

Projet TER RobotTaf

Par :

Victor Coutellier - Julien Gauttier - Vladimir Karassouloff

Faculté des sciences d'Orléans - M1 informatique

Table des matières

1	Une introduction au domaine	2
2	Une analyse de l'existant	3
2.1	Le système LOGO	3
2.2	Le langage Squeak Etoys	4
2.3	Le langage Scrath Junior	4
2.4	Le système LEGO Mindstorm	5
2.5	Les systèmes annexes	6
3	Une liste et analyse des besoins non-fonctionnels	6
3.1	Prototype test	6
3.2	Architectures	6
3.2.1	Erlang	7
3.2.2	Pcode	8
3.2.3	Lua	9
4	Une liste et analyse des besoins fonctionnels	9
4.1	Application	9
4.2	Robot	11
4.3	Mini-Langage	12
5	Une description de prototypes et des résultats de tests préparatoires	12
5.1	Prototype android	12
5.2	Prototype robot	13
5.3	Prototype compilateur	13
6	Un planning, affectations des tâches	14

L'objectif de ce projet est de concevoir un mini langage servant à manipuler un robot afin d'enseigner les bases de la programmation aux jeunes enfants. Il s'agira de concevoir un langage de programmation contenant la plupart des notions de l'algorithmique et de le rendre accessible à des enfants. Une application Android devra être développée, les enfants devront alors créer un algorithme avec le langage préalablement conçu via cette application. Cette application servira d'interface entre l'enfant et le robot. Le programme produit par l'enfant devra ensuite être envoyé sur un robot qui lancera localement son exécution.

1 Une introduction au domaine

A l'origine, si l'apparition de la programmation a déclenché un élan d'apprentissage aux jeunes enfants, les premiers langages n'étaient pas adaptés à ces derniers. En effet, la syntaxe était très exigeante et austère. De plus, la programmation était souvent introduite dans des contextes non familiers aux jeunes enfants, tel que la génération de nombres premiers, ou dessiner des formes géométriques.

Dans son livre [7], Seymour Papert's indique d'un langage de programmation éducatif doit répondre à 3 critères

- « low-floor », c'est à dire simple d'accès
- « high ceiling », c'est à dire fourni des opportunités de créer des projets complexe avec le temps
- « wide-walls », c'est à dire suffisamment vaste pour engager différents types de projets

Mais ces 3 contraintes n'étaient pas satisfaisables au vu des limitations techniques de l'époque.

Il est également remarqué d'une personne apprend mieux en travaillant sur des projets personnels, c'est pourquoi il est important d'insister sur

- La diversité des projets (jeux, histoire, animations, simulations), afin que chacun puisse se retrouver dans le langage utilisé
- La personnalisation, il est donc important de pouvoir importer ses propres images, son, voix, ainsi que des outils pour créer ses propres ressources.
- le choix de la 2D dans le développement d'une application d'apprentissage est donc plus approprié, car en effet, il est plus simple d'importer des assets 2D, tout comme il est plus simple de les personnaliser et de créer un environnement cohérent autour

Il fut également remarqué dans plusieurs études - notamment le département des sciences et de la technologie de l'éducation de l'université de Mons [10], et le Department of Psychology of Nazarene College [2] - que les enfants ayant été confrontés à des problèmes algorithmiques de façon ludique progressent de façon significative dans des problèmes mathématiques. Mais malgré cela, il fut aussi démontré que laisser l'enfant livré à lui-même dans un environnement de programmation ne suffit pas à développer ses capacités, le rôle de l'enseignant est primordial pour observer de réels effets positifs sur le développement de ces compétences.

« Si nous observons des gains d'apprentissage élevés dans le domaine des grandeurs, des nombres et de la structuration spatiale, l'impact du dispositif semble plus réduit pour les compétences relatives à la résolution de problèmes et pour le raisonnement logique. Un autre résultat intéressant qui ressort de notre analyse est l'effet du scénario en termes d'équité. Les résultats des apprenants sont en effet plus homogènes au terme de l'apprentissage. » [10] Indique l'étude de l'université de Mons.

Pour Seymour Papert, l'utilisation de l'ordinateur de manière éducative afin de créer des objets, dessins, ou programmes via un environnement adapté devient un catalyseur pour un changement positif dans le parcours scolaire de l'enfant [1].

2 Une analyse de l'existant

2.1 Le système LOGO

Cet article [6] parle de l'évolution de l'enseignement assisté par ordinateur et de ses différentes approches tel que des didacticiels, des simulations, apprentissage des différents langages. En effet, il s'agit d'une approche ludique pour les enfants avec résolution des problèmes et exploration de la géométrie grâce à une mise à disposition de différents éléments, ici avec Logo, on dirige une tortue que l'on déplace à l'aide de diverses fonctions munies d'arguments pour effectuer des opérations (exemple : av 150 td 120 av 150 td 120 av 150 dessinera un triangle)

L'apprentissage de la programmation passe par l'apprentissage de la logique pour maîtriser les algorithmes. On remarque également que la programmation peut aussi apprendre dans d'autres domaines tel que la musique, calcul, etc. . .

Parle des avantages de la pratique de la programmation

- Permet d'aborder des concepts difficiles à aborder tel que la récursivité, les variables
- Capacité de prévoir le résultat lors d'une suite de diverses transformations d'un objet
- Découverte du concept d'automates
- Augmentation de la capacité à résoudre des problèmes

Logo permet d'acquérir de l'expérience réelle sur les objets de connaissance à travers des concepts tel que

- Le concept d'état (le fait que la tortue repérable sur l'écran par sa position et sa direction)
- Les commandes qui permettent de changer l'état de la tortue
- La transposition de la tortue dans le monde réel via des systèmes robotiques, permettant à l'enfant de ressentir directement les effets de son travail

Les enfants apprécient la possibilité de personnaliser les programmes en donnant les noms qu'ils veulent aux variables / fonctions. C'est un véritable "laboratoire d'apprentissage et de recherche" mis à disposition de l'enfant.

2.2 Le langage Squeak Etoys

Squeak Etoys [3] est un logiciel inspirée par LOGO, il s'agit d'un environnement de développement qui inclue des graphismes 2D et 3D, des images, textes, particules et permet même des pages web ou des clips vidéo. Cet environnement permet également de partager ses créations en temps réel avec d'autres utilisateurs à travers le monde. C'est un système gratuit, open source et multi-plateforme.

Tout dans Squeak est considéré comme un objet, et l'on peut ajouter des contraintes et donner des instructions à tout ces objets. En effet, l'interface est des plus minimalistes pour ne pas submerger l'utilisateur par des dizaines de boutons et d'options, afin de laisser l'enfant découvrir le logiciel à son rythme sans être intrusif sur ce qui est considéré comme fait ou à faire.

Squeak fusionne la notion de « zone d'édition » et de « zone de rendu », en effet, tout les objets sont modifiables à la volée lors de l'exécution du projet. Cela permet de clarifier l'interface dans le sens ou il n'existe qu'un seul contexte, celui de la page principale. Des bulles d'aides apparaissent tout de même avec un certains délais afin de guider l'utilisateur sur les informations utiles en fonction de sa sélection [12].

De part ces mécaniques, Squeak permet à l'utilisateur final d'accéder à la simplicité sous-jacente de différent concept important tel la gestion du son ou des images.

2.3 Le langage Scrath Junior

Scratch à été conçu afin de permettre de programmer à des personnes qui n'ont jamais été amené à une carrière de développeurs. Il a été désigné pour tout le monde, de tout âges, afin de créer des jeux, simulations, animations ou histoire interactives [4].

Le but de cette plateforme n'est pas de former des programmeurs professionnels mais bel et bien de permettre à des personnes de tout horizons d'exprimer leurs idées [8]. En effet, de nombreuses personnes jouent à des jeux vidéos, regardes des animations ou utilisent des logiciels, mais peut on l'occasion d'en créer, et c'est cette opportunité que donne Scratch, ces compétences ainsi acquises peuvent être transposées dans des domaines qui n'ont rien à voir avec l'apprentissage ou l'utilisation de la programmation.

En effet, certains langages permettait déjà ces possibilités, tel que Squeak Etoys ou Alice, mais scratch était destiné à pousser encore plus loin l'accessibilité et les possibilités [8].

Scratch fut conçu via les 3 principes suivant :

- Le rendre le plus flexible
- Le rendre plus riche de sens pour les jeunes enfants
- Le rendre plus social qu'un environnement de programmation traditionnel

Scratch est conçu comme de manière graphique [8]. En effet, aucune syntaxe n'est présente, les structures de données et de contrôles traditionnelles sont remplacées par des forme géométriques tel les pièces d'un puzzle. Par exemple, une boucle sera présenté en forme de C, de manière à pouvoir placer des blocs à l'intérieur afin de visuellement expliciter le concept de suite d'instruction se répétant.

Cette visualisation graphique permet donc à un enfant d'expérimenter toute sortes de combinaisons, tout cela couplé à une exécution des plus simples, en effet, le simple clic sur un élément déclenche son code dans l'application, il est même possible de modifier une séquence d'instruction à la volée pendant son exécution.

L'interface de scratch est prévu comme un bureau physique [8], des instructions peuvent ainsi être disséminée ici et là sans lien les unes envers les autres afin d'être réutilisées par la suite en cas de besoin. Ce mode de fonctionnement incite donc à la découverte et à l'expérimentation plus qu'un langage de programmation traditionnel, qui implique rigueur et une conception préalable.

Une extension physique de Scratch existe, la Scratch PicoBoard, qui permet aux projet Scratch d'interagir avec le monde extérieur via une batterie de capteur, tel un capteur de luminosité, un bouton, un potentiomètre et même un micro, afin de rendre les projet plus interactif et plus palpable par l'utilisateur [8].

2.4 Le système LEGO Mindstorm

LEGO Mindstorm est une gamme de la marque LEGO basée sur le concept de « brique programmable », c'est une brique qui permet de charger et d'exécuter des programmes utilisant toute une batterie de capteurs et d'actuateurs. Il devient donc très simple de créer un robot capable de se déplacer, d'attraper ou de reconnaître des objets. [5]

Cette approche de la programmation par la robotique est des plus enrichissante, car l'aspect programmation pur est distillé par tout l'aspect ludique de la construction de son propre robot, tout en recevant des feedback physique à partir de son travail [11]. Seymour Papert, déjà un des pionnier du langage LOGO, nomme cette pratique d'apprentissage de la programmation par des plateformes robotique le « constructionisme » [7]

Le second facteur de l'adoption des robots dans l'apprentissage de la programmation est la chute du prix du matériel nécessaire sur la dernière décennie. Il est en effet très bon marché même pour les écoles les plus modestes d'accéder aux composant nécessaires. Car même si ces composant ne fournissent pas la précision nécessaire à un usage industriel, ils sont largement suffisant pour un usage éducatif.

2.5 Les systèmes annexes

Ardublock est un programme de développement pour l'arduino a base de blocks. Cet outils à été testé sur des enfants et montre que l'apprentissage de la programmation en utilisant arduino est possible et qu'une évolutions dans certains domaines tel que la résolution de problèmes ont été observées chez les enfants. [9]

<https://www.aldebaran.com/fr/cool-robots/nao/en-savoir-plus-sur-nao> : robots existants possédant plusieurs capteurs afin d'interagir avec son utilisateur, on notera surtout la présence de capteurs pour se repérer, la possibilité de se mouvoir ainsi que la présence d'une carte wifi/ethernet.

3 Une liste et analyse des besoins non-fonctionnels

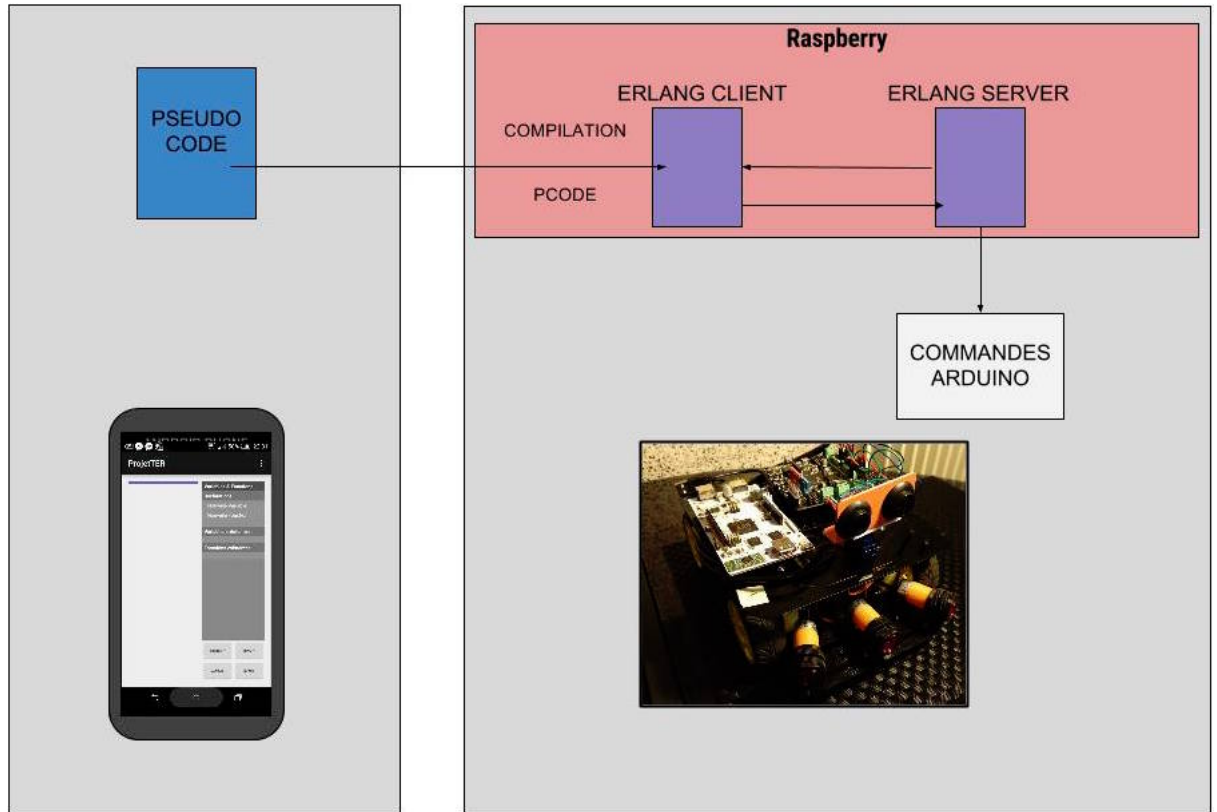
3.1 Prototype test

Télécommande : sera utile aux tests. En effet, le simple fait de contrôler le robot via une télécommande permet de vérifier plusieurs points. D'une part, cela permet de tester le fonctionnement et les capteurs du robot, et d'autre part, cela permet de se familiariser avec la "chaîne d'action" du projet. Enchaînant l'application android, la production du code en mini-language, la communication avec le robot, la compilation sur ce dernier et son exécution. Cela sera donc la première étape du projet.

3.2 Architectures

Nous avons imaginé ici différentes architectures possibles, elles diffèrent de par leur façon de transcrire notre pseudo code en Arduino, l'acheminement de la sortie pseudo code de l'application Android jusqu'à son exécution sur le robot peut en effet se voir de différentes manières.

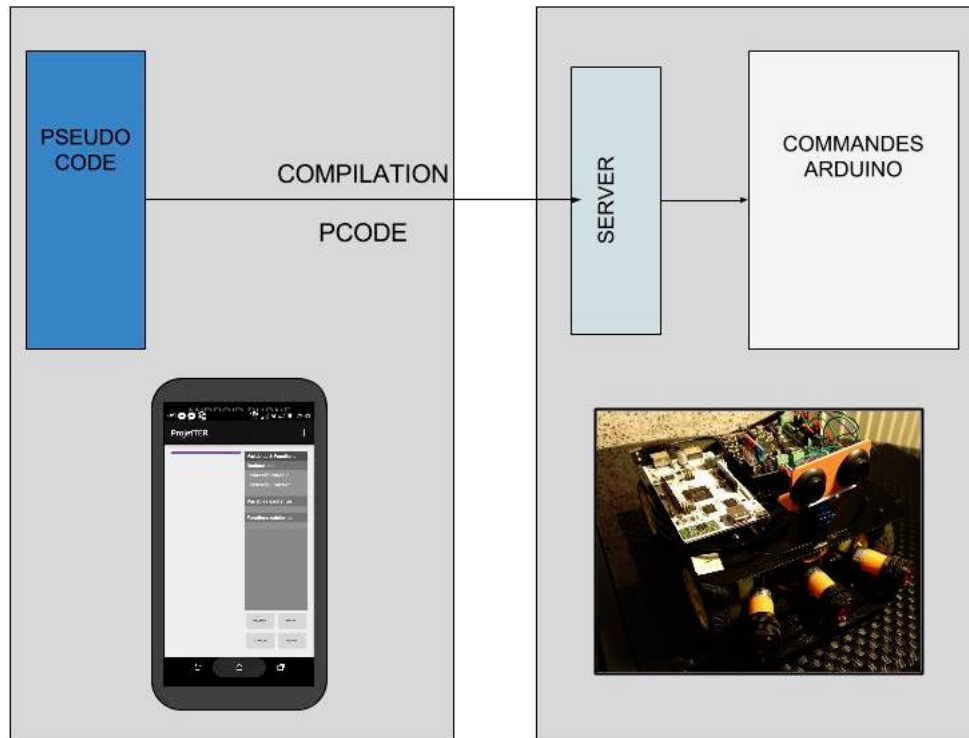
3.2.1 Erlang



Cette méthode profite de la facilité de communication entre plusieurs processus dont dispose Erlang, elle s'avère en premier lieu simple d'implémentations, on pourrait également se demander si une telle architecture ne serait pas simplifiable, et si nos commandes Arduino ne pourraient elles pas découler directement de la compilation.

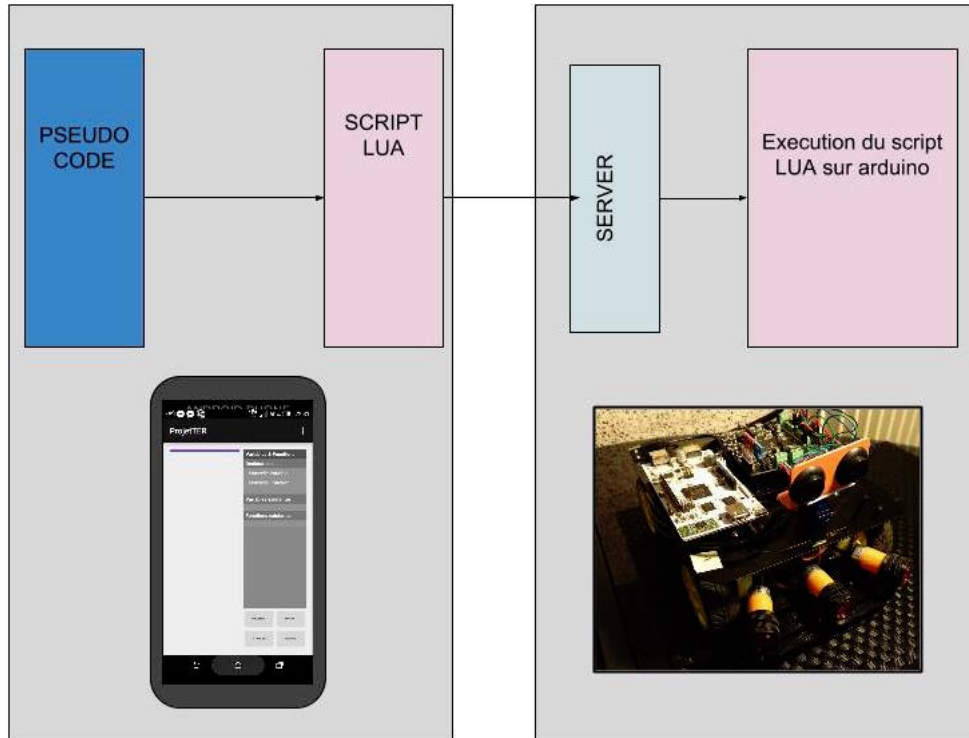
Le PCode nous sert de langage machine, en effet, ce langage proche de l'assembleur nous permettrait de l'interpréter en commande erlang directement exécutable par le robot.

3.2.2 Pcode



Avec cette méthode on allège le cheminement du code, mais on perd le côté pratique de Erlang, c'est une façon plus directe, on passe de notre pseudo code à des commandes Arduino avec plus de transparence. On se sers la encore du PCode comme langage machine intermédiaire, c'est ce PCode qui est interprété directement par l'Arduino.

3.2.3 Lua



Cette architecture est basée uniquement sur la capacité de la carte arduino à recevoir une implementation de LUA Script, si la carte controleur peut exécuter une version basique de LUA, il est donc possible de directement envoyer un script LUA sur l'arduino, et ce dernier l'exécutera sans problème sans autre forme de compilation.

La seule tâche à effectuer sera de transformer le pseudo code issu de l'interface de l'application android en LUA Script.

4 Une liste et analyse des besoins fonctionnels

4.1 Application

L'application Android servira à construire un algorithme à travers une interface utilisateur imaginée pour les enfants.

L'interface utilisateur de l'application est pensée pour les enfants, il nécessite donc de ne pas afficher directement du code face à l'utilisateur, mais plutôt des blocs s'imbriquant les uns les autres pour former un programme. Il est également nécessaire d'insister sur les formes et les couleurs afin de rendre l'application ludique pour l'utilisateur. En effet, toutes les combinaisons de blocs possibles qu'offre l'application doivent être exécutables, afin que l'enfant puisse comprendre directement les conséquences de ses actions, et non se confronter à des erreurs.

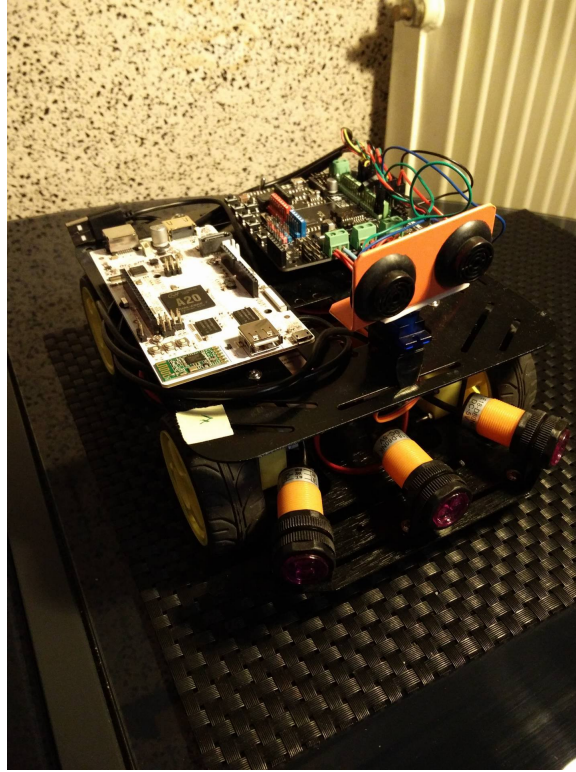
Les principales inspirations pour l'application furent notamment l'application LEGO EV3 et Scratch JR pour tablette, où les programmes sont composées de blocs tel des pièces d'un puzzle, où chaque bloc est utilisable indépendamment et représente une action. Ces deux applications montrent une fenêtre principale dite de "bureau", c'est à dire la fenêtre où l'utilisateur peut façonner librement son programme, en imbriquant les briques les unes avec les autres pour former un algorithme complet.

Sous cette fenêtre de bureau se trouve un système d'onglet, et chaque onglet correspond à une catégorie de blocs représentant une action. Par exemple l'onglet des capteurs proposera des blocs pour lire la valeur de tel ou tel capteurs, l'onglet des actionneurs contiendra des blocs pour actionner les moteurs, jouer un son ou émettre de la lumière.

Dans l'application LEGO EV3, un onglet flux permet de catégoriser les blocs contrôlant le déroulement de l'application, on y retrouve donc un bloc "début" symbolisant le début du programme, ou un bloc "répéter", en forme de C afin de montrer que ce bloc n'est qu'un conteneur pour placer d'autres blocs d'action à l'intérieur. Chaque bloc est ainsi personnalisable pour en limiter le nombre de base. Par exemple le bloc "jouer un son", une fois placé dans le programme, va permettre de choisir quel type de son jouer, et à quelle intensité pendant combien de temps.

Dans l'application Scratch JR, étant plus adaptée aux enfants, on remarque la possibilité d'importer et de modifier ses propres images et son depuis la tablette, afin d'offrir des capacités de personnalisation poussée du programme, pour que l'enfant s'identifie plus au projet. Il est également nécessaire que l'espace de travail soit perçu comme un bureau physique, où l'on peut laisser des blocs de côté, afin d'être réutilisés plus tard, ou seuls les blocs collés après le bloc "début" sont pris en compte dans l'exécution finale.

4.2 Robot



Le robot est, dans le cadre du projet, composé :

- D'un chassis à 4 roues motrices, sur lequel viens se fixer les differents capteurs, moteurs et carte électronique
- De 4 roues motrices actionnés par une batterie externe pour délivrer plus de puissance
- D'un capteur à ultrasons, posté sur un servo-moteur
- De 3 capteurs de proximités infrarouges
- D'une carte controleur de type Romeo, basée sur un Arduino Leonardo
- D'un mini pc tel un raspberry pi, avec wifi et bluetooth, fonctionnant sous ubuntu qui est relié en USB à la carte contrôleur
- De deux batterie, une pour la carte contrôleur et le mini pc, l'autre spécifique pour les moteurs

Le but du robot est de communiquer en direct avec l'application Android via Wifi, afin de recevoir le code produit par l'utilisateur sur l'application, et selon l'architecture choisie de directement l'exécuter sur l'arduino, ou de préalablement recompiler ce code pour l'exécuter ensuite.

Il sera par la suite possible d'améliorer ce robot afin d'ajouter différents capteurs, ou même de se passer du mini pc si les instructions passées à l'arduino sont déjà pré-compilées par le téléphone afin de les envoyer directement à la carte contrôleur via un shield Arduino Wifi, cela baisserait drastiquement la taille et le coût de production du dit robot.

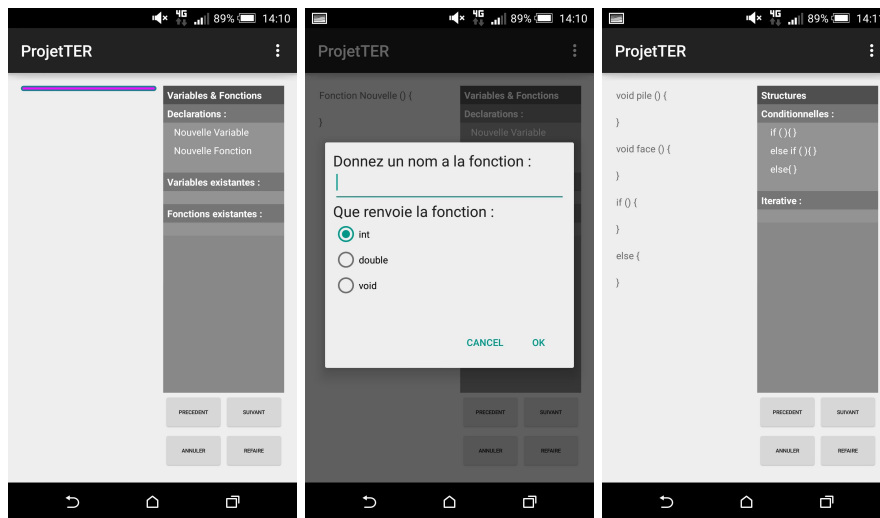
4.3 Mini-Langage

Le développement d'un mini-langage est crucial dans ce projet, en effet, l'application ne sert que d'interface pour facilement écrire ce langage, qui sera traduit de manière formelle via l'application. Ce langage formel, défini via une grammaire, subira plusieurs phases d'analyse soit : lexicale, syntaxique et sémantique, afin de traduire ce mini-langage dans un langage machine exécutable sur le robot afin d'effectuer les requêtes de l'utilisateur.

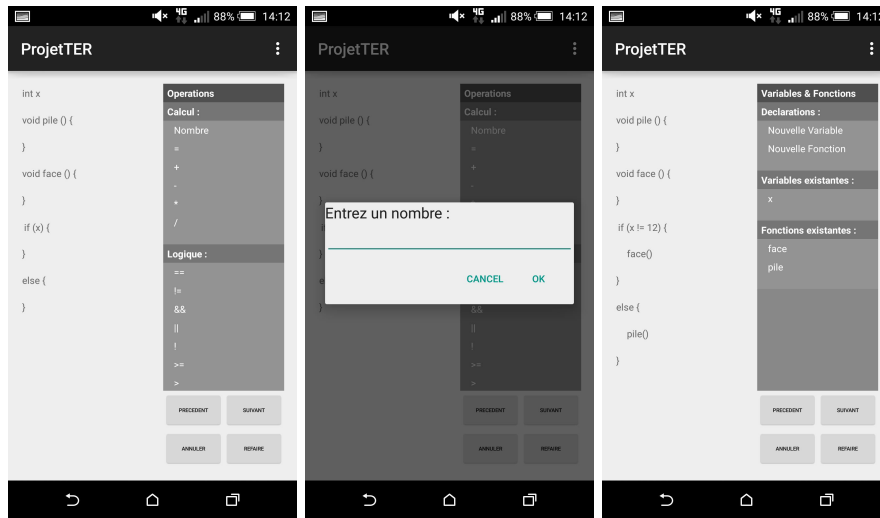
Ce mini-langage sera une interprétation du "bureau" de l'application, c'est à dire de la suite de blocs prise en compte dans le programme final. Le langage cible sera donc une série de commandes très simple, envoyée sur le port série de l'Arduino ou directement via wifi si une architecture sans mini pc est choisie, afin que ces commandes soient directement comprises et exécutées via le code de la carte contrôleur arduino.

5 Une description de prototypes et des résultats de tests préparatoires

5.1 Prototype android



Une Viewanimator à droite, contenant une liste de scrollview contenant différents éléments de programmations tels que les structures (conditionnelles et boucles), les variables ou encore les opérateurs de calculs et logique. Les différents éléments peuvent être drag and drop vers d'autres vues.



ScrollView a gauche servant de réceptacle aux éléments de codes. Lors du drag and drop, soit les éléments donnent lieu à une nouvelle ligne de code, soit les éléments enrichissent les lignes déjà présentes (exemple : drop d'une variable dans un if)

5.2 Prototype robot

Un code arduino a été écrit afin de comprendre des instructions basiques envoyées sur son port serie, et exécute des instructions en fonctions de la commande reçue.

Par exemple, on peut envoyer la suite de commande "w w w a s d" sur le port serie, qui aura pour effet de faire avancer 3 fois pendant 1 secondes le robot vers l'avant, puis de le faire tourner à gauche pendant une seconde, le faire reculer puis tourner à droite pendant une seconde.

Pour éviter tout dommage sur le robot, des routines dites "de sécurité" sont mises en place dans le code de l'arduino, ces routines permettent d'éviter au robot d'entrer en collision avec tout obstacle présent sur sa route. En effet, le robot verifie en permanence l'espace devant lui, et va chercher à tout prix à éviter la collision avec un obstacle, et cela même si l'instruction du programme qu'il est en train d'exécuter lui indique d'aller vers cette collision. Cette vérification pourra être désactivée mais sera déconseillée, car un choc à pleine vitesse peut avoir de graves conséquences sur l'intégrité du robot.

5.3 Prototype compilateur

Définition d'une grammaire via un outil d'analyse lexicale et sémantique, ATNLR4, qui permet via la description d'expressions régulières combinée de formuler un langage.

Une fois cette grammaire définie, l'outil permet de créer un AST, qui permet en-

suite d'effectuer l'analyse sémantique via une table des symboles, ce qui aboutira à la production du code machine exécutable sur le robot

6 Un planning, affectations des tâches

Victor : mini-langage et compilateur

Julien et Vladimir : travail sur l'application

Références

- [1] Douglas H Clements. Young children and technology. *Dialogue on early childhood science, mathematics, and technology education*, pages 92–105, 1999.
- [2] Douglas Degelman, John U Free, Michelle Scarlato, Janice M Blackburn, and Thomas Golden. Concept learning in preschool children : Effects of a short-term logo experience. *Journal of Educational Computing Research*, 2(2) :199–205, 1986.
- [3] Alan Kay. Squeak etoys authoring & media. *Viewpoints Research Institute*, 2005.
- [4] John Maloney, Mitchel Resnick, Natalie Rusk, Brian Silverman, and Evelyn Eastmond. The scratch programming language and environment. *Trans. Comput. Educ.*, 10(4) :16 :1–16 :15, November 2010.
- [5] Fred Martin and Mitchel Resnick. Lego/logo and electronic bricks : Creating a scienceland for children. In *Advanced educational technologies for mathematics and science*, pages 61–89. Springer, 1993.
- [6] Patrick Mendelsohn. L’enfant et les activites de programmation. *Grand N*, 35 :47–60, 1985.
- [7] Seymour Papert. *Mindstorms : Children, Computers, and Powerful Ideas*. Basic Books, Inc., New York, NY, USA, 1980.
- [8] Mitchel Resnick, John Maloney, Andrés Monroy-Hernández, Natalie Rusk, Evelyn Eastmond, Karen Brennan, Amon Millner, Eric Rosenbaum, Jay Silver, Brian Silverman, et al. Scratch : programming for all. *Communications of the ACM*, 52(11) :60–67, 2009.
- [9] Won-Sung Sohn. Design and evaluation of computer programming education strategy using arduino. 2014.
- [10] Gaëtan Temperman, Caroline Anthoons, Bruno De Lièvre, and Joachim De Stercke. Programmation tasks with Scratch : Observation and analysis of the progression in mathematics. *Frantice.net*, (9) :[http ://www.frantice.net/document.php?id=1013](http://www.frantice.net/document.php?id=1013), November 2014.
- [11] Jerry B Weinberg and Xudong Yu. Robotics in education : Low-cost platforms for teaching integrated systems. *Robotics & Automation Magazine, IEEE*, 10(2) :4–6, 2003.
- [12] Maizatul HM Yatim and Maic Masuch. Educating children through game making activity. *Otto-von-Guericke University of Magdeburg*, [http ://www.learnit.org.gu.se/digitalAssets/862/862887_yatim_masuch.pdf](http://www.learnit.org.gu.se/digitalAssets/862/862887_yatim_masuch.pdf), 2007.