

ProTIS 2021

*Robot Piloté à distance*

TAKTAK *Rami*  
TOUJANI *Adem*  
KHITOUS *Hamza*  
AZAIZ *Malek*

## Sommaire

I.	Introduction :	3
II.	Mise en œuvre :	4
	1. Répartition des tâches	4
	2. Pilotage des moteurs	4
	3. Transmission	6
	4. Asservissement des moteurs	8
	5. Interfaçage	9
III.	Résultats et interprétations :	10
IV.	Retour d'expérience :	10
V.	Annexes :	11
	Initialisation :	11
	Pilotage des moteurs :	11
	Asservissement des moteurs :	12
	Transmission sans fils :	12

# I. Introduction :

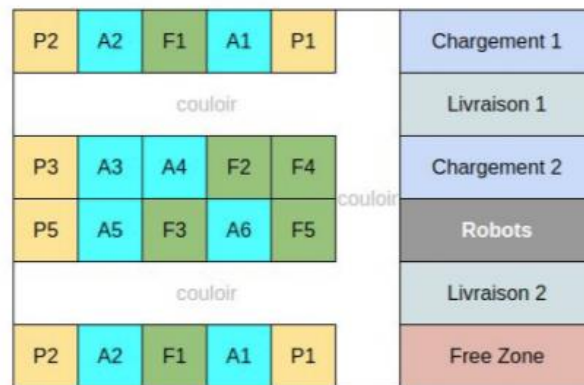
## Problématique et description du projet :

*Comment transmettre un ordre à un robot pour qu'il se déplace entre différentes régions d'une usine afin de transporter des pièces ?*

Cette problématique se sépare lors en plusieurs parties :

- Interface Homme/machine : créer une interface afin d'envoyer les différentes commandes de déplacement
- Transmission : trouver une méthode de transfert de données d'utilisateur de l'ordinateur au moteur et avoir un retour (émission / réception)
- Pilotage des moteurs : utiliser les cartes NUCLEO afin de déplacer moteur en translation et rotation (pour tourner)

Le projet consiste à piloter le moteur à distance à l'aide d'un émetteur radiofréquence (sans fil). Les deux moteurs de chaque roue du robot posséderont un récepteur pour traduire l'information reçue de la part de l'ordinateur telles que le robot avance recule tourne à gauche ou bien à droite en utilisant le protocole de communication RS232.



Exemple de plan d'une usine de production SOLEC

Figure 0 : schémas simplifiant les zones de déplacement de moteur

Le schémas suivant (figure 1) représente le fonctionnement prévu du robot l'utilisateur renvoie des données bien précises pour commander le moteur. Une transmission sans fil qui assure la réception de ces données par le moteur et déplacer le moteur alimenté par une pile de 9V la boucle d'asservissement assure le bon fonctionnement et déplacement du moteur et une rétroaction qui repère la position du moteur est prévu afin d'assurer l'achèvement de la commande.

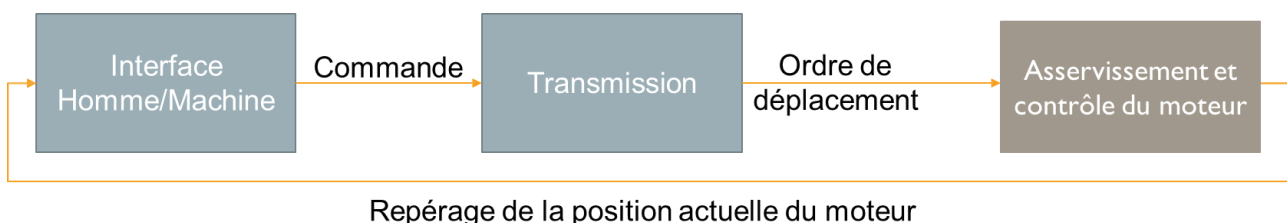


Figure 1 : schémas de fonctionnement de moteur

## II. Mise en œuvre :

### 1. Répartition des tâches

Equipe	Code	Hardware	Gestion d'équipe	Communication
Rami	x	x		x
Malek	x		x	
Hamza	x	x		
Adem		x		x

Tableau 1: Répartition des tâches

### 2. Pilotage des moteurs

Afin de piloter notre robot, nous avons utilisé deux moteurs CC (courant continu), le moteur se compose de : un encodeur, un corps (moteur + actionneur) et 6 pins. Le moteur qu'on a utilisé a un facteur de réduction qui vaut 19. Ce facteur de réduction est un coefficient qui se trouve entre le moteur et la partie qui va mettre en marche la roue, c'est le rapport entre le nombre de tour du moteur et le nombre de tour de l'arbre.

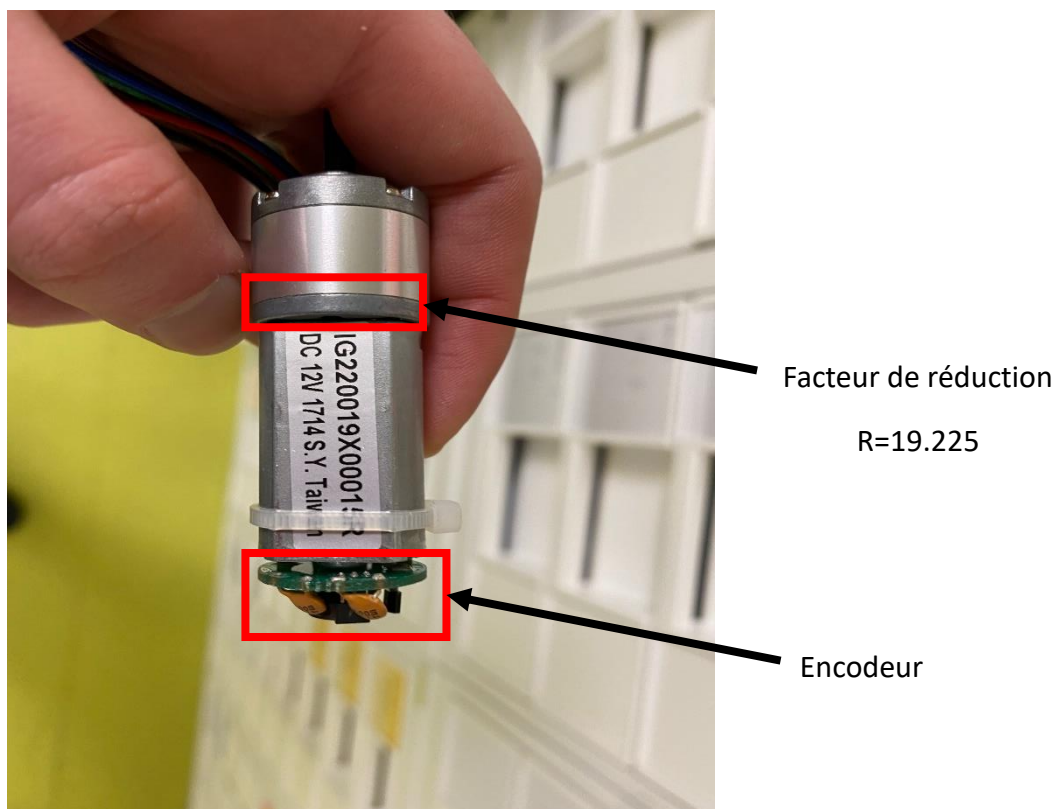


Figure 2 : Moteur utilisé pour piloter le véhicule

- **Fonctionnement des moteurs :**

On utilise le principe de la modulation de largeur d'impulsions c'est-à-dire le mouvement mécanique est directement associé à la valeur moyenne du signal envoyé et par la suite au rapport cyclique ce qui est la variable à changer afin de changer la vitesse du moteur.

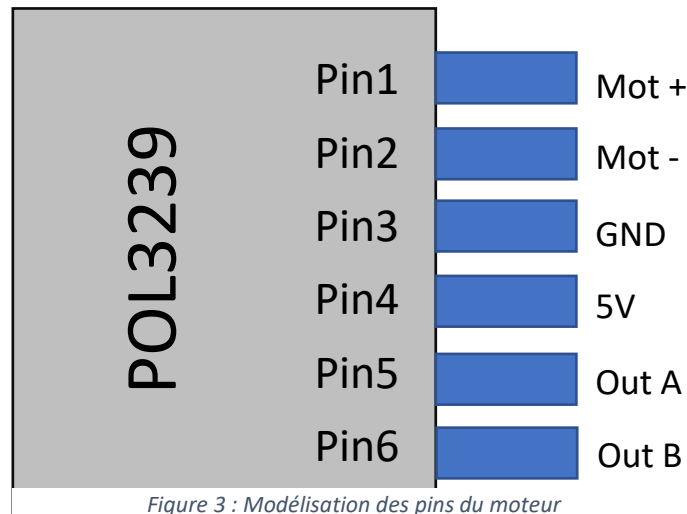


Figure 3 : Modélisation des pins du moteur

Le module POL3239 nous a servi comme pont entre la carte nucléo et le moteur CC.

Pour pouvoir piloter un moteur à courant continu dans les deux directions, il faut inverser le sens du courant (et la tension aux bornes du moteur) entre *Mot+* et *Mot-*.

Le rapport cyclique (ou RC) décrit la durée pendant laquelle le signal est à l'état haut (actif) en pourcentage de la durée d'un cycle complet. La fréquence détermine la vitesse à laquelle la MLI effectue un cycle (par exemple, 1000 Hz serait 1000 cycles par seconde) et par conséquent à quelle vitesse il passe de l'état haut à l'état bas et vice versa.

Le pin 4 (5V) sert à alimenter l'électronique du moteur (composants électroniques) mais pas le moteur lui-même. Pour cela, nous avons utilisé des batteries.

Dès la première séance nous avons réussi à faire marcher les deux moteurs simultanément dans le même avec la même consigne d'entrée. Nous avons remarqué que le robot n'allait pas en une ligne droite lorsqu'on l'a mis au sol.

A partir de la deuxième séance, nous avons pu changer le châssis du véhicule afin de bien répartir le poids des composants équitablement sur la surface du robot. Cette étape est cruciale afin de faciliter la mise en œuvre de l'asservissement.

- **Fonctionnement des encodeurs :**

Nous utilisons un encodeur magnétique qui possède 12 ticks/tour c'est-à-dire à chaque tour l'encodeur va renvoyer 12 périodes d'un signal rectangulaire.

Un encodeur est un dispositif électromécanique qui génère un signal électrique en fonction de la position ou du déplacement de l'élément mesuré. Ce dispositif va servir principalement dans partie asservissement (mesure de l'erreur).

L'encodeur est en quadrature de phase (décalage de 90° entre les deux signaux). L'intérêt d'une telle forme de signal est de pouvoir connaître le sens de rotation du moteur. En effet, en regardant quel signal est en avance par rapport à l'autre on peut en déduire si le moteur avance ou recule. Sur l'image ci-dessus, c'est la courbe jaune (Out A) qui est en avance sur la courbe bleue (Out B), le moteur fonctionne en marche avant.

Sur la figure à droite, on visualise deux signaux en quadrature de phase. L'intérêt d'une telle forme de signal est de pouvoir connaître le sens de rotation du moteur. En effet, en regardant quel signal est en avance par rapport à l'autre on peut en déduire si le moteur avance ou recule.

Ici c'est la courbe jaune qui est en avance de phase par rapport à la courbe en bleu, le moteur fonctionne en marche avant.

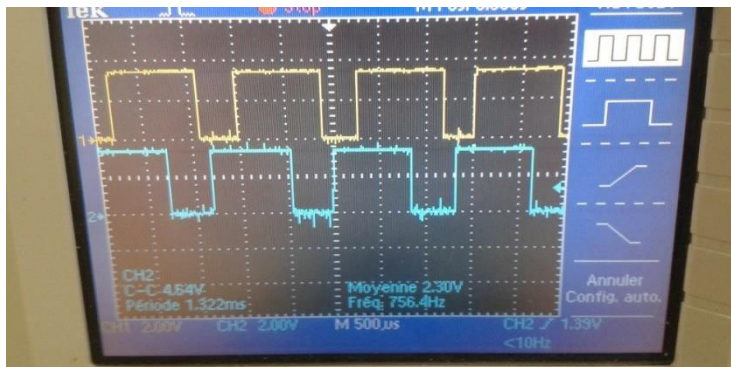


Figure 4 : Signaux des capteurs de l'encodeur

*courbe jaune= voix A et la courbe bleue = voix B*

### 3. Transmission

Dans cette partie, on s'intéresse à la manière de transmettre les ordres de commande du robot communiqués via l'interface Matlab sous la forme des données de la trajectoire (distance, vitesse) par l'intermédiaire d'un module RF (KAPPA-M868).

- **Schéma récapitulatif du circuit de Transmission :**

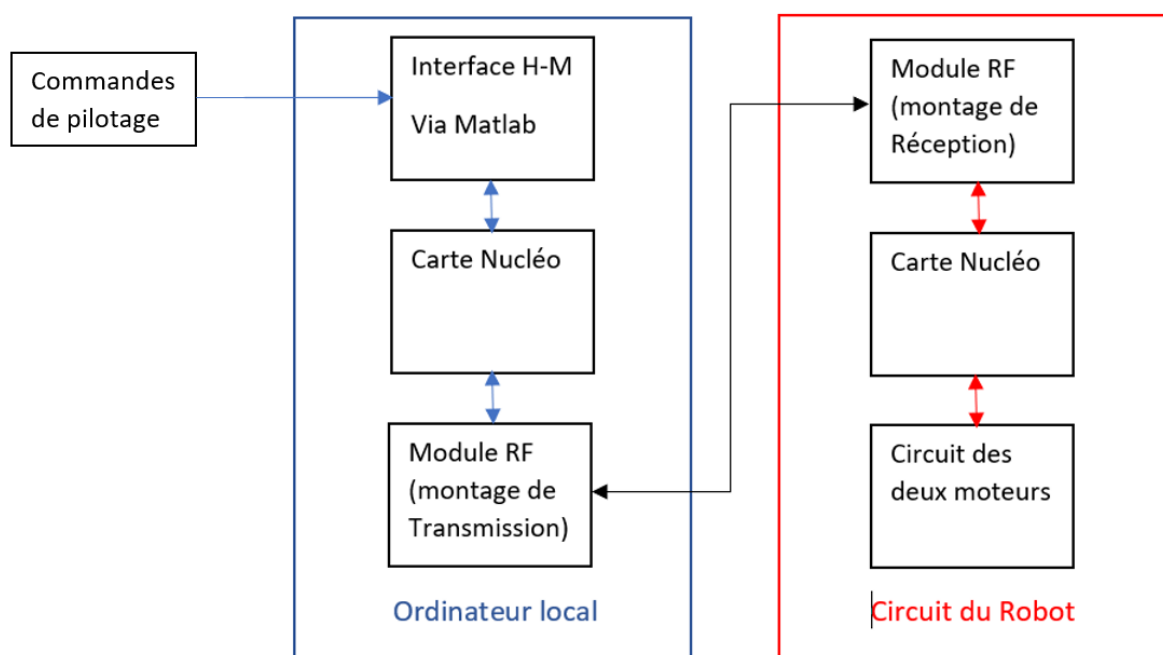


Figure 5 : Le circuit de transmission

On donne ici le détail du circuit de liaison entre la carte Nucléo et le module RF valide pour les deux configurations de la transmission ainsi que la réception :

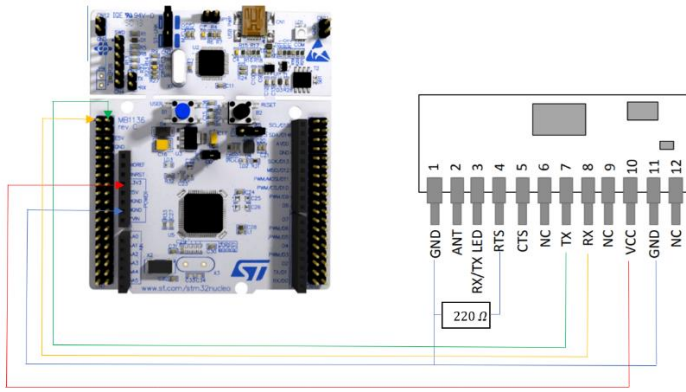


Figure 1: Câblage pour un circuit de Transmission/Réception

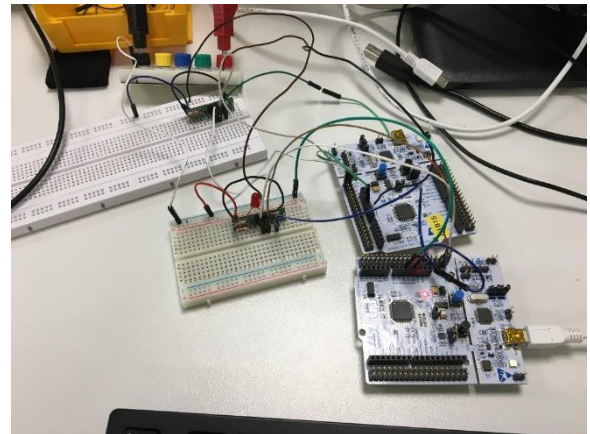


Figure 7: Le circuit réel de la transmission

On teste notre circuit en envoyant une simple lettre 'A' (sans passer par l'interface Matlab) via un petit code de transmission sous MBED puis on la reçoit de la même façon via un petit code de réception sous MBED : ces deux codes sont à la base de toute communication possible développée ultérieurement entre la carte Nucléo et le module RF (cf. fig8).

Ensuite cette lettre est traduite en une commande via un code de pilotage des deux moteurs sous MBED qui permet de faire avancer le robot suivant une ligne droite (les deux roues tournent à la même vitesse).

```

2 #include "mbed.h"
3 #include <string>
4
5 Serial carte2(PC_10, PC_11);
6 Serial pc(USBTX, USBRX);
7 int j;
8
9 char commande1[]="A\n"; // A désigne une commande pour avancer vers l'Avant
10
11 int main() {
12     carte2.baud(19200);
13     while(1) {
14         for(j=0;j<sizeof(commande1);j++){
15             carte2.putc(commande1[j]);
16             pc.printf(commande1);
17             wait(0.1);
18         } }
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999

```

Figure 8: La transmission d'une commande pour avancer vers l'Avant

#### 4. Asservissement des moteurs

**Objectif :** Dans cette partie on se propose d'asservir les deux moteurs **en vitesse** pour que le robot marche en ligne droite.

**Cahier de charge :** une erreur relative de 2 %

On modélise l'asservissement du moteur par la boucle suivante :

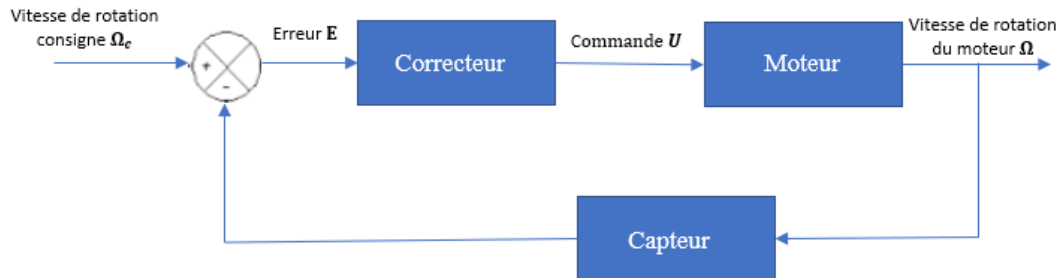


Figure 9: boucle d'asservissement du moteur

On choisit un correcteur PI (proportionnel intégral) qui annule l'erreur statique et rejette toute perturbation. La commande  $u$  à l'instant  $t$  est de la forme :

$$K_p e(t) + K_i \frac{1}{T_i} \int_0^t e(t) dt$$

L'action intégrale du correcteur est calculée en multipliant à chaque fois la somme de toutes les erreurs par  $\frac{K_p}{T_i}$ .

Mais pour calculer l'erreur, il faut comparer la valeur consigne à la vitesse réelle du moteur. Pour cela, on a créé la fonction **v\_ang** :

On détermine d'abord la résolution angulaire du moteur, c'est-à-dire la variation angulaire minimale entraînée par chaque incrémentation : A l'aide de la bibliothèque QEI et de la fonction **getPulses** on mesure **12 positions par tour complet**.

Ensuite, la vitesse angulaire est déterminée en multipliant cette résolution par la variation du nombre de *Pulses* pendant une courte période et en divisant par cette durée.

**NB :** Il faut tenir compte du facteur de réduction entraîné par l'encodeur attaché au moteur.

```
37 float resolution=2*3.1415/12 ;
38 QEI encoder0(D10, D11, NC, 48, QEI::X4_ENCODING);
39
40 float v_ang1(){ // fonction qui calcule la vitesse angulaire du premier moteur
41     int p0, p1 ;
42     float v ;
43     p0=encoder0.getPulses(); // position de l'incrément à t
44     wait_ms(10) ;
45     p1 = encoder0.getPulses(); // position de l'incrément à t+dt avec dt = 10 ms
46     v=(p1-p0)*resolution/(0.01*19.225) ; // calcul de la vitesse angulaire
47     return(v);
48 }
49
```

Figure 10: fonction qui calcule la vitesse angulaire avec D10 et D11 connectés directement aux voix A et B du moteur.



On utilise tera-term afin d'afficher les valeurs les plus significatives : la vitesse du moteur, l'erreur, la consigne, le rapport cycle ....

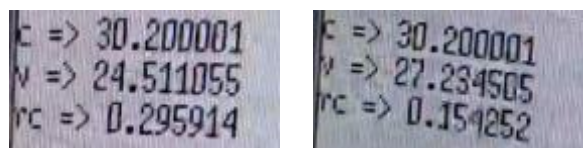


Figure 21 : Valeurs affichées par tera-term pour une vitesse angulaire consigne de 30 rad/s

L'asservissement est fonctionnel et l'erreur relative converge vers 0%.

Tera-term nous permet aussi de valider le bon fonctionnement de la fonction  $v_{ang}$ . Par exemple, lorsqu'on fait tourner le moteur à une vitesse constante, le freinage progressif de la roue à la main se manifeste par une diminution de la vitesse affichée par tera-term.

De plus, si on envoie la même commande au deuxième moteur, on constate que chaque moment du freinage du premier moteur est accompagné par une accélération du deuxième moteur : **l'asservissement est mis en place.**

## 5. Interfaçage

Pour que l'utilisateur puisse contrôler le robot à distance, on a essayé de créer une interface graphique sous *AppDesigner* : application de design d'interface sous.

Avec cette interface l'utilisateur pourra commander le robot en envoyant la valeur de la vitesse de contrôle et la distance à parcourir pour les déplacements linéaires : on a deux boutons pour orienter le robot dans la bonne direction et tourner comme ça il y a plus d'interaction entre l'utilisateur et le robot en cas d'obstacle ou un long trajet il suffit juste de connaître les distances des couloirs et la largeur des rangées.

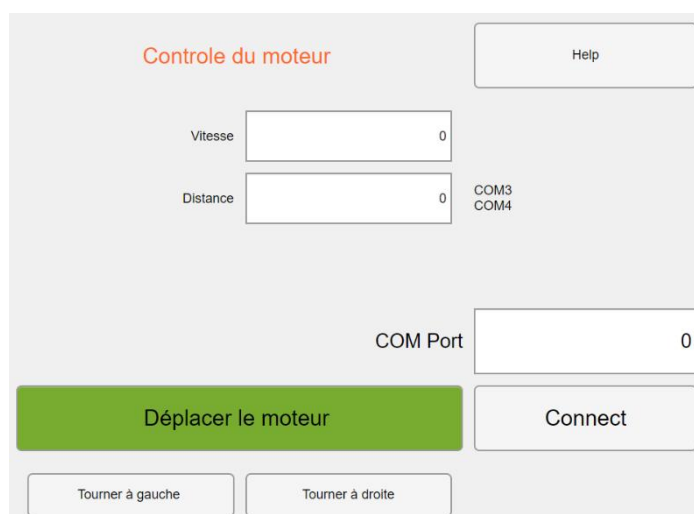


Figure 12 : interface homme machine

La valeur numérique de la vitesse et la vitesse sont ensuite envoyées en format chaîne de caractère à la carte NUCLEO afin d'envoyer les informations nécessaires de déplacement au robot.

Faute de temps, on a trouvé le bon code qui nous permet d'envoyer les informations enregistrées par Matlab qui doivent être transférées à mbed pour être envoyées comme données d'entrée au moteur pour générer son déplacement.

### III. Résultats et interprétations :

À la fin des 6 séances, notre équipe a réussi à concevoir un robot **autonome** qui avance à la suite d'une commande envoyée sans fil. Nous avons pu faire **tourner et asservir les deux roues simultanément**. La partie interface est sur laquelle, on n'a pas eu beaucoup de temps afin de la travailler.

#### Pistes d'améliorations :

- Concevoir un châssis mieux adapté à notre application.
- Appliquer un asservissement en position sur le moteur.
- Ajouter des indications lumineuses (par exemple pour la marche arrière).
- Réussir à faire une étude énergétique sur la consommation des moteurs (comme les autres groupes on fait).
- Ajouter des détecteurs afin d'éviter d'obstacles.
- Améliorer l'algorithme afin d'optimiser le traitement des données lors de l'asservissement.
- Réussir à afficher la position et la vitesse en direct sur l'interface homme-machine.

### IV. Retour d'expérience :

Nous avons trouvé cette expérience très riche en matière de nouvelles compétences, techniques de travail et outils du domaine de l'électronique et de la programmation.

Au début, nous avons eu quelques problèmes d'organisation et de gestion du temps ce qui a impacté la suite du projet puisqu'il y avait une cumulation de tâches qui n'étaient pas encore faites.

Grâce à ce projet, nous avons pu interagir avec les autres étudiants et les encadrants ce qui n'était pas évident puisque nous n'avons pas eu la chance d'avoir autant de cours en présentiel.

Ces échanges nous ont permis de mieux développer notre projet à travers les problèmes rencontrés et les solutions envisagées par les autres groupes, ce fut une expérience enrichissante sur le plan humain mais aussi sur le plan technique et académique.

Travailler sur un tel projet nous a aussi permis de mettre en évidence nos compétences en électronique, en programmation et aussi en travail d'équipe.

Dans l'idéal, nous aurions dû améliorer les performances de notre système mais faute de temps nous étions dans l'incapacité de terminer le projet.

## V. Annexes :

### Initialisation :

```
1 /*
2  * piloter deux moteurs à courant continu à distance
3  * Auteurs : TAKTAK Rami, AZAIZ Malek, KHITOUS Hamza & TOUJANI Adem
4  */
5
6 #include "mbed.h"
7 #include <stdint.h>
8
9 //moteur à gauche = ROUE 1
10 PwmOut moteur1(D8);           // controle de la roue 1
11 DigitalOut moteur1_direction(D6); // sens de rotation de la roue 1
12
13 //moteur à droite = ROUE 2
14 PwmOut moteur2(D9);           // controle de la roue 2
15 DigitalOut moteur2_direction(D3); // sens de rotation de la roue 2
16
17 Serial pc(USBTX, USBRX);       // affichage sur ordinateur
18 Serial cartel(PC_10, PC_11);   // affichage sur ordinateur
19
```

### Pilotage des moteurs :

```
55 void marche_arriere(float rc){
56     moteur1_direction = 0 ;
57     moteur2_direction = 1 ;
58
59     moteur1.write(rc);
60     moteur2.write(rc);
61
62     wait(1) ;
63
64     moteur1.write(0);
65     moteur2.write(0);
66
67 }
68
69 void marche_avant(float rc){
70
71     moteur1_direction = 1 ;
72     moteur2_direction = 0 ;
73
74     moteur1.write(rc);
75     moteur2.write(rc);
76
77     wait(1) ;
78
79     moteur1.write(0);
80     moteur2.write(0);
81
82 }
83
84 void turn_R(float rc) //( tourner de 90° à droite pour une consigne de 30 rad/s)
85 {
86     moteur1_direction = 1;
87     moteur1.write(rc);
88     moteur2.write(0);
89
90     moteur2_direction = 1;
91
92     wait(0.14); // disance de l'arc = pi/2 * R(=25cm , rayon de la roue)
93     moteur1.write(0);
94 }
95
96 void turn_L(float rc) //( tourner de 90° à gauche pour une consigne de 30 rad/s)
97 {
98     moteur1_direction = 1;
99     moteur2_direction = 0;
100
101     moteur1.write(0);
102     moteur2.write(rc);
103
104     wait(0.14); // disance de l'arc = pi/2 *R(=25cm , rayon de la roue)
105     moteur2.write(0);
106 }

```

## Asservissement des moteurs :

```
37 float resolution=2*3.1415/12 ;
38 QEI encoder1(D10, D11, NC, 48, QEI::X4_ENCODING);
39 QEI encoder2(D0 , D1 , NC, 48, QEI::X4_ENCODING);
40
41 float v_ang1(){ // fonction qui calcule la vitesse angulaire du premier moteur
42     int p0, p1 ;
43     float v ;
44     p0=encoder1.getPulses(); // position de l'incrément à t
45     wait_ms(10) ;
46     p1 = encoder1.getPulses(); // position de l'incrément à t+dt avec dt = 10 ms
47     v=(p1-p0)*resolution/(0.01*19.225) ; // calcul de la vitesse angulaire
48     return(v);
49 }
50
51 float v_ang2(){ // fonction qui calcule la vitesse angulaire du deuxieme moteur
52     int p0, p1 ;
53     float v ;
54     p0=encoder2.getPulses(); // position de l'incrément à t
55     wait_ms(10) ;
56     p1 = encoder2.getPulses(); // position de l'incrément à t+dt avec dt = 10 ms
57     v=(p1-p0)*resolution/(0.01*19.225) ; // calcul de la vitesse angulaire
58     return(v);
59 }
60
61
62
63 // La boucle d'asservissement en vitesse
64
65 void boucle_regul(void){
66     float v1;
67     float v2;
68     float erreur=0.0; // erreur proportionnelle du moteur 1 initiale
69     float Ierreur=0.0; // erreur intégrale du moteur 1 initiale ( erreur accumulée)
70     float erreur2=0.0; // erreur proportionnelle du moteur 1 initiale
71     float Ierreur2=0.0; // erreur intégrale du moteur 2 initiale ( erreur accumulée)
72
73     float Kp = 1.0/19.225;
74     float KiTe = 0.0001;
75
76     Ierreur = Ierreur + erreur; // calcul de l'erreur accumulée du moteur 1
77     Ierreur2= Ierreur2+ erreur2; // calcul de l'erreur accumulée du moteur 2
78
79     v1=v_ang1(); // vitesse angulaire réelle du moteur 1
80     v2=v_ang2(); // vitesse angulaire réelle du moteur 2
81
82     erreur = consigne - v1; // erreur du moteur 1
83     erreur2= consigne - v2; // erreur du moteur 2
84
85     moteur_1(Kp* erreur + KiTe *Ierreur ); // la nouvelle commande u(t) appliquée au moteur 1
86     moteur_2(Kp* erreur2 + KiTe *Ierreur2 ); // la nouvelle commande u(t) appliquée au moteur 2
87
88     //pc.printf("c1 => %lf \r\n",consigne); // affichage de la consigne
89     //pc.printf("v1 => %lf \r\n",v1); // affichage de vitesse angulaire actuelle du moteur 1
90     //pc.printf("rc1 => %lf \r\n",Kp * erreur + KiTe *Ierreur); // affichage de la commande appliquée au moteur 1
91 }
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
```

## Transmission sans fils :

```
106 // fonction qui traduit la lettre reçue en une commande à appliquer au robot
107 void marche( char direction, float rc ){
108     if (direction=='F'){ // F = forward
109         marche_avant(rc);
110
111     if (direction=='B'){ // B = backward
112         marche_avant(rc);
113
114     if (direction=='R'){ // R = right
115         turn_R(rc);
116
117     if (direction=='L'){ // L = left
118         turn_L(rc);
119     }
120 }
```

```
29
30 int main() {
31     double rc;
32     char data ;
33
34     rc =0.35;
35     cartel.baud(19200);
36
37     moteur1.period_ms(10);
38     moteur2.period_ms(10);
39
40
41     while(1) {
42         if (cartel.readable())
43             {
44                 data=cartel.getc(); // récupération de la lettre de commande
45                 marche(data, rc ) ; // commande associée à la lettre reçue
46                 pc.putc(data);      // affiche sur tera term la lettre reçue
47
48             }
49     }
50
51 }
52
53
```