BTS SN

E 6-2 – PROJET TECHNIQUE

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

Groupement académique : Rennes	**Session 2017**
Lycée : Saint Joseph – La Salle	
Ville : LORIENT	
N° du projet : 2	**Nom du projet : rObOtScratch (2)**

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

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.

Analyse de l'existant : le programme rObOscratch initié en 2014 a déjà donné quelques résultats. Certaines technologies ont été testées et abandonnées : RQ Huno pour sa fragilité, et Robosapien , des questions pratiques. Les meilleurs résultats ont été obtenus avec le « Petit Robot » , robot ultra bon marché, entièrement construit durant l'activité, mais avec l'inconvénient d'une liaison par fil entre le robot et le système de programmation.

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

il faut créer un robot programmable, télécommandable dont le prix unitaire est le plus réduit possible (20 € étant un maximum). À la différence du robot de la déclinaison (1) du projet, celui ci dispose de capteurs permettant de détecter des obstacles. La liaison doit être sans fil. La programmation du robot peut se faire à partir d'un ordinateur avec ***scratch 2*** installé. La version Scratch 2 en ligne ne dispose pas de l'extension HTTP expérimentale.

La communication entre Scratch 2 et le monde extérieur ne se fait plus comme dans les versions précédentes de Scratch (1.X). Scratch est configuré par un fichier ***json*** ou *XML* externe à concevoir en fonction des fonctionnalités attendues du robot, ainsi que des blocs qui doivent apparaître dans l'interface de programmation de Scratch 2. Lors de l'exécution du programme , Scratch 2 rend compte des actions occasionnées par l'exécution des nouveaux blocs en envoyant des messages vers un serveur particulier sur *localhost.* Il n'est pas possible d'utiliser une autre adresse, seul le port est choisi dans le fichier de configuration. Il convient donc d'écrire un serveur exécuté sur la même machine que Scratch 2, pour recevoir les ordres du programme. Le logiciel intégrant ce serveur est lui même client du serveur embarqué sur le robot. Le logiciel à concevoir sert donc de passerelle spécialisée.

Ainsi, la réalisation de ce projet met en œuvre :

1. la modélisation d'un robot élémentaire disposant d'E/S , cette modélisation se retrouve dans des blocs Scratch
2. L'écriture d'une passerelle Scratch / ESP8266
3. La conception d'un *firmware* adapté pour ESP8266-12 , embarquant un système temps réel *freeRTOS* ou *nilRTOS*
4. La création d'une librairie i2C pour ADC et commandes PWM, pour la carte ESP8266

- Le système :

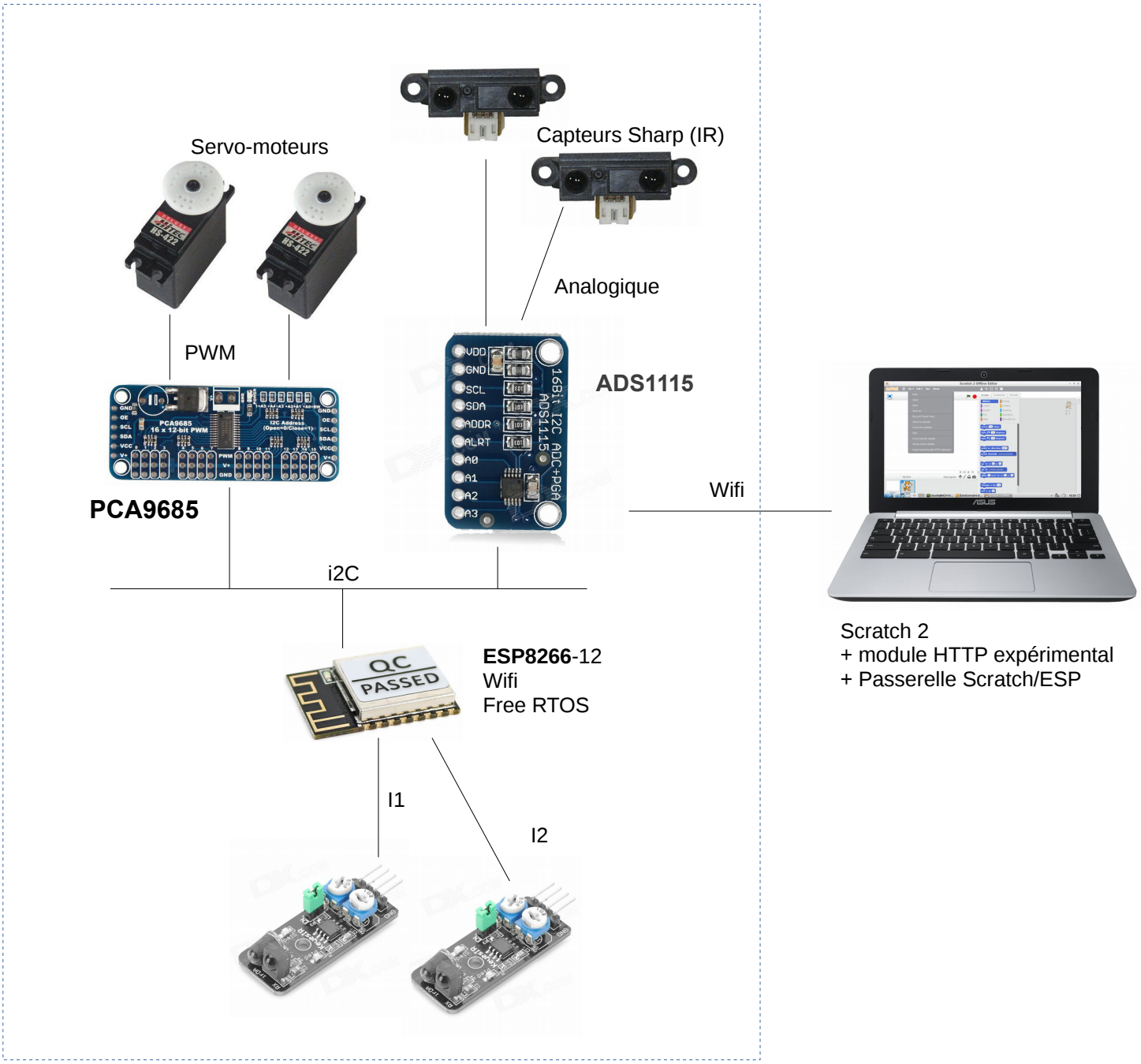


Fig. 1 – Aperçu du système

Description structurelle du système :

Principaux constituants :	Caractéristiques techniques :
Postes de programmation avec « scratch 2» (en ligne ou non)	Version de Scratch = 1.X possibilité d'utiliser S4A (Scratch for Android)
« rObOScratch-2 » autonome avec CPU ESP8266-12	ESP8266-12 (version améliorée) -

Inventaire des matériels et outils logiciels à mettre en œuvre :

Désignation :	Caractéristiques techniques :
Scratch 2	Logiciel de programmation par blocs pouvant être étendu par fichier de configuration
ESP 8266 - 12	Carte CPU/wifi à très bas prix : communication réseau, GPIO 8 e/s ; i2C; 1 ADC
« makeESP8266 »	Chaîne de développement C++ pour ESP8266
16 Bit I2C ADS1115 Module ADC 4 Channel w/ Pro Gain Amplifier	Module i2c pour l'acquisition de signaux analogiques (4)
16-Channel 12-bit PWM Servo Motor Driver I2C Module Board PCA9685	Module i2C de commande de servo moteurs (12)

2 – Spécifications

2.1 – Diagrammes SYSML

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

- Exigences (programmation avec Scratch 2)

Étant donné le contexte d'utilisation, les contraintes se résument à la facilité et l'immédiateté de la prise en main.

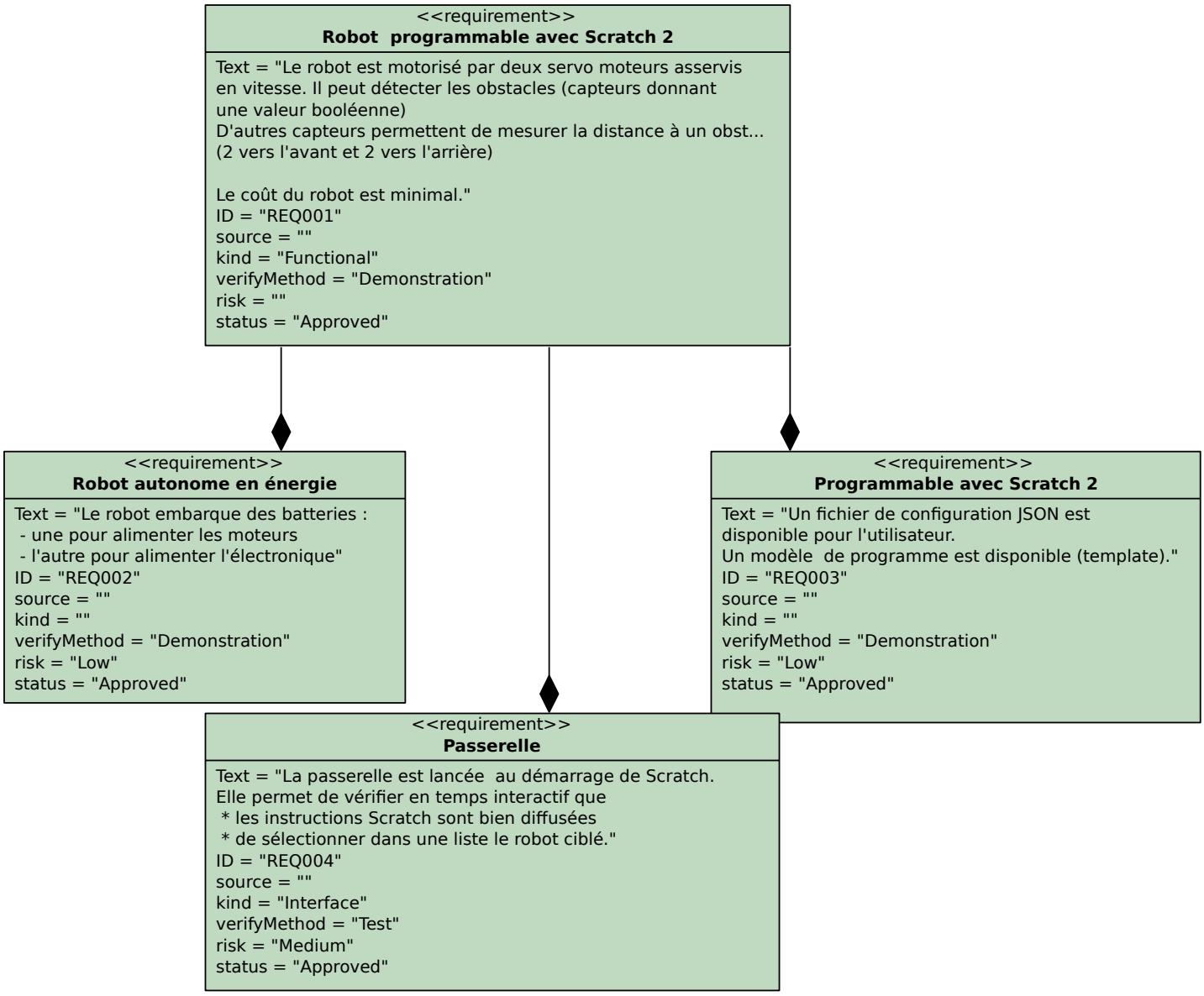


Fig. 2 – Diagramme d'exigences général.

- Exigences pour l'application « Passerelle »

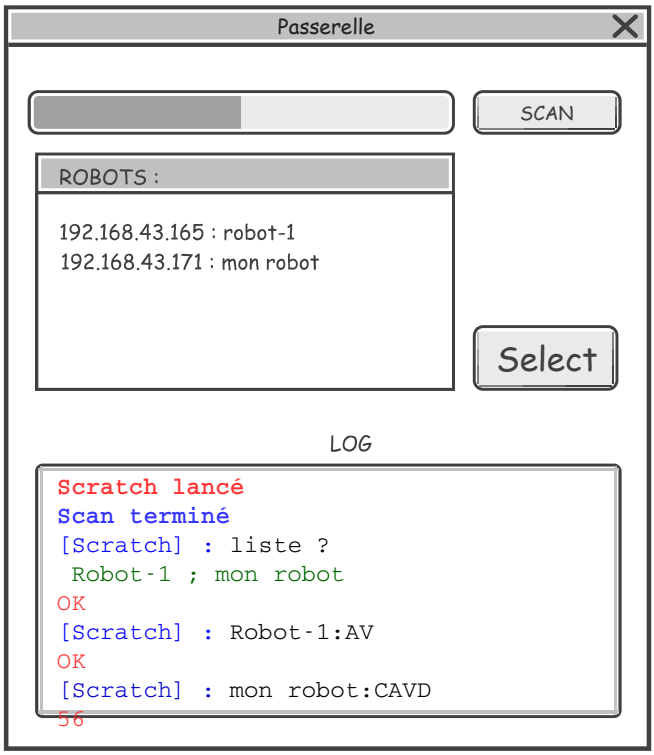


Fig. 3 - Diagramme pour l'application « passerelle ». Dans l'exemple de log, le programme scratch s'enquiert de la liste des robots disponibles. Il reçoit une liste de 2 robots. Un peu plus tard, un programme s'exécute, donnant un ordre différent à chacun des 2 robots détectés.

Ce diagramme permet d'affiner les exigences pour la passerelle :

- Au lancement de l'application, le scan du réseau est initié pour détecter les robots.
  - Une barre de chargement indique l'avancée du scan ;
  - à mesure que les robots sont détectés, ils apparaissent dans la liste du haut
- En parallèle, Scratch 2 est lancé.
- Une zone de texte (log) permet de rendre compte de toutes les actions effectuées.
- Chaque robot est identifié par une adresse IP et par un nom.
- Dans le programme Scratch, on peut utiliser indifféremment le nom du robot (indifférent à la casse) , ou bien l'adresse IP.
- Une instruction Scratch (à faire figurer dans les blocs json) permet d'importer dans Scratch la liste des robots connectés.
- Toutes les instructions de pilotage des robots utilisent comme premier paramètre le nom (ou l'adresse ip) du robot.
- On préférera cependant le nom, à cause de la portabilité du programme, dans un autre contexte.

Notes importantes :

1. dans les discussions préliminaires autour du projet, on explorera une autre solution consistant à **créer le fichier d'extension JSON de manière dynamique après le scan du réseau** . Cette méthode pourrait donner plus de souplesse.
2. Quelque soit la solution retenue, on fera bien la distinction entre (1) la création d'un programme Scratch, ou l'intégrité du fichier json est importante, et (2) une session scratch ou on modifie un programme existant. Dans ce cas, le fichier d'extensions est déjà inclus dans le fichier de sauvegarde Scratch (.sb2)

- Cas d'utilisations

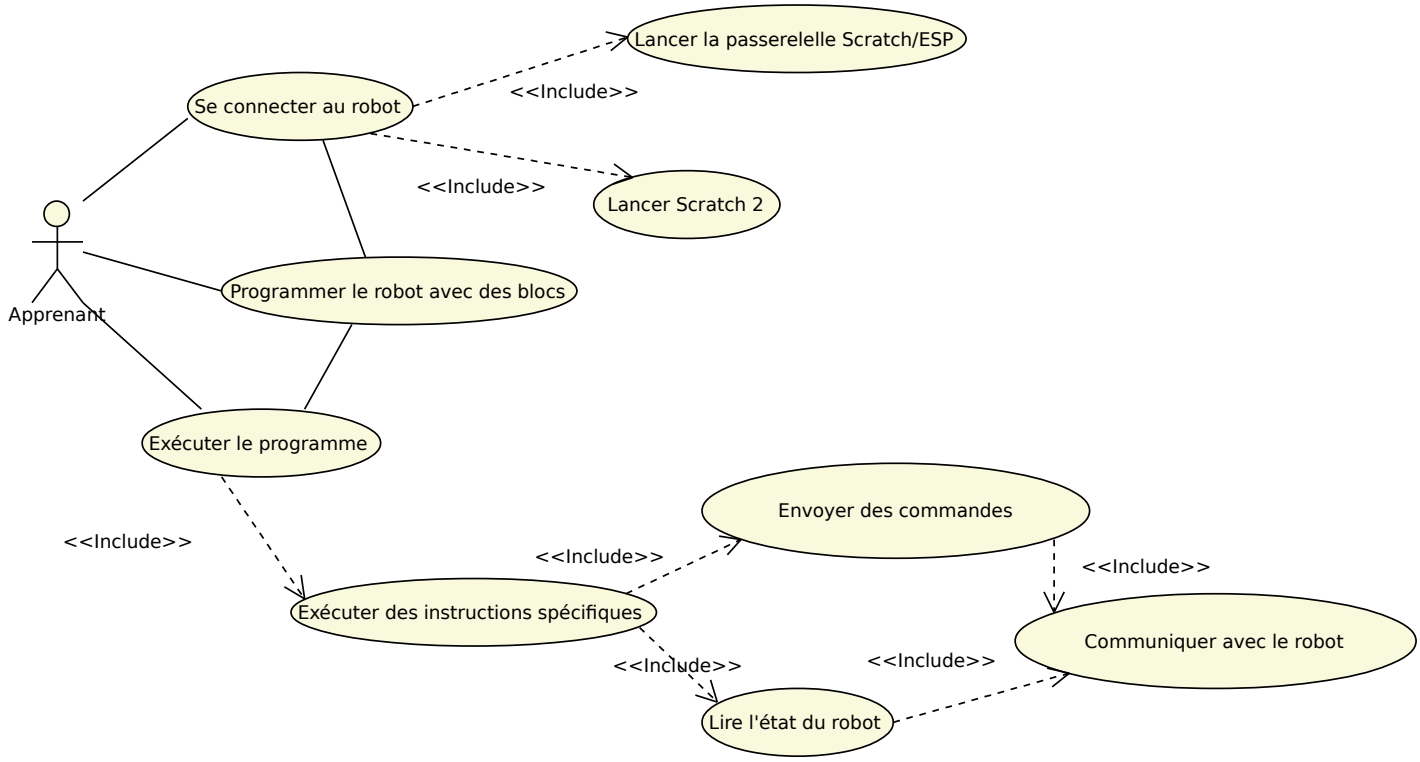


Fig. 4 – Cas d'utilisations

- Scénario général

1/ Partie Scratch

rObOScratch est en attente de connexion et d'ordres de la part de Scratch. Les ordres sont relayés par la passerelle sur lequel se connectent des clients Scratch.

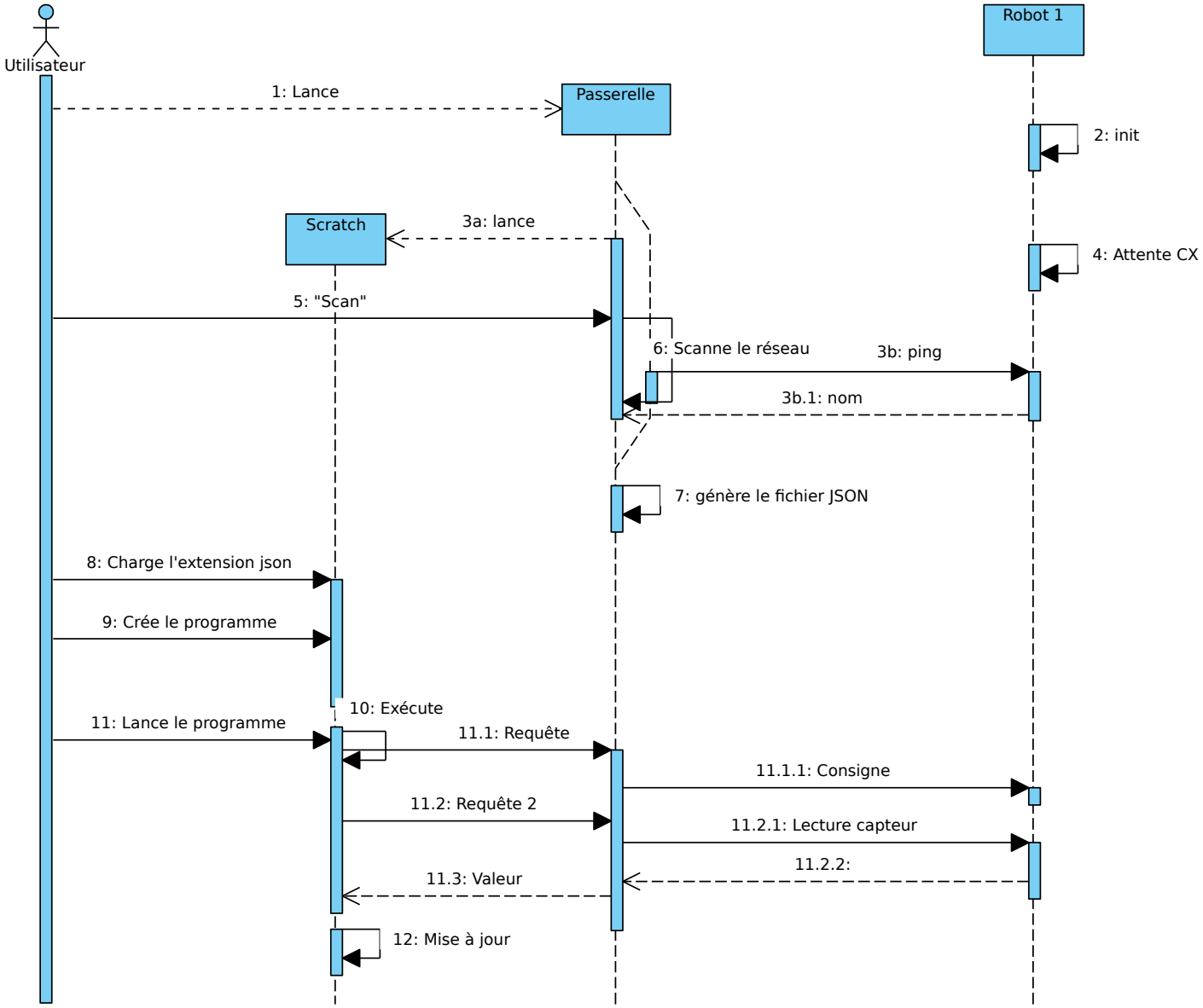


Fig. 5 – Scénario général. La création automatique d'un fichier json a été intégrée dans ce scénario.

La figure 5 donne le scénario nominal au démarrage du système .

1. C'est la passerelle qui est lancée en premier.
2. Elle lance [Scratch // scan du réseau];
3. Elle relaie ensuite les ordres de Scratch vers le(s) robot(s) concerné(s) ;

Aucun client Scratch 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

- 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 détection du robot peut se faire par un client Scratch pendant qu'un autre client lui fait exécuter un programme .

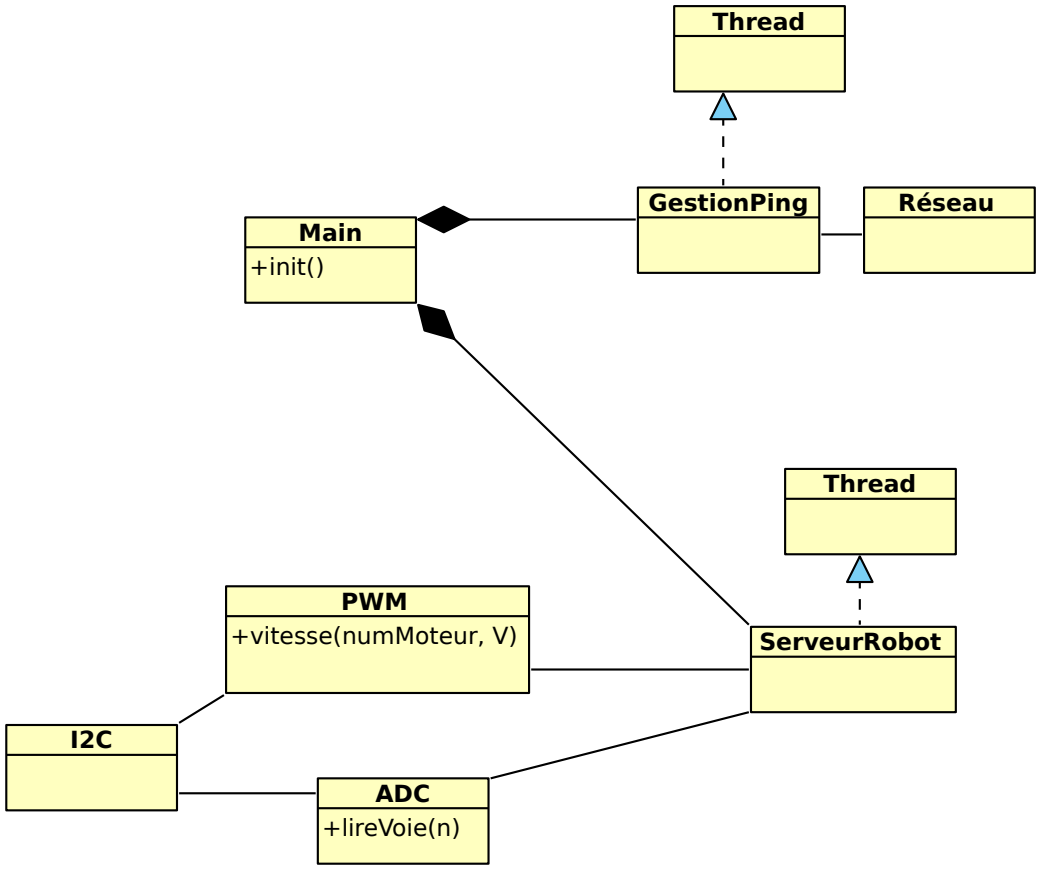


Fig. 6 – Logiciel embarqué dans le robot commandé par Scratch

2/ Passerelle

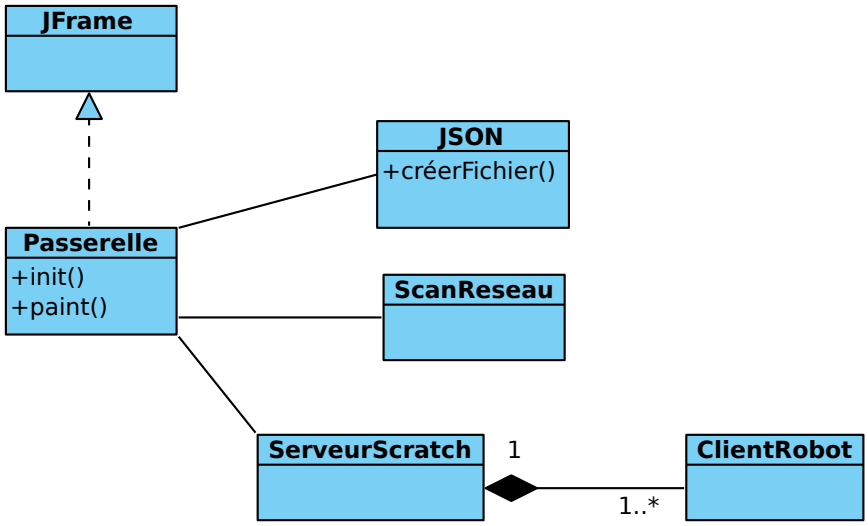


Fig.

2.2 – Contraintes de réalisation

(prix au 28 octobre 2016)

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

Contraintes logicielles :

- Passerelle sur PC = langage Java
- Documentation javadoc
- Chaîne de développement C++ pour ESP8266

Contraintes qualité (conformité, délais, …) :

- Le produit est livré « prêt à l'emploi » avec une notice..
- Notice de mise en œuvre Scratch
- Documentation pour la production d'ESP8266 pour Scratch

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 : **Réseau et firmware du robot** : 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
- Conception du serveur de commandes pour le robot. Ce serveur aura pour client la passerelle Scratch.
- Mise en œuvre de la partie i2C / ADC
- Mise en œuvre de la partie i2C / PWM
- Intégration avec free RTOS.

E2 : **Passerelle Scratch / «** rObOtScratch **»** :

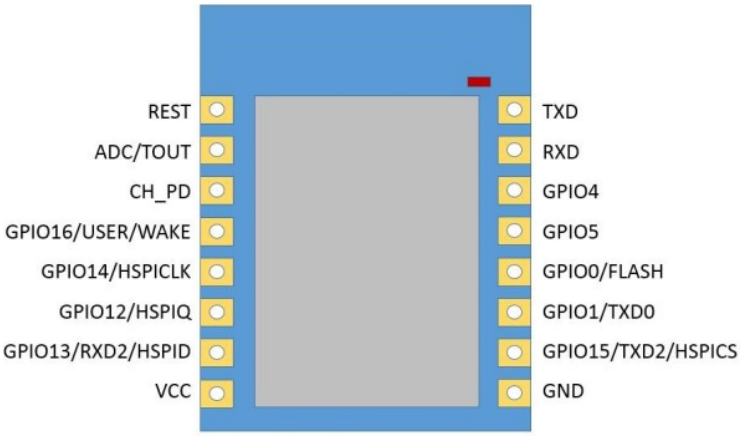
- Analyse de la partie personnelle
- Conception et test d'un fichier json
- étude du protocole de communication
- codage des fonctions de motricité du robot. En particulier, le contrôle indépendant des vitesses de chaque moteur doit être codé. Le matériel choisi pour répondre aux contraintes économiques (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)

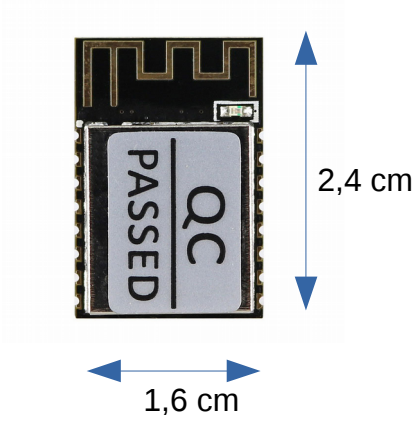
4 – Planification (Gantt)

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

5 – Annexes

- A- ESP8266-12





1/ Features

•ESP-12S is an enhanced version ESP8266-12 , improve the external circuit, enhanced impedance signal output is better, whether it is stable or anti-jamming capability , have been greatly improved ! Before selecting a user for ESP-12 's , it does not matter even if the product has been made , enhanced version is fully compatible with previous firmware. It can be used for remote monitoring of home appliances, bedroom temperature and humidity, and controlling home appliances by the mobile phone.
Features:
- Compatible with pin pitch and pinout of ESP-12
- Working frequency: 2400 - 2484 MHz
- Working voltage: 3.0 - 3.6VDC
- Working current: 200mA(MAX)
- Serial port baud rate: 115200 (default), can be modified to other values by AT command
- Serial communication format: 8N1
- On-board flash capacity: 4M bytes
- 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
- Note: When connecting with Arduino, you need to add 5V / 3.3V level converter circuit
- 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

- 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.

SCRATCH 2 - **Experimental Extensions**

Experimental extensions can be imported into Scratch by shift-clicking the **File** menu and selecting **Import Experimental Extension** in the offline editor. The resulting file dialog allows you to select an extension manifest JSON file, which contains the extension name, the port on which the extension runs, and the blocks which the extension defines. It follows this format:

```
{
    "extensionName": name,
    "extensionPort": port,
    "blockSpecs": [
        [type, spec, selector],
        ...
    ],
    "menus": {
        "menuName": ["Option 1", "Option 2", "Option 3"],
        ...
    },
}
```

The extension name should be an alphanumeric string in CamelCase, e.g., SensorBoard. The extension port can be any valid port number (above 1024).

Block specs include a block type (“ ” for command blocks, “w” for command blocks that wait until completion to continue, “f” for cap blocks, “r” for reporter blocks, or “b” for boolean reporter blocks), a block spec, and a selector (which should be a lowerCamelCase identifier). Selectors must not start with an underscore, as these are reserved for Scratch (more on that later). They can have additional array elements, which specify the block's default input values. In the blockspecs, you can use %m.menuName for a non-editable menu, or %d.menuName for a menu with an editable number parameter.

Extension files should end in .s2e.

Communication

When an experimental extension is imported into Scratch, it attempts to connect to a server running at the extension's port on the local machine using HTTP.

Scratch uses HTTP connections for extensions which declare “useHTTP”: true in their manifests. Before sending any other requests, Flash sends an HTTP request to http://127.0.0.1:extensionPort/crossdomain.xml and expects a cross-domain policy file in response. For example, an extension could send back the following policy file:

```
<?xml version="1.0"?>
<!DOCTYPE cross-domain-policy SYSTEM "http://www.adobe.com/xml/dtds/cross-domain-policy.dtd">
<cross-domain-policy>
    <allow-access-from domain="*.scratch.mit.edu" to-ports="*"/>
    <allow-access-from domain="*.media.mit.edu" to-ports="*"/>
</cross-domain-policy>
```

Command blocks

When a command block is executed, Scratch sends a request to http://127.0.0.1:extensionPort/selector and ignores the result. Any block inputs are appended this URL, escape-encoded and preceded by a solidus. For example, running the block [log [Hello, world!]] from the example extension above would send a request to http://127.0.0.1:49001/log/Hello%2C%20world%21.

Reporters

Scratch gets reporter values by sending a request to http://127.0.0.1:extensionPort/poll, and expects a newline-separated list of (selector value) pairs. String values *should* be URL-encoded, but spaces don't seem to work. Scratch sends a poll request ~30 times per second.

Example poll response:

```
brightness 75
slider 17
```

To send a message to Scratch that there is a problem with your extension (e.g. disconnected hardware), add a line to the poll response that starts with “_problem” , and then a problem to report to Scratch.
Example:

```
_problem Hardware is not connected
```

Commands that wait

When a “w” type command block is executed, Scratch adds a unique *command_id* parameter to the request before any other parameter.
Example:

```
/commandName/2437/3
```

For however long it takes to complete the command, add a _busy <commandID> line to your poll response (where <commandID> is the command id.

Reset command

When the stop sign in Scratch is pressed, Scratch sends a /reset_all command. When this is invoked, all hardware should be set to their original states.

Caveats

Projects with extensions are not to be uploaded to the Scratch website. The Scratch Team may host a small library of supported extensions, but these would have to fulfil very strict criteria.