

Dialogue robotique avec le robot Thymio II



Auteur: Nicolas Mojon

Répondant : M. Jonathan Melly

Expert : M.Claude Rochat et M.Romain Marion

Ecole Technique des Métiers de Lausanne, Mai 2014

Table des matières

1	Analyse préliminaire	4
1.1	Introduction.....	4
1.2	Objectifs (repris du cahier des charges)	4
1.2.1	Objectif SMART	4
1.2.2	Si le temps le permet	6
1.2.3	Fonctionnalités requises (du point de vue de l'utilisateur)	6
1.3	Planification initiale	6
2	Analyse.....	6
2.1	Cahier des charges détaillé	6
2.2	Stratégie de test.....	7
2.3	Budget	7
2.4	Etude de faisabilité	8
2.5	Compléments d'analyse éventuels	8
2.6	Planification	9
2.7	Historique	11
3	Conception	11
3.1	Matériel Hardware	11
3.2	Thymio II	11
3.3	Contrainte matérielle.....	12
3.3.1	Mémoire.....	12
3.3.2	Approximation des composants.....	12
3.4	Contrainte logicielles.....	12
3.5	Logiciels.....	13
3.5.1	Aseba studio (v 1.3.1).....	13
3.5.2	Audacity (v 2.0.5).....	13
3.5.3	Graphity	13
3.5.4	Github	14
3.5.5	Redmine	14
3.6	Gestion des variables	14
3.7	Conception générale du projet.....	14
3.7.1	Les boutons	14
3.7.2	Diagramme d'état	15
3.8	Les différents Etats (a revoir 3.8.x)	15
3.8.1	Initialisation	15
3.8.2	Ballade	15
3.8.3	Langue	16
3.8.4	Emotion	16
3.8.5	Activité	16
3.8.6	CatchMe	17
3.8.7	Dance	17
3.8.8	PreBallade	17
3.8.9	Pause	17
3.9	Historique	17
4	Réalisation.....	19
4.1	Dossier de réalisation	19
4.2	Description des tests effectués	19

4.3	Erreurs restantes	20
4.4	Dossier d'archivage	21
5	Mise en service.....	22
5.1	Rapport de mise en service	22
5.2	Liste des documents fournis	22
6	Conclusions.....	22
7	Annexes.....	23
7.1	Sources – Bibliographie.....	23
7.2	Journal de travail	23
7.3	23
7.4	Manuel d'Installation	23
7.5	Manuel d'Utilisation.....	23
7.6	Archives du projet.....	23
7.7	Schéma matériel.....	24
7.8	Schéma des fonction intégrée du Thymio.....	26

1 Analyse préliminaire

1.1 Introduction

Ce projet est réalisé dans le cadre des TPI de la session 2013-2014 de l'ETML. Il s'agit dans ce projet de faire communiquer deux robots Thymio II de manière à leur faire exécuter au deux une action commune. Celle-ci, sera déterminée en fonction des "émotions" de chaque un des robots.

Le choix de ce sujet de TPI vient de plusieurs facteur, tout d'abord, un fort attrait a tout ce qui touche a la robotique, ensuite, le Thymio II bien que très "cheap" pour un robot permet d'avoir une bonne approche de la programmation robotique, avec ses contrainte ou ses limitations, il est intéressant de se rendre compte que tout ce que nous faisons sur un ordinateur, n'est pas possible sur un robot.

Dans le cadre de la préparation au TPI, une approche du langage Aseba ainsi que de la programmation sur le Thymio a été effectuée. Ce langage et ses limites sont donc connues. L'inventaire des programmes crée lors de la phase de préparation au TPI est trouvable sur le lien suivant :

<https://github.com/Euxner/Projet/tree/master/Pre-TPI>

Ce chapitre décrit brièvement le projet, le cadre dans lequel il est réalisé, les raisons de ce choix et ce qu'il peut apporter à l'élève ou à l'école. Il n'est pas nécessaire de rentrer dans les détails (ceux-ci seront abordés plus loin) mais cela doit être aussi clair et complet que possible (idées de solutions). Ce chapitre contient également l'inventaire et la description des travaux qui auraient déjà été effectués pour ce projet.

Ces éléments peuvent être repris des spécifications de départ.

1.2 Objectifs (repris du cahier des charges)

1.2.1 Objectif SMART

Programmation d'un mode autonome du robot durant lequel, il se "ballade" à la recherche de rencontres :

- Lorsqu'un robot rencontre un compatriote, il s'arrête.

Définition d'un protocole de communication multilingue. Il s'agit de définir une manière de communiquer en prenant en compte qu'un robot parle plusieurs langues et qu'il peut détecter si on lui parle dans sa langue :

- Définition du protocole pour l'établissement de la connexion.
- Définition du protocole pour la négociation de la langue à utiliser.
- Définition du protocole de négociation pour l'activité à réaliser en commun.
- Définition du protocole de négociation pour la fin de connexion.

Chaque étape du protocole doit être de type négociable et on utilisera une stratégie aléatoire pour représenter l'état émotionnel du robot.

Les différentes étapes du protocole devront figurer de manière schématisée dans la documentation.

Implémentation du protocole de communication dans les robots

- Chaque partie du protocole doit être implémentée et validée avec les robots.
- Le programme doit être chargé en mémoire persistante dans le robot (pas de connexion préalable avec le câble USB pour la version livrée).

Langues supportées

- Le robot1 parle français et allemand (langue maternelle).
- Le robot2 parle français et anglais (langue maternelle).
- Chaque robot préfère sa langue maternelle par défaut.
- La communication technique (infrarouge) entre les robots devra être reflétée par des sons du robot en utilisant des voix humaines enregistrées.

Identification

- Chaque robot possèdera un identifiant unique.
- Pour simuler une reconnaissance visuelle, chaque robot enverra régulièrement un message technique (via infrarouge) avec son identifiant.

Mémoire : Le robot est capable de mémoriser les informations suivantes

- La dernière langue parlée avec un robot rencontré (identifiant du robot).
- Le nombre de rencontres faites depuis sa réinitialisation.

Actions à réaliser en commun :

- "Catch me" : Un robot propose à l'autre de l'attraper. Si l'autre robot accepte, l'initiateur exécute un parcours autonome puis s'arrête dès qu'il est rejoint par son compatriote.
- "Dance" : Un robot propose à l'autre de danser, dans le cas d'un accord, les deux robots exécutent une danse "inscrite dans leurs gênes", puis s'arrêtent (ils se synchronisent pour jouer une musique durant la danse).

Actions/Emotions/Décisions possibles :

- "HELLO" : le robot salue l'autre.
- "OK" : le robot accepte ce que l'autre propose.
- "N/A" : le robot est indécis (en réponse ou en question sur l'activité à faire).
- "KO" : le robot refuse la proposition.
- "PLAY?": le robot propose une activité.
- "QUIT?": le robot propose de s'en aller.
- "BYEBYE" : le robot dit au revoir.

Couleur des émotions :

- Le robot doit être en accord avec son dialogue en présentant une couleur de leds appropriée à son discours (non=rouge, oui=vert,...).

1.2.2 Si le temps le permet

Implémenter la notion de "vertige". Lorsque le robot perd contact avec le sol, il devient très nerveux (led, son,...).

Ajouter de nouvelles actions communes (trouver un objet particulier par exemple).

Agrémenter le protocole avec des possibilités de dialogues humoristiques pour lesquels chaque robot pourra choisir de rire ou pas.

1.2.3 Fonctionnalités requises (du point de vue de l'utilisateur)

Le robot doit être contrôlable de la manière suivante :

- Flèche en haut : Démarre le robot en mode autonome à la recherche d'un ami.
- Flèche gauche : désactiver les leds.
- Flèche droite : désactiver les sons.
- Bouton central : pause.
- Flèche bas : reset.

Ce chapitre énumère les objectifs du projet. L'atteinte ou non de ceux-ci devra pouvoir être contrôlée à la fin du projet. Les objectifs pourront éventuellement être revus après l'analyse.

Ces éléments peuvent être repris des spécifications de départ.

1.3 Planification initiale

Il y a eu d'importante modification de la planification entre la planif initiale et détaillée, notamment en ce qui concerne la conception et l'analyse du projet, de ce fait, présenté la planification initiale du projet n'est que peut utile.

Pour avoir une vue plus détaillée des différence entre les deux planification, voir la planification détaillée, cf. 2.6

Ce chapitre montre la planification du projet. Celui-ci peut être découpé en tâches qui seront planifiées. Il s'agit de la première planification du projet, celle-ci devra être revue après l'analyse. Cette planification sera présentée sous la forme d'un diagramme de Gantt et/ou de PERT (l'utilisation de MS project est conseillée).

Ces éléments peuvent être repris des spécifications de départ.

2 Analyse

2.1 Cahier des charges détaillé

Le cahier des charges complet avec toutes ses annexes:

- *multimédia: carte de site, maquettes papier, story board préliminaire, ...*
- *bases de données: interfaces graphiques, modèle conceptuel.*
- *programmation: interfaces graphiques, maquettes, analyse fonctionnelle...*

2.2 Stratégie de test

Dans le cadre de ce projet, Il y a plusieurs type de test.

Le premier type de tests consiste en un test en live après chaque phase de développement du projet, ces tests sont pour la plupart des tests dirigé, ou l'aléatoire est supprimer grâce l'affectation d'une variable fixe a la place de l'aléatoire. Ces tests sont plus de l'ordre du déboggée que de celui du réel test du projet étant donné qu'ils sont très dirigé

Le deuxième type de test est quant a lui beaucoup plus probant pour réellement tester l'application. En effet ceux-ci sont là pour vérifié le fonctionnement du projet et pour cela, tout les aléatoire sont rétabli. Ces tests sont utiles, car ils permettent de voir les réaction du projet dans des situations qui n'était pas forcement prévue ou alors qui ont peu de chance de se produire.

le dernier type de test consiste en une utilisation "manuel" du projet. il consiste à lancé les deux programmes en parallèle avec un point d'arrêt sur la zone du test puis de continué à exécuté le code manuellement pour savoir si une erreur est due à une erreur de codage ou à un problème de délai de transmission d'une information

Ces tests bien que gérant beaucoup d'erreur possible, ne seront malheureusement pas exhaustif, car bien que un nombre certain de tests aléatoire aille été fait pour éviter un maximum de bugs, les tests représentant une utilisation "classique" du programme, il est possible que certain bug existe encore lors un des robots entre dans un cas particulier. (pour exemple de cas particulier, le mode ballade, celui-ci a pour but d'éviter les objets en tournant à gauche si l'objet est à droite et vice-versa, néanmoins lors de la détection simultanée d'un obstacle à gauche et à droite, le robots est perdu.)

Décrire la stratégie globale de test:

- *types de des tests et ordre dans lequel ils seront effectués.*
- *les moyens à mettre en œuvre.*
- *couverture des tests (tests exhaustifs ou non, si non, pourquoi ?).*
- *données de test à prévoir (données réelles ?).*
- *les testeurs extérieurs éventuels.*

2.3 Budget

Le budget de ce projet est assez faible, celui-ci ne désexcite que deux robots Thymio II d'une valeur de 100.- pièce

Ce projet n'est pas rémunéré, néanmoins, il est intéressant d'estimer le cout d'un telle projet. En se basant sur le prix basique d'un informaticien cfc facturé environ 80fr de l'heures et en partant du principe que c'est un projet de 120 heures, le coup de l'employer sur le projet serait de 9600.-. Si l'on rajoute toutes les autres charges, nous arrivons facilement a un projet entre 10'000 et 15'000 franc.

Le budget inclus tous les coûts du projet:

- *achats de matériel, de livres...*
- *en cas de projet rémunéré : coût en main d'œuvre.*

Si le projet n'est pas rémunéré : "budget horaire" (nombre total d'heures de travail planifiées).

2.4 Etude de faisabilité

Pour l'étude de faisabilité, 2 aspects seront principalement étudié, les risques technique et les risques de planning et de ressources humaine

En ce qui concerne les risques techniques, ceux si sont assez faible, en effet, ayant eu 5 semaines avant le TPI pour sa familiarisé avec le langage Aseba, celui ci est connu, néanmoins, il est possible que la complexité de certain bout de code telle que la gestion des émotions posent quelque problème.

Pour le deuxième aspect, les risque sont plus grand, en effet, le projet semble clairement faisable si l'on se fil à a la planification, néanmoins certaine partie peuvent comme cité dans les risques théorique être complexe, le planning étant assez serré, il est possible que cela pose quelque problème.

Détailler les 3 aspects de l'étude de faisabilité:

- *risques techniques (complexité, manque de compétences, ...).*
- *risques concernant le planning & les ressources humaines.*
- *risques concernant le budget.*

Décrire aussi quelles solutions ont été appliquées pour réduire les risques (priorités, formation, actions, ...).

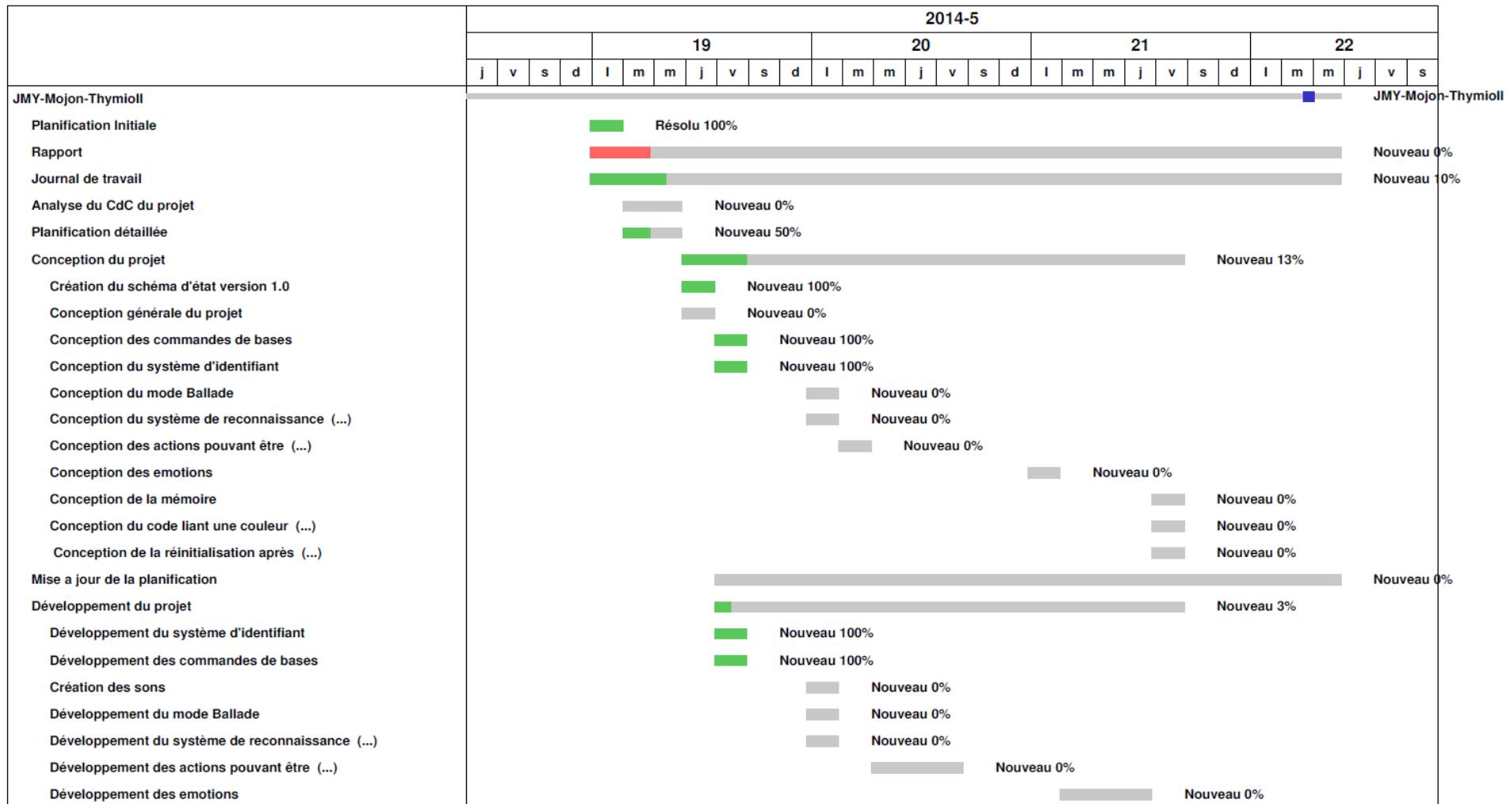
2.5 Compléments d'analyse éventuels (a possiblement enlevé)

Inclure les compléments d'analyse s'ils ne sont pas définis dans le cahier des charges:

- *pour un site: analyse audience & concurrence, navigation, charte graphique complète, ...*
- *pour une base de données: analyse des informations à stocker.*

2.6 Planification

Voici la planification définitive du projet,





Comme mentionné dans le chapitre sur la planification initiale, les deux version de la planification diffèrent beaucoup. Les principales parties concernées sont les notions d'analyse et de conception. Dans la planification initiale, il n'y avait pas de phase d'analyse du CdC ni de phase de conception proprement parlé, ces phases étaient plus ou moins contenues dans les parties de développement. Dans un souci de ne pas se perdre dans un codage trop rapide et irreflèchi, ces parties de conception ont été rajoutées et précédées d'une phase de conception globale du projet. Cela permet avant de coder, d'avoir une notion claire du projet dans son ensemble.

Révision de la planification initiale (Gantt et/ou PERT) du projet :

- *planning indiquant les dates de début et de fin du projet ainsi que le découpage connu des diverses phases.*
- *partage des tâches en cas de travail à plusieurs.*

Il s'agit en principe de la planification définitive du projet. Elle peut être ensuite affinée (découpage des tâches). Si les délais doivent être ensuite modifiés, le responsable de projet doit être avisé, et les raisons doivent être expliquées dans l'historique.

2.7 Historique

Lors de la rencontre avec l'expert et a la lecture du cahier des charges aucune modification du dit cahier n'a été effectué. Néanmoins, durant le projet en lui même , une modification a eu lieu.

Celle-ci touche l'activité CatchMe. Après la première démonstration live au chef de projet, la version développée ne correspondait pas exactement a ce qui était attendu. Après quelque observation sur la faisabilité de ce qui était demandé, il a été décidé de modifiée CatchMe en FollowMe.

En effet le premier était censé permettre une réinitialisation lorsque un robot touchais le deuxième, mais ce choc n'étant pas suffisant pour entraîné une modification visible des accéléromètres, il a été décidé de garder uniquement la partie de suivie avec une réinitialisation en fonction d'un timer.

Pour le cahier des charges: lister toutes les modifications demandées par le client, et aussi celles décidées pour d'autres raisons

- *Pour le budget : comparaison entre le budget initial et le budget.*

3 Conception

3.1 Matériel Hardware

Le projet à été développé sur un ordinateur "standard" de l'ETML dont voici les spécification de base.

Carte mère: Intel DQ67SW (vPro)

Processeur: Core i7-2600,3.4GHz

Carte graphique: Core i7-2600 with Intel HD Graphics (carte intégrée)

Ram: 4x4 Go DDR-3, 1333MHz

Disque dur:WD Green 300Go

OS: Windows 7

Pour ce projet, l'ordinateur en lui même n'as de loin pas besoin d'être une bête de course. En effet le programme le plus gourmand qu'il fera tourné est Aseba Studio. Un ordinateur ayant quelques année sera parfaitement capable de réalisé ce projet.

3.2 Thymio II

En ce qui concerne les 2 robots Thymio II, il n'y a rien de spéciale les concernant autre le fait que ces deux robots doivent être mise a jour avec la version 7 du firmware d'aseba. Cette version permettant la communication locale entre deux robots à l'aide de la fonction Prox.comm. Cette fonction est indispensable pour le projet, car elle est le seul moyen de faire communiquer deux Thymio II.

la version 7 du firmware est disponible gratuitement sur le site d'aseba ou a l'adresse suivante.

<https://aseba.wikitot.com/fr:Thymioupdate>

3.3 Contrainte matérielle

Programmer un robots ne reviens pas a programmer un programme sur un ordinateur. Dans un robots, un certain nombre de contrainte matériel se rajoute.

3.3.1 Mémoire

La contrainte principale est lié a la mémoire du robots. Celle-ci n'est de loin pas aussi grande que celle d'un ordinateur. De ce fait, il est nécessaire de bien concevoir son code pour évité la surconsommation de mémoire dans un grand programme. Pour exemple, initialisé un tableau simple a 7 entrée consomme a lui seul 1% de la mémoire du robots

3.3.2 Approximation des composants

Les composants du robots en lui même nous pose des contraintes. ceux-ci sont pour de nombreuses mesures très approximatifs. En voici quelques exemples:

▫ prox.ground.reflected	(2)
0	332
1	260
▫ prox.ground.delta	(2)
motor.left.target	0
motor.right.target	0
motor.left.speed	-3
motor.right.speed	-4
motor.left.pwm	0
motor.right.pwm	0
▫ acc	(3)
0	3
1	0
2	22

Cette image est tirée d'Aseba Studio et correspond au variables mesurée en live.

On remarque plusieurs éléments étranges.

Premièrement les deux capteurs de "sol" bien qu'exactement dans les mêmes situations, renvoi un résultat avec un écart de 60. Sachant que le maximum de ces capteurs est d'environ 600, avoir une différence de valeur de 10% sur 2 capteurs situés à 2centimètre l'un de l'autre est assez grand.

Deuxièmement, la vitesse même si le robots est arrêter n'est jamais égal à 0. Dans ce cas là, les deux vitesse sont presque égal, mais dans certain cas il y a jusqu'à 10 unité de différence. Cela qui provoque immanquablement un décalage qui fait que le robots ne va pas droit.

Ces exemples ne sont que quelque uns des erreurs que l'on trouve dans les composants du Thymio. Ces erreurs sont encore plus choquante que le robots est censé s'auto-calibré pour donné les valeurs les plus juste possible.

Néanmoins, il ne faut pas oublier que nous avons affaire ici à un robot qui coutent une centaine de francs, et que certain petit défaut peuvent être lié à l'interaction d'une pièce avec une autre.

3.4 Contrainte logicielles

En plus des problèmes matériels, la programmation du Thymio présente aussi quelque contrainte logicielles.

En effet, la programmation en langage Aseba est assez limitée. On se rend vite compte lorsque l'on conçoit son programme, que certaine chose son juste impossible avec le langage Aseba.

Par exemple, créer une méthode récursive est impossible dans ce langage

Autre exemple, en programmation, envoyer une variable dans une méthode, puis la "return" dans le programme principale est quelque chose de courant. Dans le langage Aseba, c'est impossible. Néanmoins, toutes les variables accessible ou que l'on soit dans le programme (pas de notion de variable publique ou privée).

3.5 Logiciels

Dans ce projet, un certain nombre de logiciels tiers ont été utilisés, voici la liste.

3.5.1 Aseba studio (v 1.3.1)

Ce programme est le logiciel de développement de base de Thymio II. Il s'agit d'un éditeur de code spécialement créé pour les robots de type Aseba.

Il permet surtout de créer le lien entre le code et le Thymio II en permettant "d'injecter" le code que nous avons produit dans le robot de manière à le tester ou l'utiliser. Il nous fait aussi une vision claire des paramètres du robot en permettant de voir la valeur de toutes les variables en live lors de l'exécution du programme.

En dehors de cela, il possède toutes les options de base d'un éditeur de code telle que la mise en couleur des variables, des méthodes, etc ou la complétion.

Ce logiciel a néanmoins un gros bémol, il nécessite obligatoirement la connexion d'un robot de type Aseba pour fonctionner. De plus, le logiciel s'adapte au type de robots Aseba connectés, pour que le code compile et que nous ayons accès à toutes les fonctions du Thymio, il est nécessaire qu'un Thymio soit connecté.

À titre d'exemple, le robot E-puck de Aseba, pourra être connecté au Studio et permettra de développer un programme pour le Thymio, mais la compilation ce fera pour un robot E-puck et donc renverra des erreurs qui n'existeraient pas avec un Thymio connecté.

3.5.2 Audacity (v 2.0.5)



Audacity est un logiciel libre, gratuit et facile d'utilisation permettant l'édition ou l'enregistrement de piste audio.



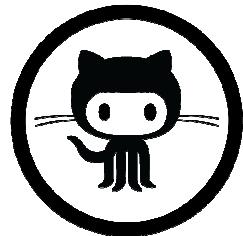
Tous les sons qui sont utilisés dans le Thymio, ont été enregistrés puis retravaillés pour être mis au bon format audio avec Audacity. Il est important de jouer un son sur le Thymio, que celui-ci remplisse diverses conditions, 5 pour être précis. Premièrement, le taux d'échantillonnage doit être de 8000Hz. Ensuite ce dernier doit être en mono ainsi que de type WAV (WAVEform audio file format). Finalement, le son doit être encodé en 8bits non signé, et il ne doit posséder aucune métadonnée.

3.5.3 Graphity

Graphity est une application web permettant de créer facilement des diagrammes et de les exporter au format png.

Dans le projet, il est utilisé pour créer le diagramme d'état.

3.5.4 Github



Github est un service de gestion de développement logiciel se basant sur le principe de Git. Il est gratuit dans le cas de l'utilisation d'un dépôt publique.

Dans notre cas, bien que l'outil de gestion nous soit utile comme l'on travail avec du code, c'est plus en temps que système de stockage et de backup qu'il est utilisé.

3.5.5 Redmine

Redmine est une application web libre de gestion de projet. Il est utilisé dans notre projet pour faciliter sa gestion et avoir une vision d'ensemble.



3.6 Gestion des variables

Comme dans tout projet informatique, la gestion des variables et surtout du nom de ces dernières est très important. Pour ce projet, le nommage de façon clair et explicite des variables et des fonctions est très important pour ne pas se perdre dans l'ensemble du code.

Pour ce faire, l'ensemble du projet a été écrit en suivant la convention de nommage CamelCase consistant à mettre les premières lettres de chaque mot en majuscule et sans espace. Cela permet des noms lisibles et surtout facilement compréhensibles.

3.7 Conception générale du projet

Le projet étant relativement vaste, il est nécessaire de concevoir en gros l'ensemble du projet.

3.7.1 Les boutons

Le Thymio possède cinq boutons sur son capot. Quatre flèches directionnelles et un bouton centrale.

La flèche arrière arrête l'exécution du programme et réinitialise toutes les variables. Pour ce faire il arrête tous les timers et exécute la méthode d'initialisation.

Les flèches gauche et droite activent respectivement les lumières et les sons pour la suite du programme. Ce sont des variables booléennes simples (true = 1 / false = 0)¹

La flèche en avant permet de lancer le programme avec les paramètres définis avec les autres touches.

¹ variables booléennes créées avec des constantes (true = 1 / false = 0)

Le bouton centrales permet de mettre en pause le robot. Pour ce faire, il est nécessaire de sauvegarder l'état dans lequel ce dernier est et de remplacer l'état actuel par un état "pause"

3.7.2 Diagramme d'état

Pour faciliter la gestion du robots ainsi que la compréhension des différentes étapes du programme, il est nécessaire, d'établir un diagramme des différents état que les robots auront. Dans ce diagramme, il y a d'une part dans quel état est le robot à l'instant x, mais aussi tout les informations sur ce qui concerne la transition entre les états

Le diagramme d'état est trouvable en annexe.

3.8 Les différents Etats

Cette partie s'attardera plus en détail sur les divers Etat que peut avoir le robot ainsi que sur la manière dont ces états sont codé.

3.8.1 Initialisation

C'est l'état qui se lance au démarrage du programme ainsi que lors du fin de communication entre deux robots. Il permet de remettre à 0 l'ensemble des variables permettant la discussion. Cette méthode n'utilise aucune fonction particulière, il s'agit uniquement de reprendre toutes les variables nécessaires pour la suite et de les reset à leur valeur de base.

Après avoir réinitialisé l'ensemble des variables, le thymio passe dans l'état un "Ballade".

3.8.2 Ballade

Au départ, nous avons donc 2 robots. Ceux-ci doivent évoluer dans un environnement simple, pouvant avoir quelque obstacle. Il y aura donc un méthode permettant d'éviter les murs.

Pour ce faire, il faut vérifier à chaque instant t, dans notre cas toute les 100ms les valeurs que donne les capteurs horizontaux de distance². Si cette valeur dépasse un certain seuil, le robot est trop près d'un obstacle. Il modifiera donc la vitesse de ses roues pour éviter l'obstacle en fonction du capteur qui a détecté l'obstacle.

Dans un premier temps, il n'est pas prévu que les robots gèrent des trous.

Au moment où les deux robots se rencontrent, la communication doit se lancer. Pour ce faire, lorsqu'il sera en mode ballade, le robot activera son système de communication local³ et émettra par ce biais son identifiant unique. Lorsque un autre robot capte cette identifiant, et qu'il sera assez proche de lui, la communication démarera.

² Dans le Thymio il s'agit des variables prox.horizontal. (ex prox.horizontal[4] > 2000)

³ Fonction prox.comm. Cette fonction permet d'envoyer jusqu'à 10 bits de donnée via les capteurs infrarouge.

lorsque la communication entre les deux robots sera établie, le robots passera a l'état deux "Langue"

3.8.3 Langue

A partir de cette connexion, les robots doivent négocier la langue qu'ils vont utilisé. Pour ce faire, une valeur "binaire est attribuée a chaque langue. Un au français, Deux a l'anglais et quatre a l'allemand. a partir de la chaque robots envoyé via sa communication local la valeur combinée des langues qu'il parle. Le robots qui reçoit les donnée n'a alors plus qu'a décomposé la valeur reçue en binaire a et le comparer a la table binaire des langues qu'il parle pour savoir quel langue il a en commun avec l'autre robot.

Au moment ou la langue est décidée, le programme passe a l'état trois "émotions"

3.8.4 Emotion

Cette état est le plus complexe. Après avoir choisi la langue de discussion, l'un des deux robots lance la discussion dans la langue choisie⁴. Après l'échange des formels bonjours, le robot qui a lancé la discussion propose une activité ou hésite sur l'activité a proposé aléatoirement entre les deux possibles. Son homologue, lui a 3 choix qui implique chaque une réaction spécifique.

Dans le premier cas, le robot peut tout simplement accepter l'activité, dans ce cas, les deux robots passent dans l'état quatre "activité", qui leur permettra de lancé l'activité choisie.

Dans le second cas, le thymio peut hésiter, être indécis. Dans ce cas, il le montre et demande a l'initiateur de lui donné une nouvelle activité.

Le dernier cas concerne le refus pur et simple de l'activité. Dans ce cas, le robots signal a l'initiateur qu'il refuse l'activité puis, s'en va en romrant la communication et en activant l'état sept "postballade"

3.8.5 Activité

Cette état a pour seul fonction de permettre aux deux robots d'exécuter la même action en fonction de celle proposée et de savoir qui a proposé l'action pour savoir qui est l'initiateur.

En fonction de l'action choisie, et de quel rôle de robot a, l'action suivant qu'il effectuera sera différente

Si la variable choisissant l'activité est a 1 les robots partiront sur l'activité CatchMe et l'état du même nom numéro cinq. Si elle est égale a 0, l'activité Danse et l'état six seront effectué.

⁴ Cela concerne surtout l'aspect audio de la chose.

3.8.6 CatchMe

Il s'agit d'une des deux activités possible, dans celle-ci, le robot initiateur passe en état ballade et ne s'arrête que lorsque le deuxième robot l'a rejoint. Pour ce faire, le second robots "suis" en réalité le 1er.

3.8.7 Dance

C'est la deuxième activité possible, il s'agit ici de faire que les deux robots fasse une danse commune qui est inscrite dans leur gène. En même temps, les deux robots doivent jouer une musique en étant syncro.

3.8.8 PostBallade

Cette méthode est presque une copie de la méthode ballade. Elle est appelée à la fin d'une des deux activités et elle permet également de se déplacer en évitant les objets. Néanmoins, elle n'a pas la communication locale d'activé, ce qui permet d'éviter que les deux robots venant de communiquer rediscute en boucle.

3.8.9 Pause

La méthode pause est une sorte d'arrêt sur image à n'importe quel moment de la communication. Cette état d'obtient en appuyant sur le bouton central du thymio. Il fonctionne à toute instant.

Fournir tous les documents de conception:

- *le choix du matériel HW*
- *le choix des systèmes d'exploitation pour la réalisation et l'utilisation*
- *le choix des outils logiciels pour la réalisation et l'utilisation*
- *site web: réaliser les maquettes avec un logiciel, décrire toutes les animations sur papier, définir les mots-clés, choisir une formule d'hébergement, définir la méthode de mise à jour, ...*
- *bases de données: décrire le modèle relationnel, le contenu détaillé des tables (caractéristiques de chaque champs) et les requêtes.*
- *programmation et scripts: organigramme, architecture du programme, découpage modulaire, entrées-sorties des modules, pseudo-code / structogramme...*

Le dossier de conception devrait permettre de sous-traiter la réalisation du projet !

3.9 Historique

Aucune modification de la conception du projet n'a été effectuée lors du projet.

Si la conception du projet a été modifiée plusieurs fois, ou de manière significative, expliquez ces changements et leurs causes.

Attention: Pour faciliter la maintenance, à la fin du projet, le dossier de conception doit correspondre à ce qui a été effectivement réalisé !

4 Réalisation

4.1 Dossier de réalisation

Comme demandé sur le cahier des charges, le logiciel est installé comme un programme par défaut dans les deux thymio. Il suffit d'allumé le thymio puis de sélectionné avec les boutons fléché la couleur "incolore" puis de pressé le bouton central pour activé le mode d'utilisation.

Si nécessaire, la version 1.0.0 modifiable du projet est trouvable sur le git: <https://github.com/Euxner/Projet/tree/master/TPI>, dans les parties robot 1 et robot 2. Il y a en effet deux codes qui ont été produit pour ce TPI. Néanmoins, ces deux codes sont extrêmement similaire, les différences venant de certain point telle que les identifiant qui doivent être codé en dur ou des langues parlée, qui doivent elles aussi être codé en dur. Il serait néanmoins possible, sous certain condition de produire un code totalement similaire. Il faudrait pour cela pour les éléments pausant problème, les stockés directement dans la carte sd et utilisé le programme pour lire un fichier de la carte sd a la position x.

Le git contient aussi un dossier code ainsi qu'un dossier documentation. Le premier contient un certain nombre de fonction utilisée pour le développement du projet. Elle sont toutes fonctionnelles mais certaine ont subi encore quelques modifications pour la version finale du projet.

Le dossier documentation quant a lui contient toute la doc du projet. Que cela soit la rapport de projet ou alors la planification ou l'image du diagramme d'état, toute les documents sont contenu dans ce dossier.

Décrire la réalisation "physique" de votre projet

- *les répertoires où le logiciel est installé*
- *la liste de tous les fichiers et une rapide description de leur contenu (des noms qui parlent !)*
- *les versions des systèmes d'exploitation et des outils logiciels*
- *la description exacte du matériel*
- *le numéro de version de votre produit !*
- *programmation et scripts: librairies externes, dictionnaire des données, reconstruction du logiciel - cible à partir des sources.*

NOTE : Evitez d'inclure les listings des sources, à moins que vous ne désiriez en expliquer une partie vous paraissant importante. Dans ce cas n'incluez que cette partie...

4.2 Description des tests effectués

Pour chaque partie testée de votre projet, il faut décrire:

- *les conditions exactes de chaque test*
- *les preuves de test (papier ou fichier)*

-
- *tests sans preuve: fournir au moins une description*

4.3 Erreurs restantes

Il existe encore quelque erreurs ou corrections pouvant être effectuée dans le projet.

Tout d'abord, lors de l'activité CatchMe, certaine fois, le robots censé exécuté l'état ballade ne le fais pas et reste comme figé sur place. Il refuse aussi de se réinitialisé. Ce problème peut être plus ou moins conséquent sur l'utilisation du produit, car il n'intervient que très rarement. néanmoins il oblige a redémarré le thymio. En ce qui concerne les actions envisagée, le problème a été suivi de manière a tenté de l'isolé, néanmoins, même dans le bon état et avec les bonne valeurs de communication locale, le robot refuse de démarrer. Le problème se situe donc fort probablement dans une boucle while qui ne se termine pas au moment voulu.

Une deuxième erreur est liée a la connexion des deux robots lors de la ballade. En effet, certaine fois un des deux robots, ne détecte pas correctement l'autre et la communication ne se lance pas. Les conséquence sont légères, pour résoudre ce problème, il suffit de réinitialisé les deux robots, avec la flèche basse et de les relancé. Après quelque test, le problème a été identifié et proviens du fait que l'un des deux robots ayant des capteurs détectant l'autre plus rapidement (cf 3.3.2) neutralise sa communication locale pour la futur communication et le second ne le vois donc plus et passe son chemin. Il faudrait réglé manuellement la valeur de détection pour éviter ce problème.

S'il reste encore des erreurs:

- *Description détaillée*
- *Conséquences sur l'utilisation du produit*
- *Actions envisagées ou possibles*

4.4 Amélioration possibles

Il existe quelque amélioration qui pourrait être faites sur le projet, de manière a le rendre plus complet et plus simple d'utilisation.

Dans un premier temps, cela concerne surtout l'état ballade, dans notre projet, celui-ci se contente de détecté les obstacle et de les évités. Ce comportement fait que lorsqu'il rencontre un mur, le robot se met simplement a le longé. Il serait bien de crée une fonction appellée aléatoirement qui permette au robot de tourné d'un certaine angle de manière a éviter que, comme dis précédemment, il longe les murs. Dans le même état, il faudrait permettre au robots de détecter et éviter les trous qui peuvent être dans son environnement. Cela permettrait d'utilisé le robots sur une table par exemple.

Une autre amélioration possible, serait de comme cité dans le chapitre 4.1, passé toute les variable devant changé entre les différents robots sur une carte sd et de lire la carte pour initialisé ces valeurs. Cela permettrait de crée ainsi un seul et unique code.

4.5 Problèmes rencontrés

4.6 Dossier d'archivage

Décrire de manière détaillée les 2 archives du projet (CD-ROM, disque zip ou jazz, bandes magnétiques, ...)

Attention: les documents de réalisation doivent permettre à une autre personne de maintenir et modifier votre projet sans votre aide !

5 Mise en service

5.1 Rapport de mise en service

Fournir une description:

- *de l'installation du projet chez le client (pour un site web: publication chez un provider)*
- *des test officiels effectués chez le client et/ou par le client.*
- *des erreurs répertoriées*
 - *description détaillée*
 - *conséquences pour le client*
 - *actions envisagées.*

5.2 Liste des documents fournis

Les documents fourni sont les suivants,

Le rapport de projet.

Le manuel d'utilisation du projet (dans les annexes).

L'ensemble du code et des sources sur un CD (dans les annexes).

Deux robots thymio 2 ayant le programme développé chargé en mémoire.

Lister les documents fournis au client avec votre produit, en indiquant les numéros de versions

- *le rapport de projet*
- *le manuel d'Installation (en annexe)*
- *le manuel d'Utilisation avec des exemples graphiques (en annexe)*
- *autres...*

6 Conclusions

Développez en tous cas les points suivants:

- *Objectifs atteints / non-atteints*
- *Points positifs / négatifs*
- *Difficultés particulières*
- *Suites possibles pour le projet (évolutions & améliorations)*

Annexes

6.1 Sources – Bibliographie

Site du Thymio 2

<https://Aseba.wikidot.com/fr:Thymio>

Aide externes:

Liste des livres utilisés (Titre, auteur, date), des sites Internet (URL) consultés, des articles (Revue, date, titre, auteur)... Et de toutes les aides externes (noms)

6.2 Journal de travail

Date	Durée	Activité	Remarques

6.3

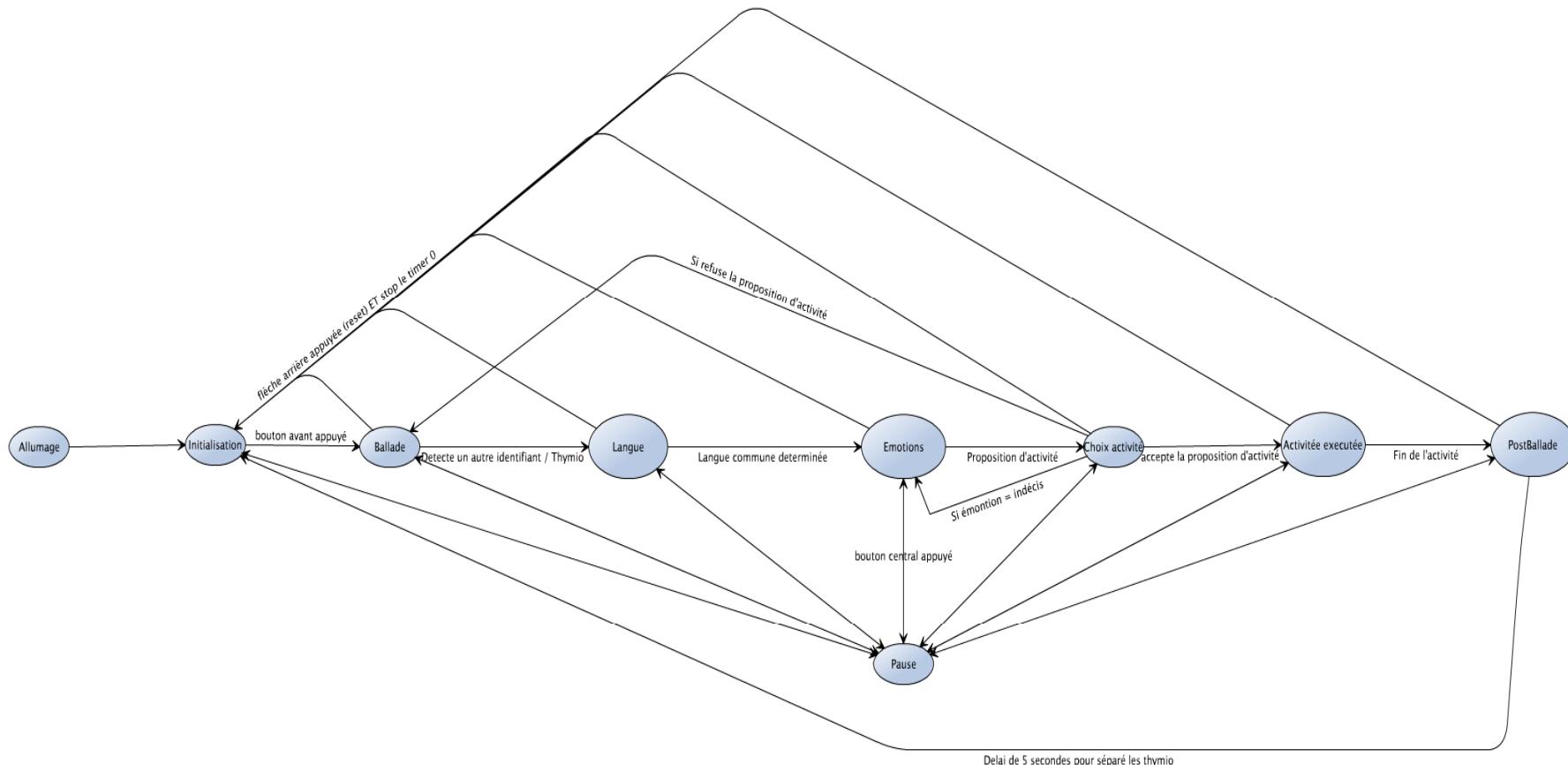
6.4 Manuel d'Installation

6.5 Manuel d'Utilisation

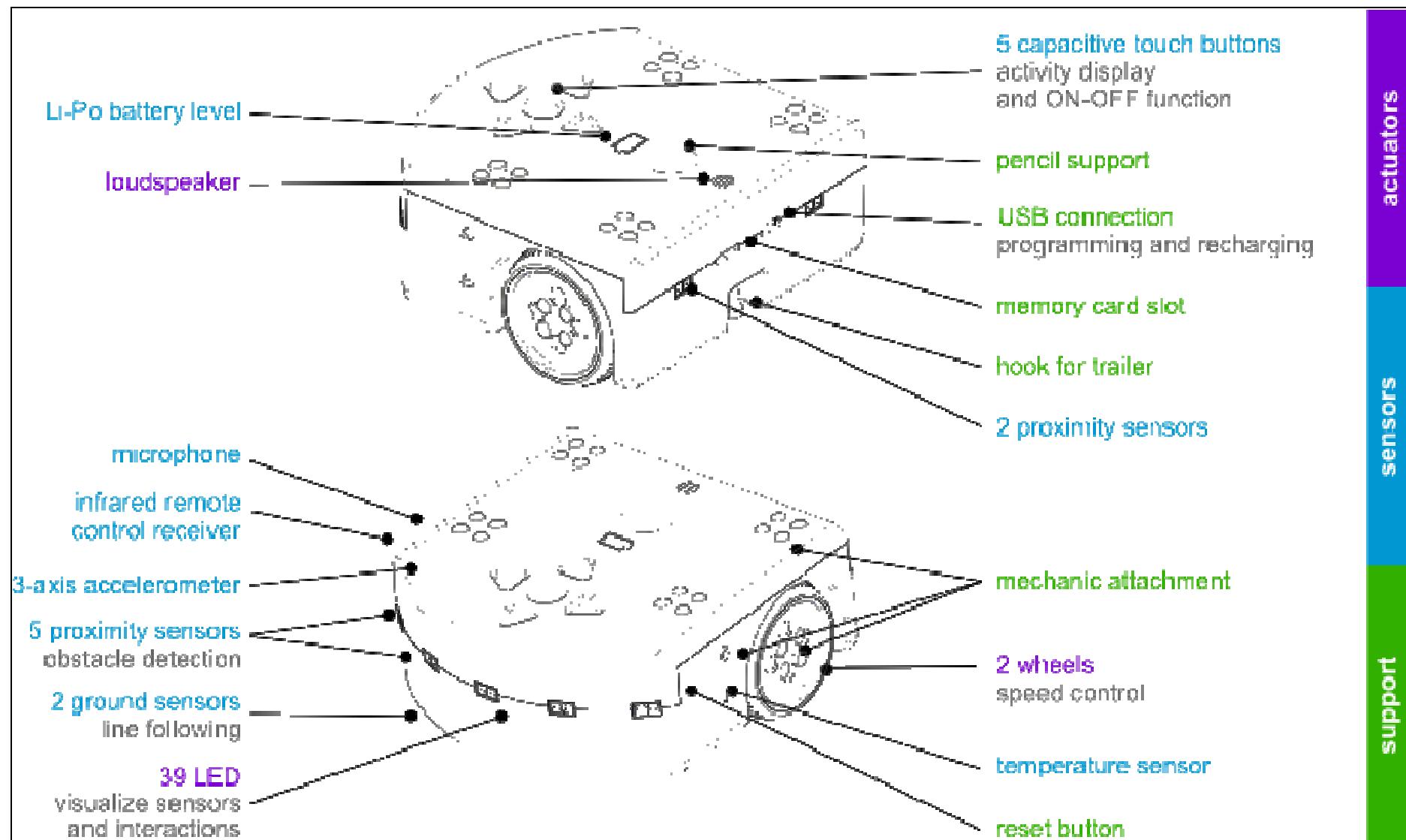
6.6 Archives du projet

CD, ... dans une fourre en plastique

6.7 Schéma d'état



6.8 Schéma matériel



6.9 Schéma des fonction intégrée du Thymio

Variables[indices]

Événements

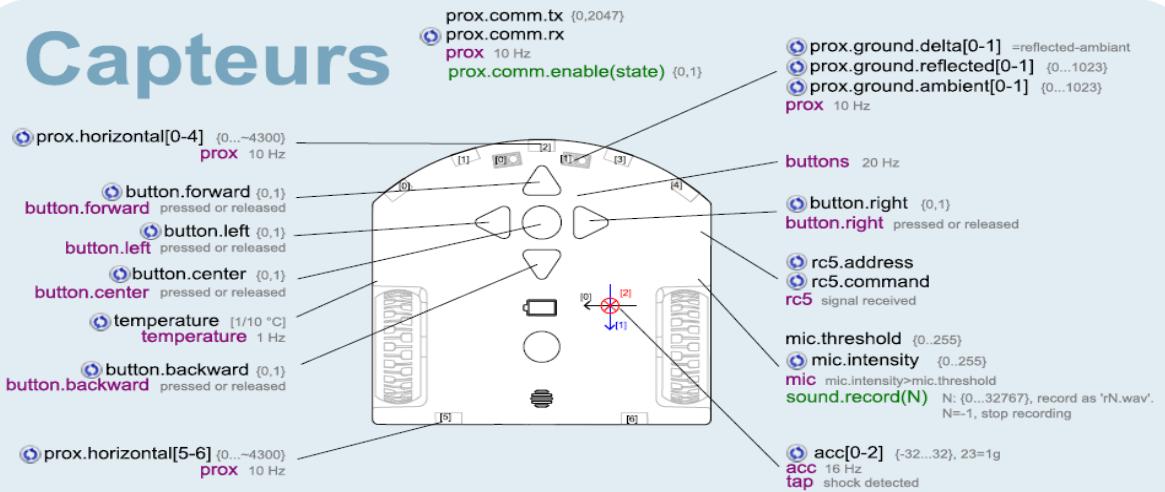
Fonctions

explication,
condition ou fréquence
de l'événement,
{plage de valeurs}
[unité]

variable mise à jour
automatiquement

timer.period[0-1] [ms]
timer0 every timer.period[0] ms
timer1 every timer.period[1] ms

Capteurs



leds.prox.h(led0, led1, led2, led3, led4, led5, led6, led7) {0...32}

leds.buttons(led0, led1, led2, led3) {0...32}

leds.circle(led0, led1, led2, led3, led4, led5, led6, led7) {0...32}

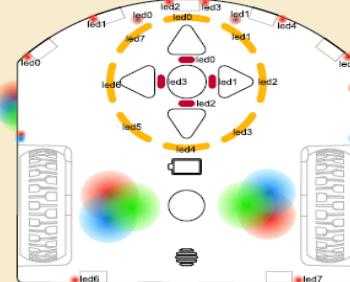
leds.bottom.left(red, green, blue) {0...32}

leds.temperature(red, blue) {0...32}

motor.left.target desired speed {-500...500}, 500 = ~20 cm/s
motor.left.speed actual speed
motor.left.pwm motor command
motor 100 Hz

leds.top(red, green, blue) {0...32}

leds.prox.h(led0, led1, led2, led3, led4, led5, led6, led7) {0...32}



leds.prox.v(led0, led1) {0...32}

leds.rc(led) {0...32}

leds.bottom.right(red, green, blue) {0...32}

leds.sound(led) {0...32}

motor.right.target desired speed {-500...500}, 500 = ~20 cm/s
motor.right.speed actual speed
motor.right.pwm motor command
motor 100 Hz

sound.finished a sound finished playing
sound.system(N) N: {0...7}, play system sound N. N=-1, stop playing
sound.freq(Hz,ds) [Hz],[1/60 s]
sound.wave(wave[142]) change primary wave, wave[] : {-128...127}
sound.play(N) N: {0...32767}, play 'pN.wav'. N=-1, stop playing
sound.replay(N) N: {0...32767}, replay 'rN.wav'. N=-1, stop playing

Actuateurs