

Rapport TP3 MCSED

BÉNISTANT Raphaël, BOURLOT Xavier, 3IMACS-AE-C, Binôme 1

30 mai 2020

Table des matières

1	Introduction	2
2	Le réseau de Petri complet	2
3	Détail du fonctionnement des modules	3
3.1	Commande du module de tri simple	3
3.2	Commande du magasin	3
3.3	Commande du préhenseur	3
3.4	Commande du convoyeur	4
4	Mesures de sécurité	4
5	Conclusion	4
A	Annexes	5
A.1	Vidéo du montage en fonctionnement	5
A.2	Code langage ST	5
A.3	Variables	6

1 Introduction

L'objectif de ce TP est de réaliser la commande d'une maquette de tri de pièces à l'aide d'un automate programmables TSX premium de TÉLÉMÉCANIQUE. La mise en place de cette commande a pour but d'assurer le déplacement des pièces stockées dans un magasin d'entrée vers une station de tri et d'effectuer le tri en fonction du type de pièce. La finalité de ce TP est d'assimiler l'ensemble des connaissances théoriques en MCSSED. Le rapport est divisé en plusieurs parties. Ces parties correspondent chacune à la commande d'une portion de la maquette.

2 Le réseau de Petri complet

Afin de gérer l'automatisation du tri des pièces nous avons mis en place le réseau de Petri suivant :

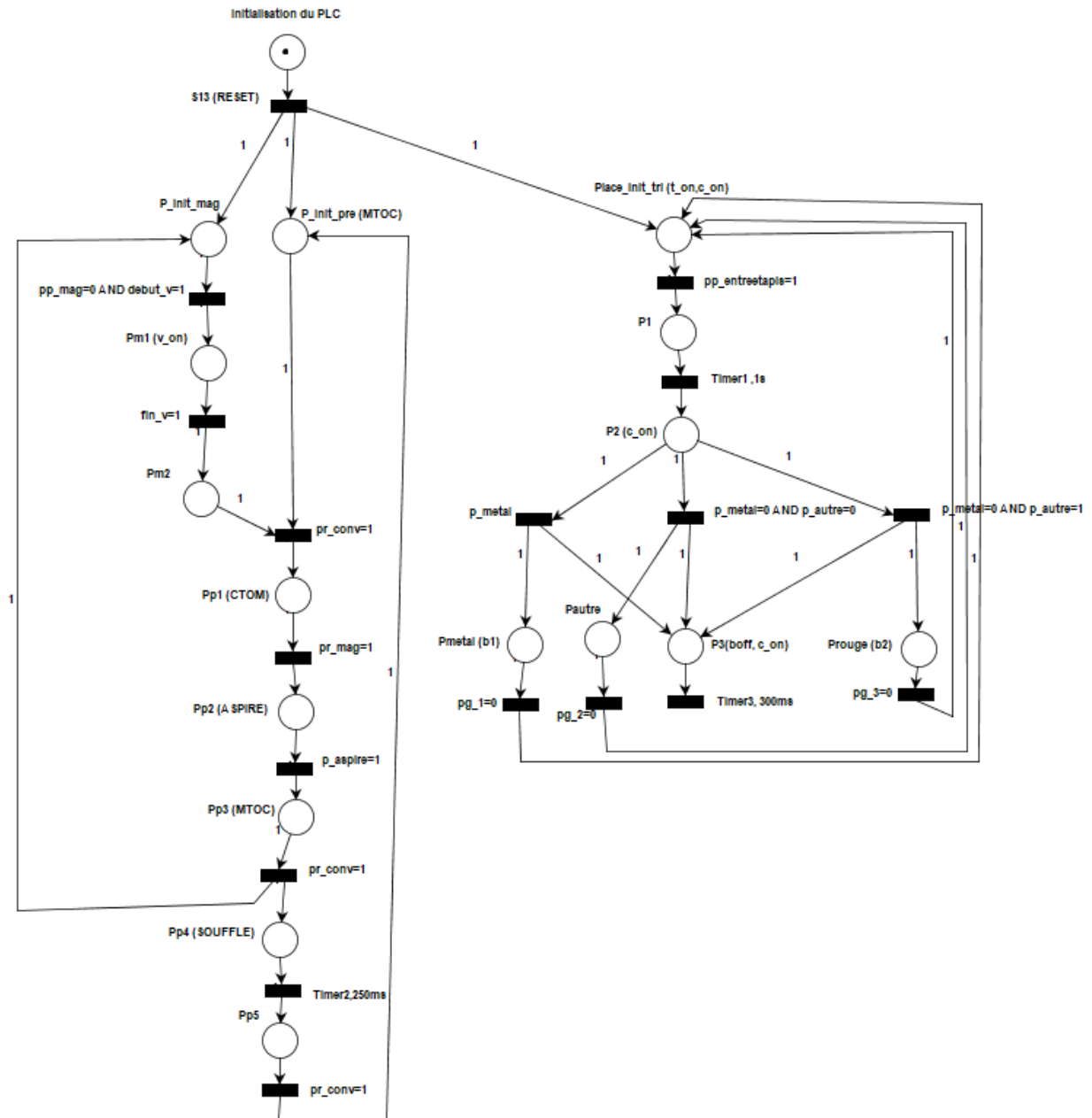


FIGURE 1 – Diagramme de Pétri du système complet

3 Détail du fonctionnement des modules

3.1 Commande du module de tri simple

La commande du module de tri a été l'élément de la maquette TSX premium le plus complexe à réaliser. En effet, pour faire fonctionner le système, il faut simultanément prendre en compte le convoyeur, le tapis du module de tri, le capteur de pièces métalliques, le capteur de pièces métalliques ou légèrement rouge, la butée ainsi que les deux goulottes. Pour ce faire nous avons implémenté le déroulement suivant :

1. Mise en marche du convoyeur.
2. Mise en marche du tapis du module de tri.
3. Temporisation de 1s lors de l'arrivée de la pièce devant la butée.
4. Analyse du type de pièce.
5. Mise en place du barrage correspondant.
6. Retrait de la butée.
7. Temporisation de 300ms pour que la pièce passe la butée.
8. Remise de la butée.
9. Passage devant le capteur de la goulotte correspondante.
10. Retour à l'état initial.

En premier lieu, nous n'avions pas mis en place la première temporisation, mais nous y avons été contraint lorsque nous avons testé le système pour des cas extrêmes. En effet, lorsque la pièce arrivait devant la butée et que celle-ci était rouge, les deux capteurs n'avaient pas le temps de réagir, puisqu'il faut que la pièce soit non métallique mais légèrement rouge. De ce fait, le système interprétait toujours la pièce comme étant noire puisqu'elle n'était pas métallique et que le capteur rouge n'avait pas le temps de se déclencher. La temporisation a donc fixé le problème.

La seconde temporisation pour la butée évite le passage d'une seconde pièce non analysée qui serait collée à la première. Même si en pratique ce cas est peu probable de se produire avec le délai généré par le bras préhenseur en amont, nous avons rencontré ce conflit en testant manuellement le module de tri et avons préféré nous en prévenir à l'avance.

L'utilisation de ce timer3 dans ce cas ne peut être visualisée correctement dans notre implémentation. En effet, la fin du timer3 redirige vers une place qui n'a aucune fonction et qui est vidée de son jeton de manière asynchrone par le franchissement d'une des trois transitions pg_x.

3.2 Commande du magasin

Afin de commander le magasin, nous avons mis en place le fonctionnement suivant :

1. Détection d'une pièce.
2. Expulsion de la pièce du magasin par avancée du vérin.
3. Retrait du vérin.
4. Attente de l'enlèvement de la pièce.

Afin de gérer l'attente dû à l'enlèvement de la pièce, le retour à l'état initial est réalisé lorsque le préhenseur a terminé son travail.

3.3 Commande du préhenseur

Le préhenseur a également été un des points les plus difficiles à maîtriser. Son utilisation n'est pas indépendante des autres systèmes et de nombreux points sont à prendre en compte. Après réflexion, nous avons mis en place le déroulement suivant :

1. Optionnel : Retour du préhenseur vers le convoyeur s'il se trouve du côté magasin.
2. Déplacement du préhenseur vers le magasin.
3. Aspiration de la pièce.
4. Déplacement du préhenseur vers le convoyeur.
5. Soufflage de la pièce.
6. Temporisation de 250ms pour que la pièce ait le temps de se décrocher.
7. Retour à l'état initial du magasin.
8. Retour à l'état initial du préhenseur.

Lors de notre premier essai, nous n'avions pas mis en place la temporisation et cela entraînait un retrait prématuré du préhenseur. Parfois, les pièces n'avaient pas le temps de se détacher et elles se retrouvaient soulevées, à la limite de la chute du convoyeur. La temporisation a donc fixé le problème.

On observe la présence d'une transition de synchronisation avant Pp1

3.4 Commande du convoyeur

Le convoyeur a été la partie la plus simple à gérer puisqu'il fonctionne quasiment sans interruption. Nous avons mis en place le modèle suivant :

1. Mise en marche du convoyeur.
2. Arrêt du convoyeur si le préhenseur est côté convoyeur, ou que la pièce arrive devant la butée, ou que la pièce n'a pas encore atteint sa goulotte.
3. Si aucune des conditions n'est vraie, retour à l'état initial.

4 Mesures de sécurité

Bien que nous n'ayons pas implémenté toutes les mesures de sécurité proposées, nous avons géré les plus importantes. Tout d'abord, nous avons éliminé le problème du passage de deux pièces collées au niveau de la butée. La temporisation est calculée afin de ne laisser passer qu'une seule pièce même si celles-ci se touchent. Deuxièmement, nous avons fait en sorte que le convoyeur s'arrête lors de la dépose d'une pièce sur celui-ci par le préhenseur. Finalement, nous avons également géré un Timer qui arrête le convoyeur si aucune pièce n'est détectée en magasin pendant 16s (non représenté sur le diagramme de Pétri, mais implémenté dans le code).

5 Conclusion

Ainsi, à travers ce TP, nous avons pu concevoir des réseaux de Pétri, puis les implémenter sur un automate programmable. Les tests effectués directement sur la maquette nous ont permis de comprendre nos erreurs et de les corriger pour fournir un système de tri de pièces fonctionnel. Nous nous sommes donc familiarisés avec les notions de places et transitions, et avons pu les relier aux capteurs et actionneurs physiques.

Des contraintes de synchronisation entre les différents modules, ainsi que de partage de ressources partagées (accès à des places par exemple) ont été abordées.

Enfin, quelques mesures de sécurité ont pu être implémentées.

Nous avons particulièrement apprécié le lien immédiat entre le réseau de Pétri et la maquette, qui permet de donner une dimension concrète au travail effectué.

A Annexes

A.1 Vidéo du montage en fonctionnement









































Vidéo du montage en fonctionnement

A.2 Code language ST

```
1 IF %S13 THEN
2     Place_Init_tri :=1;
3     P_Init_mag :=1;
4     P_init_pre :=1;
5 END_IF;
6
7 IF AND((pp_entreetapis=1),(Place_Init_tri=1))
8     THEN
9     Place_Init_tri :=0;
10    P1 :=1;
11 END_IF;
12 Timer1(IN:=P1,PT:=t#1s,Q=>P2);
13
14 IF AND((p_metal=1),(P2=1)) THEN
15     P2:=0;
16     P1:=0;
17     Pmetal:=1;
18     P3:=1;
19 END_IF;
20
21 IF AND((p_metal=0),(P2=1),(p_autre=1)) THEN
22     P2:=0;
23     P1:=0;
24     Prouge:=1;
25     P3:=1;
26 END_IF;
27
28 IF AND((p_metal=0),(P2=1),(p_autre=0)) THEN
29     P2:=0;
30     P1:=0;
31     Pautre:=1;
32     P3:=1;
33 END_IF;
34
35 Timer3(IN:=P3,PT:=t#300ms,Q=>P3_zero);
36
37 IF (P3_zero=1) THEN
38     P3:=0;
39 END_IF;
40
41 IF AND(OR(pg_1=0,pg_2=0,pg_3=0),OR(Prouge,
42     Pmetal,Pautre)=1) THEN
43     P3_zero:=0;
44     Prouge:=0;
45     Pmetal:=0;
46     Pautre:=0;
47     Place_Init_tri:=1;
48 END_IF;
49
50 Timer4(IN:=P_init_mag,PT:=t#16s,Q=>Ptimeout);
51
52 c_on :=NOT(OR(P1,Prouge,Pmetal,Pautre,Ptimeout)
53 );
54 t_on :=NOT(Ptimeout);
```

```
53 b_off:=P3;
54 b1:= Pmetal;
55 b2:= Prouge;
56
57
58
59 IF AND(pp_mag=0,debut_v=1,P_init_mag=1) THEN
60     Pm1:=1;
61     P_init_mag:=0;
62     Ptimeout:=0;
63 END_IF;
64
65 IF AND(fin_v=1,Pm1=1) THEN
66     Pm1:=0;
67     Pm2:=1;
68 END_IF;
69
70 v_on:=(Pm1);
71
72 IF AND(pr_conv=1,Pm2=1,P_init_pre=1) THEN
73     P_init_pre:=0;
74     Pp1:=1;
75 END_IF;
76
77 IF AND(pr_mag=1,Pp1=1) THEN
78     Pp1:=0;
79     Pp2:=1;
80 END_IF;
81
82 IF AND(p_aspire=1,Pp2=1) THEN
83     Pp2:=0;
84     Pp3:=1;
85
86 END_IF;
87
88 IF AND(pr_conv=1,Pp3=1) THEN
89     Pp3:=0;
90     Pp4:=1;
91     Pm2:=0;
92     P_init_mag:=1;
93 END_IF;
94
95 Timer2(IN:=Pp4,PT:=t#250ms,Q=>Pp5);
96
97 IF AND(pr_conv=1,Pp5=1) THEN
98     Pp5:=0;
99     Pp4:=0;
100    P_init_pre:=1;
101 END_IF;
102
103 MTOC:=OR(P_init_pre,Pp3);
104 CTOM:=(Pp1);
105 ASPIRE:=Pp2;
106 SOUFFLE:=Pp4;
```

A.3 Variables

	P1	BOOL		0
	P2	BOOL		0
	Pmetal	BOOL		0
	Prouge	BOOL		0
	pr_conv	EBOOL	%I0.3.5	
	pp_entree...	EBOOL	%I0.3.8	
	p_metal	EBOOL	%I0.3.9	
	p_autre	EBOOL	%I0.3.10	
	c_on	EBOOL	%Q0.2.5	
	t_on	EBOOL	%Q0.2.6	
	b1	EBOOL	%Q0.2.7	
	b2	EBOOL	%Q0.2.8	
	b_off	EBOOL	%Q0.2.9	
	Place_Init...	BOOL		
	Pautre	BOOL		0
	pg_1	EBOOL	%I0.3.11	
	pg_2	EBOOL	%I0.3.12	
	pg_3	EBOOL	%I0.3.13	
	v_on	EBOOL	%Q0.2.0	
	pp_mag	EBOOL	%I0.3.6	
	debut_v	EBOOL	%I0.3.1	
	fin_v	EBOOL	%I0.3.2	
	P_Init_mag	BOOL		
	Pm1	BOOL		0
	Pm2	BOOL		0
	P_init_pre	BOOL		
	p_aspire	EBOOL	%I0.3.3	
	pr_mag	EBOOL	%I0.3.4	
	ASPIRE	EBOOL	%Q0.2.1	
	SOUFFLE	EBOOL	%Q0.2.2	
	CTOM	EBOOL	%Q0.2.3	
	MTOC	EBOOL	%Q0.2.4	
	Pp1	BOOL		0
	Pp2	BOOL		0
	Pp3	BOOL		0
	Pp4	BOOL		0
	Pp5	BOOL		0
	P3	BOOL		0
	P3_zero	BOOL		0
	Ptimeout	BOOL		0
