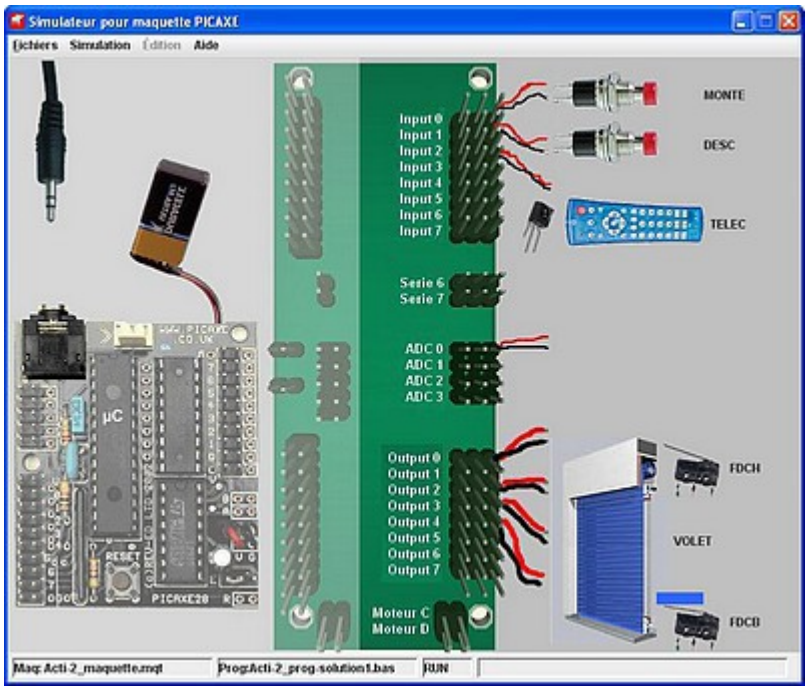
PROJET
« Simulateur maquette PICAXE »

Manuel prof



Auteur : <jean-louis.morceau@ac-toulouse.fr>
Version: 2,0 (mars 2020)

Sommaire

Les nouveautés de cette version apparaissent surlignée en jaune.

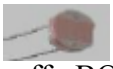
Présentation du projet

Le projet a pour but de permettre de tester les programmes PICAXE, réalisés en technologie par les élèves, de la manière la plus réaliste possible, même en l'absence de maquette physique disponible. Le simulateur de « Programming Editor » ne simulant que le circuit PICAXE et celui de « Logicator (Programming studio) » étant trop limité.

Le principe est basé sur une manipulation d'images choisies par le concepteur de la maquette. Les formats possibles sont PNG (pour bénéficier de la transparence) et JPG. Si une image existe aux 2 formats, le format PNG est choisi en priorité.

Une image figé (la maquette) est placé en fond. Les images des capteurs et des actionneurs sont positionnées au dessus. Elles changent en fonction de la situation (instructions du programme ou actions de l'utilisateur).

La banque d'image est entièrement personnalisable en respectant les conventions suivantes :

1 image pour les composants à 1 position [LDR] 

2 images pour les composants à 2 états [BOUTON_off ; BOUTON_on]  

x images pour les composants multi états [MOTEUR_0 ; MOTEUR_1 ; MOTEUR_2 ; MOTEUR_3

Des composants et leurs instructions spécifiques pourront être ajouté au fur et à mesure des besoins.

Limitations et problèmes connus

Toutes les instructions PICAXE BASIC ne sont pas traitées par l'interpréteur (voir annexe 1).

Seul les PICAXE 08M, 20M, 28X1 et 20M2 sont définis (entrées numériques, entrées analogiques, sorties et registres). Il n'est pas tenu compte de la spécificité de certaines entrées ou sortie.

Les registres octet (b) et mot (w) ne sont pas différentiés.

Le contrôle de la syntaxe du programme est limitée. Par exemple, ne pas débuter le nom d'un symbole par « b » ou « pin » car cela poserai problème.

Il n'existe pas d'aide en ligne.

Les composants ALIM et PROG doivent être présent dans la maquette.

Gestion très imparfaite des sons.

Ces limitations évolueront avec les versions du programme.

Installation

S'assurer qu'au moins Java 1-7 est installé.

Décompresser le dossier SimulPicaxe et son contenu à l'emplacement voulu.

Créer le raccourci vers « Run.bat » sur le bureau ou dans le menu « Démarrer ».

Organisation des fichiers :

- \ressources : fichiers images des maquettes et des composants
- \maquettes : mes premières maquettes qui me servent toujours de test.
- Run.bas : lanceur du simulateur
- SimulPicaxe.jar : code java du simulateur
- Simul.png : icône du simulateur
- Manuel.pdf : ce document

Principe d'utilisation

Utiliser le lanceur ou son raccourci.

Ne pas fermer la fenêtre DOS en cours d'exécution. Pour quitter le simulateur à la fin de son utilisation, utiliser la commande « Fichier / Quitter » ou le bouton « Fermer application ». La fenêtre DOS se fermera automatiquement.

Lancer le programme, ouvrir un fichier maquette, alimenter la simulation (ou clic pile), programmer la simulation (ou clic câble de programmation). La simulation se lance automatiquement.

Un clic sur un composant permet d'en modifier l'état (capteurs ou actionneurs réglable). Voir les effets sur les différents composants plus loin dans ce document.

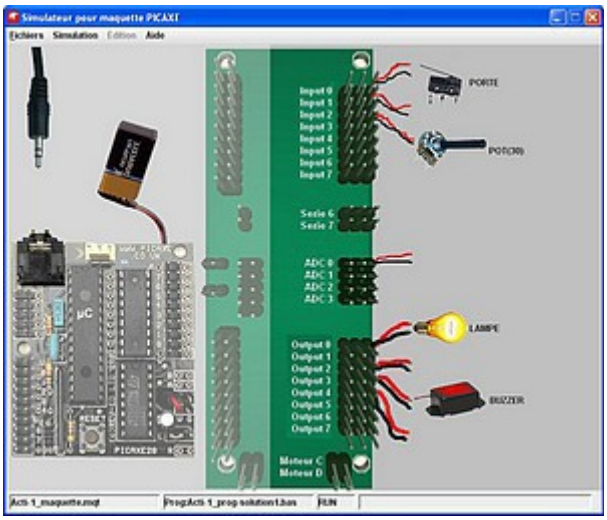
Le programme peut être arrêtée et recommencée au début en débranchant et rebranchant l'alimentation.

Les maquettes sont généralement créées au préalable par l'enseignant. L'élève les utilise pour tester ses programmes. Il est possible d'ajouter ou de modifier des composants à une maquette existante (maquette hors tension, « Édition / Modifier » puis « Édition / Ajouter» ou double clic sur le composant). Il faut alors faire évoluer le programme en conséquence.

Les menus

Fichiers

- **Créer**: Crée une nouvelle maquette vide et passe en mode édition.
- **Ouvrir** : Ouvre un fichier maquette et passe en mode simulation.
- **Enregistrer** : Enregistre la maquette courante.
- **Enregistrer sous** : Enregistre la maquette.
- **Fermer** : Ferme la maquette courante.
- **Quitter** : Quitte le simulateur



Simulation

- **Alimenter** : Met sous/hors tension la maquette virtuelle *(Idem clic sur Alim)*
- **Programmer** : Charge un programme dans la maquette *(Idem clic sur Prog)*

Édition

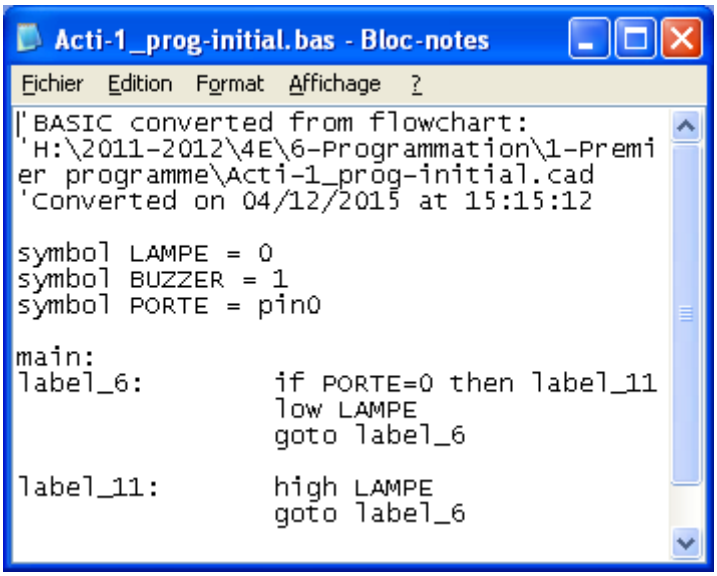
- **Modifier** : Passe_en/Sort_du mode modification.
- **Ajouter** : Ajoute un composant à la maquette courante

Aide

- **Info** : Informations sur le programme.

Le fichier programme (Prog.bas)

C'est le programme qui va être exécuté sur la maquette. Il s'agit d'un fichier texte.



Création

Il peut être généré avec « Programing Editor », « Logicator », « Programing Editor 6 » ou « PicaxeBlockly App ».

Si la programmation est faite par organigramme :

- Cliquer sur «PICAXE», puis sur « Convertir le diagramme en Basic ... ».
- Sauvegarder le fichier Basic sans le modifier.

Avec « PE6 » ou « PicaxeBlocklyApp », basculer en mode éditeur Basic (code) puis le sauvegarder directement sans modification.

Description

La description du PICAXE MANUAL 2 (Basic commands) est valable au remarques ci-dessous.

Les directives ne sont pas gérées.

Les caractères « ; » (début de commentaire), « : » (séparation de ligne), « _ » (poursuite de ligne), « {} » (délimitation de bloc de code) ne sont pas autorisés.

Un commentaires débute par une apostrophe « ' » et est ignoré. Il en est de même des lignes vides.

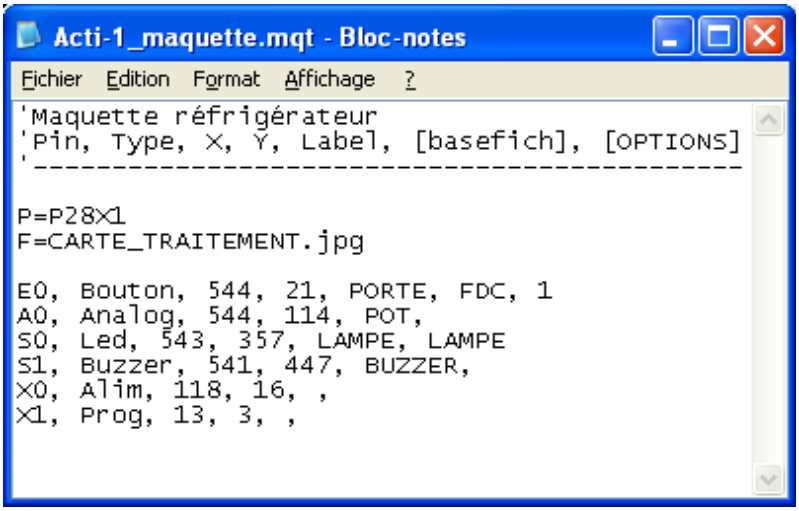
Un symbole est défini par « Symbol » en début de ligne. Le nom du symbole est suivit de « = ». Un registre est de la forme (bx ou wx), une entrée (pinx ou pinX.x) et une sortie (x ou X.x). Rien d'autre ne doit exister sur la ligne.

Les points de branchement débutent toujours la ligne et sont suivit par « : ». On peut trouver une instruction sur la même ligne.

Tout le reste est considéré comme une instruction [mnémonique paramètres] suivant la syntaxe du BASIC PICAXE.

Le fichier maquette (Maquette.mqt)

C'est la description de la maquette à utiliser. Il s'agit d'un fichier texte.



Création

Le mode « Édition » permet de le générer graphiquement et de l'enregistrer.

Description

Si une ligne débute par une apostrophe ['] (commentaire) ou est vide, elle est ignorée.

On doit trouver dans l'ordre :

- Les commentaires sur la maquette
- P= CODE_PICAXE (ex. 20M, 28X1 …) pour définir les caractéristiques du contrôleur utilisé.
- F= FICHIER_MAQUETTE.jpg pour définir l'image de fond de la maquette.
- Le descriptif des composants utilisées.

Voir le mode Édition pour plus de détail

Les broches commencent par :

- X sont spécifiques au simulateur (alimentation et programmation).
- E, S ou A sont les Entrée, Sortie et Analogique des anciennes versions de PICAXE.
- C. B. A. D. correspondent aux appellation des nouvelles versions de PICAXE (P20M2).

Le mode « Édition »

Le passage en mode « Édition » se fait lors de la création d'une nouvelle maquette ou si une maquette est chargée et non alimenté en utilisant la commande « Édition/Modifier ».

On quitte le mode en utilisant la commande « Édition/Modifier ». Il est demandé d'enregistrer la maquette si elle a été modifiée.

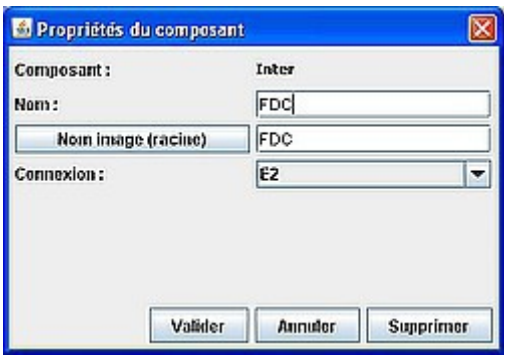
Maquette

Lors de la création d'une nouvelle maquette ou d'un double clic dessus en mode « Édition », une fenêtre permet de définir :

- Le modèle de PICAXE utilisé (parmi la liste des modèles disponibles).
- Le nom du fichier image utilisé pour représenter la maquette (nom+extension).
  L'image doit au préalable avoir été placée dans le dossier « ressources ».
  La taille de l'image est limité à 800 x 600.
  L'image peut être sélectionnée grâce au bouton «Nom image».
- Un commentaire (sans les apostrophe en début de ligne).

Cela correspond aux premières lignes du fichier maquette (voir ci dessus).

Composants



On ajoute un type de composant avec la commande « Édition/Ajouter » en le choisissant dans la liste. Le composant est ajouté avec son image par défaut.

On le positionne en le déplaçant avec la souris à l'endroit désiré.

On le personnalise ou le modifie en double cliquant dessus et en définissant les paramètres voulus.

Il peut être supprimé de la maquette en double cliquant dessus et en utilisant le bouton « Supprimer ».

La description du composant est enregistré sur une ligne du fichier maquette. Les paramètres sont séparés par des virgules. Les paramètres suivants existent pour tous les composants.

- Composant : Type du composant (fixé lors du choix dans la liste des possibilités).
- Connexion : Type X(spécial pour Alim X0 et Prog X1), ou broche fonction du Picaxe utilisé. La liste de choix ne propose que les connexions disponibles.
- Position X, Position Y : Position dans la fenêtre de simulation du point supérieur gauche de l'image du composant.
- Nom (OPTIONNEL) : Texte qui sera affiché à la suite du composant.
- Nom image (OPTIONNEL) : Nom de base des fichiers images du composant. Chaque image associé au composant doit débuter par ce nom. Le nombre des fichiers images et les compléments nécessaire au nom de base sont indiqués à chaque composant ci après. Il existe un nom par défaut pour chaque composant (entre crochet ci-après). Les fichiers doivent être placés dans le dossier « ressources ». Une des image peut être sélectionnée grâce au bouton «Nom image», seule la base du nom sera conservée. Les images peuvent être au format JPG ou PNG (qui gère la transparence).

Les autres paramètres possibles sont décrit avec les composants concernés.

Les composants Alim et Prog existent en 3 versions dans le dossier ressource :

- ---0.jpg version discrète (lettre) pour utilisation à partir du menu
- ---.png positionnement vertical
- ---2.png positionnement horizontal

Alim [ALIM]

Obligatoire. Dispositif d'alimentation de la maquette. Commande « Simulation / Alimenter ».
Toujours connecté en X0 et sans affichage du nom.

Il possède un état « BASE_off » et un état « BASE_on ».
Le changement d'état est réalisé par un clic sur l'image.

Prog [PROG]

Obligatoire. Dispositif de programmation de la maquette. Commande « Simulation/Programmer».
Toujours connecté en X1 et sans affichage du nom.

Il possède un état « BASE_off » et un état « BASE_on ».
Le changement d'état est réalisé par un clic sur l'image.

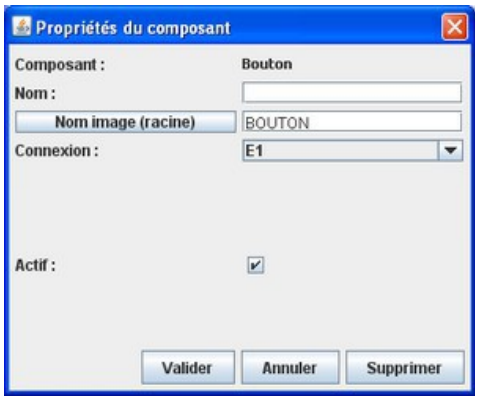
Inter [INTER]

Capteur 2 états stable, au repos par défaut. Le changement d'état se fait par un clic sur l'image.

Il possède un état « BASE_off » et un état « BASE_on ».
Les valeurs de retour sont 0 et 1 sur une entrée numérique (0 et 255 sur une entrée analogique).

Autre paramètre : Le bouton est par défaut actif.

Bouton [BOUTON]



Capteur 1 seul état stable, au repos par défaut. Le changement d'état se produit lors de l'appui ou du relâchement du bouton gauche de la souris sur l'image. Le bouton droit permet de le verrouiller sur l'état « instable » (apui maintenu).

Il possède un état « BASE_off » et un état « BASE_on ».
Les valeurs de retour sont 0 et 1 sur une entrée numérique (0 et 255 sur une entrée analogique).

Autre paramètre : Le bouton est par défaut actif.

Analog [LDR]

Il simule un capteur analogique.
Le changement de valeur se fait en cliquant avec le bouton droit (incrément) et gauche (décrément) de la souris. Le pas est de 10 si la touche majuscule est maintenue appuyée lors du clic.

Il ne possède qu'un seul état (sa valeur est inscrite à la suite du nom).
Les valeurs de retours vont de 0 à 255 (si entrée numérique 0 ou 1 – seuil fixé à 100).

Num [TMP]

Il simule un capteur analogique branché sur une entrée série.
Le changement de valeur se fait en cliquant avec le bouton droit (incrément) et gauche (décrément) de la souris. Le pas est de 10 si la touche majuscule est maintenue appuyée lors du clic.

Il ne possède qu'un seul état (sa valeur est inscrite à la suite du nom).
Les valeurs de retours vont de 0 à 255 (invalide sur une entrée analogique).

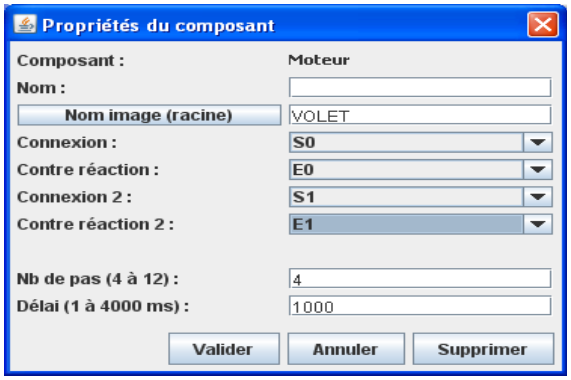
Led [LED_rouge]

Il simule un simple actionneur 2 états.
Il possède un état « BASE_off » et un état « BASE_on ».

Buzzer [BUZZER]

Il simule un actionneur générant un son.
Il possède un état « BASE_off » et un état « BASE_on ».
Les versions futures permettront de gérer la durée du son et la commande SOUND.

Moteur [MOTEUR]



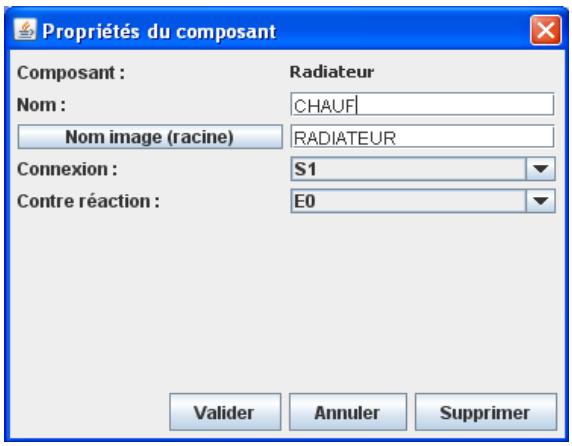
Il simule un moteur ou un vérin (déplacement continu).

Il possède x états « BASE _0 », « BASE _1 », « BASE _2 » ... « BASE _x » qui se succèdent grâce à un timer. Par défaut il n'a qu'un sens de rotation et n'utilise qu'une seule sortie. Grâce aux options, on peut lui attribuer une 2ème sortie pour inverser le sens de rotation et prévoir une contre réaction (ex. contacts de fin de course).

Autres paramètres :

, fdc1	A chaque passage à l'état nbpas-1, il active l'entrée numérique indiqué. (Vide sinon) Dans la liste de choix, seule les entrées déjà utilisées sont disponibles.
, s2	Inversion du sens de rotation en utilisant une 2ème connexion. (Vide sinon)
, fdc2	A chaque passage à l'état 0, il active l'entrée numérique indiqué. (Vide sinon) Dans la liste de choix, seule les entrées déjà utilisées sont disponibles.
, nbpas	Nombre de pas possibles (images) pour le moteur (4 par défaut).
, delais	Délais en ms entre chaque image lors du fonctionnement (200ms par défaut).

Radiateur [RADIATEUR]



Il simule un changement d'état régulier en faisant varier une valeur analogique invisible) par rapport à une référence. La valeur de référence est seule affichée à la suite du nom. Elle peut être changée par un clic droit ou gauche de la souris. La touche majuscule multiplie la pas par 10.

Il possède 2 états « BASE_on » s'il est activé (la valeur analogique augmente régulièrement) et « BASE_off » sinon (la valeur analogique diminue ou augmente régulièrement pour atteindre la valeur de référence.

Autre paramètre :

Entrée analogique destinée à recevoir (et afficher) la valeur courante de l'actionneur. Dans la liste de choix, seule les entrées utilisées sont indiquées (vide par défaut).

Afficheur [AFF]

Il simule un afficheur 2 x 16 caractères compatible PICAXE.

La vitesse de transmission et la parité ne sont pas gérés. Les commandes d'affichages sont celles de la documentation PICAXE.

Il ne possède qu'un état.

Telec [Telec]

Il simule la télécommande PICAXE bleu et son récepteur infra-rouge.

Si un code est envoyé, il s’affiche entre parenthèse jusqu’à ce qu’il soit lu.

Les touches utiles sont : les chiffres du pavé numérique ; les touches de direction, les touches page up et down, la touche [Echape] (POWER). Les codes transmis sont ceux de la documentation officielle de la télécommande.

Il ne possède qu'un état.

Annexe 1 : Liste des instructions interprétées

Basic Picaxe	Opérateur	Organig. ramme	Bloc
Sorties :			
high	variable	Oui	Oui
low	variable	Oui	Oui
toggle	variable	Oui	Oui
let pins	expression	Oui	Oui
sound		Non	Non
serout	variable, cmd, txt (cmd n'est pas utilisé)	Oui	Oui
servo		Non	Non
play		Non	Non
tune		Non	Non
infraout		Non	Non
pulsout		Non	Non
pulsin		Non	Non
irin	[delay, label], variable, registre	Basic	???
infrain		Oui	Oui
Delais :			
pause	variable/constante	Oui	Oui
wait	variable/constante	Oui	Oui
sleep	variable/constante	Oui	Oui
nap	variable/constante	Oui	Oui
Entréess :			
if … then	variable d'entrée, variable/constante, label	Oui	Oui
readadc	variable, registre	Oui	Oui
readtemp	variable, registre	Oui	Oui
Déroulement :			
goto	label	Basic	Oui
gosub/call	label	Oui	Oui
do [until/while]	variable d'entrée, variable/constante, label	Basic	Oui
loop [until/while]	variable d'entrée, variable/constante, label	Basic	Oui
return	--	Oui	Oui
stop	--	Oui	Oui
setint		Non	Non
Autres :			
let	registre, expression	Oui	Oui
debug		Non	Non
random		Non	Non

read		Non	Non
write		Non	Non
peek		Non	Non
poke		Non	Non

Une variable est, soit le contenu d'un registre, soit le numéro d'une sortie ou d'un canal d'entrée (sans préfixe). Elle peut être remplacée par un symbole.

Une variable/constante est, soit une valeur numérique entière comprise inclusivement entre 0 et 255 qui peut être préfixée par '$' ou '0x' si elle est indiquée en hexadécimal ou '%' si c'est en binaire, soit la valeur d'un registre.

Une variable d'entrée peut faire référence à une entrée (préfixe 'pin') un registre (préfixe 'b'). Elle peut être remplacée par un symbole.

Une expression est une variable/constante ou un calcul à partir de plusieurs.

Un symbole permet de donner un nom explicite à une entrée [pin], d'une sortie [] ou d'un registre [b] (voir table des symboles). Il doit être déclaré en début de fichier programme (ex. : symbol nom = b12).

Un label est une adresse de branchement dans le programme.

Annexe 2 : Historique

Version 0.1 (fin 2013) : Vérification de la faisabilité du projet
- création de l'interpréteur PICAXE BASIC (instructions de base),
- définition de 2 composants (Bouton et Led),
- la maquette est créée avec un éditeur de texte et l'exécution à lieu en fenêtre simple.

Version 0.1 (mars 2014) : Mise au point de la hiérarchie des classes utiles
- ajout de composants dérivés d'un composant de base,
- définition de 2 processeurs dérivés d'un processeur de base,
- mise au point de l'interaction actionneur → capteur.

Version 0.3 (mai 2014) : Création de la version de test
- Création de l'interface utilisateur, (passage d'Eclipse à Netbeans)
- Création de l'éditeur de maquette intégré.

Version 0.4 (été 2014) :
- Améliorations du code (en général)
- Meilleure gestion du son (Buzzer, en particulier)
- Regroupement des messages (fichier Bundle) pour une meilleure gestion (et une internationalisation future)
- Possibilité de supprimer un composant à partir de sa boîte de propriété.
- Prise en compte des modifications d'une maquette sans la recharger.
- Bouton de choix des fichiers images et contrôle de leur présence.
- Ajout du contrôleur PICAXE 08M

Version 0,5 (avril 2015)
- Changement de classe pour les Ldr → Analog
- Correction d'un bug lors de l'écriture dans le registre (READTEMP, READADC)

Version 0,6 (mai 2015)
- Ajout du composant télécommande (TELEC) et des codes associés (IRIN et INFRA)
- Traitement de la commande SOUND
- Passage en multi position (>4 pour les moteurs)

Version 1,0 (juin 2015) perte des sources

Version 1,1 transition (décembre 2015)
- Réécriture des sources manquants.
- Améliorations de sections de codes et corrections de bug.
- Adaptation pour utilisation avec PicaxeEditor6 (codes pour boucles et tests)

Version 1,2 (mars 2020)
- Adaptation pour utilisation avec PicaxeBlocklyApp (Ajout du contrôleur PICAXE 20M2 et gestion du nommage des broches par ports)
- Corrections de menus bug (commentaire maquette, traduction de texte)

Version 2,0 (mars 2020)
- Retour sous Eclipse et mise en package
- Correction de nombreux bugs sur l'éditeur de maquettes
- Amélioration de l'interface
- Organisation des maquettes et activité exemples