



ECOLE SUPERIEURE POLYTECHNIQUE D'ANTSIRANANA

MENTION GENIE ELECTRIQUE

Parcours STIC

Sciences et Technologie de l'Information et de la Communication

MEMEOIRE DE FIN D'ETUDES EN VUE DE L'OBTENTION DU DIPLOME DE LICENCE

CONTRIBUTION A LA REALISATION D'UN SMART DISTRIBUTEUR DE BOISSON

Par :

IANJARISON Jean Dulan

RAKOTONIRINA Haja Elson

Encadreurs :

Docteur ANDRIANAJAINA Todizara

Docteur RAZAFIMAHEFA Tsivalalaina David

ANNEE UNIVERSITAIRE 2019-2020

REMERCIEMENTS

Avant tout, nous tenons à remercier Dieu tout puissant de nous avoir donné la force de pouvoir tenir sans avoir abandonné durant ce travail. Nous remercions aussi Monsieur Razafimahefa Tsivalalaina David et Monsieur Andrianajaina Todizara de leur disponibilité de nous avoir aidé et donné des conseils sur le projet.

Je tiens aussi à remercier tous les membres du Jury qui ont porté leurs jugements constructifs au travail, ainsi que leur remarque.

Nous profitons de cette occasion pour exprimer aussi nos vifs remerciements à tous les enseignants de l'Ecole Supérieure Polytechnique d'Antsiranana qui nous ont prodigués leurs connaissances. Sans oublier aussi nos amis qui nous ont donné des coups de main au transport des matériels.

Une profonde gratitude à toute nos familles et surtout nos parents pour leur soutien financiers et moral, pour leur encouragement et leur bénédiction ; enfin toutes personnes qui nous ont apporté leur aide durant notre travail : une grande merci à vous tous !

CAHIER DE CHARGES

Titre : « Contribution à la réalisation d'un Smart distributeur de boisson »

Contexte

L'idée est de réaliser un robot capable de distribuer un bon nombre de liquides différents que l'on peut sélectionner depuis un écran tactile (tablette, téléphone ou un écran autonome). Contrôlé par un Raspberry Pi. Conçu avec un principe de projet évolutif, le dispositif existe déjà. Le rajout d'autres fonctionnalités est indispensable pour améliorer le système.

Objectifs

Ce travail consiste à rajouter les différentes fonctionnalités suivantes :

- Distribution de boisson chaude
- Distribution de sucre
- Système d'aide aux malvoyants
- Interface de communication sonore

Travaux demandés

Etude et simulation du circuit de commande

Réalisation du circuit de commande

Amélioration du boîtier et l'ergonomie

Ajout des différentes fonctionnalités de distribution

Encadreurs

ANDRIANAJAINA Todizara

RAZAFIMAHEFA Tsivalalaina David

RESUME

TABLE DES MATERES

Remerciements.....	i
CAHIER DE CHARGES	ii
RESUME	iii
TABLE DES MATERES.....	iv
LISTE DES FIGURES.....	vii
Introduction générale.....	1
Chapitre 1. Généralité et les matériels à connaitre	2
1.1. Introduction.....	2
1.2. Le smart distributeur de boisson en général	2
1.3. Le Raspberry Pi.....	3
1.3.1. Définition.....	3
1.3.2. Les versions.....	3
1.3.3. L'alimentation.....	3
1.3.4. Les GPIO.....	4
1.4. Le module relais.....	4
1.4.1. Connexion au secteur	5
1.4.2. Câblage des broches	5
1.5. L'électrovanne.....	6
1.5.1. Définition.....	6
1.5.2. Les différents types d'électrovannes.....	6
1.5.3. Les avantages	7
1.6. L'électroaimant	7
1.6.1. Fonctionnement :	7
1.6.2. Les avantages :	8
1.7. Bouilloire	8
1.7.1. Définition :	8

1.7.2. Fonctionnement :	9
1.7.3. Avantage :	9
1.8. Bouton poussoir :	9
1.8.1. Définition :	9
1.8.2. Fonctionnement :	9
1.8.3. Avantage	10
1.9. Conclusion	10
Chapitre 2. Le circuit de commande et la conception du programme informatique	11
2.1. Introduction	11
2.2. Circuit de commande	11
2.2.1. Etude et simulation du circuit de commande :	11
2.2.2. Réalisation du circuit de commande	12
2.3. Comment installer un système d'exploitation dans un Raspberry PI :	13
2.4. Intégration du programme dans le Raspberry pi :	14
2.4.1. Installation de la bibliothèque manquante	14
2.4.2. Lancement du programme dans le Raspberry PI	14
2.4.3. Mise en place de l'écran tactile TFT :	15
2.4.4. Installation de haut-parleur	16
2.4.5. Augmentation du son :	16
2.5. Qu'est-ce-que kivy ? :	16
2.5.1. Historique :	17
2.5.2. Les fonctionnalités disponibles :	17
2.5.3. Le langage Kv (Kivy Language) :	17
2.5.4. Avantage de kivy :	17
• Rapide	17
• Flexibilité	18
• Frais	18

2.5.5. Création d'une interface graphique.....	18
2.5.6. Composant graphique :	20
2.5.7. Widget.....	21
2.5.8. Gestionnaire de mise en page.....	22
2.5.9. Le langage KV	24
2.6. Extraits des codes importants.....	25
Chapitre 3. Les différentes fonctionnalités ajouter de la distribution et Amélioration du boitier	
28	
3.1. Introduction.....	28
3.2. Les fonctionnalités ajouter	28
3.2.1. Les boissons chaudes	28
3.2.2. Système d'aide aux malvoyant et interface de communication sonore.....	28
3.3. Fonctionnement.....	29
3.4. Amélioration du boitier	30
3.4.1. Représentation du boitier en différente vue	30
3.4.2. Réalisation du distributeur	34
Conclusion générale	35

LISTE DES FIGURES

Figure 1:PROGRAMME DE TRAITEMENT	2
Figure 2:un Raspberry Pi	3
Figure 3:GPIO (General-purpose input/output)	4
Figure 4:un module relais a deux canaux.....	4
Figure 5:symbole d'un relais	5
Figure 6:Description des prises et des broches d'un module relais.....	5
Figure 7: Electrovanne à deux voies	6
Figure 8:Une électrovanne multivoie	6
Figure 9:Un électro-aimant de manœuvre simple.....	7
Figure 10:schéma du noyau sorti et rentré	8
Figure 11:bouilloire	8
Figure 12:bouton poussoir	9
Figure 13:Schéma d'une capture d'écran du chronogramme.....	10
Figure 14:schéma d'un circuit à l'intérieur d'un module relais	11
Figure 15:schema de simulation sur Proteus dont le Logic State est à l'état 0.....	12
Figure 16:schema de simulation sur Proteus dont le Logic State est à l'état 1.....	12
Figure 17:Fenêtre d'accueil du logiciel Raspberry Pi imager.....	13
Figure 18:fenêtre pour sélectionner l'image du système d'exploitation.....	14
Figure 19:Emplacement du module TFT sur un Raspberry PI	15
Figure 20:port d'entrer d'une prise Jack	16
Figure 21:prise Jack mâle	16
Figure 22>HelloWord en Kivy	19
Figure 23:schéma d'imbrication des widgets	20
Figure 24:Capture d'écran d'une fenêtre utilisant BoxLayout.....	21
Figure 25:Fenêtre contenant des Button contenus dans un GridLayout.....	23
Figure 26: Fenêtre contenant des Button contenus dans un FloatLayout	24
Figure 27: code qui affiche touche moi	25
Figure 28: code qui affiche le menu principal	26
Figure 29: code qui affiche les boissons chaudes.....	26
Figure 30: code qui affiche le taux du sucre.....	27
Figure 31: capture du premier vu sur l'écran	29
Figure 32: capture du menu principal	29

Figure 33:.....	30
Figure 34: Vue de face à présent	30
Figure 35 : Vue de face avant	30
Figure 36 : vue de derrière avant	31
Figure 37:vue de derrière à présent	31
Figure 38:Vue de dessus avant du boîtier	32
Figure 39:Vue de dessus à présent	32
Figure 41: Vue de droite à présent	Erreur ! Signet non défini.
Figure 40: Vue de gauche avant.....	Erreur ! Signet non défini.
Figure 42: Vue de gauche avant.....	42
Figure 43: Vue de gauche à présent.....	Erreur ! Signet non défini.

INTRODUCTION GENERALE

Les chercheurs cherchent à simplifier la vie humaine en évitant de faire trop d'efforts qui pourront être fatiguant. Tous les jours, des nouvelles opinions apparaissent dans le but de rendre la perfection voulue. Maintenant si on veut rendre celle-ci un peu plus facile et un plus parfait évitant de trop travail en distribuant des variétés de types de boissons. Donc là, on veut parler d'un distributeur de boissons disons automatiques par l'intermédiaire d'un SMART. Pas besoin de remplir le verre de chaque individu manuellement. Que ce dispositif existe déjà. Mais nous avons une idée de le rajouté d'autre fonctionnalités indispensables pour améliorer le système.

Donc ce fait, pour qu'on peut rester sur le cadre d'amélioration, on va parler d'abord dans le premier chapitre, la généralité sur le projet et des études des matériels utile à la réalisation. En suit, dans le deuxième chapitre, on va discuter des configurations et de la conception du programme informatique qui commande la distribution. Et pour finir, on va voir l'amélioration du boitier, les différentes fonctionnalités ajouter de la distribution et sa réalisation.

CHAPITRE 1. GENERALITE ET LES MATERIELS A CONNAITRE

1.1. INTRODUCTION

Dans ce premier partir, on va voir de la manier dont on va comprendre le projet en présent le schéma bloc. Donc on sera obligé de discuter des matériels et des composantes à utiliser en faisant des descriptions de chacun eux, suivie leurs avantages particuliers dans ce travail.

1.2. LE SMART DISTRIBUTEUR DE BOISSON EN GENERAL

Ce projet comporte à distribue de sucre et des différents types de boissons chaude. Sans oublie aussi le système d'aide aux malvoyants. Le contexte insiste d'utiliser un Raspberry pi pour contrôle le dispositif. Donc on peut intégrer un écran et des bouton poussoirs pour que l'utilisateur faire sa commande. C'est l'électrovanne qui assure le contrôle des liquide et l'électroaimant pour les poudres. On sépare le circuit de commande et de puissance pour alimenter l'électrovanne et l'électroaimant. Alors on aura besoin de plusieurs relais car on utilisera cinq électrovannes et un électro-aimant. Donc en voici un schéma qui résume le fonctionnement du système.

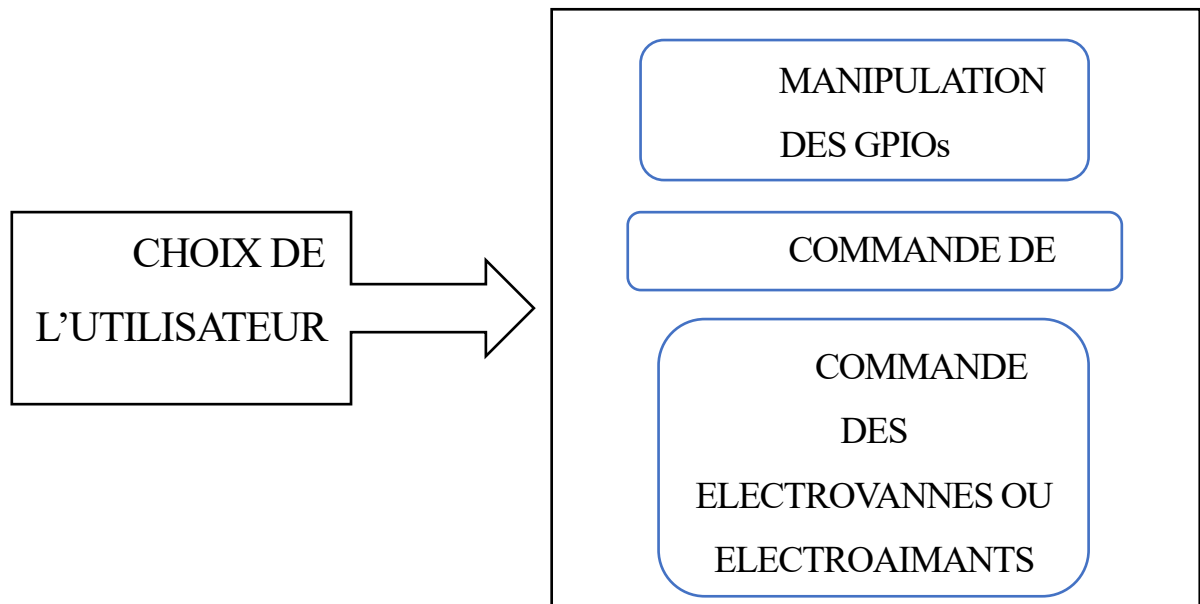


Figure 1:PROGRAMME DE TRAITEMENT

1.3. LE RASPBERRY PI

1.3.1. Définition

Le Raspberry Pi est un nano-ordinateur monocarte à processeur ARM conçu par des professeurs du département informatique de l'université de Cambridge dans le cadre de la fondation Raspberry Pi.



Figure 2: un Raspberry Pi

Cet ordinateur, de la taille d'une carte de crédit, est destiné à encourager l'apprentissage de la programmation informatique. Il permet l'exécution de plusieurs variantes de systèmes d'exploitation libre GNU/Linux, notamment Debian, et des logiciels compatibles. Mais Il fonctionne aussi avec l'OS Microsoft Windows, Windows 10 IoT Core et celui de Google, Android Pi. Il est fourni nu, c'est-à-dire la carte mère seule, sans boîtier ni alimentation, ni clavier, ni souris et ni écran dans l'objectif diminuer le coût et de permettre l'utilisation de matériel de récupération.

1.3.2. Les versions

Depuis 2012, le Raspberry Pi s'est décliné en plusieurs versions mais actuellement on est à la version 4. Et même cette version est déjà disponible sur le marché, les versions antérieures continuent toujours d'être dans le bon marché car ils sont moins chers par rapport aux versions récentes.

1.3.3. L'alimentation

Le Raspberry Pi utilise officiellement une alimentation électrique USB micro 5,1 V. Le courant réel dépendra de ce que vous faites. Normalement, toute alimentation électrique réputée de 2.5 A ou l'alimentation officielle pour le Raspberry Pi sera suffisante.

1.3.4. Les GPIO



Figure 3:GPIO (General-purpose input/output)

On les trouve sur le bord du Raspberry Pi et ils ressemblent à deux rangées de pines en métal. A partir d'un GPIO, on peut connecter un matériel comme une LED et un capteur en les contrôlant à partir d'un programme qu'on crée. Il en existe au maximum 40 pines male. Chaque pine est capable d'envoyer un signal électrique et aussi recevoir un d'où son abréviation il y a le "input/output » qui veut dire entrée et sortie.

1.4. LE MODULE RELAIS

Ce module relais qu'on voit sur la figure ci-dessous a deux canaux, ça se voit avec ces deux cubes en blues. Il existe aussi d'autres modèles, ceux avec quatre canaux et huit canaux.



Figure 4:un module relais a deux canaux

Un relais est un interrupteur qui se commande avec une tension contenue de faible puissance. La partie interruptrice sert à piloter des charges de secteur de forte puissance.

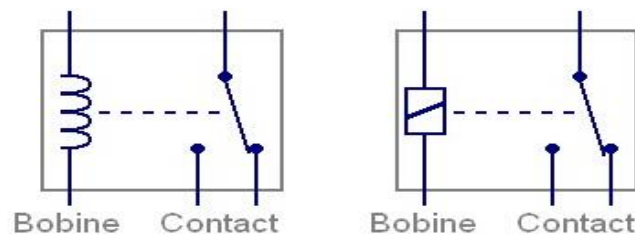


Figure 5:symbole d'un relais

1.4.1. Connexion au secteur

Dans chaque cube, il y a trois prises : commun (COM), normalement fermé (NC) et normalement ouvert (NO). Avec COM, c'est la broche qui est connecter directement au secteur. Avec NC, celle-ci est utilisée lorsqu'on souhaite que le relais soit fermé par défaut, ce qui signifie que le courant circule sauf si on désire de l'ouvrir à partir d'une commande. Et avec NO, elle est toujours ouverte sauf si on désire le fermer à partir d'une commande.

1.4.2. Câblage des broches



Figure 6:Description des prises et des broches d'un module relais

Le côté basse tension comporte un ensemble de quatre broches et un ensemble de trois broches. L'ensemble à droite comprend VCC et GND pour alimenter le module, ainsi que l'entrée 1 (IN1) et l'entrée 2 (IN2) pour contrôler les relais inférieurs et supérieurs, respectivement. Et le second ensemble de broches comprend les broches GND, VCC et JD-VCC. La broche JD-VCC alimente l'électroaimant du relais.

1.5. L'ELECTROVANNE

1.5.1. Définition

Une électrovanne est un dispositif électromécanique d'un circuit hydraulique, qui utilise un courant pour générer un champ magnétique et actionner ainsi un solénoïde qui contrôle l'ouverture de flux de fluide dans une vanne. Une électrovanne est constituée d'un seul bloc et il est impossible de dissocier le système d'actionnement du corps l'électrovanne.



Figure 7: Electrovanne à deux voies

1.5.2. Les différents types d'électrovannes



Figure 8: Une électrovanne multivoie

Les électrovannes peuvent être à deux voies ou multivoies. Généralement définies par deux chiffres, l'un détermine le nombre de voies et l'autre le nombre de positions. La plupart des électrovannes fonctionnent en tout ou rien. Elles peuvent être conçues pour fonctionner de façon proportionnelle au courant ou à la tension d'alimentation. Une électrovanne normalement fermée

s'ouvre lorsqu'elle est alimentée électriquement et normalement ouvert se ferme lorsqu'elle est alimentée électriquement.

1.5.3. Les avantages

On utilise une électrovanne si on a besoin de contrôler l'écoulement d'un liquide ou d'un gaz, que ce soit en régulation tout ou rien. On peut donc utiliser une électrovanne pour ouvrir ou fermer un circuit pour faire des dosages de produits, pour mélanger des gaz ou des liquides. L'un de ses avantages par rapport aux vannes classiques est que leur principe de fonctionnement permet un temps de réponse très rapide.

1.6. L'ELECTROAIMANT

Définition

Un électro-aimant est un appareil électromécanique qui se profite du courant pour produire des lignes de champs magnétique qui se murent par le noyau, induisant un effort et un déclassement de celui-ci. En l'inexistence de courant le noyau reste dans sa position et devient libre (sauf pour les électro-aimants monostables et bistables).



Figure 9: Un électro-aimant de manœuvre simple

1.6.1. Fonctionnement :

Quand la bobine est mise sous tension et le ligne de champ est créé, le noyau est attiré par la bobine dans l'électro-aimant et il redevient libre en absence de courant. La plupart des électro-aimants sont équipés d'un noyau traversant l'appareil. Lors du déplacement de ce noyau, il y a donc coté tirant et un coté poussant. Les 2 cotés peuvent être utilisés pour assurer une fonction.

Donc, il y a 3 types des fonctions liées à la forme du noyau :

- La fonction tirant : l'action magnétique induit un effort tirant

- La fonction poussante : l'action magnétique induit un effort poussant
- La fonction tirant/poussant : l'action magnétique induit un effort tirant d'un côté du noyau et poussant de l'autre.

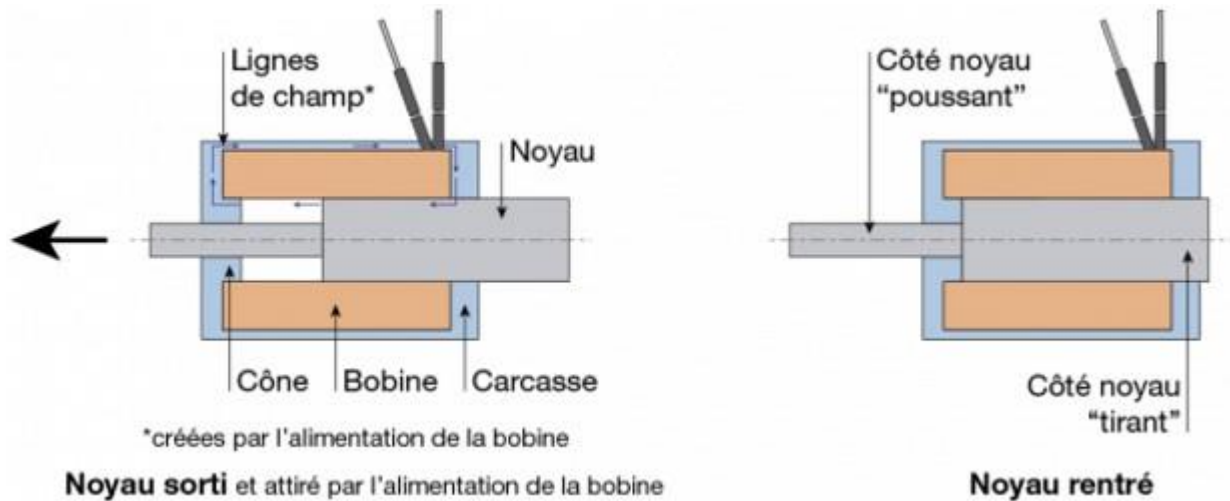


Figure 10:schéma du noyau sorti et rentré

1.6.2. Les avantages :

Les électro-aimants sont un type d'aimant qui transforme l'énergie électrique en énergie mécanique. Son avantage principal par rapport aux aimants permanents est la maîtrise de la force du champ magnétique, susceptible de contrôler la quantité de courant électrique. Mais dans ce projet, on utilise un électro-aimant pour contrôler le déversement des poudres de lait, de café et du sucre qu'ils soient tous solubles.

1.7. BOUILLLOIRE

1.7.1. Définition :

Une bouilloire électrique est un type de pot qui utilise l'électricité pour chauffer l'eau.



Figure 11:bouilloire

1.7.2. Fonctionnement :

Les bouilloires équipées de cette fonctionnalité sont les plus précises et permettent de régler la température ou degré près. Elle fonctionne en utilisant un élément chauffant à l'intérieur de la casserole pour amener l'eau à ébullition en quelques minutes.

1.7.3. Avantage :

La première chose qu'on retient de l'utilisation d'une bouilloire est le fait qu'on puisse faire bouillir de l'eau en très peu de temps. Car on a besoin de l'eau chaude pour préparer la boisson chaude comme : le thé, le thé au lait et le café. En général, l'eau bouillit en seulement 3 minutes. En plus, malgré cette rapidité de chauffe, elle consomme moins d'énergie

1.8. BOUTON POUSSOIR :

1.8.1. Définition :

Ce sont des interfaces homme machine très simples, ils permettent de Lancer un fonctionnement par l'action « appuyer / relâcher ». Ces dispositifs sont des astatables.

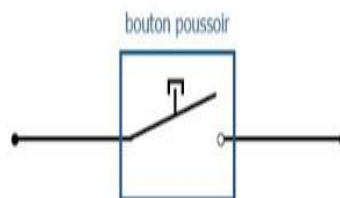


Figure 12: bouton poussoir

1.8.2. Fonctionnement :

Objectivement, C'est un organe électrique de commande permettant de démarrer une ou plusieurs actions. Quand en actionne elle met à l'état 1 et si en relâche, elle revient à l'état 0. On va voir le schéma d'une capture d'écran du chronogramme ci-après pour mieux comprendre son fonctionnement.

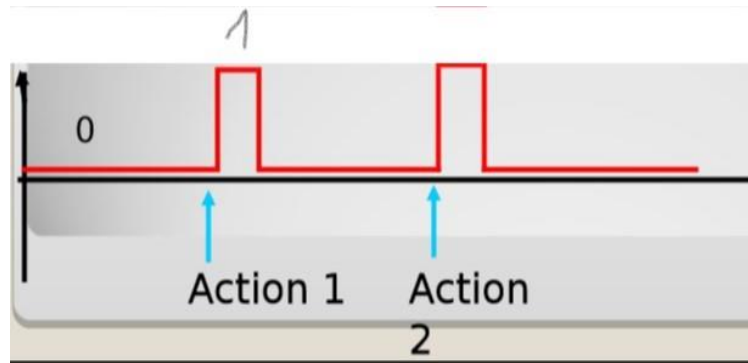


Figure 13:Schéma d'une capture d'écran du chronogramme

1.8.3. Avantage

Le bouton poussoir joue un rôle très important sur le boîtier. On connaît que c'est très difficile pour les hommes aveugle de faire la commande sur l'écran tactile. Donc on utilise un bouton poussoir pour les aides à faire de commande de boisson qu'ils voulaient.

1.9. CONCLUSION

On connaît déjà sur ce premier chapitre que notre circuit est bien séparé. La première est le circuit de commande et c'est que nous étudierons au chapitre suivant puis le second est le circuit de puissance. On voit aussi plus des matériels comme la bouilloire et le bouton poussoir.

CHAPITRE 2. LE CIRCUIT DE COMMANDE ET LA CONCEPTION DU PROGRAMME INFORMATIQUE

2.1. INTRODUCTION

Maintenant, on va discerner du circuit de commande d'électrovanne et d'électro-aimant. Et l'algorithme qui est une suite d'initiation qui manier les éléments de la partie matérielle et le travailler pour construire l'interface graphique. Chaque instruction a une mission pour le bon processus du programme, par exemple régir l'interface graphique et manipuler les GPIOs. Mais on va voir le circuit de commande d'abord et le configurer le Raspberry Pi suivi de la bibliothèque d'interface graphique utilise avant d'entre au programme informatique.

2.2. CIRCUIT DE COMMANDE

2.2.1. Etude et simulation du circuit de commande :

C'est par l'entremise de relais que notre système de commande repose sur le maniement d'électro-aimant et d'électrovanne. Et les relais sont manipuler par le GPIO du Raspberry pi, il est examiné comment être en marche le module relais.

On remarque sur le schéma ci-après qu'il y a un optocoupleur dans le relais. Un optocoupleur est un composant électronique capable de transmettre un signal d'un circuit électrique a un autre et un relatif aux courants électrique de basse tension entre eux. Par conséquence le circuit est séparé elle-même en deux à cause de cet optocoupleur. Donc, allons voir d'abord en priorité le schéma à l'intérieur d'un module relais pour le voire.

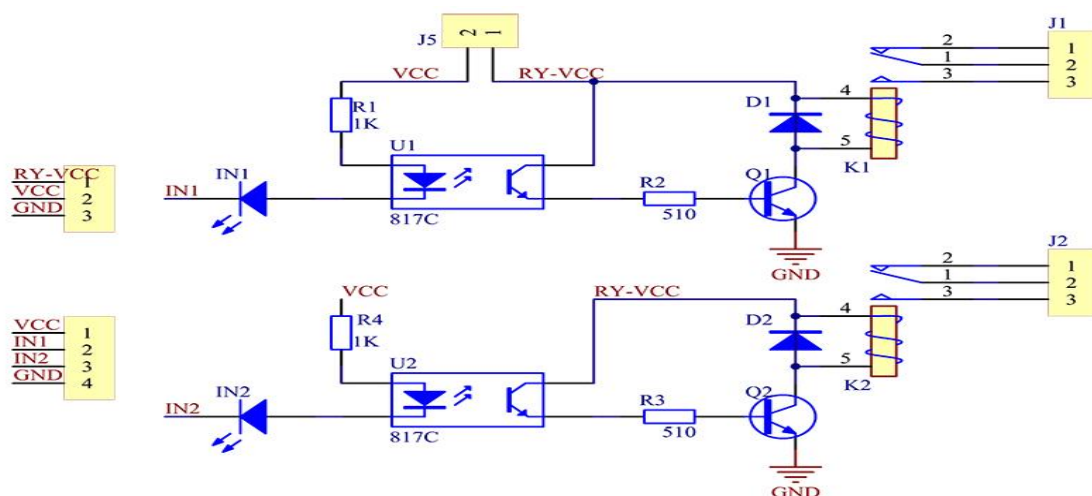


Figure 14:schéma d'un circuit à l'intérieur d'un module relais

On voit ci-dessous aussi, la simulation du circuit avec le logiciel Proteus version 8. Pourtant, on a capture que la manœuvre d'un seul relais qui est vif à partir de 5volt et une lampe de 12volt. Donc on a qu'un seul optocoupleur et un seul « Logic State » pour remplacer le IN1 par exemple. Et observer que dès qu'on met le Logic State à l'état 0, la lampe juste à droite s'éclaire et dans cas ou il est à 1, la lampe cesse de s'allumer.

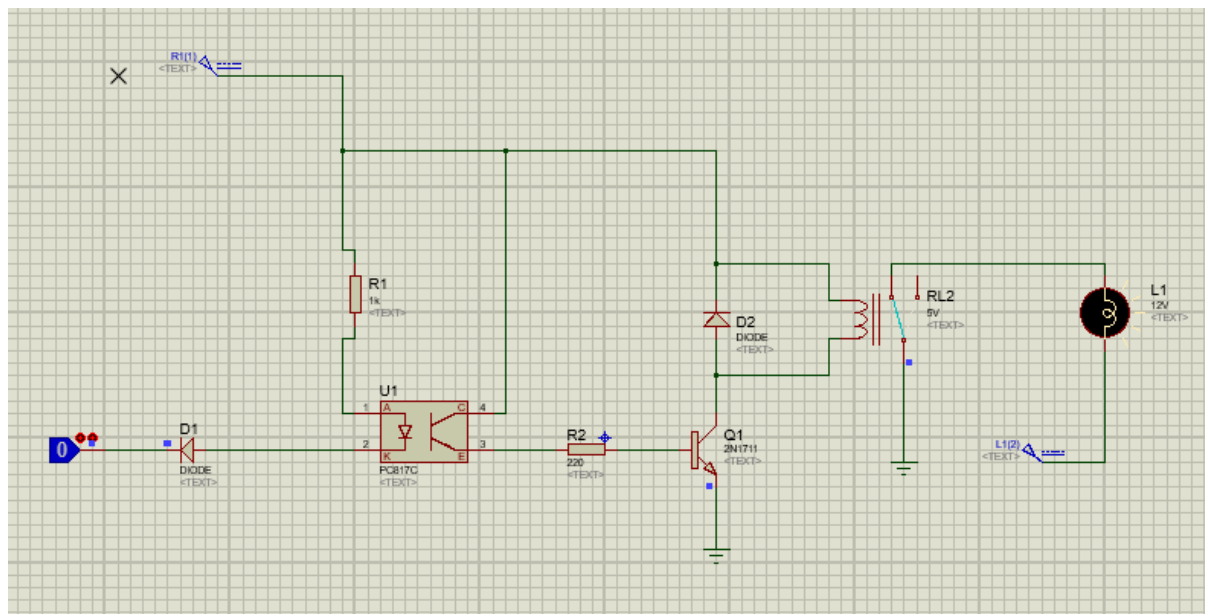


Figure 15:schema de simulation sur Proteus dont le Logic State est à l'état 0

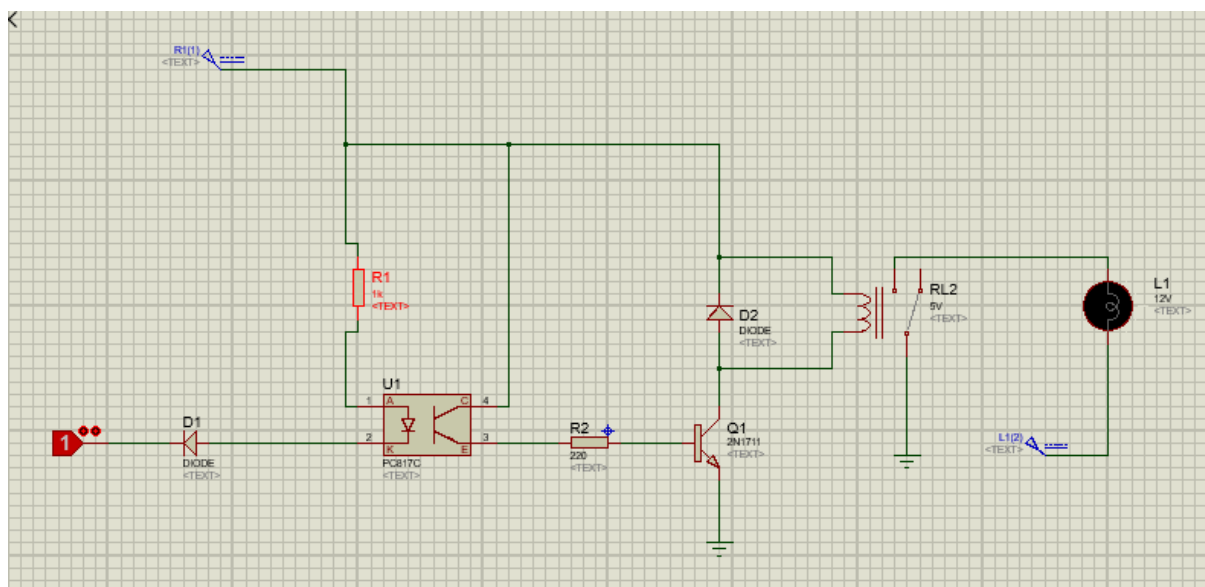


Figure 16:schema de simulation sur Proteus dont le Logic State est à l'état 1

2.2.2. Réalisation du circuit de commande

Pour réalisation le circuit de commande, il faut connecte une pine du module relais à une pine de GPIO pour maitrise un cube de relais. Et de brancher un fil dans a broche COM et un autre dans la broche

NO. On le branche de cette façon parce qu'on veut que l'électro-aimant et l'électrovanne sont verrouiller au lancement du système. Et une fois que les pines de commande sont à 3.3volt ou à 5volt, la bobine du relais est actionnée et les appareils verrouiller se déverrouille. Mais après un temps bien déterminer, le relais se relâche et les appareils revient à son état initial.

2.3. COMMENT INSTALLER UN SYSTEME D'EXPLOITATION DANS UN RASPBERRY PI :

Pour commence, il faut télécharger sur le site web(<https://www.raspberrypi.org/downloads/>) officiel du Raspberry Pi les image des systèmes d'exploitation pour le Raspberry Pi. Il faut aussi télécharger le logiciel d'écriture du système nommé « Raspberry Pi imager » dans une carte mémoire SD après avoir téléchargé l'image et il est accessible que ce soit sous Windows, sous MacOS et sous Linux. Ensuite on lance le logiciel d'écriture et on appuie sur le bouton « select image », sur la ci-après, pour cliquer sur le fichier image à partir d'un dialogue aussi bien que la figure 1 et 2. Ainsi qu'après il y a le maintenant du fichier et c'est après on appuie sur le bouton « select carte SD », pour choisir la carte que l'on veut écrire. Puis enfin, on appuie sur bouton « écriture » pour démarrer l'écriture du carte SD.



Figure 17:Fenêtre d'accueil du logiciel Raspberry Pi imager

Ainsi que l'écriture est achevée, on éjecte la carte SD, et on retire de l'adapter la carte micro-SD et on l'injecte dans le Raspberry PI. Et une fois placé, il suffit juste de démarrer le Raspberry PI et directement le système marche.

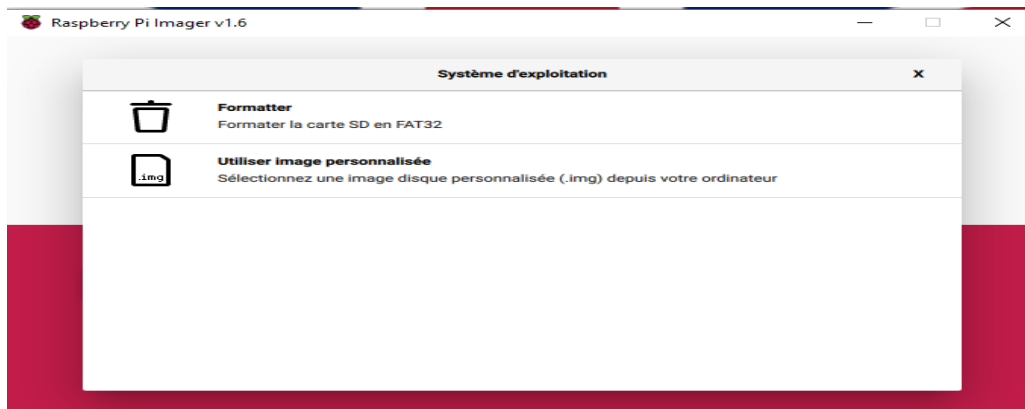


Figure 18:fenêtre pour sélectionner l'image du système d'exploitation

2.4. INTEGRATION DU PROGRAMME DANS LE RASPBERRY PI :

2.4.1. Installation de la bibliothèque manquante

Pour produire l'interface graphique, il faut d'abord installer la bibliothèque marquante « kivy » si le système est déjà en marche que ça soit en mode graphique ou en mode console. Il faut d'abord ouvrir un terminal, si on est en mode graphique ou pas pour l'installer. Mais, si on est en mode console, et une fois le terminal ouvert, on y écrit : `sudo apt install python3-kivy` pour python3 ou `sudo apt install python-kivy` pour python2. Et après avoir dactylographier la commande, on doit taper sur le bouton entre du clavier et le téléchargement des paquets d'installation est lancé. Une la réception est parfaite la bibliothèque sera installée systématiquement. Alors que e nécessaire soit installé régulièrement, il faut tester si elle est installée convenablement en tapant sur le terminal la commande « `python` » ou « `python3` ». Lorsqu'on entre dans le compilateur de python, on doit importer la bibliothèque kivy en tapant une ligne de code : `import kivy`. Après son exécution si le copulateur ne programme pas d'erreur donc elle est belle et bien installée dans les modules c de python.

2.4.2. Lancement du programme dans le Raspberry PI

On clone l'algorithme dans un carte ou dans un clef USB et on le mettre dans l'arborescence des fichiers mais c'est bien de la place sur le bureau pour esquiver des problèmes dans le système d'exploitation et dans le but de le facilité d'accès. Il faut bien donc démarre l'application obligatoirement lorsque le projet marche automatiquement au démarrage. D'après cela il faut lancer un fichier contenu dans « `/etc/` » nomme « `profile` ». Pour pouvoir faire l'édition, il faut ouvrir un éditer de texte soit « `vi` » ou « `nano` » en tapant la commande : `sudo nano /etc/profile` pour l'éditeur « `nano` » et `sudo vi /etc/profile` pour « `vi` » et on l'exécute en appuyant la

touche entrer du clavier. Une fois le fichier ouvert, on écrit juste a la fin du script la commande de lancement du programme voulu en écrivant : python ou python3 suivi du répertoire du programme a lance. Avant de fermer le programme d'édition il faut sans oublier l'enregistrement en tapant sur le ctrl+o et pour quitter en tapant ctrl+z pour quitter quand on utilise « nano » et : wq pour enregistrer et quitter sur « vi ». Et une petite remarque, avec « vi » il faut appuyer sur esc+i pour insérer du texte et esc pour pouvoir éditer dans la zone de commandes. Pour finir, on redémarre le Raspberry Pi à partir de la commande « sudo reboot » pour aller plus vite ou on va vers le menu de démarrage. Au démarrage, le programme se lance directement.

2.4.3. Mise en place de l'écran tactile TFT :

Pour mise en place l'écran tactile sur le Raspberry PI, il suffit de le brancher sur le Raspberry PI suivant l'alignement du côté droit vers la gauche. Il faut installer le pilote dans le système d'exploitation, lorsque est brancher.



Figure 19:Emplacement du module TFT sur un Raspberry PI

Pour l'installer il faut taper les commandes ci-dessous :

```
cd /home  
  
sudo rm -rf LCD-show  
  
git clone https://github.com/goodtft/LCD-show.git  
  
chmod -R 755 LCD-show  
  
cd LCD-show/  
  
sudo ./MHS35-show  
  
sudo reboot
```


Et après le lancement des commandes, le système redémarre et l'écran se met tout en blanc d'abord et soudainement il se met en mode console pour le démarrage. A partir du stylet, on peut cliquer sur les éléments cliquables plus précisément.

2.4.4. Installation de haut-parleur

Il faut juste de relier la prise femelle du Raspberry PI et le jack mâle de haut-parleur.



Figure 20:port d'entrer d'une prise Jack



Figure 21:prise Jack mâle

2.4.5. Augmentation du son :

Le niveau de son est très faible au lancement du Raspberry pi. Par conséquent, on doit le programme systématiquement dès le démarrage. Pendant cela, il faut dactylographier la commande ci-dessous dans « /etc/profile », pour augmenter le son au maximum, juste au-dessus du programme de distributeur de boisson :

```
amixer -q -D pulse sset Master 100%+
```

2.5. QU'EST-CE-QUE KIVY ? :

Il existe de nombreuses bibliothèques pour réaliser des interfaces graphiques avec Python. Les différences entre eux c'est le support de plusieurs systèmes d'exploitation (Linux, Windows, MacOS...) ainsi que les composants graphiques proposés (bouton, zone de texte, menu, liste déroulante...). Parmi les choix majeurs, il

y a les plus connues qui sont PyQt, Kivy et WxPython. Et ici, on va parler de Kivy. Kivy est une bibliothèque pour python qui est utile pour la création d'une application pour écran tactiles pourvues d'une interface utilisateur naturelle. Elle est libre et open source.

2.5.1. Historique :

Kivy est un framework développé par l'organisation Kivy, en parallèle avec Python-for-Android, Kivy iOs, ainsi que plusieurs autres bibliothèques destinées à être utilisable sur toutes ces plateformes. Kivy a gagné un financement de 5000 dollars de la Python Software Fondation pour le portage vers Python3.3 en 2012. Il supporte aussi le Raspberry Pi et pour cela qu'on l'utilise pour ce projet.

2.5.2. Les fonctionnalités disponibles :

Il contient plusieurs éléments pour la construction d'une application :

- Il nous fournit des fonctionnalités de saisie étendues pour la souris ; le clavier ; les interfaces utilisateurs tangibles, ainsi que les évènements multitouch générés par ces différents matériels.

- Il est basé sur OpenGL ES2 donc il utilise les objets Tampons Vertex et les Shaders.

- Il possède une large gamme de widgets acceptant le multi touch.

- A l'aide d'un langage intermédiaire, le Kv, on peut construire facilement des widgets personnalisés.

2.5.3. Le langage Kv (Kivy Language) :

Le langage Kv est un langage spécifique à la description des interfaces et des interactions avec l'utilisateur. Comme en QML de Qt, il est possible de créer facilement l'ensemble de l'interface utilisateur d'un programme et y relier les actions utilisateurs.

2.5.4. Avantage de kivy :

- **RAPIDE**

kivy est rapide. Sa rapidité se montre au niveau de développement et à la fois à la vitesse d'exécution des applications. Les concepteurs ont mis en oeuvre une fonctionnalité critique au niveau C pour une exploitation de la puissance des compilateurs existants. Comme la puissance de calcul des cartes graphiques actuelles dépasse celle des processeurs actuels pour certaines tâches et certains algorithmes, ils ont

profité de cette puissance pour laisser le processeur graphique à effectuer le maximum de travail dans le but d'augmenter considérablement de puissances.

- **FLEXIBILITE**

Kivy est souple. Cela veut dire qu'on peut l'utiliser sur différents appareils, notamment sur les smartphones et les tablettes Android. Il supporte aussi certains systèmes d'exploitation comme Windows, Linux, OSX, etc. Vue sa rapidité de développement, il peut aussi s'adapter rapidement aux nouvelles technologies.

- **FRAIS**

Kivy est fait pour aujourd'hui et demain. De nouvelles méthodes de saisie telles que Multi-Touch sont devenues de plus en plus importantes. Nous avons créé Kivy à partir de rien, spécialement pour ce type d'interaction. Cela signifie que nous avons été en mesure de repenser beaucoup de choses en termes d'interaction homme-machine, alors que les kits d'outils plus anciens (pour ne pas dire «obsolètes», plutôt «bien établis») portent leur héritage, ce qui est souvent un fardeau. Nous n'essayons pas d'aboutir à une nouvelle approche informatique entre ordinateurs et modèles de modèles existants (interaction point à pointeur / souris). Nous voulons le laisser fleurir et vous laisser explorer les possibilités. C'est ce qui distingue vraiment Kivy

2.5.5. Création d'une interface graphique

Pour faire simple, commençons d'abord à créer une simple interface « Hello World » avec Kivy.

Code source 1.1:

```
1- from kivy.app import App
2- from kivy.uix.label import Label
3- class HelloApp(App):
4-     def build(self):
5-         return Label(text='Hello World!', font_size='100sp')
6- HelloApp().run()
```



Figure 22:HelloWord en Kivy

Une interface graphique se définit à l'aide d'une nouvelle classe de type `App`, comme on l'a indiqué entre parenthèses après le nom de la classe. On a déjà vu cette construction précédemment, pour définir ses propres exceptions. Dans cette classe, il faut, au minimum, définir une méthode « `build` » qui va construire, et renvoyer, les composants à placer dans la fenêtre. Dans notre cas, on va se limiter à un composant de type `Label`, c'est-à-dire une zone de texte non éditable. Enfin, pour lancer le programme, il suffit de créer une instance de la classe qui a été définie, et d'appeler `run`. Cette méthode `run`, spécifique aux objets de type `App`, initialise toute une série de choses permettant d'obtenir une interface graphique. Elle s'occupe notamment de mettre en place l'interaction avec le hardware de l'écran, de discuter avec les dispositifs d'entrée/sortie tels que l'écran multitouch, le clavier, l'accéléromètre, etc. et enfin, elle planifie plusieurs tâches à exécuter de manière régulière.

Le principal élément d'une application qui possède une interface graphique est la fenêtre. Par défaut, une application Kivy se charge de créer une fenêtre, dans laquelle les composants créés par la méthode « `build` » seront placés. On peut facilement personnaliser quelques caractéristiques de cette fenêtre. Tout d'abord, on peut modifier son titre et l'icône associée à la fenêtre et à l'application à l'aide des variables d'instance « `title` » et « `icon` », à modifier dans la méthode « `build` ». On peut également modifier la taille, en pixels, de la fenêtre en configurant deux options. L'exemple suivant modifie ces trois éléments :

Code source 1.1:

```

1- from kivy.app import App

2- from kivy.config import Config

3- from kivy.uix.button import Label

4- class HelloApp(App):

5- def build(self):

6-self.title = 'Hello World'

7- self.icon = 'hello.png'

8- return Label(text='Hello World!')

9- Config.set('graphics', 'width', '300')

10- Config.set('graphics', 'height', '150')

11- HelloApp().run()

```

La modification de la taille de la fenêtre passe donc par une fonction dans la classe Config du module kivy.config et elle doit être faite avant le lancement de l'application.

2.5.6. Composant graphique :

Une interface graphique est construite à partir de composants que l'on place sur une fenêtre. On retrouve plusieurs types de composants tels que des zones de texte éditables ou non, des boutons, des listes, des cases à cocher, etc. Les composants possèdent un composant parent. Certains composants, appelés conteneurs, n'ont aucun rendu visuel. Ils permettent simplement d'en accueillir d'autres, afin d'organiser l'aspect visuel global de l'interface graphique. Dans l'exemple de la figure 3, on en a utilisé deux.

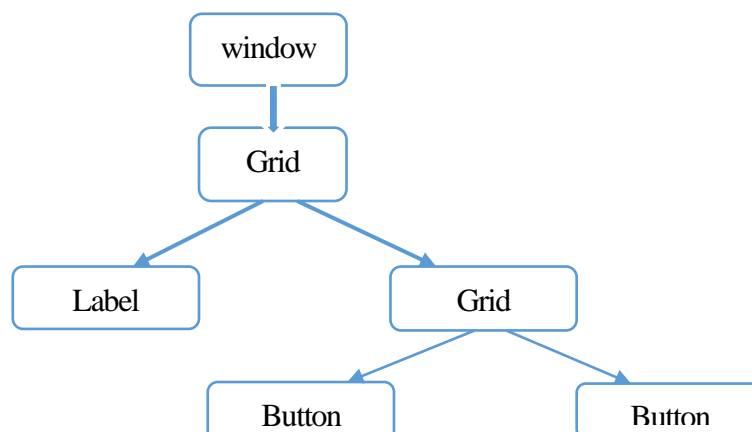


Figure 23:schéma d'imbrication des widgets

Pour comprendre comment sont organisés les composants d'une fenêtre, on peut représenter les liens entre ces derniers sous forme d'un arbre. Une flèche arrive sur chaque composant, reliant son composant parent à lui. De plus, une ou plusieurs flèches partent de chaque composant, le reliant à ses composants enfants, c'est-à-dire ceux qui y sont imbriqués.

2.5.7. Widget

Un widget est un composant qui offre une interaction spécifique avec l'utilisateur, en pouvant lui montrer une information et en offrant une ou des interactions possibles avec lui. Un bouton permet, par exemple, de montrer un texte et de recevoir un clic gauche. On ne va pas passer en revue tous les widgets proposés par Kivy mais uniquement les principaux.

Commençons avec un exemple d'une simple interface graphique qui regroupe trois composants de base :

Le « label » est une zone permettant d'accueillir un texte qui ne peut pas être modifié. On l'utilise, par exemple pour renseigner le nom d'un champ à remplir dans un formulaire.

La « zone de texte » est une zone dans laquelle l'utilisateur peut entrer un texte. On l'utilise, par exemple, comme champ d'un formulaire.

Le « bouton » est une zone sur laquelle l'utilisateur peut cliquer, pour déclencher une action. On l'utilise, par exemple, pour permettre à l'utilisateur de valider un formulaire.

Pour ce faire, on va utiliser un conteneur de type « BoxLayout » qui aligne les composants qui lui sont ajoutés, de gauche à droite et horizontalement.



Figure 24: Capture d'écran d'une fenêtre utilisant BoxLayout

Voici le programme qui permet de construire l'interface graphique que l'on souhaite pour la fenêtre de connexion :

```
1- from kivy.app import App
```

```

2- from kivy.config import Config
3- from kivy.uix.boxlayout import BoxLayout
4- from kivy.uix.button import Button
5- from kivy.uix.label import Label
6- from kivy.uix.textinput import TextInput
8- class LoginApp(App):
9- def build(self):
10- self.title = 'Se connecter'
11- box = BoxLayout(orientation='horizontal')
12- box.add_widget(Label(text='Pin code'))
13- box.add_widget(TextInput())
14- box.add_widget(Button(text='Entrer'))
15- return box
17- Config.set('graphics', 'width', '350')
18- Config.set('graphics', 'height', '50')
20- LoginApp().run()

```

Le premier composant que l'on crée, et qui sera renvoyé par la méthode « build », est le « BoxLayout ». Ce dernier va contenir tous les autres composants qui lui seront ajoutés à l'aide de sa méthode « add_widget ». De plus, ils seront alignés horizontalement comme on l'a précisé dans le constructeur lors de la création du conteneur. Les composants sont placés dans le « BoxLayout » en suivant l'ordre des appels à « add_widget ».

2.5.8. Gestionnaire de mise en page

Un conteneur de type « BoxLayout » positionne ses composants les uns à côté des autres, soit horizontalement, soit verticalement. On précise le sens que l'on désire à l'aide du paramètre nommé « orientation » du constructeur, qui peut valoir « horizontal » ou « vertical ». Mais un autre conteneur très utilisé est le « GridLayout », qui permet d'organiser les composants qu'il héberge sous forme d'une grille. Le nombre de lignes et de colonnes que l'on souhaite est spécifié par les paramètres nommés « rows » et « cols » du constructeur. Les éléments que l'on ajoute à ce conteneur sont placés dans la grille de gauche à droite et de haut en bas. L'exemple suivant crée une interface graphique avec 12 boutons placés dans une grille de trois lignes et quatre colonnes :

```

1- from kivy.app import App

```

```

2- from kivy.config import Config
3- from kivy.uix.button import Button
4- from kivy.uix.gridlayout import GridLayout
6- class GridGameApp(App):
7- def build(self):
8- self.title = 'Grid Game'
9- grid = GridLayout(rows=3, cols=4)
10- for i in range(12):
11- grid.add_widget(Button(text=str(i + 1)))
12- return grid
14- Config.set('graphics', 'width', '200')
15- Config.set('graphics', 'height', '150')
17- GridGameApp().run()

```

Après avoir exécuté ce programme ci-dessus, vous aurez une fenêtre similaire à la figure 1.4.

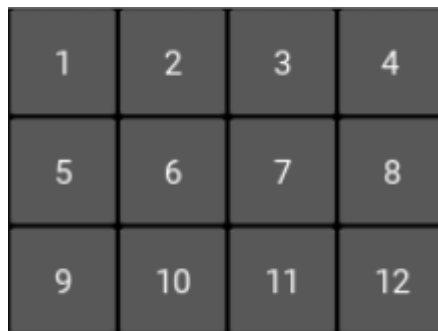


Figure 25: Fenêtre contenant des Button contenus dans un GridLayout

On peut aussi positionner les éléments exactement là où on le souhaite avec la taille que l'on souhaite en utilisant un conteneur de type « FloatLayout ». Mais il faut d'abord spécifier la taille du conteneur avec le paramètre nommé `size_hint` de son constructeur. Ensuite, lorsqu'on lui ajoute des composants, ceux-ci seront placés à la position qui a été spécifiée par le paramètre nommé `pos`. Pour le programme ci-dessous on va essayer de créer une fenêtre avec un conteneur de type « FloatLayout » qui contient deux boutons qui ont été placés à des endroits bien précis. Ces boutons ont par ailleurs une taille fixée à (0.3, 0.2), c'est-à-dire 30% de la fenêtre en largeur et 20% en hauteur :

```

1- from kivy.app import App
2- from kivy.config import Config
3- from kivy.uix.button import Button

```



```

4- from kivy.uix.floatlayout import FloatLayout
5- class FreePosApp(App):
6- def build(self):
7- self.title = 'Free Positioning'
8- box = FloatLayout(size=(200, 150))
9- box.add_widget(Button(text='A', size_hint=(0.3, 0.2), pos=(0, 0)))
10- box.add_widget(Button(text='B', size_hint=(0.3, 0.2), pos=(50, 80)))
11- return box
12- Config.set('graphics', 'width', '200')
13- Config.set('graphics', 'height', '150')
14- FreePosApp().run()

```

Après avoir exécuter ce programme ci-dessus vous aurez une fenêtre comme celle de la figure ci-dessous.



Figure 26: Fenêtre contenant des Button contenus dans un FloatLayout

2.5.9. Le langage KV

Simplifier la construction de l'interface graphique, c'est-à-dire établir la liste des composants avec leurs propriétés et leur placement, peut se faire plus facilement à l'aide du langage KV. La description se fait dans un fichier séparé, qui porte le même nom que celui de l'application (à savoir celui de la classe sans App), et avec l'extension .kv. Prenons l'exemple précédent de la fenêtre de login, le programme principal est :

```

1- from kivy.app import App
2- from kivy.config import Config
4- class LoginApp(App):
5- pass

```

7- Config.set('graphics', 'width', '350')

8- Config.set('graphics', 'height', '50')

10- LoginApp().run()

Et puis dans le même dossier que le programme principal, on crée un fichier nommé login.kv, avec le contenu suivant :

1- BoxLayout:

2- orientation: 'horizontal'

3- Label:

4- text: 'Pin code'

5- TextInput

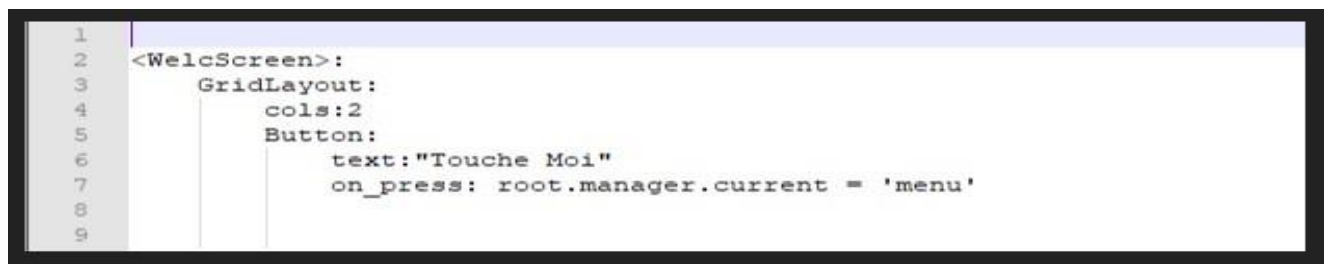
6- Button:

7- text: 'Entrer'

Le langage KV permet en fait de décrire un genre de dictionnaire de manière textuelle. Le premier composant déclaré dans ce fichier est celui qui sera inséré dans la fenêtre, c'est-à-dire celui qui est renvoyé par la méthode « build », c'est-à-dire un conteneur de type « BoxLayout » dans notre cas. Vient ensuite un bloc indenté qui reprend les propriétés du composant et les autres composants qu'on veut lui imbriquer. Dans notre cas, on imbrique, dans l'ordre, un label, une zone de texte et un bouton, dans le conteneur.

2.6. EXTRAITS DES CODES IMPORTANTS

Derrière son interface graphique, il y a des extraits de code de programmation essentiel mieux à connaître. Pour cela on ne va tout expliquer le programme tout en entier mais on va juste mettre en discussion quelques points intéressants du programme. C'est dans cette partie qu'on va les voir les points intéressants. Au début on voit sur l'écran « touche moi » et voici son extrait de code.



```
1
2 <WelcScreen>:
3     GridLayout:
4         cols:2
5         Button:
6             text:"Touche Moi"
7             on_press: root.manager.current = 'menu'
8
9
```

Figure 27: code qui affiche touche moi

Après que l'utilisateur clique sur « touche moi » il entre sur le menu principal et on va voir ci-dessus le code qui l'exécute.

```
10 <MenuScreen>:
11     GridLayout:
12         cols:1
13         GridLayout:
14             cols:2
15             Button:
16                 text:"BoissonChaud"
17                 on_press: root.manager.current ='chaud'
18             Button:
19                 text:"BoissonFroid"
20                 on_press: root.manager.current = 'froid'
21         GridLayout:
22             cols:3
23             Button:
24                 text:"RETOUR"
25             Label:
26                 id:label_menu
27                 text:"My menu"
28             Button:
29                 text:"Settings"
30                 on_press: root.manager.current ='settings'
31
```

Figure 28: code qui affiche le menu principal

Sur le menu principal, on ne voit que boisson chaude, boisson chaude, retour, my menu et setting. Puis on touche sur l'un des deux boissons mais prennent l'exemple sur boisson chaude. Quand on le touche, on voit la liste des boissons chaude disponible et voici le code qui les affiche.

```
41 <ChaudScreen>:
42     GridLayout:
43         cols:1
44         GridLayout:
45             cols:5
46             Button:
47                 text:"RETOUR"
48         GridLayout:
49             cols:3
50             Button:
51                 id:bt_the
52                 text:"DU THE"
53                 on_press: root.affect_choix('CHAUD',bt_the.text)
54                 on_press: root.manager.current ='sugar'
55             Button:
56                 id:bt_lait
57                 text:"LAIT"
58                 on_press: root.affect_choix('CHAUD',bt_lait.text)
59                 on_press: root.manager.current ='sugar'
60             Button:
61                 id:bt_cocktail
62                 text:"COCKTAIL"
63                 on_press: root.affect_choix('CHAUD',bt_cocktail.text)
64                 on_press: root.manager.current = 'sugar'
65
```

Figure 29: code qui affiche les boissons chaudes

A fin, on choisit le taux de sucre qu'on voulait avoir dans la boisson. Et la boisson s'écoulera dans le verre suivit du sucre. On voit ci-après le code qui affiche le taux de sucre sur l'écran.

```
66 <SugarScreen>:
67     GridLayout:
68         cols:1
69         GridLayout:
70             cols:5
71             Button:
72                 text:"RETOUR"
73         Label:
74             id:mylabel
75             text:"CHOIX DU TAUX DU SUCRE (15 gramme => 100%)"
76         GridLayout:
77             cols:3
78             Button:
79                 text:"-- 0% --"
80                 on_press: root.affect_taux('0')
81                 on_press: root.manager.current ='bois'
82             Button:
83                 text:"-- 50% --"
84                 on_press: root.affect_taux('50')
85                 on_press: root.manager.current ='bois'
86             Button:
87                 text:"-- 100% --"
88                 on_press: root.affect_taux('100')
89                 on_press: root.manager.current ='bois'
90
```

Figure 30: code qui affiche le taux du sucre

On note bien que c'est la langage Kivy qu'on utilise pour affiche tout ce qui se dit au-dessus consternant l'affichage sur l'écran.

Conclusion

En résumé, ce second chapitre nous a menés à bien comprendre la configuration à faire sur le Raspberry pi et le fonctionnement du circuit de commande de ce projet. Puis elle nous faire connaitre aussi le langage de programmation à utiliser.

CHAPITRE 3. LES DIFFERENTES FONCTIONNALITES AJOUTER DE LA DISTRIBUTION ET AMELIORATION DU BOITIER

3.1. INTRODUCTION

Après tout le reste, on va discuter de l'amélioration du boîtier, par suit, on va examiner point par point les fonctionnalités ajouter. Puis on doit parler un peu du circuit de commande qui va commander l'électro-aimant. Et pour accomplir le projet, on va les réaliser.

3.2. LES FONCTIONNALITES AJOUTER

3.2.1. Les boissons chaudes

A l'origine de ce projet, on a que des boissons fraîches et un cocktail. Et maintenant, on va ajouter d'autre fonction essentielle comme un distributeur des boissons chaude et un cocktail de boisson chaude. On a deux (02) variétés de boisson chaude simple : le premier est le thé et le deuxième est le lait. Pour le cocktail : on a que le thé au lait.

En suit, le sujet demande aussi d'ajoute un distributeur de sucre. Alors, on l'additionne et pour plus de raison de le faire, on a besoin un peu de sucre sur les boissons chaude afin d'avoir le bon goût.

3.2.2. Système d'aide aux malvoyant et interface de communication sonore

Pour donner un coup de main aux aveugles à faire ces commandes. On doit installer un système d'aide aux malvoyants car ils n'ont pas capable d'utiliser un écran tactile. Donc on invente un moyen qui est simple et capable de communiquer avec eux. Avec ce issue, le non-voyant suffit d'actionner un bouton nome bouton poussoir.

On a aussi besoin d'interface de communication sonore comme un buzzer suivie d'une assistance vocal. Le buzzer assure la conscience du malvoyant qu'il actionne vraiment le bouton poussoirs. Et l'assistance sonore lui faire connaître par vocale ce qui est en marche à chaque fois qu'il appuis le bouton poussoir.

3.3. FONCTIONNEMENT

Avant de commande il suffit de touche n'importe quel touche et on voit sur l'écran d'application le menu principal qui contient de la boisson chaude, boisson fraiche, retour, my menu et settings. On doit sélectionner l'un des deux boissons pour entre à l'étape suivant. Si l'utilisateur clique sur la boisson chaude, après on voit la liste des boissons chaude disponible dans le boitier du distributeur comme le lait et le thé mais il y a du retour aussi. En suit-il select l'un des boissons chaudes et on entre sur le choix du sucre disponible et après le choix du sucre, la boisson dans le boitier se versera automatiquement dans le verre suivi d'un écoulement automatique du sucre. C'est pareil aussi pour la boisson fraiche mais son processus n'a pas besoin de cette dernière étape car elle est déjà sucrée.

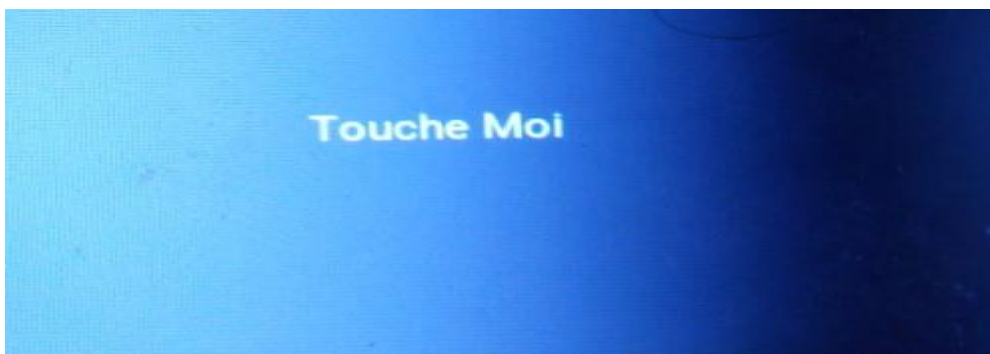


Figure 31: capture du premier vu sur l'écran

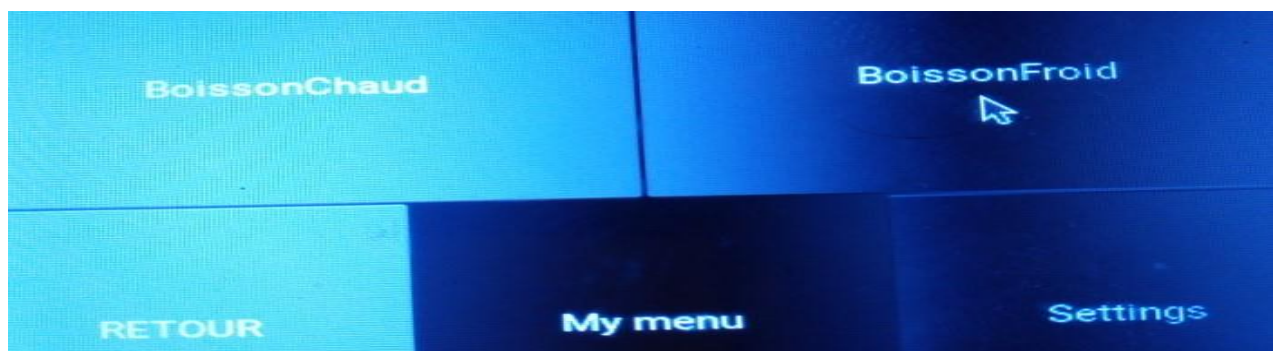


Figure 32: capture du menu principal

Mais deux cas pourrait se manifester au moment de commander de boisson. Premièrement, si tout ce qui était dit en haut est marché. On obtient une animation qui simule le versement de la boisson dans le verre et quand la boisson est prête à déguster l'animation se disparaît et l'écran revient au menu principal.

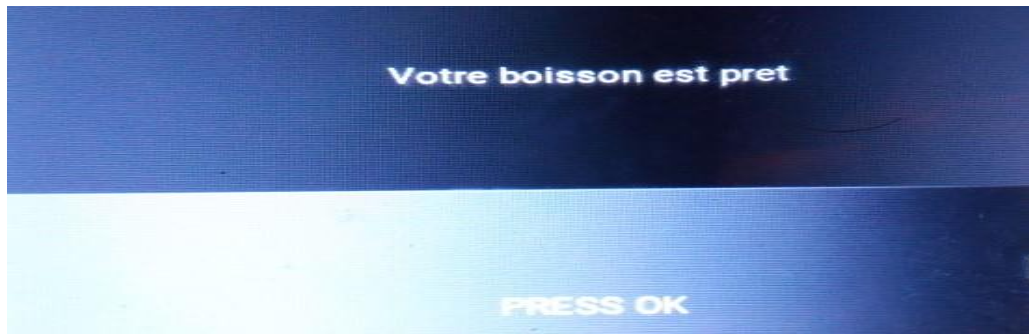


Figure 33:

Pour le deuxième cas, si la boisson choisie n'est pas encore configurée une phase qui dit la quantité versée est nul » s'affiche sur l'écran. Et après il revient au menu principal aussi.

3.4. AMELIORATION DU BOITIER

Il faut qu'on représente encore le boîtier en différente vue pour mieux voir les parties modifiées.

3.4.1. Représentation du boîtier en différente vue

- Vue de face

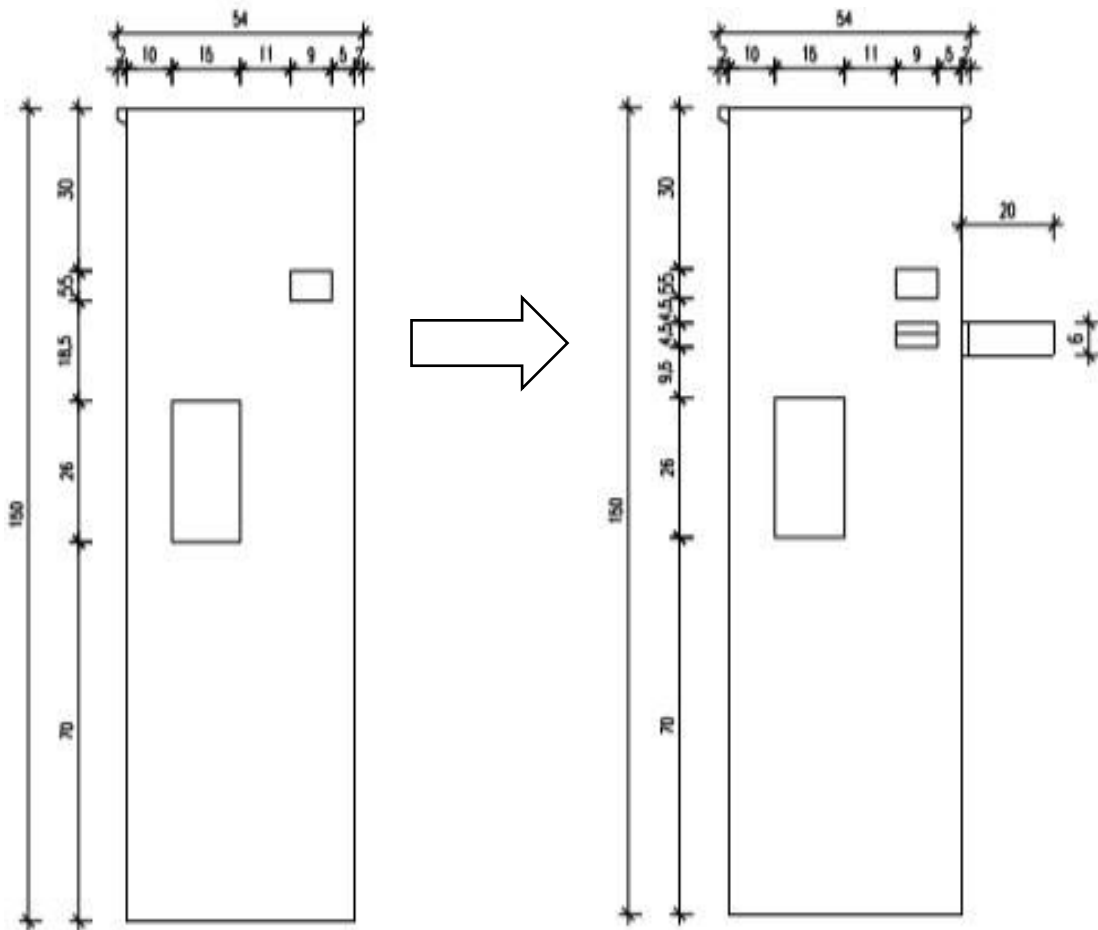


Figure 35 : Vue de face avant

Figure 34: Vue de face à présent

D'après ce vu de face à présent, on a un boîtier de 150 cm de hauteur et de 54 cm de largeur. Sur sa face, on a trois espaces dont l'un est en rectangulaire creux et l'autre est un trou sous forme de rectangle et le nouveau est l'emplacement des boutons poussoir. On voit aussi l'emplacement du verre à l'extrémité droite du boîtier.

- Vue de derrière

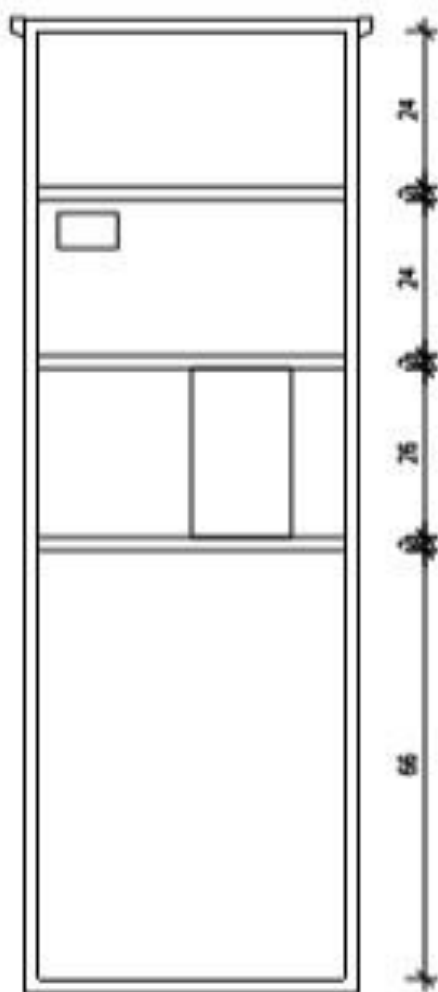


Figure 36 : vue de derrière avant

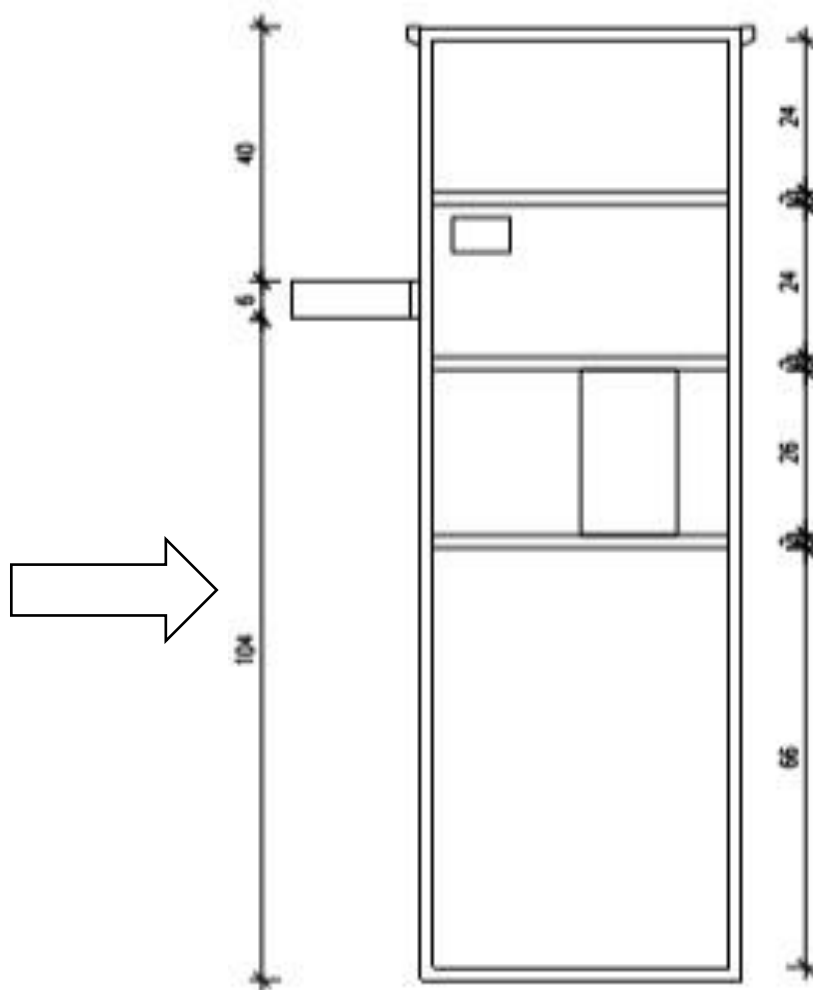


Figure 37:vue de derrière à présent

En vue de derrière, le boîtier est cloisonné en quatre cellule. Dans la première section, on voit les différents types de boissons et le sucre. La deuxième sert à contenir le circuit de commande tout entier, l'hautparleur, l'électro-aimants et les électrovannes. La troisième section, c'est là que l'on place notre verre

pendant le versement. Et la quatrième section, elle est réservée pour des projets futurs mais on peut quand même y stocker des boissons de réserve au cas où ceux de la section 1 sont tous versés.

- Vue de dessus

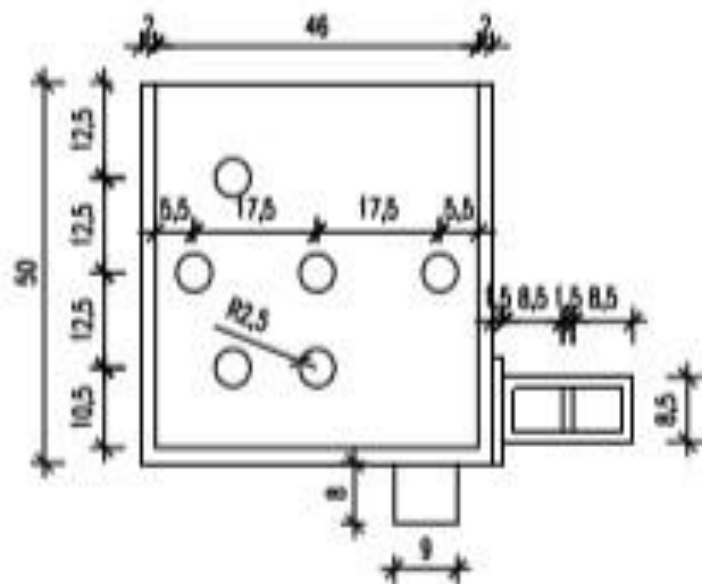
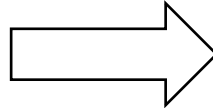
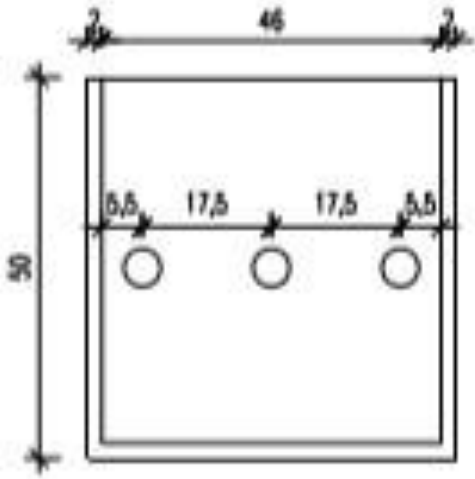


Figure 38: Vue de dessus avant du boîtier

Figure 39: Vue de dessus à présent

On voit 6 trous de 5cm de diamètre d'après la vue de dessus. Ces trous ont été perforés pour pouvoir y insérer des tuyaux de diamètre égale à 3cm. Et c'est là aussi on place les bocaux contenant les boissons.

- Vue de droite

On implante un emplacement des verres sur la vue de droite. Le boîtier a la même hauteur et la même largeur comme en vue de face, c'est à cause de cette conformité qu'on ne remet pas la mesure qu'à la modification. Mais pour l'emplacement du verre : 14cm de largeur et 6cm de hauteur.

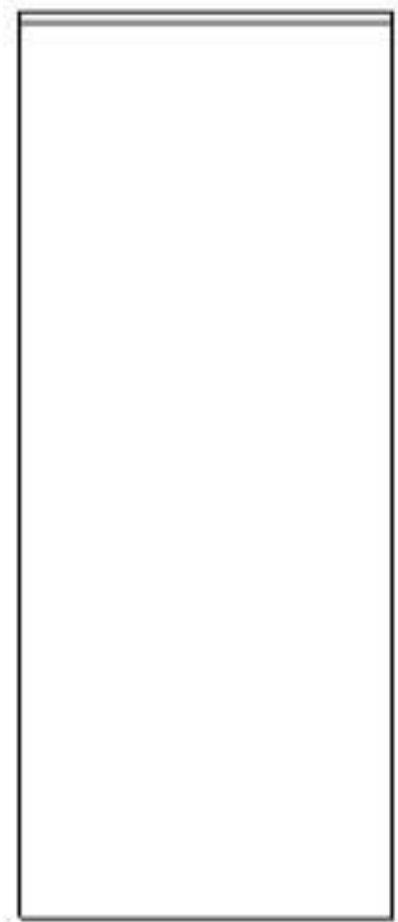


Figure 40: Vue de gauche avant

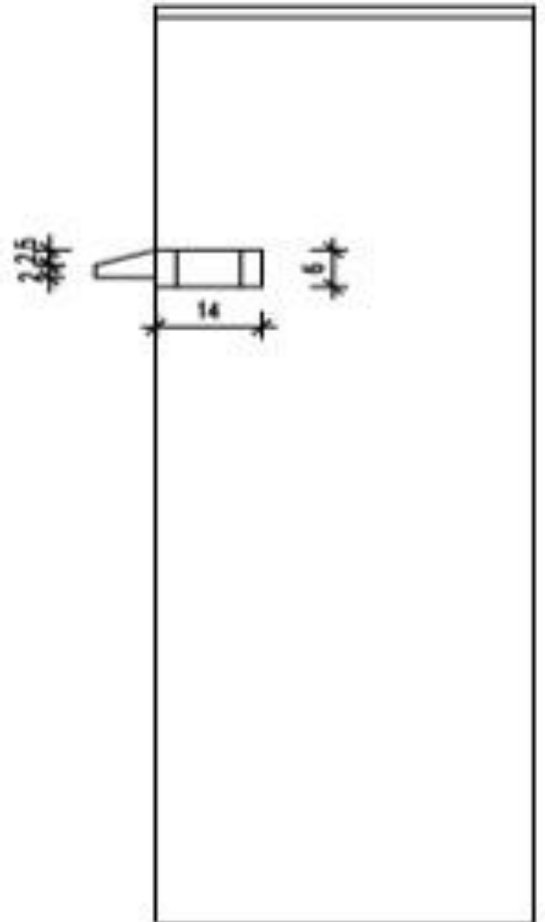
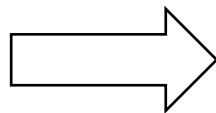


Figure 41: Vue de droite à présent

- Vue de gauche

En remarque sur la vue de gauche que rien est changé. Mais on observe l'emplacement des bouton poussoir place sur la vue de face à partir de vue de gauche. On ne met pas de la mesure sur gauche car il a le même hauteur et largeur à la vue de face.

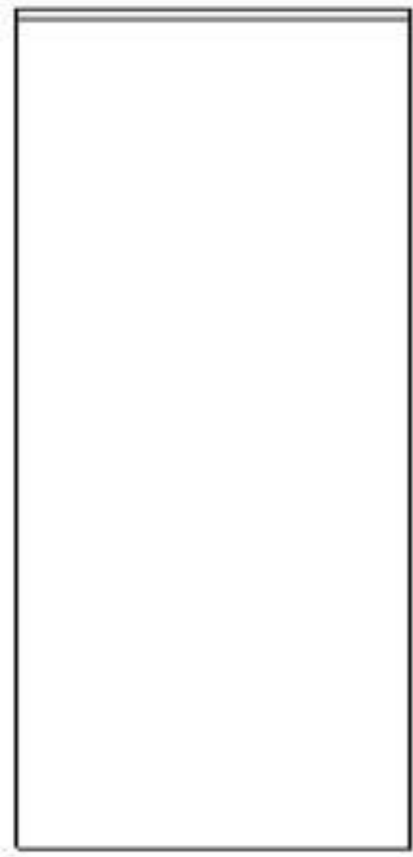


Figure 43: Vue de gauche avant

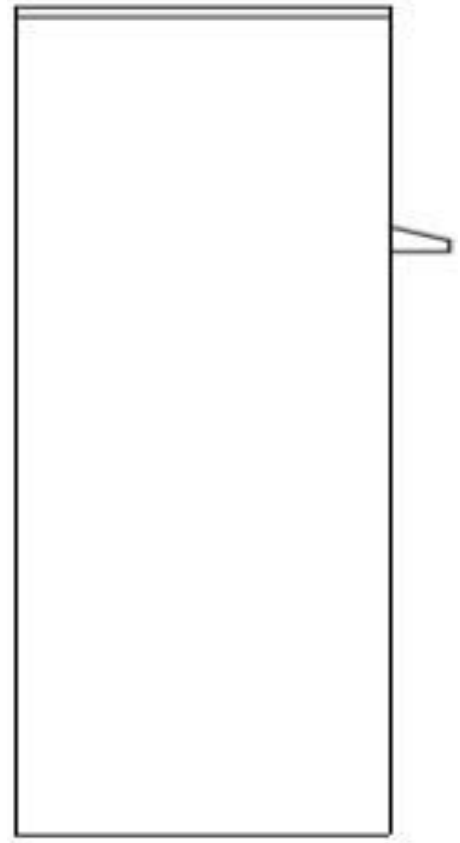
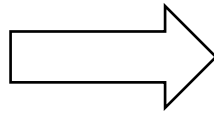


Figure 42: Vue de gauche à présent

3.4.2. Réalisation du distributeur

Le boîtier de distributeur de boisson existe déjà et il fait en bois de sapin. Alors notre but est de réaliser ce qui est amélioré. On ajoute un emplacement de verre sur la partie droite du boîtier et deux boutons-poussoirs au-dessous de l'écran. Ce dernier est relié au Raspberry Pi à l'aide de pins. D'après le premier projet, les boissons sont contenues dans les bocal plastique. Donc on est obligé d'ajouter le nombre de bocal pour mettre les boissons chaudes et la poudre de sucre. Dans le cas de sucre, il nous faut un électro-aimant pour bloquer et débloquer la poudre de sucre vers le tuyau vers le verre. Il faut augmenter le nombre de trous pour faire entrer le tuyau relié au bocal. Il y a aussi une bouilloire dans le boîtier pour faire chauffer la boisson chaude quand elle se refroidit. On découvre qu'on emploie un électro-aimant de 12V dans ce projet. Donc on doit utiliser un transformateur AC-DC ajustable dont la valeur est 12V.

Conclusion

Nous avons vu tout ce qui est amélioré et les fonctionnalités ajoutées dans ce dernier chapitre. Et sans oublier aussi ses fonctionnements et ses réalisations. À condition que tous les matériels que nous avons vus au premier chapitre soient au complet.

CONCLUSION GENERALE

Le but de ce travail est d'améliorer un smart distributeur de boisson et de réaliser ce qui est amélioré. Et sa réalisation nous a menés à bien comprendre des codes de programmation et de connaître plus des matériels et des technologies à savoir. Au début on n'en a que des simples boissons fraîches et maintenant on a des boissons chaudes et des cocktails. Alors, on aura besoin de bouilloire pour chauffer l'eau des boissons chaudes quand il est en train de se refroidir. On a un distributeur de sucre aussi dans ce boîtier. On plante aussi un système d'aide aux malvoyants suivi d'une assistance vocale pour secourir les hommes aveugles à faire ses commandes. Mais pour l'exécution et l'interprétation le programme, on doit utiliser le Raspberry pi, le langage de programmation python et kivy. Vu que dans ce progrès, certains processus se déclenchent systématiquement comme l'assistance vocale. Et il y a aussi des relais et l'électro-aimant qui y jouent de rôles importants dans le contrôle de sucre à verser, donc il faut bien programmer le Raspberry pi.

Pourtant ce progrès n'est pas encore fini, puisqu'on peut toujours pratiquer plus d'amélioration comme un distributeur de glaçon ou une commande à distance.