

Machine de Turing

ALLEGRE-COMMINGES Clément et BROUARD Romain

Table des matières

1	Introduction	2
1.1	Un peu d'histoire...	2
1.2	Fonctionnement d'une MTU	2
2	Conception	5
2.1	Cahier des charges	5
2.1.1	Partie 1	5
2.1.2	Partie 2	5
2.1.3	Partie 3	5
2.2	Définition des besoins	6
2.2.1	Diagramme bête à corne	6
2.2.2	Matrice Moscow	6
2.3	Conception Matérielle	7
2.3.1	SFN1	7
2.3.2	SFnD	8
2.3.3	Choix des technologies	12
2.3.4	Spécifications des fonctions et de leurs signaux de communication	12
2.3.5	Choix des composants	23
2.3.6	Schéma représentatif et schéma structurels	24
2.3.7	Tests	25
2.3.8	Conception Logicielle	25
2.3.9	Diagramme de Séquences	25
2.3.10	Algorithme	26
3	Annexes	30
3.1	code	30
3.2	Sources	30

Introduction

1.1 Un peu d'histoire...

En 1928, Le mathématicien allemand David Hilbert énonce le "problème de la décision". Il se demande s'il est possible de trouver une méthode « effectivement calculable » pour décider si une proposition est démontrable. Pour résoudre ce problème, il faut caractériser ce qu'est un procédé effectivement calculable. C'est alors qu'Alan Turing, alors en thèse à Cambridge, conceptualise une machine universelle et prouve grâce à cette dernière que le problème de l'arrêt est indécidable, ce qui permet de donner une réponse négative au problème d'Hilbert pour l'arithmétique. C'est alors qu'Alan Turing introduit les concepts de programme et programmation.

Ce concept de machine universelle, que nous appellerons désormais Machine de Turing Universelle (MTU) n'est pas réalisable puisqu'il s'agit d'un objet mathématique, dont on va détailler le fonctionnement plus tard. Néanmoins, une Machine de Turing à état fini peut-être construite, et la première vit donc le jour à Bletchley Park pendant la Seconde Guerre Mondiale, où Turing lui-même et une équipe de scientifique triés sur le volet par le MI6¹ construisirent une Machine de Turing pour casser les codes allemands générés par la machine Enigma.

1.2 Fonctionnement d'une MTU

Une MTU se compose de quatre éléments essentiels :

- un ruban de taille infinie, divisé en cases.
- une tête de lecture/écriture (qu'on appellera simplement tête de lecture, même si elle permet également d'écrire sur le ruban)
- un état interne
- une table de transition.

Une machine de Turing traite des symboles. L'ensemble des symboles est appelé Alphabet.

Le ruban permet d'accueillir des symboles qui seront lus et écrits par la tête de lecture.

La tête de lecture permet de lire et écrire un symbole. Elle peut se déplacer vers la gauche ou la droite.

Une machine de Turing fonctionne donc de la manière suivante :

1. La tête de lecture lit le symbole qu'elle pointe sur le ruban.
2. En fonction de son état et du symbole lu, la tête de lecture écrit un symbole à la place de celui qu'elle a lu précédemment.

1. Services de renseignement britanniques

3. Un déplacement est choisi en fonction de l'état de la tête de lecture et du symbole lu.
4. La tête de lecture change d'état.
5. Le ruban se déplace vers la droite ou vers la gauche selon le déplacement choisi précédemment.
6. Puis on recommence depuis le 1.

On peut donc résumer le fonctionnement comme cela : à chaque "cycle", on choisit un symbole à écrire, un nouvel état pour la tête de lecture, et un déplacement, en fonction d'un symbole lu et de l'état actuel. On peut donc définir une fonction de transition, qui va se charger de déterminer l'état futur d'une MTU en fonction de son état courant. L'ensemble des fonctions de transition permettant de traiter un "mot" peut être représenté sous la forme d'une table de transitions ou d'un graphe de transitions.

Un mot traité est dit accepté si une Machine de Turing s'arrête en état final après l'avoir intégralement traité.

Une MTU possède forcément un nombre fini d'états, et elle a au moins deux états obligatoires : q_0 l'état initial, et F un ensemble d'états d'acceptation.

Une machine de Turing est donc définie par :

- Q un ensemble fini d'états.
- q_0 un état initial tel que $q_0 \in Q$.
- F un ensemble d'états d'acceptation tel que $F \subseteq Q$.
- Γ un ensemble fini de symboles.
- Σ un ensemble fini de symboles d'entrée tel que $\Sigma \subseteq \Gamma$.
- B un symbole de ruban vide tel que $B \in \Gamma \setminus \Sigma$.
- δ une fonction de transition.

Une fonction de transition se formalise donc comme ceci :

$$\delta(q, Z) \rightarrow (p, Y, D)$$

avec q l'état de la tête de lecture, Z le symbole pointé, p le nouvel état, Y le nouveau symbole et D le déplacement.

Exemple d'une table de transition pour une Machine de Turing acceptant le langage $L = \{a^k b^k \mid k > 0\}$ avec q_0 comme état initial (représenté par une \rightarrow), et q_4 comme état d'acceptation (représenté par $*$) :

	symboles				
	a	b	X	Y	Blank
→ q0	(q1, X, R)			(q3, Y, R)	
q1	(q1, a, R)	(q2, Y, L)		(q1, Y, R)	
q2	(q2, a, L)		(q0, X, R)	(q2, Y, L)	
q3				(q3, Y, R)	(q4, B, R)
* q4					

Le but de ce projet va donc être de concevoir une Machine de Turing telle que définie ci-dessus.

Conception

2.1 Cahier des charges

Le cahier des charges est composé de trois parties. La partie 1 doit être traitée absolument, les parties 2 et 3 seront traitées si le temps nous le permet.

2.1.1 Partie 1

Dans un premier temps, le système conçu doit être capable de :

- exécuter un programme prédéfini (codé en dur dans le programme microcontrôleur).
- avoir un mode pas à pas pour l'exécution du programme.
- avoir un mode continu pour l'exécution du programme.
- gérer l'affichage de l'état du ruban.
- gérer l'affichage de la position de la tête de lecture.
- gérer l'affichage de la table de transition.

2.1.2 Partie 2

Dans un second temps, il faut rajouter :

- la possibilité de sélectionner un programme via un menu.
- le stockage des programmes à sélectionner.
- l'initialisation manuelle du ruban et de la position de la tête de lecture.

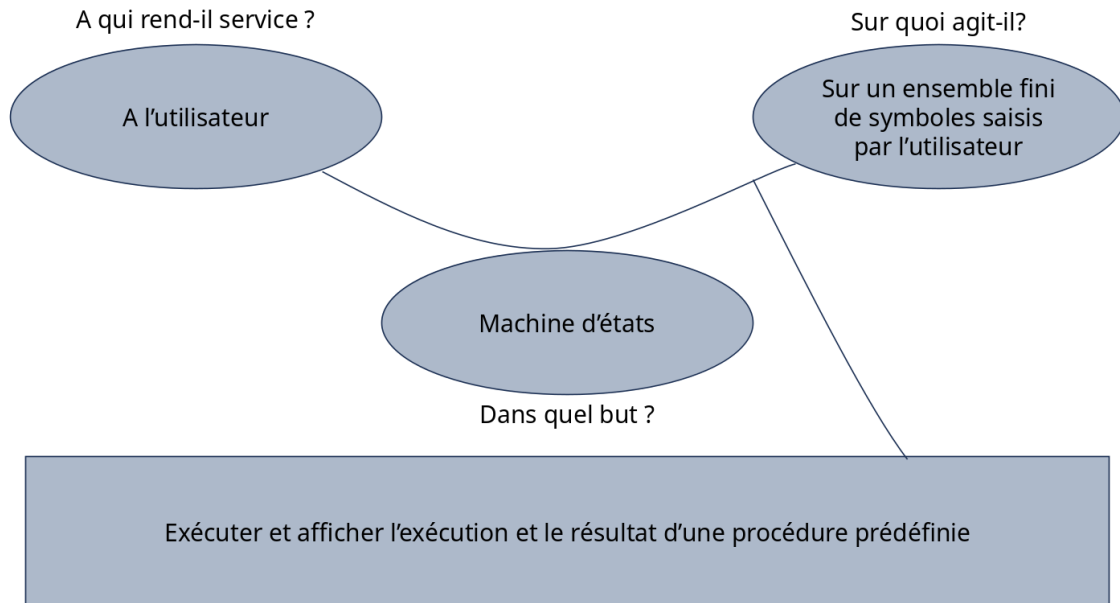
2.1.3 Partie 3

Enfin pour obtenir un système complet, il faut implémenter :

- la programmation directement sur la machine d'une table de transition.
- l'enregistrement de la table de transition programmée dans le support de stockage.
- un reset de la programmation de la ligne en cours.
- l'affichage d'une description du programme.

2.2 Définition des besoins

2.2.1 Diagramme bête à corne



2.2.2 Matrice Moscow

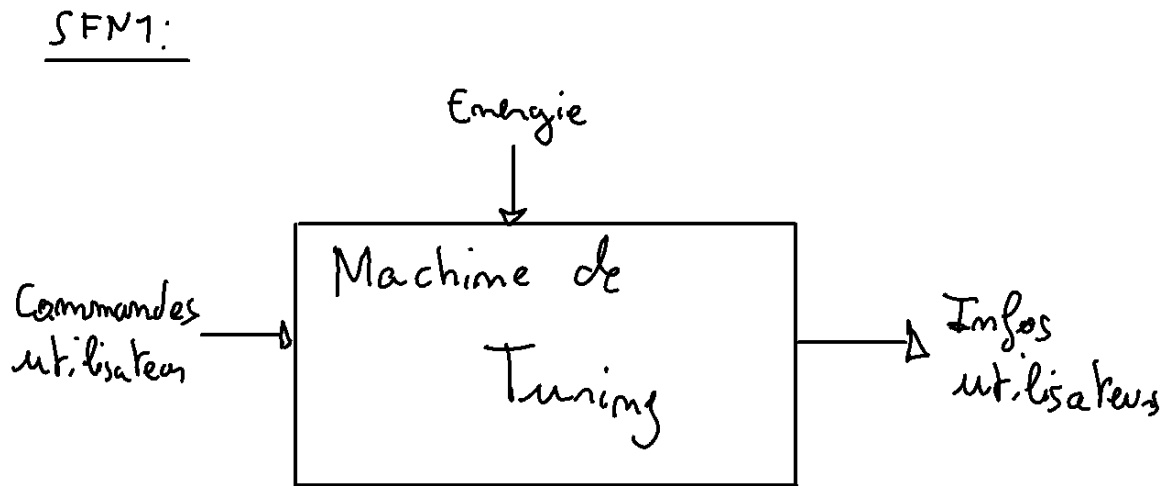
<u>Must Have</u>	<u>Should Have</u>
<ul style="list-style-type: none">- Machine de Turing capable au moins de faire l'addition de 2 nombres- Mode continu/pas à pas pour l'exécution du programme- Affichage de l'état du ruban- Affichage de la position de la tête de lecture- Gérer l'affichage de la table de transition	<ul style="list-style-type: none">- La possibilité de sélectionner un programme via un menu- Stockage des programmes à sélectionner- Initialisation manuelle du ruban et de la position de la tête de lecture.
<u>Could Have</u>	<u>Won't Have</u>
<ul style="list-style-type: none">- Programmation directement sur la machine d'une table de transition- Enregistrement de la table de transition programmée dans le support de stockage- Reset de la programmation de la ligne en cours- Affichage d'une description du programme	

2.3 Conception Matérielle

Pour concevoir notre système, nous avons décidé de traiter les trois parties en même temps, ce qui nous évite de devoir repasser par une phase de conception et d'adaptation lors de la réalisation des parties 2 et 3.

2.3.1 SFN1

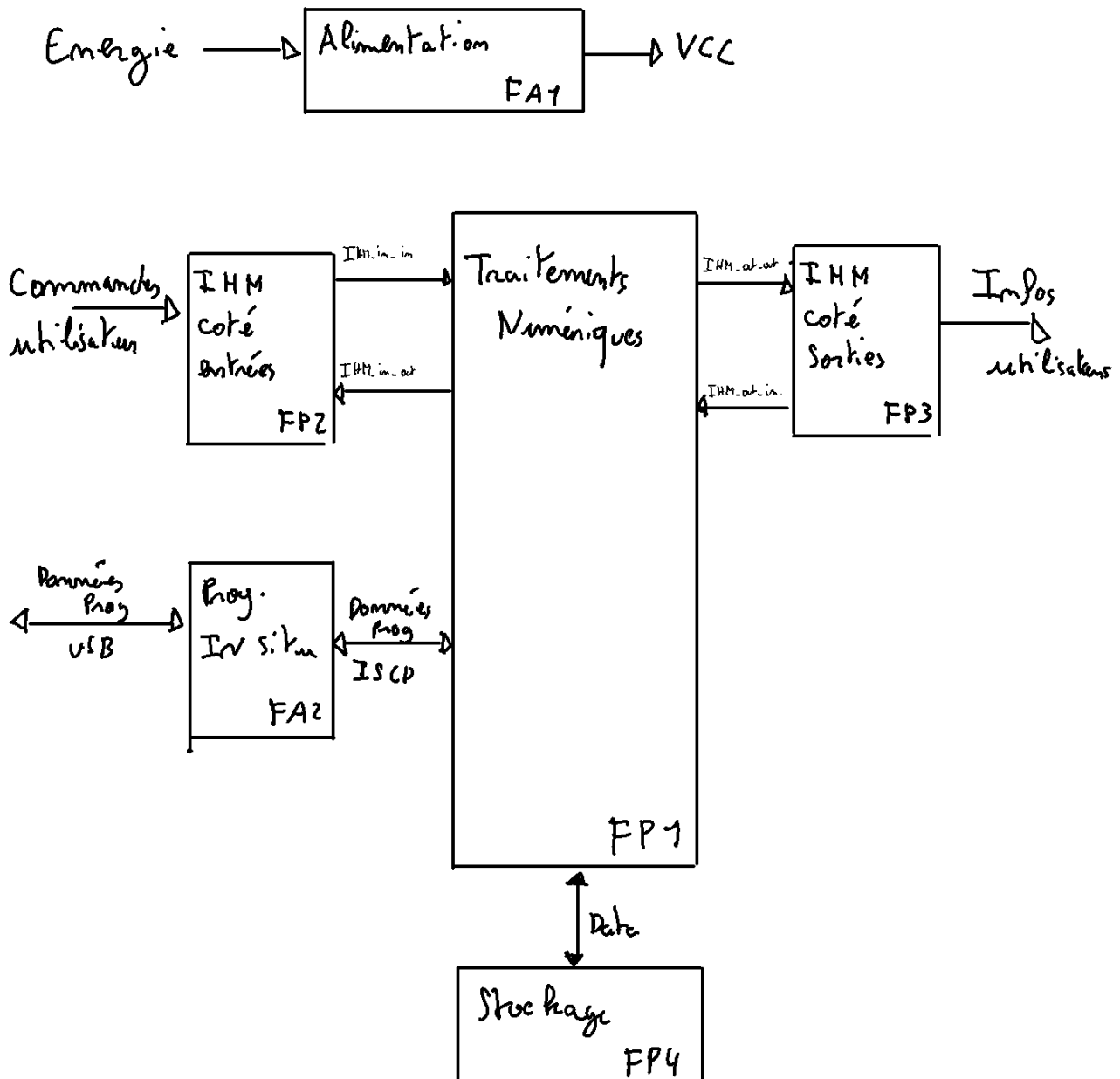
Nous avons donc commencé par dessiner un Schéma fonctionnel de premier niveau, pour définir les différentes entrées et sorties de notre système :



2.3.2 SFnD

On peut désormais rentrer un peu plus dans le détail avec un SF1D :

SF1D:

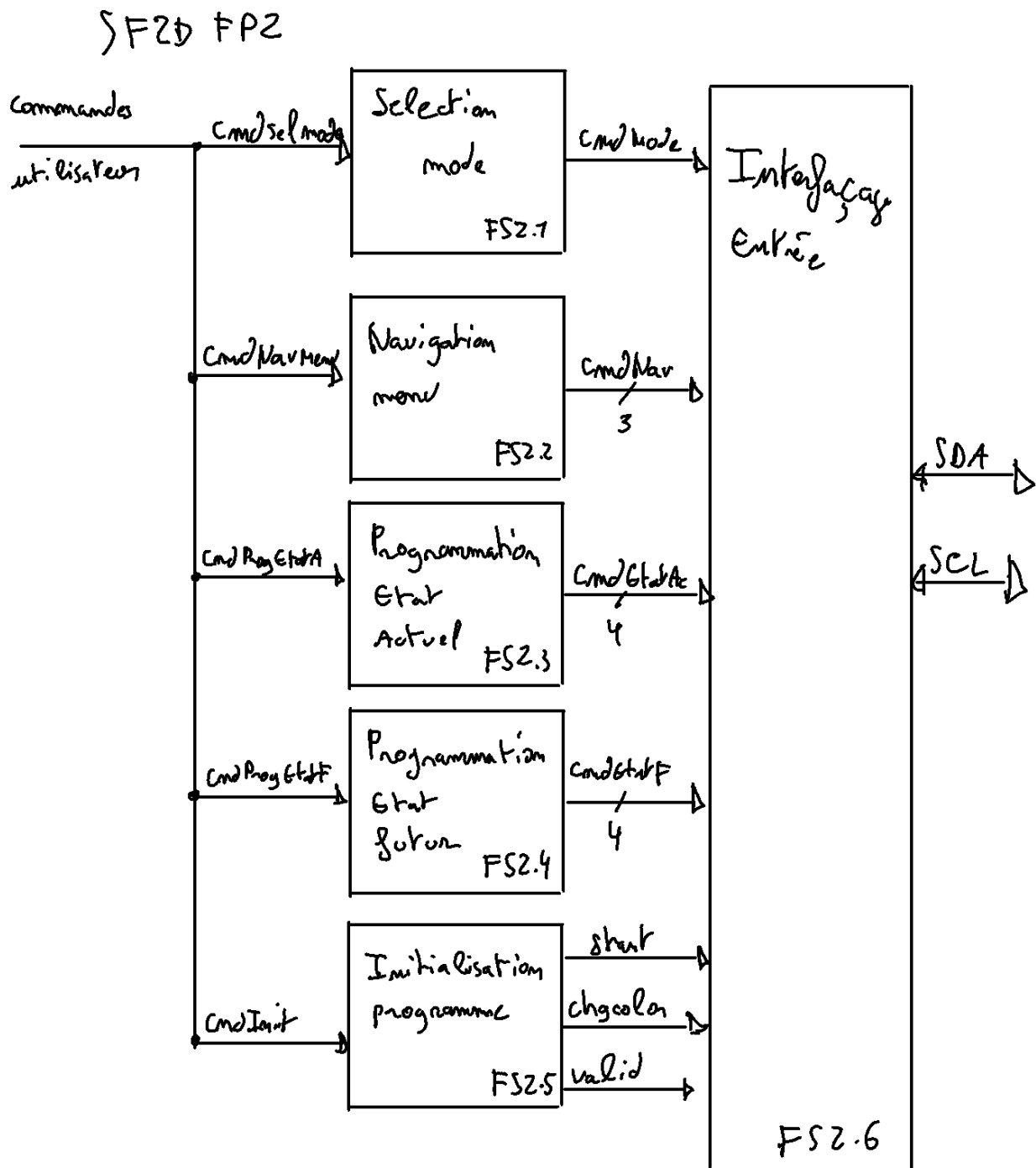


On voit qu'on a quatre fonctions principales : une qui s'occupe de la gestion des entrées (FP2), une qui gère les sorties (FP3), une fonction de stockage pour stocker les tables de transitions (FP4), et du traitement numérique qui va piloter tout ça (FP1).

On a aussi deux fonctions annexes : la fonction alimentation qui va se charger de fournir le courant nécessaire pour que la machine puisse fonctionner, et la fonction de programmation in-situ qui va faciliter le chargement du code dans la machine et nous éviter d'avoir à sortir le micro-contrôleur (MCU) à chaque fois.

On voit que les fonctions FP2 et FP3 sont encore floues, on va donc les affiner en faisant des SF2D pour qu'on se rende compte de leur fonctionnement :

FP2 :

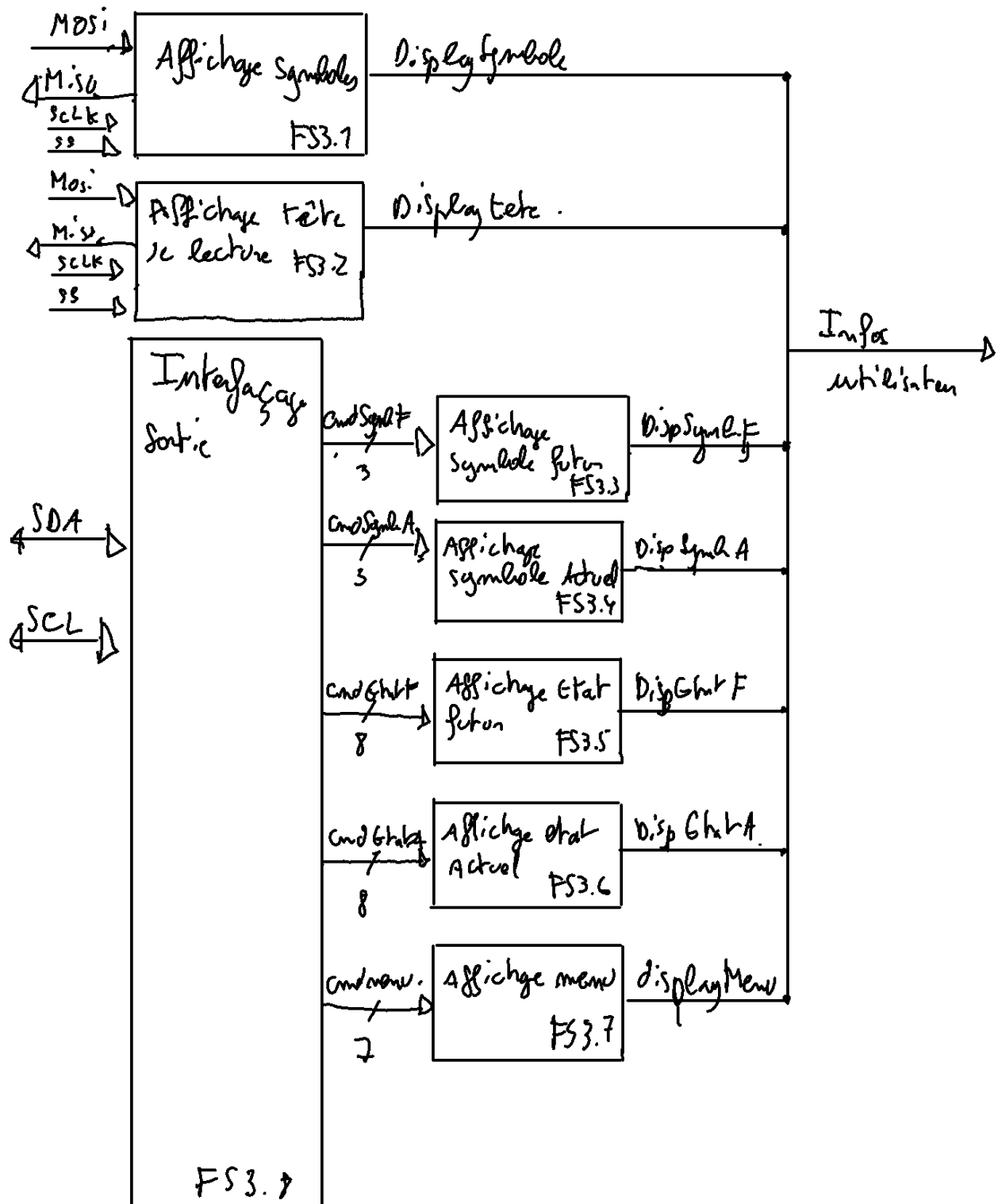


On voit clairement que FP2 est bien plus complexe qu'elle ne le paraît, et forme un ensemble

de fonctions plus spécifiques qui vont permettre à l'utilisateur d'interagir physiquement avec la machine, dans le but de la configurer, programmer, etc.

FP3 :

SF20 TP3



De la même manière que pour FP2, FP3 est en réalité un ensemble de fonctions plus spécifiques qui vont chacune avoir un rôle bien précis dans le retour des informations utilisateur. On a donc une grande variété de fonctions d'affichage qui vont, de la même manière que pour les sous-

fonctions de FP2, être spécifiées ci-dessous.

2.3.3 Choix des technologies

Communication

Bus I²C Le nombre élevé d'entrées-sorties, du aux grand nombre de boutons poussoirs et composants d'affichage, nous a orienté sur l'utilisation d'expandeurs. Nous nous sommes orientés vers des expandeurs I²C, car l'utilisation du bus I²C nous permettait de gérer toutes nos entrées / sorties en utilisant uniquement deux broches sur le MCU (SDL et SCL). Nous avons ainsi essayé d'avoir un maximum de composants en I²C.

SPI Nous nous sommes posés la question de l'utilisation du bus SPI pour notre ruban de LEDs. Nous avons aussi besoin de choisir notre type de communication avec la carte SD, et il se trouve que le bus SPI est très utilisé pour ce type de communication.

2.3.4 Spécifications des fonctions et de leurs signaux de communication

Nous avons donc spécifié ci-dessous, pour chaque fonction, son rôle ainsi que ses signaux d'entrée et/ou de sortie. Nous avons également précisé le rôle ainsi que ce qui fait l'essence même de chaque signal (type, etc.).

Tableaux de description des fonctions et des signaux

Description d'une fonction

Fonction	FP1	Traitements numériques
Description / rôle	Réalise tous les traitements numériques nécessaires au bon fonctionnement du système.	
Signaux d'entrée	➤ MISO	
Signaux de sortie	➤ MOSI, SCLK, SS, SCL	
Signaux E/S	➤ SDA, ISCP	

Fonction	FS2.1	Sélection du mode
Description / rôle	Fonction qui sert à sélectionner le mode d'exécution, entre pas à pas ou continu.	
Signaux d'entrée	➤ CmdSelMode	
Signaux de sortie	➤ CmdMode	
Signaux E/S	➤ NA	

Fonction	FS2.2	Navigation Menu
Description / rôle	Fonction qui sert à naviguer dans le menu.	
Signaux d'entrée	➤ CmdNavMenu	
Signaux de sortie	➤ CmdNav<0:2>	
Signaux E/S	➤ NA	

Fonction	FS2.3	Programmation Etat Actuel
Description / rôle	Fonction qui permet de programmer l'état actuel de la table de transition.	

Signaux d'entrée	➤ CmdProgEtatA
Signaux de sortie	➤ CmdEtatA<0:3>
Signaux E/S	➤ NA

Fonction	FS2.4	Programmation Etat Futur
Description / rôle	Fonction qui permet de programmer l'état actuel de la table de transition.	
Signaux d'entrée	➤ CmdProgEtatF	
Signaux de sortie	➤ CmdEtatF<0:3>	
Signaux E/S	➤ NA	

Fonction	FS2.5	Initialisation Programme
Description / rôle	Fonction qui permet d'initialiser les symboles et la position de la tête de lecture.	
Signaux d'entrée	➤ CmdInit	
Signaux de sortie	➤ Start, ChgColor, Valid	
Signaux E/S	➤ NA	

Fonction	FS2.6	Interfaçage Entrée
Description / rôle		
Signaux d'entrée	➤ CmdMode, CmdNav<0 :2>, CmdEtatA<0:3>, CmdEtatF<0:3>, Start, ChgColor, Valid	
Signaux de sortie	➤ SCL	
Signaux E/S	➤ SDA	

Fonction	FS3.1	Affichage Symboles
Description / rôle	Réalise l'affichage des symboles (représente le ruban).	
Signaux d'entrée	➤ MOSI, SS, SCLK	

Signaux de sortie	➤ MISO, DisplaySymboles
Signaux E/S	➤ NA

Fonction	FS3.2	Affichage tête de lecture
Description / rôle	Réalise l’affichage de la tête de lecture.	
Signaux d'entrée	➤ MOSI, SS, CLK	
Signaux de sortie	➤ MISO, DisplayTete	
Signaux E/S	➤ NA	

Fonction	FS3.3	Affichage Symbole Futur
Description / rôle	Affiche le symbole futur (après transition).	
Signaux d'entrée	➤ CmdSymboleF<0:2>	
Signaux de sortie	➤ DisplaySymboleF	
Signaux E/S	➤ NA	

Fonction	FS3.4	Affichage Symbole Actuel
Description / rôle	Affiche le symbole courant (avant transition).	
Signaux d'entrée	➤ CmdSymboleA<0:2>	
Signaux de sortie	➤ DisplaySylboleA	
Signaux E/S	➤ NA	

Fonction	FS3.5	Affichage Etat Futur
Description / rôle	Affichage l’état futur de la tête de lecture (après transition).	
Signaux d'entrée	➤ CmdEtatF<0:7>	
Signaux de sortie	➤ DisplayEtatF	

Signaux E/S	➤ NA
-------------	------

Fonction	FS3.6	Affichage Etat Actuel
Description / rôle	Affiche l'état courant de la tête de lecture (avant transition).	
Signaux d'entrée	➤ CmdEtatA<0:7>	
Signaux de sortie	➤ DisplayEtatA	
Signaux E/S	➤ NA	

Fonction	FS3.7	Affichage Menu
Description / rôle	Affiche le menu.	
Signaux d'entrée	➤ CmdMenu<0:6>	
Signaux de sortie	➤ DisplayMenu	
Signaux E/S	➤ NA	

Fonction	FS3.8	Interfaçage sortie
Description / rôle		
Signaux d'entrée	➤ CmdSymboleF<0:2>, CmdSymboleA<0:2>, CmdEtatF<0:7>, CmdEtatA<0:7>, CmdMenu<0:6>	
Signaux de sortie	➤ SCL	
Signaux E/S	➤ SDA	

Fonction	FP4	Stockage
Description / rôle	Permet le stockage des tables de transitions.	
Signaux d'entrée	➤ MOSI, SCLK, SS	
Signaux de sortie	➤ MISO	
Signaux E/S	➤ NA	

Fonction	FA1	Alimentation
Description / rôle	Alimente le système en électricité.	
Signaux d'entrée	➤ Energie	
Signaux de sortie	➤ VCC	
Signaux E/S	➤ NA	

Fonction	FA2	Programmation In Situ
Description / rôle	Permet de charger le programme dans le MCU.	
Signaux d'entrée	➤ NA	
Signaux de sortie	➤ NA	
Signaux E/S	➤ Données USB, ISCP	

Description des signaux

Signal	Fonctions concernées	Nature du signal (A/N/GP)	Taille entité	Grandeur et unité (U, I,...)	Plage de variation - Niveaux	Excursion en fréquence	Valeur au repos	Contraintes temporelles	Conformité à une norme
	Description								
MISO	FP1, FS3.1, FS3.2, FA2	N	1	U (V)	Niveau haut – niveau bas	NA	NA	NA	SPI
	Signal généré par l’esclave communiquer avec le maître.								
MOSI	FP1, FS3.1, FS3.2, FA2	N	1	U (V)	Niveau haut – niveau bas	NA	NA	NA	SPI
	Signal généré par le maître pour communiquer avec l’esclave.								
SCLK	FP1, FS3.1, FS3.2, FA2	N	1	U (V)	Niveau haut – niveau bas	NA	NA	NA	SPI
	Horloge transmise par le maître.								
SS	FP1, FS3.1, FS3.2, FA2	N	1	U (V)	Niveau haut – niveau bas	NA	NA	NA	SPI
	Sélection de l’esclave par le maître.								
SDA	FP1, FS2.6, FS3.8	N	1	U (V)	Niveau haut – niveau bas	NA	1	NA	I2C

Signal	Fonctions concernées	Nature du signal (A/N/GP)	Taille entité	Grandeur et unité (U, I,...)	Plage de variation - Niveaux	Excursion en fréquence	Valeur au repos	Contraintes temporelles	Conformité à une norme
	Description								
	Signal de transmission des données								
SCL	FP1, FS2.6, FS3.8	N	1	U (V)	Niveau haut – niveau bas	NA	1	NA	I2C
	Signal de transmission de l’horloge								
CmdSelMode	FS2.1	GP	NA	Newton (N)	NA	NA	NA	NA	NA
	Bascule du commutateur de sélection du mode.								
CmdMode	FS2.1	N	1	U (V)	Niveau haut – niveau bas	NA	NA	NA	NA
	Image du mode (Niveau Bas : mode pas à pas, Niveau haut : mode continu)								
CmdNavMenu	FS2.2	GP	NA	N	NA	NA	NA	NA	NA
	Appui sur les boutons de navigation dans le menu.								
	FS2.2	N	3	U (V)	Niveau haut – niveau bas	NA	NA	NA	NA
CmdNav	Image de l’appui sur les boutons (3 signaux car 3 boutons).								
	FS2.3	GP	NA	N	NA	NA	NA	NA	NA
CmdProgEtatA	Appui sur les boutons de programmation de l’état courant.								
	FS2.3	N	4	U (V)	Niveau haut – niveau bas	NA	NA	NA	NA
CmdEtatA	FS2.3	N	4	U (V)	Niveau haut – niveau bas	NA	NA	NA	NA

Signal	Fonctions concernées	Nature du signal (A/N/GP)	Taille entité	Grandeur et unité (U, I,...)	Plage de variation - Niveaux	Excursion en fréquence	Valeur au repos	Contraintes temporelles	Conformité à une norme
	Description								
	Image de l'appui sur les boutons (4 signaux car 4 boutons).								
CmdProgEtatF	FS2.4	GP	NA	N	NA	NA	NA	NA	NA
	Appui sur les boutons de programmation de l'état futur.								
CmdEtatF	FS2.4	N	4	U (V)	Niveau haut – niveau bas	NA	NA	NA	NA
	Image de l'appui sur les boutons (3 signaux car 3 boutons).								
CmdInit	FS2.5	GP	NA	N	NA	NA	NA	NA	NA
	Appui sur les boutons d'initialisation et lancement du programme.								
Start	FS2.5	N	1	U (V)	Niveau haut – niveau bas	NA	NA	NA	NA
	Image de l'appui sur le bouton de lancement.								
ChgColor	FS2.5	N	1	U (V)	Niveau haut – niveau bas	NA	NA	NA	NA
	Image de l'appui sur le bouton d'initialisation des symboles et de la tête de lecture.								
Valid	FS2.5	N	1	U (V)	Niveau haut – niveau bas	NA	NA	NA	NA
	Image de l'appui sur le bouton de validation.								
Energie	FA1	A	1	U (V)	230V CC	NA	NA	NA	NA

Signal	Fonctions concernées	Nature du signal (A/N/GP)	Taille entité	Grandeur et unité (U, I,...)	Plage de variation - Niveaux	Excursion en fréquence	Valeur au repos	Contraintes temporelles	Conformité à une norme
Description									
VCC									
	FA1	A	1	U (V)	5V CC 3.3V CC	NA	NA	NA	NA
ISCP									
	FP1, FA2	N	1	NA	Niveau haut – niveau bas	NA	0	NA	NA
DisplaySymboles	Signal de transmission des données pour la programmation in situ								
	FS3.1	GP	NA	Lux	NA	NA	NA	NA	NA
DisplayTete	Affichage de la fonction FS3.1								
	FS3.2	GP	NA	Lux	NA	NA	NA	NA	NA
CmdSymboleF	Affichage de la fonction FS3.2								
	FS3.3, FS3.8	N	3	U (V)	Niveau haut – niveau bas	NA	NA	NA	NA
DisplaySymboleF	Signaux de sortie de la fonction FS3.8, qui commandent les entrées de la fonction FS3.3								
	FS3.3	GP	NA	Lux	NA	NA	NA	NA	NA
CmdSymboleA	Affichage de la fonction FS3.3								
	FS3.4, FS3.8	N	3	U (V)	Niveau haut – niveau bas	NA	NA	NA	NA

[illegible]

2.3.5 Choix des composants

Composants

Nous avons donc besoin des composants suivants :

- 2 rubans de LEDs RGB communicant en SPI
- 2 afficheurs 7 segments
- 1 écran LCD communicant en I²C
- 2 LEDs RGB
- 2 LEDs pour indiquer le mouvement, et 1 LED d'erreur
- 15 boutons poussoirs
- 2 commutateurs à bascule
- 1 lecteur de carte micro-SD communicant en SPI
- 3 expandeurs E/S - I²C de 16 I/O

Choix du micro contrôleur

Nous avons déjà un PIC24FJ64GA002 car c'est le micro-contrôleur que le groupe précédent utilisait. Ce micro-contrôleur s'alimente en 3,3V et possède deux modules I²C et deux modules SPI. Comme nous utilisons des expandeurs pour toutes nos E/S, nous pouvons reprendre ce micro-contrôleur pour notre machine de Turing.

Commandes

Initialement, nous avons fait un premier choix de composants qui correspondait à nos spécificités. Toute la commande était prévue chez Farnell, mais nous avons dû nous raviser et mettre au point une autre liste à cause du délai de commande trop important.

Commande initiale Farnell

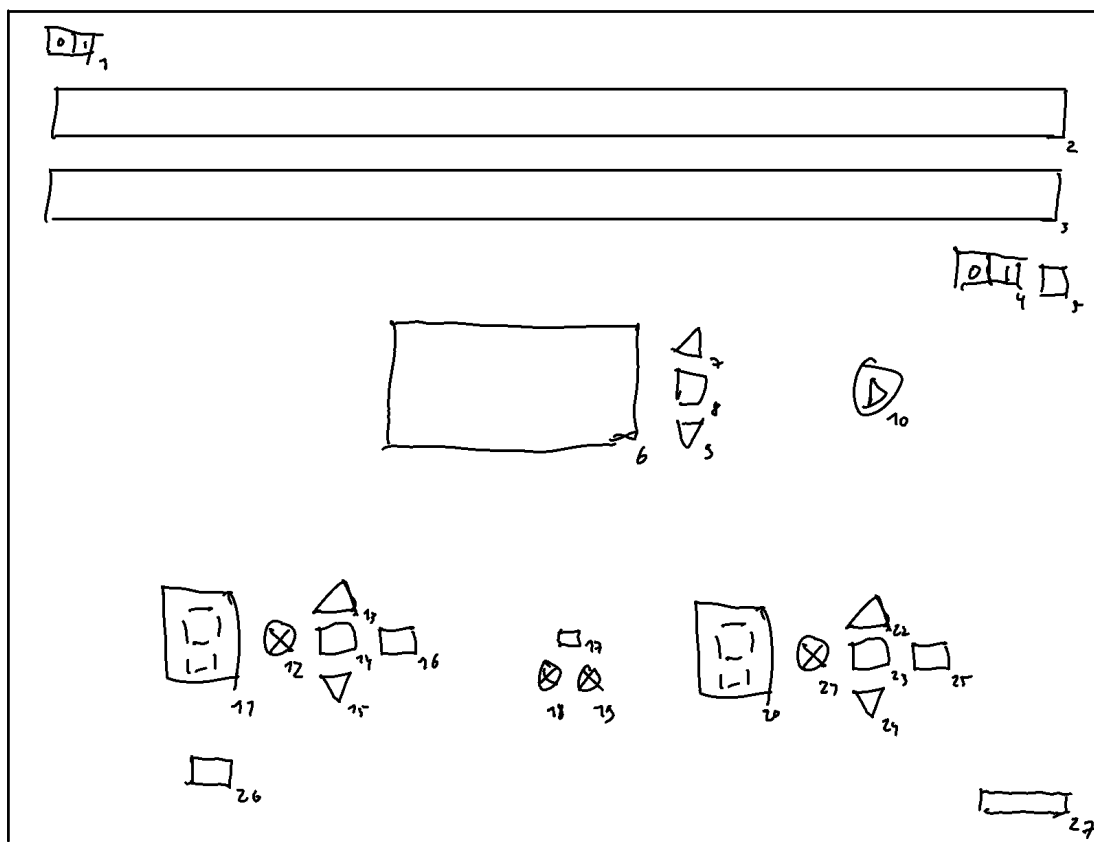
- rubans de LEDs adressables [SK9822](#)
- afficheurs 7 segments [HDSM-283B](#)
- LEDs RGB [L-59EYC](#)
- expandeurs E/S - I²C [MCP23017](#)
- écran LCD I2C [MC21605C6W-BNMLWI-V2](#)
- boutons poussoirs [ESE20C321](#)
- commutateurs [2AS2T2A1M7RE](#)

Commande réelle (RS et Amazon)

- rubans de LEDs adressables [SK9822](#)
- expandeurs E/S - I²C [MCP23017](#)
- écran LCD I²C [NHD-C0220BiZ-FSW-FBW-3V3M](#)
- LEDs RGB [L-154A4SURKQBDZGW](#)

2.3.6 Schéma représentatif et schéma structurels

Le schéma ci-dessous nous donne une idée de ce à quoi pourrait ressembler notre machine de Turing une fois tous les composants montés :



Légende :

- 1 : Interrupteur d'alimentation
- 2 : Ruban de LEDs pour l'affichage des symboles
- 3 : Ruban de LEDs pour l'affichage de la tête de lecture
- 4 : Commutateur pour choisir entre le mode pas à pas et automatique
- 5 : Bouton de réinitialisation lors de l'initialisation manuelle des rubans de LEDs
- 6 : Ecran LCD pour l'affichage du menu
- 7, 8, 9 : Boutons de navigation dans le menu (monter, sélectionner, descendre)
- 10 : Bouton de lancement du programme. Sert également à passer à l'étape suivante en mode pas à pas
- 11 | 20 : Afficheurs 7 segments pour l'affichage de l'état courant | de l'état suivant
- 12 | 21 : LEDs RGB pour l'affichage du symbole courant | du symbole suivant
- 13, 14, 15, 16 | 22, 23, 24, 25 : Boutons de décrément, validation, incrément, ré-initialisation de l'état et du symbole courant | de l'état et du symbole futur
- 17 : Bouton de changement du déplacement pour l'état suivant
- 18, 19 : LED représentant le déplacement pour l'état suivant (gauche si LED 18 allumée, droite si LED 19 allumée)

- 26 : Bouton de sélection d'un état d'acceptation
- 27 : Lecteur de carte micro-SD

2.3.7 Tests

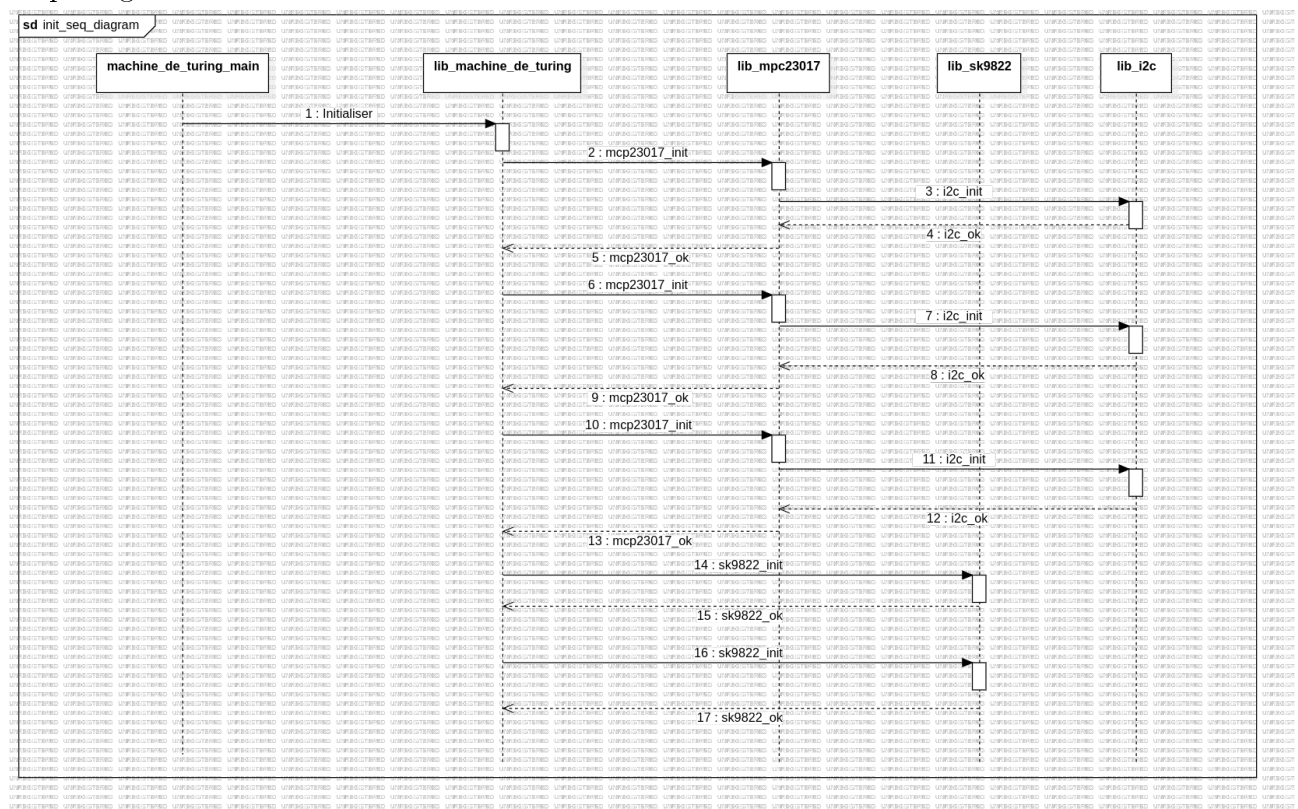
2.3.8 Conception Logicielle

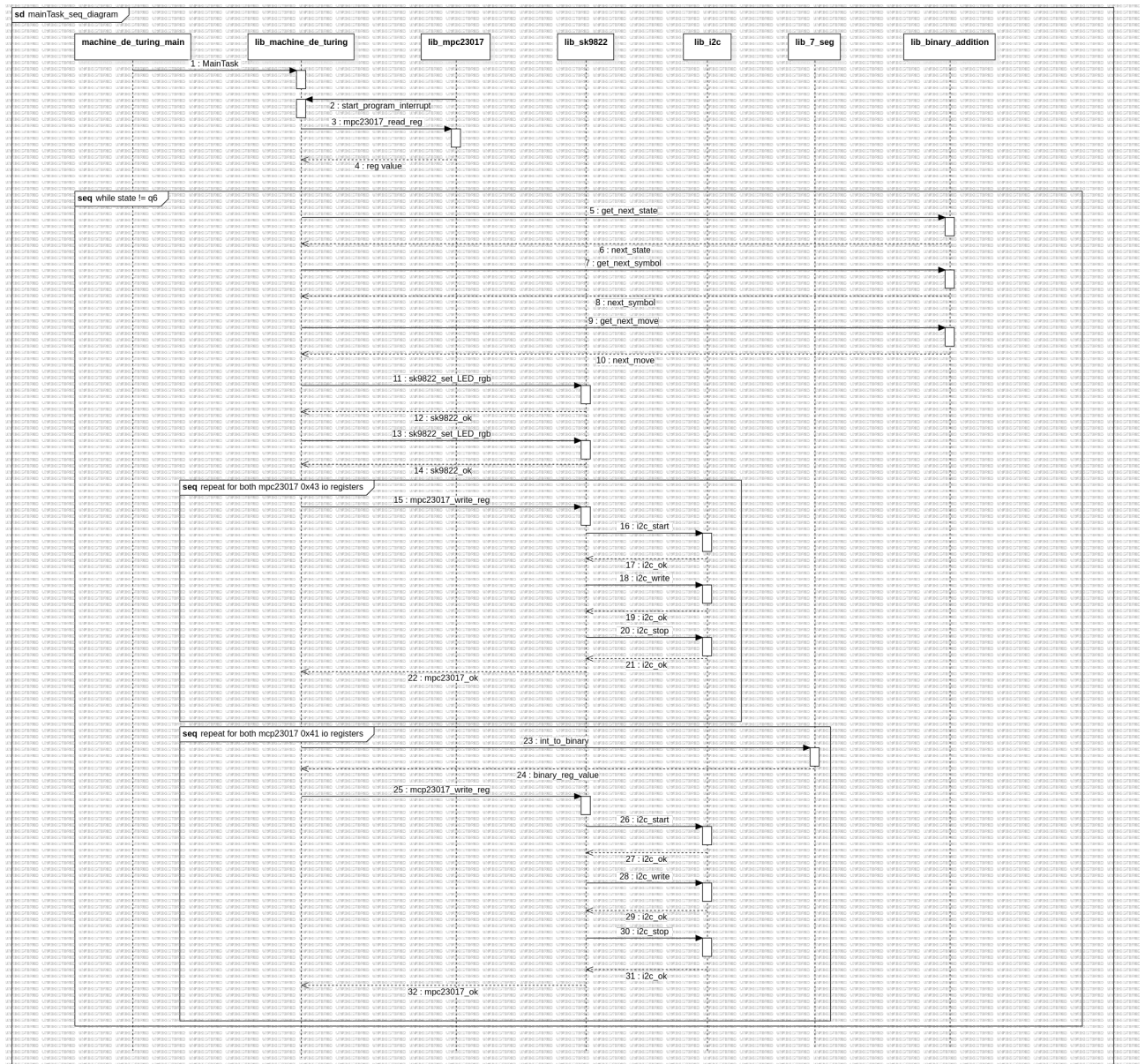
Cette section ne traitera que la partie 1 du cahier des charges, les parties 2 et 3 ne pouvant être réalisées par manque de temps.

2.3.9 Diagramme de Séquences

Les deux diagrammes de séquence ci-dessous montrent respectivement la phase d'initialisation et la phase de déroulement du programme au niveau du micro-contrôleur (on ne prend pas en compte les actions de l'utilisateur).

Ce sont des diagrammes "best-case", c'est à dire qu'on ne prend pas en compte les erreurs qui pourraient survenir, et on déroule le fonctionnement du programme en supposant que tout se passe bien. Cela permet principalement de ne pas faire trop de diagrammes, ni de diagrammes trop longs.





2.3.10 Algorithmme

```

1  LIB_MACHINE_DE_TURING
2  VARIABLES
3      mcp23017_desc_t mpc23017_7_seg
4      mcp23017_desc_t mcp23017_bp
5      mcp23017_desc_t mcp23017_general
6      i2c_desc_t I2CModule
7
8  ENUMERATIONS
9      mcp23017_err_t
10     MCP23017_OK
11     MCP23017_ERROR
12
13     mcp23017_i2c_init_type_t
14     INIT_WITH_I2C1
15     INIT_WITH_I2C2
16     INIT_ALREADY_DONE
17

```

```

18  working_mode_t
19      MANUAL_MODE
20      AUTOMATIC_MODE
21
22  STRUCTURES
23  mcp23017_desc_t
24      i2c_desc_t  *pi2c
25      uint8_t      i2c_Address
26
27  mcp23017_config_t
28      i2c_desc_t  *pi2c;
29      mcp23017_i2c_init_type_t    initType;
30      uint8_t i2c_Address;
31
32
33  machine_de_turing_desc_t
34      uint8_t state
35      uint8_t symbol
36      uint8_t next_state
37      uint8_t next_symbol
38      uint8_t next_move
39
40  CONSTANTES
41      MCP23017_7_SEG_ADDRESS      0100001
42      MCP23017_BP_ADDRESS        0100010
43      MCP23017_GENERAL_ADDRESS    0100011
44
45      MCP23017_I0CONA_ADDRESS     0x0A
46      MCP23017_I0A_ADDRESS       0x12
47      MCP23017_I0B_ADDRESS       0x13
48
49  DEBUT
50      PROCEDURE Initialiser()
51      VARIABLES
52          mcp23017_err_t    Res;
53          mcp23017_config_t    mcpCfg;
54      DEBUT
55          Configuration du port B en sortie
56
57          // MPC23017 7 segs
58          mcpCfg.pi2c = &I2CModule;
59          Initialisation avec I2C_1
60          Adresse = MCP23017_7_SEG_ADDRESS;
61
62          Res = mcp23017_init(@mpc23017_7_seg, @mcpCfg)
63          SI Res != MCP23017_OK ALORS
64              error_handler()
65          FIN SI
66
67          // MPC23017 BP
68          mcpCfg.pi2c = &I2CModule;
69          Initialisation avec I2C_1
70          Adresse = MCP23017_BP_ADDRESS;
71
72          Res = mcp23017_init(@mpc23017_bp, @mcpCfg)
73          SI Res != MCP23017_OK ALORS
74              error_handler()
75          FIN SI
76
77          // MPC23017 GENERAL

```

```

78     mcpCfg.pi2c = &I2CModule;
79     Initialisation avec I2C_1
80     Adresse = MCP23017_GENERAL_ADDRESS;
81
82     Res = mcp23017_init(@mcp23017_bp, @mcpCfg)
83     SI Res != MCP23017_OK ALORS
84         error_handler()
85     FIN SI
86
87     FIN Initialiser()
88
89     PROCEDURE MainTask()
90     VARIABLES
91         uint8_t working_mode
92         uint8_t next_step
93         led_color_t rw_pointer_color
94     DEBUT
95         Res = mcp23017_read_reg(@mcp23017_bp, MCP23017_IOA_ADDRESS,
@working_mode)
96         SI Res != MCP23017_OK ALORS
97             working_mode = MANUAL_MODE
98         FIN SI
99
100        TANT QUE mtu->state != ACCEPT_STATE FAIRE
101            get_next_state(mtu->state, mtu->symbol, mtu->next_state)
102            get_next_symbol(mtu->state, mtu->symbol, mtu->next_symbol)
103            get_next_move(mtu->state, mtu->symbol, mtu->next_move)
104            set_next_move(mtu->position, mtu->next_move)
105
106            sk9822_set_LED_rgb(sk9822_1, mtu->position, mtu->symbol)
107            sk9822_set_LED_rgb(sk9822_1, mtu->position, rw_pointer_color)
108
109            RegValue = int_to_bin(mtu->state)
110            Res = mcp23017_write_reg(@mcp23017_7_seg, MCP23017_IOA_ADDRESS,
RegValue)
111            SI Res != MCP23017_OK ALORS
112                error_handler()
113            FIN SI
114
115            RegValue = int_to_bin(mtu->next_state)
116            Res = mcp23017_write_reg(@mcp23017_7_seg, MCP23017_IOB_ADDRESS,
RegValue)
117            SI Res != MCP23017_OK ALORS
118                error_handler()
119            FIN SI
120
121            RegValue = generate_general_reg(mtu->symbol, mtu->next_symbol, mtu
->next_move)
122            Res = mcp23017_write_reg(@mcp23017_general, MCP23017_IOA_ADDRESS,
RegValue)
123            SI Res != MCP23017_OK ALORS
124                error_handler()
125            FIN SI
126
127            mtu->state = mtu->next_state
128            mtu->symbol = mtu->next_symbol
129
130            SI working_mode = MANUAL_MODE ALORS
131                TANT QUE next_step = 0 FAIRE
132                    FIN TANT QUE

```

```
133         SINON
134             attendre(2000)
135         FIN SI
136     FIN TANT QUE
137 FIN MainTask()
138
139 PROCEDURE error_handler()
140     allumer_led()
141     TANT QUE 1 FAIRE
142 FIN TANT QUE
143 FIN error_handler
144
```

Annexes

3.1 code

3.2 Sources