Alexis DORIAN 15/12/2018

Rapport final de projet de fin d'études

Projet 28 : Clonage d'une calculatrice open-source NumWorks

Encadrants :
Xavier REDON
Alexandre BOÉ
Thomas VANTROYS

Remerciements

Je tiens tout d’abord à remercier monsieur Xavier REDON pour m’avoir proposé ce projet de fin d’étude et pour son aide tout au long du projet.

Ensuite, je tiens à remercier tout particulièrement monsieur Thierry FLAMEN pour ses conseils et son aide.

Merci également à l’équipe du *Fabricarium* et *Robotech* pour l’accueil et la mise à disposition du matériel ainsi qu'à monsieur Clément FROISSART pour le partage de ses connaissances sur les différentes machines.

Table des matières

1. Contexte du Projet....4
1.1. « Mode examen » des calculatrices....4
1.2. Calculatrice Numwork....4
1.3. Objectif du projet....5
1.4. Liste Matériel et Logiciel....5
2. Travail effectué....7
2.1. État de l'art....7
2.2. Partie Logicielle....8
2.2.1. Mode examen....8
2.2.2. Mémoire python....11
2.2.3. Calcul Formel....12
2.2.3. Utilisation de Flash externe....13
2.2.4. Modifications Esthétiques....16
2.2.5. Utilisation de Dfu_web....17
2.2.6. Logiciel de mise à jour facilité....18
2.2.7. Modification passe temps....19
2.3. Partie Hardware....20
2.3.1. Pièce sur mesure....20
2.3.2. Boîtier....21
2.3.3. Ajout Flash Externe....24
2.3.4. Réalisation PCB....25
2.3.5. Clavier....26
3. Difficultés rencontrées / Remarques....27
3.1. Partie logicielle....27
3.2. Partie Hardware....27
3.3. Générale....27
4. Suite....28
4.1. Travail restant....28
4.2. Améliorations possibles....28
Conclusion....29

1. Contexte du Projet

1.1. « Mode examen » des calculatrices

À partir du 1er janvier 2018, dans toutes les épreuves où la calculatrice est autorisée à l'examen, et si le candidat possède une calculatrice ayant une mémoire alphanumérique et/ou avec écran graphique, la calculatrice devra disposer d'une fonctionnalité « *mode examen* ».

Le mode examen doit disposer des fonctionnalités suivantes :

- la neutralisation temporaire de l'accès à la mémoire de la calculatrice ou l'effacement définitif de cette mémoire ;

- le blocage de toute transmission de données, que ce soit par wifi, Bluetooth ou par tout autre dispositif de communication à distance ;

- la présence d'un signal lumineux clignotant sur la tranche haute de la calculatrice, attestant du passage en « *mode examen* » ;

- la non réversibilité du « *mode examen* » durant toute la durée de l'épreuve. La sortie du « *mode examen* » nécessite une connexion physique, par câble, avec un ordinateur ou une calculatrice.

1.2. Calculatrice Numwork

Pour répondre à cette nouvelle directive de l’éducation nationale, l’entreprise *Numwork* a réalisé une calculatrice open-source comme demandée par l’éducation nationale. Cette calculatrice possède un mode examen ayant pour fonctionnalités :

- d'effacer la mémoire de la calculatrice ;

- d'allumer une LED rouge et d'afficher une icône examen sur la calculatrice ;

- de demander la présence d’une alimentation par câble USB pour sortir du mode examen ;

Chose intéressante : cette calculatrice, bien qu’ayant un prix d’achat intéressant par rapport aux autres calculatrices, n’est pas moins puissante puisque qu’elle embarque un STM32F412VGT6 (à 100 Mhz) alors que son homologue la Ti 83 n’embarque qu’un Zilog Z80 (à 48 Mhz).

1.3. Objectif du projet

Le projet se décompose en deux parties distinctes :

Partie logiciel :

- Modification du logiciel pour falsifier un mode examen ;
- Modification du logiciel pour augmenter la mémoire allouée à python ;
- Modification du logiciel pour rajouter du calcul formel ;
- Diverses modifications pour utiliser au mieux la performance de la calculatrice.

Partie Hardware :

- Réalisation un clone physique de la calculatrice ;
- Modification d'une calculatrice originale pour en augmenter sa puissance.

1.4. Liste Matériel et Logiciel

Je décomposerai les listes matériel et logiciel si une personne souhaite reprendre certaines partie de ce projet dans le futur.

Partie logicielle :

Matériel :

Câble Micro-USB/USB A (usb classique)

Logiciel :

Linux :

bison build-essential dfu-util flex gcc-arm-none-eabi libfltk1.3-dev libfreetype6-dev libpng12-dev

Windows :

mingw-w64-x86_64-gcc mingw-w64-x86_64-freetype mingw-w64-x86_64-pkg-config mingw-w64-x86_64-fltk git make bison python

Partie Hardware :

Matériel :

AT25SF641 (Flash externe)

STM32F412VGT6 (micro-contrôleur)

ET024QV01-K (écran)

connecteur nappe (30 pin FPC mais celui utilisé est un connecteur nappe 40 Pin dû au changement d'ecran)

RT9365GQW (contrôle rétro éclairage)

RT9078-28GJ5 (régulateur de tension)

RT9526AGE (chargeur lipo)

USBLC6-2SC6 (protection USB)

TST-S310F2KT (LED multi couleur)

Oscillateurs (25Mhz)

Logiciel :

Altium designer : Conception de carte electronique

Inkscape : Conception de boîtier découper

Onshape : Visualisation et modification modèle existant a imprimer en 3D

Cura : Prépare le modèle 3D pour l'imprimante 3d

2. Travail effectué

2.1. État de l’art

Un petit état de l’art a été réalisé afin de déterminer quels sont les différents concurrents de la calculatrice *Numwork* et quels sont ses avantages et ses inconvénients.

Dans cette gamme de prix, ses deux principaux concurrents sont la *TI 83* de *Texas instrument* et la *Casio graph 35+* de *Casio*.

On remarque vite que la conception de la *Numwork* a été très bien faite pour disposer des avantages de la concurrence sans en avoir les désavantages :

- elle permet de programmer en python est non pas dans une langue propriétaire comme la TI ;

- elle possède un écran couleur de une bonne résolution comme la TI au lieu d’un écran base qualité noir et blanc comme la Casio ;

- elle possède le deuxième meilleur CPU par rapport à ses concurrents, la Casio étant la première ;

- nativement, c’est elle qui a le moins de mémoire flash (1Mo contre 3Mo pour TI et 4Mo pour Casio). Cependant, elle dispose d’un emplacement permettant de rajouter une flash externe ;

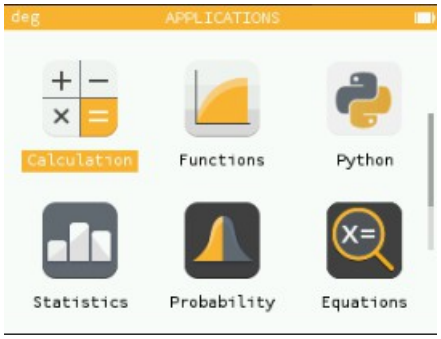
- elle possède plus de RAM, chose importante pour l’utilisation de certains programmes et une utilisation graphique.

Mais le plus gros avantage qu'elle possède c'est qu'elle est totalement open-source (sous certaines conditions).

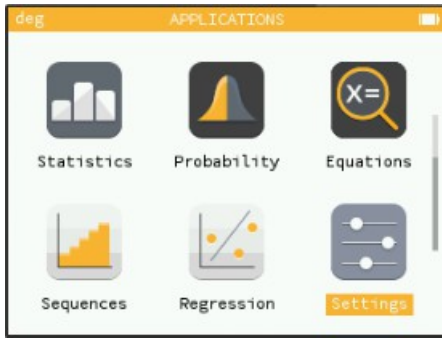
2.2. Partie Logicielle

2.2.1. Mode examen

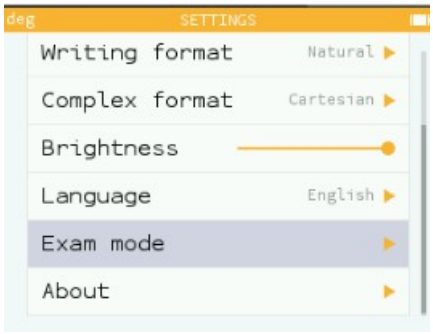
La première tâche du travail était de modifier le mode examen de la calculatrice. Dans un premier temps, nous avons étudié le fonctionnement normal de la calculatrice : ils ont fait le choix d'utiliser un reset mémoire pour effacer les script python et aussi tout ce qui a pu être rentré comme des variables et/ou fonctions. Efficace, mais tout les programmes python sont effacés à vie : il faut donc penser à les sauvegarder avant chaque examen.



Menu de Base



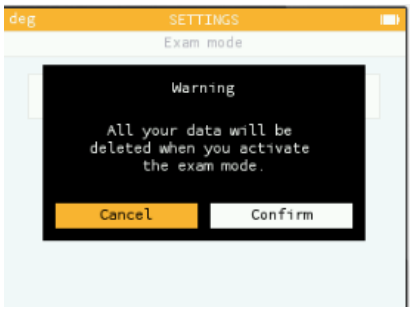
Application Settings



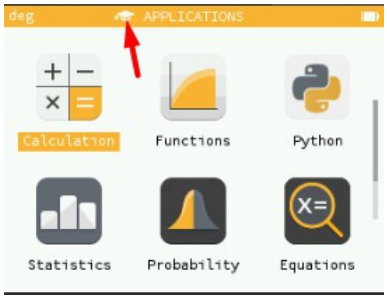
Mode Exam

Le mode examen se trouve dans l'application « *Settings* ».

Une fois activé, la calculatrice prévient qu'elle va effacer toutes les données avant de passer en mode examen.



Pop-Up d'avertissement



Mode Exam Actif

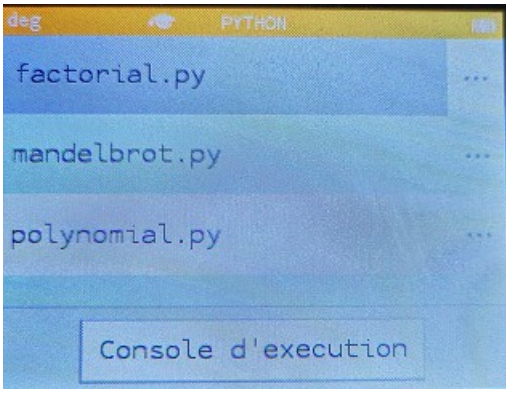
On peux vérifier que le mode examen a bien fonctionné.

Script Python vide

On voit bien que les scripts python ont été effacés de la mémoire de la Numwork et une petite LED rouge clignote à travers le boîtier.

J'ai développé plusieurs moyens de contourner le mode examen : cependant, ils ont des avantages mais aussi des inconvénients lors de la mise en place et lors de son utilisation .

La première méthode consiste à tout simplement modifier le mode examen et retirer les lignes qui provoquent le reset de la mémoire. La méthode est simple et peut être mise en place assez facilement par une personne qui connaît l'informatique et l'environnement de la Numwork, mais hélas, pas discret : le surveillant a juste à aller vérifier que les fichiers bien étaient supprimés pour remarquer qu'il y a un problème.



Programme bien concerver malgré le mode exam

La seconde méthode est basée sur le fait que lorsqu'on relance la calculatrice \Numwork en sortant du mode examen les programmes de base soient bien présents.

Programmes de base

Pourtant la Numwork efface bien les scripts et ne possèdant pas de gestionnaire fichiers il doivent bien être stocker en dur quelque part. C'est en fouillant dans les fichier de Numwork que j'ai pu retrouver les trois scripts de base écrits en dur dans un fichier.

```
constexpr ScriptTemplate fibonacciScriptTemplate("fibonacci.py", "\x01" R"(def fibo(n):
  a=0
  b=1
  for i in range(1,n+1):
    c=a+b
    a=b
    b=c
  return a

def fibo2(n):
  if n==0:
    return 0
  elif n==1 or n==2:
    return 1
  return fibo2(n-1)+fibo2(n-2))");
```

Un script de base

Puis, lorsqu'on relance la calculatrice hors du mode examen, celle-ci n'a plus qu'à appeler ses scripts pour les recréer à leur place.

```
ScriptStore::ScriptStore()
{
  addScriptFromTemplate(ScriptTemplate::Factorial());
  addScriptFromTemplate(ScriptTemplate::Mandelbrot());
  addScriptFromTemplate(ScriptTemplate::Polynomial());
}
```

Création Script de base

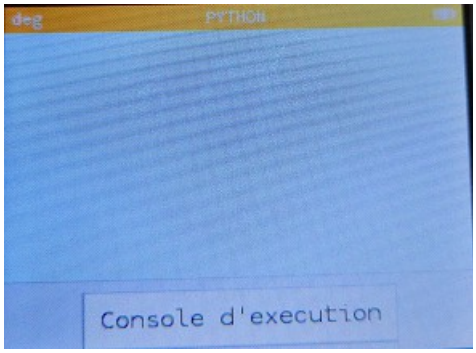
Ces fonctions peuvent être appelées n'importe où. J'ai donc créé une fonction qui, une fois qu'on appuie sur la touche gauche de la calculatrice dans le menu de python, permet aux scripts de base de retourner en mode examen.

Pour le tricheur, il suffit juste de modifier un Script de base ou en ajouter un nouveaux avec les réponses voulues.

Cette méthode, à contrario de la première est plus dure à mettre en place mais plus efficace et on peux imaginer que l'appui (ou combinaison) peux faire apparaître ou disparaître tel ou tel script au besoin de l'étudiant selon ce qu'il a besoin et pour rester le plus discret possible au cas ou un surveillant ait envie de vérifier que la calculatrice soit bien vide,

Suite à la réalisation de la deuxième méthode, j'ai décider de réaliser une dernière méthode basée sur la première.

Cette méthode, tout comme la première, évite le reset de la calculatrice. Mais pour la rendre plus discrete, le fait de désactiver la fonction d'affichage des script python provoque un petit bug graphique.



Bug graphique

La calculatrice affiche dans la même fonction les scripts et le bouton « ajouter un script » (qui en fait le cas par défaut du script vide) : les bugs graphiques ne restent pas moins mineurs par rapport a avant.

Comme ça, il existe trois méthodes pour contourner le mode examen qui sont plus ou moins difficiles à mettre en place sachant que plus on fait de grosse modification plus c'est discret (la 2 étant totalement indétectable sans connaître la séquence de touches ou vérifier le checksum de l'os).

2.2.2. Mémoire python

Dans un premier temps il faut mesurer combien python dispose de place sur les 256kb de RAM présente. Pour cela j'ai utilsé un petit script python développé par une personne d'une communauté.

```
def mem():
  try:
    l=[]
    try:
      l+=[0]
      l+=[""]
      l[1]+="x"
      while True:
        try:
          l[1]+=l[1][l[0]:]
        except:
          if l[0]<len(l[1])-1:
            l[0]=len(l[1])-1
          else:
            raise(Exception)
    except:
      print("+",len(l)>1 and len(l[1]))
      return 64+8*len(l)+(len(l) and 24+4*(l[0]>0)+(len(l)>1 and 49+len(l[1])))+mem()
  except:
    return 0
```

Script de mémoire

Le script nous ressort que nous pouvons utiliser dans les 6kb de ram : dans les faits, il y a 16kb de ram allouée.

En modifiant les fichiers, on peux augmenter la taille et après plusieurs tests, on peut monter dans les 128kb de mémoire mais une fois qu'on dépasse les 64kb, la calculatrice met un temps fou à lancer le petit script. J'ai pensé plusieurs fois que la calculatric avait planté. Mais après lecture du code et expérimentation, on remarque que la console ne peux pas être lancée si on alloue trop de place par rapport à ce qui esr disponible. La calculatrice alloue directement un bloc mémoire pour python et fait un test simple Memoire Libre = Memoire disponible si c'est faux la console n'est même pas lancer pour éviter le crash.

2.2.3. Calcul Formel

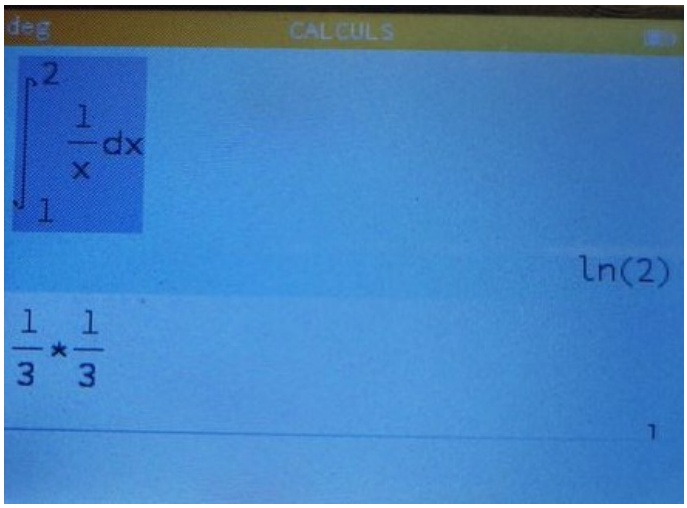
Pour rajouter le calcul formel, le plus simple est de compiler l'application math avec Giac

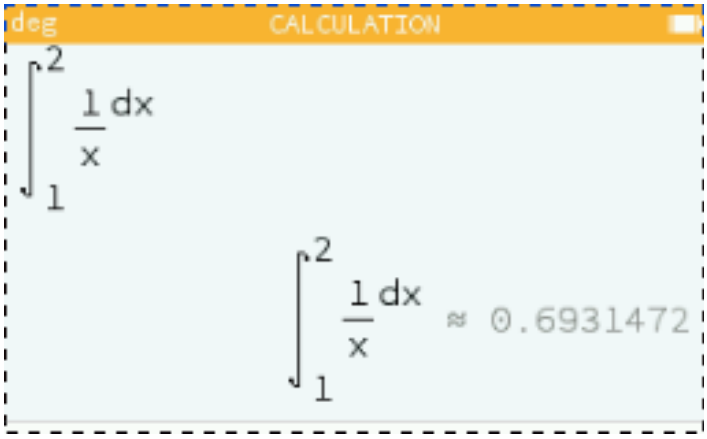
Il est physiquement impossible de rajouter Giac dans un numwork non modifié ! Giac a lui seul prend plus de 1 Mb pour pouvoir intègre Giac a une numwork il faut soit utiliser un numwork modifié avec un puce de flash rajouter (voire section suivante) ou soit il faut rester sur le simulateur

Toute la partie logicielle qui suit a été réalisée sur un numwork modifié et ne traitera que du calcul formel en lui même.

J'ai utilisé une version 1.4.9 de Giac et le début des travaux de zardam pour cette partie.

Le code original et dans une ancienne version de l'os de numwork mais est quand même fonctionnel





Une simple comparaison entre un calcul réalisé avec la calculatrice et le simulateur suffit à montrer que le calcul formel fonctionne bien !

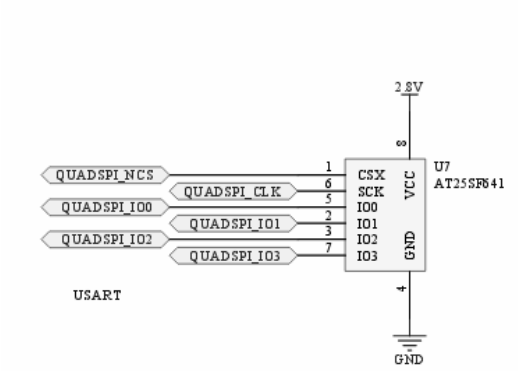
Mettre à jour la partie application math pour utiliser le calcul formel en elle même n'est pas dur en soit le vrai problème vient de l'utilisation d'une flash externe (Partie développée dans la suite).

2.2.3. Utilisation de Flash externe

Comme dit dans la section précédente l'utilisation d'une flash externe n'est pas triviale pour la calculatrice bien que la place a été prévue sur la carte, mais niveau logiciel il n'y a rien .

Attention, dans cette partie on parle de manipulations pouvant être dangereuses pour la flash externe rajoutée et même la calculatrice entière.

Pour écrire sur la flash externe, j'utilise un programme appelé *QSPI_loader* qui permet comme son nom l'indique de charger un programme via le QSPI.



Branchement AT25SF641

QSPI_loader a un petit utilitaire qui fonctionne avec le paquet flashrom qui permet tout simplement de savoir comment flasher n'importe quelle RAM. Cependant, dans mon projet, la puce que nous utilisons et un peu particulière : elle n'est pour le moment pas nativement connue dans la paquet il suffit de rajouter quelques lignes.

```
{
    .vendor     = "Adesto",
    .name       = "AT25SF641",
    .bustype    = BUS_SPI,
    .manufacture_id = ATMEL_ID,
    .model_id   = ATMEL_AT25SF641,
    .total_size = 8192,
    .page_size  = 256,
    .feature_bits   = FEATURE_WRSR_WREN | FEATURE_OTP,
    .tested     = TEST_UNTESTED,
    .probe      = probe_spi_rdid,
    .probe_timing   = TIMING_ZERO,
    .block_erasers  =
    {
        {
            .eraseblocks = { {4 * 1024, 8192 / 4} },
            .block_erase = spi_block_erase_20,
        }, {
            .eraseblocks = { {32 * 1024, 8192 / 32} },
            .block_erase = spi_block_erase_52,
        }, {
            .eraseblocks = { {64 * 1024, 8192 / 64} },
            .block_erase = spi_block_erase_d8,
        }, {
            .eraseblocks = { {8 * 1024 * 1024, 1} },
            .block_erase = spi_block_erase_60,
        }, {
            .eraseblocks = { {8 * 1024 * 1024, 1} },
            .block_erase = spi_block_erase_c7,
        }
    },
    .printlock  = spi_prettyprint_status_register_plain,
    .unlock     = spi_disable_blockprotect,
    .write      = spi_chip_write_256,
    .read       = spi_chip_read,
    .voltage    = {2700, 3600},
},
```

Ligne a rajouter

Grâce a ses lignes QSPI_loader va savoir quelle taille font chaque blocs et mémoires et comment il sont repartis une fois QSPI_loader bien configuré, et il peux être charger dans la calculatrice comme si c'était une mise a jour classique a se moment la on peux vérifier que la calculatrice est bien repérée avec le QSPI_loader de charger (on peux utiliser la page de test DFU_usb).

Une fois QSPI_loader bien chargé, il faut « patcher » la flash car de base elle est protégée en écriture pour éviter qu'elle se déprogramme toute seule pendant un fonctionnement normal il suffit juste de change le bit QE et le tour et jouer .

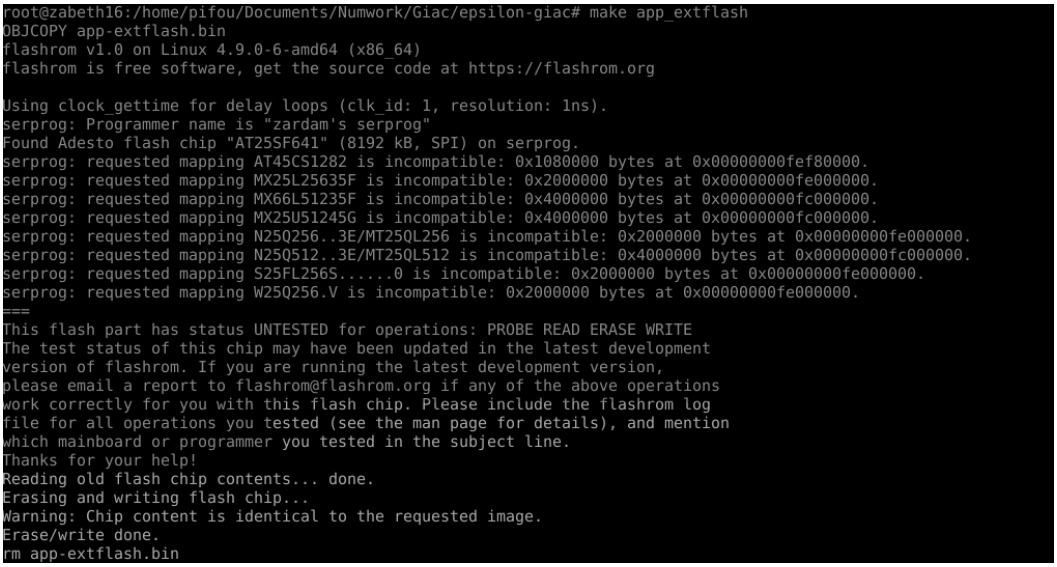
Maintenant que toutes les étapes été réalisées, la flash externe est prête à être utilisée .

Toutes ces étapes longues seront à refaire a chaque fois qui faudra utiliser la flash.

Maintenant partie OS de la calculatrice : il faut lui ajouter quel que fichier dans sont noyaux pour qu'il sache utiliser la flash externe

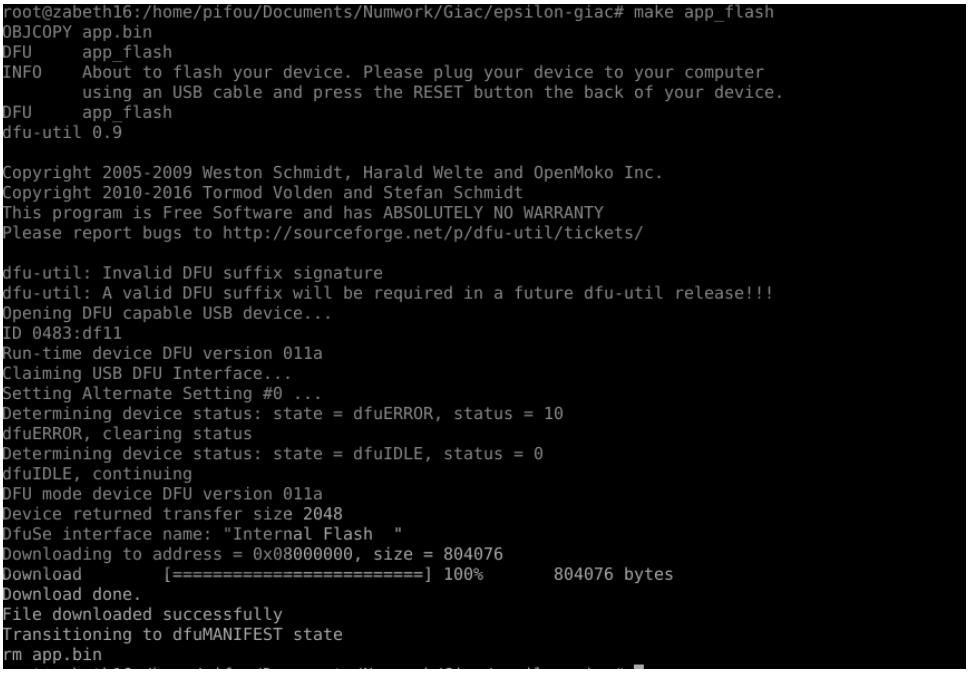
```
void initGPIO() {
  // Configure GPIOs to use AF
  GPIOB.MODER()->setMode(2, GPIO::MODER::Mode::AlternateFunction);
  GPIOB.MODER()->setMode(6, GPIO::MODER::Mode::AlternateFunction);
  GPIOC.MODER()->setMode(8, GPIO::MODER::Mode::AlternateFunction);
  GPIOC.MODER()->setMode(9, GPIO::MODER::Mode::AlternateFunction);
  GPIOD.MODER()->setMode(12, GPIO::MODER::Mode::AlternateFunction);
  GPIOD.MODER()->setMode(13, GPIO::MODER::Mode::AlternateFunction);

  // Set GPIOs AF
  GPIOB.AFR()->setAlternateFunction(2, GPIO::AFR::AlternateFunction::AF9);
  GPIOB.AFR()->setAlternateFunction(6, GPIO::AFR::AlternateFunction::AF10);
  GPIOC.AFR()->setAlternateFunction(8, GPIO::AFR::AlternateFunction::AF9);
  GPIOC.AFR()->setAlternateFunction(9, GPIO::AFR::AlternateFunction::AF9);
  GPIOD.AFR()->setAlternateFunction(12, GPIO::AFR::AlternateFunction::AF9);
  GPIOD.AFR()->setAlternateFunction(13, GPIO::AFR::AlternateFunction::AF9);
}

void initQSPI() {
  QSPI.CR()->setPRESCALER(11); // 8MHz QSPI frequency
  QSPI.DCR()->setFSIZE(23);  // Number of bytes in Flash memory = 2^[FSIZE+1] (16 MBytes)
  QSPI.CCR()->setFMODE(3); // Memory mapped mode
  QSPI.CCR()->setDMODE(3); // Quad data lines
  QSPI.CCR()->setDCYC(8); // 1 dummy byte
  QSPI.CCR()->setADSIZE(2); // 24 bits address
  QSPI.CCR()->setADMODE(1); // Address on a single line
  QSPI.CCR()->setIMODE(1); // Instruction on a single line
  QSPI.CCR()->setINSTRUCTION(0x6B); // "Quad Output Fast Read" from the flash
  QSPI.CR()->setEN(true); // Enable QSPI Controller
}
```

Selon la qualité de la soudure réalisée, il peut être nécessaire de baisser la vitesse d’accés a la flash sinon elle fait tout planter. Maintenant, le noyau de la calculatrice est enfin compatible et peut utiliser la flash externe. Le fichier voulu peut ensuite être compilé en deux parties : la partie destinée à la flash externe et l'autre, à la flash interne.



flash externe

Ensuite, la flash interne.



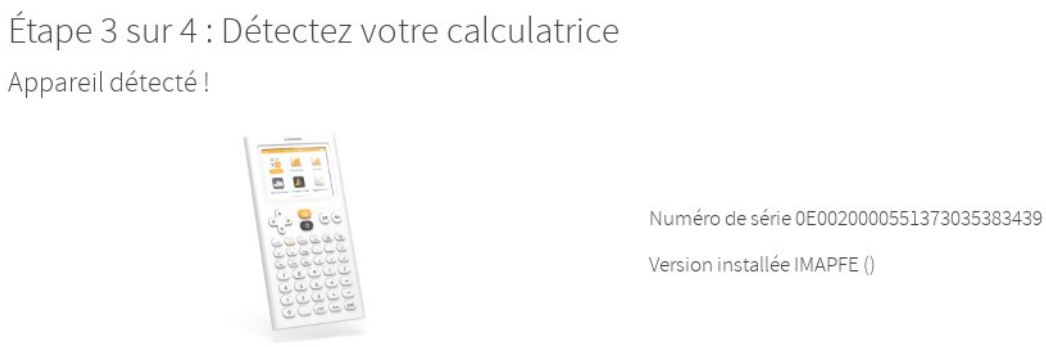
flash interne

La calculatrice est maintenant fonctionnelle avec les nouveaux OS chargés quels qu'ils soient

2.2.4. Modifications Esthétiques

Il est possible de faire plusieurs modifications esthétiques dans la calculatrice pour la rendre plus personnelle :

Modifier la version pour que sa marque autre chose :

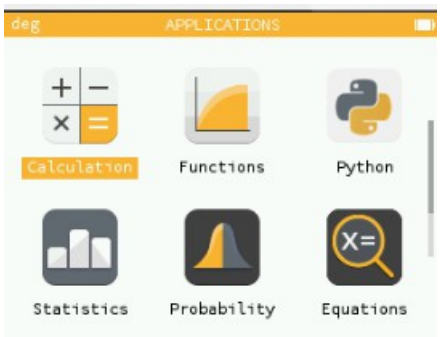


Modifier la couleurs de led dans le mode examen si le rouge ne nous conviens pas ou changer sa fréquence

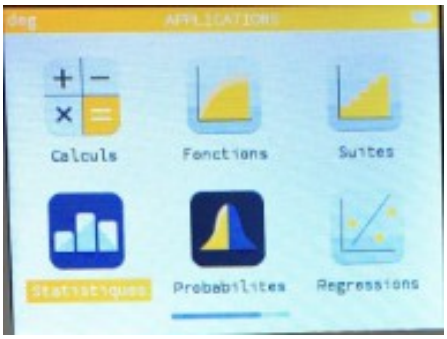
```
Ion::LED::setColor(KDColorRed);
Ion::LED::setBlinking(1000, 0.1f);
```

Modifier chaque icône de chaque application (ou mode examen).

Modifier l'affichage général des applications, dans les anciennes versions de la calculatrice l'affichage était « téléphone » on se déplacer de gauche a droite et dans les nouvelle version l'affichage est de haut en bas



Nouveaux

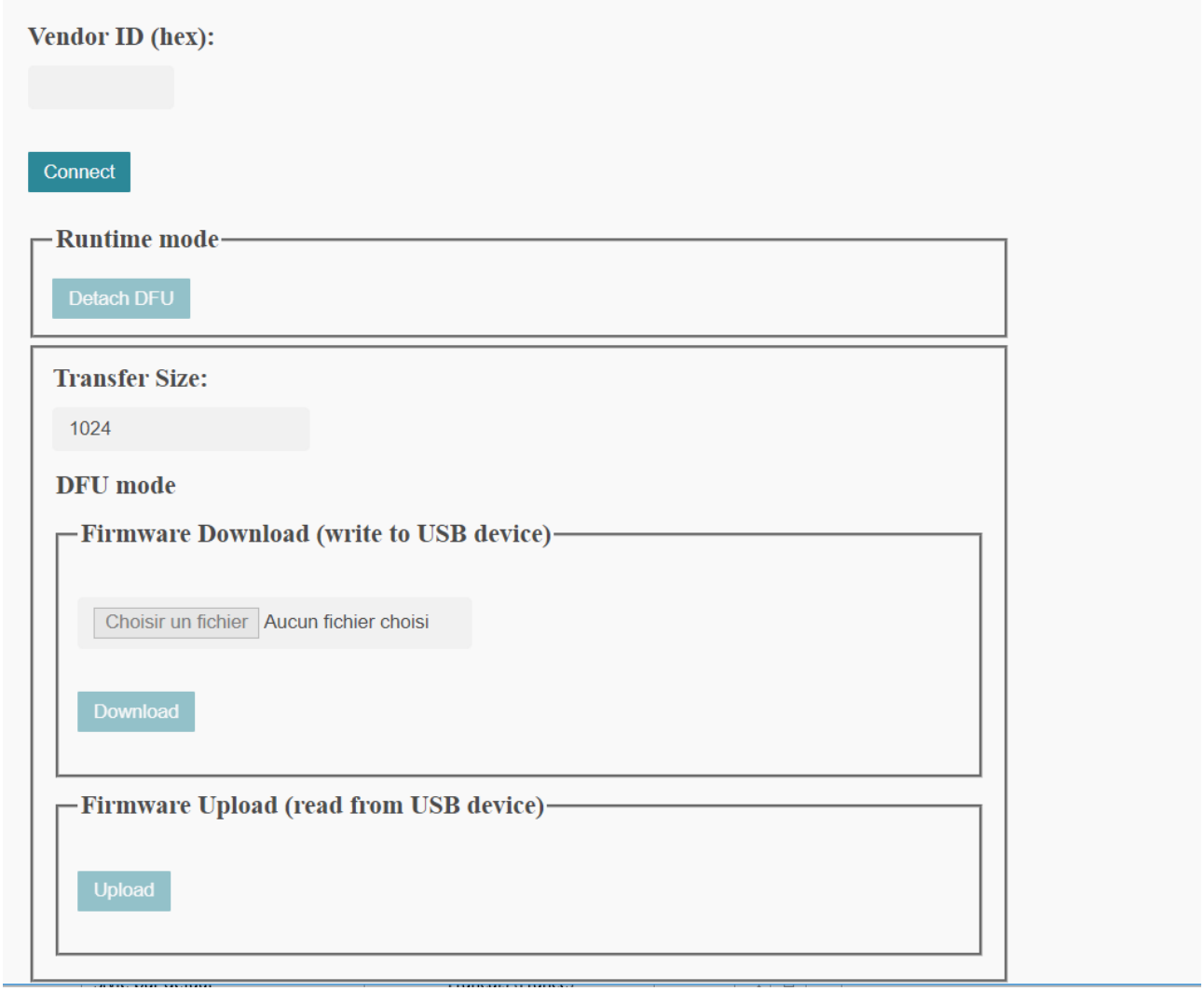


Ancien

Préfèrant l'ancien affichage type « téléphone », je l'ai mis sur les dernières versions. Je le trouve plus naturel

2.2.5. Utilisation de Dfu_web

Dfu_web est un paquet qui permet d'utiliser Dfu via un navigateur ; c'est d'ailleurs la solution choisie par numwork pour pouvoir mettre à jour facilement ses calculatrices sans devoir compiler soit même l'os et l'importer dans la calculatrice en utilisant la page test de base il est possible de créer un petit serveur web qui fait tourner la page et qui permet donc de récupérer une version d'une calculatrice et la mettre sur une autre (ne marche pas si utilisation de la flash externe).



Page de test

En complexifiant la page, il est possible de charger toujours la même version et donc de faire un site qui permet de mettre à jour n'importe quelle calculatrice avec la version voulue.

En utilisant cette page, il est donc possible pour un professeur de permettre à ses élèves de mettre à jour leurs calculatrices plus facilement avec la même mise à jour, ou à une personne mal-attentionnée de répandre une version contournant la mode examen.

2.2.6. Logiciel de mise à jour facilité

Comme nous l'avons vu dans la partie utilisation de la flash externe, il est compliqué et fastidieux de faire à chaque fois la mise à jour de la calculatrice (et surtout dangereux si on rate les manipulations liées à la flash externe). Donc, j'ai créé un simple utilitaire qui décrit chaque étape de fonctionnement de la mise à jour et quelles manipulations doivent être faites physiquement.

```
//chargment du qspi loader
if((system("make -C ./qspi_loader-master/qspi_loader/ run_qspi_loader"))!=0)
{
        printf("Error DFU\n");
        return(0);
}
sleep(1);
//patche de la flash
if((system("./qspi_loader-master/tools/enable_qspi_AT25SF641.py "))!=0)
{
        printf("Error Script py \n");
        return(0);
}
sleep(1);
//instalation de la rom externe (on la re compile avant au cas ou)
system("make -C ./Giac_maj/epsilon-giac/ clean");
system("make -C ./Giac_maj/epsilon-giac/");
sleep(1);
system("make -C ./Giac_maj/epsilon-giac/ app_extflash");
//action de l'utilisateur demander il faut appuyer sur reset pour installer la flash interne
printf("appuyer sur le bouton reset de la calculatrice\n");
printf("une fois fait appuyer sur n'importer quel touche du clavier et faite entrée\n");
scanf("%d");
system("make -C ./Giac_maj/epsilon-giac/ app_flash");
printf("Fin\n");
return(0);
```

Le script est basique et pas parfait, mais il permet d'éviter de faire toutes les commandes à la main et surtout d'oublier une étape dans le processus qui pourrait nuire au bon fonctionnement de la calculatrice.

Il n'y pas de vérification que le code compile bien car il est fait pour réaliser une mise à jour qui marche et qui est fonctionnelle.

2.2.7. Modification passe temps

Il est possible de modifier logiciellement la Numwork pour y intégrer un émulateur NES qui se rajoutera en application comme les autres dans la calculatrice.

Emulateur en fonctionnement sur une vrai calculatrice

Hélas, la calculatrice n'embarquant pas de gestionnaire de fichier, on ne peut donc avoir qu'un seul jeux à la fois .

On ne peut aussi mettre n'importe quel jeu vu qu'il n'y a plus de place de base dans la calculatrice (de base) ; il faut soit sacrifier des applications, soit ne mettre que des petits jeux

Bien sûr, avec une calculatrice ayant une flash externe, le problème de ne se pose pas. Cette modification n'est purement gadget mais elle met en avant quand même deux choses : la facilité de rajouter une application et on peut donc imaginer facilement une application de dessin rajoutée par des étudiants ou un mini dictionnaire des expressions /formules mathématiques. Tout cela montre de nouveau la puissance de la calculatrice.

2.3. Partie Hardware

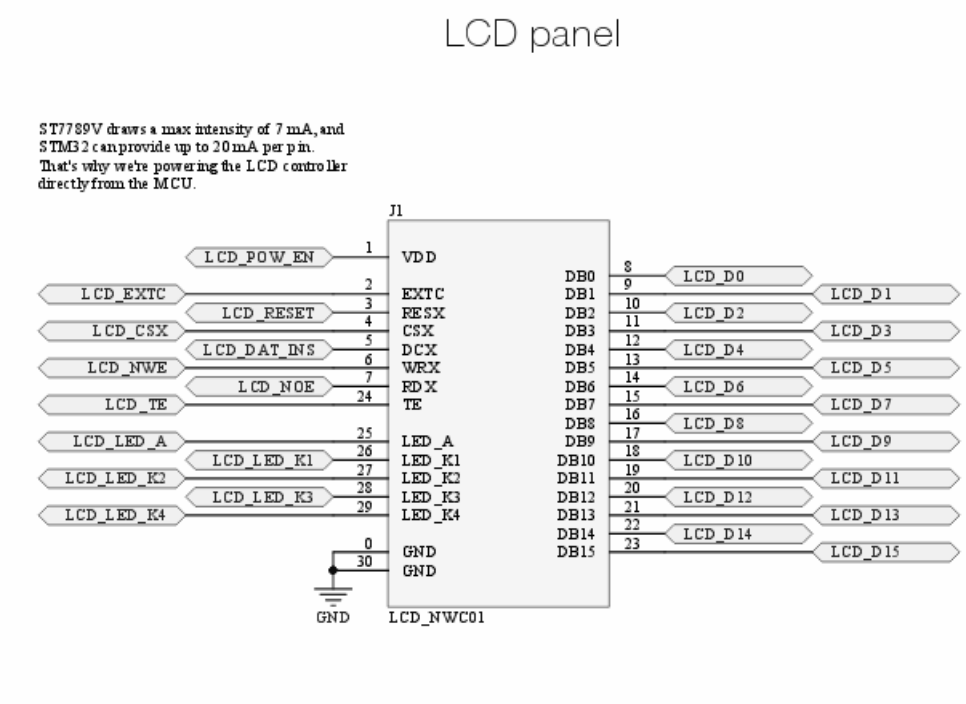
2.3.1. Pièce sur mesure

Bien que open-source, la Numwork utilise plusieurs pièces faites sur mesure qui complique bien la tâche quand il s'agit de la refaire comme les composants suivants :

- la batterie ;
- le clavier ;
- l'écran.

Et c'est sur ce dernier que l'on va s'attarder car il est facile de retrouver un alternatif aux deux premiers mais les choix faits sur l'écran sont vraiment spécifiques et Numworks ne souhaite pas communiquer chez qui l'écran a été commandé, même en ayant contacté un grand nombre de fabricants d'écrans : shtdo, shenzhen youritech, das, guangzhou zhihua electronic, formike, sinocrystal, zelma, cnelida et oklcm.

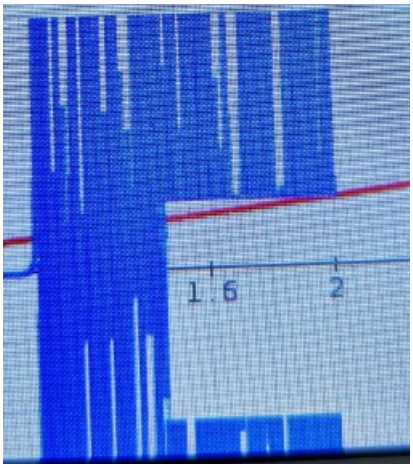
Pour ne citer qu'eux, seulement deux fabriquants avaient des écrans compatibles sinocrystal et cnelida dû au fait que cnelida ne vendait que sur alibaba (aliexpresse des professionnel). L'écran choisi fut celui de sinocrystal même s'il lui manque quelques fonctionnalités.



Les deux fonctionnalités manquantes sont les suivantes : LCD_EXTC et LCD_TE,

- LCD_EXTCe bien que reliée et configurée dans le code de la calculatrice, n'est pas encore utilisée donc pour faire une copie dans l'état actuel il peux être négligé.

- LCD_TE, quant à elle, est bien utilisée et sert à ce que l'affichage soit synchronisé dans les faits. Cette fonctionnalité n'est importante que dans les applications très gourmandes comme l'affichage d'une courbe complexe.

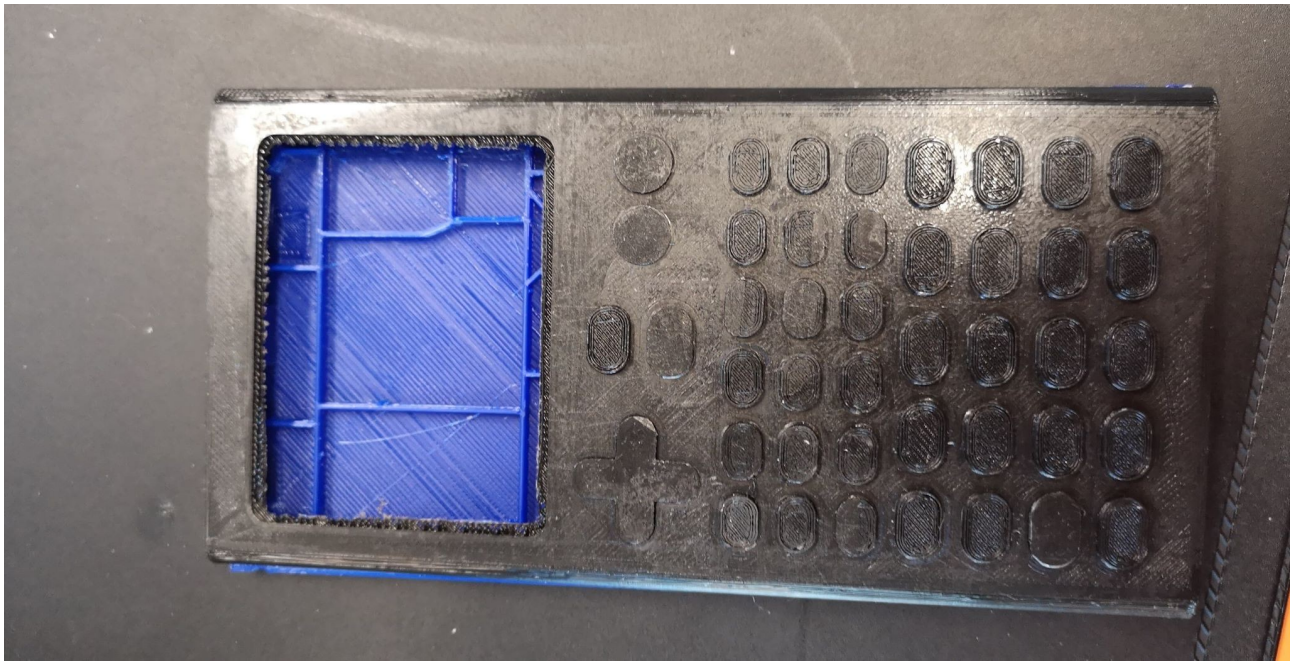


Sans le TE

Avec la fonction, la seule différence c'est que l'écran se « bloque » pour ne pas qu'on puisse voir le moment où un bout de courbe disparaît car il l'affiche. Bien qu'utile pour le confort de l'utilisation, ce n'est pas une fonction vitale qui empêchera le fonctionnement de la calculatrice.

2.3.2. Boîtier

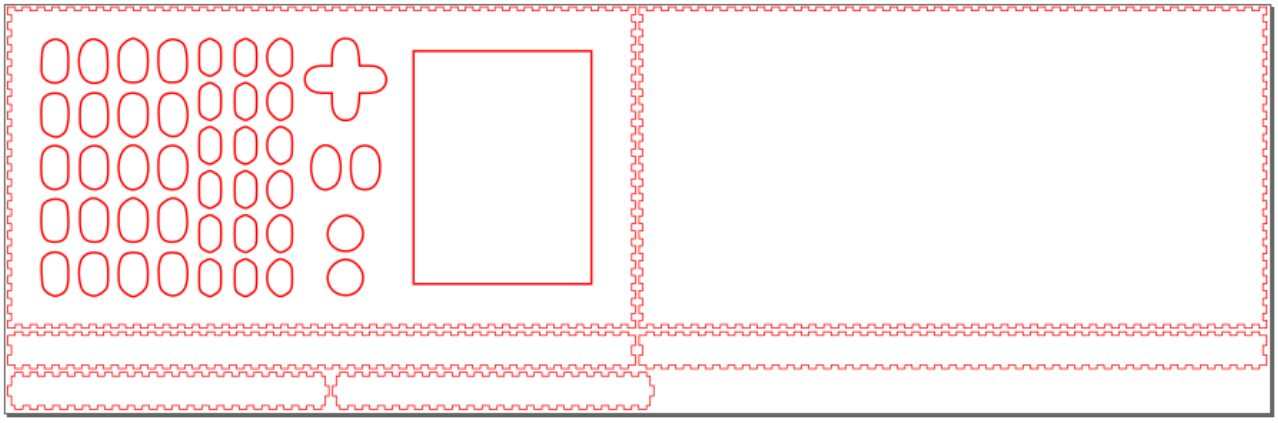
Grâce au Fabricarium, j'ai pu réaliser le boîtier de la numwork à l'imprimante 3D.



Boitier Imprimer en 3D

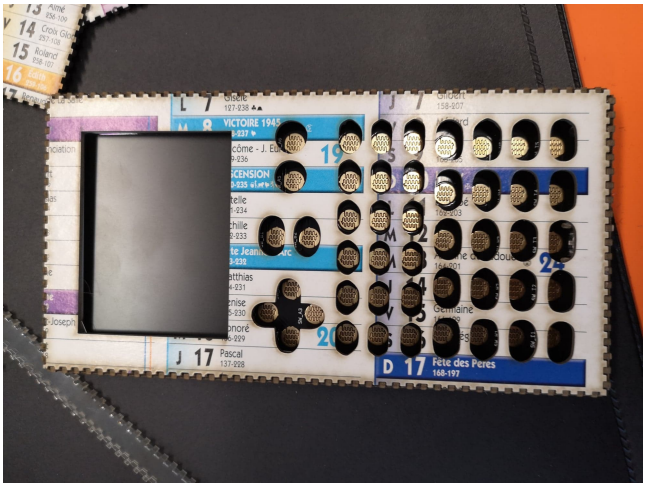
En ayant une qualité d'impression correcte, il faut 8 heures pour imprimer totalement le boîtier et cela coûte une dizaine d'euros. Il faut fait attention à deux choses pendant la réalisation : le modèle et la base prévus pour le moulage plastique. Les jeux laissés sont donc faibles ce qui résulte que les boutons sont un petit peu « gros » et qui faut les limer un minimum pour qu'ils soientt parfaitement fonctionnels (de l'épaisseur du fils soit 0,5mm).

Dans un but de baisser les coûts de la calculatrice et d'augmenter la cadence de fabrication, j'ai réalisé en parallèle un plan vectoriel de la calculatrice pour pouvoir réaliser un prototype à la découpeuse laser.



Plan svg

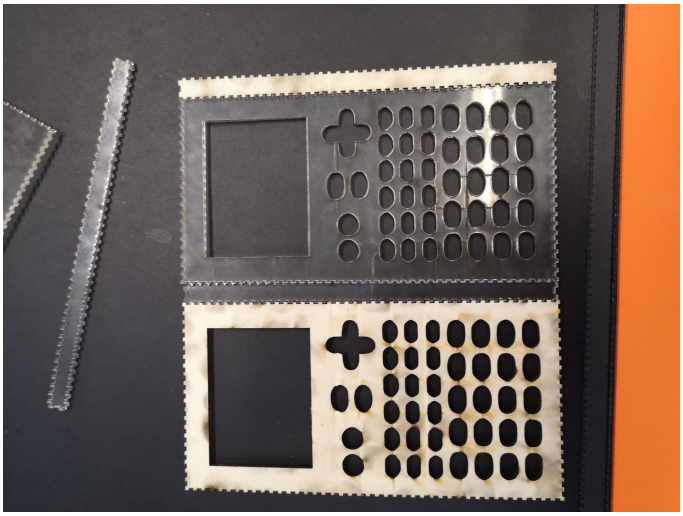
Le plan de base est prévu pour utiliser un matériaux de 1mm d'épaisseur (pour que les boutons originaux restent compatibles), mais le plan reste correct jusqu'à 3mm. La finition ne sera par contre pas parfaite et à 3mm, les boutons sont vraiment justes pour pouvoir être utilisés.



Calculatrice carton 1,5mm

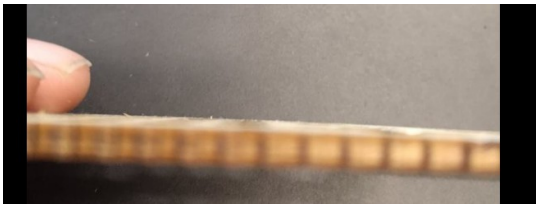
Un modèle en carton a été réalisé et comme la photo le montre il fait la bonne taille et est fonctionnel.

Deux autres modèles ont été réalisés en bois et en plexi de 3mm chacun.



Les 2 modèles

Ces modèle mettent bien en évidence le problème des touches pas à la bonne taille.



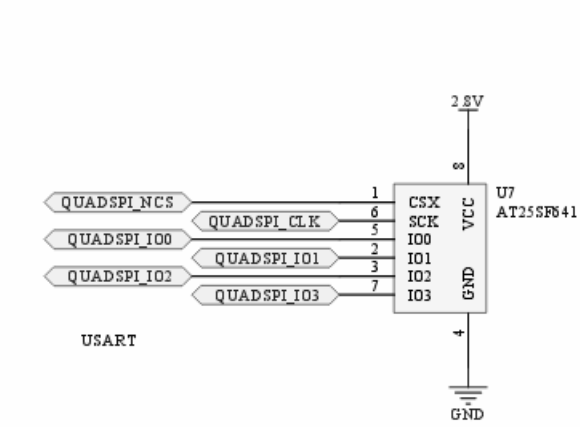
Quand on met de profil, on voit bien que la touche fait exactement la taille de la plaque de bois

La réalisation de ces boîtiers coûte bien moins cher et sont bien moins longs.

Baser sur les prix de *Leroy Merlin*, le modèle bois revient dans les 2€, le modèle plexi dans les 4€ et le modèle carton dans 0.15€ et sur le temps de fabrication autre que la mise en place de la machine chaque modèle mets moins de 5 minutes (le plexi étant le plus long car il demande plusieurs passages pour être sûr qu'il soit bien découpé).

2.3.3. Ajout Flash Externe

Grâce à la datasheet du fabricant, nous savons que Numwork préconise une AT25SF641 qui permet de rajouter 8Mb de mémoire à la Numwork (multipliant pas 9 sa mémoire totale).



Il faut faire attention en soudant une mémoire flash car c'est un composant extrêmement sensible et fragile. Ma première mémoire externe soudée à d'ailleurs rendu l'âme assez vite.

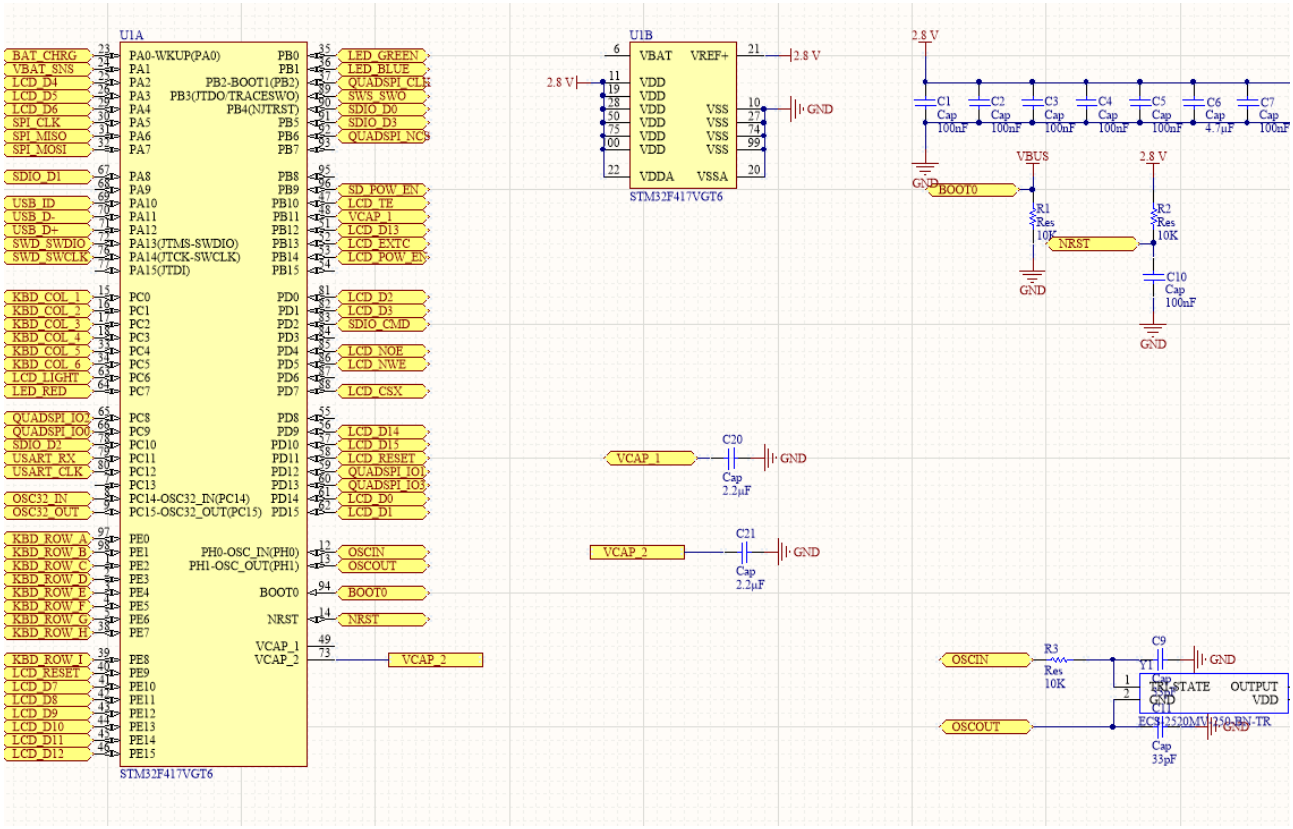
2éme mémoire flash

En soit, d'un point de vue hardware, il n'y a aucune difficulté à rajouter une mémoire flash externe sur une Numwork ; la qualité de soudure impliquera seulement sur la fréquence d'utilisation maximale.

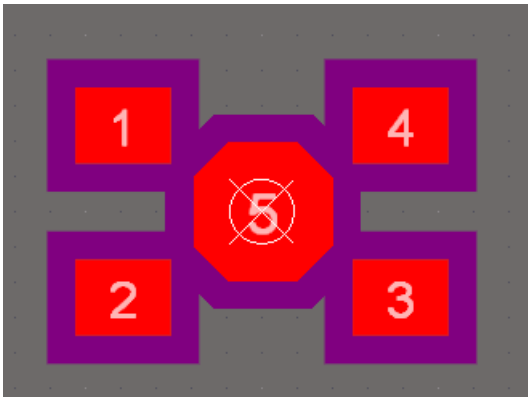
2.3.4. Réalisation PCB

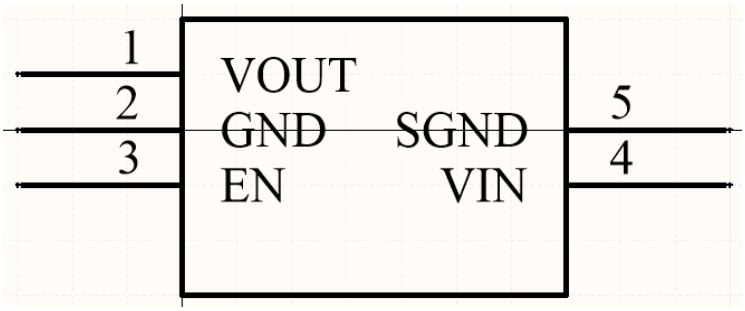
Pour commencer la réalisation du PCB, il faut d'abord créer de nouveau tous les schémas électroniques de la Numwork. Cela ne poserait pas de problème si tous les composants étaient facilement disponibles.

Un bon nombre de ces composants ne sont pas répertoriés par *Altium* (ou d'autre logiciel de CAO) ce qui a énormément ralenti la création du PCB.



Voici une partie du schematic de la Numwork qui et assez long :





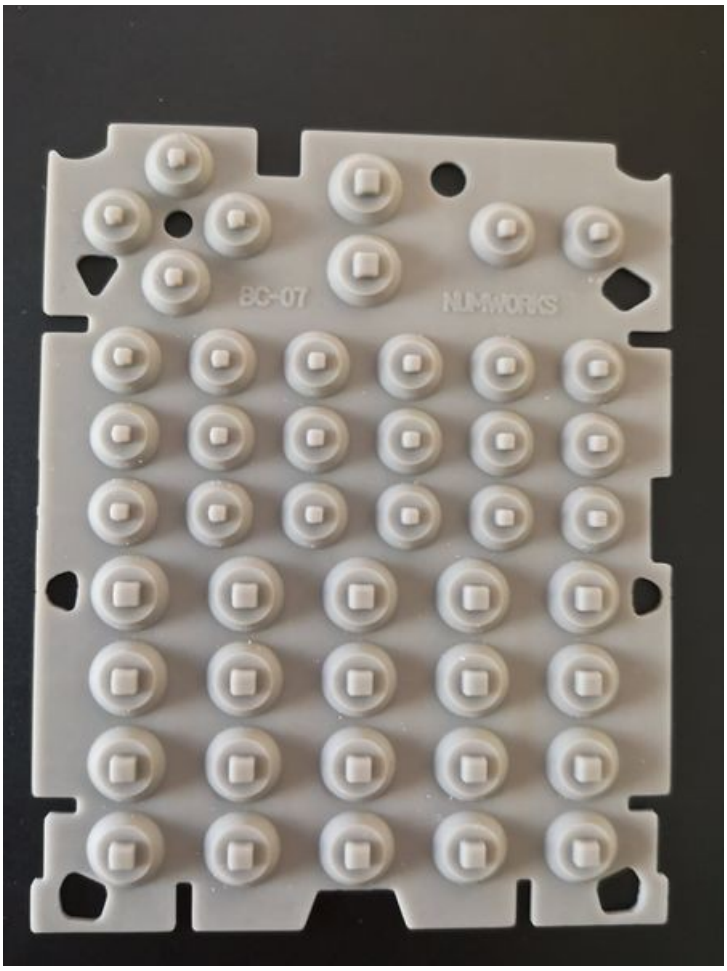
Exemple Composant recrée

Comme pour l’écran, on remarque que Numwork a choisi des composants « rares » qui ont même était difficiles à trouver pour certains.

Pour ce qui est du PCB, il n’est à ce jour toujours pas fini : un vrai problème se pose sur le routage de mon côté.

2.3.5. Clavier

Numwork utilise un clavier réalisé sur mesure pour leur calculatrice. Celui-ci est en silicone et carbone :



Clavier

Réaliser un tel clavier pour un seul modèle ne serait que trop onéreux (prix indique de base sur internet), c'est pourquoi une simple matrice de boutons poussoirs ferra l’affaire.

3. Difficultés rencontrées / Remarques

3.1. Partie logicielle

La seule vraie difficulté rencontrée dans la partie logicielle est au niveau des packages utilisés : il faut toujours faire attention à la version et bien souvent les compiler soi même et légèrement les modifier. Cela complique l'utilisation pour un novice car il faut souvent s'amuser à bidouiller soi même avec les différentes versions pour avoir une chance que ça marche.

Sinon, pour le code en lui même, la Numwork et claire et bien « rangée » ce qui facilite les recherches dans les nombreux fichiers ; par exemple, si on veux quelque chose relatif à une application, il serra dans *app*, si cela concerne la calculatrice en elle-même, c'est dans *ion*, si c'est le noyaux logiciel, dans *escher.*

3.2. Partie Hardware

La difficulté majeure pour réaliser le PCB est due au changement d'écran : le routage devient bien plus complexe, c'est un problème qui n'était pas prévu à la base. Je ne pensais pas qu'avoir à changer un seul composant rendrait le routage si chaotique, mais qu`aussi l'absence de nombreux composants seraient exclus des librairies classiques des fabricants (même la puce de flash produit par adesto n'est pas dans la librairie adesto).

3.3. Générale

En générale on remarque que les problèmes viennent de choix exotiques faits par Numwork ; il prône l'open-source que si on le veut, on peut faire soi-même sa calculatrice. Mais ils ont fait le choix de composants « rares », voir même sur-mesures alors que niveau logiciel ils n'ont pas fait de choix aussi complexes. On dirait que l'optique de l'entreprise est plus « d'acheter notre calculatrice et après bidouiller la comme vous voulez, mais si c'est pour la fabriquer vous même, bonne chance ! » C'est une optique totalement normale, c'est tout de même une entreprise, ils se doivent de gagner de l'argent et le fait de rendre open-source fait qui on une armée de développeurs qui travaillent pour eux gratuitement.

Numwork répond rapidement à toutes les questions que l'on pose, même des questions techniques plus ou moins poussées. C'est assez sympa de savoir que dès qu'on a une question, on peut aller la leur poser.

Je pense finir le PCB en dehors du projet pour ne pas laisser un trou dans le projet même si cela sera trop tard pour le rendu actuel.

4. Suite

4.1. Travail restant

Finir le PCB.

Utilisation des fonctions UART de façon plus poussées : faire des légers tests sans vraiment m'y pencher vu qu'un autre projet était plus axé là-dessus.

Optimisation matérielle : il est possible de changer quelques composants comme le CPU et/ou la flash avec des composants compatibles pour obtenir une calculatrice plus efficace.

Essayer de faire une fonction low-cost : pas vraiment vu de moyen de baisser le coût de fabrication des calculatrices autrement que par le boîtier qui permet de faire des économies de quelques euros par calculatrice.

4.2. Améliorations possibles

Modification du petit installateur pour avoir plusieurs version dedans.

Ajout d'un stockage interne et fonction annexe :

- il y a déjà un port pour rajouter une micro SD ce qui permettrait de faire un vrai stockage d'information dans Numwork (dans les 3go vu que le cpu est 32bit) ;

- un gestionnaire de fichier serait dans ce cas le bienvenu ;

- pouvoir faire des sauvegarde de courbe / calcul.

Conclusion

Ce projet a mobilisé beaucoup de compétences diverses que j'ai vues dans toute ma formation même si certaines ont était très peu utilisées. C'est un projet complet et transversal qui mêle électronique et informatique et même un peu de mécanique.

Certains problèmes auraient pu être évités en commencent la partie physique du projet en premier. Cependant, j'ai préféré finir les parties informatiques que j'avais commencées car cela aurait peut être mené à une partie physique finie et une partie informatique inachevée.