BTS SN

E 6-2 – PROJET TECHNIQUE

Dossier de présentation et de validation du projet *(consignes et contenus)*

Groupement académique : Rennes	Session 2016
Lycée : Saint Joseph – La Salle	
Ville : LORIENT	
N° du projet :	**Nom du projet : rObOtScratch (1)**

Projet nouveau	Oui ☒	Non ☐	Projet interne	Oui ☐	Non ☒
Délai de réalisation			Statut des étudiants	Formation initiale ☒	Apprentissage ☐
Spécialité des étudiants	EC ☐	IR ☒ Mixte ☐	Nombre d'étudiants	3	
Professeurs responsables	Claude Guéganno				

Sommaire

1 – Présentation et situation du projet dans son environnement

1.1 – Contexte de réalisation

Constitution de l'équipe de projet :	Étudiant 1 EC ☐ IR ☒	Étudiant 2 EC ☐ IR ☒	Étudiant 3 EC ☐ IR ☒	Étudiant 4 EC ☐ IR ☒
Projet développé :	Au lycée ou en centre de formation ☒	En entreprise ☒	Mixte ☐	
Type de client ou donneur d'ordre (commanditaire) :	Entreprise ou organisme commanditaire : Oui ☒ Non ☐ Nom : Ecole Saint Christophe..... Adresse : 14 rue metayer 56100 Lorient..... Contact : Mme F. Le Gouic / M. Anthony Le Danvic..... Origine du projet : ➢ Idée : Lycée ☒ Entreprise ☐ ➢ Cahier des charges : Lycée ☒ Entreprise ☒ ➢ Suivi du projet : Lycée ☒ Entreprise ☒			
Si le projet est développé en partenariat avec une entreprise :	Nom de l'entreprise : Espace des Sciences / Maison de la Mer ..... Adresse de l'entreprise : 6 bis rue François Toullec - 56100 LORIENT..... Adresse site : http://www.maisondelamer.org..... Tél. : ..... Courriel :			

1.2 – Présentation du projet

(Présentation succincte / synoptique de l'architecture / limite de l'étude /attente du point de vue du client)

Le projet à développer rentre dans le cadre du développement des outils d'enseignement de l'informatique et de la robotique dans le bassin de Lorient. L' « **Espace des Sciences** » de Lorient , responsable de l'organisation de la « Fête de la science » en partenariat avec les écoles, les lycées l'université et les entreprises reçoit chaque année de nombreux visiteurs. À l'occasion de cet événement, nous présentons chaque année un produit dont les utilisateurs sont les écoles. Le projet présenté ici, sera présenté à l'occasion de la fête de la science 2017 .

Il s'agit de concevoir un système modulaire permettant à un groupe d'enfants en classe primaire de s'initier à la robotique, en ayant comme point d'entrée des notions de programmation par blocs (environnement de programmation « scratch », ou « Pocket Code »).

Le matériel mis en œuvre doit tenir dans un **budget réaliste** : il reste à demeure dans la classe. (Ce n'est pas une démonstration ou animation ponctuelle, avec un robot onéreux).

La personne en charge de la classe doit pouvoir s'approprier le système : après la mise en œuvre, il ne devrait pas avoir besoin d'assistance.

1.3 – Situation du projet dans son contexte

Domaine d'activité du système support d'étude :	☐ télécommunications, téléphonie et réseaux téléphoniques ; ☒ informatique, réseaux et infrastructures ; ☐ multimédia, son et image, radio et télédiffusion ; ☒ mobilité et systèmes embarqués ; ☐ électronique et informatique médicale ; ☐ mesure, instrumentation et micro-systèmes ; ☒ automatique et robotique.

1.4 – Cahier des charges – Expression du besoin

L'objectif du projet rObOtScratch est de fournir un package aux classes dans le domaine de l'éducation, dans le domaine de l'initiation à la robotique.

- Le point de départ :

- Une classe équipée d'ordinateurs portables disposant d'une interface *wifi*
- Les enfants ont reçu une formation sur la programmation par blocs (« Scratch » ou « Pocket Code »).

- Le système :

- Tablette ou smartphone équipé de l'application « Pocket Code » modifiée pour ce projet
- « **petitRobot** » : robot à très bas prix , composé d'un corps et de deux moteurs à courant continu, pilotés par un double pont en H, autonome (CPU ESP8266)
- PC de programmation Scratch (ou S4A = scratch for arduino)

Il y a deux sous systèmes à développer :

1/ Le robot qui sera utilisé **avec Scratch** : dans ce cas le robot fait office de hotspot wifi. Lorsque Scratch a rejoint le réseau du robot, l'apprenti roboticien peut alors piloter le robot avec les instructions Scratch (Fig. 1).

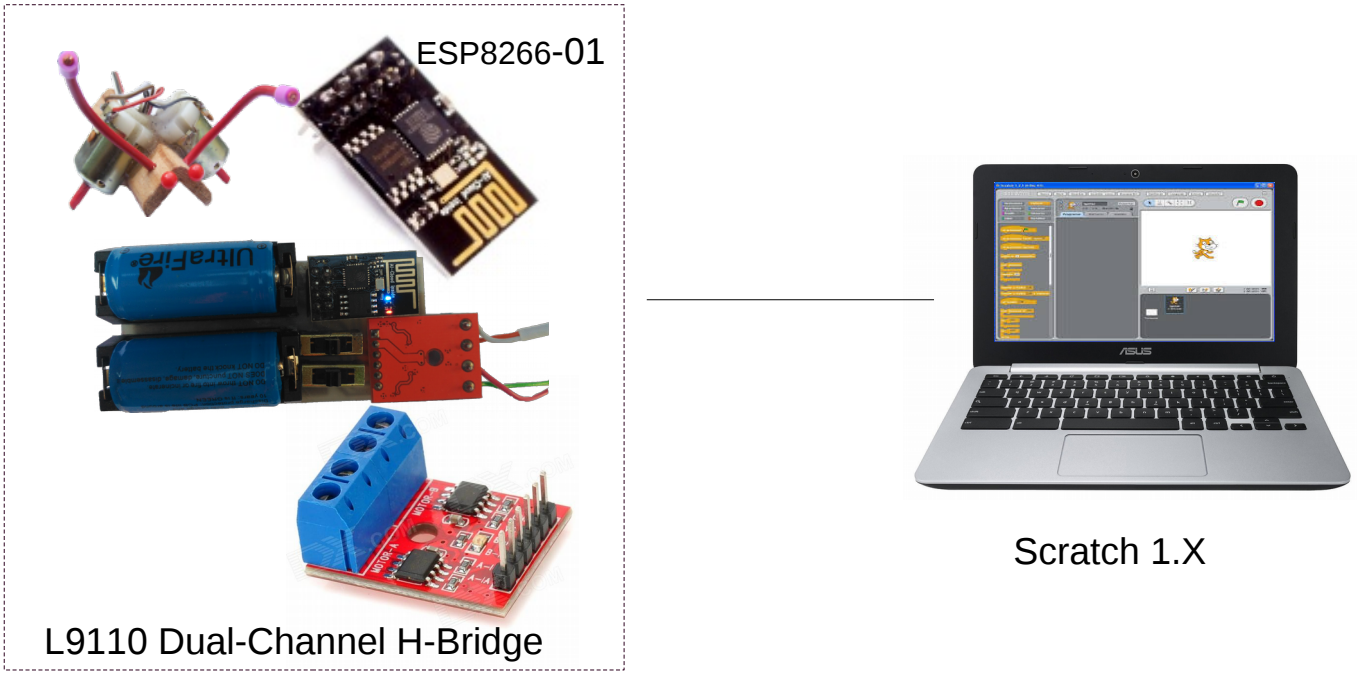


Fig. 1 – Le robot piloté par Scratch

2/ Le robot qui sera piloté avec « Poket Code » sur smartphone: dans ce cas , c'est le smartphone qui partage sa connexion et le(s) robot(s) présents viennent automatiquement s'identifier. Ils apparaissent automatiquement dans les briques du programme Pocket Code. Ceci nécessite bien sûr une modification en profondeur de l'application « Pocket Code » (Fig. 2).

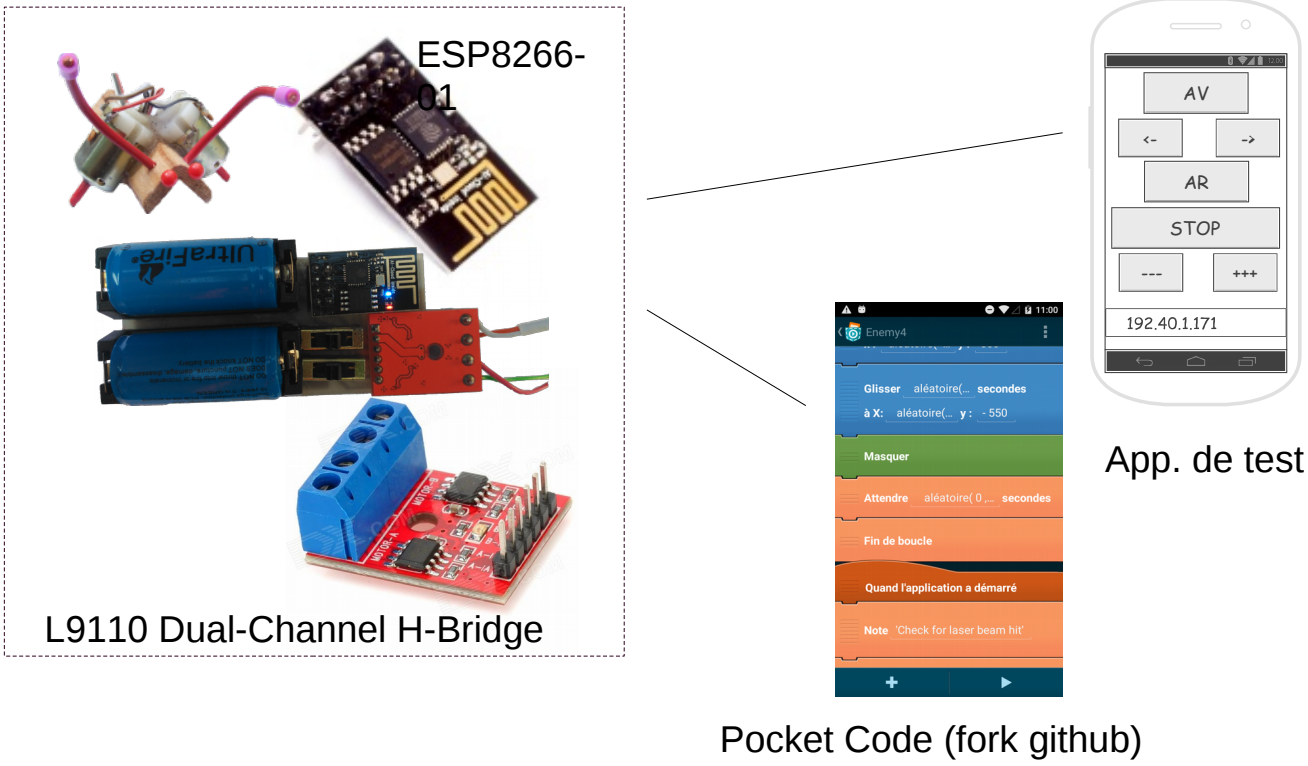


Fig. 2 – Le robot piloté par Pocket Code et une application de test pour Android

Le matériel mis en œuvre :

- CPU ESP8266-01 (GPIO + wifi)
- Des «rObOtScratch » moteurs CC + Doubles hacheurs en pont L9110
- Plusieurs postes clients équipés d'une image scratch autorisant les accès en réseau (join mesh).
- Stations de développement Android Studio
- Smartphone Android

La réalisation du projet est découpée en 4 parties (voir Section3)

2 – Spécifications

2.1 – Diagrammes SYSML

Diagramme d'exigence / Diagramme de contexte / Diagramme des cas d'utilisation / Diagramme séquence ...

- **Exigences (programmation avec Scratch E1 E2)**

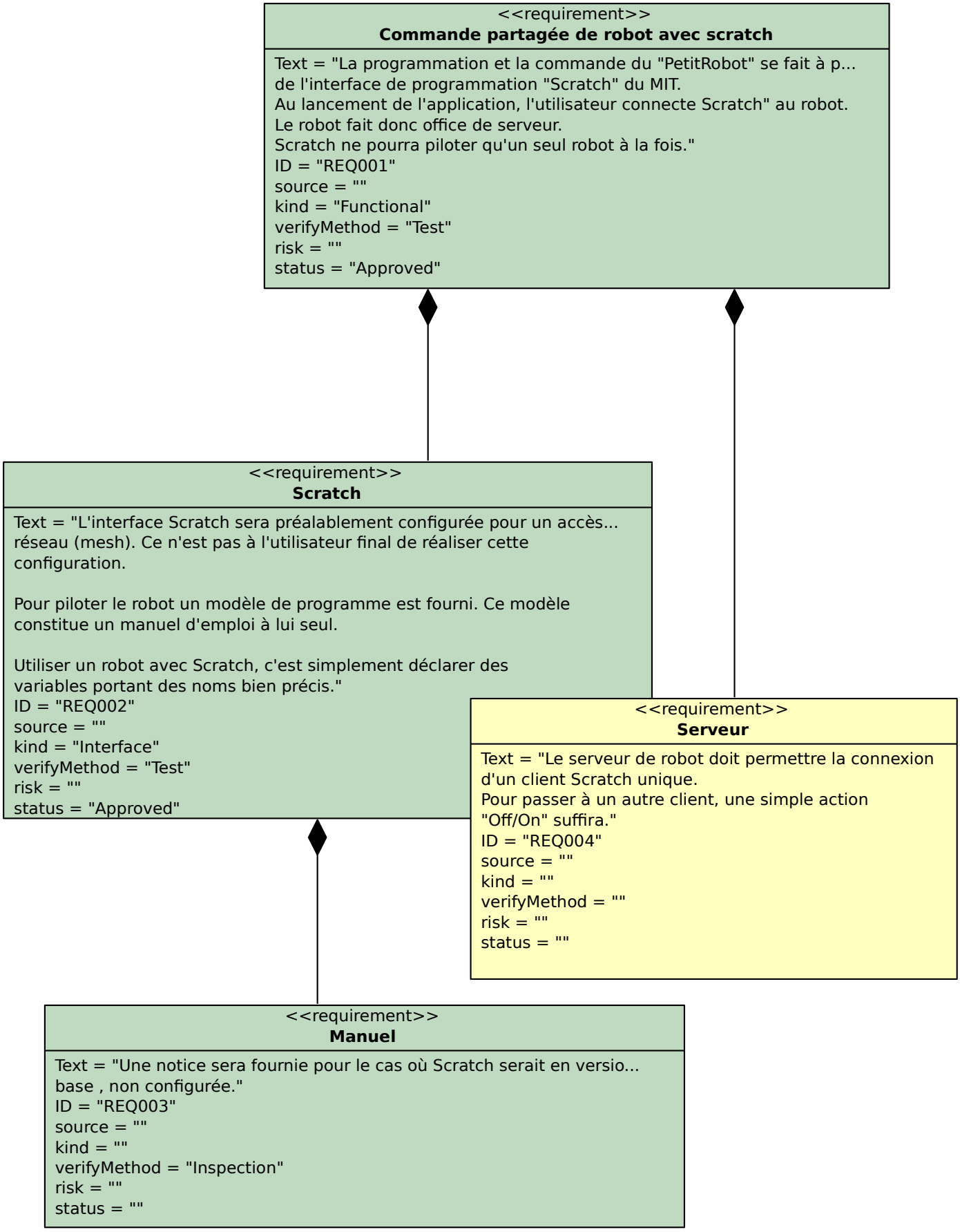


Fig.3 - Exigences pour la partie « Scratch »

La contrainte sur l'arbitrage des ressources est discutée. Dans l'état actuel des choses, l'exclusion mutuelle n'est pas à la charge du système informatique. La crainte de l'utilisateur final étant qu'un verrou reste permanent à cause d'un programme mal fait. C'est donc un travail collaboratif du groupe d'utilisateurs qui réglera le problème.

- Exigences (programmation avec Pocket Code E3 E4)

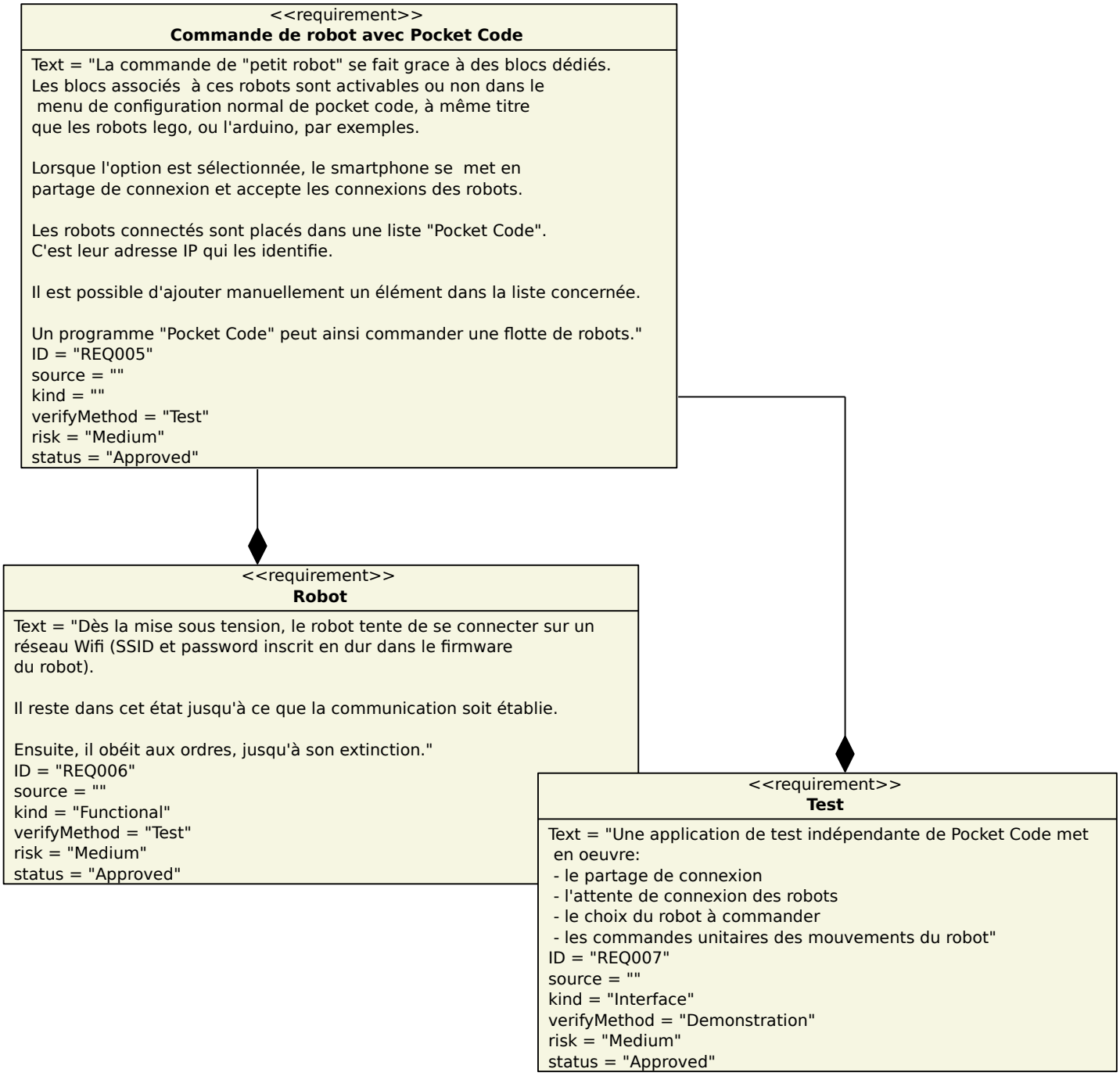


Fig. 4 - Exigences pour la partie «Pocket Code»

- Exigences pour l'application de test (*Wireframe android diagram)*

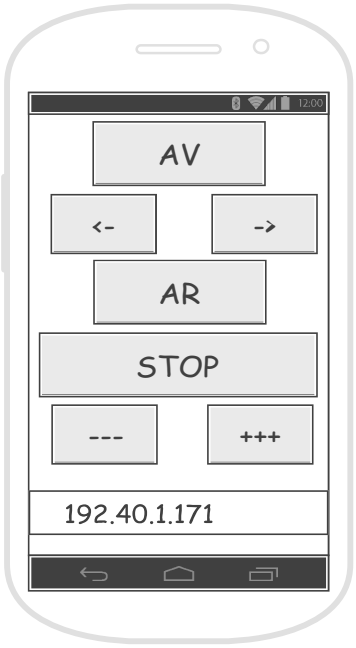


Fig. 5 - Diagramme pour l'application de test android.

- Au lancement de l'application, le partage de connexion est activé
- La liste du bas est une liste à sélection multiple, dans laquelle se placent automatiquement tous les robots qui se sont connectés.
- Tous les robots sélectionnés par l'utilisateur du programme de test sont concernés par les actions sur les boutons
  - AV : aller en avant
  - AR : aller en arrière
  - ← : tourner vers la gauche
  - → : tourner vers la droite
  - - - - : aller moins vite
  - +++ : aller plus vite.

Cette partie est à la la charge de E3. La limite du travail est la bonne réception des ordres dans la ou les carte(s) ESP8266.

- Cas d'utilisations

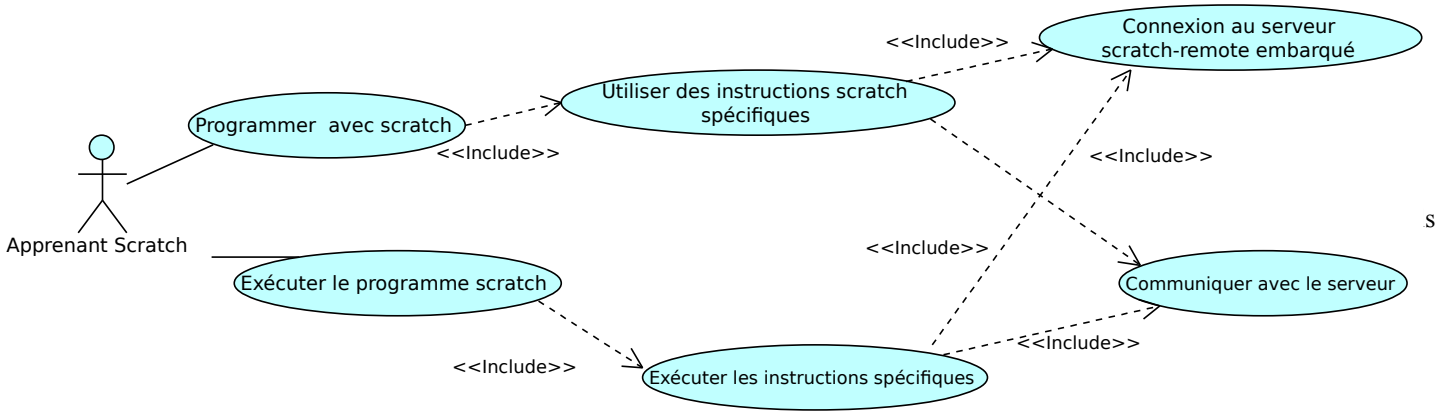


Fig. 6 - Cas d'utilisations pour la programmation avec Scratch

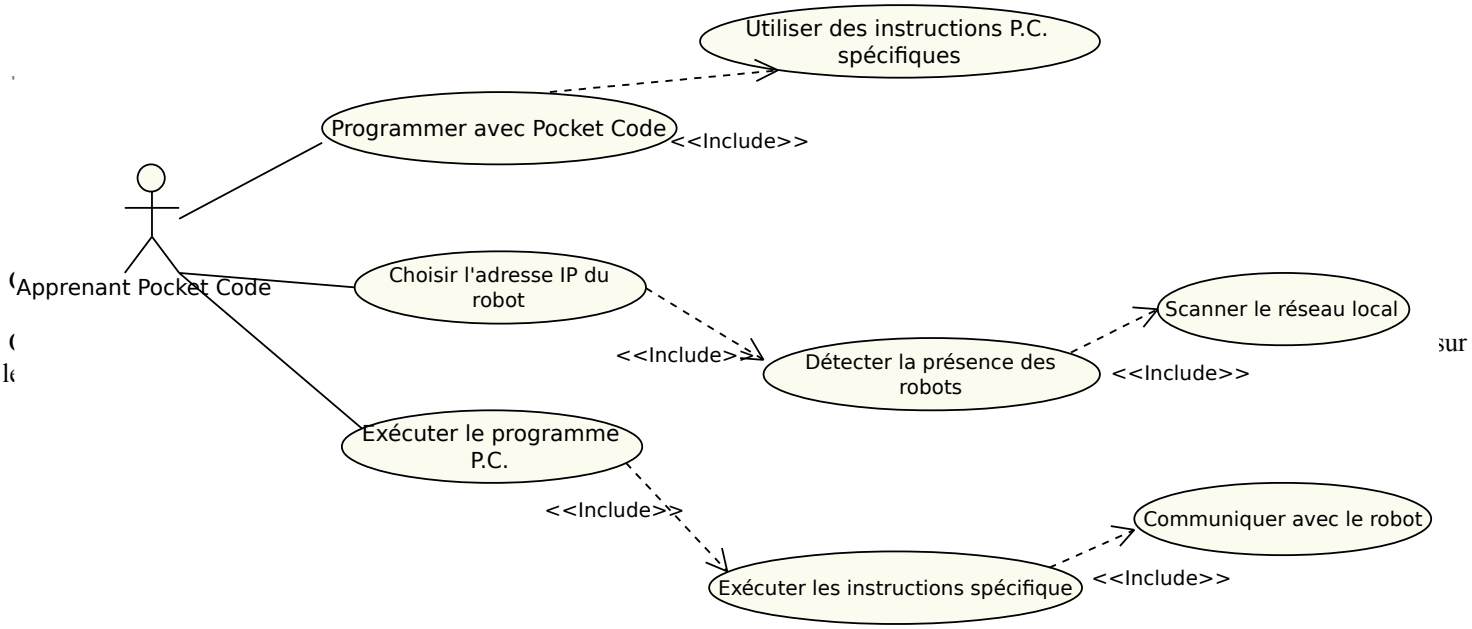


Fig.7 - Cas d'utilisations pour la programmation avec Pocket Code

Fig. 8 - Saisies d'écran de Pocket Code. Dans la version finale, rObOScratch apparaîtra dans les paramètres. Les blocs spécifiques seront activés dans les « écrans de sélection des blocs d'instructions. Dans les saisies d'écran, on voit le procédé appliqué au matériel arduino.

- Scénario général

1/ Partie Scratch

rObOScratch est un hotspot Wifi sur lequel se connectent des clients Scratch.

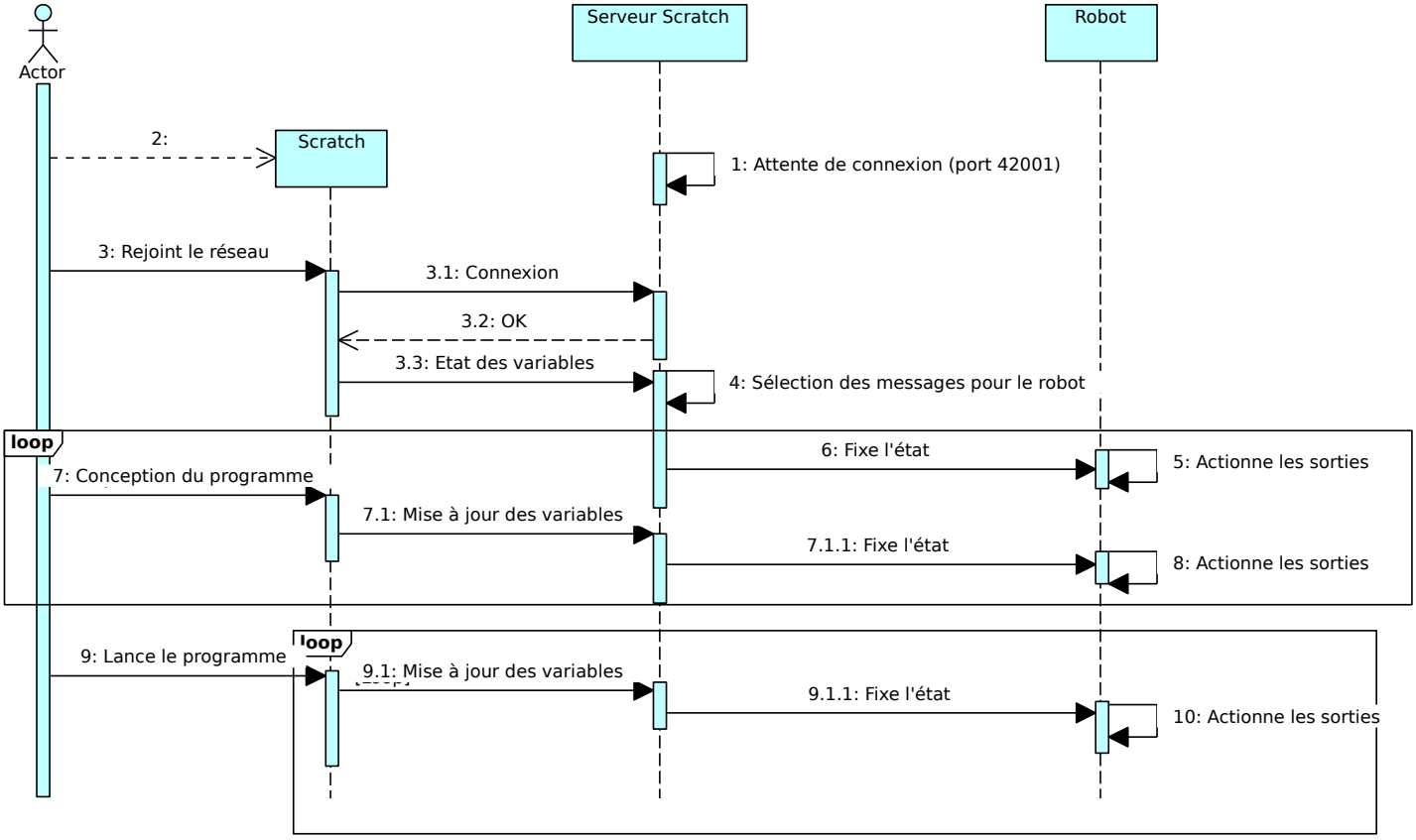


Fig. 9 - Scénario (partie Scratch)

La figure 6 donne le scénario nominal au démarrage du système .

1. « rObOtScratch » procède aux initialisations diverses en fonction du matériel connecté et de sa configuration
2. il attend les connexions des clients ;
3. lorsqu'un client est connecté, il est pris en charge par un *thread* et le système se remet en attente ;
4. un client connecté accède à l'ensemble des ressources disponibles

Aucun client n'est « propriétaire » exclusif d'une ressource pendant sa session. Il en résulte des problèmes techniques pour lesquels les solutions sont à apporter en projet . En effet, même s'il n'y a pas d'exclusion mutuelle au niveau d'une session, il convient de protéger le matériel et de garantir la cohérence des mouvements élémentaires.

Intérêt de gérer l'exclusion avec la granularité la plus fine :

1. Permettre au groupe de travailler de la manière la plus fluide possible, l'exclusion est alors gérée par le dialogue ou par l'arbitrage du maître
2. Autoriser un programme à gérer une action mettant en scène plusieurs personnages

2/ Partie Pocket Code

Le smartphone partage une connexion wifi à laquelle les rObOScratch se connectent (selon le [SSID, passwd] codé dans leur firmware.

Remarque : les firmware pour la partie « Scratch » et pour la partie « Pocket Code » sont donc différents, sauf pour ce qui concerne la partie motricité.

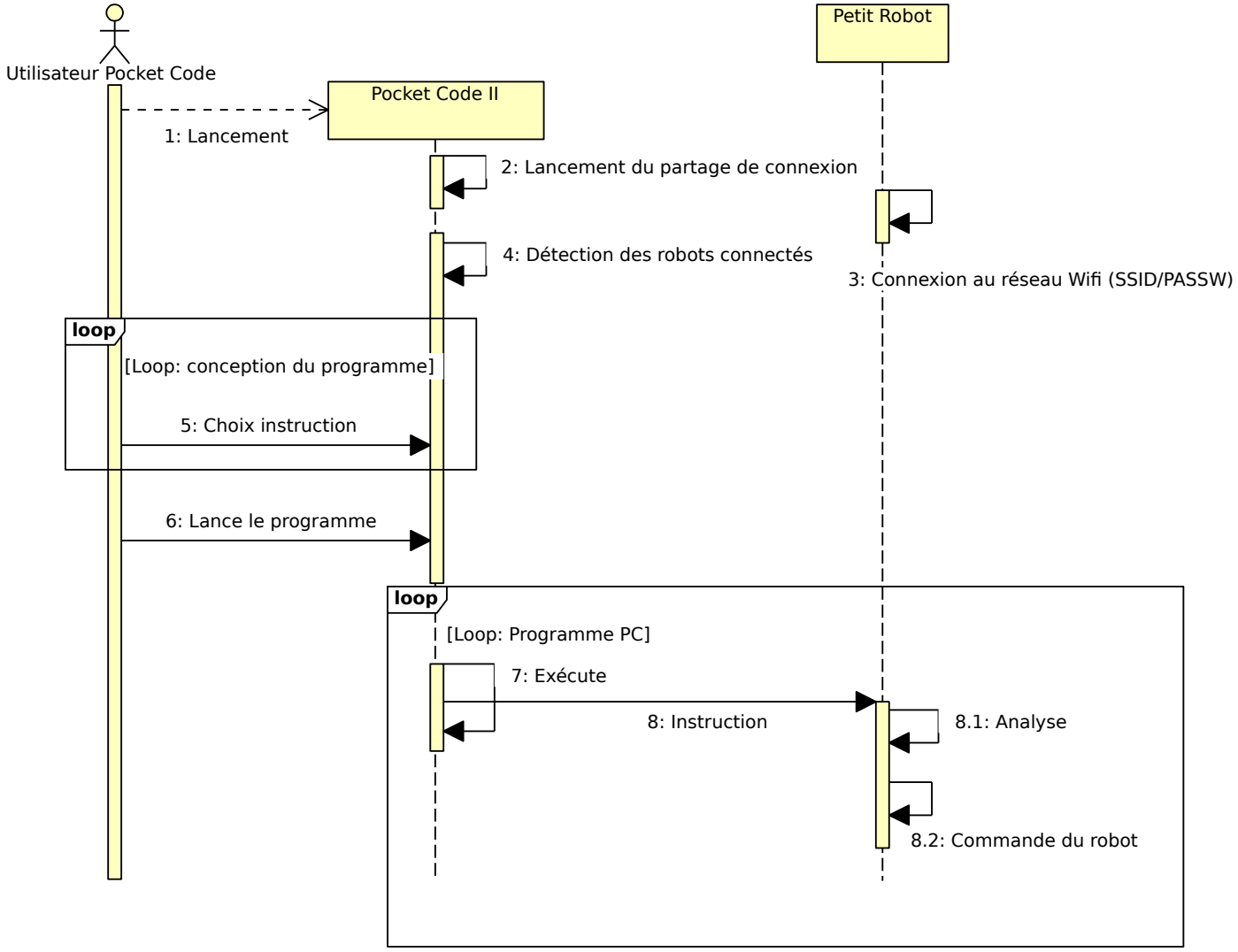


Fig 10 – Scénario pour la partie Pocket Code

- Architecture logicielle

1/ Logiciel embarqué dans le robot commandé par Scratch

Le diagramme suivant donne une idée de l'architecture du logiciel à écrire. Les méthodes sont données à titre indicatif.
La répartition du travail est donc :

E1 : classes `ServeurScratch`, `SessionScratch`

E2 : classes `Main` et `rObOScratch`

Bien sûr, au cours de l'analyse, d'autres classes seront créées.

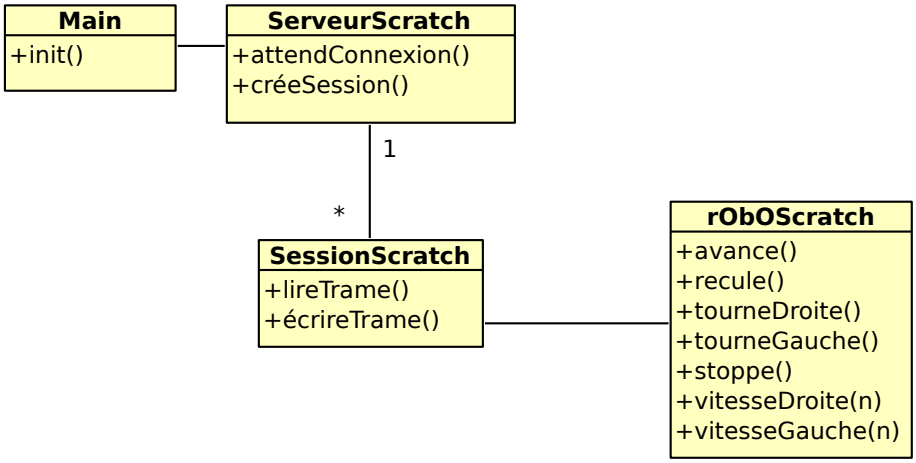


Fig. 11 – Logiciel embarqué dans le robot commandé par Scratch

2/ Logiciel embarqué dans le robot commandé par Pocket Code

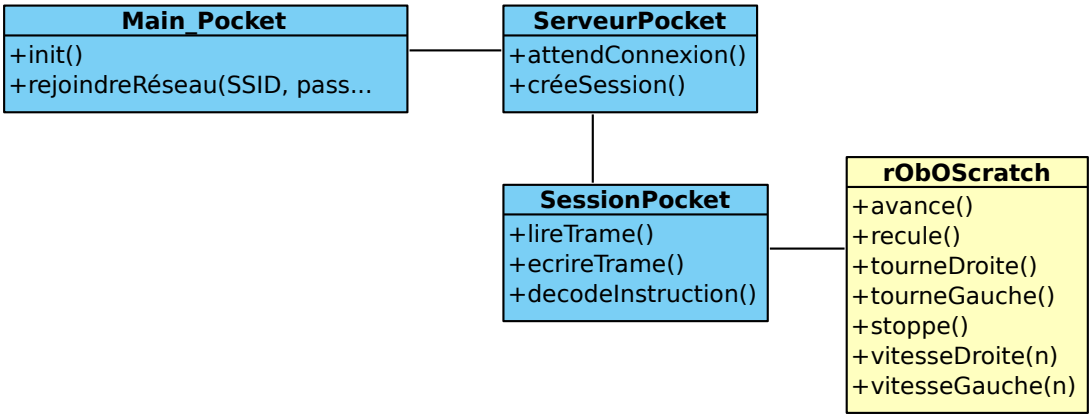


Fig. 12 – Logiciel embarqué dans le robot commandé par « Pocket Code ». La partie concernant la commande des moteurs est identique à celle de la version « Scratch »

3/ Application de test Android

C'est une architecture ordinaire d'application Android avec une activité unique. La figure 5 montre ce qui est possible. La gestion des connexions/déconnexions doit obligatoirement être traitée en multi-threading pour rentrer dans la technologie Android.

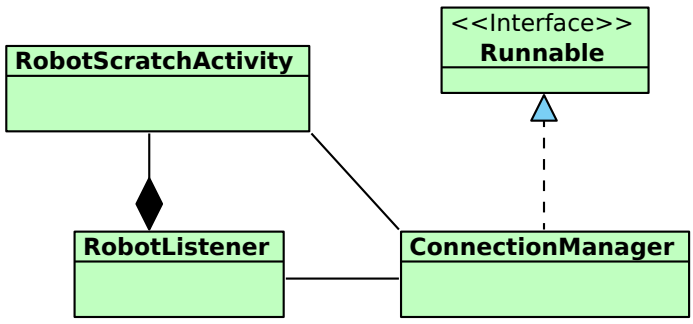


Fig. 13 – Modèle d'architecture de l'application de test

4/ Application Catroid

Dans notre projet, il s'agit du fork (github) de « Pocket Code » .
La version actuelle de « Pocket Code »contient plus de 750 classes, totalisant plus de 64000 méthodes. Au lieu d'hériter d'une classe dérivée d' `Activity` elle hérite de la classe `MultiDexApplication`, qui prend en charge les *« très grosses »* applications (celles qui dépassent les 64 K-méthodes) , la technique usuelle ne fonctionnant plus.

Des logiciels d'analyse comme « Visual Paradigm » en version professionnelle ne peuvent pas représenter le diagramme de classes dans son intégralité. D'autre part, une partie conséquente des relations entre classes est représentée de manière descriptive dans une liste de fichiers XML, ce qui, de toutes façons, complique sérieusement les représentations usuelles UML.

Pour y palier, les outils d'analyses de dépendances d' android studio seront d'un meilleur secours. D'autre part un contact (Paul Schreiner) est établi avec l'Université Technologique de Gräz (Autriche) porteuse du projet.

L'apport de ce projet à l'application Catroid concerne le scan du réseau local, l'identification des appareils connectés reconnus, et le pilotage par web-services d'une population d'appareils.

Notons enfin que « Pocket Code », au-delà du logiciel d'apprentissage devient un outil de prototypage rapide d'appareils connectés et de l' IoT (Internet of Things). En particulier, la technologie NFC, le pilotage du GPIO d'une Raspberry, d'une Arduino sont déjà prises en compte .

L'architecture logicielle de Catroid n'est donc pas présentée ici.
Il conviendra, dans le rapport de projet de présenter l'architecture par une liste de diagrammes de classes existantes simplifiés. Dans ces diagrammes , on fera apparaître les classes ajoutées.

La nouvelle fonctionnalité ESP8266 apportée par le projet, respectera le mode opératoire induit par les fonctionnalités existantes (NFC, arduino, PI ...)

2.2 – Contraintes de réalisation

Contraintes financières :

OPEN-SMART USB to ESP-01 Adapter + ESP-01 ESP8266 Wi-Fi Module	DX	5,82 €	3	17,46 €
L9110 Dual-Channel H-Bridge Motor Driver Module	DX	2,89 €	3	8.67 €
MPX06-3 Micro Motor Driver Module	DX	6,60 €	1	6.60 €
DIY 1-Slot 3V CR123A/16340/17335	DX	2,33 €	3	6.99 €
US Plug UltraFire 16340 Battery Charger w/ 4 x 3.6V "1000mAh" 16340 Batteries	DX	9,16 €	3	27.48 €
TOTAL TTC				**67.20 €**

(prix au 28 octobre 2016)

Contraintes de développement (matériel et/ou logiciel imposé / technologies utilisées) :

Contraintes logicielles :

- langage `Java` (pour Android)
- github -> `Catroid`
- Documentation `javadoc`
- Chaîne de développement C++ pour ESP8266

Contraintes qualité (conformité, délais, …) :

- Le produit est livré « prêt à l'emploi » avec une notice.
- Notice générale d'jout de fonctionnalité à Catroid.
- Notice de mise en œuvre Scratch
- Documentation pour la production d'ESP8266 pour Scratch
- Documentation pour la production d'ESP8266 pour Catroid (Pocket Code)

Contraintes de fiabilité, sécurité :

Il n'y a pas de contrainte de sécurité particulière. Le système est le plus ouvert possible.

2.3 – Ressources mises à disposition des étudiants (logiciels / matériels / documents)

- Livres et poly de cours Java / Android
- Documentation technique ESP8266
- Le matériel cité dans le tableau de la section 2.2
- 4 ordinateurs portables Linux
- Contacts avec l'Université Technologique de Gräz

3 – Répartition des fonctions ou cas d'utilisation par étudiant

E1 : **Communication et réseau** : liaison entre « scratch » et rObOtScratch
- Analyse de la partie personnelle
- Installation et mise en œuvre de la chaîne de développement ESP8266
- configuration de l'ESP8266 en « hotspot » wifi
- installation/configuration d'un serveur DHCP
- mise en œuvre du protocole scratch/server (documentation sur le site du MIT)
- configuration des clients scratch pour permettre la connexion au réseau (« join mesh »)
- production d'une image scratch définitive ne nécessitant plus de configuration
- Logiciel : serveur multi-sessions à l'écoute des clients scratch. Une tâche autonome est créée pour chaque connexion
  Remarque : l'exclusion mutuelle sur les ressources matérielles est proscrite . On ne doit pas interdire à deux utilisateurs de travailler sur le même robot.
  Le serveur multi-session est exécuté dans l'ESP8266
  Test : le logiciel fonctionne avec un robot virtuel (seulement des affichages).

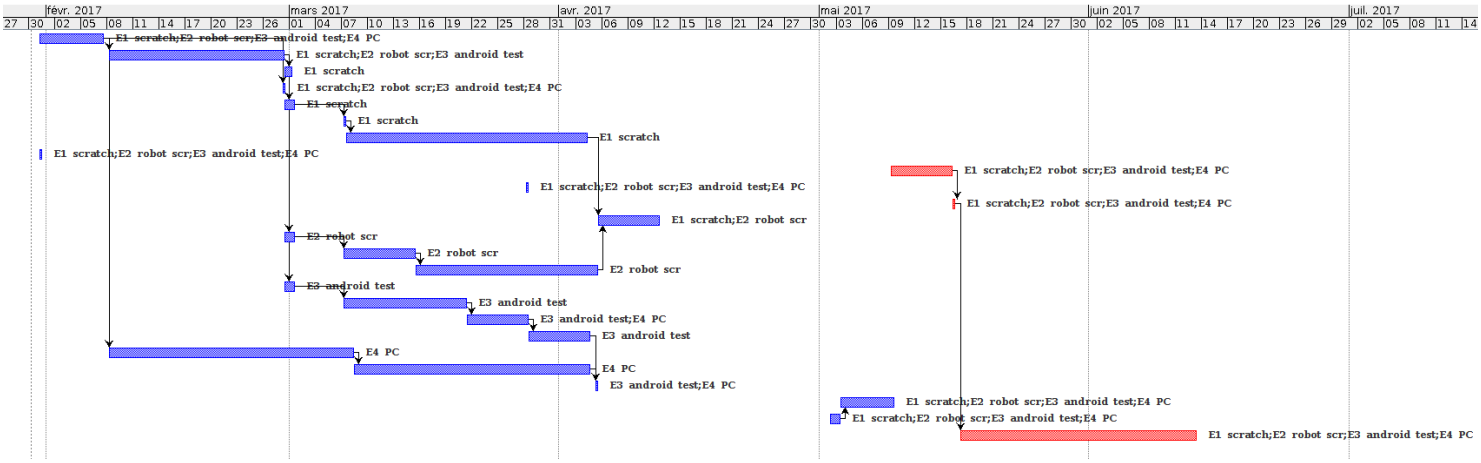
E2 : **Robot « rObOtScratch »** : Architecture logicielle et commande des mouvements de rObOtScratch
- Analyse de la partie personnelle
- montage, mise en œuvre et test du « rObOtScratch »
- étude du protocole de communication
- codage des fonctions de motricité du robot. En particulier, le contrôle indépendant des vitesses de chaque moteur doit être codé. Le matériel choisi pour répondre aux contraintes économiques (L9110) ne permet pas de le faire directement.
- choix des commandes/contrôles qui seront rendus disponible pour l'interface scratch
- Logiciel : classe(s) à intégrer dans le serveur pour rendre opérationnel le robot à partir du serveur
- Test :
  - test unitaire pour la connexion wifi
  - test unitaire pour l'intégration dans le serveur (sans scratch)
  - test d'intégration (avec un client scratch)

E3 :**Application de test Android**: étude de la communication et intégration dans rObOtScratch
- Analyse de la partie personnelle
- Étude de la communication Wifi entre rObOtScratch et Android : le smartphone partage sa connexion et le robot vient automatiquement se connecter au [SSID, password] programmé.
- Création de l'app. de test : les robots dans l'environnement s'identifient et sont commandés par une interface unique.
- Collaboration avec E2 pour l'intégration des classes de E2 dans cette partie (motricité du robot)
- Logiciel : classe(s) à intégrer dans le serveur pour rendre opérationnel le robot à partir du serveur
- Test :
  - test unitaire pour la connexion wifi
  - test unitaire pour les robots
  - test d'intégration.

E4 : «**Pocket Code**» : fork github de l'application Catroid
- Analyse de la partie personnelle
- Analyse de l'application Catroid avec Visual Paradigm : identification des classes à compléter, et des classes à écrire.
-
- Librairie à créer avec les commandes `droite`, `gauche`, `avant`, `arrière`, `stop`, et `vitesse`.
- Logiciel : classe(s) à intégrer dans le serveur pour rendre opérationnel le robot à partir du serveur
- Test :
  - tests unitaires pour i2c et la librairie
  - test unitaire pour l'intégration dans le serveur
  - test d'intégration
- Si le temps le permet chaînage de plusieurs « petitRobots » avec I2C. (2 cartes sont prévues)

4 – Planification (Gantt)

Début du projet	semaine 5 (30 janvier 2017).
Revue 1 (R1)	semaine 9 (28 février 2017).
Revue 2 (R2)	semaine 13 (28 mars 2017).
Revue 3 (R3)	semaine 20 (16 mai 2017).
Remise du projet (Re)	semaine 4 (26 janvier 2017).
Soutenance finale (Sf)	semaine 25 ? (?)
Livraison (Li)	semaine 27

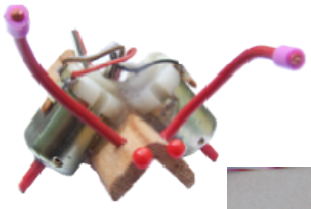


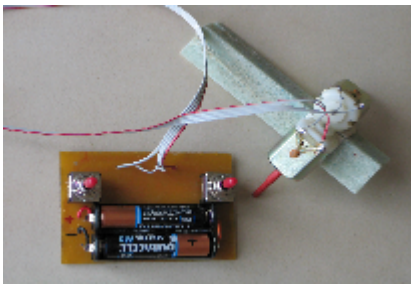
Analyse	4 jours?	30/01/17 08:00	07/02/17 17:00		E1_scratch;E2_robot_scr;E3_android_test;E4_PC
Inst. ESP8266	2 jours	08/02/17 08:00	28/02/17 13:00	1	E1_scratch;E2_robot_scr;E3_android_test
Hotspot/DHCP	1 jour?	28/02/17 13:00	01/03/17 10:00	2	E1_scratch
RV1	1 jour?	28/02/17 08:00	28/02/17 15:00	1	E1_scratch;E2_robot_scr;E3_android_test;E4_PC
Protocole Scratch	2 jours	28/02/17 13:00	01/03/17 17:00	2	E1_scratch
Image Scratch	1 jour?	07/03/17 08:00	07/03/17 15:00	5	E1_scratch
Serveur Multi sessions Scratch	10 jours?	07/03/17 15:00	04/04/17 10:00	6	E1_scratch
Documentation	1 jour?	30/01/17 08:00	31/01/17 15:00		E1_scratch;E2_robot_scr;E3_android_test;E4_PC
Rapport	3 jours	04/05/17 07:00	16/05/17 10:00		E1_scratch;E2_robot_scr;E3_android_test;E4_PC
RV2	1 jour?	28/03/17 07:00	28/03/17 15:00		E1_scratch;E2_robot_scr;E3_android_test;E4_PC
RV3	1 jour?	16/05/17 10:00	16/05/17 17:00	9	E1_scratch;E2_robot_scr;E3_android_test;E4_PC
Tests et intégration Scratch	3 jours?	05/04/17 15:00	12/04/17 17:00	7;15	E1_scratch;E2_robot_scr
Câblage Pont/ESP8266	2 jours?	28/02/17 13:00	01/03/17 17:00	2	E2_robot_scr
Classe Moteur	5 jours?	07/03/17 08:00	15/03/17 15:00	13	E2_robot_scr
Controle de vitesse	8 jours?	15/03/17 15:00	05/04/17 15:00	14	E2_robot_scr
Connexion SSID	2 jours?	28/02/17 13:00	01/03/17 17:00	2	E3_android_test
Protocole rObOScratch	6 jours?	07/03/17 08:00	21/03/17 13:00	16	E3_android_test
App Android.Test réseau	3 jours?	21/03/17 13:00	28/03/17 15:00	17	E3_android_test;E4_PC
App Android.Test E/S	3 jours?	28/03/17 15:00	04/04/17 17:00	18	E3_android_test
App Android.Catroid int. ESP8266	6 jours?	08/02/17 08:00	08/03/17 13:00	1	E4_PC
App Android.Catroid int. Blocs	10 jours?	08/03/17 13:00	04/04/17 17:00	20	E4_PC
Intégration Catroid	1 jour?	05/04/17 08:00	05/04/17 15:00	19;21	E3_android_test;E4_PC
Présentation générale	2 jours?	03/05/17 13:00	09/05/17 17:00	24	E1_scratch;E2_robot_scr;E3_android_test;E4_PC
Tests	2 jours?	02/05/17 07:00	03/05/17 13:00		E1_scratch;E2_robot_scr;E3_android_test;E4_PC
Documentation/Correction code	10 jours?	17/05/17 08:00	13/06/17 13:00	11	E1_scratch;E2_robot_scr;E3_android_test;E4_PC

5 – Annexes

- A - Quelques précisions sur « rObOtScratch »

« rObOtScratch » est le nom générique pour un projet sur la durée, initié dans le cadre de l'introduction du numérique dans l'éducation. Il est composé de deux moteurs à courant continu qui lui servent de roues. Son intérêt majeur est son coût très faible, et le fait qu'il puisse être fabriqué en classe. Il permet une connexion avec les arts plastiques, ce qui permet de contextualiser l'apprentissage de l'informatique et de la robotique dans un environnement , et un projet pluridisciplinaire.





Il est initialement prévu pour une commande directe avec une console électromécanique. Dans les photos on voit le kit monté et soudé. Sur la photo de droite, un petit robot habillé en arts plastiques.

Pour réaliser la connexion avec la programmation, un projet a été mené en 2015, basé sur ce matériel, connecté par fils à un poste de développement. La console électromécanique a été remplacée par un système électronique. Techniquement, les résultats étaient là, mais d'un point de vue pratique, l'utilisation de connexions filaires tant pour l'énergie que pour les commandes ont vite montré les limites du système.

Tout en ayant comme contrainte la maîtrise du coût de la réalisation, il s'agit ici de passer à un système bénéficiant d'une technologie de pointe tant pour le matériel que pour le logiciel.

- B - Un mot sur « scratch »

Scratch est un langage de programmation graphique (programmation par blocs) créé par le MIT en 2006 et largement utilisé dans le monde pour l'apprentissage de la programmation. Depuis l'apparition récente de la programmation dans les programmes des écoles primaires, le nombre d'utilisateurs ne cesse de croître. Il est également utilisé pour l'apprentissage de l'algorithmique et des notions avancées jusqu'au lycée et l'enseignement supérieur. Il permet d'aborder ;

- la programmation impérative habituelle (instructions, variables, répétitions, conditionnelles …)
- la programmation événementielle
- le parallélisme
- la communication par signaux
- plus récemment : la structuration de données

Un paramétrage caché et extrêmement sophistiqué permet de configurer scratch – entre autres - pour un accès à la programmation réseau. Ainsi, plusieurs programmes peuvent communiquer d'une machine à l'autre. C'est cette propriété que nous utilisons ici, mais en faisant communiquer scratch avec un serveur connecté à du matériel.

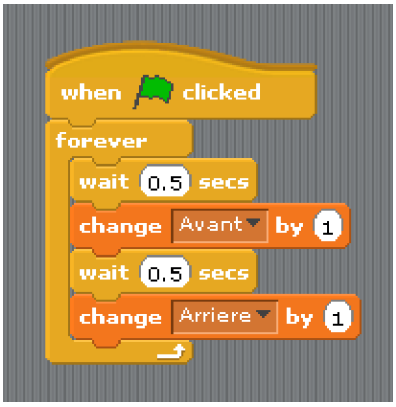


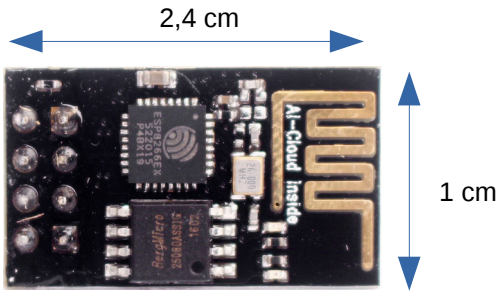
Fig 2 : Exemple de programme scratch

La figure 2 donne un exemple de programme scratch, qui pourrait être une application à la fin du projet : ici , on demande à un appareil connecté d'exécuter un mouvement alternatif. Dans ce cas particulier, l'utilisation de scratch n'est pas sans rappeler ce que l'on fait avec « Ardubloc » pour la programmation rapide des cartes arduino,

Il est important de noter ici, que le programme présenté dans la Fig 2 suffit à piloter le robot (pas d'initialisation, pas de déclaration préliminaire …). Les noms de variables *Avant* et *Arriere* correspondent à un robot précis.

Il sera important de définir un dictionnaire englobant les actions possibles pour les 3 robots visés.

- C- ESP8266-01



1/ Features

•ESP-01 is Wi-Fi serial transceiver module based on ESP8266.The SOC has Integrated TCP/IP protocol stack.It is TTL serial communication interface and its parameters can be set by AT command. It is widely used in networking, smart home project when it is connected to the wifi router. It can be used for remote monitoring of home appliances, bedroom temperature and humidity, controlling home appliances and smart car by the mobile phone.
USB to ESP-01 adapter module has CH340G USB to TTL driver IC onboard, so you can easily use your computer to do ESP-01 functional debugging.
Features:
- Working voltage: 4.5~5.5V (On-board 3.3v LDO Regulator)
- Working current: 240mA(Max.)
- Serial port baud rate: 115200 (default), can be modified to other values by AT command
- Serial communication format: 8N1
- Antenna Type: Built-in PCB antenna is available.
- Wireless Network Mode: station / softAP / SoftAP + station
- Wireless criteria: 802.11 b / g / n
- WIFI @ 2.4 GHz, support for WPA / WPA2 security mode
- Applications: Home automation, sensor networks, industrial wireless control, smart car.

2/ SDKs

In late October 2014, Espressif released a software development kit (SDK) that allowed the chip to be programmed, removing the need for a separate microcontroller.[5] Since then, there have been many official SDK releases from Espressif; Espressif maintains two versions of the SDK — one that is based on RTOS and the other based on callbacks.

An alternative to Espressif's official SDK is the open source esp-open-sdk[7] that is based on the GCC toolchain. ESP8266 uses the Cadence Tensilica LX106 microcontroller and the GCC toolchain is open-sourced and maintained by Max Filippov.[8] Another alternative is "Unofficial Development Kit" by Mikhail Grigorev.

Other open source SDKs include:

- NodeMCU: a Lua-based firmware.
- Arduino: a C++ based firmware. This core enables the ESP8266 CPU and its Wi-Fi components to be programmed like any other Arduino device. The ESP8266 Arduino Core is available through GitHub.
- MicroPython: a port of the MicroPython (an implementation of Python for embedded devices) to the ESP8266 platform.
- ESP8266 BASIC: An open source basic interpreter specifically tailored for the internet of things. Self hosting browser based development environment.
- Mongoose Firmware: An open source firmware with complimentary cloud service