Licence d'Informatique
Rapport de Stage
Du 5 novembre 2015 au 27 juin 2016
(durée effective de 4 semaines)

Extension des capacités musicales du robot éducatif "Thymio"

Stagiaire : Gaëtan CHAMBRES

Responsable de stage : Pierre COCHARD
Co-dirigeant de stage : Phillipe GUILLEM

Stage réalisé au sein du :

SCRIME
Batiment A37 - Château Bonnefont
Université de Bordeaux
351 cours de la libération
33405 Talence cedex

Remerciements

Tout d'abord, j'adresse toute ma gratitude à Monsieur Pierre COCHARD, en qualité de maître de stage, ainsi qu'à Monsieur Philippe GUILLEM, pour leur disponibilité, leur soutien, leur compréhension et leur attention. Leur confiance et le partage de leurs connaissances ont fait de ce stage un apport crucial de connaissances et d'expériences.

Je tiens également à remercier Madame Annick MERSIER pour son aide et son soutien tout au long de l'année, ainsi que pour son accueil chaleureux à elle et à toute l'équipe du SCRIME.
Je souhaite aussi remercier Madame Myriam DESAINTE-CATHERINE pour sa confiance et sa compréhension lors de ma première démarche, grâce à qui ce stage à pu voir le jour.

J'adresse également mes remerciements à toutes les personnes qui ont pu m'aider lors de mes recherches durant ce stage, toutes celles qui auront lu et répondu à mes mails de questionnement, et en particulier à Mr Didier ROY qui aura été un contact précieux à l'INRIA, notamment pour le prêt d'un robot Thymio afin d'effectuer mes premiers tests.

Je souhaite aussi remercier Madame Émilie DOS SANTOS pour sa disponibilité, son dynamisme et sa réactivité qui ont permis d'organiser au mieux ce stage.

Pour finir, je tiens à remercier mon entourage et mes proches qui m'ont soutenu toute l'année, tant dans mes études que dans ce stage, ainsi que ceux qui m'ont conseillé dans la rédaction de ce rapport, ou bien qui l'ont relu.

Résumé

Ce stage, effectué au Studio de Création et de Recherche en Informatique et Musique Électroacoustiques (SCRIME), lié à l'équipe Image et Son du Laboratoire Bordelais de Recherche en Informatique (LaBRI) s'intéresse au robot pédagogique Thymio, et en particulier aux moyens de le rendre capable de jouer de la musique. Une première partie est consacrée à la présentation du robot Thymio et du projet qui lui a donné vie. Une seconde partie présente les possibilités natives de Thymio concernant le son, tandis qu'une troisième partie détaillera comment ce stage à permis d'étendre ces possibilités.
Pour finir, ce rapport présente l'archive du travail rendu ainsi que les possibilités d'amélioraitons et les suites prévues pour ce projet.

Table des matières

1 Introduction **1**
1.1 Domaine du Stage 1
1.1.1 Le SCRIME 1
1.1.2 Les missions du SCRIME 1
1.1.3 L'équipe du SCRIME 2
1.2 Le contexte du stage 2
1.2.1 Pourquoi ce projet ? 2
1.2.2 Les objectifs du stage 3
1.3 Le contenu du rapport 3

2 Le Robot Thymio **4**
2.1 Présentation [11] 4
2.2 La philosophie d'un projet collaboratif [10] 4
2.3 Les différents acteurs du projet 5
2.4 Les engagements de la société créatrice 6

3 Les fonctionnalités de Thymio concernant le son **7**
3.1 Les fonctionnalités natives de Thymio 7
3.1.1 L'intensité sonore ambiante 7
3.2 La synthèse sonore de Thymio 7
3.2.1 La génération de sons 7
3.2.2 L'enregistrement et la relecture 8
3.2.3 La lecture de "Samples" 8
3.2.4 La création de son depuis un ordinateur 8
3.2.5 Les sons système 9
3.2.6 La bibliothèque de sons système 9
3.3 Les limites 9

4 L'extension des possibilités sonores de Thymio **11**
4.1 Les outils informatiques utilisés 11
4.1.1 Le langage de programmation Aseba 11

4.1.2 Les outils ASEBA 11
4.2 L'externalisation des événements 12

5 La réalisation de cette extension **13**
5.1 La communication entre Thymio et un ordinateur 13
5.2 L'idée principale 13
5.3 L'écriture dans un fichier texte 14
5.4 La lecture du fichier texte 14

6 Le contenu de l'archive finale **16**
6.1 Le README 16
6.2 Makefile 16
6.3 Le fichier Aseba 16
6.4 Le fichier loop_read 16

7 Bilan **18**
7.1 Apports du stage sur le plan personnel 18
7.2 Conclusion 19
7.3 Perspectives 19

Chapitre 1

Introduction

1.1 Domaine du Stage

1.1.1 Le SCRIME

Ce stage a été effectué au sein du Studio de Création et de Recherche en Informatique et Musiques Expérimentales (ou Électro-acoustiques).[7]
Ce groupement d'intérêt scientifique et artistique est rattaché à l'équipe Image et Son du Laboratoire Bordelais de Recherche en Informatique (LaBRI).
Les champs d'activités du SCRIME (page 1)s'étendent de la recherche scientifique à la création artistique, en passant par la formation, la diffusion des musiques contemporaines et la pédagogie en milieu scolaire et universitaire.
Le SCRIME (page 1) est soutenu par le Ministère de la Culture, le Conseil Régional d'Aquitaine, la Direction Régionale des Affaires Culturelles (DRAC) et le Centre National de la Recherche Scientifique (CNRS).

1.1.2 Les missions du SCRIME

Les différentes missions du SCRIME [7] sont :
- Favoriser la collaboration entre scientifiques et artistes à des fins de recherche et de création
- Favoriser la diffusion internationale des résultats des recherches et des oeuvres artistiques
- Promouvoir les collaborations scientifiques et artistiques avec d'autres centres de recherche et de création en France ou à l'étranger
- Favoriser les actions pédagogiques dans les institutions partenaires ainsi que dans les autres établissements publics d'enseignement et de création et ce à tous les niveaux

1.1.3 L'équipe du SCRIME

L'équipe de ce groupement d'intérêt scientifique et artistique est composée de :
Myriam Desainte-Catherine : Direction scientifique
Jean-Louis Agobet : Direction artistique
Gyorgy Kurtag : Coordination Arts et Sciences
Annick Mersier : Gestionnaire
Christian Faurens : Assistant ingénieur audiovisuel
Pierre Cochard : Réalisateur en Informatique Musicale
Antoine Hubineau : Chargé de communication
Julia Hanadi Al Abed : Régisseur son
Jean-Michel Rivet : Compositeur et Enseignant
Raphaël Marczak : Ingénieur de recherche
Nicolas Vuaille : Développeur
Pierre COCHARD aura été le responsable de ce stage, mais cette responsabilité a été partagée avec Philippe GUILLEM, professeur de musique et enseignant en classe de maternelle.

1.2 Le contexte du stage

1.2.1 Pourquoi ce projet ?

Philippe GUILLEM est maître formateur en musique et enseignant en école maternelle. Son objectif est de montrer à ses élèves ce que le monde moderne leur offre. Cela va des réseaux sociaux à la robotique, en passant par la programmation de manière plus vaste. Seulement une interrogation naissait en même temps que cette idée :

"Comment, chez des élèves de 5 ans, aider à construire une image du Robot et des relations que l'homme peut entretenir avec ces machines ?" [3]

Lors d'une conférence, Philippe GUILLEM a découvert le projet du robot Thymio (page 4). De par sa robustesse, sa simplicité d'accès, d'utilisation et de programmation, ainsi que par sa conception, il était tout à fait adapté aux jeunes enfants et donc à ce projet. Enfin, il restait la question de représenter la relation homme / robot. Pour cela, M. GUILLEM a tenu à faire un lien avec la musique, car c'est ce qu'il y a de plus parlant pour les jeunes auditorats. C'est donc suite à ce croisement d'idées que le SCRIME a été sollicité, en particulier Pierre COCHARD, et tout deux ont travaillé sur le projet de ce stage.

1.2.2 Les objectifs du stage

L'objectif initial de ce stage était de réaliser une interface permettant de piloter le robot Thymio (page 4) avec le logiciel i-score [4] afin d'en permettre l'utilisation en classe avec des élèves de maternelle par monsieur Phillipe GUILLEM.
Dans ce but, ce stage devait permettre l'implémentation d'un système de communication entre Pyo [8] (ou tout autre logiciel adapté) et le logiciel i-score [4].
Or le stage étant très étalé, mais d'une durée effective courte, les objectifs ont été redéfinis. L'objectif du stage est devenu d'écrire un programme permettant de traiter les évènements de Thymio extérieurement au robot, mais en temps réel. En effet, si ce traitement est valide, il devient aisé d'y insérer une transmission en protocole *Open Sound Control* (OSC) [2] par exemple et ainsi de communiquer avec i-score [4]. L'objectif pédagogique était d'obtenir des compétences en robotique et dans le traitement des informations d'un capteur.

1.3 Le contenu du rapport

Afin de mieux comprendre la finalité de ce stage, ce rapport va d'abord permettre de définir ce qu'est Thymio (page 4) et le projet pédagogique et communautaire auquel il appartient. Les fonctionnalités du robot seront ensuite détaillées, principalement ses compétences relatives au son et à la musique. Ensuite, seront expliquées les méthodes envisagées et sélectionnées afin d'étendre les compétences de base de Thymio (page 4). La partie suivante concernera l'archive finale de travail, ainsi que son contenu. Enfin, une conclusion détaillera les résultats obtenus, les apports réalisés et les améliorations possibles.

Chapitre 2

Le Robot Thymio

2.1 Présentation [11]

Thymio (page 4) , conçu par l'École polytechnique fédérale de Lausanne, est un petit robot dédié à la pédagogie. Ses deux roues lui permettent déplacements et manoeuvres, tandis qu'il embarque des boutons ainsi que de nombreux capteurs infrarouges pour détécter des obstacles ou des surfaces, un micro, un haut parleur, un accéléromètre et même un thermomètre. Son accessibilité est due à la mise à disposition de ses propres interfaces de programmation. Il y en a en effet plusieurs, à commencer par l'interface de programmation texte, ASEBA STUDIO qui permet d'exploiter entièrement le langage ASEBA (page 11) . Une adaptation de BLOCKLY [9] est aussi disponible, qui permet d'aborder la programmation texte de manière simplifiée, et enfin il y a VPL [12] qui permet de programmer Thymio visuellement grâce à des vignettes. C'est grâce à ces interfaces que ce robot se trouve adpaté aux plus jeunes.

Pour résumer, il s'agit d'un robot complet, qui embarque un grand nombre de capteurs, et accessible à tous, aux plus jeunes comme aux programmeurs confirmés, c'est pourquoi ce robot a été choisi pour ce projet de stage.

2.2 La philosophie d'un projet collaboratif [10]

Le but du projet Thymio (page 4) était de permettre à un large public, et en particulier aux enfants, la découverte et l'apprentissage de la programmation, de la robotique, de l'ingénierie et des technologies numériques. Pour permettre cela, il a fallu regrouper plusieurs corps de métiers et des connaissances poussées dans différents domaines. C'est pourquoi Thymio (page 4) est un projet communautaire, qui implique plusieurs entités. Il s'agit donc d'un modèle basé sur le partage libre des connaissances, c'est à dire que tout le matériel, le logiciel, et la documen-

tation sont développés par les partenaires sous licence libre. Ainsi, les partenaires s'engagent à fournir l'information de conception gratuitement, de manière à ce que chacun puisse comprendre, utiliser, ou modifier cette information, à condition de distribuer à son tour les résultats de son travail. Il est donc important que ce stage s'inscrive dans cette dynamique, et que le travail ainsi que les résultats soient partagés à la communauté.

2.3 Les différents acteurs du projet

Comme expliqué précédemment, plusieurs catégories d'acteurs se sont alliés dans ce projet :

Les institutions de recherche : qui permettent de diffuser le travail et de l'amener auprès du public. Plus la base d'utilisateurs grandit, meilleures sont les possibilités d'évaluation et d'amélioration. Les résultats de la recherche sont ainsi mis rapidement en application.

Les entités de production : qui donnent accès à des produits proches de l'état de l'art qui n'ont plus qu'à être industrialisés. Les frais liés à des licences sont réduits et les produits sont validés scientifiquement. Les entités de production distribuent toute l'information (schémas, plans, pdf des exercices, etc.) gratuitement et commercialisent les supports physiques (robots, accessoires, livrets imprimés, etc.).

Les contributeurs permettent de participer à un grand projet selon leurs moyens et de choisir leur implication, de proposer des idées, de recevoir de l'aide et de l'information et de voir leur travail diffusé.

Les utilisateurs leur apport se perçoit dans du matériel moins cher, beaucoup de ressources gratuites, une transparence dans le projet, et une plus grande pérennité du produit.

Parmis les principaux acteurs, on pourra citer en particulier :

EPFL - École Polytechnique Fédérale de Lausanne Initiatrice du projet.

Association Mobsya Association responsable de la production et de la distribution du robot.

ETH - Eidgenössiche Technische Hochschule Zürich Le laboratoire de Systèmes Autonomes et le centre de technologies de jeu de l'ETH Zürich qui ont contribué à ASEBA et à VPL.

NCCR Robotics Aide financière et aide à la validation scientifique.

la Loterie Romande, Suisse

et le Fond National Suisse de la Recherche Scientifique Respectivement pour l'aide financière pour les premières productions et l'aide financière pour la mise en place de kits éducatifs

INRIA - L'Institut National de Recherche en Informatique et en Automatique Développeur principal de l'adaptation de SCRATCH au Thymio, qui donnera BLOCKY, et développement du matériel éducatif pour les enseignants, notamment avec le projet "Dessine-moi un robot".

L'INRIA aura grandement aidé dans ce projet, grâce notamment au contact de Didier ROY ; docteur en informatique cognitive, lié de près au projet Thymio (page 4), qui aura aidé dans les réfléxions, mais aussi prêté un Thymio de test pendant plusieurs mois avant que le SCRIME n'ai pu en faire l'acquisition.

2.4 Les engagements de la société créatrice

Pour suivre cette philosophie basée sur la collaboration et les sources ouvertes (*open source*), la société créatrice a choisi les modes de diffusion suivants :

- pour le logiciel, le code source est en *open source*, sous licence *LGPL* .
- pour le matériel, sont distribués tous les documents de conception, y compris schémas électroniques, plans des pièces, etc. en open hardware sous une licence *CC-BY-SA 3.0* .
- pour la documentation, les exercices, les cursus, sont distribués également tous les documents en format numérique sous une licence *CC-BY-SA 3.0* .

Afin de permettre une production et un distribution à grande échelle, une association à but non lucratif a été crée, Mobsya. L'association assure la production du matériel. Pour le logiciel, elle contribue à la maintenance et au développement en collaboration avec les autres partenaires. L'association gère aussi l'infrastructure qui permet de créer une communauté d'utilisateurs, qui peuvent s'entraider grâce à des listes de discussion et un site de partage d'information qui contient la documentation, des exemples de réalisation, du support technique, etc. Cette infrastructure a également apporté son aide durant ce stage.

Chapitre 3

Les fonctionnalités de Thymio concernant le son

3.1 Les fonctionnalités natives de Thymio

Le robot Thymio (page4) est déjà conçu pour une interaction basée sur le son. C'est possible entre autre grâce à la présence d'un micro et d'un haut-parleur, mais aussi grâce au slot micro-SD. Seulement les possibilités sont limitées.

3.1.1 L'intensité sonore ambiante

Le robot peut détecter lorsque le bruit ambiant est au-dessus d'une intensité donnée. C'est possible, grâce à la variable *mic.intensity* qui donne l'intensité du microphone embarqué ou encore grâce à la variable *mic.threshold* qui contient la limite de l'intensité pour l'événement. Un évènement est alors généré si *mic.intensity* est au-dessus de *mic.threshold*.

3.2 La synthèse sonore de Thymio

3.2.1 La génération de sons

Le langage Aseba (page 11) possède une fonction native *sound.freq* qui joue une fréquence, spécifiée en Hertz (Hz), pour une certaine durée, spécifié dans 1/60 s. Une durée égale à 0 joue le son à l'infini et une durée de -1 arrête le son.
La génération de sons de synthèse fonctionne par ré-échantillonnage d'une onde primaire. Par défaut, l'onde est triangulaire, mais il est possible de la définir à l'aide de la fonction native *sound.wave*. Cette fonction prend en entrée un tableau de 142 échantillons, avec des valeurs de -128 à 127. Ce tableau doit représenter

une onde tonique de la fréquence spécifiée dans *sound.freq*.
Comme Thymio joue des sons à 7812,5 Hz, ce tableau est joué entièrement à la fréquence de 7812.5/142 = 55 Hz. Jouer un son de fréquences plus élevées fait sauter des échantillons dans le tableau.

3.2.2 L'enregistrement et la relecture

Il est aussi possible d'enregistrer des sons en utilisant la fonction native *sound.record* qui prend en paramètre un numéro d'enregistrement de 0 à 32767. Le fichier est stocké sur la carte micro-SD sous le nom *Rx.wav* où x est le paramètre passé à la fonction sound.record. Pour arrêter un enregistrement, il faut appeler la fonction *sound.record* avec la valeur -1.
On peut également rejouer un son enregistré en utilisant la fonction native *sound.replay* . Cette fonction prend en paramètre un nombre de 0 à 32767 et rejoue le fichier *Rx.wav* de la carte SD où x est le paramètre passé à la fonction *sound.replay* . Pour arrêter une lecture, il faut appeler la fonction sound.replay avec une valeur de -1.

3.2.3 La lecture de "Samples"

Il est également possible de jouer d'autres sons que les sons simples générés par synthèse ou les sons système. En utilisant une carte micro-SD formatée FAT, il est possible d'enregistrer et de jouer des "samples", des morceaux de musique pré-enregistrés. Les fichiers doivent cependant être stockés sur la carte micro-SD au format wave, 8-bit non-signé, 8 kHz. Lorsque le robot finit la lecture d'un son demandée par ASEBA, il déclenche l'événement *sound.finished*. Il ne se déclenche pas d'événement si la lecture est interrompue ou si un nouveau son est joué.

3.2.4 La création de son depuis un ordinateur

Voici la démarche, selon les créateurs de Thymio (page 4), pour créer, à l'aide du logiciel Audacity, un son compatible avec Thymio II qui puisse être joué :

- Une fois démarré audacity, changer le *project rate* situé en bas à gauche de 44100 Hz (défaut) à 8000 Hz.
- Enregistrer le son avec la touche *record* rouge en haut à gauche. Le curseur doit avancer et la forme d'onde se visualiser. Stopper une fois fini avec le bouton *stop*.
- Le son doit être en mono (*Piste -> Piste stéréo vers mono*)
- Aller sur le menu *File sous Export...*

- Donner un nom de ficher.
- Choisir comme format : *other uncompressed files.*
- Sous "options" choisir un *header WAV* (Microsoft) et comme *Encoding* choisir *Unsigned 8 bit PCM.*
- Sauver ou copier le ficher sur une carte micro-SD.

3.2.5 Les sons système

Les sons système de Thymio (page 4) peuvent être utilisés à l'aide de la fonction native *sound.system* , qui prend un nombre de 0 à 32767 en tant que paramètre. Certains sons sont disponibles dans le *firmware* (voir ci-dessous), mais on peut remplacer ces sons et en ajouter de nouveaux en utilisant la carte micro-SD. Pour arrêter la lecture d'un son, il faut rappeler la fonction *sound.system* avec la valeur -1.

3.2.6 La bibliothèque de sons système

Les sons suivant sont disponibles :
paramètre description
-1 arrête de jouer un son
0 son de démarrage
1 son d'arrêt (son non reconfiguration)
2 son des boutons fléchés
3 son du bouton central
4 son de chute libre (peureux)
5 son de choc
6 son de suivi d'objet
7 son de détection d'objet pour suivi
Pour ne plus entendre de sons système, Il existe une archive à décompresser sur une carte micro-SD qui contient des pistes muettes, c'est à dire qu'elles seront jouées mais ne produiront pas de son.

3.3 Les limites

Cette gestion du son de par les fonctions natives de Thymio (page 4) est assez limitée. En effet, il est seulement possible de lire soit des sons système, soit des sons pré-enregistrés. Ces sons doivent être échantillonés a 8kHz, ce qui impacte fortement leur qualité. Quant à la synthèse, il faut se contenter de jouer des fréquences en utilisant des ondes primaires, comme indiqué précemment. C'est pour ces raisons que l'objectif de ce stage sera d'établir un protocole de communication

entre Thymio (page 4) et l'ordinateur de manière à ce que ce soit l'ordinateur qui synthétise le son en fonction des informations communiquées par le robot.

Chapitre 4

L'extension des possibilités sonores de Thymio

4.1 Les outils informatiques utilisés

La réalisation de ce stage aura nécessité la combinaison de plusieurs outils, tels que le langage Aseba (page 11), le langage C, l'hébergement de projet sur Github, la programmation système ou encore la recherche sur forums et canaux de discussion.

4.1.1 Le langage de programmation Aseba

Le langage Aseba [6] est un langage spécialement adapté pour la programmation de comportements pour robots à micro-contrôleurs. Ce langage *open source* est une architecture de contrôle temps réel et distribué de robot mobile, basée sur la programmation évènementielle. Aseba dispose de son propre environnement de développement (Aseba Studio) qui est totalement intégré au robot Thymio (page 4). C'est pour cela que ce langage a été utilisé lors de ce stage. Sémantiquement, il s'agit d'un langage de programmation impératif avec un seul type simple (nombres entiers signés sur 16 bits) et des tableaux.

4.1.2 Les outils ASEBA

Mais Aseba (page 11) c'est aussi un ensemble de programme et d'outils qui peuvent permettre d'aider la programmation de robots. En témoigne l'installation des programmes tels que *asebadump*, *asebaswitch*, *asebaexec*, *asebaeventlogger*, etc... Certain seront étudiés, comme *asebaexec* ou *asebadump*, d'autre serviront directement lors de la réalisation du travail comme *asebaswitch*.

4.2 L'externalisation des événements

Le stage s'étalant sur une durée effective relativement courte (4 semaines de travail), il aura été principalement articulé autour de la possibilité d'externaliser les évènements générés par le robot lors de la stimulation de ses capteurs. Il est nécessaire de pouvoir traiter les données des capteurs depuis l'ordinateur récepteur, et de cette manière, une synthèse sonore précise et de haute qualité peut être calculée.

Chapitre 5

La réalisation de cette extension

5.1 La communication entre Thymio et un ordinateur

Au cours de ce stage, deux générations de Thymio (page 4) auront été étudiées, le Thymio II et le Thymio II Wireless. La seule différence se fait d'un point de vue matériel, puisque que le Thymio II Wireless permet de s'affranchir du câble USB qui le relie à l'ordinateur. Tout l'intérêt de cet investissement réside dans le fait de pouvoir maintenir la communication antre l'ordinateur et le robot, peu importe la distance, sa place dans le parcours et la taille de celui-ci. En revanche, d'un point de vue logiciel, il n'y à pas de différence, l'ordinateur capte les valeurs des capteurs de la même manière, que ce soit par USB ou par radiofréquence.

5.2 L'idée principale

Premièrement, il a fallu mettre en place un code Aseba (page 11) qui génère des évènements lorsque cela est souhaité, autrement dit à chaque stimulation de capteurs. Grâce à l'étude de la documentation de Aseba (page 11), l'idée d'utiliser la fonction *emit* est apparue assez naturellement. Elle est définie ainsi :

- « Un script peut envoyer des événements en utilisant le mot clé emit, suivi par le nom de l'événement ainsi que du nom de la variable à envoyer. [...] Les événements permettent ainsi à un script de déclencher l'exécution de code sur un autre nœud ou de communiquer avec un programme externe. »

Suite à l'écriture de ce programme, le robot envoie bien des évènements. Ceux-là sont visibles dans la fenêtre dédiée d'Aseba Studio. Il était alors possible de lancer une partie du code à exécuter sur le Thymio (page 4) en fonction de l'évènement envoyé, mais le problème n'était pas résolu, cela ne suffisait pas à externaliser

complètement les évènements.

Après étude du code source d'Aseba [5] et de celui des outils qui lui sont relatifs, le manque de solutions aura nécessité des recherches au sein de la "communauté Thymio" (page 4). C'est en cherchant de l'aide sur le forum principal d'Aseba que l'idée d'inscrire les évènements dans un fichier texte est apparue.[1]
En effet, il est possible de faire en sorte que chaque évènement envoyé ainsi que les données qui s'y rapportent soient inscrits dans un fichier texte en temps réel. Une fois cette méthode lancée, la lecture de ce fichier permet d'analyser les données des évènements et de les traiter comme on le souhaite, ce qui est l'objectif principal de ce stage.

5.3 L'écriture dans un fichier texte

Il existe donc des commandes qui permettent l'écriture dans un fichier texte de la totalité des évènements envoyés. Pour cela, il faut se connecter au port sur lequel est connecté le robot Thymio afin de lire les données envoyées grâce à la commande :

```
~/asebaswitch "ser:name=Thymio"\\
```

Il faut ensuite indiquer par une commande que l'on souhaite enregistrer la totalité des données passantes par ce port dans un fichier avec la ligne :

```
~/asebarec "tcp:localhost;port=33333" > <nom_du_fichier>
```

A la suite de ça, on peut executer, sur le robot, le code écrit dans aseba et voir s'inscrire en temps réel les données des évènements dans le fichier passé plus tôt en paramètre.

5.4 La lecture du fichier texte

Une fois le résultat souhaité obtenu en écriture, il reste à lire le fichier. Pour cela, il y a le programme *loop_ read*. Ce programme prend deux paramètres, le nom du fichier dans lequel les évènements sont écrits, et le nom d'un second fichier, au choix de l'utilisateur dans lequel chaque donnée des évènements traités sont écrites proprement avec leur signification. Cela permet d'avoir une trace de fonctionnement du programme, des évènements traités et donc de leur nombre.
La principale difficulté aura été de faire en sorte que le programme de lecture fonctionne en même temps que l'écriture, et que tous les évènements soient traités en temps réel.
Pour cela, le programme *loop_ read* scanne le fichier des évènements ligne par ligne

(on sait qu'a une ligne correspond un évènement). Si le numéro d'identifiant de l'évènement est différent de celui traité précédemment, alors on le traite. Sinon, on scanne la ligne suivante. Cet algorithme permet de s'assurer que la totalité des évènements déjà ecrits sont traités, et qu'il n'y ait pas de répétition d'évènements. La dernière modification consiste à répéter cet algorithme tant que l'évènement final n'est pas détécté. L'évènement final est celui identifié par la constante NBR_STOP_EVENT en début du fichier. Il s'agit d'un évènement à part qui va déterminer que l'on termine le programme à son émission.

Chapitre 6

Le contenu de l'archive finale

6.1 Le README

L'archive finale contient différents fichiers qui sont nécessaires pour l'éxécution de ce code. Elle contient en particulier un fichier "README.md" qui détaille le contenu des autres fichiers de l'archive mais aussi qui sert de guide pas-à-pas au lancement du programme de traitement des évènements.

6.2 Makefile

Le makefile présent est aussi simple qu'il est efficace. Il permet de compiler le programme *loop_ read.*

6.3 Le fichier Aseba

Ce fichier (*prox-event-sender.aesl*) contient le code d'exemple qui sera à exécuter sur le robot. Il ne prend pas en charge de déplacements, seulement la gestion des évènement en fonction de la stimulation des capteurs de proximité horizontale. Il est extensible à n'importe quel autre capteur, et intégrable à d'autre codes Aseba.

6.4 Le fichier loop_read

Il s'agit du programme principal. Il est nécessaire d'ajuster la constante NBR_STOP_EVENT en fonction de l'identifiant donné à l'évènement final qui est déterminé dans le code du fichier Aseba. Les autres modifications à apporter concerneront le traitement plus fin des données des évènements (la transmission

en protocole de communication avec un séquenceur par exemple). Ce traitement sera à insérer dans ce programme à l'endroit indiqué par les commentaires ou par le README.

Chapitre 7

Bilan

7.1 Apports du stage sur le plan personnel

Ce stage aura été très particulier. En effet, il se sera étalé tout au long de l'année scolaire de début Novembre à début fin Mai, tandis que le temps de travail effectif aura été très court, puisque cela correspond à environ 4 semaines. Cette organisation très particulière aura été difficile à gérer. En effet, il aura été compliqué de concilier convenablement les cours, les projets et les examens en plus du stage. Cependant, cela m'aura appris à m'organiser à long terme, car c'est ainsi que ce stage a pu aboutir. Une autre leçon principale aura été de me rendre compte à quel point il était important d'avancer efficacement car en me présentant seulement une demi-journée par semaine, il fallait optimiser mon avancée sur cette demi-journée. Ensuite, le fait de travailler sur le robot Thymio, et au coeur du projet pédagogique qui l'entour, m'aura appris à travailler sur un projet libre. C'est à dire que cela m'a appris à contacter des personnes afin de demander de l'aide, d'utiliser les réseaux sociaux comme liste de contacts, et de rendre mon code accessible au plus grand nombre. Cela aura eu un impact direct sur mon travail puisque c'est pour ces raisons que j'ai choisi de commenter entièrement le code du programme de lecture de fichier en anglais, et que j'ai passé un certain temps à rédiger le fichier "README", indispensable si quelqu'un souhaite récupérer mon travail et l'adapter au sien.
Techniquement, ce stage m'a aussi permis de mettre en pratique des connaissances acquises au cours de l'année universitaire, comme les cours de programmation système, qui m'auront inspiré pour la gestion de fichier (ouverture, lecture et écriture). Mais j'ai également pu apprendre beaucoup en étudiant au plus près un nouveau langage (Aseba) ainsi que son code source en anglais. Le fait de travailler avec un robot aura aussi été quelque chose de nouveau, et aura contribuer à élargir le champs des possibilités d'application de l'informatique que je connaissais déjà.

7.2 Conclusion

Il aura été difficile de rendre le robot Thymio "ordonnant", tandis qu'il est entièrement conçu pour être exécutant. Cepandant le programme écrit lors de ce stage semble fonctionner, et serait adaptable à n'importe quel projet. Il suffirait pour cela de bien consulter le fichier README ou bien se référer aux parties correspondantes dans ce rapport. En revanche, cette volontée d'adaptabilité a impacté le résultat, puisque l'archive contenant les fichiers n'est pas totalement utilisable en soi. En d'autres termes, les fichiers fournis permettent de tester le programme, et d'avoir une visualisation des évènements et de leur traitement, mais actuellement le traitement ne consiste qu'à la réécriture de ces évènements dans un nouveau fichier. Cela n'a pas d'autre utilité en soi que la visualisation. C'est pour cela qu'il n'existe pas encore de démonstration d'applications.

7.3 Perspectives

Les perspectives d'évolution de ce projet sont nombreuses. Premièrement, il faudra insérer au programme une partie de code qui permettra la communication avec un logiciel séquenceur. Le première idée serait d'utiliser une bibliothèque C pour la gestion de paquets *Open Sound Control* (OSC) qui est un protocole de communication entre ordinateurs, synthétiseurs ou robots qui est plus complet que la norme MIDI.
Ensuite il faudra déterminer quelles seront les instructions à transmettre et comment les traduire en musique au sein du séquenceur audio. Une fois cela effectué, il sera possible de faire une première démonstration. Les autres perspectives concernent la diffusion et la mise à disposition de ce code. Comme détaillé précemment, ce stage s'inscrit au coeur d'un projet communautaire en pleine évolution. Il est alors important de rendre ce travail accessible, et il sera certainement intéressant de suivre son évolution au sein des différents autres projets auquels il est susceptible de se greffer.

Bibliographie

[1] Michael Bonani. "record sensor data continuously to external file", https ://www.thymio.org/forum/t-706858/record-sensor-data-continuously-to-external-file. ., 2016.

[2] Open Sound Control. "introduction to open sound control", http ://opensoundcontrol.org/introduction-osc. ., 2016.

[3] Philippe GUILLEM. "fragments de classes - thymio en musique / arts et sciences à l'école maternelle", http ://fragmentsdeclasse.blogspot.fr/2016/02/2-thymio-mais-quils-sont-nombreux.html. ., 19/02/2016.

[4] i score. "i-score", http ://i-score.org/. ., 2016.

[5] Stephane MAGNENAT. "code source de aseba", https ://github.com/aseba-community/aseba. ., 2016.

[6] Stéphane MAGNENAT. "aseba - manuel de l'utilisateur", http ://mobots.epfl.ch/data/aseba/aseba-user-manual-fr.pdf. ., 2016.

[7] SCRIME. "présentation du studio de création et de recherche en informatique et musiques expérimentales", http ://scrime.labri.fr/le-scrime-2/. ., 2016.

[8] Ajax Sound Studio. "pyo - dedicated python module for digital signal processing", http ://ajaxsoundstudio.com/software/pyo/. ., 2016.

[9] Thymio. "blockly - programmation par blocks", https ://www.thymio.org/fr :blocklyprogramming. ., 2016.

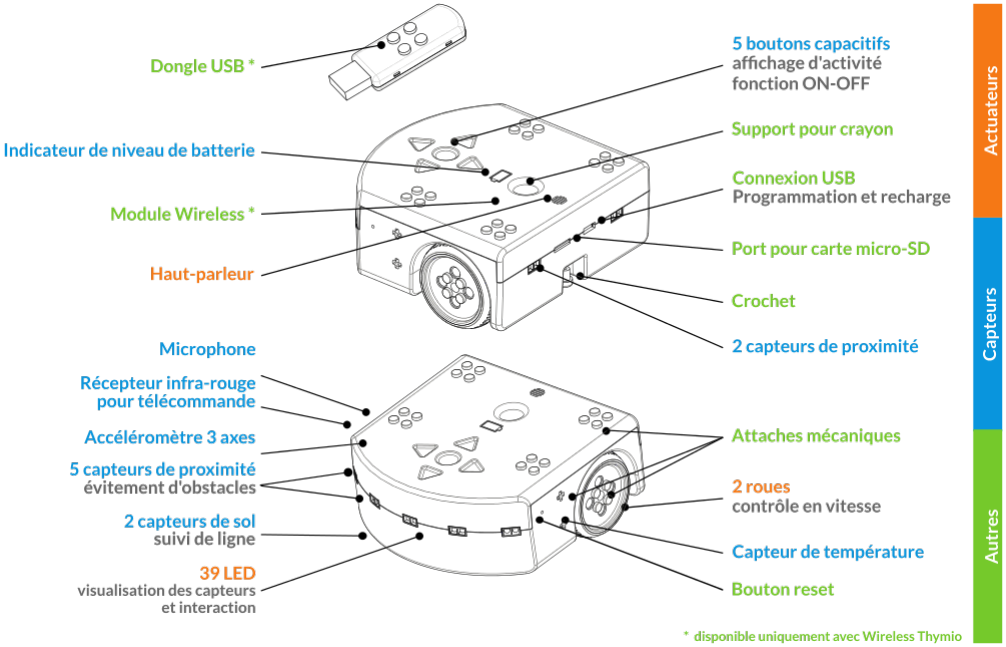
[10] Thymio. "philosophie du projet thymio", https ://www.thymio.org/fr :thymiophilosophy. ., 2016.

[11] Thymio. "présentation de thymio", https ://www.thymio.org/fr :thymio. ., 2016.

[12] Thymio. "vpl - programmation visuelle", https ://www.thymio.org/fr :visual-programming. ., 2016.

Annexes

Caractéristiques du robot



Contenu de l'archive

Readme

```
# StageSCRIME
   ----------------------
  /                       \
 /***** Introduction *****\
/_________________________\

Objectif : externaliser les informations captées par les capteurs du Thymio

Solution -> Inscription des évènements en direct dans un fichier texte,
qui est lu en temps réel afin de traiter chaque évènement.



   ---------------------------
  /                           \
 /***** Partie I - Aseba *****\
/_____________________________\

Il est nécessaire d'adapter le code édité dans ASEBA STUDIO pour l'utilisation.


Exemple, voir : "prox-event-sender.aesl"
----------------------------------------

Dans cet exemple, dès qu'un capteur de proximité horizontale est stimulé
(sa valeur dépasse 100), on envoi un évènement grâce à la fonction "emit".
(voir commentaires code)

Chaque évènement est un évènement global qu'il faut créer manuellement
 dans l'interface Aseba.
```

(Événements globaux -> "+" -> <nom_evenement> -> identique à celui utilisé dans le code
   -> <Nombre d'arguments -> Correspond au nombre
 d'arguments de l'évènement à envoyer.
Ici c'est prox.horizontal qui donne les valeurs des capteurs de proximité, qui sont au nombre de 7; il y a donc 7 arguments.

Cet exemple est adaptable à tout code Aseba pilotant le thymio, il suffit d'ajouter des "emit" dès qu'on le souhaite, et que l'on ajuste les évènements globaux.

** /!\ ** Garder un évènement final! Ici "button.center == 1".
Cet évènement est nécessaire pour stoper le script de traitement.
(Voir "Partie II")
Conseil : de manière à le repérer, il est préférable de le placer en dernier évènement global.
Son numéro d'évènement sera alors le nombre d'évènements différents, plus facile à identifer lors de la lecture.
(Ne pas le placer en premier car tableau initialisé à 0!
<=> Si tab[nbr_event] == 0 -> stoper le prgm <=> le traitement n'aura jamais lieu.

```
   ---------------------------------
  /                                 \
 /***** Partie II - Traitement *****\
/-----------------------------------\
```

Voir "loop_read.c"

Constante NBR_STOP_EVENT :: Il s'agit du nombre de l'évènement final
(Voir "Partie I -- ** /!\ ** Garder un évènement final!")
Il faut ajuster cette constante en fonction du nombre correspondant à l'évènement final.

Pour cet exemple, il y a 6 évènements globaux (1 par capteur de proximité horizontale stimulé), + l'évènement final de pression sur le boutton central ce qui donne "7".

Le reste du code est expliqué par commentaires; à savoir :

1- Ouverture des fichiers nécessaires;
Créer les fichiers passés en paramètres s'ils n'existent pas,
les effacent sinon.
(/!\ un fichier existant sera effacé sans demande de confirmation,
pensez a sauvegarder, ou utiliser un autre nom si vous souhaitez conserver
une acquisition antérieure)

2- Déclaration du tableau ("array") de taille 11; du compteur ("cpt") du nombre
d'évènements traités; et de la date du dernier évènement traité ("event_time").

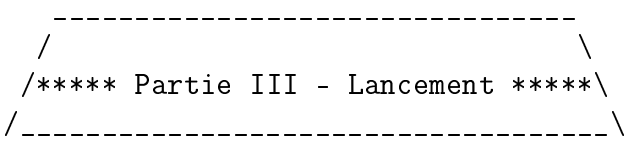
3- Traitement;
Le traitement se fait en boucle grâce au "while(array[2] != NBR_STOP_EVENT)".
On quitte cette boucle à la détection de l'évènement dit final
(identifié par la constante NBR_STOP_EVENT).

Le programme lit le fichier ligne par ligne, et stocke les valeurs
dans le tableau.
La condition de la suite du traitement ("if(array[0] != event_time)")
est nécessaire pour éviter de traiter plusieurs fois le même évènement.

C'est donc à la suite de cette condition que le code de traitement
que l'on souhaite appliquer doit être inséré. On utilisera les données
telles qu'elles sont stockées dans le tableau.

A titre d'exemple, le reste du programme réécrit les données relatives aux
évènements dans un fichier de sortie. Par gain de temps et/ou de place,
on peut éventuellement commenter cette partie".

4- Fermeture des fichers, fin du programme.



Routine de lancement du programme

```
--------------------------------

1- Brancher Thymio ou bien sa clé puis l'allumer

2- Entrer la commande qui permet de se connecter en lecture au port sur lequel
est connecté Thymio. Appuyer une seconde fois sur entrée pour récupérer
la main dans le terminal.

~/asebaswitch "ser:name=Thymio" &

/!\ Cette commande ne sera entrée qu'une seule fois par session de travail!

3- Entrer la commande qui permet d'enregistrer les évènements générés
dans un fichier texte.

~/asebarec "tcp:localhost;port=33333" > <nom_du_fichier_log> &

Puis lancer Aseba Studio et ouvrir le fichier contenant le code que l'on souhaite
executer sur Thymio. (Voir l'exemple "prox-event-sender.aesl")

~/asebastudio &

fichier -> ouvrir -> <fichier_code_thymio>

4- Compiler puis lancer enfin le programme de traitement du texte. Passer en argume
 les noms des fichiers souhaités. S'ils n'existent pas, le programme les créera.

~/make clean
~/make
~/./loop_read <nom_du_fichier_log> <nom_du_fichier_de_sortie>

Dans asebastudio : "charger" puis "executer"

Tout devrait alors fonctionner. Thymio devrait inscrire ces évènements dans
<nom_du_fichier_log> alors que le programme "loop_read"_ lit ce fichier pour
traiter les données et les inscrit dans <nom_du_fichier_de_sortie>.
Le programme fonctionnera jusqu'à ce que l'on active l'"évènement final"
(le bouton central dans les fichiers exemples).

/!\ Pour relancer le programme a nouveau, il faut remonter à l'étape 3.
```

Code source du script

```
#include <unistd.h>
#include <stdlib.h>
#include <stdio.h>
int NBR_STOP_EVENT = 7;
///// The ID of the event used to stoped the prgm
///// according to aseba exemple file, event 7 = button.center pressed

int main(int argc, char** argv)
{

//~~~~~~~~~~ USAGE : error arg number ~~~~~~~~~~//
                if(argc != 3 )
        {
                printf("usage :: ./read <file_in> <file_out>\n");
                exit(EXIT_FAILURE);
        }
//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~//

//~~~~~~~~~~~ Open INPUT file (arg1) ~~~~~~~~~~//
        FILE* file_in = NULL;
        file_in = fopen(argv[1], "w+");
        if(file_in == NULL)
        {
                printf("fatal error :: cannot open input file\n");
                exit(EXIT_FAILURE);
        }
//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~//

//~~~~~~~~~~ Open OUTPUT file (arg2) ~~~~~~~~~~//
        FILE* file_out = NULL;
        file_out = fopen(argv[2], "w+");
        if(file_out == NULL)
        {
                printf("fatal error :: cannot open output file\n");
                return EXIT_FAILURE;
        }
//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~//
```

```
//~~~~~~~~~~~~ Reading and treating ~~~~~~~~~~~~//

        double event_time = 0;
///// Timestamp of last treated event
        int cpt = 0;
///// Number of treated events
        double array[11] = {0};
///// creating the array with event description

        while(array[2] != NBR_STOP_EVENT)
///// tab[2] refere to event nbr ;;
///// according to aseba exemple file
///// <=> while button center of Thymio not pressed
        {
                fscanf(file_in,
 "%lf %lf %lf %lf %lf %lf %lf %lf %lf %lf %lf",
&array[0], &array[1], &array[2], &array[3], &array[4], &array[5],
&array[6], &array[7], &array[8], &array[9], &array[10]);

                if(array[0] != event_time)
///// if Timestamp of event currently scanned is
/////  != than Timestamp of previous event ;;

///// prevent from infinite loop treatment on the current event ;;
                {
                        for(int i = 0; i <= 10; i++)
                        {
                                printf("array[%d] = %lf ",i,array[i]);
//// debug help ;; disp the array for each new event treated
                                printf("\n\n");
                        }
                        cpt++;
///// Updating number of events treated
                        fprintf(file_out,"EVENT %d\n
 Date = %lf ;
NbThymio = %lf ;
 NbEvent = %lf ;
 NbArgEvent = %lf\n",
cpt,array[0],array[1],array[2],array[3]);
///// writing in the output file
```

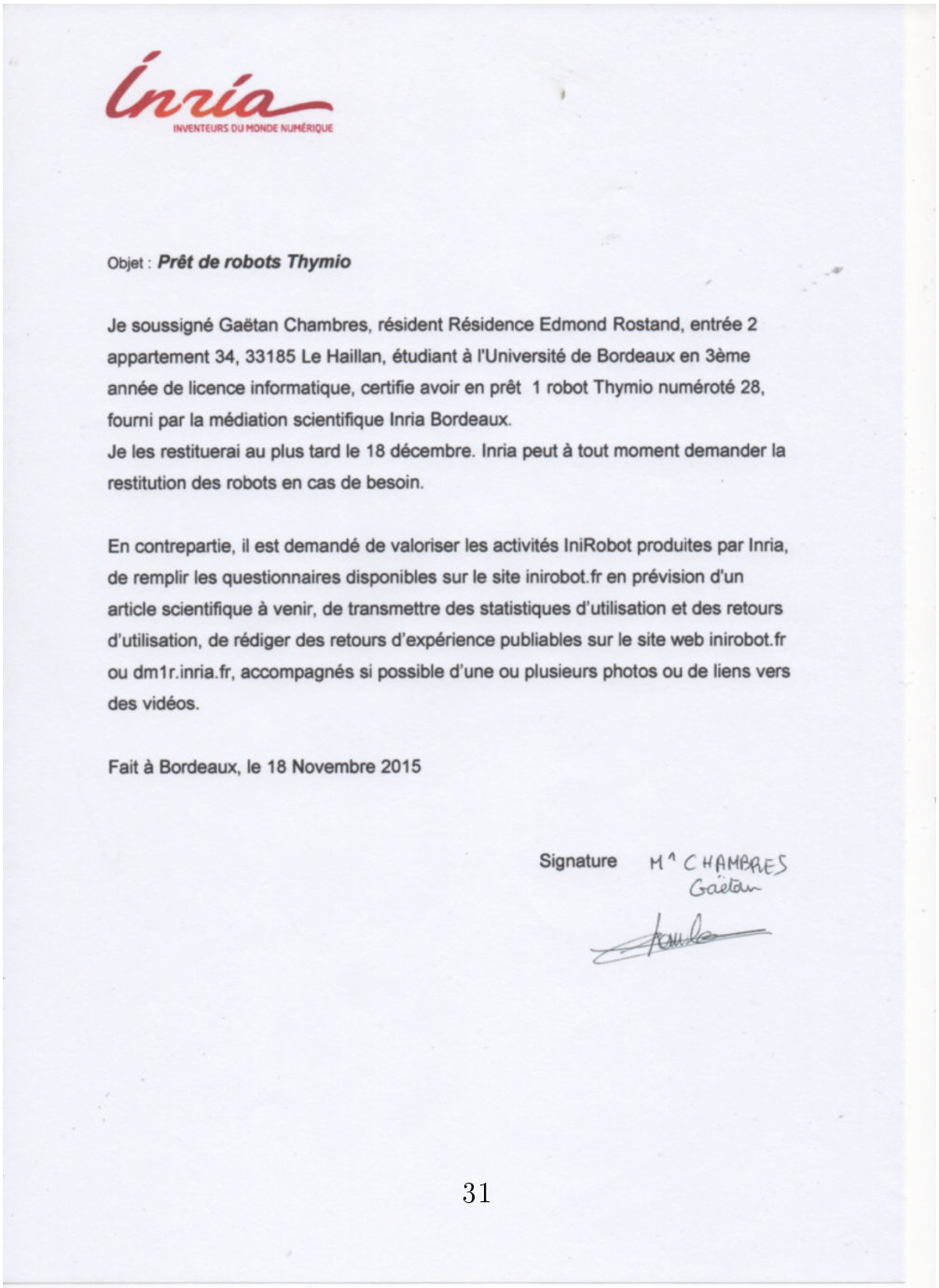
```
                        fprintf(file_out,
"Sensor0 = %lf;
 Sensor1 = %lf;
Sensor2 = %lf;
Sensor3 = %lf;
Sensor4 = %lf;
Sensor5 = %lf;
Sensor6 = %lf;\n",
array[4],
array[5],
array[6],
array[7],
array[8],
array[9],
array[10]);
///// writing in the output file
                        fprintf(file_out,"\n\n");
                        event_time = array[0];
///// Update Timestamp ;; current event is treated so
///// it will become the ancient one for next iteration

                }

        }
//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~//

//~~~~~~~~~~~~ Closing files & Ending ~~~~~~~~~~~~//
        fclose(file_in);
        fclose(file_out);
        return EXIT_SUCCESS;
//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~//
}
```

Pret du Thymio de l'INRIA



Inria
INVENTEURS DU MONDE NUMÉRIQUE

Objet : ***Prêt de robots Thymio***

Je soussigné Gaëtan Chambres, résident Résidence Edmond Rostand, entrée 2 appartement 34, 33185 Le Haillan, étudiant à l'Université de Bordeaux en 3ème année de licence informatique, certifie avoir en prêt 1 robot Thymio numéroté 28, fourni par la médiation scientifique Inria Bordeaux.
Je les restituerai au plus tard le 18 décembre. Inria peut à tout moment demander la restitution des robots en cas de besoin.

En contrepartie, il est demandé de valoriser les activités IniRobot produites par Inria, de remplir les questionnaires disponibles sur le site inirobot.fr en prévision d'un article scientifique à venir, de transmettre des statistiques d'utilisation et des retours d'utilisation, de rédiger des retours d'expérience publiables sur le site web inirobot.fr ou dm1r.inria.fr, accompagnés si possible d'une ou plusieurs photos ou de liens vers des vidéos.

Fait à Bordeaux, le 18 Novembre 2015

Signature M^ CHAMBRES Gaëtan