

Ergo_JR marin















Etape 1

Objectifs

- Découverte du projet
- Bilan des positions homme
- Choix du porte outil sur Ergo_JR
- Association positions homme / positions robot

Exemple de travaux réalisables

Bilan des positions homme

Numéro	0	1	2	3	4	5	6
Bras droit							
Lettres	Z	F,G	A,K,L,M,N	B,H,O,P,Q,R,S	C,T,U,Y,ANN	D,I,NUM,J,V	E,W,X
Bras Gauche							
Lettres	H,I	A,B,C,D	G,N,S,ANN,V,X	J,F,M,R,Y,W,Z	E,L,Q,U,NUM	K,P,T	O

Attention : deux signaux de service sont identiques. On pourra les distinguer par leur durée.

- EOW End Of Word

Position : Right 1 Left 1 pause longue

- Double letter

Position : Right 1 Left 1 pause courte

Etape 2

Objectifs :

- Fabriquer le drapeau
- Repérer les positions robots/valeurs moteurs
- Écrire un bloc



Position0

- qui positionne le robot dans la position 0
- Généraliser en créant les blocs pour les autres positions



Position1

...



Position2

...

- Tester l'enchaînement successif des diverses positions

Etape 3

Pré-requis :


On suppose que l'étape 2 est TOTALEMENT validée

- Tous les blocs PositionX sont codés






- Que le robot passe sans encombre d'une position à l'autre.

Objectifs :

- Réaliser le bloc 
- qui pose une question et permet à l'utilisateur de saisir une phrase :



- Créer maintenant un bloc  qui devra :

1. Utiliser le bloc  qui affecte la variable 
2. Pour chaque mot de la phrase et pour chaque lettre de la phrase afficher pendant 2 s la lettre à



envoyer



Etape 4

Objectifs :

Repérer la position que doit prendre le robot en fonction des lettres de l'alphabet.

- Compléter votre tableau en indiquant pour chaque position les lettres correspondantes
- Créer une variable `caractere` qui contiendra la liste des 26 lettres de l'alphabet.
- Créer une variable `codeD` qui contiendra une liste des 26 positions du bras droit

`list 2 3 4 5 6 1 1 3 5 5 2 2 2 2 3 3 3 3 3 4 4 5 6 6 4 0`

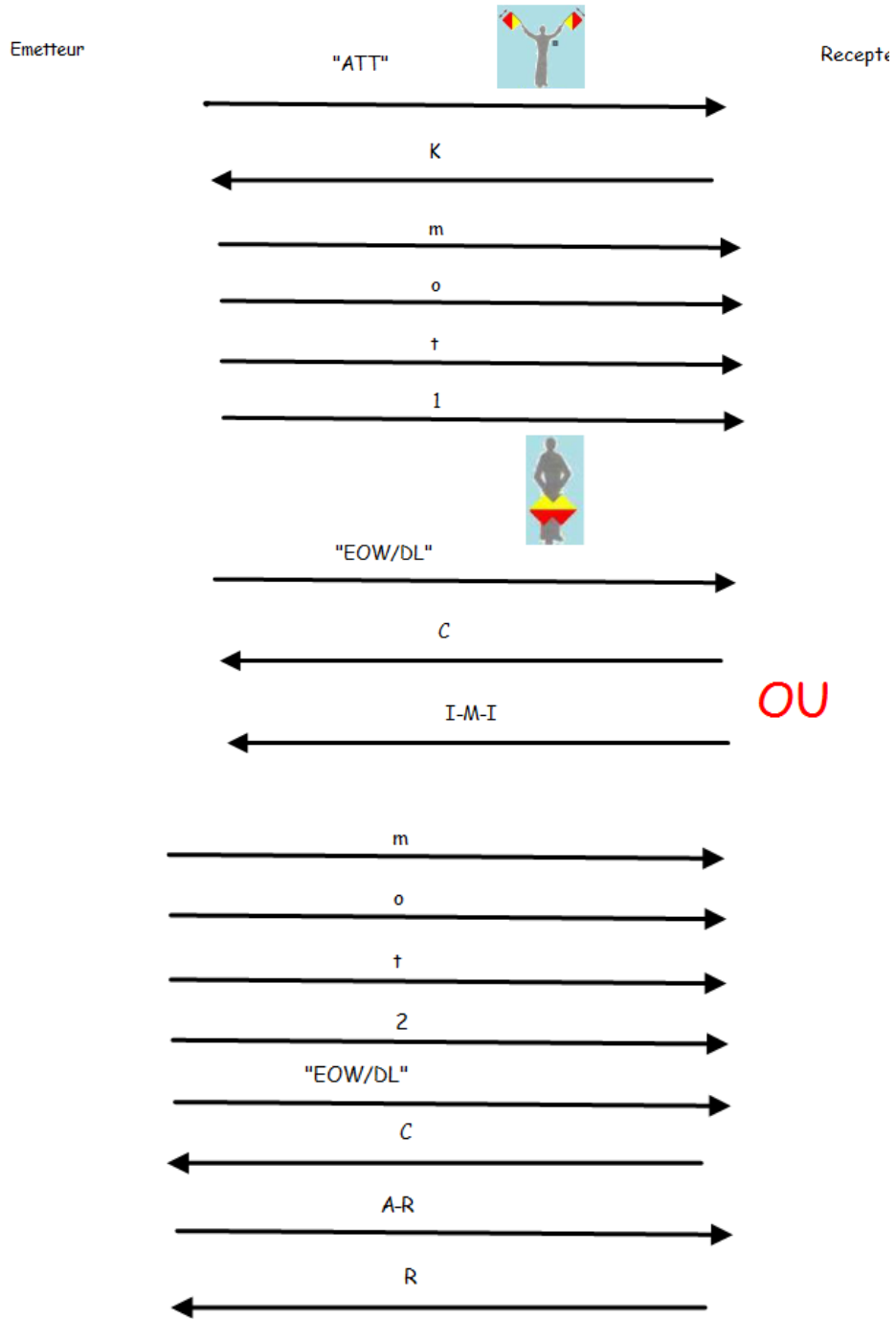
Numéro	0
Bras droit	
Lettres	Z
Bras Gauche	
Lettres	H,I

- Créer la procédure `InitEncodD` qui initialise la table `caractere` et `codeD`
- Créer les blocs qui positionnent le robot pour les signaux de service. « ATT » (Attention) et « EOW/DL » End off word ou « DL» letter double..

Noter que EOW et DL correspondent à une position 1 plus ou moins longue.

Etape 5

Objectifs : Dessiner un diagramme du protocole de communication
Exemple de travaux réalisables



Etape 6

Il s'agit maintenant de coder tout le protocole pour qu'il réponde diagramme précédemment établi. Vous trouverez ci-dessous des blocs que vous devrez adapter aux programmes que vous avez déjà réalisés. Attention : Il ne s'agit pas de recopier ces blocs sans comprendre leur rôle dans la réalisation du protocole. C'est la raison pour laquelle quelques erreurs sont présentes qu'il vous faudra corriger.

```

+Main+
Initialisation
script variables E_R
set E_R to EmitOrReceive
if E_R = E
  Emitter
  Repos
else
  if E_R = R
    Receiver
    Repos
  else
    think Hmm... for 2 secs
  
```

```

+Initialisation+
InitEncodD
set motor(s) all motors stiff
Repos
  
```

```

+InitEncodD+
set caractere to list A
for i = 1 to 26
  add unicode i + 65 as letter to caractere
set codeD to
list 2 3 4 5 8 1 1 3 5 5 2 2 2 2 3 3 3 3 3 4 4 5 6 6 4 0
  
```

caractere	
1	A
2	B
3	C
4	D
5	E
6	F
7	G
8	H
9	I
10	J
11	K
12	L
13	M
14	N
15	O
16	P
17	Q
18	R
19	S
20	T
21	U
22	V
23	W
24	X
25	Y
26	Z

codeD	
1	2
2	3
3	4
4	5
5	6
6	1
7	1
8	3
9	5
10	5
11	2
12	2
13	2
14	2
15	3
16	3
17	3
18	3
19	3
20	4
21	4
22	5
23	6
24	6
25	4
26	0

```

+ EmitOrReceive +
ask Error? and wait
report answer
    
```

```

+Emitter+
repeat until key r pressed?
  ATTD
  Transmit
    
```

```

+ATTD+
repeat 1
  ATTD1
  ATTD2
    
```

Boucle qui pourra être ajustée par la suite

```

+ATTD1+
for i = 1 to 8
  set position(s) item i of list -90 0 80 0 -90 70 of motor(s)
  item i of list m1 m2 m3 m4 m5 m6 in 0.2 seconds | wait ?
    
```

Pos0

```

+ATTD2+
for i = 1 to 8
  set position(s) item i of list -90 -40 80 0 -90 70 of motor(s)
  item i of list m1 m2 m3 m4 m5 m6 in 0.2 seconds | wait ?
    
```

Pos0

```

+Transmit+
script variables Sentence word letters
set Sentence to Message
for each item of split Sentence by
  set word to item
  set letters to word -> list word
  TryToSend letters
EndOfMessage
    
```

Chaque mot

```

+Message+
ask What is your message? and wait
report answer
    
```



```

+TryToSend + Word : +
script variables DoOne Ans DoTwo
set DoOne to false
repeat until DoOne
  SendWord Word
  set DoTwo to false
  repeat until DoTwo
    set Ans to Wita
    if Ans = C
      set DoOne to true
      set DoTwo to true
    else
      if Ans = IMI
        set DoTwo to true
      else
        think I'mWaitingforAnswerIMIorC for 2 secs
  
```

```

+SendWord + letters : +
for i = 1 to length of letters
  if i = 1
    Emit item i of letters
  else
    if item i of letters = item i - 1 of letters
      Emit DL
    else
      Emit item i of letters
Emit EOW

```

```

+Wita +
ask what is the answer? and wait
report answer

```

```

+Emit + char = x +
say char for 1 secs
if char = DL
  PosR 1
  // Lettre double
else
  if char = EOW
    PosR 1
    wait 1 secs
    // Fin de mot
  else
    PosR item index of A in caractere of codeD

```


Etape 7

Vous devez maintenant coder le bloc PosR qui reçoit en paramètre le numéro de la position à prendre et qui appellera les blocs Position créés à l'étape 2.

Exemple de travaux réalisables

```

+ PosR + PosNumber # = 7 +
if PosNumber = 0
  PositionD0
else
  if PosNumber = 1
    PositionD1
  else
    if PosNumber = 2
      PositionD2
    else
      if PosNumber = 3
        PositionD3
      else
        if PosNumber = 4
          PositionD4
        else
          if PosNumber = 5
            PositionD5
          else
            if PosNumber = 6
              PositionD6
            else
              Repos
    
```

Etape 8

On ajoute maintenant les blocs qui correspondent à la situation de récepteur.

```

+Receiver +
WaitForAtt
WaitForAR
think End.. for 5 secs
    
```

```

+WaitForAtt +
script variables AttVal ▶
set AttVal to false
repeat until AttVal
ask ATTvalidateY/N and wait
if answer = Y
set AttVal to true
say AnswerK for 2 secs
PosR 2 ▶ Emission K
    
```

```

+WaitForAR +
script variables Att_AR ▶
set Att_AR to false
repeat until Att_AR
say I'm in reception for 2 secs
TryToReceive
ask ReceiveAR=Y/N and wait
if answer = Y
set Att_AR to true
    
```

```

+TryToReceive +
script variables Att_C
set Att_C to false
repeat until Att_C
  ReceiveWord
  ask Wat is your answer IMI/C? and wait
  if answer = C
    set Att_C to true
    PosR 1 Emet C
  else
    if answer = IMI
      say Ask to repeat for 2 secs
      PosR 5 Emet I
      PosR 2 Emet M
      PosR 5 Emet I
    else
      think Hmm...
+ReceiveWord +
script variables Att_EOW
set Att_EOW to false
repeat until Att_EOW
  say I'm in reception for 2 secs
  ask Is it EoW Y/N and wait
  if answer = Y
    set Att_EOW to true
    
```

Etape 9

Comment gérer deux robots ?

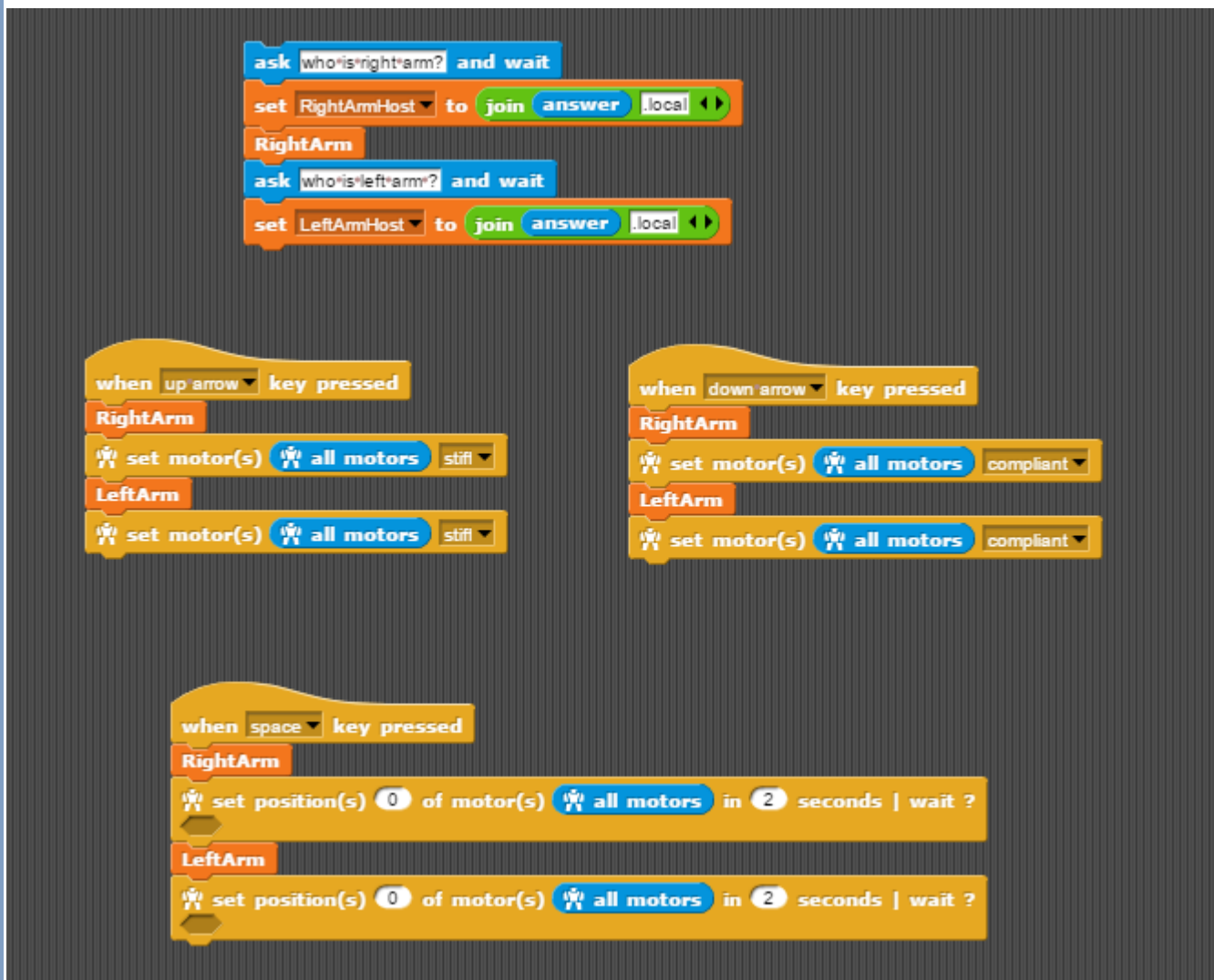
On crée 2 variables



Puis deux blocs



On peut maintenant tester le programme suivant :



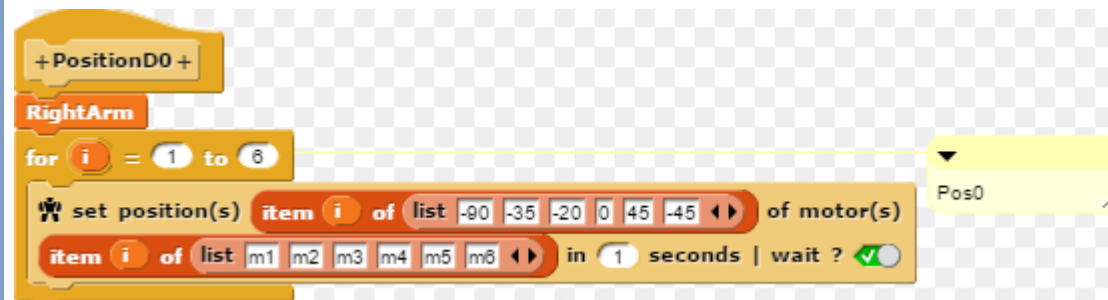
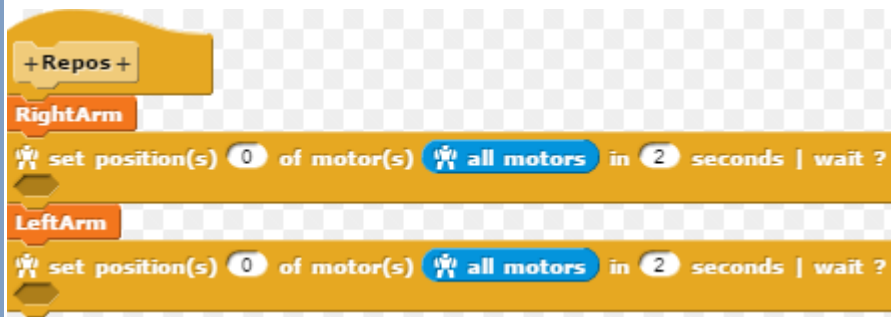
Etape 10

Corriger tous les blocs créés précédemment :



Pour qu'ils pilotent le bon robot.

Exemple de travaux réalisables



Et les blocs





Etape 11

Modifier l'initialisation



Réalisation finale

Tous les travaux précédents sont regroupés dans un fichier « Ergo_JR_Marin_1.32.xml »

Une approche de programmation en parallèle est proposée en évolution dans le fichier « Ergo_JR_Marin_2.1.xml »

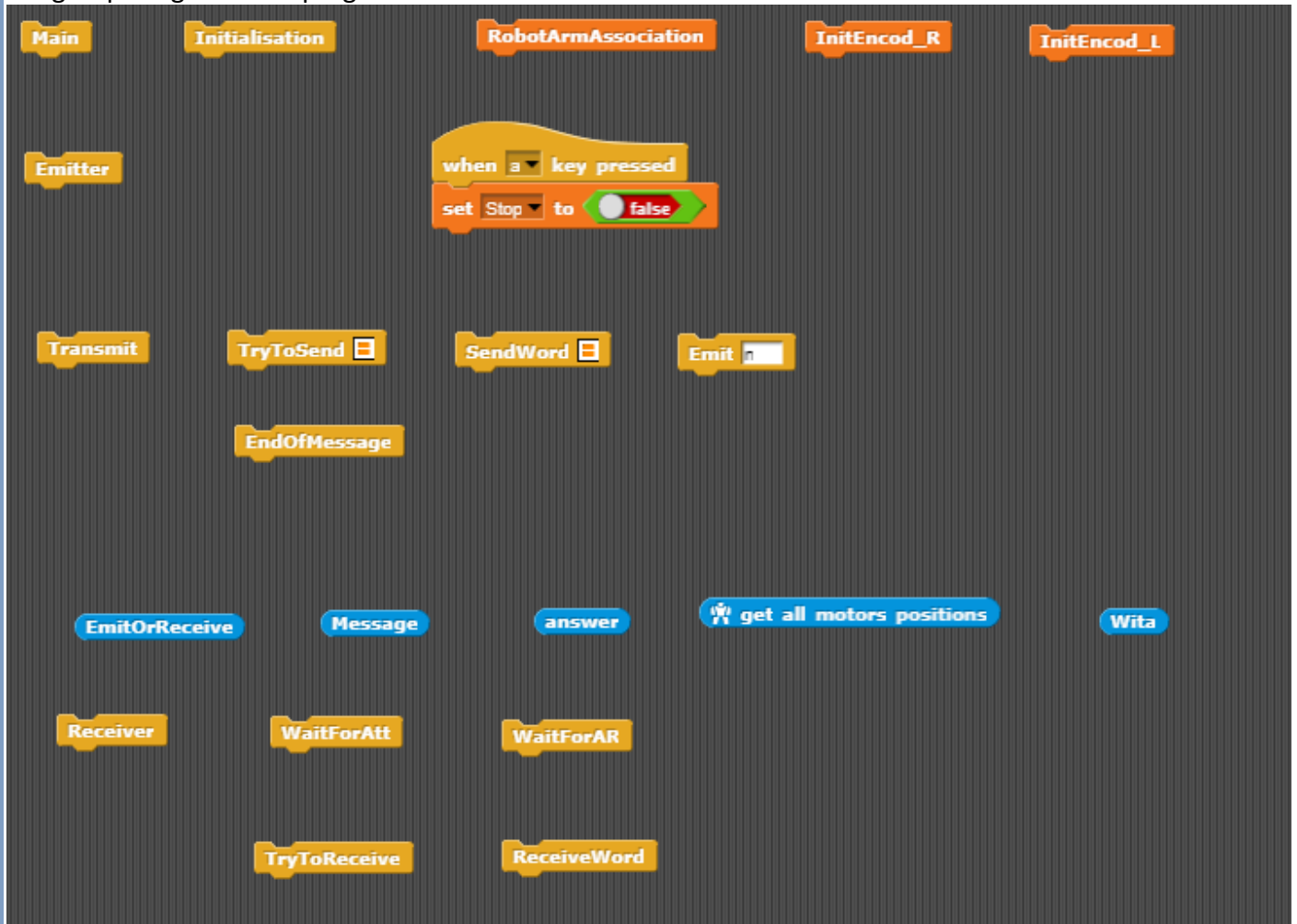
Comment faire bouger deux robots en même temps ?

Créer 3 sprites : 1 par robot plus un sprite qui contiendra le programme principal.



Le sprite main

Il regroupe la gestion du programme



Le sprite galileo

The code for 'Le sprite galileo' consists of the following blocks:

- test connection** (blue block)
- set host to RightArmHost** (orange block)
- when up arrow key pressed** (yellow block)
 - set motor(s) all motors stiff** (yellow block)
- when down arrow key pressed** (yellow block)
 - set motor(s) all motors compliant** (yellow block)
- when space key pressed** (yellow block)
 - set position(s) 0 of motor(s) all motors in 2 seconds | wait ?** (yellow block)
- when I receive Position** (yellow block)
 - Pos_R(n)** (yellow block)
- when I receive rest** (yellow block)
 - Rest_R** (yellow block)
- when I receive Att** (yellow block)
 - Att_R** (yellow block)

Le sprite Eremurus

The code for 'Le sprite Eremurus' consists of the following blocks:

- set host to LeftArmHost** (orange block)
- test connection** (blue block)
- when space key pressed** (yellow block)
 - set position(s) 0 of motor(s) all motors in 2 seconds | wait ?** (yellow block)
- when up arrow key pressed** (yellow block)
 - set motor(s) all motors stiff** (yellow block)
- when down arrow key pressed** (yellow block)
 - set motor(s) all motors compliant** (yellow block)
- when I receive rest** (yellow block)
 - Rest_L** (yellow block)
- when I receive Att** (yellow block)
 - Att_L** (yellow block)
- when I receive Position** (yellow block)
 - POs_L(n)** (yellow block)