Rapport de stage

Sujet : Mise en place d'une tête de robot expressive

Rapport du stage effectué du 13 Juin au 5 Juillet

Geoffroy KEIME

Kevin BERNARD

Maria-Lorena POUPET

TABLE DES MATIÈRES :

Table des matières

Introduction :

Notre stage a débuté le 13 juin par un rendez-vous à Porte d'Italie avec notre tuteur de stage, Philippe GAUSSIER ainsi que la directrice au Centre Tedybear : Jacqueline NADEL. Toute la matinée, nous avons été briefés sur le fonctionnement de la tête expressive (nommée Roboto) ainsi que ses différentes utilités au sein du Centre de recherche. Nous avions donc pour mission de réparer Roboto qui est défectueux ainsi que de faciliter son fonctionnement.

Nous avons donc opté communément qu'un contrôle du robot via un appareil Androïd pourrait être une solution viable pour le contrôler. Après avoir établi avec M. GAUSSIER les différents axes de direction de notre stage, nous avons ramené la tête au site de Saint-Martin, où nous allions travailler au laboratoire ETIS.

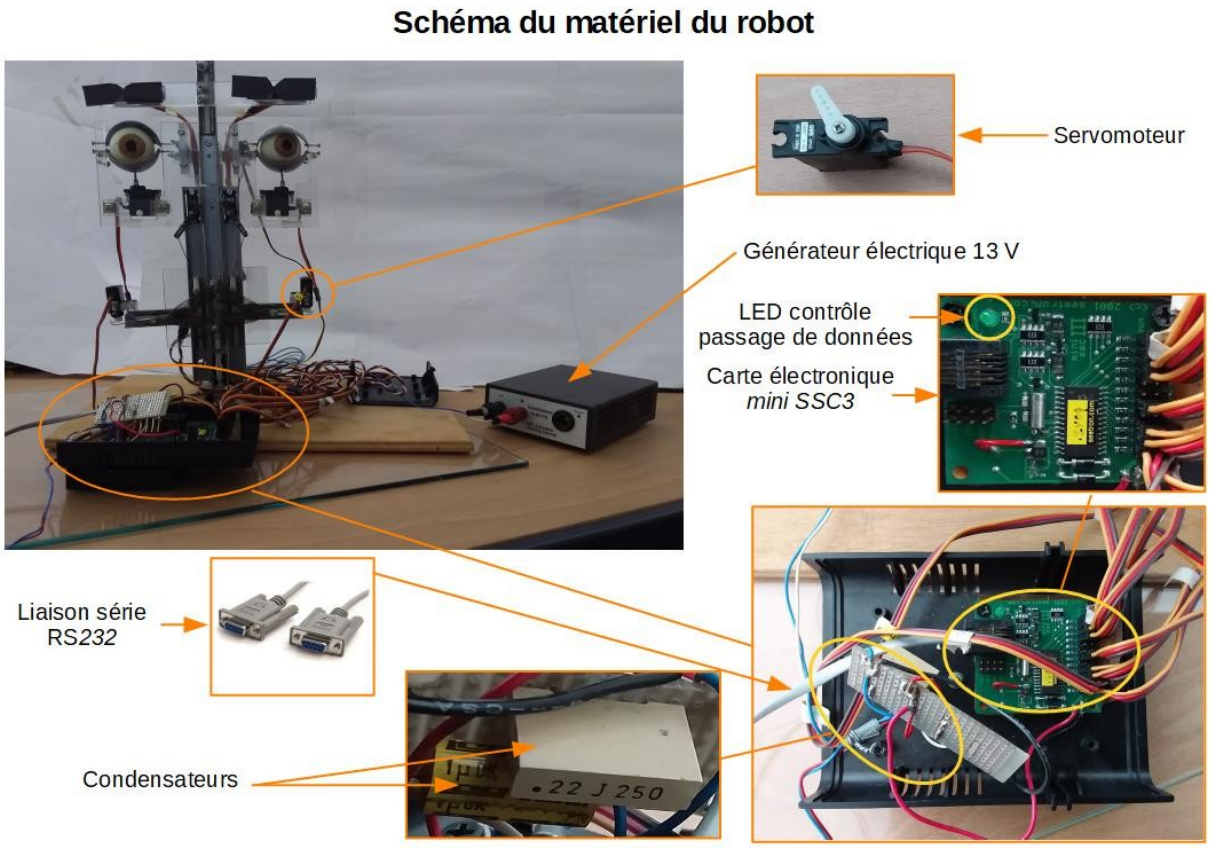


Illustration 1: Matériel fourni

Présentation du projet :

Pour pouvoir répondre aux exigences de Mme NADEL, nous avons mis en place en plan commun pour pouvoir travailler dans les bonnes conditions avec un objectif clair en tête.

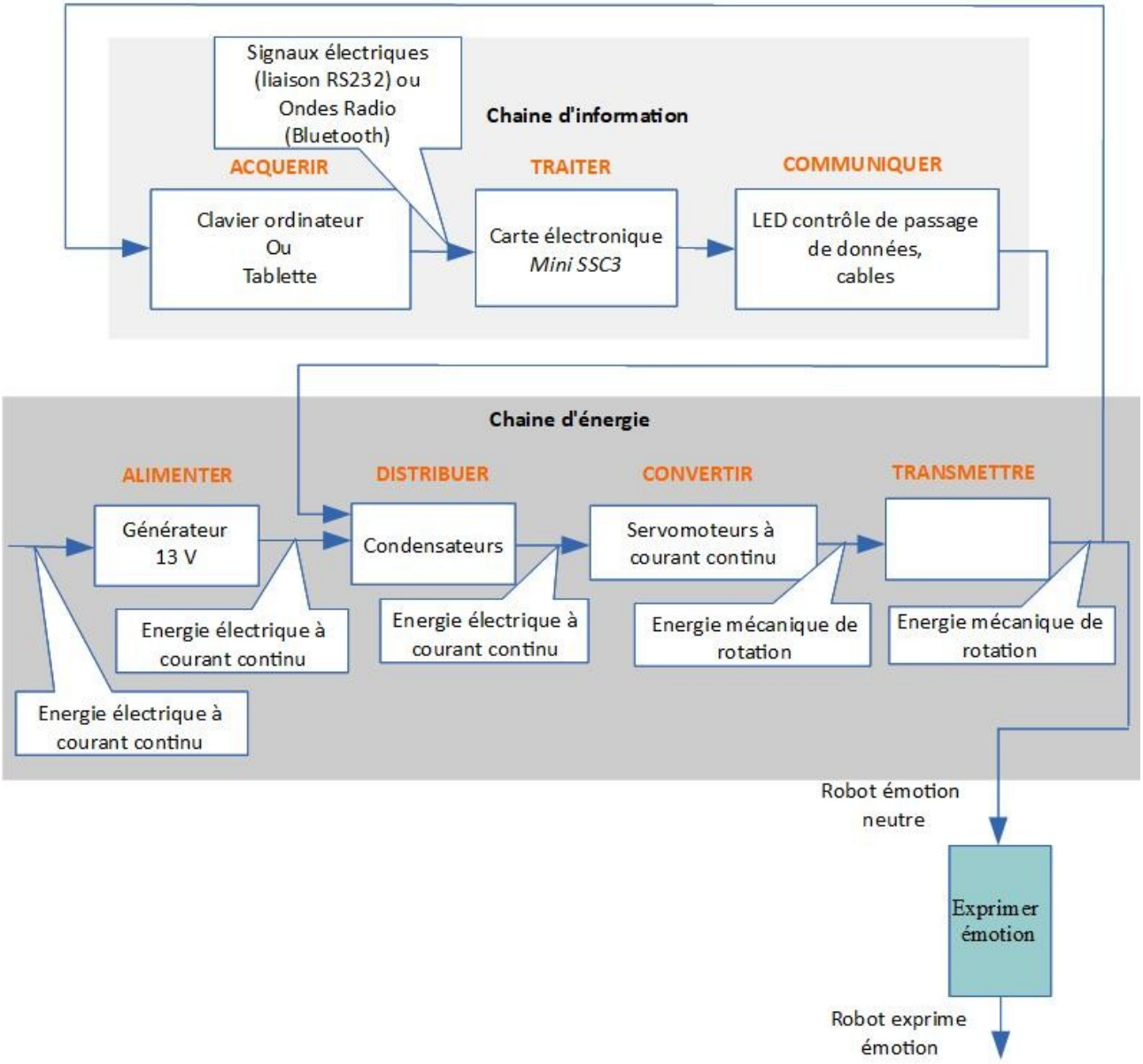


Illustration 2: Schéma des chaînes d'information et d'énergie

Problèmes :

La première étape fut de comprendre pourquoi Roboto ne fonctionnait pas. Le problème pouvant être mécanique ou logiciel. Nous avons donc procédé par élimination pour déterminer la source du problème. Tout d'abord, nous avons testé tous les servomoteurs avec un Arduino. Cinq servomoteurs étaient défectueux. Ensuite, nous avons vérifié que le courant délivré à la carte par l'alimentation était bien de 5 volts. Nous avons mesuré 13 volts à la sortie de l'alimentation et de 5 volts après le condensateur. La partie électrique du montage était donc intacte.

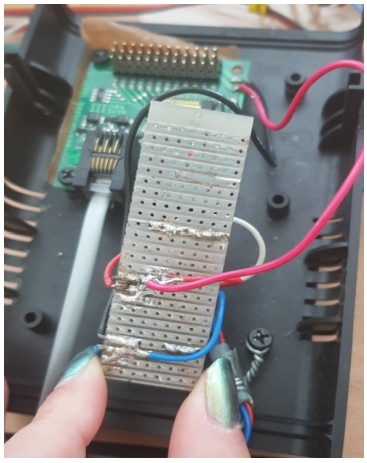


Illustration 3: Photo des soudures

Cependant, suite à nos tests, un câble s'est rompu. Nous avons donc dû le souder sur la carte.

Ensuite, nous avons branché Roboto à l'ordinateur pour essayer de le faire fonctionner. Lors de la mise en tension, les servomoteurs fonctionnels se plaçaient bien sur leur position par défaut.

Pour faire bouger le robot, nous devions nous assurer d'envoyer 3 octets à la carte Mini-SSCIII. Comme ci-dessous :

To command a servo to a new position requires sending three bytes at the appropriate serial rate (2400 or 9600 baud, depending on the setting of the B jumper; see Configuring the Mini SSC II). These bytes

Byte 1	Byte 2	Byte 3
[sync marker (255)]	[servo # (0-254)	[position (0-254)]

Cependant, ne sachant pas comment envoyer les données, nous avons demandé le code créé pour Roboto à Mme NADEL. Elle nous a confirmé qu'elle n'avait plus le code mais allait essayer de le retrouver. Néanmoins, n'ayant jamais eu de retours, nous en avons déduis qu'il était perdu.

```
sbd[0] =(unsigned char)255;
sbd[1]=(unsigned char)2;
sbd[2]=40;

f=fopen("/dev/ttyUSB0","w");

    fwrite(sbd,1,3,f);
    fflush(f);
    usleep(delai);
    fclose(f);
```

Nous avons donc demandé à M. GAUSSIER de l'aide pour faire fonctionner le robot.
Après explication des commandes manquantes, le robot bougeait à nouveau et fonctionnait de la manière suivante :
On envoie un octet pour signifier le début de l'envoie de données : 255, puis la pin du servomoteur que l'on veut faire bouger et enfin l'angle désiré (entre 40 et 255). Le tout envoyé par l'intermédiaire de Minicom.

Ainsi, nous avons commencé à calibrer les servomoteurs en vue de créer les émotions souhaitées.

Interface **C**ommande **R**oboto

L'objectif de cette application est de pouvoir créer les émotions et de les tester. Le programme de l'application permet de modifier la position des servomoteurs pour chaque émotions et de les sauvegardées dans un fichier.

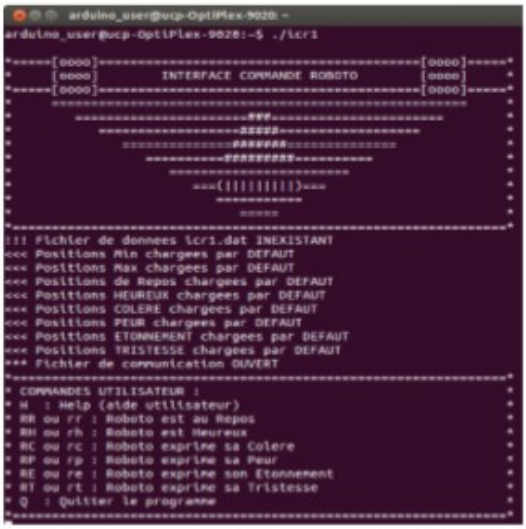
L'application peut-être utilisée hors communication. C'est-à-dire que le fonctionnement de l'application peut-être testée même s'il n'y a pas de lien de communication série avec le robot. Pour l'utiliser en communication le 1 doit être changé en 0 (cf code icr1.c ligne 28 en annexe)

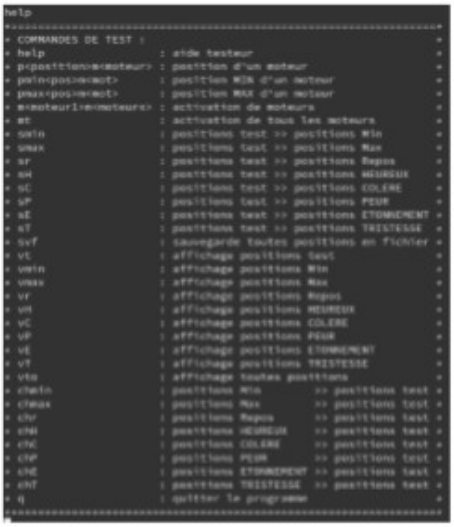
```
#define ICR_SW_COM_LINK    1
```

Cette fonctionnalité est utile lors de la construction de l'application mais nécessite une intervention dans le code c'est pourquoi ce n'est pas une commande utilisateur.

Pour commander le robot, il faut ouvrir le canal de communication et paramétrer la liaison série. Pour cela, on utilise le programme de contrôle MINICOM. On configure la vitesse de la liaison série *dev/ttyUSB0* à *9600 baud.*

Une fois la liaison série configurée on peut compiler le programme de contrôle du robot et l'exécuter.





Le programme est utilisé pour la première fois. Le fichier *icr1.dat* n'existe pas encore. Si bien que les positions sont chargé depuis les tableaux écrit en dure dans le programme.

Le programme vérifie également l'ouverture du canal de communication */dev/ttyUSB0* qui est ci-dessus ouvert. Sinon si le canal est fermé le programme affiche :

```
!!! Canal de communication INEXSISTANT
!!! Activation moteur impossible
```

La commande *help* permet d'afficher l'aide concernant les commandes de test qui vont permettre le calibrage des servomoteurs pour les différentes émotions.

Pour éviter que les moteurs ne forcent et ne grillent les positions minimales et maximales des servomoteurs sont testées au préalable et écrites en dur dans le programme. Il est possible de les sauvegardées dans le fichier *icr1.dat* seulement pour les gardées en visuel *(cf annexes)*. Si bien que si quelqu'un change ces positions dans le fichier sans tester le forçage avec le robot elle ne seront de toutes manières pas prises en compte par le programme.

```
chH
<<< Positions HEUREUX chargees
p100m1
>>> Moteur : 1 Position : 100 OK
```

La commande *ch* permet de rapatrier les valeurs des positions des émotions dans le tableau des positions de test. Dans le cas ci-dessus il s'agit de celles de l'émotion heureux.

```
vt
*** Moteurs :                    |  1|  2|  3|  4|  5|  6|  7|  8|  9| 10| 11| 12|
*** Positions de test :          |100|150|140|  0|150| 50| 60|100|200|150|110|120|
```


```
vt
*** Moteurs :                    |  1|  2|  3|  4|  5|  6|  7|  8|  9| 10| 11| 12|
*** Positions de test :          |100|150|140|  0|150| 50| 60|100|200|150|110|120|
```

```
m1m2m3
>>> Message a Roboto : pin :  0 position : 100
>>> Message a Roboto : pin :  1 position : 150
>>> Message a Roboto : pin :  2 position : 140
mt
>>> Message a Roboto : pin :  0 position : 100
>>> Message a Roboto : pin :  1 position : 150
>>> Message a Roboto : pin :  2 position : 140
>>> Message a Roboto : pin :  3 position : 0
>>> Message a Roboto : pin :  4 position : 150
>>> Message a Roboto : pin :  5 position : 50
>>> Message a Roboto : pin :  6 position : 60
>>> Message a Roboto : pin :  7 position : 100
>>> Message a Roboto : pin :  8 position : 200
>>> Message a Roboto : pin :  9 position : 150
>>> Message a Roboto : pin : 10 position : 110
>>> Message a Roboto : pin : 11 position : 120
```

L'émotion heureux étant chargée comme d'écrit précédemment, l'utilisateur peut tout d'abord afficher les positions de test à l'aide de la commande *vt* puis modifier la position d'un des servomoteurs en utilisant *p<position>m<moteur>*. Une fois les modifications effectuées, l'utilisateur peut constater les modifications dans les positions de test en les affichant de nouveau.
Il peut aussi faire fonctionner les servomoteurs comme ci-contre.

```
sH
<<< Positions HEUREUX sauvegardees
```

```
svf
*** Ecriture fichier icr1.dat OK
```

Les modifications apportées, les positions de test peuvent être sauvegardées dans une émotion avec la commande *s* suivit de l'émotion correspondante en majuscule. Dans notre cas, il s'agissait uniquement d'une modification d'un des servomoteurs de l'émotion heureux. Nous sauvegardons donc de surcroît les positions de test dans l'émotion heureux.
La commande *svf* permet de sauvegarder les positions des servomoteurs des différentes émotions, comme afficher ci-dessus, dans le fichier *icr1.dat* (cf annexes) .

```
vH
*** Positions emotion HEUREUX :         |100|150|140|  0|150| 50| 60|100|200|150|110|120|
vto
*** Moteurs :                           |  1|  2|  3|  4|  5|  6|  7|  8|  9| 10| 11| 12|
*** Positions de test :                 |100|150|140|  0|150| 50| 60|100|200|150|110|120|
*** Positions Min :                     | 40| 40| 60|  0| 50| 40| 40| 40| 40| 30| 70| 40|
*** Positions Max :                     |255|255|250|250|200|255|150|200|255|240|200|255|
*** Positions de Repos :                |100|100|100|180|100|100|100|100|120|130|130|150|
*** Positions emotion HEUREUX :         |100|150|140|  0|150| 50| 60|100|200|150|110|120|
*** Positions emotion COLERE :          | 40|220|100|140|140|100|100|100| 90| 90|100| 40|
*** Positions emotion PEUR :            | 40| 40| 80|140|100| 40| 40|100|100| 90|180|140|
*** Positions emotion ETONNEMENT :      |220|220| 60|200|100|100|100|100|100|100|160|220|
```

```
<<< Positions Mln chargees du FICHIER icr1.dat
<<< Positions Max chargees du FICHIER icr1.dat
<<< Positions de Repos chargees du FICHIER icr1.dat
<<< Positions HEUREUX chargees du FICHIER icr1.dat
<<< Positions COLERE chargees du FICHIER icr1.dat
<<< Positions PEUR chargees du FICHIER icr1.dat
<<< Positions ETONNEMENT chargees du FICHIER icr1.dat
<<< Positions TRISTESSE chargees du FICHIER icr1.dat
<<< Positions Mln chargees par DEFAUT
<<< Positions Max chargees par DEFAUT
*** Fichier de communication OUVERT
```

Une fonctionnalité permet également de modifier les valeurs des positions depuis le fichier, hormis les positions minimales et maximales des servomoteurs.
On quitte le programme, on le relance et on constate que le programme à bien trouver le fichier *icr1.dat* (cf annexes) et qu'il charge désormais les positions depuis le fichier.

Une fois l'ensemble des positions des servomoteurs des émotions paramétrées l'utilisateur peut activer l'émotion sur le robot en utilisant l'ensemble des commande de type R suivit du nom d'une émotion comme ci-dessous :

```
RH
>>> Message a Roboto : pin :  0 position : 220
>>> Message a Roboto : pin :  1 position : 40
>>> Message a Roboto : pin :  2 position : 220
>>> Message a Roboto : pin :  3 position : 40
>>> Message a Roboto : pin :  4 position : 100
>>> Message a Roboto : pin :  5 position : 100
>>> Message a Roboto : pin :  6 position : 100
>>> Message a Roboto : pin :  7 position : 150
>>> Message a Roboto : pin :  8 position : 140
>>> Message a Roboto : pin :  9 position : 140
>>> Message a Roboto : pin : 10 position : 160
>>> Message a Roboto : pin : 11 position : 180
>>> Roboto est HEUREUX
```

Le programme gère également les codes d'erreurs (cf code icr1.c en annexe) comme par exemple :

```
bdfhvbwdn
!!! Commande non supportee
```

Le code entier disponible en Annexe (Figure 1).

Mise en place du Module Bluetooth et la création de l'application :

Une fois que Roboto fonctionnait, nous avons commencé à mettre en œuvre un projet de construction de commande sans-fil. Nous avons donc opté pour une connexion Bluetooth, passant par un module HC-06 que nous connections à un Arduino pour l'alimenter. Nous en avons testé plusieurs, comme Sylvain COLOMER nous l'avait conseillé. Finalement, nous en avons trouvé un seul qui fonctionnait. Nous l'avons configuré pour que la vitesse de transmission soit à 9600 baud grâce à un code trouvé sur internet (joint en annexe, figure 2).

Nous l'avons ensuite connecté au port série de Roboto comme ceci :

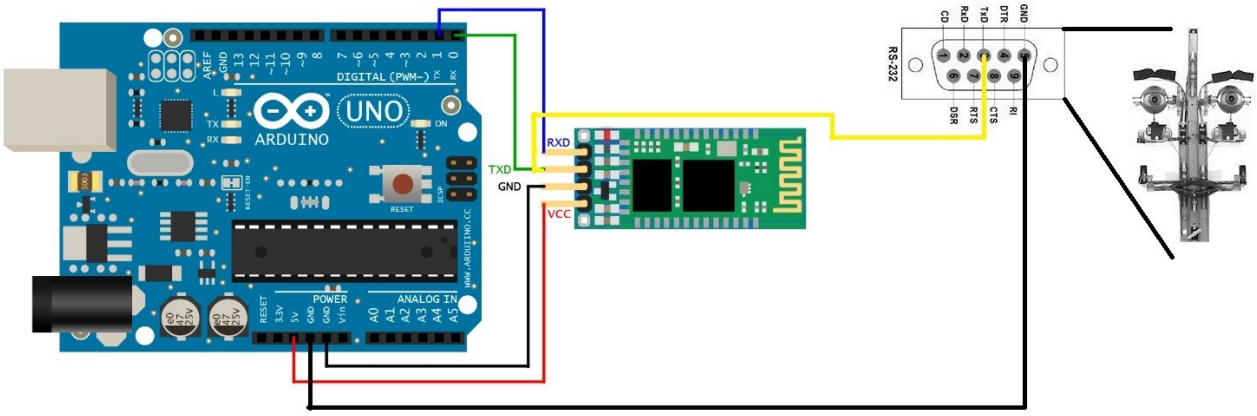


Illustration 4: Schéma de connections entre le module et Roboto

A l'aide du site web MIT APP INVENTOR (un site qui permet de développer des applications simples sans utiliser de langage de programmation), nous avons créé une application qui permet de faire fonctionner les servomoteurs indépendamment.

Cependant, le test s'est soldé par un échec. Nous n'avions aucun contrôle sur les servomoteurs qui se comportaient n'importe comment. Le test nous a quand même permis de vérifier que nous envoyions bien des données via le port série.

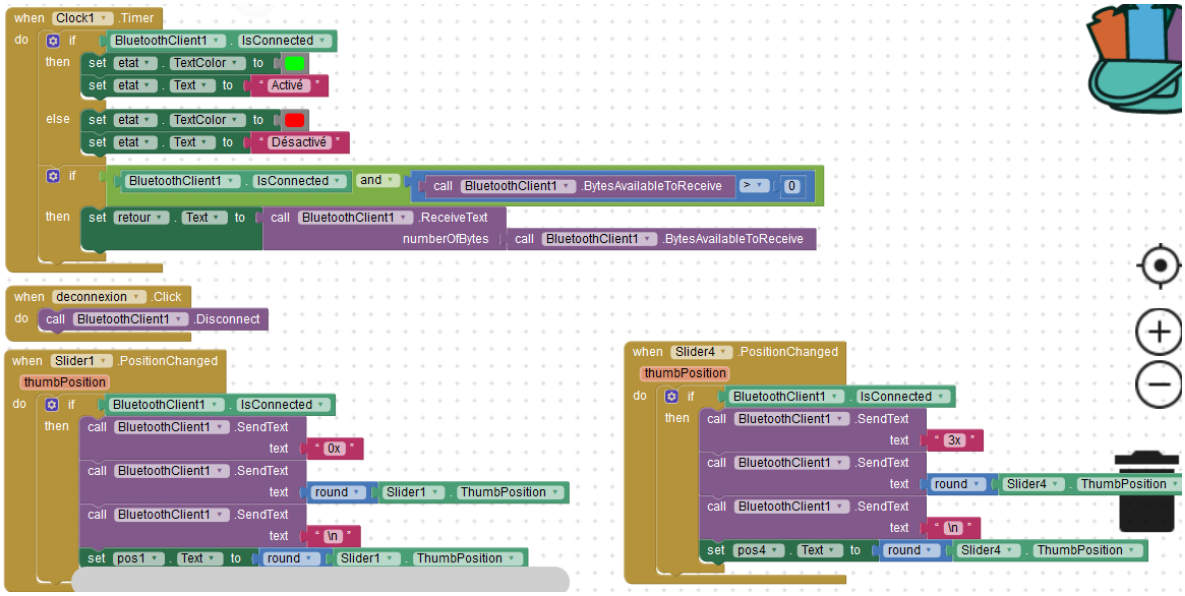


Illustration 5: Code de l'application test

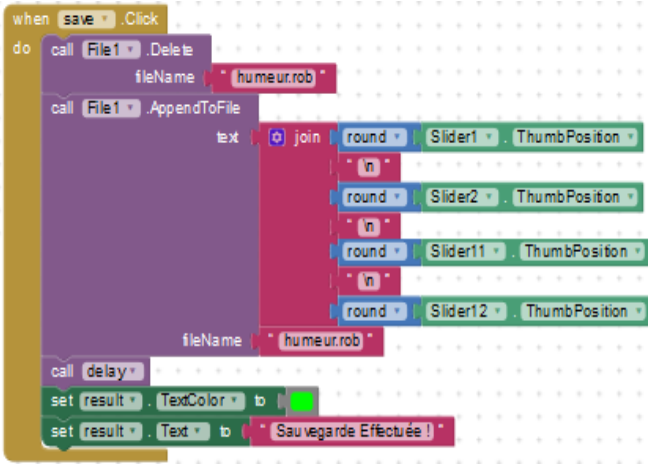


Illustration 6: Code de sauvegarde de l'application

L'application permet aussi de sauvegarder la position des sliders et donc des servomoteurs.
L'image ci-contre montre comment marche la sauvegarde. Elle créé un fichier stocké dans le téléphone (pas dans la carte SD) dans lequel elle va sauvegarder les positions des sliders s'il on appuie sur bouton « sauvegarde ».

Lorsque l'on teste l'application en envoyant bien en texte trois commandes, les données se transfèrent bien mais le robot n'exécute pas les positions demandées. Il nous semblait qu'il ne comprenait pas ce qu'on lui envoyait. Nous en avons donc déduis que nous n'envoyions pas les commandes sous la bonne forme. Nous avons avons cherché à envoyer des entiers et finalement des octets. Seulement, lorsque nous avons testé l'envoie sous forme d'octets, les servomoteurs n'effectuaient toujours pas les angles souhaités et plantaient. Nous avons alors décidé d'utiliser l'Arduino comme intermédiaire stable pour envoyer et recevoir les données du module Bluetooth.

```
#include <SoftwareSerial.h>

SoftwareSerial hc06(0,1);

unsigned char message;

void setup () {
    Serial.begin(9600, SERIAL_8N1);
    hc06.begn(9600);

}

void loop(){

    if (hc06.available()){
        message = hc06.read();
        Serial.write(message);

    }
}
```

Illustration 7: Code C de test Bluetooth

Ce programme nous permet de regarder ce que recevait le module Bluetooth et donc ce qu'il était sensé renvoyer au port série de Roboto. Donc lorsqu'on envoyait une position de l'application au module elle s'affichait correctement sur le moniteur série.On en déduit que le problème ne venait pas de la transmission au module Bluetooth et de ce qu'il renvoyait. Nous avons ensuite regardé ce que recevait Roboto en se connectant via Minicom. Nous avons alors observé que les caractères qui apparaissaient étaient des caractères en ASCII, donc les octets que nous envoyions n'étaient clairement pas compris par le port série. Nous avons essayé de trouver une correspondance entre ce que l'on envoyait et les caractères que l'on recevait, mais il n'y en avait pas.

En parallèle pour tester un autre support, nous avons essayé d'adapter le programme icr.c sur windows (donc sans passer par Minicom), donc directement écrire dans le programme le port série où envoyer et la vitesse de transmission. Nous souhaitions rencontrer Mme Nadel afin qu'elle nous donne ses impressions sur les expressions que nous avions créé, comme nous en avions discuté au début du stage. Mais nous n'avons pas eu de réponses, nous lui avons donc renvoyé un mail avec directement les expressions en pièces jointes. Mais encore une fois pas de réponses donc nous

n’avons pas creusé plus loin la question d’envoyer directement le programme au robot (manque de connaissances en la matière et de temps).

Passage à une carte Pololu :

Nous sommes concentrés sur la transmission par Bluetooth, ainsi nous avons branché un servomoteur sur une carte Pololu, mis une alimentation et branché la carte à l’Arduino et à l’ordinateur. Puis nous avons utilisé *Pololu maestro control center* qui nous a permis de contrôler très facilement un servomoteur. Nous avons cherché à le faire bouger depuis l’Arduino grâce à un code trouvé sur Internet, dans lequel on envoyait au port série un texte qui permet d’envoyer les octets correspondant à la carte Pololu. En fusionnant notre programme fonctionnel du Bluetooth avec ce nouveau Programme, il nous était possible de contrôler le robot par Bluetooth.
Avec notre nouveau programme Arduino, il est dorénavant possible d’envoyer du texte qui est interprété et envoyé sous forme d’octet à la carte Pololu. Ainsi, on envoie simplement via l’application le numéro de la pin suivi de la position du slider. Le mouvement est donc possible. De même pour l’application Roboto Reader qui envoie sous forme de texte les différentes émotions pré-configurés. On envoie pour chaque émotion les pin des servomoteurs et leur positions entre 500 et 10 000 séparé d’un ‘x’ (pin x angle).

En contrôlant les servomoteurs, un problème inattendu s’est déclaré. Tous les servomoteurs « tremblotaient » de façon non contrôlée. En éliminant un par un les facteurs pouvant créer ce dysfonctionnement, nous nous sommes rendus compte que le problème venait en fait de l’alimentation (piles) qui était instable et sûrement trop faible.
En changeant l’alimentation pour une alimentation secteur, le problème se régla par lui même.
La dernière étape fut donc de faire définitivement les changements sur les robots, donc la fixation de la carte Arduino, de la carte Pololu, du module Bluetooth sur Roboto, et de la création d’un boîtier pour protéger et ranger les fils et les cartes.

Conclusion :

Ce stage fut pour nous une expérience intéressante et instructive. D’une part par son côté hardware et software. Et d’une autre part pour la souplesse dont nous avons dû faire preuve pour comprendre et adapter notre travail à un projet déjà existant.

En outre, nous pouvons tous affirmer que ce stage nous a permis d’acquérir de nouvelles connaissances en la matière. Ces expériences ne sont pas exactement les mêmes pour tout le groupe comme nous n’avons pas travaillé sur les mêmes points.

Néanmoins, si nous devions nous auto-évaluer, beaucoup de points seraient encore à affiner. Sur le côté hardware, le robot n’est pas encore tout-à-fait comme nous l’avions imaginé, de plus certaines réparations sont encore un peu sommaires. Sur le plan software, il nous manque une sauvegarde complètement opérationnelle ainsi qu’une optimisation de nos programmes (sliders et interfaces).

Nous trouvons cependant que ce stage se finit avec une note positive ; le robot étant fonctionnel et achevé. Notre objectif étant réalisé, nous sommes tous fiers du résultat.

Avec la création du GitHub à l'aide une petite fiche technique, ce projet pourrait être encore perfectionné,

dans les prochaines années, avec d'autres stagiaires.

Sincères remerciements :

-Philippe GAUSSIER
-Pierre ANDRY
-Sylvain COLOMER
-Kevin GOBIN

Bibliographie :

- Forum Convertisseur RS232-TTL : https://forum.arduino.cc/index.php?topic=527165.0
- https://www.aranacorp.com/fr/arduino-et-le-module-bluetooth-hc-06/
-Manuel de Linux
-*Programmation en Langage C* de Stephen Kochan
-MIT app inventor : https://appinventor.mit.edu/explore/
- Documentation SSC-II : https://www.seetron.com/docs/ssc2mnl.pdf

Annexe :

Figure 1 : Code contrôle et test de Roboto : (en pièce-jointe).

Figure 2: Un github est aussi existant: https://github.com/GeoffroyK/controle_tete_expressive dans lequel apparaît certains codes qui nous ont été utiles, dont celui là :

```ino
#include <SoftwareSerial.h>

SoftwareSerial hc06(2,3);

void setup(){
  //Initialize Serial Monitor
  Serial.begin(9600);
  Serial.println("ENTER AT Commands:");
  //Initialize Bluetooth Serial Port
  hc06.begin(9600);
}

void loop(){
  //Write data from HC06 to Serial Monitor
  if (hc06.available()){
    Serial.write(hc06.read());
  }

  //Write from Serial Monitor to HC06
  if (Serial.available()){
    hc06.write(Serial.read());
  }
}
```

Figure 1: Bluetooth Arduino

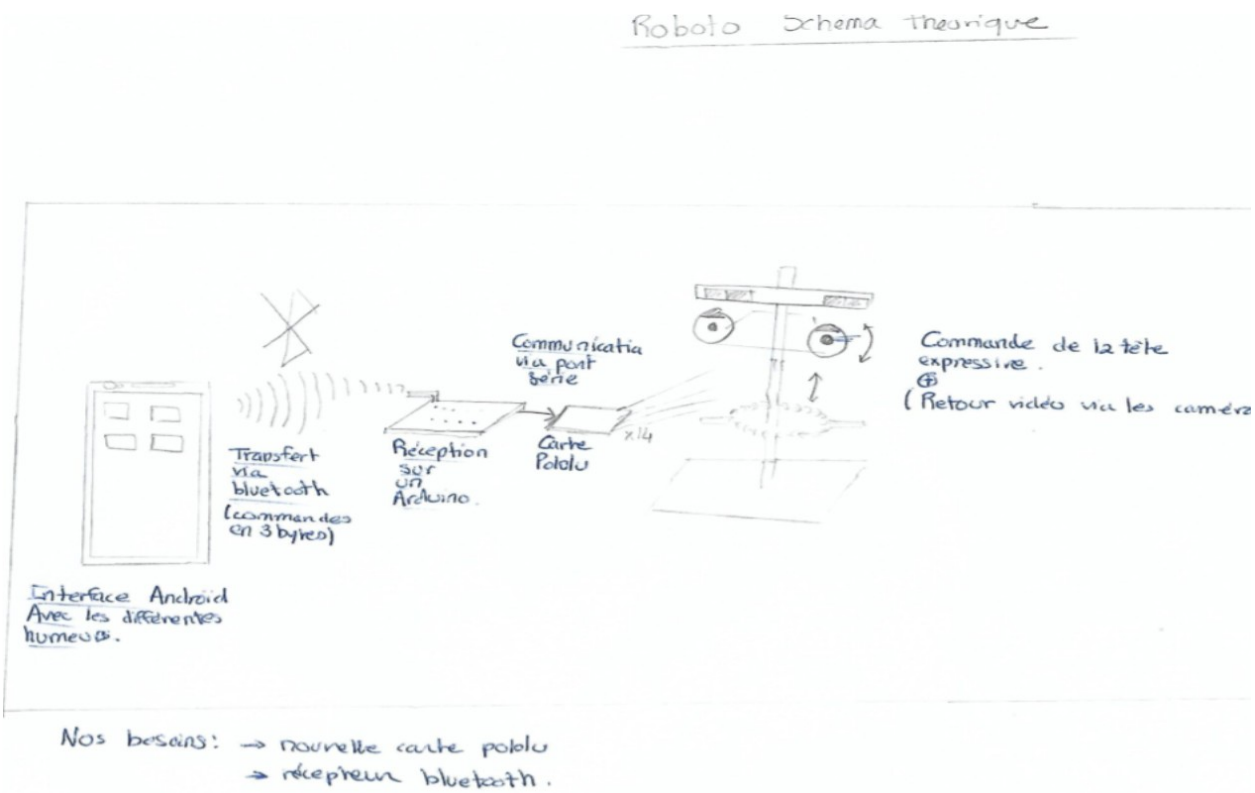


schéma explication
stage 2018-2019

Kévin Maria
BERNARD POUPET

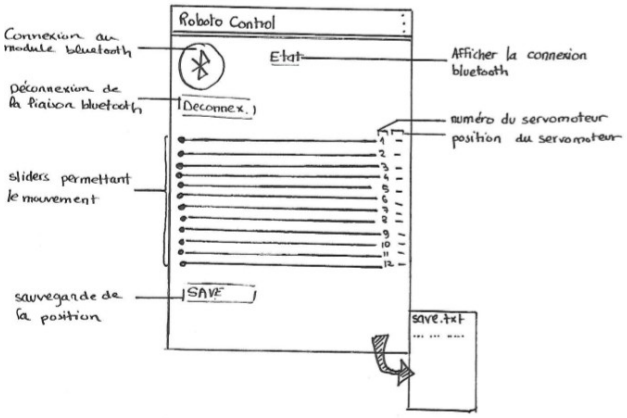
Geoffroy
KEIME

ROBOTO CTRL

sur Android

Options:
→ Contrôler 12 servomoteurs indépendamment. (Via slider)
→ Enregistrer la position de l'utilisateur dans un fichier externe

Vue d'ensemble:



schema explication
stage 2018-2019

Kévin Maria
BERNARD POUPET

Geoffroy
KEIME

ROBOTO READER

sur Android

Options:
→ Lire une position prédéfinie sur un fichier externe
→ Commander aux différents servomoteurs

Vue d'ensemble.