APIMOU

Version 2.6

Manuel d'utilisation

AVERTISSEMENT – EXCLUSION DE RESPONSABILITÉ

Ce logiciel est fourni "tel quel", sans garantie d'aucune sorte, explicite ou implicite, y compris, mais sans s'y limiter, les garanties de bon fonctionnement, d'adéquation à un usage particulier ou d'absence d'erreurs.

En aucun cas l'auteur ne pourra être tenu responsable de tout dommage, perte de données, panne matérielle ou tout autre effet indésirable résultant directement ou indirectement de l'utilisation de ce logiciel, même si l'auteur a été informé de la possibilité de tels dommages.

L'utilisation de ce logiciel se fait sous votre seule responsabilité.

Table des matières

1 Présentation 3
2 Informations de licences 3
3 Prise en main rapide 4
3.1 Exemple de programme en logique combinatoire 4
3.2 Exemple de programme en logique séquentielle 4
4 Variables 7
4.1 Modification de mnémonique 7
4.2 Variables d'entrée 7
4.3 Variables de sortie 7
4.4 Variables d'étape 8
4.5 Bits internes 8
4.6 Variables de temporisation 8
4.7 Variables de comptage 9
5 Equations logiques 9
6 Temporisations et Compteurs 9
7 Grafcet 11
7.1 Etapes et actions 11
7.2 Transitions et réceptivités 12
7.3 Liaisons orientées 13
7.4 Divergences et convergences en ET 13
7.5 Divergences et convergences en OU 14
7.6 Grafcets multiples 14
7.7 Edition multipages 14
7.8 Ordres de forçage 14
7.9 Fonctionnalités d'édition diverses 15
8 Fonctions particulières 17
8.1 Signal d'horloge ou clignotant 17
8.2 Front montant ou front descendant 17
8.3 Bit d'initialisation 18
9 Fonctionnement en mode simulation 19
10 Fonctionnement en mode Connecté 19
11 Génération de code Arduino 20
12 Transfert direct d'un programme vers une carte Arduino 21
13 Génération de code ESP32 22
13.1 Généralités 22
13.2 Prérequis 23
13.3 Modes Station et Access Point 23
13.4 Procédure 24
14 Transfert direct d'un programme vers un ESP32 26
15 Gestion des fichiers 27



1 Présentation

Apimou est un simulateur d'Automate Programmable Industriel qui permet créer et tester des programmes simples mettant en œuvre des fonctions de logique combinatoire et séquentielle. Il peut fonctionner en mode simulation ou en mode connecté, relié à une partie opérative logicielle ou réelle. Il dispose de 32 entrées et 32 sorties binaires ainsi que de variables internes (variables d'étape, variables de temporisation, variables de comptage, bits internes).

Les sorties peuvent être des fonctions directes des entrées, et/ou dépendre d'une séquence matérialisée par un grafcet.

L'archive Apimou.zip comprend :

-> Apimou.exe, le programme principal (qui se suffit éventuellement à lui-même en mode simulation),

-> Apimou.pdf, ce document,

-> Bees.exe, Afficheur.exe, Feux_tri.exe, Convoyeur&Verins.exe, Ascenseur.exe, Chariots.exe (et peut-être d'autres) qui sont les parties opératives logicielles disponibles actuellement. Apimou ne pourra communiquer qu'avec une partie opérative à la fois,

-> un dossier Sources qui contient le code source (C++Builder 10.3.3) de la partie opérative Bees. Ce code est fourni pour comprendre au besoin le protocole de communication entre Apimou et les parties opératives. On peut parfaitement se dispenser d'ouvrir ce dossier.

-> un dossier Fichiers api qui contient, dans les sous-dossiers correspondants, quelques exemples de programmes pour les parties opératives proposées. Les sous-dossiers "pour Afficheur", "pour Ascenseur", "pour Feux tricolores", "pour Convoyeur et vérins" ne contiennent qu'un unique fichier "config_(nom de la PO).api". Lorsqu'on ouvre ce fichier dans Apimou on trouve simplement des mnémoniques (pour les entrées/sorties) correspondant à ce que l'on voit dans la partie opérative correspondante (libellés des boutons, des capteurs, etc.). Le programme reste à créer. On trouve sur Internet des exercices relatifs à ces parties opératives, Apimou permet notamment de tester en pratique les solutions de ces exercices.

2 Informations de licences

Apimou est un logiciel gratuit.

Le code source d'Apimou est disponible sur le site de l'auteur, à défaut sur simple demande.

Apimou utilise le composant DOSCommand de TurboPack, sous licence MPL 1.1.

Apimou utilise la librairie Boost, sous licence Boost Software License 1.0.

Apimou utilise DelphiZXingQRCode de Foxit Software, sous licence Apache-2.0.

3 Prise en main rapide

Les deux exemples qui suivent seront testés en mode simulation.

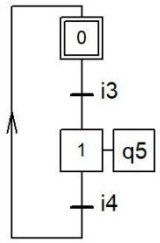
3.1 Exemple de programme en logique combinatoire

1. Lancer Apimou
2. Vérifier que le mode Simulation est sélectionné (Mode→Simulation)
3. Mettre l'automate en Run (Automate→Run)
4. Cliquer sur les cases %I1.0, %I1.1, etc. (mnémoniques correspondants i0, i1, etc.) : on voit sur l'automate les leds d'entrée s'allumer (et c'est tout : pour l'instant l'automate n'est pas programmé)
5. Mettre l'automate en Stop (Automate→Stop)
6. Cliquer sur l'onglet Sorties, et dans la colonne Equation de la sortie %Q2.3 (mnémonique q3) taper i2+i7
7. Mettre l'automate en Run (Automate→Run)
8. Cliquer sur l'onglet Entrées et tester le programme : cliquer sur les cases %I1.0, %I1.1, etc. On constate que la sortie 3 (adresse %Q2.3, mnémonique q3) s'active (la led de sortie correspondante s'allume) si on clique sur %I1.2 (mnémonique i2) ou sur %I1.7 (mnémonique i7).

On a ainsi programmé l'automate, avec un exemple purement combinatoire (q3 = i2 + i7), et testé ce programme en mode simulation.

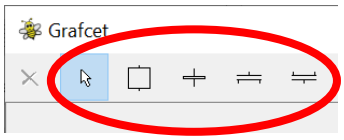
3.2 Exemple de programme en logique séquentielle

Soit à programmer le grafcet ci-dessous :

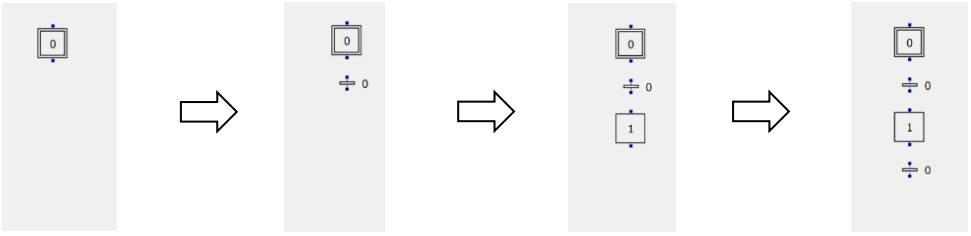


(réalisé avec Editsab)

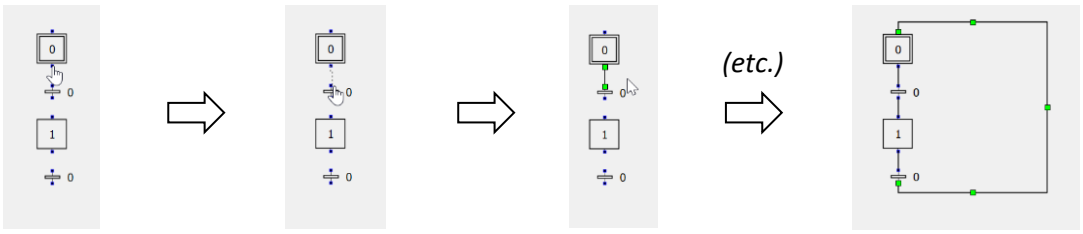
1. Automate-→Stop.
2. Fichier→Nouveau (on efface tout).
3. Affiche→Grafcet. Dans la barre d'outils de la fiche qui apparait se trouvent les boutons correspondant aux éléments nécessaires pour la constitution d'un grafcet : étape, transition, divergence et convergence en ET.



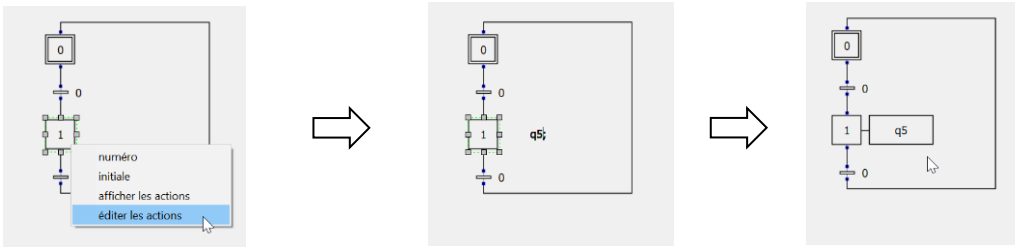
4. Choisir une étape et la positionner sur la fiche, puis une transition et la positionner en dessous. Recommencer de façon à obtenir tous les éléments voulus.
   La première étape insérée sur la fiche est par défaut une étape initiale (carré double). Les numéros des étapes sont automatiquement incrémentés. Les transitions ont par défaut des réceptivités toujours fausses (valeur 0).



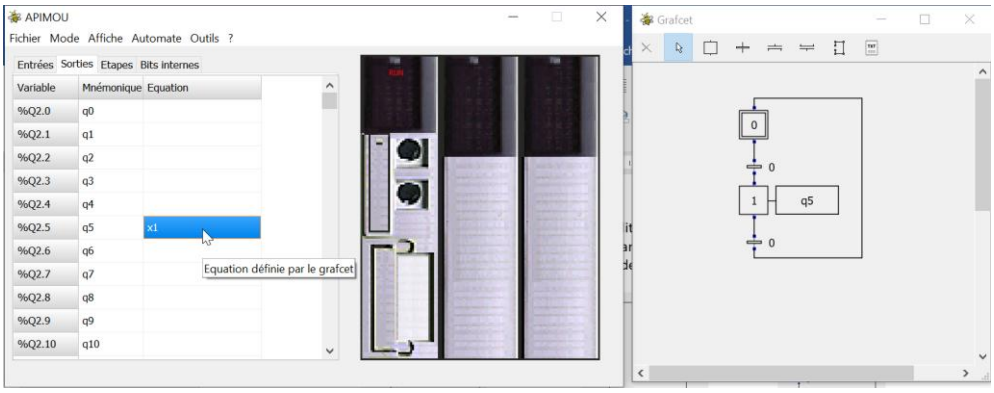
5. Relier les étapes aux transitions et les transitions aux étapes en cliquant sur un point de liaison (petit carré bleu), tirant puis relachant sur le second point de liaison. Pour la dernière liaison (dernière transition vers étape initiale), cliquer sur la poignée de droite (petit carré vert) et tirer vers la droite afin de l'écarter un peu pour laisser de la place aux actions.



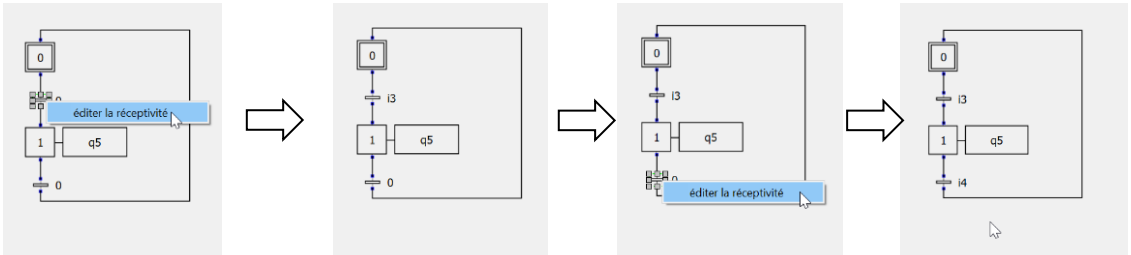
6. Faire un clic droit sur l'étape 1 et choisir éditer les actions. Saisir l'action choisie (ici q5). Le point-virgule ";" sert à séparer les actions. Dans la mesure où il n'y en a qu'une ici, on peut le supprimer (ce n'est pas une obligation). Valider en cliquant sur un espace libre de la fiche ou avec la combinaison de touches Ctrl+Entrée.



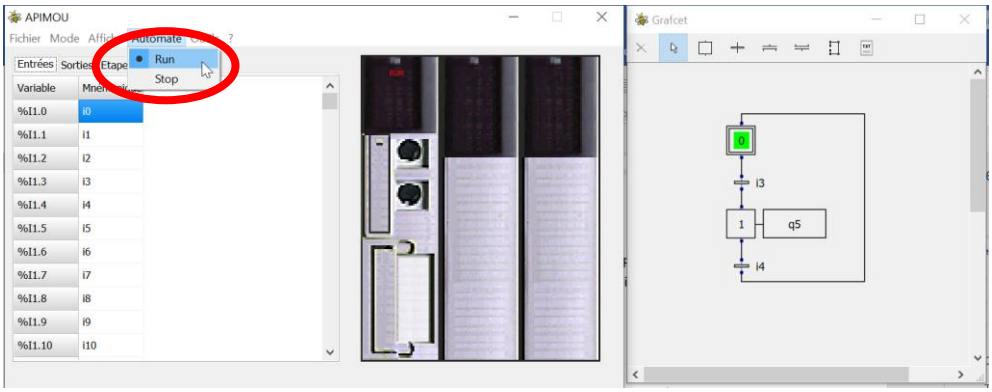
On peut remarquer dans la fiche principale, en cliquant sur l'onglet sorties, que l'équation de la sortie q5 a été automatiquement inscrite (q5 = x1).



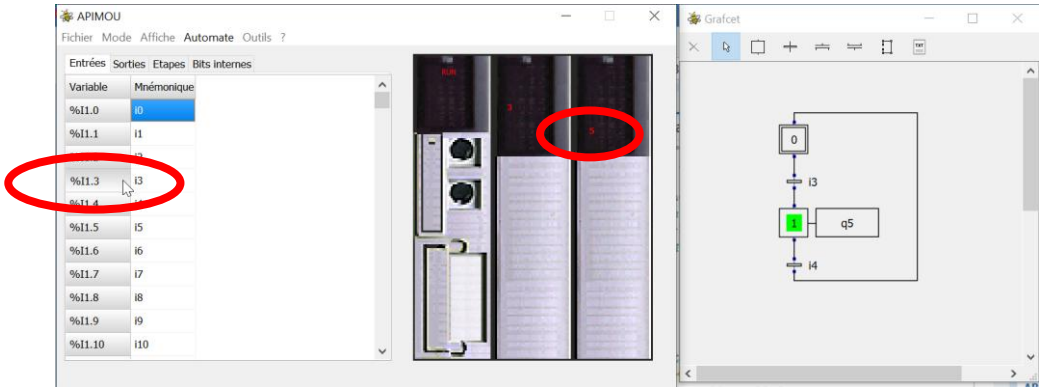
7. Faire un clic droit sur la première transition et choisir éditer la réceptivité. Saisir la réceptivité choisie (ici i3). Valider en cliquant sur un espace libre de la fiche ou avec la combinaison de touches Ctrl+Entrée. Faire de même pour la seconde transition, de réceptivité i4. La saisie du grafcet (du programme) est terminée.



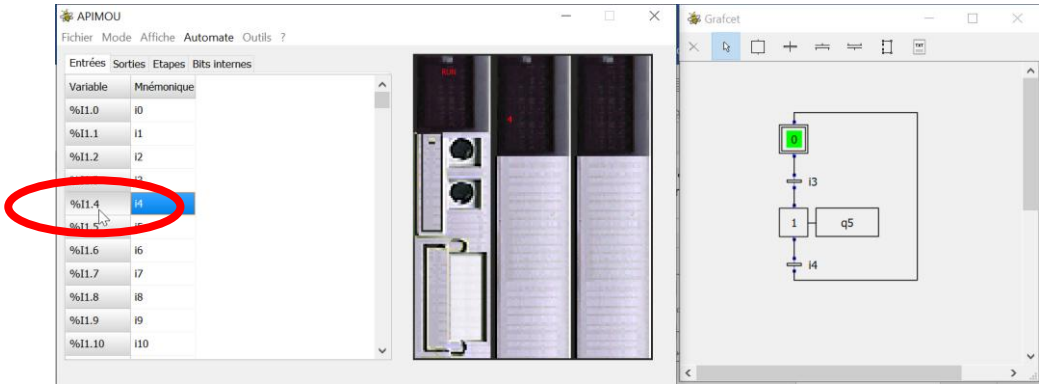
8. Essai du programme : dans la fiche principale, menu Automate→Run. Un carré vert (le "jeton") indique sur le grafcet que l'étape 0 (initiale) est active.



9. Cliquer sur l'onglet Entrées puis cliquer sur la case %I1.3 (mnémonique i3) : on voit sur le grafcet que l'étape 0 a été désactivée et que l'étape 1 est maintenant active. L'action correspondante s'exécute, donc la sortie q5 passe à 1 (la led de la sortie 5 s'allume).



10. Cliquer sur la case %I1.4 (mnémonique i4) : on voit sur le grafcet que l'étape 1 a été désactivée et que l'étape 0 est maintenant active, on est revenu à l'étape initiale. La sortie q5 repasse à 0 (la led de sortie correspondante s'éteint).



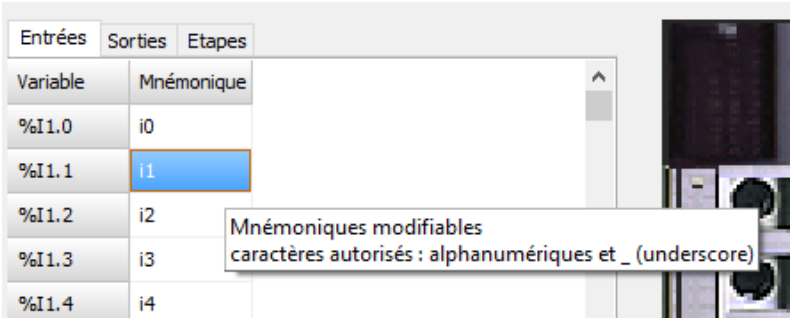
4 Variables

Les variables d'entrée, de sortie, d'étape et les bits internes sont identifiés par un couple (adresse, mnémonique). Les variables de temporisation et de comptage ne sont identifiées que par un mnémonique (t0 à t7 pour les temporisations, ct0 à ct7 pour les compteurs).
Les adresses ne servent ici qu'à des fins de repérage, il n'est pas possible de les utiliser dans les expressions logiques.
Les mnémoniques sont quant à eux essentiels, ils sont utilisés dans les expressions logiques. Chaque mnémonique est unique.

4.1 Modification de mnémonique

Les mnémoniques proposés par défaut peuvent être modifiés à partir de la fenêtre principale, à condition que l'automate soit en STOP.

Les caractères autorisés pour les mnémoniques sont tous les caractères alphanumériques (non accentués) ainsi que l'underscore (_). Un mnémonique peut n'être constitué que d'un caractère, mais les mnémoniques 0 et 1 sont réservés (0 correspondant à une condition toujours fausse et 1 à une condition toujours vraie). Les mnémoniques t0 à t7 et ct0 à ct7 sont également réservés (variables de temporisation et de comptage).
Les mnémoniques sont sensibles à la casse : A1 et a1 sont deux mnémoniques différents.
Pour modifier un mnémonique il faut le sélectionner puis cliquer afin d'entrer dans le mode édition du tableau :



4.2 Variables d'entrée

Elles correspondent aux états des capteurs tout ou rien reliés à la carte d'entrées de l'automate.
Leurs valeurs sont lues par l'automate. En mode simulation ces variables sont forcées : on leur affecte une valeur en cliquant sur les cases %I.x.

4.3 Variables de sortie

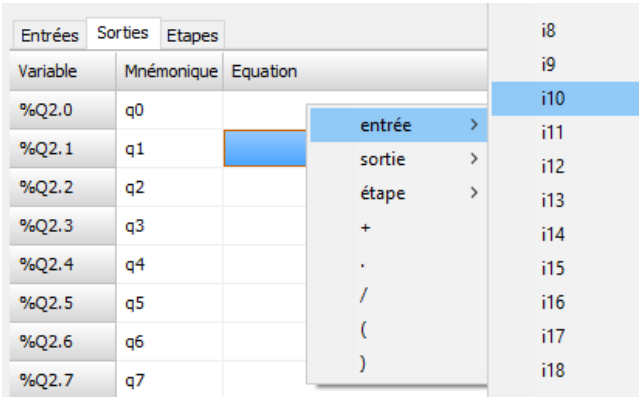
Elles correspondent aux états des (pré-, micro-) actionneurs tout ou rien reliés à la carte de sorties de l'automate.
Leurs valeurs sont calculées par l'automate.
Ces calculs sont les résultats de fonctions logiques qui sont ici définies sous l'onglet Sorties de la fiche principale, dans la colonne Equation.

Si un (ou plusieurs) grafcet a (ont) été créé(s), les équations des sorties correspondant aux actions de ce(s) grafcet sont automatiquement définies et il n'est pas possible de les modifier. On peut aussi ne pas programmer d'action dans le grafcet et saisir manuellement les équations des sorties, ce qui laisse plus de liberté.

Il y a deux façons de saisir une équation logique pour une variable de sortie :

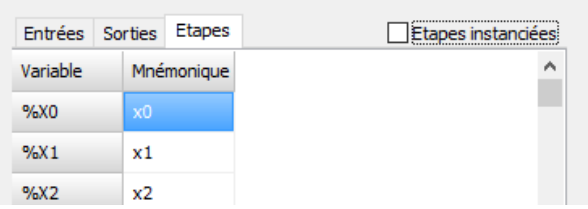
- soit en sélectionnant la cellule puis en cliquant dessus pour entrer dans le mode édition. On saisit alors l'équation au clavier.

- soit en sélectionnant la cellule puis en effectuant un clic droit, sans entrer dans le mode édition, afin de sélectionner les variables et les opérateurs nécessaires :



4.4 Variables d'étape

Elles correspondent aux états des étapes du (ou des) grafcet. Ce sont des variables internes dont les valeurs sont calculées par l'automate.
On peut, par défaut, visualiser toutes les variables d'étape, même si toutes ne sont pas utilisées.
Pour ne visualiser que les étapes utilisées (instanciées), il faut cocher la case Etapes instanciées de l'onglet Etapes de la fenêtre principale :



4.5 Bits internes

Les bits internes servent à ce qu'on veut, ils sont définis, comme les variables de sortie, par une équation logique.

On peut par exemple les utiliser pour exprimer :

- une condition logique "complexe", qui sera ainsi résumée à une variable,
- un front montant ou descendant,
- un bit d'initialisation.

4.6 Variables de temporisation

Les variables de temporisation (t0 à t7) sont des variables internes qui passent à 1 lorsque la temporisation correspondante (T0 à T7) est écoulée (voir § Temporisations et Compteurs). Elles peuvent être utilisées selon le besoin pour exprimer une sortie, un bit interne, une réceptivité ou toute autre expression logique.

4.7 Variables de comptage

Les variables de comptage (ct0 à ct7) sont des variables internes qui passent à 1 lorsque le comptage correspondant (CT0 à CT7) est terminé (voir § Temporisations et Compteurs). Elles peuvent être utilisées selon le besoin pour exprimer une sortie, un bit interne, une réceptivité ou toute autre expression logique.

5 Equations logiques

Les équations logiques permettent de définir les sorties, les réceptivités d'un grafcet, les bits internes. Elles permettent aussi de déclarer les conditions d'activation ou de remise à 0 des blocs fonction (temporisation, comptage). Les règles qui suivent s'appliquent dans tous les cas :

Les opérandes sont les mnémoniques des variables d'entrée, d'étape, de temporisation, de comptage, éventuellement de sortie, et des bits internes.

Les trois seuls opérateurs utilisés sont ET symbolisé par un point (.), OU symbolisé par un plus (+) et NON symbolisé par un slash (/).

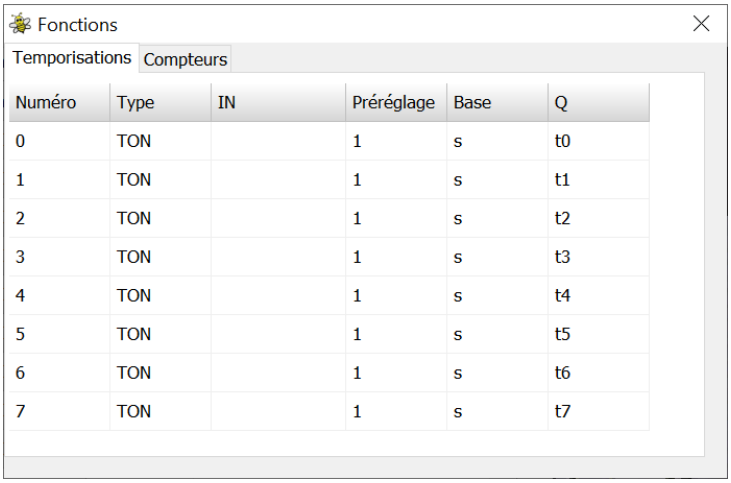
Exemple : I1 OU i2 ET i3 ET NON i4 s'écrira i1+i2.i3.i4/

N.B. :

- l'opérateur / s'applique à ce qui le précède (variable ou expression entre parenthèses),
- une équation est déclarée directement dans l'espace qui la reçoit, elle ne commence pas par un signe d'égalité,
- des parenthèses permettent si nécessaire de prioriser les opérations : i1+i2.i3 correspond à i1+(i2.i3), différent de (i1+i2).i3,
- l'opérateur . n'est pas implicite : i1.(i2+i3) ne peut pas s'écrire i1(i2+i3),
- il est possible de séparer les variables des opérateurs par des espaces,
- les valeurs 0 et 1 (faux et vrai) peuvent être utilisées (1 permet par exemple de déclarer une réceptivité toujours vraie).

6 Temporisations et Compteurs

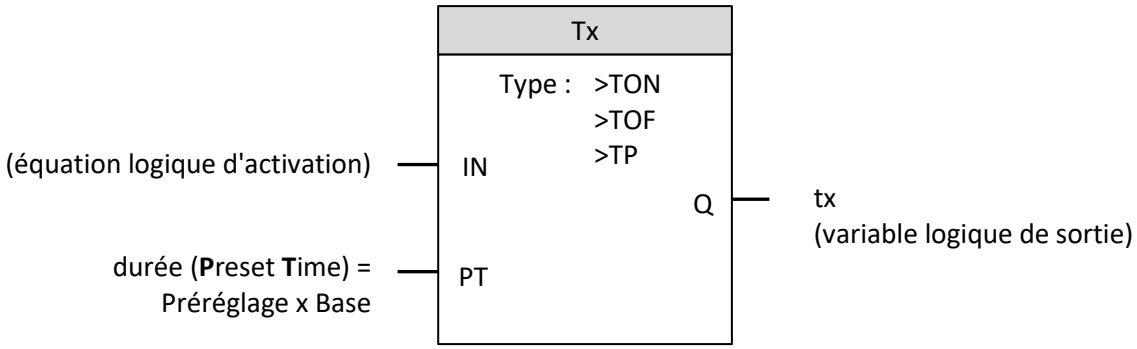
Apimou permet d'utiliser 8 temporisations (numérotées de 0 à 7) paramétrables en "TON", "TOF", "TP" et 8 compteurs (également numérotés de 0 à 7) de type "CTU". On accède à ces fonctions par le menu Affiche → Fonctions.



Numéro	Type	IN	Préréglage	Base	Q
0	TON		1	s	t0
1	TON		1	s	t1
2	TON		1	s	t2
3	TON		1	s	t3
4	TON		1	s	t4
5	TON		1	s	t5
6	TON		1	s	t6
7	TON		1	s	t7

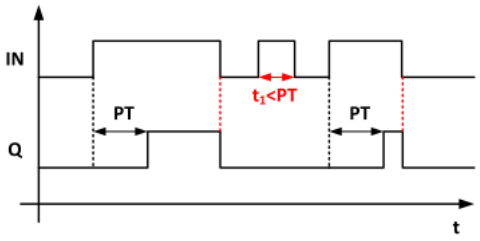
A chaque temporisation ou compteur est associée une variable logique de sortie Q imposée (t4 pour la temporisation 4 et ct3 pour le compteur 3 par exemple).

Les temporisations peuvent se représenter par le bloc fonction suivant :



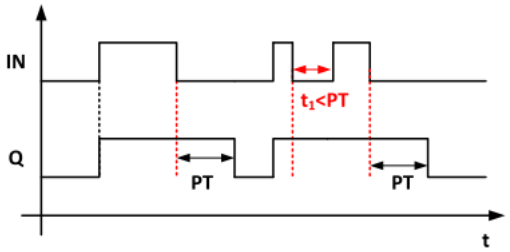
L'équation logique d'activation (IN) peut être saisie au clavier ou avec le menu contextuel qui apparaît lorsqu'on effectue un clic droit sur la cellule. Elle peut, comme toute autre équation, n'être constituée que d'une variable.

Chronogramme d'une temporisation TON (retard à l'enclenchement) :



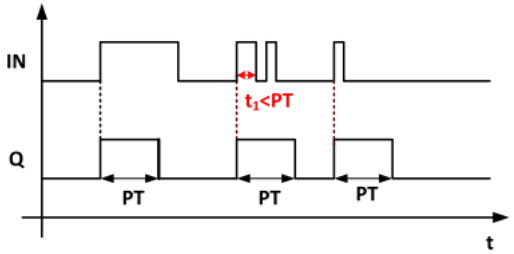
source : International Journal of Mechatronics and Applied Mechanics, 2018, Issue 4

Chronogramme d'une temporisation TOF (retard au déclenchement) :



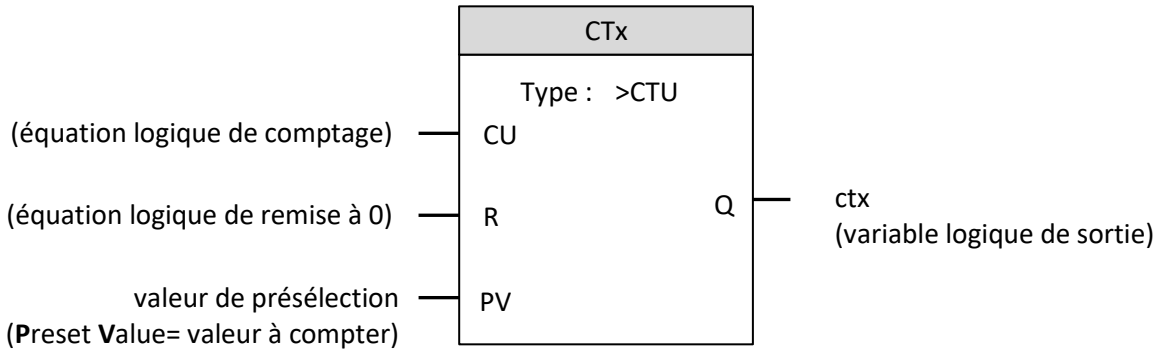
source : International Journal of Mechatronics and Applied Mechanics, 2018, Issue 4

Chronogramme d'une temporisation TP (impulsion de durée fixe) :



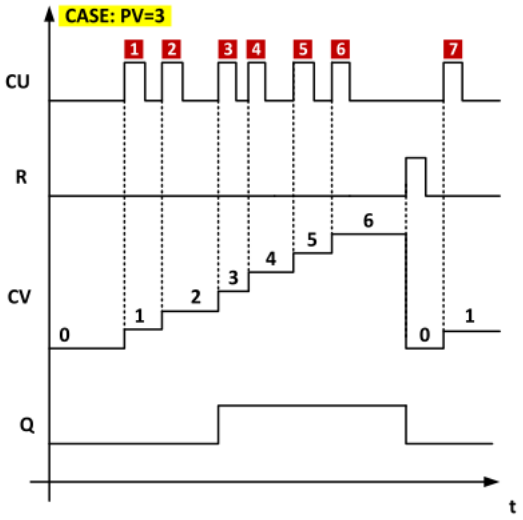
source : International Journal of Mechatronics and Applied Mechanics, 2018, Issue 4

Les compteurs peuvent se représenter par le bloc fonction suivant :



Les équations logiques de comptage (CU) ou de remise à 0 (R) peuvent être saisies au clavier ou avec le menu contextuel qui apparaît lorsqu'on effectue un clic droit sur la cellule. Elles peuvent, comme toute autre équation, n'être constituées que d'une variable.

Chronogramme d'un compteur CTU (pour PV = 3) :



CV : Current Value = valeur courante

source : International Journal of Mechatronics and Applied Mechanics, 2018, Issue 4

7 Grafcet

L'essentiel de ce qu'il y a à connaître pour saisir un grafcet est décrit dans le paragraphe correspondant de prise en main rapide.

7.1 Etapes et actions

Les étapes peuvent être simples ou initiales (activées au passage en RUN de l'automate). Le choix se fait par un menu contextuel (clic droit sur l'étape -> initiale).

Chaque étape a un numéro unique. Il est possible de le modifier, toujours par menu contextuel (clic droit sur l'étape -> numéro, entrer la valeur souhaitée, valider en cliquant sur un espace libre de la fiche ou avec la combinaison de touches Ctrl+Entrée).

La définition d'une action associée à une étape se fait par menu contextuel (clic droit sur l'étape -> éditer les actions, entrer l'expression souhaitée, valider en cliquant sur un espace libre de la fiche ou avec la combinaison de touches Ctrl+Entrée).

Si plusieurs actions sont associées à une même étape elles doivent être séparées par un point-virgule (;).

Une action continue doit être déclarée par le mnémonique de la sortie correspondante. Il n'est pas possible d'affecter un bit interne à une action (on peut toujours le faire hors grafcet en définissant l'équation du bit souhaité, m2=x5 par exemple si on veut activer le bit m2 avec l'étape 5).

Une action conditionnelle doit être déclarée par la syntaxe condition ? sortie.

Les équations des sorties correspondant aux actions du (des) grafcet sont *automatiquement* définies et il n'est pas possible de les modifier. On peut aussi ne pas programmer d'action dans le grafcet et saisir manuellement les équations des sorties.

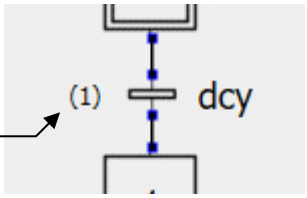
Exemples d'actions :

Grafcet	Grafcet Apimou	Equations des sorties (définies automatiquement)
3 — q7	3 — q7	q7 = x3
3 — q7 9 — q7	3 — q7 9 — q7	q7 = x3 + x9
3 — q7 q10	3 — q7 ; q10	q7 = x3 q10 = x3
I0.(i1/+i4) 3 — q7 q10	3 — i0.(i1/+i4)?q7 ; q10	q7 = x3.(i0.(i1/+i4)) q10 = x3
I0.(i1/+i4) 3 — q7 q10 9 — q7	3 — i0.(i1/+i4)?q7 ; q10 9 — q7	q7 = x3.(i0.(i1/+i4)) + x9 q10 = x3

7.2 Transitions et réceptivités

Les transitions sont repérées par un numéro qui les identifie de façon unique. Il est possible d'afficher ou de masquer ces repères par un clic droit sur une transition quelconque, en choisissant "afficher les repères".

Repère de transition



Une transition a par défaut une réceptivité associée toujours fausse (0), ce qui implique que même si la transition est validée (étapes amont actives), elle ne sera jamais franchie. La modification d'une réceptivité se fait par menu contextuel (clic droit sur la transition -> éditer la réceptivité, entrer l'expression souhaitée, valider en cliquant sur un espace libre de la fiche ou avec la combinaison de touches Ctrl+Entrée).

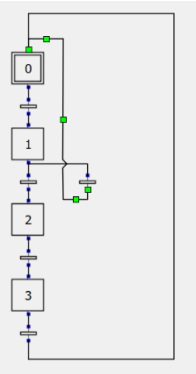
Une réceptivité toujours vraie est définie par la valeur 1. Une réceptivité non définie équivaut à une réceptivité toujours fausse.

7.3 Liaisons orientées

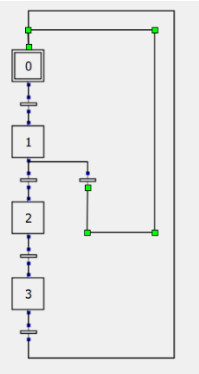
Les liaisons étape <-> transition ne peuvent être dessinées que dans le sens de leur lecture :

- ✓ Pour une liaison étape->transition on part du point de liaison aval de l'étape pour aller vers le point de liaison amont de la transition, l'inverse est impossible.
- ✓ Pour une liaison transition->étape on part du point de liaison aval de la transition pour aller vers le point de liaison amont de l'étape, l'inverse est impossible.

Si le mode de dessin par défaut de la liaison (automatique) ne donne pas satisfaction visuellement, en raison de lignes qui peuvent se croiser, on peut choisir dans la barre d'outils le mode "liaison manuelle" qui permet de réaliser des lignes multi-segments et d'agencer ces segments comme souhaité.



liaison automatique



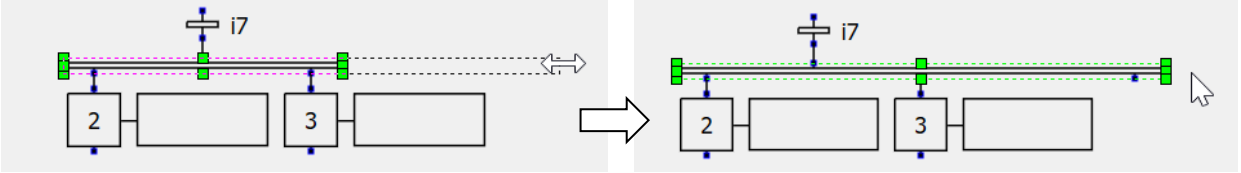
liaison manuelle

Les constructions ne correspondant pas aux règles du grafcet, en particulier la règle d'alternance étape-transition, ne sont pas permises. Tout ce qui peut être dessiné est par conséquent conforme.

7.4 Divergences et convergences en ET

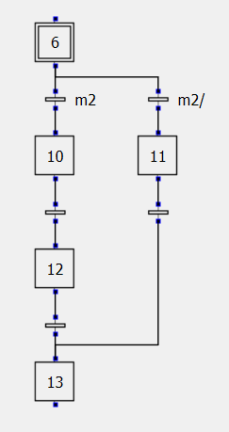
On se sert de ces constructions pour les séquences simultanées.

Les divergences ont une entrée et par défaut deux sorties. Si on souhaite démarrer trois séquences simultanées (ou plus) il suffit de redimensionner le bloc en le tirant vers la droite :



Le principe est le même pour les convergences.

7.5 Divergences et convergences en OU

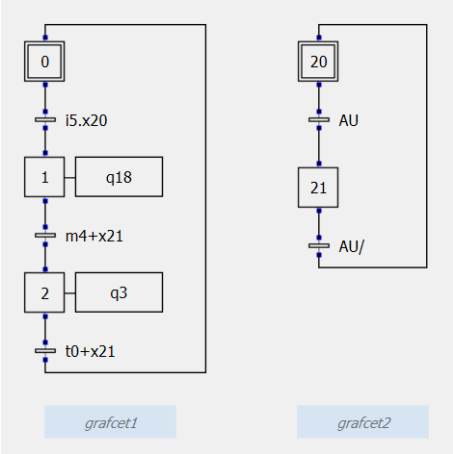


On se sert de ces constructions pour les sélections (ou les sauts) de séquences.

Il n'y a rien ici de particulier à signaler, il faut simplement veiller à ce que les réceptivités des transitions sur lesquelles se fait la sélection soient (logiquement ou physiquement) exclusives, pour éviter une prolifération de jeton.

7.6 Grafcets multiples

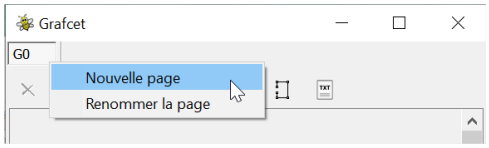
Il est possible, pour une même application, de saisir plusieurs grafcet, synchronisés ou indépendants. Exemple de grafcet synchronisés :



7.7 Edition multipages

Si plusieurs grafcets doivent être saisis, ils peuvent l'être chacun sur une page différente. Si l'on veut structurer le programme en hiérarchisant les grafcets et en utilisant des ordres de forçage, cette façon de faire est obligatoire : chaque grafcet est ainsi nommé et on peut alors déclencher un ordre de forçage désignant un grafcet précis.

Par défaut il n'y a qu'une page sur l'espace d'édition, nommée "G0". On peut en ajouter en faisant un clic droit sur l'onglet et en choisissant "nouvelle page" dans le menu contextuel qui apparaît.



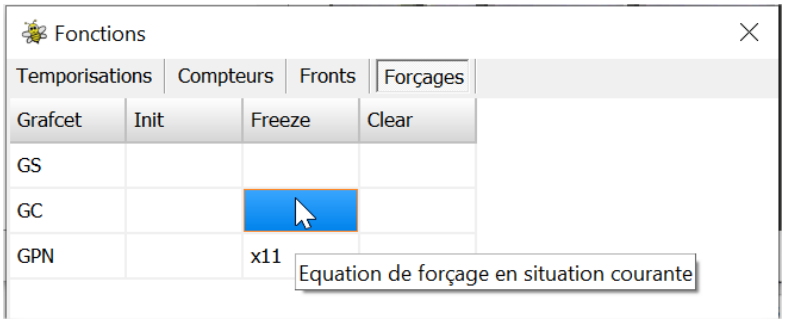
7.8 Ordres de forçage

Un système complet de logique de commande nécessite souvent une structuration qui permet de traiter en les coordonnant plusieurs modes de marche prenant en compte toutes les situations envisageables (cf GEMMA et grafcets hiérarchisés qui en découlent). Il peut être utile dans ce cas d'avoir recours à des ordres de forçage permettant de mettre immédiatement un grafcet dans une situation particulière.

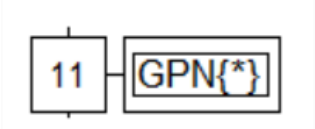
Apimou permet trois actions de forçage pour chaque grafcet : INIT, FREEZE et CLEAR. Ces actions sont disponibles dans le menu Affiche -> Fonctions, onglet Forçages.

INIT	Réinitialise le grafcet : l'étape initiale est activée (les étapes initiales s'il y en a plusieurs), les autres désactivées.
FREEZE	Fige le grafcet dans la situation courante : il n'évolue plus, aucune transition ne peut être franchie quelles que soient les valeurs des réceptivités.
CLEAR	Met le grafcet dans une situation vide : toutes les étapes, initiales ou non, sont désactivées

Chacune de ces actions est déclenchée par une équation logique. Elle peut évidemment se résumer à une simple variable d'étape.



La configuration ci-dessus, par exemple, correspond en représentation grafcet normée à :



(réalisé avec Editsab)

NB : si deux ou plus de ces trois ordres devaient être simultanément actifs pour un même grafcet (ce qui ne devrait normalement pas se produire), CLEAR est prioritaire sur INIT, lui-même prioritaire sur FREEZE.

7.9 Fonctionnalités d'édition diverses

Pour insérer plusieurs blocs (étape ou transition ou commentaire) sans cliquer à chaque fois sur la barre d'outils : appuyer sur la touche *Maj (Shift)* du clavier et sur le *bouton Etape* ou sur le *bouton Transition* ou sur le *bouton Commentaire* de la barre d'outils puis relâcher la touche *Maj (Shift)*. Cliquer sur l'espace d'édition autant de fois que nécessaire en déplaçant la souris pour répartir les blocs. Cliquer sur le bouton *Cancel inserting* de la barre d'outils pour arrêter.

Pour aligner verticalement plusieurs éléments : sélectionner les éléments (cliquer-tirer-lâcher) puis cliquer sur le bouton droit de la souris et choisir *aligner verticalement* dans le menu contextuel.

Pour répartir verticalement plusieurs éléments : sélectionner les éléments (cliquer-tirer-lâcher) puis cliquer sur le bouton droit de la souris et choisir *répartir verticalement* dans le menu contextuel.

Pour augmenter ou diminuer l'espace entre plusieurs éléments verticalement alignés et répartis : sélectionner les éléments (cliquer-tirer-lâcher) puis appuyer sur la touche *Alt* du clavier, et, en maintenant la touche enfoncée, faire tourner la *molette de la souris* vers le haut ou vers le bas.

Pour insérer une séquence linéaire de 1 à 9 étapes : placer la souris sur l'espace d'édition à l'endroit où la séquence doit être insérée puis appuyer simultanément sur la touche *Ctrl* du clavier et sur l'une des *touches 1 à 9* du clavier (pas celles du pavé numérique).

Pour zoomer/dézoomer : appuyer sur la touche *Ctrl* du clavier, et, en maintenant la touche enfoncée, faire tourner la *molette de la souris* vers le haut ou vers le bas.

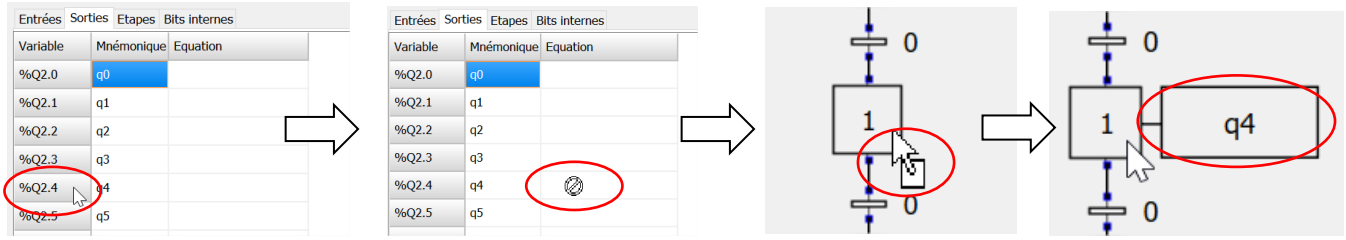
Pour revenir au niveau de zoom par défaut (100%) : cliquer sur le *bouton central de la souris*.

Pour renseigner les actions et réceptivités par glisser-déposer (drag & drop) :

Entrées, *sorties*, *étapes* et *bits internes* peuvent être glissés-déposés sur le grafcet en cliquant avec le bouton gauche sur la case *Variable* correspondante de la fiche principale puis en déplaçant la souris vers le grafcet et en relâchant le bouton sur une étape ou une transition. Pour les *temporisations* et les *compteurs* on peut faire de même en cliquant sur la case de sortie (Q) de la fiche fonctions correspondant à la temporisation ou au compteur souhaité.

Les *étapes* n'acceptent que les *variables de sortie*, les *transitions* acceptent *toutes* les variables.

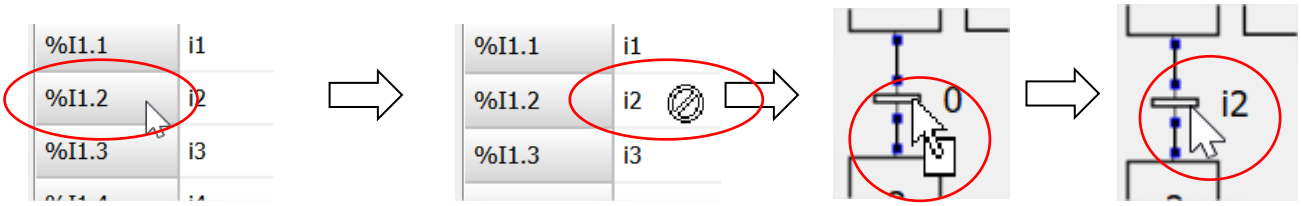
→ Cas d'une variable de sortie glissée-déposée sur une étape :



Si la sortie est déposée sur une étape possédant déjà une ou plusieurs actions, elle s'ajoute aux actions existantes. Il n'est pas possible de déclarer de cette façon des actions conditionnelles.

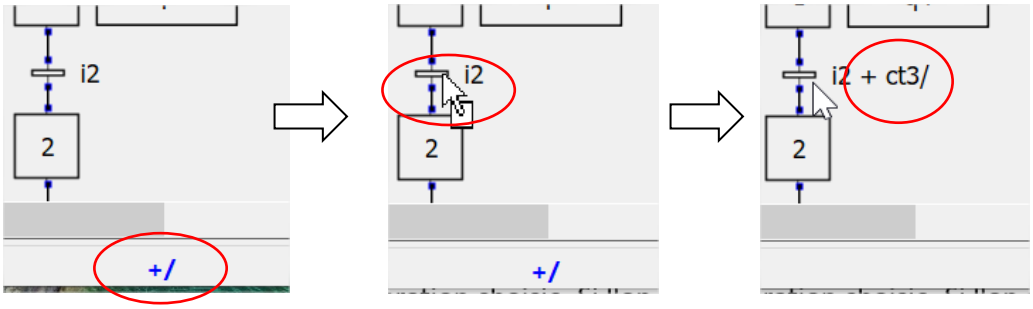
→ Cas d'une variable quelconque glissée-déposée sur une transition :

Sans action préalable, une réceptivité définie ou non (non vide, vide ou ayant la valeur 0 par défaut) sera remplacée par la variable déposée :



Si l'on veut déposer le complément de la variable choisie (ie par exemple NON i2 soit i2/), il faudra au préalable appuyer sur la touche "/" (Slash) du clavier. On verra alors ce caractère ("/") inscrit dans la barre d'état de la fiche grafcet. On pourra alors effectuer le glisser-déposer.

Pour compléter une réceptivité avec une suite d'opérations logiques, il faudra au préalable appuyer sur la touche correspondant à l'opération souhaitée ("+" pour OU ou "." pour ET). Le caractère correspondant ("+" ou ".") sera inscrit dans la barre d'état de la fiche grafcet. La variable glissée-déposée sera alors ajoutée à la réceptivité avec l'opération choisie. Si l'on souhaite ajouter de la même façon le complément d'une variable, il faudra successivement appuyer sur l'opération choisie ("+" ou ".") puis sur la touche "/" avant de glisser-déposer la variable :



Il n'est pas possible de prioriser les opérations de cette façon.

8 Fonctions particulières

Les fonctions qui suivent ne sont pas directement disponibles dans Apimou mais il est possible d'en créer des équivalents.

8.1 Signal d'horloge ou clignotant

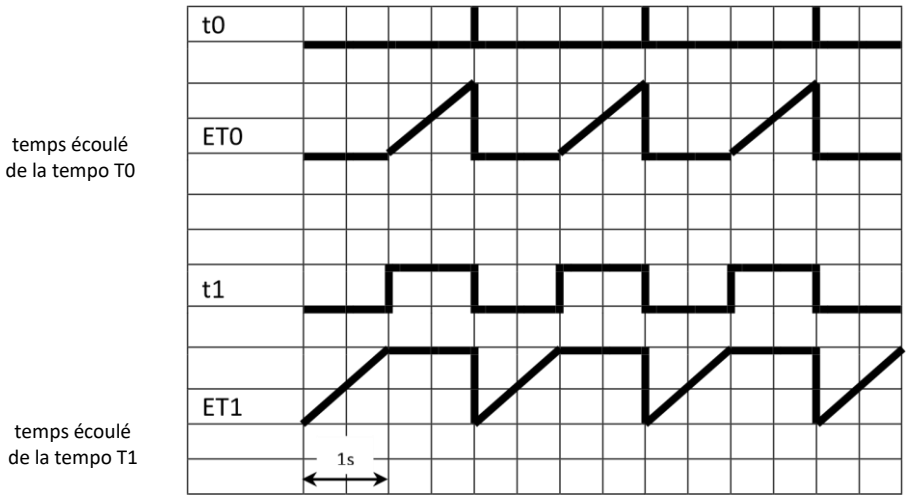
Pour obtenir un bit clignotant on peut utiliser deux temporisations Ton :

Fonctions

Temporisations | Compteurs

Numéro	Type	IN	Préréglage	Base	Q
0	TON	t1	1	s	t0
1	TON	t0/	1	s	t1

Le fonctionnement obéit au chronogramme suivant :

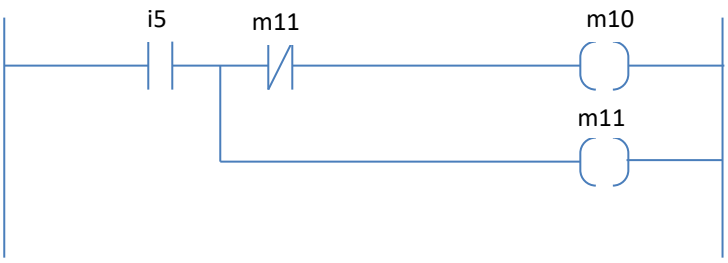


En faisant varier le temps de présélection (PT) des deux temporisations on fait varier la fréquence du signal. Avec un automate "réel" ce type de signal est souvent disponible sur un bit système.

8.2 Front montant ou front descendant

Pour créer un front montant sur une variable, i5 par exemple, il faut se servir de deux bits internes, m10 et m11 par exemple.

Le schéma équivalent en langage à contacts (ladder) serait le suivant :



L'automate scrute le programme de gauche à droite et de haut en bas. Lorsque i5 passe à 1, le bit m11 étant à 0, le bit m10 passe à 1 (1ère ligne). m11 passe immédiatement après (dans le même cycle automate) à 1 puisque i5 est à 1. Au cycle (automate) suivant, i5 est toujours à 1, m11 à 1 et m10 retombe à 0. On obtient ainsi m10 = i5↑, et on peut donc se servir de m10 comme front montant pour i5.

Apimou ne se programme pas en ladder mais en équations, ce qui donne :

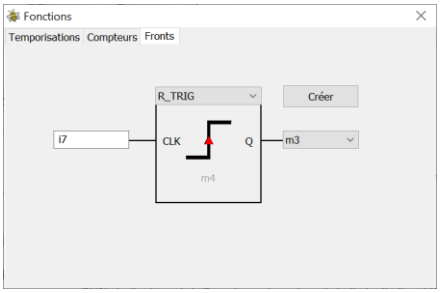
(i5↑ =) m10 = i5.m11/ et m11 = i5

Pour un front descendant, on aurait :

(i5↓ =) m10 = i5/.m11/et m11 = i5/

N.B.: Le bit qui sert de front montant (m10 pour l'exemple) doit impérativement avoir un numéro inférieur à celui qui sert de variable intermédiaire (ici m11), Apimou calculant les bits internes par ordre croissant de leurs numéros.

Pour créer rapidement un bit de front, selon le principe décrit ci-dessus, on peut se servir de l'assistant accessible depuis le menu Affiche->Fonctions, Onglet Fronts :



Il suffit alors de définir la variable (ou l'expression) dont on souhaite détecter le front, le type de front (montant ou descendant) et le bit interne image du front. L'appui sur le bouton Créer génère automatiquement les équations nécessaires.

Les mnémoniques des bits internes acceptent les caractères "↑" (Alt+24) et "↓" (Alt+25). Si, par exemple, on crée un front montant de %I1.7 (mnémonique par défaut "i7") avec le bit interne %M3 (mnémonique par défaut "m3"), on pourra modifier le mnémonique de %M3 en "i7↑".
N.B. : "↑" et "↓" ne sont que des caractères, pas des fonctions. La création d'un front passe nécessairement par la procédure détaillée ci-dessus.

8.3 Bit d'initialisation

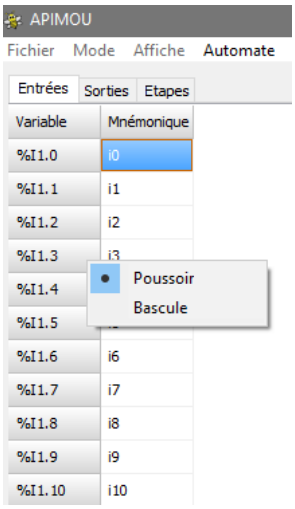
Un bit d'initialisation est un bit qui vaut 1 au premier cycle automate et 0 par la suite (donc un front montant sur le passage en RUN de l'automate). Pour les automates réels il s'agit d'un bit système. Apimou n'en possédant pas, on peut en faire un équivalent de la façon suivante :

Entrées | Sorties | Etapes | Bits internes

Variable	Mnémonique	Equation
%M0	init	1.m1/
%M1	m1	1

L'explication du fonctionnement découle de ce qui est écrit au paragraphe précédent.

9 Fonctionnement en mode simulation



Menu Mode →Simulation.

L'automate étant en RUN (c'est-à-dire qu'il exécute le programme), on peut modifier l'état des entrées en cliquant sur les variables correspondantes dans l'onglet Entrées de la fenêtre principale.
On voit alors les diodes de la carte d'entrées s'allumer.

Pour chaque variable, il est possible de choisir un fonctionnement en poussoir (mode par défaut) ou en bascule en faisant un clic droit sur la variable choisie :

La visualisation de l'état des étapes se fait sur le grafcet (étape(s) active(s) en vert).

L'état des bits internes est visualisé sur la table correspondante (cellule du mnémonique en vert = bit à 1).

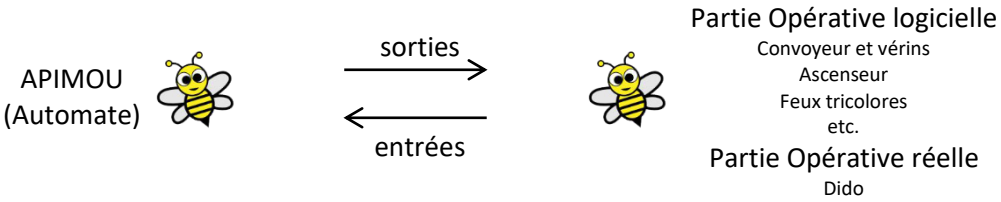
L'état des sorties est directement visualisé par les diodes de la carte de sorties.

10 Fonctionnement en mode Connecté

Menu Mode →Connecté.

Le fonctionnement d'Apimou est le même qu'en mode simulation, lorsque l'automate est en RUN on peut visualiser l'état des entrées/sorties sur les cartes, celui des bits internes sur la table et celui des éventuelles étapes sur le grafcet.

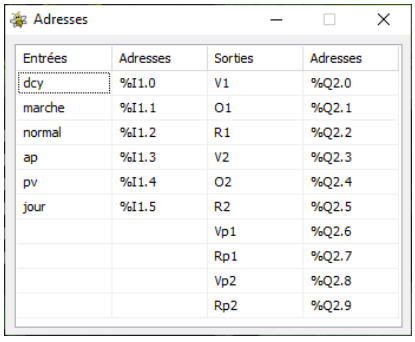
Dans ce mode, Apimou échange avec une Partie Opérative logicielle ou réelle :



Il faut pour cela que la partie opérative soit connectée à Apimou, ce qui se fait par le menu Apimou→Connecter.

Il faut d'autre part connaître le schéma de câblage de l'automate (Apimou) pour cette partie opérative : à quelles entrées sont reliés les capteurs, à quelles sorties sont reliés les actionneurs, donc avoir à disposition un tableau d'entrées/sorties. On l'obtient par le menu Affiche→Adresses.

Exemple pour les feux tricolores :



Entrées	Adresses	Sorties	Adresses
dcy	%I1.0	V1	%Q2.0
marche	%I1.1	O1	%Q2.1
normal	%I1.2	R1	%Q2.2
ap	%I1.3	V2	%Q2.3
pv	%I1.4	O2	%Q2.4
jour	%I1.5	R2	%Q2.5
		Vp1	%Q2.6
		Rp1	%Q2.7
		Vp2	%Q2.8
		Rp2	%Q2.9

Le bouton-poussoir ap (appel piétons), par exemple, est relié à la quatrième entrée de l'automate (Apimou), adresse %I1.3.

Dans Apimou, le mnémonique par défaut de l'entrée %I1.3 est i3. Si on le remplace par ap, on peut alors utiliser ap dans le programme pour faire référence à l'entrée %I1.3.

Les fichiers "config_(nom de la PO).api", une fois chargés dans Apimou, évitent la fastidieuse tâche consistant à modifier tous les mnémoniques pour les faire correspondre aux libellés de la partie opérative.

Pour une partie opérative réelle (huit entrées/huit sorties maximum), on peut utiliser Dido qui permet de contrôler un module externe d'entrées/sorties sur liaison RS-485 en Modbus RTU (il n'y a aucune nécessité de comprendre ce qu'est la liaison RS-485 ou le protocole Modbus RTU).

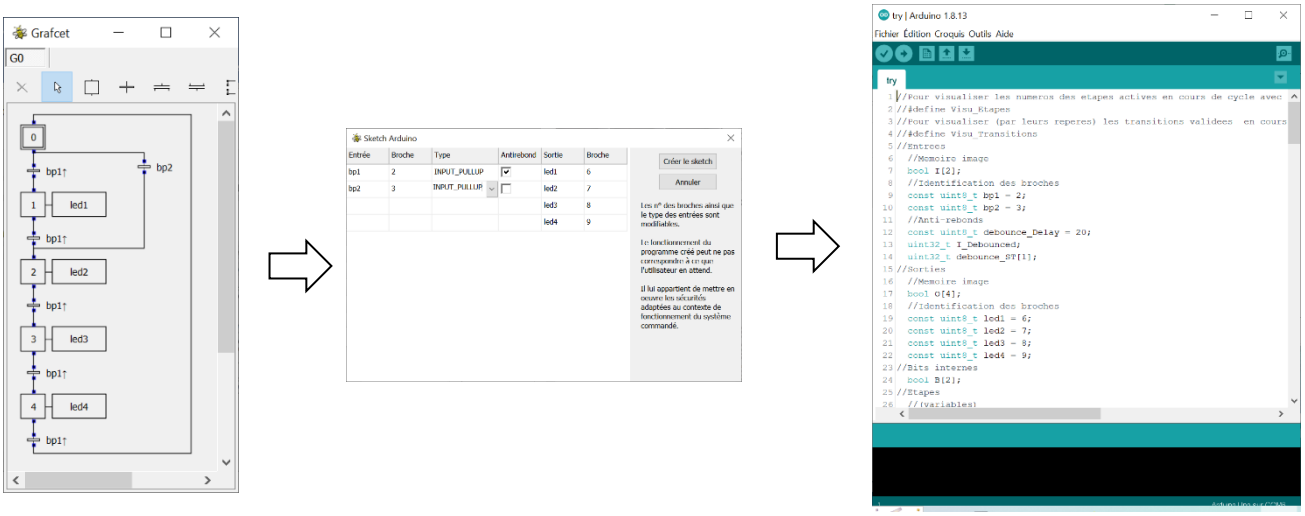
N.B. : Les Parties Opératives fournies avec les versions 1.x.x d'Apimou (Dido inclus) ne sont pas compatibles avec les versions 2.x.x.

11 Génération de code Arduino

N.B. : Cette fonction ne fait que générer un code C++ compatible Arduino. Il n'y a aucune communication entre Apimou et Arduino.

Le menu Fichier -> Créer un sketch Arduino permet de transformer le programme en un sketch Arduino (fichier ".ino"). Il est possible de modifier les broches des entrées/sorties ainsi que de choisir pour chaque entrée le type de fonctionnement (INPUT_PULLUP, PULLUP, PULLDOWN).

On peut aussi affecter à chaque entrée un antirebonds. Cette option peut être utile pour des capteurs mécaniques dont le signal est susceptible de rebondir brièvement sur des changements d'état. Si l'on doit par exemple tester par un front montant des appuis successifs sur un bouton poussoir, ou incrémenter un compteur de la même façon, les rebonds du signal peuvent provoquer un fonctionnement erratique. Un antirebonds permet de remédier à ce problème. L'antirebonds est ici réalisé par une temporisation qui fige la valeur de l'entrée pendant une durée déterminée dès que celle-ci change d'état. Cette durée est par défaut de 20 ms, il est possible de la modifier dans le menu Outils->Options.



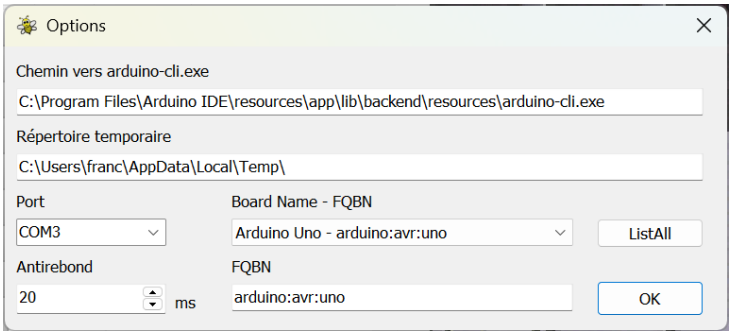
12 Transfert direct d'un programme vers une carte Arduino

Un programme créé avec Apimou peut être directement transféré (téléversé) vers une carte Arduino, sans avoir à passer par l'IDE Arduino. Cette fonctionnalité nécessite que le programme ***arduino-cli.exe*** soit installé sur l'ordinateur. Il est *déjà présent* sur votre ordinateur si vous avez installé l'***IDE Arduino*** en version ***2.x***, dans le cas contraire, avec un IDE Arduino 1.x, il faudra installer ce programme indépendamment.

La seule chose qui est faite ici par Apimou est de récupérer les informations nécessaires pour que arduino-cli puisse exécuter la compilation et le transfert du programme, par la commande :

arduino-cli compile --upload --verbose --no-color (sketch.ino) --port (nom du port) --fqbn (identificateur FQBN)

Avant de pouvoir effectuer un premier transfert direct, il faut renseigner les paramètres nécessaires en cliquant sur le menu Outils->Options (ces paramètres seront par la suite sauvegardés pour une utilisation ultérieure dans un fichier nommé Apimou.ini) :



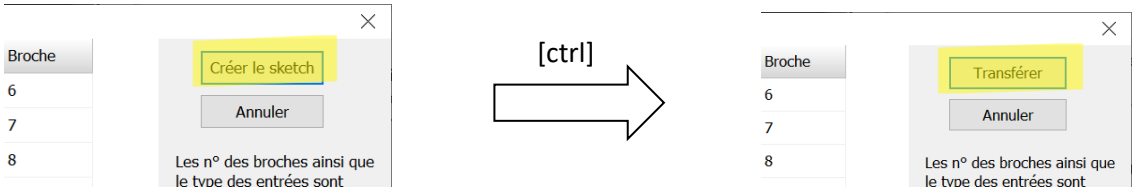
Pour les utilisateurs de l'Arduino IDE 2.x, le chemin vers arduino-cli.exe doit en principe être renseigné automatiquement. Si arduino-cli.exe a été installé indépendamment, ou si Apimou ne le trouve pas, il faudra le définir.

Le répertoire temporaire est par défaut celui de l'utilisateur. Il est possible de le modifier. Apimou va créer dans ce répertoire un sous-répertoire apimou_tmp dans lequel il enregistrera un fichier apimou_tmp.ino. C'est ce fichier, effacé et réécrit à chaque nouveau transfert, qui sera compilé et transféré par arduino-cli.

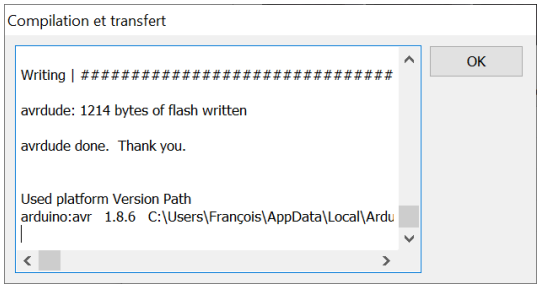
Le port, à sélectionner dans la liste déroulante, doit être celui auquel la carte Arduino est connectée. Si vous avez plusieurs périphériques USB connectés, le port sélectionné par défaut n'est pas forcément le bon. Si la carte n'est pas connectée, le port n'apparaitra pas. Il faudra alors connecter la carte, puis fermer et ouvrir à nouveau la fenêtre des options.

Le FQBN (Fully Qualified Board Name) est celui de la carte utilisée. Si la carte est installée et si le chemin vers arduino-cli est valide, il est possible de trouver le FQBN en cliquant sur le bouton ListAll puis en choisissant la carte dans la liste déroulante.

La procédure permettant de transférer directement le programme vers la carte passe ensuite par le menu Fichier -> Créer un sketch Arduino, comme expliqué au paragraphe précédent. Sur la fiche qui apparait, en appuyant de façon continue sur la touche [ctrl] du clavier, le bouton "Créer le sketch" est renommé en "Transférer" :



En cliquant sur ce bouton, on lance la procédure de compilation et de transfert. Une fenêtre affiche les résultats de l'opération, fournis par arduino-cli :



Si la procédure échoue, vous devrez peut-être :

- Détecter la carte connectée avec la commande *arduino-cli board list*
- Lister les cartes installées et obtenir leur FQBN avec la commande *arduino-cli board listall*
- Lister les familles de cartes installées avec la commande *arduino-cli core list*
- Lister les familles de cartes qu'il est possible d'installer avec la commande *arduino-cli core search*
- Installer une famille de cartes avec la commande *arduino-cli core install (…)*
- Etc. La documentation officielle se trouve sur github : https://github.com/arduino/arduino-cli

Si nécessaire, pour lancer une commande, vous pouvez par exemple :

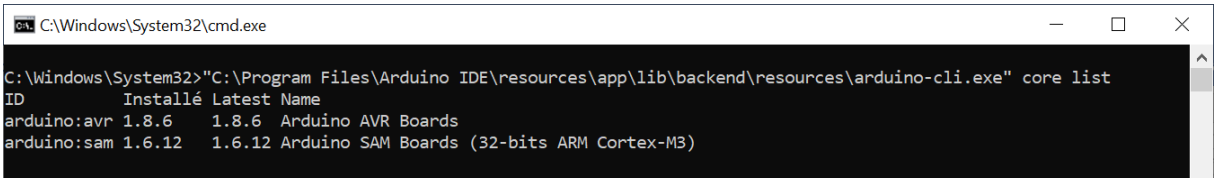
- Copier/coller le chemin vers arduino-cli dans un éditeur de texte (le bloc-notes par exemple)
  C:\Program Files\Arduino IDE\resources\(…)\arduino-cli.exe

- L'encadrer avec des guillemets (pour éviter une erreur d'interprétation due aux espaces éventuels)
  "C:\Program Files\Arduino IDE\resources\(…)\arduino-cli.exe"

- Ajouter la commande voulue
  "C:\Program Files\Arduino IDE\resources\(…)\arduino-cli.exe" core list

- Ouvrir l'interpréteur de commande Windows cmd.exe (taper cmd.exe dans le champ de recherche du menu Démarrer)

- Copier la ligne à partir de l'éditeur de texte et la coller dans cmd.exe :



13 Génération de code ESP32

13.1 Généralités

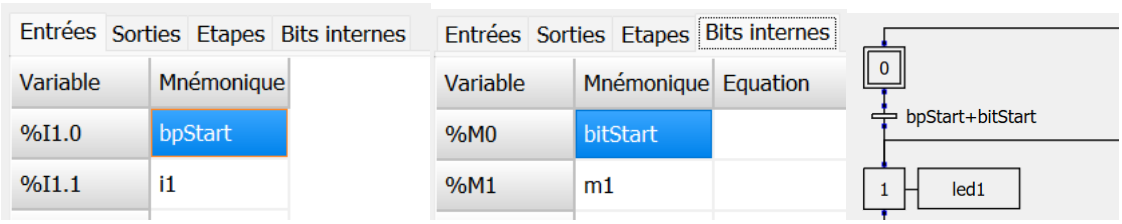
Avec un ESP32, il est possible de générer un code C++ (fichier .ino) et un ensemble de fichiers (.html, .png) permettant de visualiser avec un navigateur web quelconque l'évolution du ou des grafcet(s) constituant le programme. La communication se fait en Wifi, on peut donc utiliser n'importe quel terminal (PC, smartphone, tablette) relié au même réseau que l'ESP32.

Outre cette visualisation, on peut aussi éventuellement accéder, par une page web dédiée, à des réglages :

- ✓ modifications de bits, nommés *bits IHM* (s'ils sont utilisés),
- ✓ modifications de valeurs de présélection PT des temporisateurs (s'ils sont utilisés),
- ✓ modifications de valeurs de présélection PV des compteurs (s'ils sont utilisés).

Les *bits IHM* sont, dans Apimou, des bits internes qui *ne sont pas définis par une équation* et qui *interviennent dans une équation quelconque* (sortie ou temporisation ou comptage ou forçage ou transition).

Exemple :



bpStart est une entrée (un bouton poussoir par exemple), reliée physiquement à une broche de l'ESP32. bitStart est un bit interne qui n'a pas d'équation. bitStart intervient dans la réceptivité de la transition étape 0 -> étape 1 (équation "bpStart + bitStart", ce qui signifie que le cycle démarre si on appuie sur le bouton poussoir bpStart ou si bitStart passe à 1). Lors de la traduction du programme en sketch, bitStart sera considéré comme un bit IHM et il sera possible de le forcer à partir de la page de réglages.

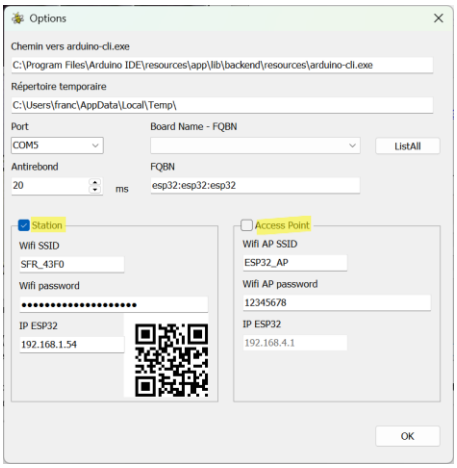
Pour qu'un réglage soit *validé*, il faut cliquer sur le bouton de validation correspondant. Ce réglage reste valable tant que l'ESP32 ne redémarre pas. Pour que l'ensemble des réglages validés soit *sauvegardé* (y compris les bits IHM), il faut cliquer sur le bouton de sauvegarde. La sauvegarde est réalisée en mémoire Flash (NVS) et chargée à chaque nouveau démarrage.

13.2 Prérequis

- IDE Arduino 2.x
- Carte ESP32 installée
- Plugin LittleFS Uploader (voir https://randomnerdtutorials.com/arduino-ide-2-install-esp32-littlefs/) installé
- Bibliothèque Async TCP par ESP32ASync installée
- Bibliothèque ESP Async WebServer par ESP32Async installée

13.3 Modes Station et Access Point

Avant de générer le sketch, il faut choisir dans le menu *Outils ->Options* le mode Wifi qui sera utilisé : **Station** et/ou **AccessPoint**.



En mode **Station** l'esp32 se connecte à un réseau Wifi existant. Il faut pour cela connaitre le nom (SSID) du réseau ainsi que le mot de passe (password) qui en autorise l'accès. Le routeur de ce réseau (la "box") attribue automatiquement une adresse IP à l'ESP32. Sous réserve d'un signal suffisant, le mode Station est le plus performant.

En mode **Access Point** l'esp32 génère son propre réseau Wifi et agit comme un routeur. On définit le nom (SSID) du réseau ainsi que le mot de passe (password). L'adresse IP de l'esp32 est imposée, égale à 192.168.4.1. Le mode Access Point est limité à 4 clients, en pratique 1 ou 2 pour un rafraîchissement fluide et sous réserve que la fréquence ne dépasse pas 100 à 200 ms.

Il est possible de choisir simultanément les deux modes, **Station** + **Access Point**. Le mode Access Point pourra dans ce cas être une situation de secours, au cas où le réseau Wifi existant tombe ou soit hors de portée. Ce choix reste moins performant que le mode Station uniquement.



Pour qu'un appareil (PC, smartphone, tablette) accède au serveur de l'ESP32 en mode **Access Point**, il devra au préalable se connecter au réseau Wi-Fi créé par l'ESP32. Si cet appareil était connecté au serveur (box) du réseau local existant en Wi-Fi, il perdra cette connexion et donc, entre autres, l'accès à internet.

Les **QR codes** qui apparaissent dans la fiche Options renvoient directement aux adresses IP de l'ESP32 (adresse sur le réseau local et/ou adresse en Access Point), sous réserve que l'appareil soit connecté au réseau Wi-Fi correspondant.

13.4 Procédure

Pour le **mode Station**, il est *préférable* de définir *au préalable* le nom (Wifi SSID) du réseau Wifi ainsi que le mot de passe de ce même réseau (Wifi password). Ces deux paramètres peuvent aussi être définis ultérieurement.

Pour le mode **Access Point**, il faut définir *au préalable* le nom (Wifi AP SSID) du réseau Wifi AP ainsi que le mot de passe de ce même réseau (Wifi AP password). Des valeurs sont proposées par défaut. Le mot de passe peut être vide (le réseau sera dans ce cas ouvert).

Après avoir sélectionné le menu ***Fichier -> Créer un sketch -> ESP32***, la procédure est identique à celle décrite dans le paragraphe Génération de code Arduino. Contrairement à une carte Arduino, on peut ici configurer une entrée en INPUT_PULLDOWN.

Après avoir cliqué sur ***Créer le sketch***, Apimou va créer un fichier .ino (le sketch) et un sous-dossier \data contenant les fichiers .html et .png nécessaires. Il est important que le fichier .ino soit créé dans un dossier du même nom (pour exemple, ci-dessous, projet.ino se trouve dans un dossier nommé projet).

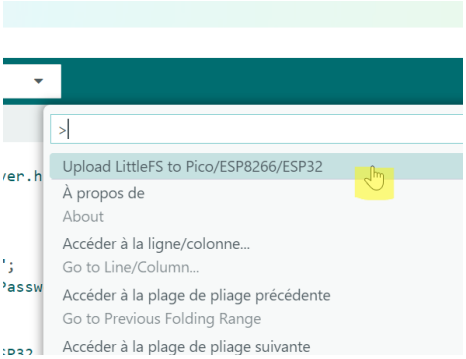
- projet\
  - projet.ino
  - (projet.api)
  - data\
    - home.html
    - vue.html
    - grafcet0.html
    - grafcet0.png
    - grafcet1.html
    - grafcet1.png
    - (…)

Il faut ensuite ouvrir le fichier *.ino* créé dans l'**IDE Arduino**.

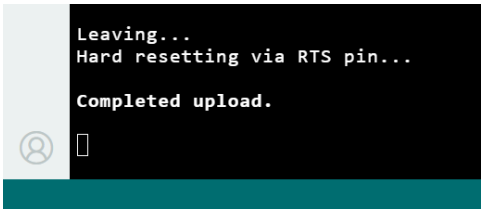
Si le SSID et le mot de passe du réseau WiFi en mode **Station** n'ont pas été préalablement renseignés dans le menu *Options* d'Apimou, il faudra le faire dans l'IDE Arduino (en gardant les guillemets).



Après s'être assuré que la carte sélectionnée est la bonne, que le port COM est correct et que **le moniteur série est fermé**, il faut ensuite appuyer simultanément sur les touches **[Ctrl] + [Shift (maj)] + [P]** afin d'ouvrir le menu des commandes et choisir dans celui-ci *Upload LittleFS to Pico/ESP8266/ESP32*.



Le plugin LittleFS Uploader va alors charger dans l'ESP32 tous les fichiers du dossier \data du projet. Après quelques secondes, le message *Completed upload* doit s'afficher.

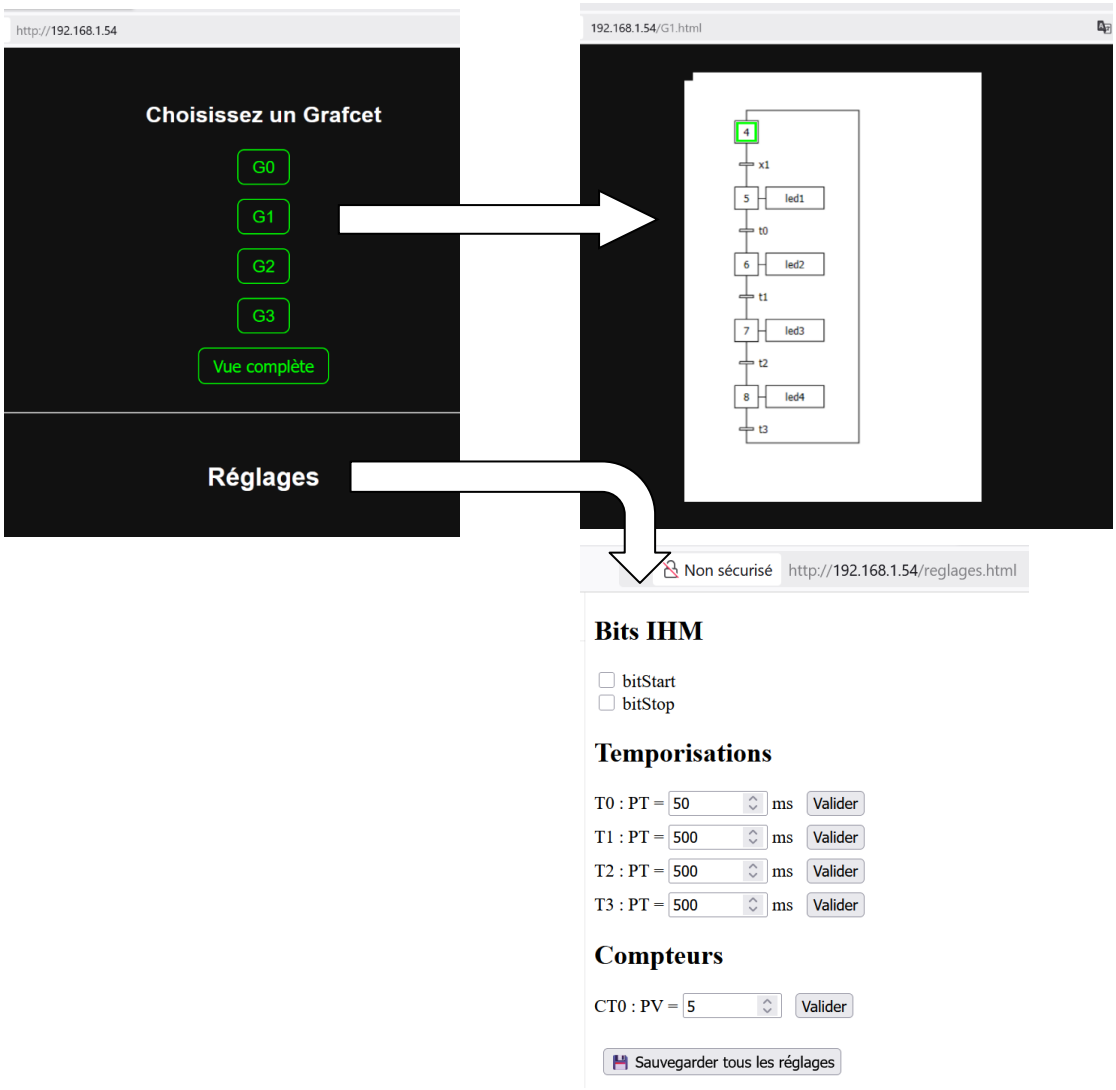


Il reste ensuite à **ouvrir le moniteur série** à 115200 bauds puis **téléverser le sketch** (*Croquis -> Téléverser* ou [Ctrl]+[U]).

A la fin du téléversement, on doit obtenir dans le moniteur série un message de confirmation de la connexion au réseau WiFi et l'adresse IP de l'ESP32 pour le mode Station et/ ou pour le mode Access Point.



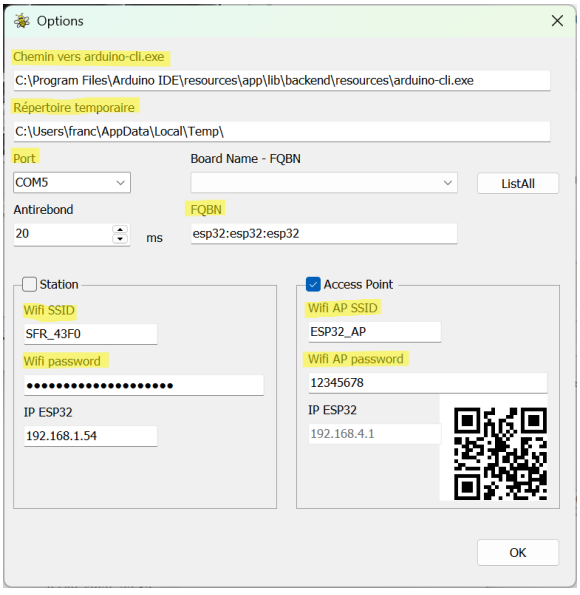
Il suffit ensuite de se connecter au réseau voulu (réseau local existant en mode Station ou réseau créé par l'ESP32 en mode Access Point), de copier l'une de ces adresses et de la coller dans la barre d'adresse d'un navigateur web (ou de scanner l'un ou l'autre des QR codes) pour accéder à la page d'accueil de l'interface de visualisation. On peut à partir de là choisir le grafcet voulu (il peut n'y en avoir qu'un), et le voir s'animer en temps réel. On peut également ouvrir la page Réglages et modifier les bits IHM ou les présélections des temporisations et des compteurs.



14 Transfert direct d'un programme vers un ESP32

Avant de mettre en œuvre cette procédure, il est nécessaire d'avoir au préalable généré et transféré au moins une fois un programme avec l'IDE Arduino comme expliqué au paragraphe précédent, pour s'assurer que tout est en place et fonctionne.

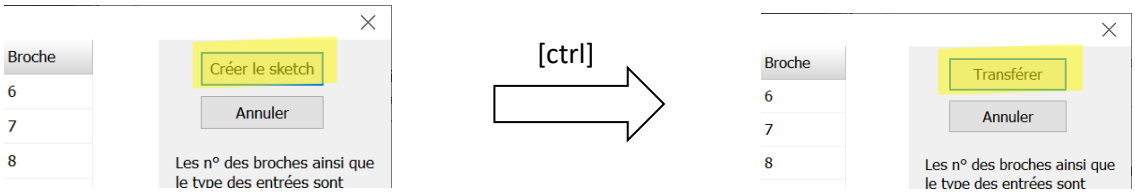
Il faut ensuite renseigner dans la fenêtre des options (*Outils->Options*) tous les paramètres indispensables : **chemin vers arduino-cli**, **répertoire temporaire**, **port**, **FQBN**, **SSID** et **mot de passe** du Wifi pour le(s) mode(s) choisi(s) (Station et/ou Access Point).



Les explications concernant les quatre premiers paramètres sont détaillées dans le paragraphe Transfert direct d'un programme vers une carte Arduino.

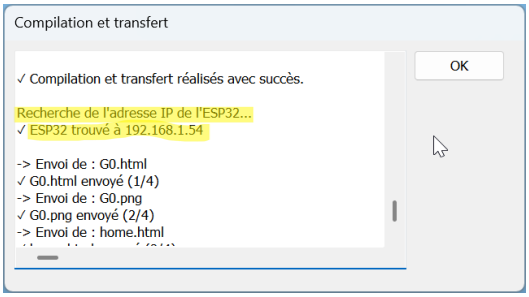
L'adresse IP de l'ESP32 en mode **Station** devrait être automatiquement trouvée, le champ "IP ESP32" peut être laissé vide. Si un premier transfert échoue parce que cette adresse n'est pas trouvée, il faudra peut-être le renseigner.

Après avoir sélectionné le menu ***Fichier -> Créer un sketch -> ESP32*** et configuré les entrées/sorties, il suffit d'appuyer de façon continue sur la touche [ctrl] du clavier, le bouton "Créer le sketch" est renommé en "Transférer" :

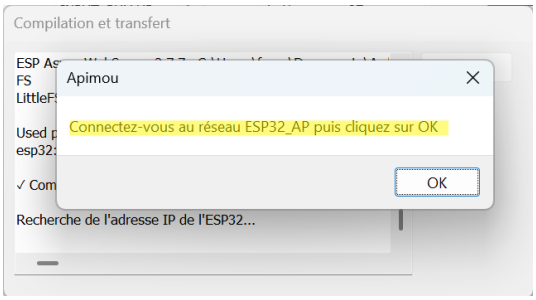


En cliquant sur ce bouton, on lance la procédure de compilation et de transfert. Dans un premier temps, arduino-cli compile puis transfère le sketch. Dans un deuxième temps, Apimou envoie par WiFi les fichiers .png et .html nécessaires (ici le plugin LittleFS Uploader n'est pas utilisé).

Lorsque le mode **Station** ou le mode **Station + Access Point** sont sélectionnés, les fichiers html et png sont transférés en mode Station, il faut donc que le PC soit connecté au réseau local existant. Après la compilation et le transfert du sketch, l'adresse IP de l'ESP 32 est recherchée puis les fichiers html et png sont transférés.



Lorsque le mode **Access Point** seul est sélectionné, la procédure s'interrompt après le transfert du sketch. Un message indique alors qu'il faut se connecter au réseau créé par l'esp32 (par défaut ESP32_AP avec comme mot de passe 12345678), **puis** cliquer sur OK pour que les fichiers html et png puissent être transférés.



A la fin de la procédure, l'adresse IP de l'ESP32 est copiée dans le presse-papier. Il suffit de la coller ([ctrl]+[V]) dans la barre d'adresse du navigateur, ou de revenir au menu Options pour scanner l'un ou l'autre des QR Codes.

15 Gestion des fichiers

Le menu Fichier →Nouveau efface tout, sans avertissement préalable (mnémoniques, équations, grafcets).

Pour enregistrer un fichier (un programme), on sélectionne le menu Fichier → Enregistrer sous... L'extension des fichiers Apimou est ".api" (extension ajoutée par défaut).

L'ouverture d'un fichier se fait par le menu Fichier → Ouvrir. Il est également possible, si Apimou n'est pas en cours d'exécution, d'ouvrir un fichier (et de fait Apimou) en effectuant un double clic à partir du dossier dans lequel il se trouve. Le système d'exploitation demandera alors l'exécutable à associer à l'extension ".api".