

ENSEMBLE SCOLAIRE JEAN XXIII

TIPE



Ilyes Achaq , Antoine Berviller , Roman Masse, Samira Ntsame

Années 2021-2023

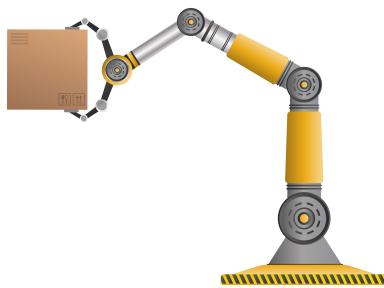
Table des matières

1 Présentation	2
1.1 <u>En quoi consiste le projet ?</u>	2
1.2 <u>Pourquoi ce projet ?</u>	2
2 Partie technique	3
2.1 <u>Cahier des charges</u>	3
2.2 <u>Frontière d'étude</u>	3
2.3 <u>Diagramme des cas d'utilisations</u>	3
2.4 <u>Diagramme des cas d'exigences</u>	4
2.5 <u>Chaîne fonctionnelle</u>	4
2.6 <u>Schéma cinématique</u>	4
	5
3 Etude scientifique	6
3.1 <u>Quel couple pour les servos-moteurs ?</u>	6
3.2 <u>Comment fonctionne le capteur geste ?</u>	6
4 Réalisation	7
4.1 <u>Tableaux comparatifs :</u>	7
4.1.1 <u>Capteurs :</u>	7
4.1.2 <u>Cartes arduinos :</u>	7
4.2 <u>Modélisation</u>	8
4.2.1 <u>Scénario/Explications</u>	8
4.2.2 <u>Voici d'autres points de vues de la maquette :</u>	8
4.2.3 <u>Modélisation Arduino Tinkercad</u>	9
5 Fabrication	11
5.1 <u>Montage robot</u>	11
5.2 <u>Fabrication maquette en carton</u>	12
5.3 <u>Fabrication maquette finit</u>	12
6 Code arduino & application mobile	13
6.1 <u>Code arduino</u>	13
6.1.1 <u>Distributeur</u>	13
6.1.2 <u>Robot</u>	13
6.2 <u>Application mobile</u>	18

1 Présentation

1.1 En quoi consiste le projet ?

Le projet consiste en la réalisation d'une maquette arduino de bras robot 3 axes. Le but étant de reproduire un robot qui pourrait aider des opérateurs à soulever/porter/déplacer des objets lourds en réalisant la même action répétée indéfiniment. Il pourra également réaliser des actions précises selon le besoin. Il sera facilement pilotable par l'opérateur.



1.2 Pourquoi ce projet ?

Nous pouvons donner plusieurs raisons importantes à la réalisation de ce robot :

— Réduction de la charge de travail des opérateurs :

Le bras robotisé permet de soulever et déplacer des objets lourds sans effort physique important de la part de l'opérateur, ce qui réduit la charge de travail et le risque de blessure.

À noter que 20% des accidents de travaux sont à cause d'une blessure au dos. De plus la durée moyenne d'arrêt de travail lors d'un accident lié à une lombalgie est de deux mois en moyenne.

— Amélioration de la productivité :

Le bras robotisé peut effectuer des tâches de manière répétitive et précise, ce qui peut améliorer la productivité et la qualité des produits finis.

— Flexibilité :

Le bras robotisé peut être programmé pour effectuer différentes tâches en fonction des besoins de production, ce qui le rend très flexible. Il peut réaliser ces tâches en continu.

— Sécurité :

Le bras robotisé peut être programmé pour éviter les collisions avec les objets ou les personnes, ce qui réduit le risque d'accidents.

— Contrôle en temps réel :

La maquette Arduino du robot permet de contrôler le bras robotisé en temps réel, ce qui facilite la programmation et le débogage.

2 Partie technique

2.1 Cahier des charges

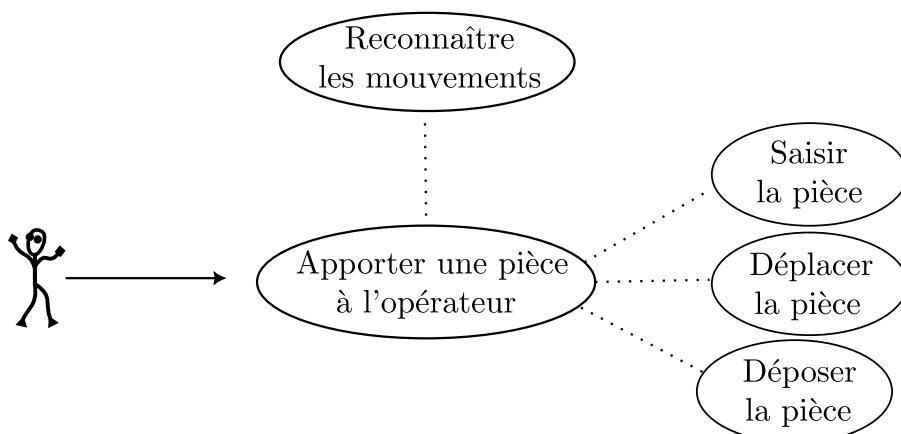
Le robot doit :

- Pouvoir être piloté à distance via application mobile.
- Pouvoir être piloté via une interface homme-machine (ex : capteur).
- Pouvoir communiquer à distance.
- Pouvoir porter et translater des objets.
- Avoir une grande mobilité.

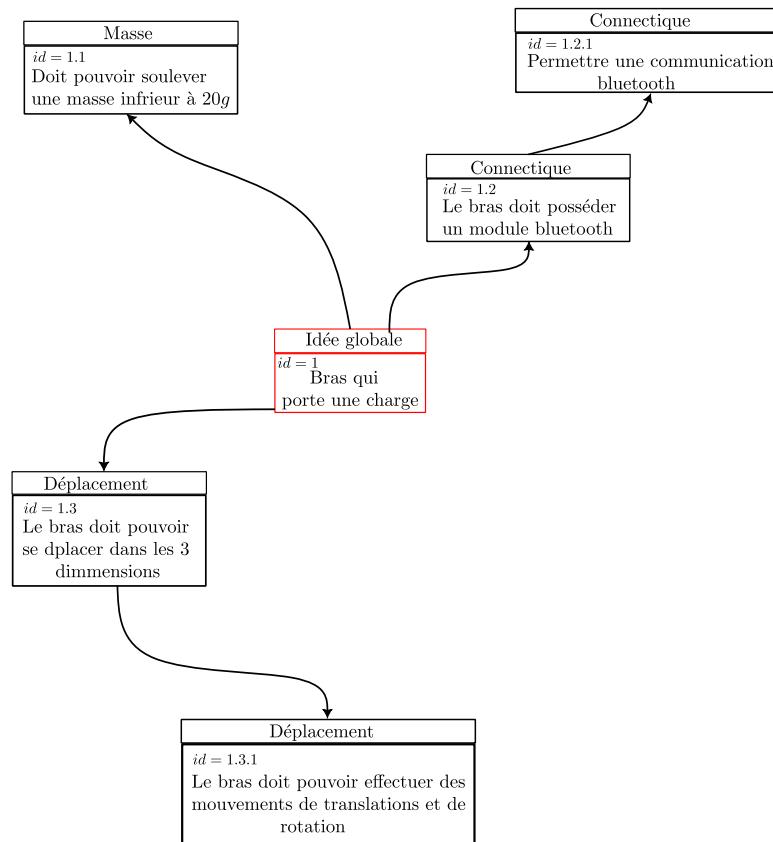
2.2 Frontière d'étude

Nous nous donnerons comme frontière pour la maquette un poids d'objet de maximum 20 grammes, ainsi qu'une portée du robot maximale de 15cm autour de l'axe z du robot. Le poids de l'objet sera calculé par la suite en fonction du couple des servo-moteurs.

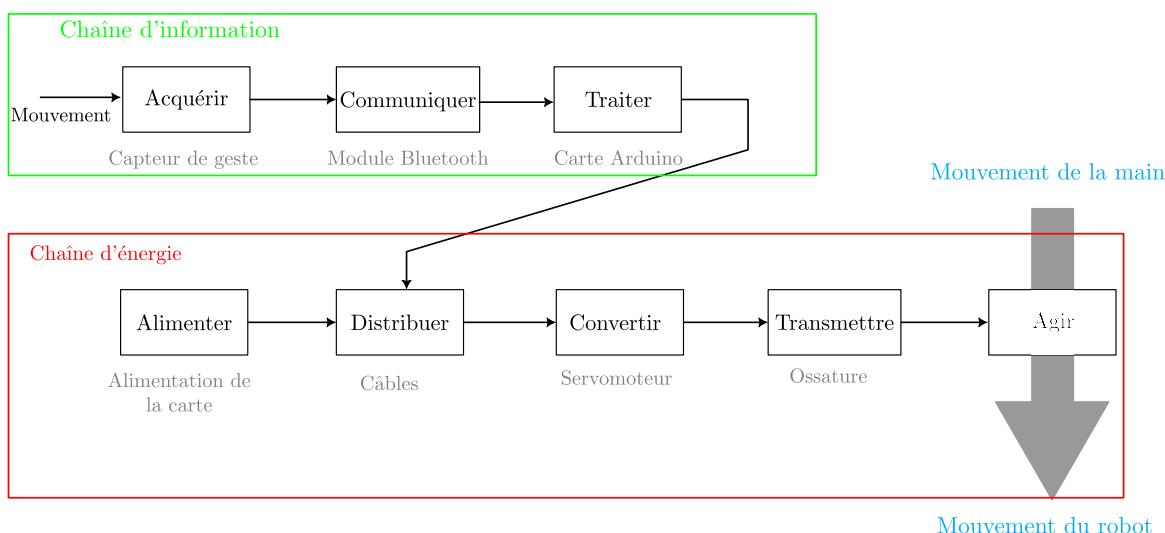
2.3 Diagramme des cas d'utilisations



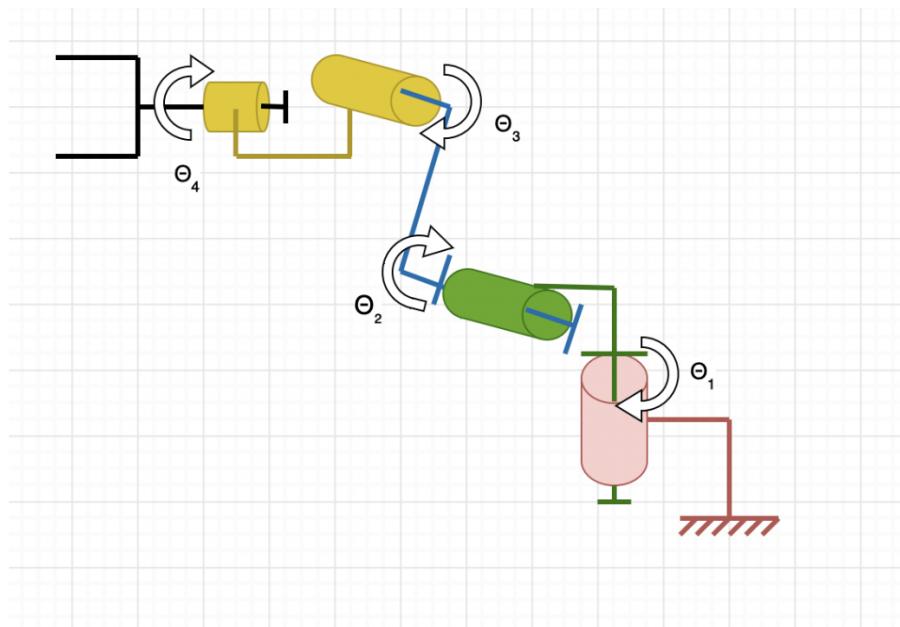
2.4 Diagramme des cas d'exigences



2.5 Chaîne fonctionnelle



2.6 Schéma cinématique

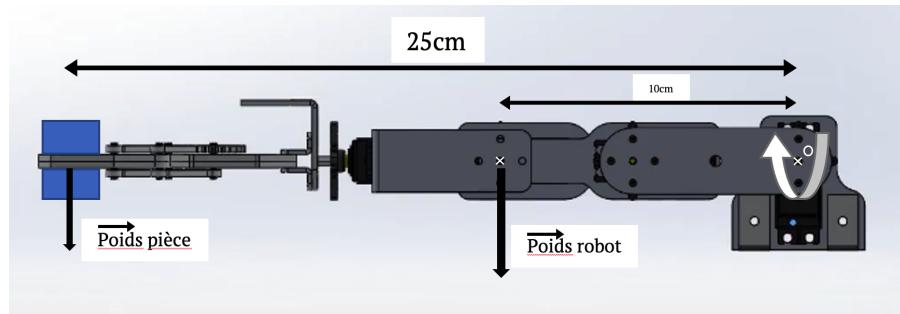


Remarquons que pour respecter le cahier des charges et avoir un robot 3 axes, il faut un minimum 3 servo-moteurs. Ici nous en utiliserons 4 puisque nous avons 4 angles. Ajoutons à celà le servo qui pilotera l'ouverture de la pince, 5 au total.

3 Etude scientifique

3.1 Quel couple pour les servos-moteurs ?

Afin que le robot puisse se relever en étant à l'horizontal avec une pièce, il faut un couple de servo adapté au poids de cette dernière.



Calculons le couple nécessaire du servo à la base du robot. Nous prendrons une pièce pesant 100 grammes

Le robot pèse 400 grammes.

Rappelons tout d'abord que :

$$\sum \overrightarrow{(\text{Moment forces})}_{/o} = \vec{0}$$

De plus, le moment du bras en 0 est :

$$\overrightarrow{\text{Moment}}_o(\vec{P}) = l \vec{P}$$

Avec l la longueur du bras de levier et P le poids du système. On a alors :

$$||\overrightarrow{\text{Moment}}_o(P_{bras})|| = 0.10 \times (0.4 \times 9.81) = 0.4 N.m^{-1}$$

$$||\overrightarrow{\text{Moment}}_o(P_{pièce})|| = 0.25 \times (0.100 \times 9.81) = 0.25 N.m^{-1}$$

D'après le théorème du moment statique on obtient :

$$|C| = 0.4 + 0.25 = 0.65 N.m^{-1}$$

On prendra des servos de 2Nm de couple pour être sûr.

4 Réalisation

4.1 Tableaux comparatifs :

4.1.1 Capteurs :

Nom et image	Description	Commentaires
Bracelet Myo 	- Compatible Bluetooth 4.0 - Permet de paramétrer des mouvements de base que l'objet exécutera quand le bracelet les détectera	- Manque de précision - Relativement cher (~149\$)
Capteur de gestes 101020083 	- Capteur spécialisé PAJ7620U2 intégré - Reconnaît jusqu'à 9 mouvements de la main	- Module Grove, compatible Arduino - Mode et nombre de détection idéaux
Joystick 101020028 	- 2 potentiomètres linéaire de 10k Ohms - Fonction bouton poussoir	- Module Grove, compatible Arduino - Bonne alternative si panne du capteur de gestes

Le capteur se présentant comme idéal pour notre projet est le 2nd, le paj76. Il permet de reconnaître 6 gestes. Nous choisirons celui-ci.

Nous utiliserons en plus un capteur infrarouge (présence / distance) qui nous est donné.

Pour ce qui est du module bluetooth, on se servira d'un module Grove.

4.1.2 Cartes arduinos :

	GPIO	PINS NUMÉRIQUE	PINS ANALOGIQUES	MÉMOIRE PROGRAMMABLE	PRIX
	20	14	6	32kb et 2kb de RAM	25,40 €
	20	14	6	32kb et 2kb de RAM	31,70 €
	54	38	16	256kb et 8kb de RAM	50,40 €

La carte Arduino classique est la plus adaptée.

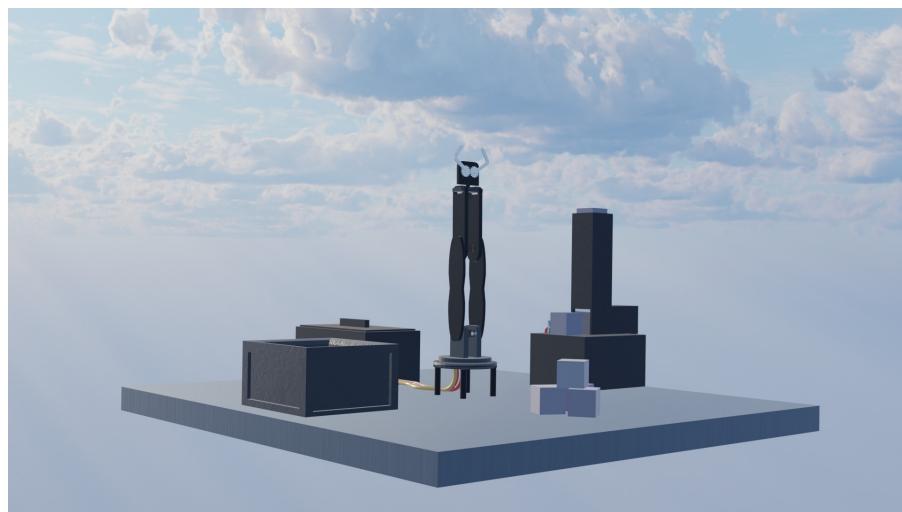
Le choix des composants est fait. Récapitulons, nous utiliserons :

- Deux cartes arduinos.
- Un capteur geste paj76.
- Un capteur infrarouge.
- Un module Grove bluetooth.
- 5 servos.

4.2 Modélisation

Avant de commencer la fabrication de la maquette, il est important de représenter le projet en 3d afin de bien visualiser les choses.

Pour cela, nous réalisons une modélisation sur Blender :



4.2.1 Scénario/Explications

Le robot est présent au milieu d'un socle, qui sera le domaine.

À droite, le pavé portant une tour remplie de bloc servira de distributeur pour le robot.

Un capteur infrarouge est placé au pied de cette tour de sorte à ce que dès qu'un bloc est pris par le robot, un autre est distribué. En tout il y a 5 bloc dans la tour.

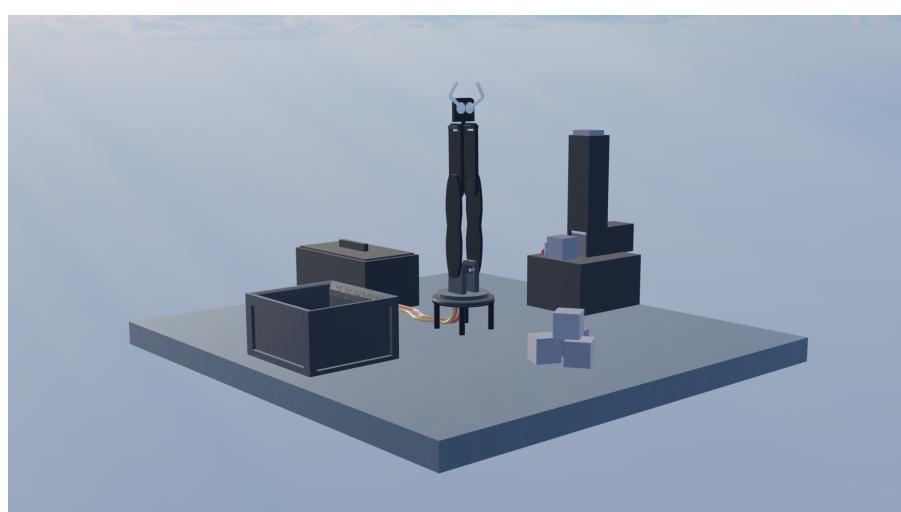
Deux cartes arduinos sont utilisées, une pour piloter le robot et une autre pour le distributeur.

Le capteur geste Paj76, placé à l'arrière de la boîte du fond permet de contrôler le robot tel que :

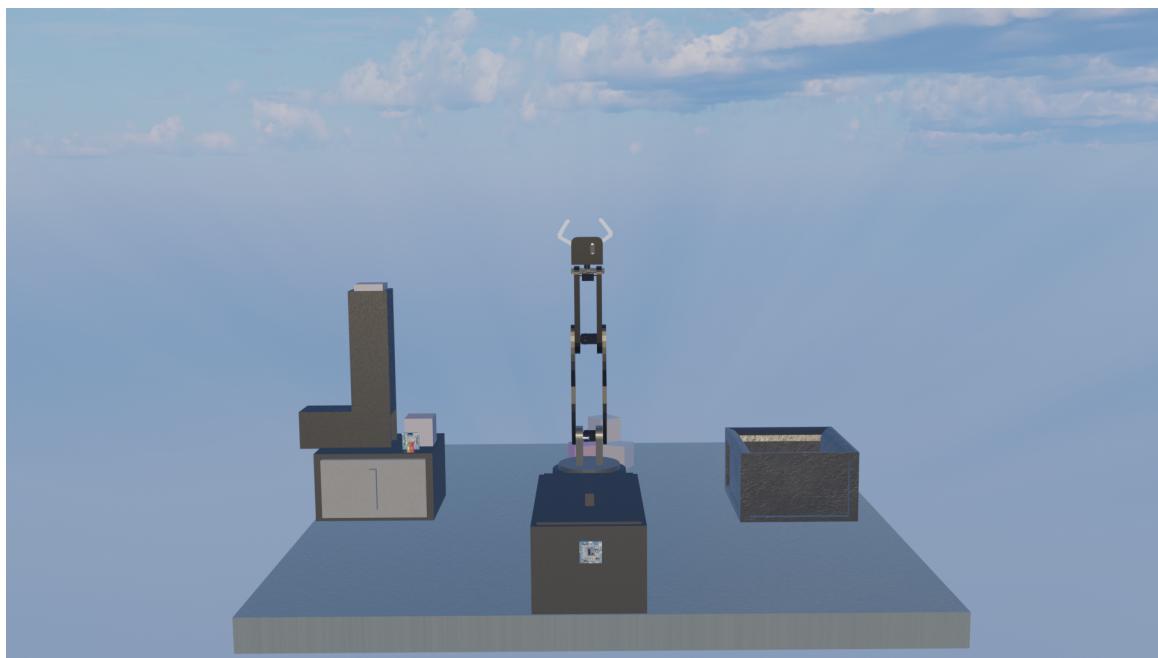
Si l'on fait un mouvement vers la gauche, le robot prendra un bloc. Puis si on fait un mouvement vers le haut et vers la droite, le robot ira déposer le bloc dans la cuve de gauche.

Tandis que le robot est pilotable avec le capteur, il est en même temps pilotable via bluetooth sur une application mobile.

4.2.2 Voici d'autres points de vues de la maquette :

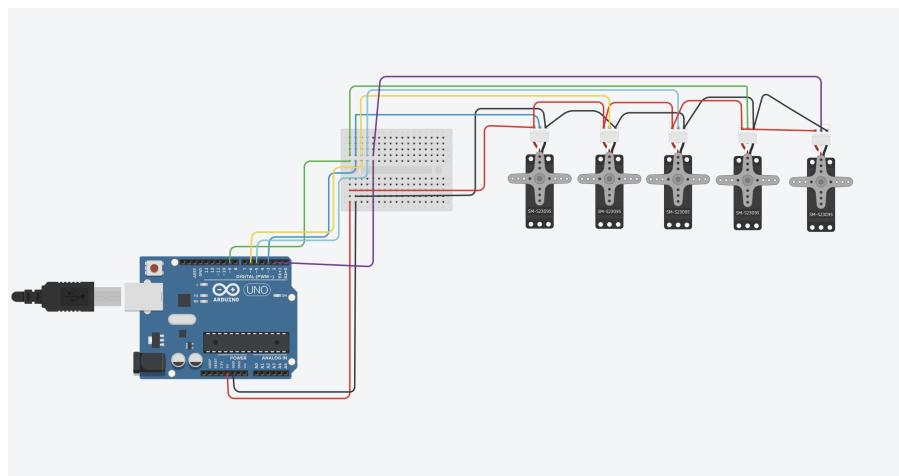


Vue de derrière (on aperçoit le capteur geste dans la boîte du milieu, ainsi que le capteur infrarouge au pied de la tour) :



4.2.3 Modélisation Arduino Tinkercad

La modélisation arduino à l'aide de tinkercad est utile afin de comprendre comment arduino fonctionne. Il s'agit d'une première prise en main. Nous faisons un montage de 5 servos et essayons de les piloter :



Dans la partie code on essaye de piloter les servos, d'en faire bouger certains puis d'autres. On essaye également les capteurs dans l'interface tinkercad. Voici un exemple de notre code de prise en main :

```

1 #include <SoftwareSerial.h> // TX RX software library for bluetooth
2
3 #include <Servo.h> // servo library
4 Servo servo1;
5 Servo servo2;
6 Servo servo3;
7 Servo servo4;
8 Servo servo5;
9
10 int bluetoothTx = 10; // bluetooth tx to 10 pin

```

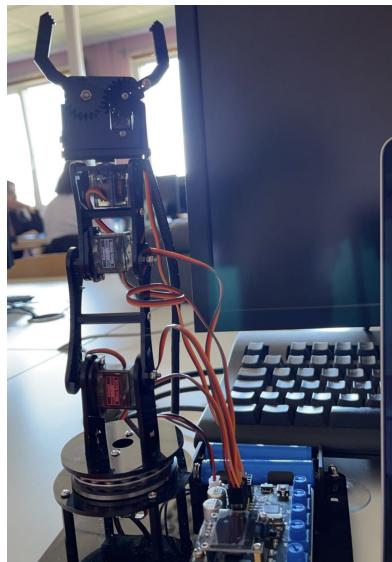
```
11 int bluetoothRx = 11; // bluetooth rx to 11 pin
12 int servoPos1, servoPos2, servoPos3, servoPos4, servoPos5;
13 int servoPPos1, servoPPos2, servoPPos3, servoPPos4, servoPPos5;
14
15 SoftwareSerial bluetooth(bluetoothTx, bluetoothRx);
16
17 void setup()
18 {
19     servo1.attach(3);
20     servo2.attach(6);
21     servo3.attach(5);
22     servo4.attach(9);
23     servo5.attach(2);
24     //Setup Bluetooth serial connection to android
25     bluetooth.begin(38400);
26     bluetooth.setTimeout(1);
27     // position initiale du robot:
28     servoPPos1 = 1;
29     servo1.write(servoPPos1);
30     servoPPos2 = 1;
31     servo2.write(servoPPos2);
32     servoPPos3 = 1;
33     servo3.write(servoPPos3);
34     servoPPos4 = 1;
35     servo4.write(servoPPos4);
36     servoPPos5 = 1;
37     servo5.write(servoPPos5);
38 }
39 void loop()
40 {
41
42 {
43     myservo.write(180); //va a 15 degres
44     delay (2000); //attendre 2 secondes
45     myservo.write(90); //va a 30 degres
46     delay (2000); //attendre 2 secondes
47     myservo.write(0); //va a 45 degres
48     delay(2000); //attendre 2 secondes
49 }
50 }
```

Un code assez simple. Maintenant que nous avons modéliser entièrement le projet, nous pouvons passer à la fabrication.

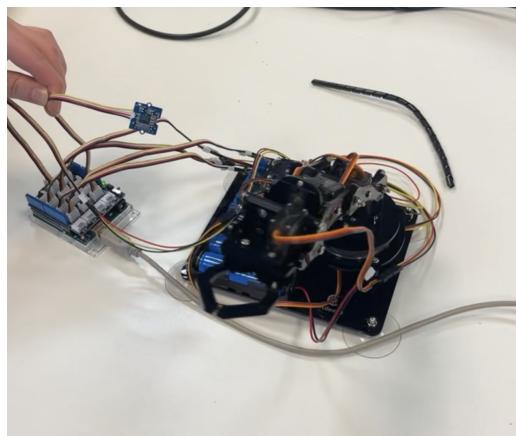
5 Fabrication

5.1 Montage robot

En premier lieu, nous devons monter le robot. Il est livré avec une carte arduino et un socle que nous n'utiliserons pas. Nous utiliserons simplement la structure :



Nous remplaçons la carte fournit par la carte arduino programmable que nous avons choisi.



5.2 Fabrication maquette en carton

Pour avoir les bonnes dimensions du projet final nous essayons d'abord une maquette en carton afin de voir si cela nous convient. Nous démontons le robot de son socle d'origine afin de ne récupérer uniquement son ossature. On le fixe sur un nouveau socle plus grand. On fait une maquette du distributeur. Pour pouvoir servir des blocs, nous utiliserons un système de piston. En effet nous devons convertir une rotation du servo, en une translation.

L'idée est que dans la boîte en dessous de la tour, nous mettrons un servo qui sera relié à un chariot poussoir (qui poussera le bloc en dehors de la tour de stockage). De ce fait, lorsque le servo fait une rotation de π , le poussoir avance et pousse un bloc en dehors de la tour.

Voici le système dans la boîte TEST :



Sur la figure de droite, vous pouvez voir le servo relié au poussoir du dessus par deux segments qui peuvent pivoter au niveau de leur liaison. Nous rencontrons des problèmes à cause du carton, en effet il y a trop de frottement dans la tour et les cubes ne peuvent pas bien descendre.

5.3 Fabrication maquette finit

Après de nombreux essais et de restructuration de la maquette en carton, elle nous convient finalement. Nous prenons alors ses dimensions puis nous la modélisons sur Autodesk avant de l'imprimer en 3d. Voici le résultat :



6 Code arduino & application mobile

6.1 Code arduino

6.1.1 Distributeur

Nous voulons qu'à chaque fois le robot prend un cube, le distributeur en redonne un. Autrement dit dès que le capteur infrarouge ne détecte plus la présence du bloc devant lui, on envoie la commande au servo de tourner puis de revenir à sa position initiale. Voici le code :

```

1 #include <Servo.h>
2 #include <Wire.h>
3 Servo servo1;
4 int pos, pos1, pos2, nbrcube;
5 void setup() {
6     servo1.attach(2);
7     nbrcube=0;
8     servo1.write(13); // position initiale du servo, quand le poussoir est au fond.
9     Serial.begin(9600);
10    pinMode(SCL, INPUT);
11
12 }
13
14
15 void loop() {
16     short val=0;
17     val=digitalRead(SCL);
18     Serial.println((int)val);
19     delay(1000);
20     if (nbrcube!=5){ // tant que 5 bloc n'ont pas ete distribue.
21         if (val==1){ // si le capteur infrarouge ne detecte plus le bloc.
22             servo1.write(150); // servo tourne au max, poussoir devant.
23             delay(1000); // attend 1 sec
24             servo1.write(13); // remet le servo dans sa pos initial.
25             delay(2000);
26             nbrcube +=1;
27         }
28     }
29 }
30
31
32 }
33

```

6.1.2 Robot

Cette fois-ci le code est plus compliqué, en premier lieu nous mettons les servos à leur position intial, lorsque le robot est debout. Dans la même boucle il y a le capteur geste et le module bluetooth ce qui fait que le robot peut être piloter en même temps par les gestes ou par le téléphone. Le code est assez long, la partie la plus intéressante est dans le "voidloop".

Voici le code :

```

1 #include <paj7620.h>
2
3 #include <paj7620.h>
4 #include <Servo.h>
5 #include <Wire.h>
6 #include <SoftwareSerial.h>
7 #define GESREACTIONTIME 500 //LIBRAIRIES
8 #define GESENTRYTIME 800 // DEFINITIONS DES VARIABLES/ SERVO
9 #define GESQUITTIME 1000
10 Servo servo1;
11 Servo servo2;

```

```

12 Servo servo3;
13 Servo servo4;
14 Servo servo5;
15 int servoPos1, servoPos2, servoPos3, servoPos4, servoPos5;
16 int X;
17
18 SoftwareSerial bluetooth(8, 9);
19 char instruction;
20
21
22 void setup()
23 {
24     servoPos1 = 90;           // PARTIE SERVO POSITION INITIAL SETUP
25     servoPos2 = 80;
26     servoPos3 = 90;
27     servoPos4 = 64;
28     servoPos5 = 90;
29     X = 0;
30     servo1.attach(2);
31     servo2.attach(3);
32     servo3.attach(4);
33     servo4.attach(5);
34     servo5.attach(6);
35     servo1.write(servoPos1);
36     servo2.write(servoPos2);
37     servo3.write(servoPos3);
38     servo4.write(servoPos4);
39     servo5.write(servoPos5);
40     uint8_t error = 0;
41
42     bluetooth.begin(9600);
43
44     Serial.begin(9600);
45     Serial.println("\nPAJ7620U2 TEST DEMO: Recognize 9 gestures.");
46
47     error = paj7620Init();           //NE PAS TOUCHER SETUP DU GESTURE
48     if (error)
49     {
50         Serial.print("INIT ERROR, CODE:");
51         Serial.println(error);
52     }
53     else
54     {
55         Serial.println("INIT OK");
56     }
57     Serial.println("Please input your gestures:\n");
58 }
59
60 void loop()
61 {
62
63     uint8_t data = 0, data1 = 0, error;           // PARTIE GESTES
64     error = paj7620ReadReg(0x43, 1, &data);
65     if (!error)
66     {
67         switch (data)
68     {
69         case GES_RIGHT_FLAG:
70             delay(GES_ENTRY_TIME);
71             paj7620ReadReg(0x43, 1, &data);
72             if(data == GES_FORWARD_FLAG)
73             {
74                 Serial.println("Forward");
75                 delay(GES_QUIT_TIME);           // DETECTE QUAND CA AVANCE
76             }
77             else if(data == GES_BACKWARD_FLAG)

```

```

78     {
79         Serial.println("Backward");
80         delay(GES_QUIT_TIME);           // DETECTE QUAND CA RECULE
81     }
82     else
83     {
84         Serial.println("Right");      // DETECTE QUAND DROITE
85         if (X==0){                  // COMMANDE LORSQUE MOUV DROIT
86             servoPos1 = 0;
87             servo1.write(servoPos1);
88             delay(500);
89             servoPos2 = 45;
90             servo2.write(servoPos2);
91             delay(500);
92             servoPos3 = 160;
93             servo3.write(servoPos3);
94             delay(500);
95             servo5.write(30);
96             X+=1;
97         }
98
99
100    }
101
102    break;
103    case GES_LEFT_FLAG:
104    delay(GES_ENTRY_TIME);
105    paj7620ReadReg(0x43, 1, &data);
106    if(data == GES_FORWARD_FLAG)
107    {
108        Serial.println("Forward");
109        delay(GES_QUIT_TIME);
110    }
111    else if(data == GES_BACKWARD_FLAG)
112    {
113        Serial.println("Backward");
114        delay(GES_QUIT_TIME);
115    }
116    else
117    {
118        Serial.println("Left");       // DETECTE QUAND GAUCHE
119        if (X==0){                  // COMMANDE LORSQUE MOUV GAUCHE
120            servoPos1 = 180;
121            servoPos2 = 40;
122            servoPos3 = 155;
123            servoPos4 = 64;
124            servoPos5 = 57;
125            servo1.write(servoPos1);
126            delay(500);
127            servo5.write(servoPos5);
128            delay(1000);
129            servo2.write(servoPos2);
130            delay(500);
131            servo3.write(servoPos3);
132            servo4.write(servoPos4);
133            X+=1;
134        }
135    }
136
137
138
139
140    }
141    break;
142    case GES_UP_FLAG:
143    delay(GES_ENTRY_TIME);

```

```

144     paj7620ReadReg(0x43, 1, &data);
145     if(data == GES_FORWARD_FLAG)
146     {
147         Serial.println("Forward");
148         delay(GES_QUIT_TIME);
149     }
150     else if(data == GES_BACKWARD_FLAG)
151     {
152         Serial.println("Backward");
153         delay(GES_QUIT_TIME);
154     }
155     else
156     {
157         Serial.println("Up");           // DETECTE QUAND MONTE
158         servoPos1 = 90;             // COMMANDÉ LORSQUE MOUV VERS LE HAUT
159         servoPos2 = 80;
160         servoPos3 = 90;
161         servoPos4 = 64;
162         servoPos5 = 75;
163
164         servo5.write(servoPos5);
165         delay(1000);
166         servo2.write(servoPos2);
167         delay(300);
168         servo3.write(servoPos3);
169         servo4.write(servoPos4);
170         delay(500);
171         servo1.write(servoPos1);
172         X=0;
173
174     }
175     break;
176 case GES_DOWN_FLAG:
177     delay(GES_ENTRY_TIME);
178     paj7620ReadReg(0x43, 1, &data);
179     if(data == GES_FORWARD_FLAG)
180     {
181         Serial.println("Forward");
182         delay(GES_QUIT_TIME);
183     }
184     else if(data == GES_BACKWARD_FLAG)
185     {
186         Serial.println("Backward");
187         delay(GES_QUIT_TIME);
188     }
189     else
190     {
191         Serial.println("Down");
192     }
193     break;
194 case GES_FORWARD_FLAG:
195     Serial.println("Forward");
196     delay(GES_QUIT_TIME);
197     break;
198 case GES_BACKWARD_FLAG:
199     Serial.println("Backward");
200     delay(GES_QUIT_TIME);
201     break;
202 case GES_CLOCKWISE_FLAG:
203     Serial.println("Clockwise");
204     break;
205 case GES_COUNT_CLOCKWISE_FLAG:
206     Serial.println("anti-clockwise");
207     break;
208 default:

```

```

210     paj7620ReadReg(0x44, 1, &data1);
211     if (data1 == GES_WAVE_FLAG)
212     {
213         Serial.println("wave");
214     }
215     break;
216 }
217 delay(100);
218
219
220
221
222
223 if (bluetooth.available()) {
224     instruction = bluetooth.read(); // PARTIE BLUETOOTH
225     Serial.println(instruction);
226 }
227
228 if (instruction == 'R') { // COMMANDE DU TELEPHONE RIGHT/DROIT
229     if (X==0){
230
231         servoPos1 = 0;
232         servo1.write(servoPos1);
233         delay(500);
234         servoPos2 = 45;
235         servo2.write(servoPos2);
236         delay(500);
237         servoPos3 = 160;
238         servo3.write(servoPos3);
239         delay(500);
240         servo5.write(30);
241         instruction = 'rien';
242         X+=1;
243     }
244
245
246
247 }
248 if (instruction == 'L'){ // COMMANDE DU TELEPHONE LEFT/GAUCHE
249     if (X==0){
250
251         servoPos1 = 180;
252         servoPos2 = 50;
253         servoPos3 = 155;
254         servoPos4 = 64;
255         servoPos5 = 57;
256         servo1.write(servoPos1);
257         delay(500);
258         servo5.write(servoPos5);
259         delay(1000);
260         servo2.write(servoPos2);
261         delay(500);
262         servo3.write(servoPos3);
263         servo4.write(servoPos4);
264         delay(500);
265         X+=1;
266     }
267 }
268
269
270 }
271 if (instruction == 'U'){ // COMMANDE DU TELEPHONE UP/HAUT
272     servoPos1 = 90;
273     servoPos2 = 80;
274     servoPos3 = 90;
275     servoPos4 = 64;

```

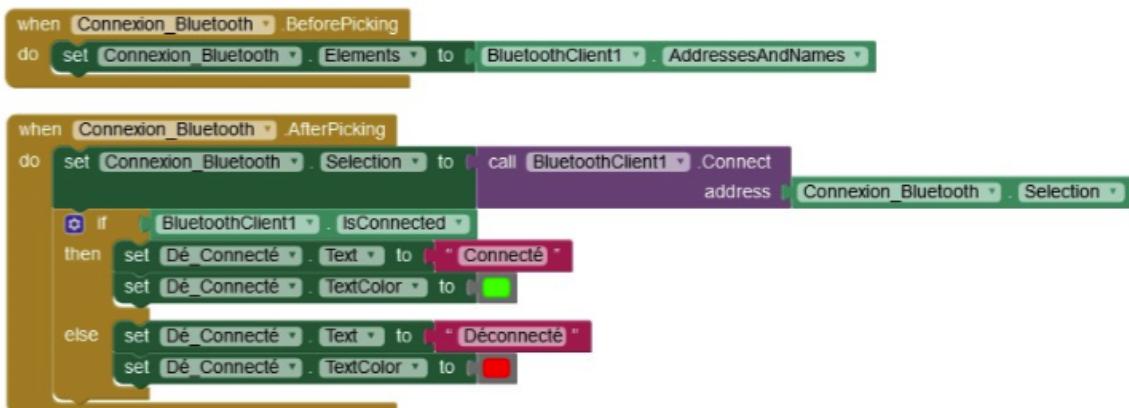
```
276     servoPos5 = 75;  
277  
278     servo5.write(servoPos5);  
279     delay(1000);  
280     servo2.write(servoPos2);  
281     delay(300);  
282     servo3.write(servoPos3);  
283     servo4.write(servoPos4);  
284     delay(500);  
285     servo1.write(servoPos1);  
286     X=0;  
287 }  
288  
289 }  
290 }
```

Vous pouvez remarquez au début de chaque commande de servo il y a une condition notée X, en fait il s'agit d'une sécurité que nous avons mit afin d'éviter que par accident nous faisons un geste vers la gauche alors que le robot est en position droite, cela évite qu'il fasse un rotation de 180 degré sans passer par la position haute.

6.2 Application mobile

L'application mobile est réalisée avec app inventor. Cela se présente sous formes de code en bloc.

initialisation bluetooth :



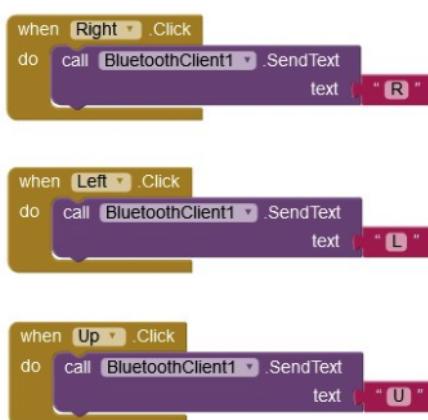
Explications :

Avant d'avoir sélectionné un élément de la liste de "Connexion Bluetooth", c'est-à-dire lorsqu'on clique sur "Connexion Bluetooth", la liste des appareils émettant un signal Bluetooth apparaît (avec leur nom et adresse)

Après avoir sélectionné un élément de la liste de "Connexion Bluetooth", le Client Bluetooth de l'application est appelé et connecte l'appareil au module Bluetooth sélectionné.

Si l'appareil est connecté en Bluetooth, alors il sera indiqué "Connecté" en vert sur l'application. Dans le cas contraire, il sera indiqué "Déconnecté" en rouge.

Contrôle du bras robot :



- Lorsqu'on clique sur le bouton "Right" ("Déposer le cube"), l'application envoie en Bluetooth la lettre "R".

- Lorsqu'on clique sur le bouton "Left" ("Prendre le cube"), l'application envoie en Bluetooth la lettre "L".

- Lorsqu'on clique sur le bouton "Up" ("Lever le bras"), l'application envoie en Bluetooth la lettre "U".

Remarquez que dans le code arduino en fonction de la lettre reçue on associe un mouvement de servo (ligne 228)

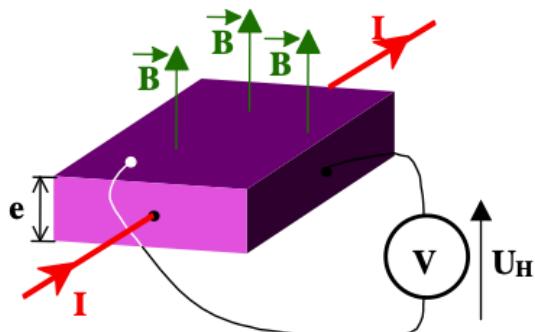
Enfin, voici à quoi ressemble l'application :



Etude du fonctionnement d'un capteur

Effet hall

Un barreau de semi-conducteur soumis à un champ magnétique uniforme B et traversé par un courant I est le siège d'une force électromotrice U sur 2 faces :



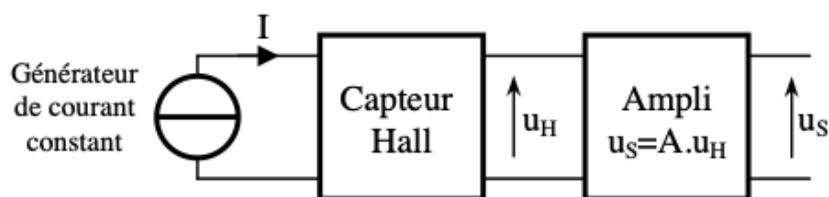
On note U_H la tension de Hall :

$$U_H = R_H \frac{I \cdot B}{e}$$

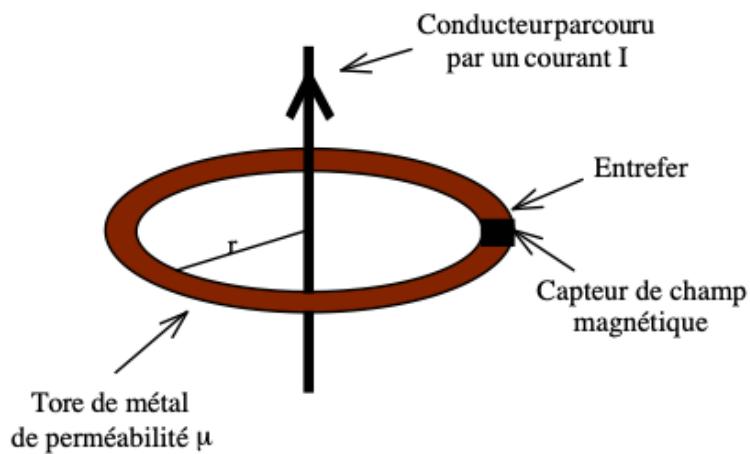
Avec R_H = Constante de Hall (dépend du conducteur)

Remarquons qu'en maintenant le courant constant, on a $U_H = k \cdot B$, donc la tension sera proportionnelle au champ B .

Pour capter un champ magnétique on peut alors réaliser le montage :



Ce type de capteur permet de nombreuses applications dont la mesure d'une intensité électrique (montage vu en cours) :



Le courant I crée un champ magnétique proportionnel à ce courant (voir cour) :

$$B = \frac{\mu \cdot I}{2\pi \cdot r}$$

Le capteur donne une tension $U_s = k \cdot B = k' \cdot I$ k et k' constantes.