de notre digicode, que nous allons ensuite mettre en commun avec les autres équipes de notre projet. A la fin nous pouvons assembler les VIs que les différents groupes auront réalisés afin d'obtenir le coffre-fort final.

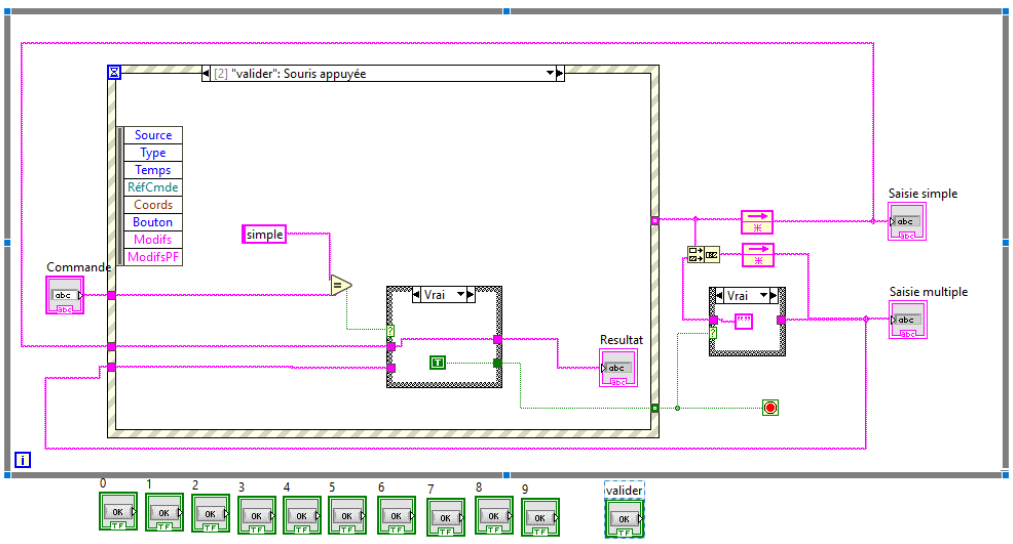


Figure 1 - Fonctionnement général de la VI

Le cœur de notre programme est la boucle événementielle qui nous permet de définir les actions à l'appui de nos différents boutons. Nous avons deux boucles, une pour l'appui des boutons chiffres et une dernière plus complète pour l'appui du bouton valider. Celle pour l'appui des chiffres est très simple, elle renvoie juste la valeur du bouton saisi. La valeur est traitée de différente façon à la sortie suivant la saisie qui est demandée : simple ou multiple. Nous expliquerons ces deux types de saisie dans la suite. Pour choisir le type de saisie, nous avons une entrée appelée qui commande qui dirige le système vers une saisie simple si la commande d'entrée est "simple" sinon ce sera une saisie multiple. Quelque soit la saisie, on renvoie en entrée la valeur de la saisie et lorsque l'on appuie sur le bouton "valider", on renvoie le résultat qui prend la valeur de la saisie demandée en entrée, puis on arrête la saisie. Et voici la face avant de notre VI :

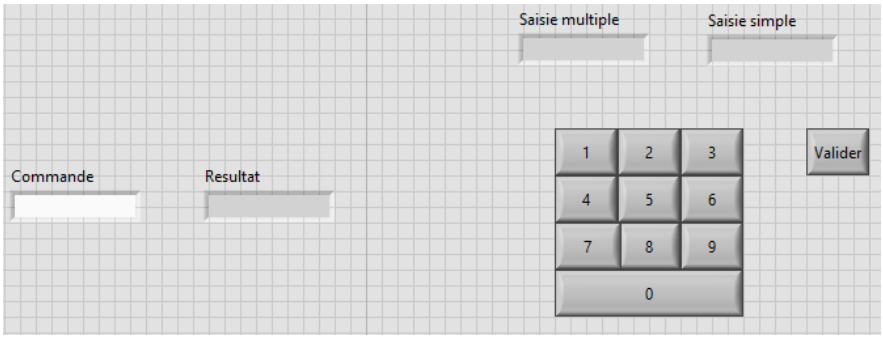


Figure 2 - Face Avant Labview

II - Saisie Unitaire

Dans cette partie l'objectif sera de limiter la saisie de l'utilisateur sur le digicode à un seul caractère sans donner la possibilité de faire des manipulations (suppression de caractères ou annulation) qui ne seront disponible que pour la saisie multiple.

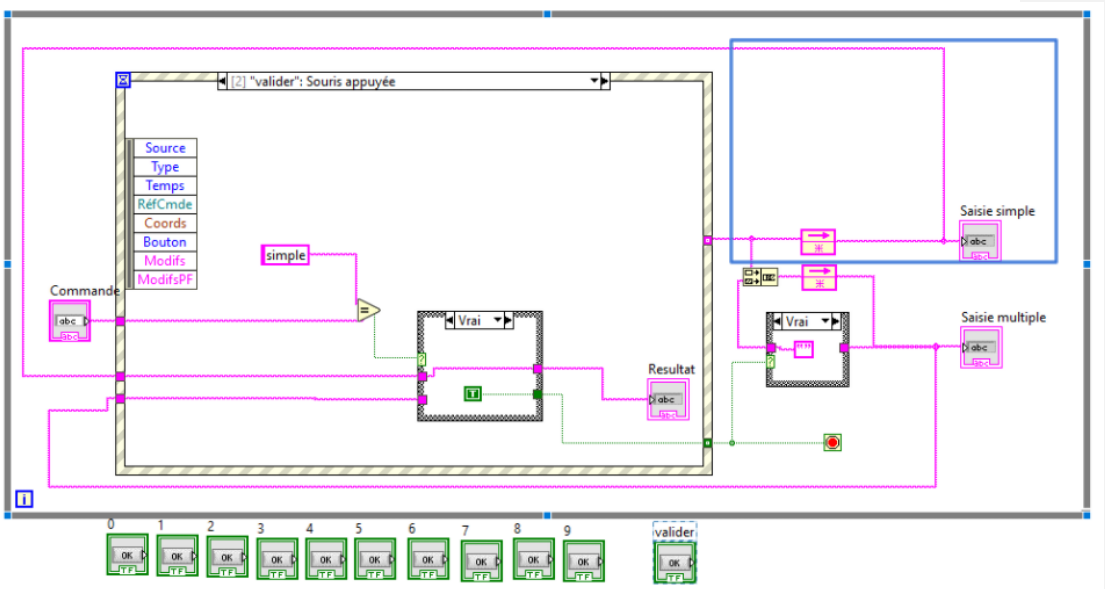


Figure 1 - Fonctionnement général de la VI

Pour faire une saisie simple, il faut que “simple” soit passer en commande. Ensuite on saisit la valeur voulue. Si on a saisi une valeur et que l’on ne veut pas envoyer celle-ci, il suffit d’appuyer sur un autre bouton qui changera la valeur. Et quand on a la valeur voulue, on appuie sur le bouton valider qui envoie la valeur dans résultat et qui arrête la saisie.

III - Saisie Multiple

Le but de ce digicode, est de pouvoir donner des ordres avec une saisie simple, mais aussi de pouvoir saisir un code de 6 chiffres.
Tout d’abord, si on veut une saisie simple on doit envoyer “simple” en entrée au digicode pour qu’il ne renvoie qu’une seule valeur, et donc si on veut une saisie multiple on envoie en commande autre chose que “simple” pour que la VI comprenne qu’elle doit renvoyer une saisie multiple.
Ensuite, on doit trouver un moyen de stocker les saisies multiples. Pour cela, il faut stocker l’ensemble des 6 chiffres dans un buffer (ci-dessous dans cadre bleu).

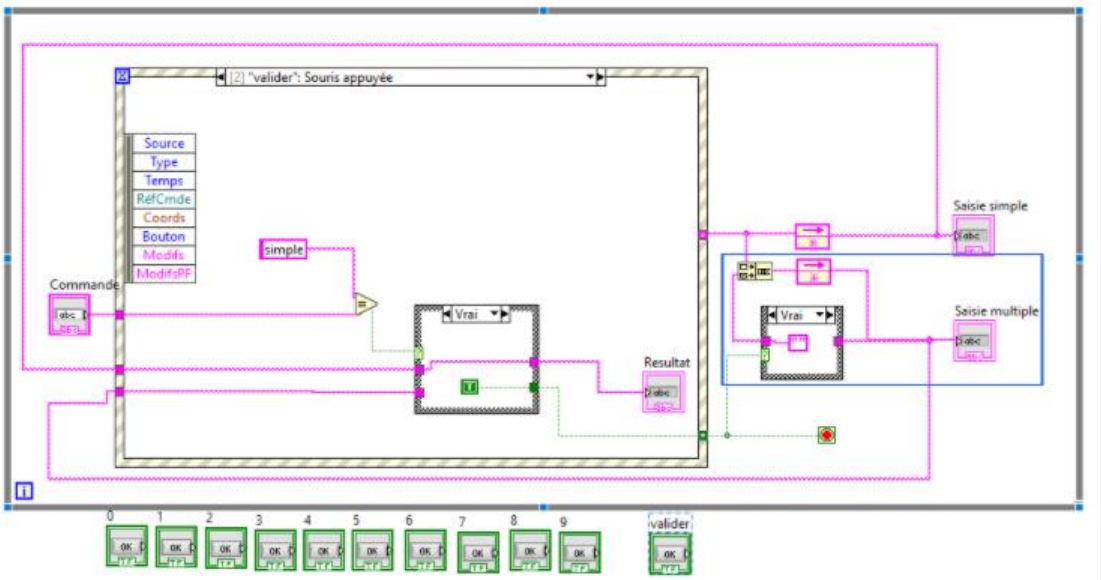


Figure 1 - Fonctionnement général de la VI

Comme on le voit à la sortie de la boucle événementielle, on part vers un buffer qui, dans une entrée, la deuxième, prend le chiffre saisie, et pour garder les saisies précédentes, on fait revenir dans la première entrée du buffer la valeur stockée dans la saisie multiple. La petite boucle condition permet de réinitialiser la saisie multiple lorsque l'arrêt du programme est activé sinon on continue à renvoyer la valeur dans saisie multiple dans le buffer. On renvoie dans la boucle événementielle la valeur de saisie multiple, et lorsque le bouton valider est activé on retourne la valeur dans résultat et on arrête la saisie.

IV- Conclusion

En guise de conclusion, nous avons pu respecter le cahier des charges en mettant une entrée et sortie du mode veille lors de l'envoie d'un résultat ou de la réception d'une commande de l'unité de traitement. Nous avons aussi pu réaliser la saisie unitaire ainsi que la saisie multiple qui sont gérées par une des commandes d'entrées de l'unité de traitement. Et finalement nous avons créé deux touches, une pour la suppression et l'autre pour l'annulation.

Malgré le manque de temps et la complexité du coffre fort nous avons pu réaliser notre composant.

VI- Bibliographie

Dans nos débuts de recherche, nous nous sommes renseigné sur les différentes manières d'aborder le codage de notre digicode sous Labview. Tout d'abord, nous pensions que nous devions passer par Raspberry pi ou Arduino, donc nous avons consulté plusieurs sites. Ensuite, nous nous sommes dit que nous restions uniquement sur Labview. On a commencé à regarder ce qui se faisait sur Labview en termes de digicode. Nous avons trouvé différentes choses mais toutes des plus complexes et avec le temps qui nous était accordé nous devions aller au plus simple. Nous avons ensuite trouvé une vidéo expliquant certaines choses de manière simple. Nous nous sommes donc inspirés de cela, et nous avons continué seuls.

- https://www.youtube.com/watch?v=f7ciRDdkUi4&t=231s

Lecteur RFID

I- Introduction :

Dans le cadre de notre projet dans Instrumentation et programmation visuelle, on avait comme mission de réaliser un outil qui nous permet de faire lire une carte spécifique pour ouvrir un coffre. Nous avons donc décidé d'utiliser un Raspberry Pi qui est un microcontrôleur qui va nous permettre de reconnaître une Carte NFC à travers un lecteur NFC 1. Pour réaliser cela il faut prendre un matériel spécifique pour pouvoir réaliser le montage et pour ensuite programmer faire un programme dans notre Raspberry Pi pour que le système fonctionne

NFC 1 : ou Near Field Communication, est une technologie permettant d'échanger des données entre un lecteur et n'importe quel terminal mobile compatible ou entre les terminaux eux-mêmes. C'est la technologie qu'utilise votre carte bancaire pour le paiement sans contact, ou votre carte de transport. L'avantage de cette technologie est qu'en principe, aucune application n'est requise. Il suffit de rapprocher les deux supports. Attention, il ne faut pas que ces derniers soient trop éloignés l'un de l'autre : une dizaine de centimètres maximum !

II- Étapes

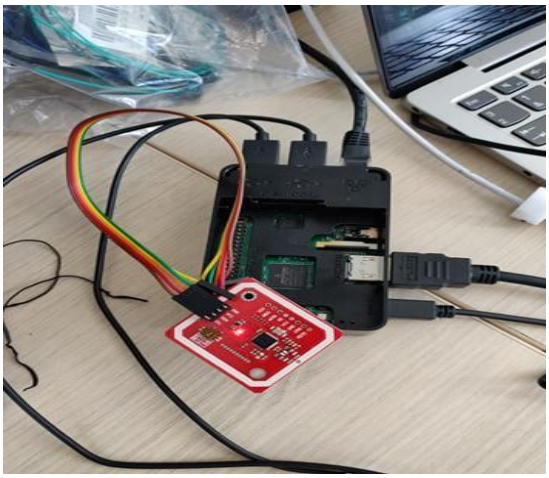
Dans les deux premières semaines, nous n'avons pas pu faire grand-chose car il n'y avait pas de matériel donc nous avons cherché sur internet et donc tout ce que nous avons fait a été théorique.

1. Première étape :

Nous avons commencé par réaliser le montage nécessaire pour travailler. Dans le début nous avons trouvé difficulté, mais après avoir fait de la recherche nous avons connecter tous les pins que nous avons du lecteur NFC avec la Raspberry Pi selon le tableau suivant

```
# NFC module pin -> Pi GPIO physical pin #
GND -> 6
VCC -> 4
SDA -> 3
SCL -> 5
```

Finalement, nous avons obtenu le montage suivant :



2. Deuxième étape :

Lors de notre première connexion avec la Raspberry Pi nous avons réalisé qu'elle n'était pas connectée à internet. Nous avons donc essayé dans un premier temps de faire la connexion

avec le réseau Wifi, sauf que nous n'avons pas pu la connecter donc nous avons décidé de le faire connecter directement avec un câble Ethernet.

Après être connecter à internet nous avons installé plusieurs bibliothèques pour que la Raspberry Pi puisse lire le lecteur NFC, ensuite nous avons fait des tests pour savoir si le lecteur est bien lisible pour la Raspberry Pi à travers la commande suivante :

```
root@raspberrypi:~# i2cdetect -y 1
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- 24 -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
```

3. Troisième étape :

Ensuite nous avons entamé le codage, nous avons implémenté un code qui nous permet de lire la carte NFC spécifique que l'on a avec le lecteur NFC qui est connecté avec le Raspberry Pi. (Chaque carte NFC contient un code unique avec lequel elle peut être identifié)

Après cette étape nous avons fait la partie montage qui était prête. Lorsqu'on pose notre carte NFC sur le lecteur le code renvoie vrai, et dans le cas contraire il renvoie faux.

4. Quatrième étape :

Cette partie consistait à faire intégrer notre code dans LabVIEW, chose que nous n'avons pas fait à cause plusieurs problèmes rencontrés :

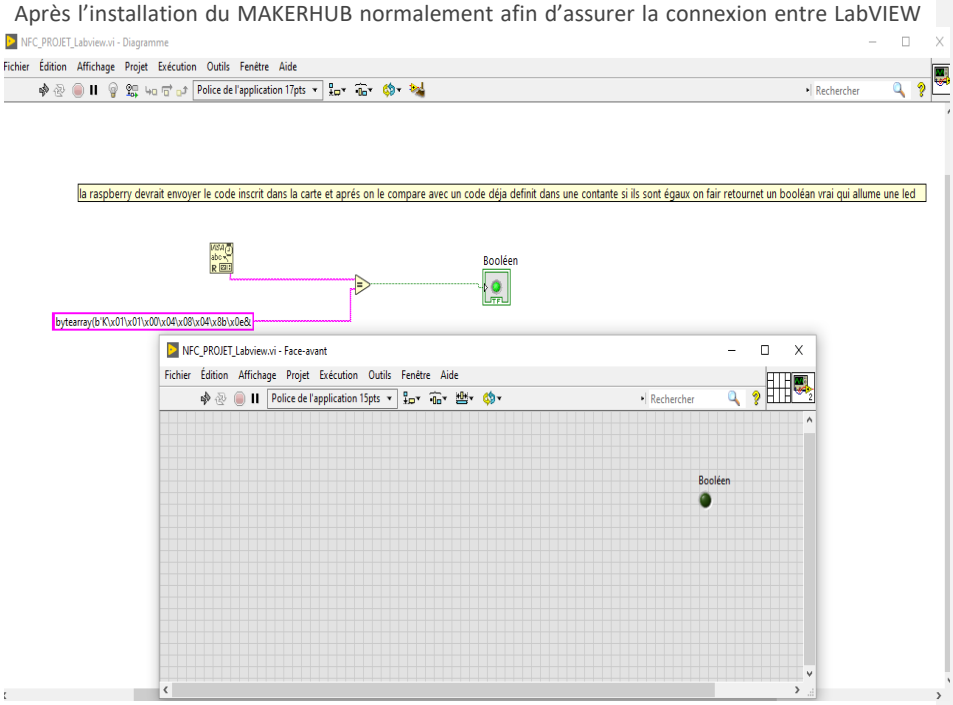
a- Problème n°1 :

Nous n'avons pas pu télécharger une extension (MAKERHUB1) nécessaire sur le pc de l'école pour faire connecter LabVIEW avec la carte Raspberry, mais grâce à l'aide du staff de l'école nous avons réussi à installer le logiciel

MAKERHUB1 : c'est une extension qui lie LabVIEW avec notre projet Raspberry

b- Problème n°2 :

Après l'installation du MAKERHUB normalement afin d'assurer la connexion entre LabVIEW



on devait entrer l'adresse IP du raspberry et le mot passe mais malheureusement encore une fois Labview ne détectait pas la Raspberry . on a essayé pas mal de fois, par plusieurs manières , mais sans résultat

Finalement nous avons fait quand même un schéma Labview :

La dans ce schéma y a trois composants importants, composant visa qui est responsable de prendre en paramètre l'identifiant de la carte RFID, ensuite une constante qui contient la valeur de l'identifiant et comparateur qui compare les deux valeurs et si il sont égaux il renvoie un Boolean True qui allume la LED

III- Bibliographie :

https://les-enovateurs.com/rfid-raspberry-pi-code/

https://www.raspberry-pi.fr

https://forums.ni.com/t5/Instrument-Control-GPIB-Serial/How-to-make-labview-to-receive-information-from-an-rfid-reader/td-p/1459334?profile.language=en

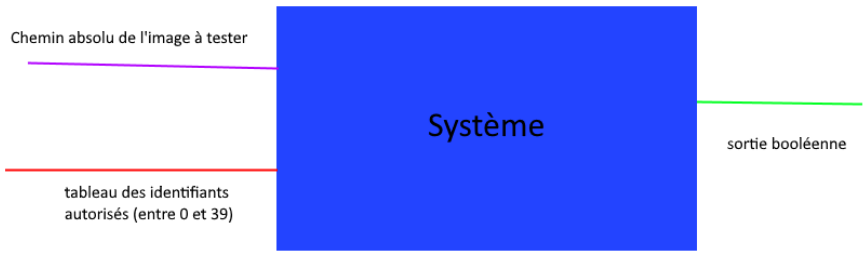
https://forums.ni.com/t5/Instrument-Control-GPIB-Serial/How-to-make-labview-to-receive-information-from-an-rfid-reader/td-p/1459334?profile.language=en

https://raspberry-pi.fr/rfid-raspberry-pi/

https://fr.wikipedia.org/wiki/Raspberry_Pi

Reconnaissance faciale

I- Introduction: Le principe global de notre système



Figure(1) Boîte noire du système

Comme schématisé ci-dessus, le système prend en entrée deux entrées, le chemin absolu de l'image à tester et le tableau des identifiants autorisés. Le tableau contient des éléments de type int de 0 à 39 car notre base d'images contient 40 personnes. Ensuite, le système renvoie true si l'image entrée correspond bien à une personne dont l'identifiant est autorisé, false sinon. L'interface du système se fera sous labview et le code qui implémente le système sera rédigé en python.

II- Étude de ce qui se fait aujourd'hui:

Tous les géants du Web ont pris pour habitude de publier leurs découvertes théoriques dans le secteur de l'intelligence artificielle et de la reconnaissance d'image pour faire avancer au plus vite l'état de l'art.

Les derniers résultats des tests (Biometric Technology Rally) conduits en mars 2018, pilotés par le Ministère de l'Intérieur américain, ainsi que ceux du NIST (Institut National des normes et de la technologie des Etats Unis) de 2018 et 2019 donnent aussi une excellente image des meilleures technologies de reconnaissance faciale disponibles sur le marché.

Voyons cela.

- L'algorithme GaussianFace développé dès 2014 par des chercheurs de l'université de Hong Kong affichait des scores d'identification faciale de 98,52% contre 97,53% pour les humains. Et ce malgré des progrès à réaliser quant à la capacité de mémoire requise et au temps de calcul.

- En 2014 toujours, Facebook annonçait le lancement de son programme baptisé DeepFace, capable de déterminer si deux visages photographiés appartiennent à la même personne, avec une précision de 97,25%. Soumis au même test, les humains répondent correctement dans 97,53% des cas, soit à peine 0,28% de mieux que le programme de Facebook.

- En juin 2015, Google renchérit avec FaceNet, nouveau système de reconnaissance faciale aux scores jamais égalés: 99,63% % de précision au test de référence Labeled Facebooks in The Wild ; 95% sur la base YouTube Faces DB. En utilisant un réseau neuronal artificiel et un nouvel algorithme, la firme de Mountain View est parvenue à rapprocher un visage et son propriétaire à la quasi-perfection. Une technologie intégrée dans Google Photos pour trier les clichés et les tagger automatiquement en fonction des personnes reconnues et, preuve de sa pertinence, devenue rapidement disponible en ligne dans une version open-source officieuse, OpenFace.

- En mai 2018, Ars Technica a annoncé qu'Amazon faisait déjà activement la promotion de son service de reconnaissance faciale basé sur le cloud, appelé Recognition, auprès des forces de l'ordre. La solution peut reconnaître jusqu'à 100 personnes dans une seule image et peut effectuer des comparaisons avec des bases

de données contenant des dizaines de millions de visages. Et, malgré les critiques, Amazon continue à faire évoluer son logiciel pour y intégrer la reconnaissance des émotions comme la peur, la joie, la surprise, la tristesse...selon France 24 du 15 août 2019.

- Fin mai 2018, la Direction Science et Technologie du Ministère de l'Intérieur des Etats-Unis (Homeland Security Science and Technology Directorate) a publié les résultats des tests sur son site du Maryland en mars. Ces tests en environnement réel ont mesuré les performances de douze logiciels de reconnaissance faciale dans un couloir de 2m sur 2,5m. La solution de reconnaissance faciale de Thales (LFIS) a révélé toutes ses performances avec un taux d'acquisition de 99,44% d'un visage en moins de 5 secondes (pour une moyenne de 68%) et un taux réel d'identification en moins de 5 secondes (Vendor True Identification Rate) de 98% pour une moyenne de 66% et un taux d'erreur de 1% contre 32% pour la moyenne.

Au final, on s'appuiera sur le projet ci-dessous afin d'écrire le code

- [2]: Face Recognition on Olivetti Dataset
  - https://www.kaggle.com/serkanpeldek/face-recognition-on-olivetti-dataset

L'article ci-dessous nous a permis de comprendre certains principes utilisés :

- [3]: Analyse en Composantes Principales
  - https://chambreuil.com/public/education/3.2/stat/projet
  - https://medium.com/apprentice-journal/pca-application-in-machine-learning-4827c07a61db

III- Explication et description du projet:

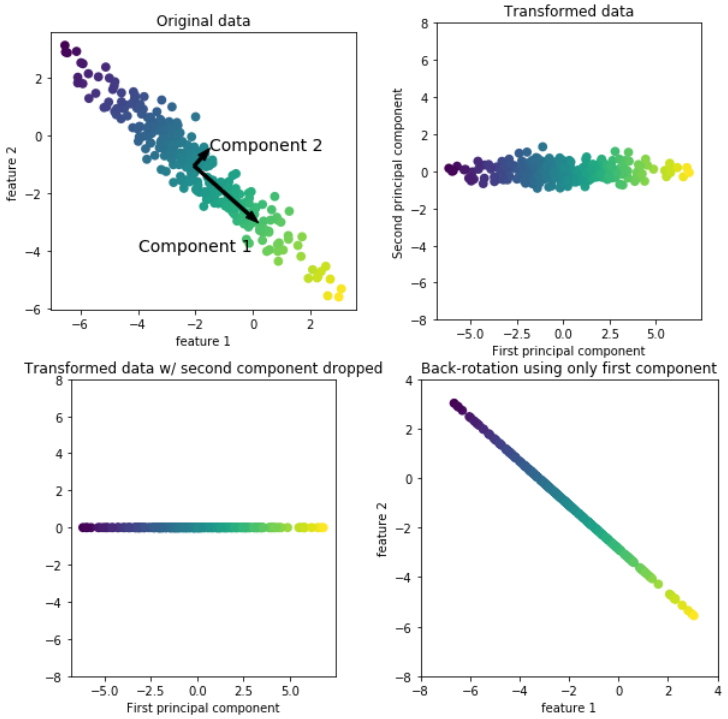
Explication de la méthode ACP (Analyse en Composantes Principales)

1. Son but:

Le but principal de cette méthode est de réduire la dimension effective des données et les visualiser.

2. Son fonctionnement:

Il consiste à transformer des variables liées entre elles en nouvelles variables décorrélées les unes des autres nommées composantes principales, ou axes principaux. Cela permet au praticien de réduire le nombre de variables et de rendre l'information moins redondante en ne sélectionnant que les composantes importantes.



Figure(2) Ensemble de données synthétiques bidimensionnelles

3. Ses défauts par rapport aux autres méthodes de reconnaissance de visage:

En comparaison avec la méthode LBP, cette dernière est plus efficace que la ACP mais son opérateur fondamental est incapable de saisir certaines caractéristiques dominantes. En outre, la méthode ACP n'est pas optimisée pour la séparabilité (discrimination) de classe tandis que l'analyse discriminante linéaire ADL tient compte de ceci.

4. Ses points forts et ses points faibles:

Sur le plan mathématique, l'ACP est donc une méthode simple à mettre en œuvre. En outre, l'ACP permet d'appréhender une grande partie de ses résultats d'un simple coup d'œil grâce aux graphiques qu'elle fournit. De plus, cette méthode est très souple, puisqu'elle s'applique sur un ensemble de données de contenu et de taille quelconque, pour peu qu'il s'agisse de données quantitatives organisées sous forme individus/variables.

Cependant, l'ACP nécessite que les images de visage soient mises sous formes de vecteurs, ce qui a pour effet de détruire la structure géométrique de l'image.

5. Pourquoi va-t-on l'utiliser ?

On a opté pour la technique ACP car elle est rapide, simple et populaire dans l'identification de modèle. De plus, l'étude théorique de l'algorithme ACP est très pédagogique et permet d'acquérir de solides bases pour la reconnaissance du visage.

IV- Explication du déroulement du code Python:

1. Diviser le code sous forme d'étapes:

Dans ce projet, on a importé des images de visages à partir de la base de données 'Olivetti'. On dispose alors de 10 images différentes de 40 personnes distinctes. Ces images ont été prises à des moments, des éclairages, des expressions et détails du visage différents. On associe à chaque personne un ID.

Figure(3) Exemple des images de la base de données

Premier pas :

```
data = np.load("/content/drive/MyDrive/labview/olivetti_faces.npy")
target = np.load("/content/drive/MyDrive/labview/olivetti_faces_target.npy")
```

Figure(4) Chargement de la data

La première variable 'data' contient une base de données des images des visages, tandis que la deuxième variable 'target' contient les indices ID de chaque visage.

```
from PIL import Image
import numpy as np

var = 0
for i in data:
  rescaled = (255.0 / i.max() * (i - i.min())).astype(np.uint8)
  im = Image.fromarray(rescaled)
  im.save('/content/drive/MyDrive/labview/data'+str(var+1)+'.png')
  var+=1
```

```
import imageio
from scipy import misc

length = len(data)
data = []
for i in range(0,length):
  image = imageio.imread('/content/drive/MyDrive/labview/data'+str(i+1)+'.png')
  data.append(np.array(image))

data = np.array(data)
```

Figure(5) conversion de la base de données .npy vers .png et enregistrement dans un dossier

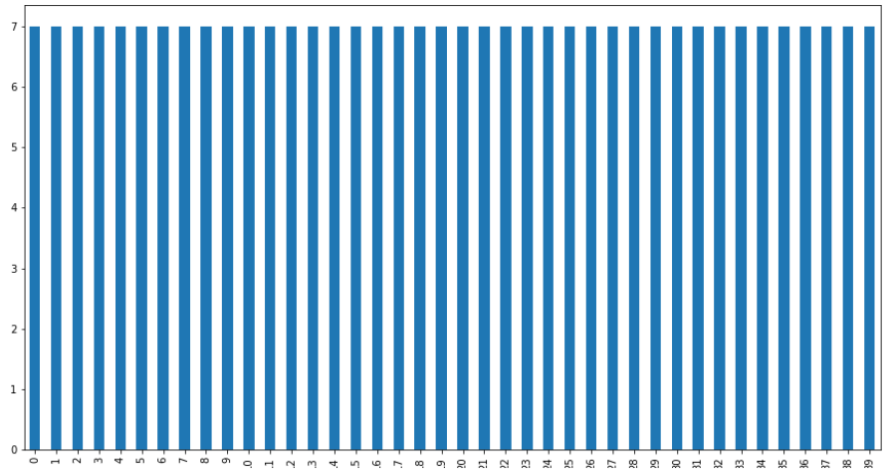
Ici après l'importation des données, afin de pouvoir choisir une image spécifique de la personne comme entrée (pour Labview) on a généré un dossier contenant toutes les images des visages en format .png.

```
[ ] X_train, X_test, y_train, y_test =  train_test_split(X, target, test_size=0.3, stratify=target, random_state=0)
    print("X_train shape:",X_train.shape)
    print("y_train shape:{}".format(y_train.shape))

    X_train shape: (280, 4096)
    y_train shape:(280,)
```

Figure(6) division de la data en deux datas

Pour entraîner et tester notre modèle on a divisé et mélangé la data (70% pour Lx_train et 30% pour x_test), les 70% sont réservés à l'apprentissage du modèle et les 30% restants sont pour les tests et la validation du modèle.



Figure(7) data d'entraînement : 40 personnes, 7 images par personne

```
from sklearn.decomposition import PCA
pca=PCA(n_components=2)
pca.fit(X)
X_pca=pca.transform(X)

n_components=90
pca=PCA(n_components=n_components, whiten=True)
pca.fit(X_train)

X_train_pca=pca.transform(X_train)
X_test_pca=pca.transform(X_test)
```

Figure(8) application de la méthode PCA

Application de la méthode PCA sur notre data avant de l'envoyer à notre modèle SVM:

```
[ ] clf = sklearn.svm.SVC()
    clf.fit(X_train_pca, y_train)
    y_pred = clf.predict(X_test_pca)
    print("accuracy score:{:.2f}".format(metrics.accuracy_score(y_test, y_pred)))
```

Figure(9) Modèle SVM

Création du modèle SVM importé depuis Scikit-Learn et apprentissage avec la data mise en forme par la méthode PCA, on obtient le ratio de réussite en comparant les étiquettes réelles et les étiquettes prédites.

```
accuracy score:0.93
```

Figure(10) Ratio de réussite

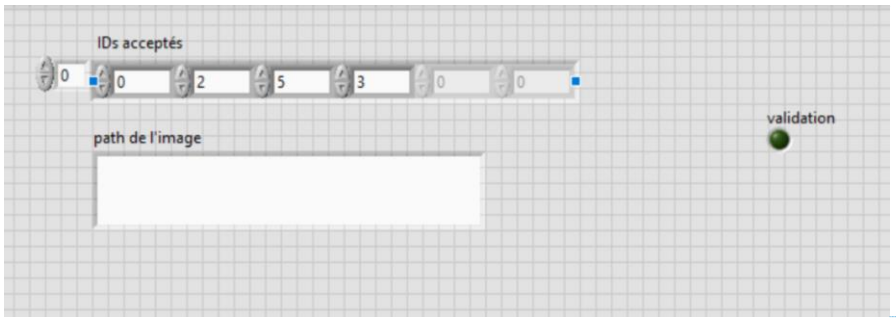
```
[ ]  def isAllowed(path,list):
       retrn=0
       image = imageio.imread(path)
       image = np.array(image)
       image=image.reshape(1,image.shape[0]*image.shape[0])
       image_pca=pca.transform(image)
       y_pred = clf.predict(image_pca)
       if(y_pred in list):
         retrn = 1
       return retrn
```

Figure(11) Fonction isAllowed qui link python et labview

La création de la fonction principale qui sera appelée par labview prend comme paramètres le path de l'image à vérifier et un tableau contenant les indices ID des personnes qui ont le droit d'accès.

V- L'intégration au projet principal:

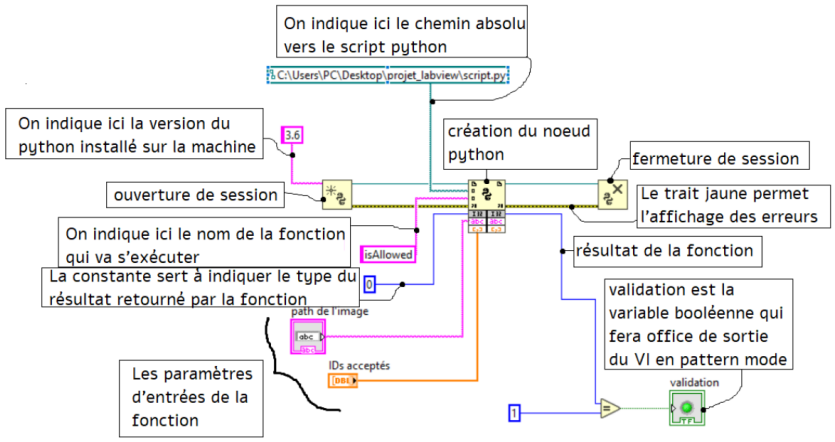
1. Intégration sous Labview:



Figure(12) Face avant du fichier labview

Il faut préciser que cette interface est une interface intermédiaire, dans le projet final, l'utilisateur n'aura pas accès au tableau des identifiants acceptés sinon ça serait absurde. Mais elle nous sert à nous, à tester le bon fonctionnement du code python et de son intégration à labview. Le champ path de l'image permet à l'utilisateur d'entrer le path de l'image à tester. Dans un projet optimal, ce champ aurait été remplacé par un bouton qui aurait allumé la webcam afin de prendre instantanément la photo qui aurait été l'image à tester.

Commenté [1]: Il faut aussi ajouter les photos des tests



Figure(13) Face arrière du fichier labview

Un détail à retenir c'est que labview ne supporte que les versions 2.7 et 3.6 de python.

Python version specifies the version of Python in which the Python session runs. This function supports Python of version 2.7 and 3.6 only

Figure(14) Aperçu de l'aide du composant pythonNode

2. Intégration de notre partie à l'unité de traitement:

Nous ne sommes pas parvenus à intégrer notre module à l'unité centrale car l'unité centrale n'est pas parvenu à intégrer le digicode, on ne peut donc pas voir le déroulement total d'une use case où l'utilisateur entrerait une saisie au digicode puis déverouillerait le système avec notre module de reconnaissance faciale.

VI- Rétrospective sur le projet:

1. Difficultés rencontrées dans le projet:

- La première difficulté rencontrée a été de trouver une base d'images propre et assez dense. Propre étant des images déjà en noir et blanc, des prises de vue identiques. Dense car lors de l'apprentissage on élimine 70% des images pour l'apprentissage et il ne reste que 30% des images pour les tests donc plus il y a d'images, plus l'apprentissage est efficace.

- Une autre difficulté rencontrée a été d'installer la bonne version de python et tous les modules nécessaires au bon fonctionnement du script python, étant donné que les versions de python acceptables sont la 3.6 et 2.7 qui ne sont pas récentes.

- une difficulté de transformation des images .npy vers des images .png pour faciliter l'intégration du script python dans labview ainsi une autre re-transformation pour filer les images dans notre modèle c.à.d une format numpy array .

2. Remarques et observations du chef d'équipe:

- Le projet s'est vraiment bien déroulé pour ma part, j'ai remarqué que l'équipe est plus productive lorsqu'elle est subdivisée en binômes et il est important de garder un œil sur l'avancement, de savoir si les directives n'ont pas été comprises de travers par exemple. Nous sommes 5 dans le groupe et nous nous sommes répartis en 3 sous-groupes, 2 binômes et moi seul. Un binôme s'occupait du code en python, l'autre binôme s'occupait de faire le rapport et moi j'avais pour tâche de faire le lien entre python et labview ainsi qu'avec l'unité centrale, à la toute fin. Même si le projet n'a pas abouti sur un programme fonctionnel, du point de vue global, chacun a pu apporter quelque chose au projet et apprendre une nouvelle chose, moi par exemple j'ai appris à intégrer un code python à labview, ce qui est plutôt intéressant à savoir. Je n'aurais qu'une seule remarque sur le management global, on aurait peut-être dû organiser les groupes en groupe de compétences et non d'affinités pour gagner en efficacité.

3. Perspectives d'amélioration pour le projet:

- Nous sommes 16 dans notre groupe projet, nous aurions pu créer notre propre base d'images, avec 10 prises de vue différentes cela aurait donné 160 images, ce qui est léger mais acceptable.

- Dans la même continuité, on aurait pu créer sous labview un accès à la webcam de l'ordinateur et exploiter l'image prise comme image de test et ainsi se rapprocher encore plus d'un système réel.

Unité de traitement

I- Introduction :

- Entrée:

Entier: qui contient soit l'option choisie, soit le code du digicode dans le cas où on choisit l'option digicode.

Booléen: Un pour vérifier si l'authentification RFID a bien réussie. Et un deuxième pour vérifier l'authentification faciale.

- Traitement:

Au tout début, le système est en mode veille.

- Sortie:

Un entier qui contient le code correspond à l'affichage demandé.

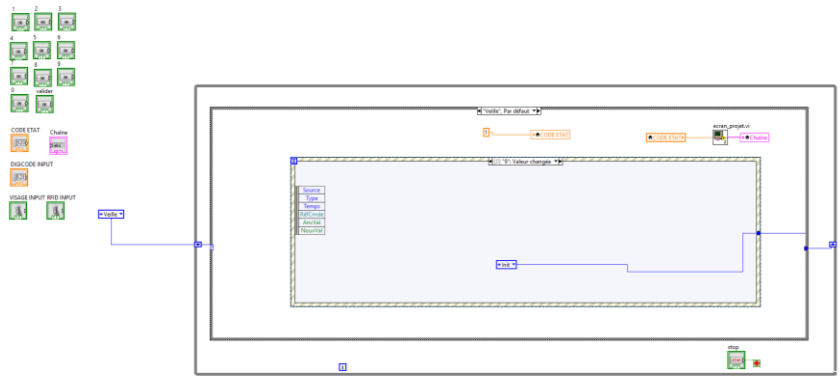


Figure 1 : Mode veille

A la première saisie de l'utilisateur, le système s'active et affiche le menu à l'écran. Les 3 options qui s'afficheront seront:

1. Digicode

L'utilisateur tape sur 1, on attend la saisie du code. Ensuite on vérifie si le code est bon, si oui on affiche à l'écran "porte ouverte", sinon on affiche un message d'erreur.

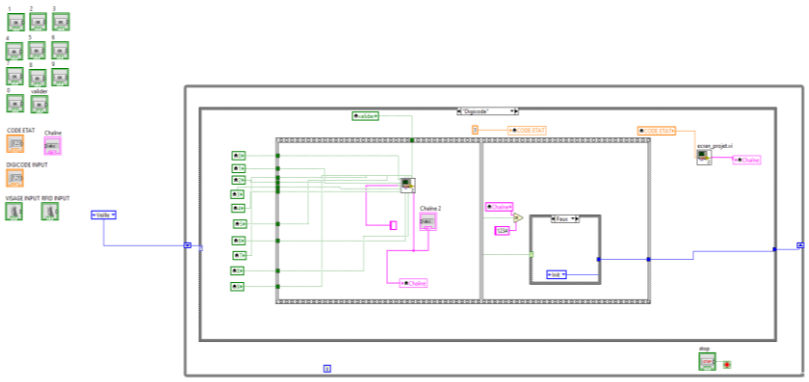


Figure 2 : Mode digicode

2. Carte RFID

L'utilisateur tape sur 2, on affiche "attente RFID" à l'écran, et on vérifie le booléen qui provient du sous VI qui traite la partie RFID, si true, on affiche à l'écran "porte ouverte", si false on affiche le message d'erreur qui correspond.

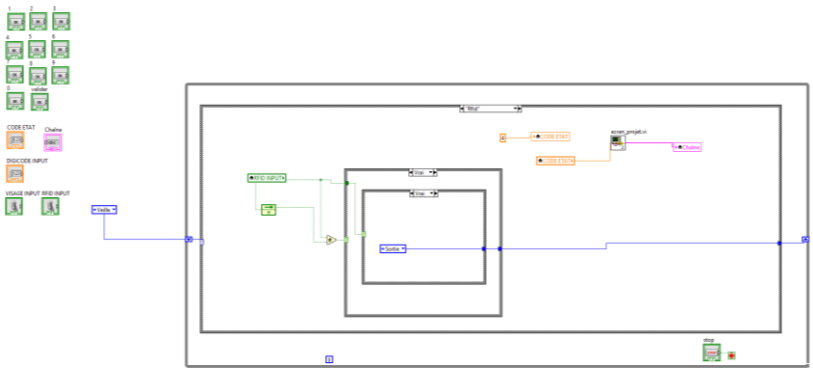


Figure 3: Mode Rfid

3. Reconnaissance faciale

L'utilisateur tape sur 3, on affiche "attente reconnaissance faciale" à l'écran, et on vérifie le booléen qui provient du sous VI qui traite la partie reconnaissance faciale, si true, on affiche à l'écran porte ouverte, si false on affiche le message d'erreur qui correspond.

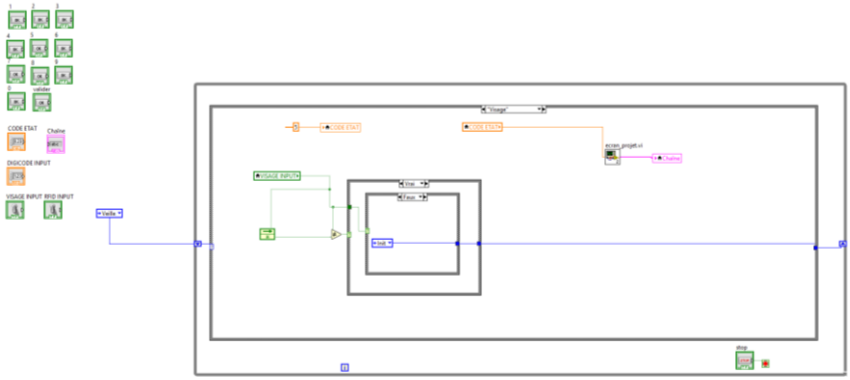


Figure 4: Mode reconnaissance faciale

II- Problème rencontrés:

Pour le sous VI qui fait l'unité de traitement, nous avons eu un problème de synchronisation de données. C'est-à-dire, on a eu un retard de la réponse du VI par rapport aux entrées. Le problème a été résolu en créant des structures à événements.

Le deuxième problème était au niveau de l'intégration du sous VI du digicode avec l'unité de traitement. En effet, les touches références sur le sous VI du digicode ne correspondent pas à la même référence des touches présentes sur la face avant de l'unité de traitement.

Conclusion

Ce projet nous à permis de nous apercevoir réellement des possibilités de programmation avec l'outil Labview. La force de LabView est de sa multi-compatibilité Software et Hardware. De plus, nous avons appris que la collaboration au sein d'une grande équipe requiert un certain travail en amont pour permettre de ne pas diverger de notre but et de régulièrement dialoguer entre les têtes d'équipes pour discuter des problématiques communes afin de ré-aiguiller le projet. Nous n'avons pas eu le temps d'avoir un prototype final avec tous les composants implémentés, malgré ceci chaque composants est fonctionnel de façon unitaire. Le temps nécessaire pour finir à été sûrement emputer par fonctionnalité de LabView que l'on a pas vu en cours et que l'on a dû apprendre lors du projet.

Remerciement

Nous tenons à remercier dans un premier temps, toute l'équipe pédagogique au sein de l'EIL Côte d'Opale qui assure la continuité des études en utilisant de différentes plateformes.

Nous sommes également reconnaissants à notre professeur Monsieur LEROY qui n'a pas cessé de nous aider et de nous donner des conseils concernant les missions évoquées dans ce rapport, qu'il nous a apporté lors des différents suivis et la confiance qu'il nous a témoignée.