Le Tychoscope du crocodile

Version : V1.0
Date : 22 Dec 2014
Auteur : croco31

Résumé :
Ce document décrit la construction d'un Tychoscope, petit robot mobile à déplacement aléatoire qui permet de tester l'effet psychokinétique de l'esprit sur la matière via le tirage au hasard du déplacement.

Avertissement :

Il n'y en a pas : à priori ce montage ne présente pas de risque car il est alimenté sur batterie NiMH, mais l'auteur décline toute responsabilité sur sa construction et son usage.

1. Introduction au Tychoscope

Si on recherche sur le Web le mot « tychoscope » , on tombe rapidement sur Wikipédia et le site du GERP, mais on reste sur sa faim car aucune info technique complète n'est disponible permettant sa réalisation pratique, l'appareil n'étant plus fabriqué aujourd'hui.
Je cite Wiki :
« Un **tychoscope** est un mobile autoporteur qui se déplace selon un mouvement brownien. Il fut notamment utilisé au cours des années 1970 par des parapsychologues dans des expériences visant à étudier l'effet de la psychokinèse, c'est-à-dire l'effet de la pensée sur le mouvement et la matière. »

Le tychoscope a été conçu dans les années 1975 par l'ingénieur Pierre Janin, qui collaborait avec Remy Chauvin.
En 1985 le docteur René Peoc'h a soutenu une thèse sur l'influence des poussins sur le tychoscope, en ayant imprégné les poussins sur ce mobile, de telle sorte qu'ils le prennent pour leur mère. Des sceptiques zététiciens ont critiqué cette expérience, arguant de mauvaise interprétation des statistiques du déplacement.

Ma première rencontre avec cet objet a été dans les années 1979, dans la défunte revue « Le Monde Inconnu » qui montrait JeanPierre.Girard en train de manipuler cet engin.
Ayant participé à un stage de JPG en 2013, et vu les effets que lui et d'autres avaient sur des barres d'alu, j'ai acquis un de ses livres (Manuel de parapsychologie appliquée JP.Girard page 313) où est justement cité le tychoscope.

C'est là que j'ai décidé d'en construire un avec les technologies actuelles pour voir ce qu'il en est.

2. Le principe de base du tychoscope

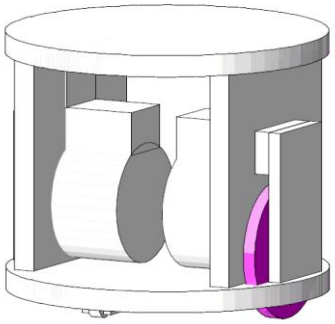
Le principe est de faire un tirage aléatoire sur l'angle (0 à 360°) et la distance (quelques centimètres) que doit parcourir l'objet. Ce tirage est issu habituellement du bruit électrique d'une tension issue d'une diode zener. Ce bruit est très faible (µV) et est la conséquence de l'agitation thermique des électrons dans le semi-conducteur composant cette diode.
On conçoit que si un effet de la pensée existe sur la matière, il doit être plus aisé d'agir sur ces électrons que sur une barre d'alu.

A partir de ces deux tirages, on active 2 moteurs qui provoquent la rotation et l'avance du robot sur une surface plane couverte d'un papier, sur lequel un crayon (feutre) enregistre la trajectoire qui sera une suite de segments visualisant le déplacement.

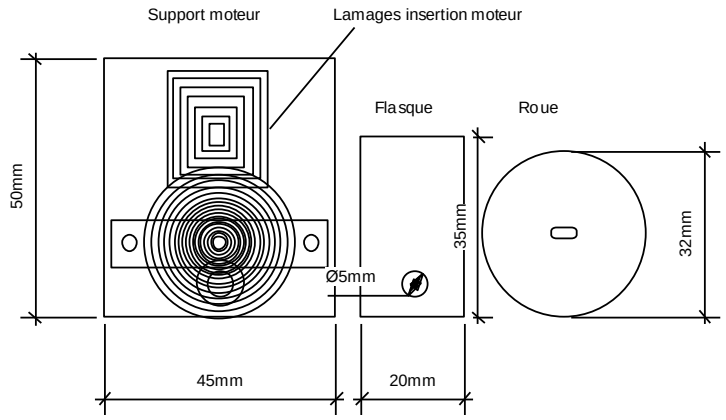
Quand les 2 moteurs tournent dans le même sens, le robot avance ou recule en ligne droite (même nombre de pas pour les deux moteurs). Quand les moteurs tournent du même nombre de pas en sens inverse, le robot effectue une rotation sur lui-même.

3. La mécanique du tychoscope

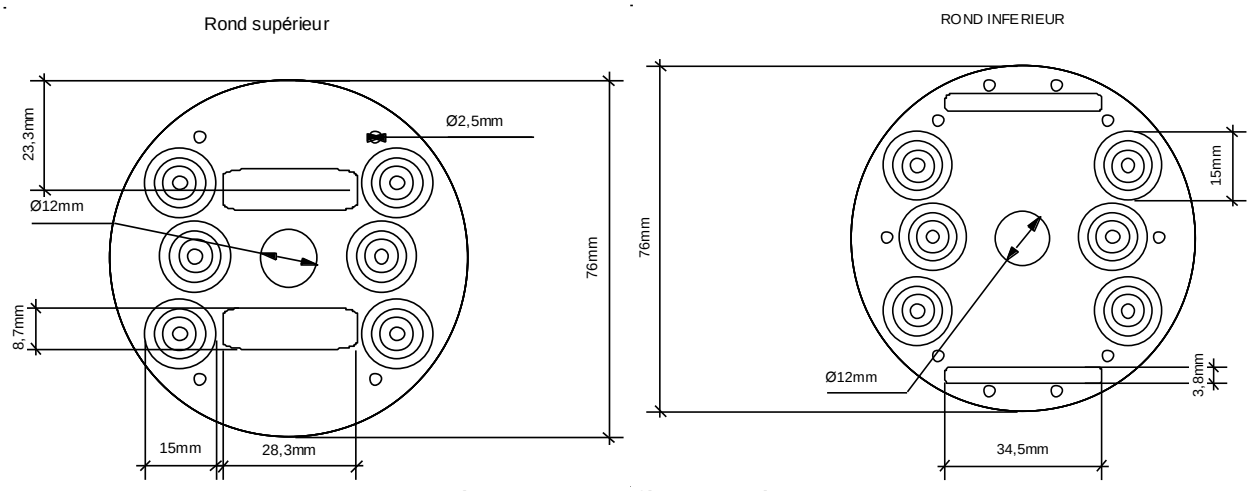
La structure porteuse est réalisée en CTP5mm, usiné à la CNC. Deux flasques latéraux portent chacun un moteur pas-à-pas miniature, dont l'axe s'emmanche sur une roue de D30mm, elle-même fraisée en CTP3mm (multiplis modélisme). Le corps du moteur est encastré dans la paroi latérale interne, afin d'avoir assez de place pour passer (juste) un tube de D12mm support du crayon.
Le rond inférieur est équipé de deux fentes qui laissent passer les roues diamétralement opposées. Ce rond supporte 5 accus AA NiMH montés verticalement dans des encoches rondes et fixés à la colle chaude. Les mêmes encoches sont présentes sur le rond supérieur.
Une flasque latérale est collée sur un intercalaire un peu supérieur à 3.5mm, et soutient l'axe de chaque moteur afin d'éviter un flambage de la roue, qui reste ainsi bien verticale.
Le rond supérieur est usiné pour laisser passer les fils des moteurs et support le circuit électronique (circulaire) de commande.
Le générateur de bruit aléatoire est dans un petit circuit électronique blindé monté verticalement entre 2 accus.
Deux plots silicone (butée de tiroir) sont montés dessous pour éviter le basculement lors des déplacements. Ceci remplace avantageusement une roulette folle difficile à réaliser à cette taille.
Un tube central D12mm en alu est inséré pour glisser un stylo-feutre sans accrocher les fils internes.
L'ensemble sera glissé dans un tube PVC D80mm de descente de gouttière (plus fin qu'un tube d'évacuation), qui sera fixé à la structure par deux vis à bois latérales.



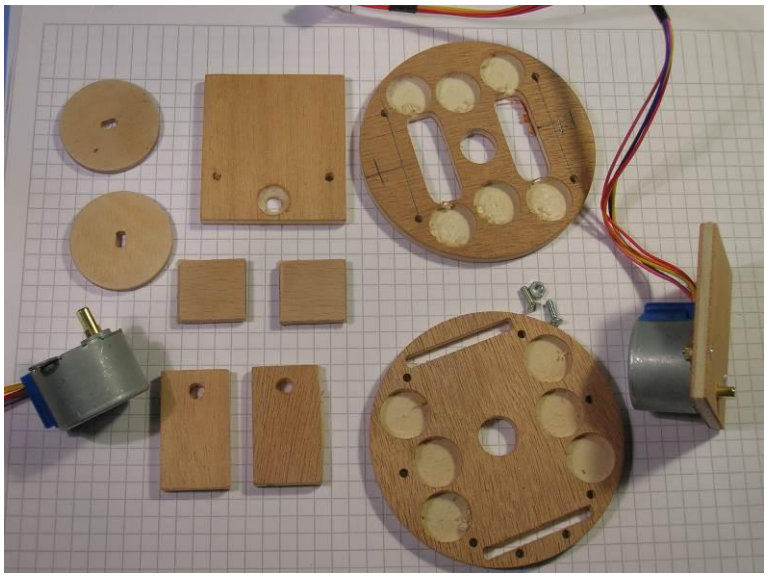
Structure mécanique tu tychoscope



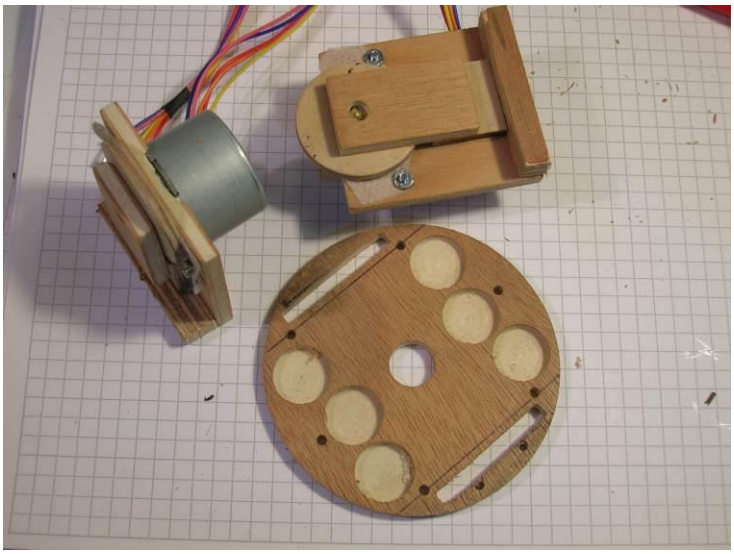
Découpes des flancs et roue



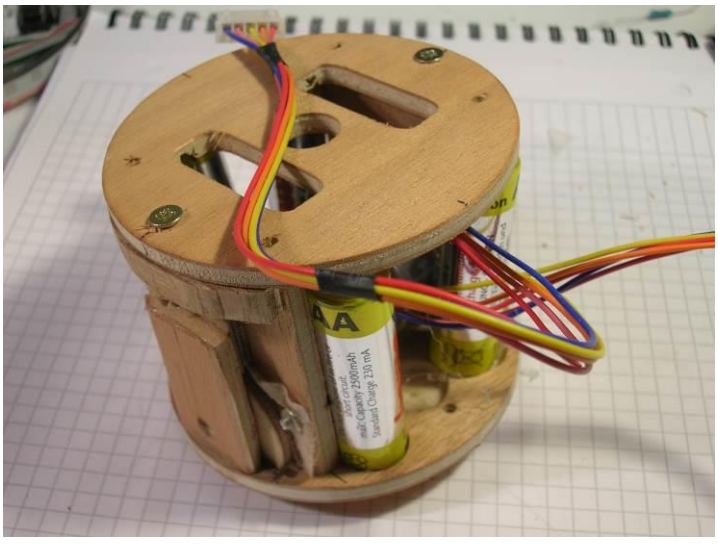
Découpe des ronds inférieur et supérieur



Les découpes et un moteur monté sur sa paroi latérale.



Détail montage des moteurs et de la roue avec flasque antiflambage



La structure montée avec les accus (pièce de montage arrondie par ponçage)

Les roues sont enduites de colle chaude pour améliorer leur adhérence sur un plateau type plancher.

4. L'électronique

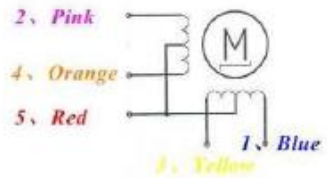
Le moteur (il y en a deux) utilisé est un moteur unipolaire 5 phases 28BYJ-48-5V que l'on trouve pour quelques euros sur ebay. Ce moteur pas à pas possède 64 pas, mais inclût un réducteur à engrenages 1/64, ce qui implique qu'un tour de l'axe demande 64x64 pas. Il a un couple très important, et consomme environ 0.2A à 100Hz.
En pratique on peut le commander jusqu'à 200Hz sans qu'il décroche et manque des pas. Il y a 5 fils à connecter, le rouge étant à relier au +5V (en fait la batterie NiMH monte à 7.2V quand elle est pleinement chargée, ce qui ne pose pas de problème à part un échauffement). Un simple circuit ULN2003 suffit à commuter les 4 phares dans le bon ordre (X Y X/ Y/) pour un moteur unipolaire. Le circuit ULN2003 ne contient que 7 drivers, ce qui oblige à en utiliser deux, mais l'intérêt est qu'il contient aussi les diodes de protection de surtension des sorties dont le point commun est à relier au +5V aussi.

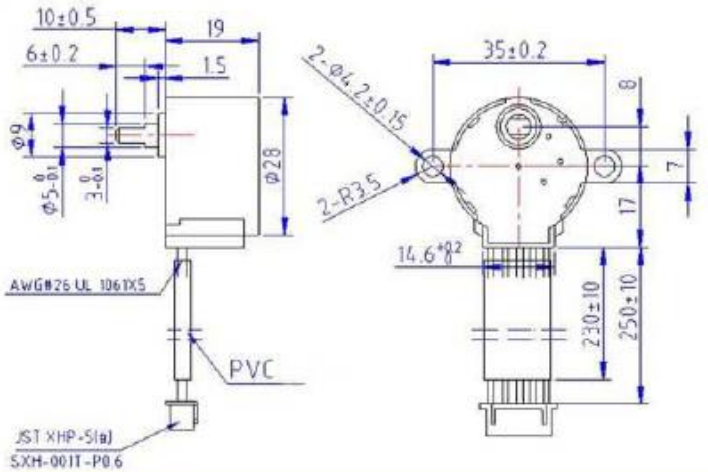
Kiatronics
electronic design and manufacture

28BYJ-48 – 5V Stepper Motor

The 28BYJ-48 is a small stepper motor suitable for a large range of applications.

Rated voltage :	5VDC
Number of Phase	4
Speed Variation Ratio	1/64
Stride Angle	5.625°/64
Frequency	100Hz
DC resistance	50Ω±7%(25℃)
Idle In-traction Frequency	> 600Hz
Idle Out-traction Frequency	> 1000Hz
In-traction Torque	>34.3mN.m(120Hz)
Self-positioning Torque	>34.3mN.m
Friction torque	600-1200 gf.cm
Pull in torque	300 gf.cm
Insulated resistance	>10MΩ(500V)
Insulated electricity power	600VAC/1mA/1s
Insulation grade	A
Rise in Temperature	<40K(120Hz)
Noise	<35dB(120Hz,No load,10cm)
Model	28BYJ-48 – 5V





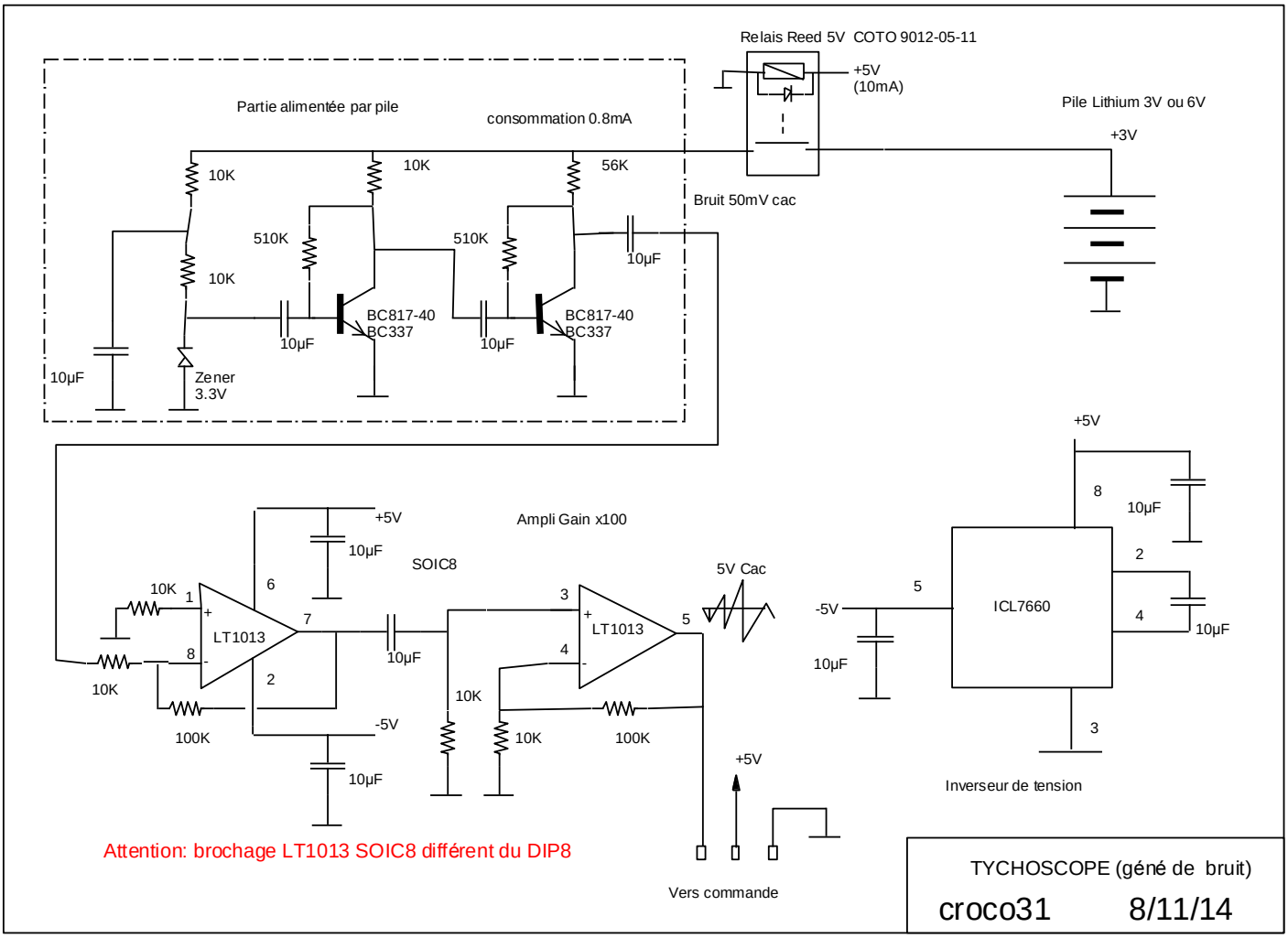
P.O. Box 8211 Cherrywood Tauranga New Zealand Phone: +64 7 578 7739 Fax: +64 7 578 7749 E-mail: enquiry@kiatronics.com
Website: www.kiatronics.com

5. Le schéma

Le schéma est scindé en deux parties :

- La partie générateur de bruit aléatoire, situé sur un circuit séparé et blindé et alimenté par une pile lithium interne pour éviter tout problème d'oscillation avec le circuit de commande et les perturbations des moteurs.
- La partie commande des moteurs basée essentiellement sur un microcontrôleur PIC16F1708 qui fait quasiment tout, associé à deux drivers ULN2003.

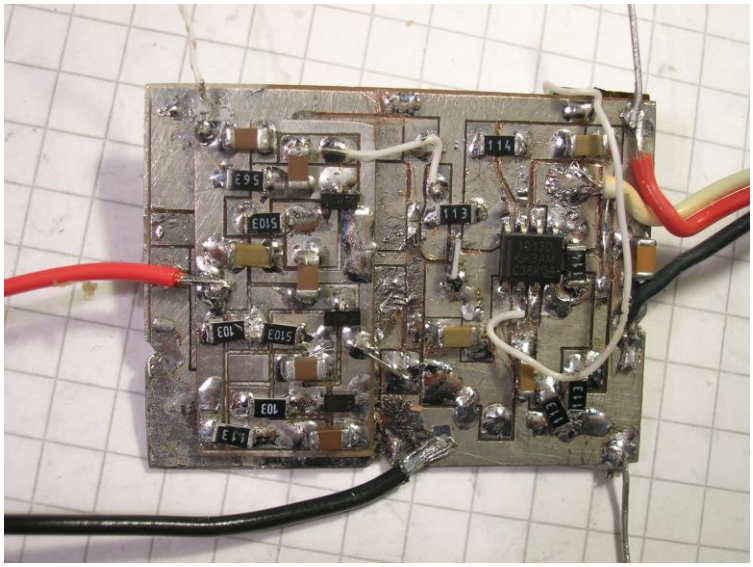
Le générateur de bruit



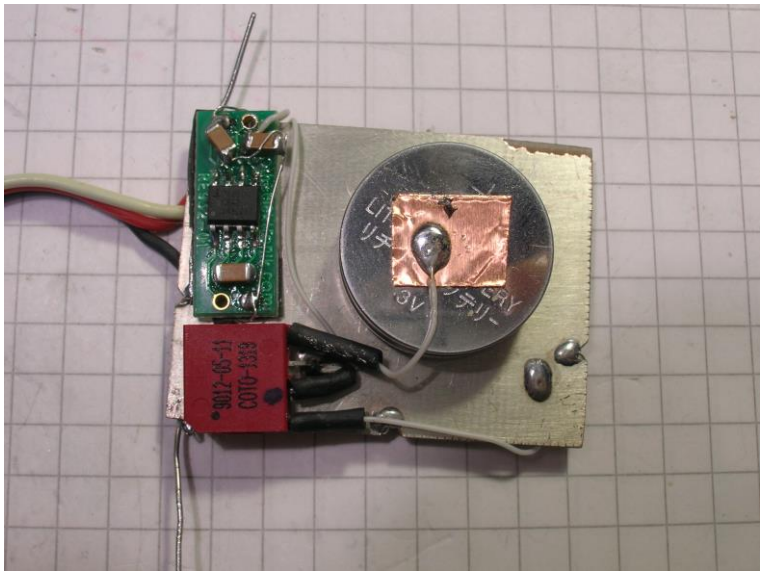
Ce générateur utilise le bruit thermique d'une diode zener et des transistors d'amplification associés. En fait la zener de 3.3V est très en-dessous de sa tension (pile de 3V), avec un faible courant, ce qui permet de consommer très peu sur la pile (0.8mA environ pour tout l'étage). On pourrait mettre deux piles lithium en série pour avoir 6V sans problème.
Cette pile est mise en service par le relais Reed (faible courant de bobine 10mA) dès que le 5V venant de la carte commande est présent, ce qui évite un interrupteur double et des fils de plus entre les cartes.
Ayant eu des problèmes d'oscillation en alimentant directement la zener avec le 5V du circuit commande, je me suis orienté sur la solution pile qui elle est parfaitement stable en bruit.
Une autre solution utilisée sur le tychoscope original de 1979, est un circuit élévateur de tension à 10V à partir du 5V ce qui à priori devrait éviter la rétroaction et l'oscillation, mais je ne l'ai pas testé, ayant trouvé trop tard le schéma original de l'étage de bruit du tychoscope d'origine.
En sortie de l'étage à transistors on obtient environ 50mV crête à crête autour de 1V avec un bruit à peu près blanc (énergie constante sur la bande) sur la bande 0-100KHz.
Toutes les caps sont en CMS et 10µF pour normaliser, car je les avais en stock et cela évite les erreurs de soudure.

L'étage suivant basé sur l'AOP double LT1013 permet d'obtenir un gain de 100 pour amener le signal de bruit à environ 5V crête à crête autour de la masse. Le fait d'utiliser les 2 étages de gain x10 permet d'avoir une bande passante de 100KHz (le produit gain x bande du LT1013 est de l'ordre de 1MHz), ce qui suffit pour notre affaire.
L'AOP est alimenté en double tensions positive et négative +5V et -5V, pour pourvoir référencer à la masse directe et éviter des propagations de signaux parasites en mode commun venant des alimentations (réjection de 100dB de l'AOP).

Le circuit à pompe de charge ICL7660 permet d'obtenir le -5V à partir du 5V entrant.



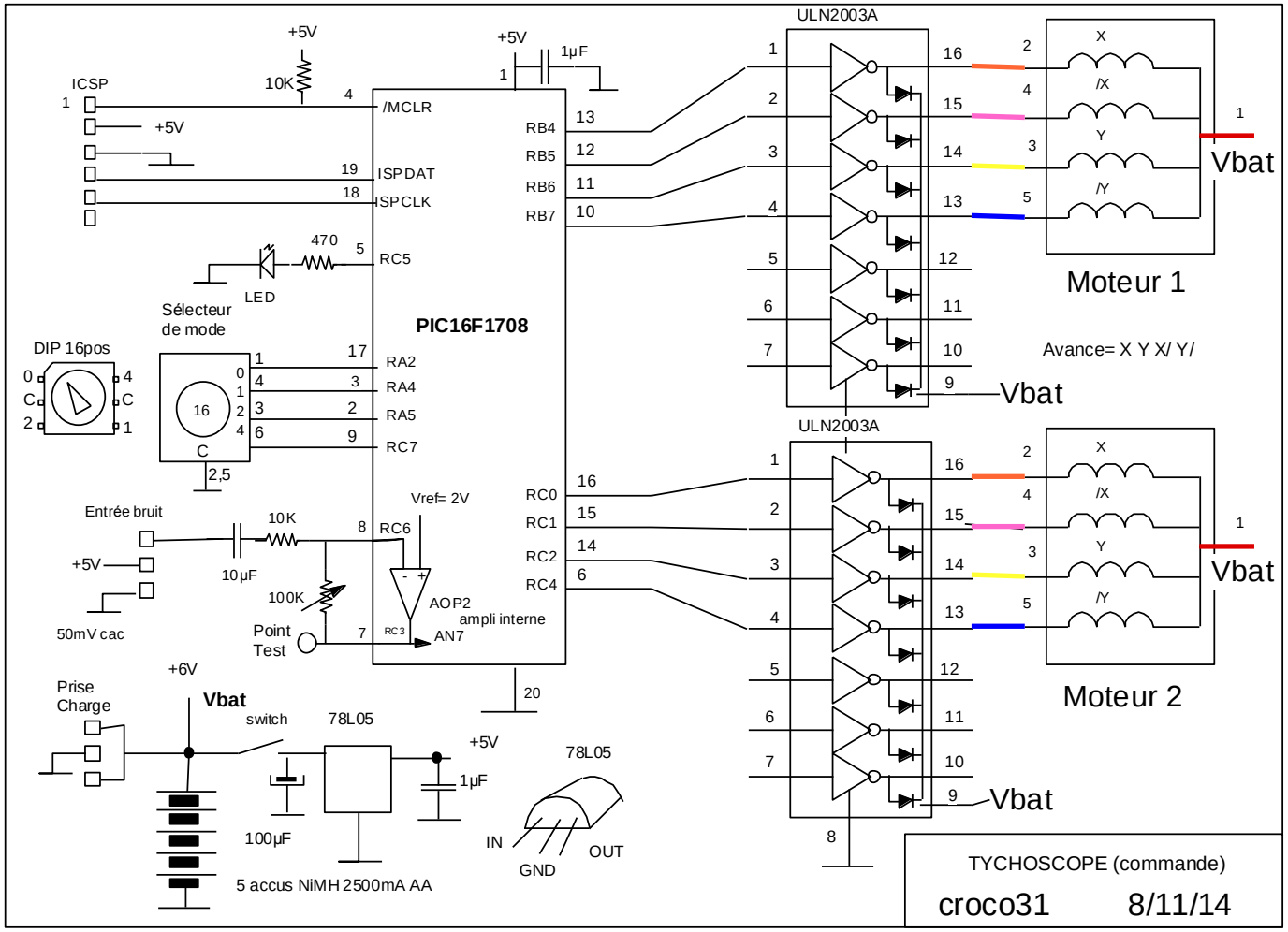
La platine CMS coté face : pas mal de modifs car elle a servi de testeur.



La platine CMS coté pile montrant l'inverseur de tension la pile et le relais reed minuscule rajoutés en fin de mise au point.

Le générateur de bruit terminé est blindé par du feuillard cuivre relié à la masse (cela ne blinde pas magnétiquement mais les moteurs ne gênent pas).

La partie commande



Le circuit commande utilise un PIC16F1708 20 broches qui intègre tout ce qu'il faut, y compris un AOP interne avec une tension Vref interne pour faire un étage d'amplification réglable centré sur 2V, ce qui permet de régler par le potard multi tours de 100K le niveau crête à crête du signal aléatoire et changer sa courbe gaussienne de répartition des valeurs acquises par l'ADC interne du PIC.

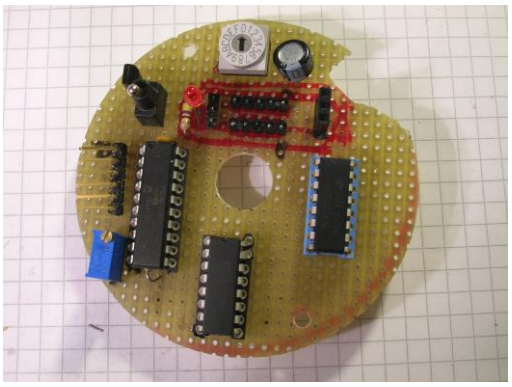
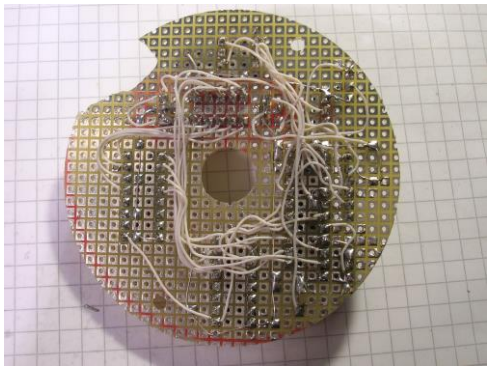
Le PIC associé à des drivers de moteurs ULN2003.

Un sélecteur hexa à 16 positions permet de choisir le mode de fonctionnement du tychoscope.

Une LED permet de signaler ce qu'il se passe et indiquer notamment que l'accu est déchargé.

Une prise ICS permet la programmation du PIC à partir d'un module PicKIT3, et, utilisant un module PicKIT2 en mode UARTTool (le PicKIT3 n'a plus cette fonction). En effet le PicKIT2 permet son usage en tant qu'interface série UART, le logiciel configurant les ports TXD et RxD de l'UART interne sur les broches 18 et 19 : c'est une fonction très intéressante de multiplexage des nouveaux PICs récents de ce type.

L'ensemble est alimenté par 5 éléments d'accus NiMH, associé à un régulateur de tension +5V pour l'électronique et alimentant directement les moteurs (un peu au-dessus des 5V du moteur quand pleinement chargés mais cela marche).

La platine à trous côté face et côté pile.

Une face avant usinée CNC dans une plaque d'alu 1mm est monté au-dessus de la platine et laisse l'accès :

- A la prise de charge d' accu
- A la LED
- Au sélecteur de mode 16 positions
- A l'interrupteur marche/arrêt

Cela donne le résultat final avant insertion dans le tube PVC de D80mm :

6. Les premiers essais

Cet objet est assez amusant quand il se déplace sur le parquet : on dirait une petite bête intelligente qui furète.
Il ne reste plus qu'à faire des essais pour voir si une concentration mentale peut influencer son déplacement.
Et là on sort de l'électronique pure….

7. Le logiciel

Le logiciel du PIC16F1708 est développé en C (compilateur MPLAB-XC8 dont une version Lite gratuite peut être téléchargée sur www.microchip.com)
Le PIC 16F1708 est configuré pour utiliser son oscillateur interne 16MHz, et utiliser toutes ses broches en broches IOs.

Il se programme avec un programmateur de PIC (Uniquement PicKIT3 si on utilise MPLAB-X) à partir du fichier .HEX qui contient aussi les bits de configuration du PIC.

En voici une copie (c'est du texte) :

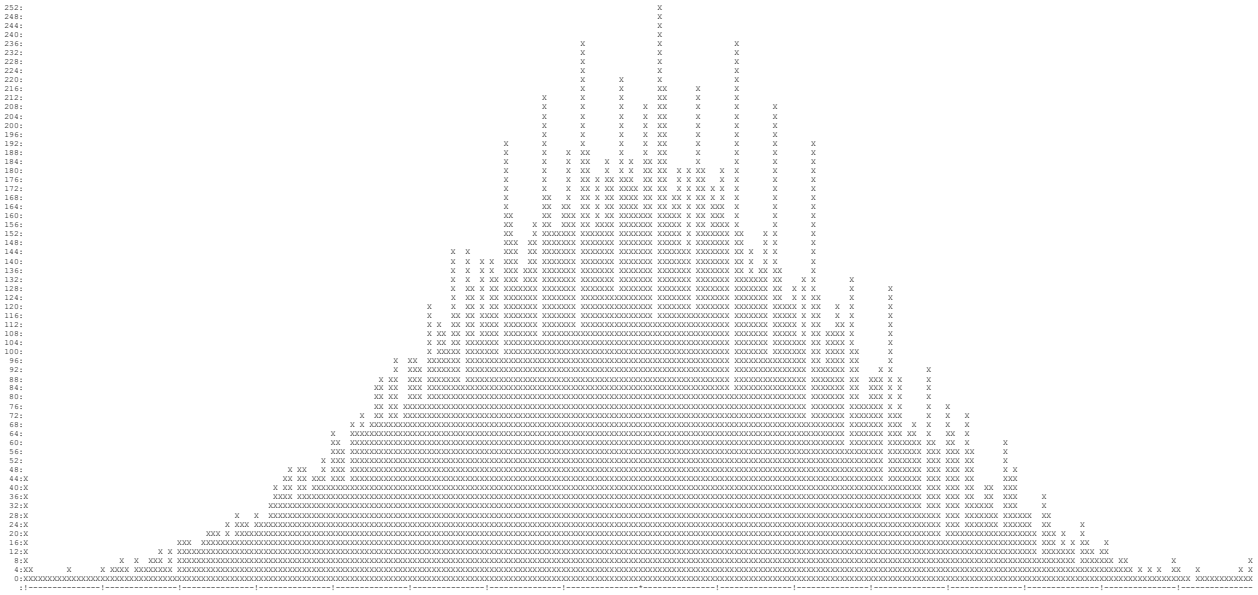
```
:0800000080310228843160 2CDC
:10052200640001308001013 1FE020030FF3B7F0890
:100532007E0403190034922AFC30B2004130A0003C
:100542008830A100072382316B30A0008830A100DF
:100552003208A200A30107238231B301B401F030B3
:10056200FC002030FD0033087C07860034087D3D06
:10057200870001083202031 8C02A5830C12A2030ED
:10058200E123823153248231031933020 31CB02A3E
:100592004130A0008830A10007238231B301B401A9
:1005A2009110911CD22AB30A0319B40A0030340202
:1005B2000A3003193302031CD12AB2080319E42AB0
:1005C2000430B202A52A6630A0008830A1000723B9
:1005D2008231B301B4013308803A3404031DF32A93
:1005E2002A30FA2A33080F39031DF92A2130FA2A50
:1005F2002D30E123823153248231031933020 31C4B
:10060200EC2A4130A0008830A100072B2230A8003C
:1006120020002108472482318 10803190800253A35
:10062200031917 2B3108E1238231092BAF01B001E5
:10063200AA015A248231E8238231031C482BAF01DC
:10064200B0010A30F000F1013008F3002F08F20087
:100652008631AD2682317108B0007008AF005A248D
:100662008231D03EA400FF3003180030A5002408D8
:10067200AF072508B03DA00A0319A10A200884008B
:1006820021 0885000008E82382310318222B210863
:1006920047248231003A0319E02B643A0319552B9F
:1006A2000D3A0319552B092B280886008701403F74
:1006B200AB00413FAC00A80AA80AAC1F672B03306D
:1006C200AA04AB09AC09AB0A0319AC0AB101B10A1D
:1006D20031083B248231A600013FA7002C02031DF2
:1006E200742B26082B02031C7B2BB10A3108053A16
:1006F200031D692B30082F040319872B2A0803399D
:10070200319872BFF30AF07031CB0033108A40085
:10071200A501A41BA5032508803AA6003008803A4B
:1007220026020 31D962B2F0824020318A22B200059
:10073200 3108A400A501A41BA503AF022508B03B04
:10074200A52B2000AF01B00130082F040319B52BEF
:10075200 2030E1238231FF302000AF07031CB003B9
:1007620030082F04031DA92B2A0803390319BC2BB7
:10077200 2D30E123823120003108A900DB2B0A3021
:10078200F700F80129083B248231F000013FF10013
:10079200 2C08F3002B08F200F62382317108FA00CC
:1007A2007008F9001B2482317708303EB10031080D
:1007B200E12382312000A903290FC02B092B080055
:1007C200F000111EE22B700823009A000800F100CD
:1007D2003A30F00171020318F42B30307102031C1D
:1007E200F42BF001F00A700C0800F401F501710815
:1007F20070040319162CF601F60AF11B032CF035CE
:10080200F10DFD2BF435F50D71087302031D0B2C50
:1008120070087202031C122C7008F2027108F33B7A
:100822007414F136F00CF60B032C7508F100740801
:10083200F000080078087704 0319362CFB01FB0A44
:10084200F81B262CF735F80D202C78087A02031DA8
:100852002C2C77087902031C322C7708F9027808CD
:10086200FA3BF836F70CFB0B262C7A08F8007908CD
:10087200F7000800A4000030A501A435A50D240747
:1008820084008830253D8500003F0800A50020082F
:10089200A400A00A0319A10A840025088500000803
:1008A200B10008002000B30A0319B40A013034026F
:1008B200003420088400210885000080800F03078
:1008C200840020308500013 0FE01FF008231912238
:1008D2008431BF01C001C101C2017E102000712C10
:1008E200172784311B27843126278431EC26843153
:1008F200BD01BE019110911C7C2CBD0A0319BE0AD8
:10090200000303E02643003193D02031C7B2C7F271A
:10091200843 1FF268431FB278431CA2684319D2706
:1009220084310319 3D02031C8E2C33308F2783310F
:10093200072384310030 3C028F3003193B02031835
:10094200B32CE83022008E058D0188278431911066
:10095200911CA92CBD0AEA27843103193D02031810
:10096200A62CA82CC001C101BA01632DBA01FB2734
:10097200843 1CA2684319D27843103193D02031C28
:10098200BA2C33308F27833107238431003 3C0265
:100992008F3003193B0203186A2DE83022008E05BE
:1009A2008D0188278431911 0911CD52CBD0AEA272C
:1009B200843103193D020318D22CD42CD827843158
:1009C2000A30FD018F258431632DD827843105300B
:1009D200FD018F258431632DF52784310530FD011A
:1009E2008F258431632DF52784310A30FD018F254F
:1009F2008431632D203022008E060A30FC00143030
:100A0200FD01FD0A8F258431632D22008E16CA2630
:100A12008431B900903039020318062D22008E125B
:100A2200082D882784310B110B1D152DBD0AAF2708
:100A3200843103193D02031C142DFF30F001F00A2A
:100A42002F268431632D882784310B110B1D272D0E
:100A5200BD0AAF27843103193D02031C262D013044
```

```
:100A6200F001F00A2F268431632DBB278431911CBB
:100A7200382DFF30F001F00A2F268431632DBB2779
:100A82008431911C422D0130F001F00A2F2684316D
:100A9200632D882784310B110B1D4D2DBD0AAF2705
:100AA200843103193D02031C4C2DC22684314208B5
:100AB20072278431003A0319632D82319D228431D9
:100AC200FF268431EC2684313230BA0A3A02031806
:100AD200B72C3F08003A0319DF2C013A0319E62C20
:100AE200033A0319ED2C013A0319F42C073A0319BE
:100AF200FB2C0E3A0319082D013A0319122D073A5D
:100B02000319242D013A0319362D033A0319402DF6
:100B1200013A03194A2DDF2C803102282000B2004D
:100B22009110911C922DC226843180304202031C06
:100B3200A92D4208F0008030F101F007E42784314A
:100B4200AD268431EF278431B303B401B40AB42D46
:100B5200DE278431E4278431AD268431EF278431C6
:100B6200B30AB401B4037D080319C22D063038025A
:100B72000F3003193702031CC22D0E30B7000630A6
:100B8200B8005530A0008830C12783310723843153
:100B92002000380837040319E22DB501B6010B1104
:100BA2000B1DD12DB50ACD27843103193502031C43
:100BB200D02D3408F00033082F268431A627843143
:100BC200C92D9110911CE32DC2268431803042023E
:100BD200031CF92D4208F0008030F101F007D22702
:100BE2008431AD2684317108B8007008B7000B2E2D
:100BF200DE278431D2278431AD2684317108B800D2
:100C02007008B7007D0803190B2EB301B303B401BA
:100C1200B4030F2EB301B30AB401B40A2030A0000A
:100C22008830C12783310723843120003808370 4F4
:100C320003190800B501B6010B110B1D1E2EB50AD2
:100C4200CD27843103193502031C1D2E3408F00010
:100C520033082F268431A6278431162EF10071081D
:100C620003196E2E632E22008D010D167108803A33
:100C72007F3E031C402E2000C001C00A702E0330AC
:100C82004B2E22008D010D177108803A7F3E031C06
:100C92004E2E02302000C000702E2000C001702EA7
:100CA20022008D018D167108803A7F3E0318402E76
:100CB2003C2E22008D018D177108803A7F3E031869
:100CC2004E2E4A2E4008013A0319422E033A0319C6
:100CD200512E013A03195A2E342E22008D0170082A
:100CE2000319A92E9D2EE83022008E050E14F01F46
:100CF2007E2E2000C101C10A080003302000C1007D
:100D02000800E83022008E050E15F01F8C2E0230EE
:100D12002000C10008002000C1010800E8302200C4
:100D22008E058E14F01B7E2E7A2EE83022008E0560
:100D32000E16F01B8C2E882E20004108013A031952
:100D4200822E033A03198F2E013A0319962E742E1E
:100D5200E83022008E050800F401F501701CB52E62
:100D62007208F4077308F53DF235F30DF136F00C15
:100D720071087004031DAF2E7508F1007408F000AD
:100D8200800021009D149D18C42E1C082000C200DA
:100D92000800803022009800990079302100 9D00DF
:100DA20020009110911CD32E9110911CD62E21005F
:100DB2009D149D18DA2E1C08FC001D30B427443007
:100DC2002000A0008830A1007C08A200A30183318A
:100DD20007237C080800F00120000C1DF00A0C1A01
:100DE200F42EF00AF00A8C1AF82E0430F0078E1B4B
:100DF200FC2E0830F0077008BF000800F201F30172
:100E0200F030F0002030F1007208700786007308 9D
:100E1200713D87008101F20A0319F30A013073025E
:100E22000030031972020318080001 2F78302100E4
:100E32009900080095129511151195141514FA30A0
:100E420020009B004E309C00080021000C150C165F
:100E52008C168E1723000C110C128E132100951381
:100E620024000C150C168C168E1727000C150C1668
:100E72008C168E1721008E1222008E128D012100F7
:100E82008D0124008D0122008E0121000E108E1092
:100E92000E110E1223000E108E100E118E150E174B
:100EA2008B3022009700C3302A00950024000E13D5
:100EB2001D302100B42714303D00900023000C1097
:100EC20021000C108C1423008C1001303C00A40073
:100ED200203023009E009D171D161E1533309B00E7
:100EE2000800F000F100C7278431810A7108C72782
:100EF20084310108FF3A0319013400340A30A0009A
:100F02008830A1003F08A200A3018331072B2030C3
:100F120022008E062000BD01BE010800BC36BB0CBB
:100F2200BC36BB0CBC36BB0CA0008830A1003C0810
:100F3200A3003B08A2000800BB070318BC0ABD0AB5
:100F42000319BE0A00303E020834013020 00B70205
:100F52000030B83B203022008E0608000319BE0A7A
:100F620000303E0205349D002B309E009F0123007D
:100F72008E150800203022008E062000911008 00F5
:100F8200A1003808A3003708A2000800F03E86003E
:100F920020308701873D08000319B60A0030360267
:100FA2000534031CF1037C08F200F3010800203031
:100FB20022008E060430FC0008008030F000F101AF
:100FC2004208F0020800031CF1033208F200F301A8
:100FD200080003 19BE0A00303E020A347108B80044
:100FE2007008B700B3010800203022008E060830D6
:100FF200FC000800BB01BC01BD01BE0108000134B8
:1010020000340A34003464340034E83403341034D5
:10101200273454347934633468346F34733463342A
:101022006F347034653420346D346F346434653415
:10103200 3D342034253464340D340A3400342034F1
:1010420041347634613 46E34633465343A34253451
:10105200343464342034733474346534703473 3407
:1010620020340034563464346434203 46D346F34A4
:1010720079346534 6E3420343D3420342534643 47C
:101082000D340A3400344D34653473347534723 49B
:10109200653420345634643464343D342034253489
:1010A20064340D340A3400340D340A3452346F344B
:1010B20074343A342534343464342034733474341C
:1010C200653470347334203400342034203420342034B6
:0E10D2003A340034253433346434 3A34003474
:020000040001F9
:04000E00E4CFFFDE5E
:00000001FF
```

Le logiciel est assez simple :

- il initialise les périphériques internes du PIC16F1708
- il contrôle la tension de l'accu avant de démarrer : en fait on teste si le Vcc du PIC est en-dessous de 4.5V ce qui indique un accu faible, bien qu'on aurait pu prélever la tension directe VBAT via un pont diviseur de tension, mais pas assez de broches disponibles.
- le mode courant du sélecteur est testé et enchaîne le bon cas dans une boucle de traitement infinie. Les divers modes permettent de tester le fonctionnement élémentaire du générateur de bruit et des moteurs, et aussi de modifier la façon dont le tychoscope applique les tirages aléatoires sur l'angle de rotation et le déplacement.

Le mode 15 affiche l'histogramme de répartition des titrages aléatoires sur le signal entrant, ce qui permet de vérifier qu'il n'écrête pas et que l'on a bien une gaussienne autour du 2V (soit 128 converti sur 8bits pour une tension de référence de 4V de l'ADC).



Le réglage de gain en face avant permet d'aplatir la gaussienne, ce qui sature les points aux extrémités, mais rend plus équiprobable les tirages des valeurs des tensions converties par le PIC : cela peut être utile.

Le fichier .H

```
/*
 * File:   tychoscope.h
 * Author: croco31
 *
 * Created on 16 novembre 2014, 14:38
 */

#ifndef TYCHOSCOPE_H
#define     TYCHOSCOPE_H
#include <xc.h>

/* Definition des ports utilises sur le PIC16F1708 */
/* ------------------------------------------------*/
// Moteurs unipolaires a 4 bobines: commandes des phases 1= bobine active
// RB4 = X moteur 1
// RB5 = /X moteur 1
// RB6 = Y moteur1
// RB7 = /Y moteur1
// RC0 = X moteur 2
// RC1 = /X moteur 2
// RC2 = Y moteur 2
// RC6 = /Y moteur 2

/* Acquisition du sélecteur de mode 16 positions */
/* ------------------------------------------------*/
/* pullups du port a activer sur ces entrees: le commutateur  met l'entree à 0V */
// RA2 = LSB 0
// RA4 = 1
// RA5 = 2
// RC7 = MSB 4

/* Commande de la LED */
/* -------------------*/
/* LED allumee si RC5= 1 */
#define pLED LATC5

/* AOP interne generateur de bruit */
/* --------------------------------*/
// RC4 = entree inverseuse AOP2 interne
//  entree non iverseuse sur VREF interne a 2V
// RC3 = sortie AOP2 pour rebouclage gain et point test
//  la sortie AOP2 est dirigee sur AN7 en interne

#endif      /* TYCHOSCOPE_H */
```

Le source C :

```
/*
 * File:   tychoscope.c
 * Author: croco31
 * Date: 25/10/14
 * Role: prog de controle du robot tychoscope
 * Modifs:
 *
 * Created on 25 octobre 2014, 12:45
 * Compilation sur Microchip XC8
 */

#include <stdio.h>
#include <stdlib.h>
#include "tychoscope.h"

/* Variables globales */
unsigned char mode; // mode de fonctionnement du logiciel
unsigned char seq_mot1,seq_mot2; // numeros de sequence de chaque moteur
unsigned char val_adc; // valeur 8 bits lue sur ADC

/* Bits de configuration du PIC pour XC8 */
/* #pragma config <Macro Name>=<Setting>, <Macro Name>=<Setting>, ...         */
// #pragma config statements should precede project file includes.
// Use project enums instead of #define for ON and OFF.

// CONFIG1
#pragma config FOSC = INTOSC    // Oscillator Selection Bits (INTOSC oscillator: I/O function on CLKIN pin)
#pragma config WDTE = OFF       // Watchdog Timer Enable (WDT disabled)
#pragma config PWRTE = OFF      // Power-up Timer Enable (PWRT disabled)
#pragma config MCLRE = ON       // MCLR Pin Function Select (MCLR/VPP pin function is MCLR)
#pragma config CP = OFF         // Flash Program Memory Code Protection (Program memory code protection is disabled)
#pragma config BOREN = ON       // Brown-out Reset Enable (Brown-out Reset enabled)
#pragma config CLKOUTEN = OFF   // Clock Out Enable (CLKOUT function is disabled. I/O or oscillator function on the CLKOUT pin)
#pragma config IESO = OFF       // Internal/External Switchover Mode (Internal/External Switchover Mode is disabled)
#pragma config FCMEN = OFF      // Fail-Safe Clock Monitor Enable (Fail-Safe Clock Monitor is disabled)

// CONFIG2
#pragma config WRT = OFF        // Flash Memory Self-Write Protection (Write protection off)
#pragma config PPS1WAY = ON     // Peripheral Pin Select one-way control (The PPSLOCK bit cannot be cleared once it is set by software)
#pragma config ZCDDIS = ON      // Zero-cross detect disable (Zero-cross detect circuit is disabled at POR)
#pragma config PLLEN = OFF      // Phase Lock Loop enable (4x PLL is enabled when software sets the SPLLEN bit)
#pragma config STVREN = ON      // Stack Overflow/Underflow Reset Enable (Stack Overflow or Underflow will cause a Reset)
#pragma config BORV = LO        // Brown-out Reset Voltage Selection (Brown-out Reset Voltage (Vbor), low trip point selected.)
#pragma config LPBOR = OFF      // Low-Power Brown Out Reset (Low-Power BOR is disabled)
#pragma config LVP = OFF        // Low-Voltage Programming Enable (High-voltage on MCLR/VPP must be used for programming)


void putch(char val)
{
    while(!TXIF) continue;
    TXREG= val;
}

void ConfigureOscillator(void)
{
    /* Activation INTOSC a 16MHz PLL= OFF  */
    OSCCON= 0x78; /* SPLLEN=0 IRCF=1111 SCS=00*/
}

/* Activation des timers */
void ConfigureTimers(void)
{
   /* Freq/4 de base = 4MHz */
   /* TIMER0 utilise pour generer un Tick sur TMR0IF et ITs eventuelle */
```

```
   /* Prescaler = 1:16 soit 4uS en entree */
   /* Timer0 reboucle en 4x256uS soit un tick de 1.2ms environ */
    TMR0CS= 0; /* Source = Freq/4 */
    PSA= 0; /* Prescaler */
    PS2 = 0; /* PS = 011 pour 1:16 */
    PS1 = 1;
    PS0 = 1;

    /* Timer2 ticks TMR2IF a 10ms */
    /* utilise Fosc/4 Prescaler = 1/16 PR2=249 Postcaler= 10 */
    PR2= 250;
    T2CON= 0b01001110;

    /* Timer4 et Timer6 sont identiques a Timer2 */
}
/* Configuration des ports du PIC 16F1708 */
void Init_ports(void)
{
    /* port de lecture du selecteur de mode */
    TRISA2= 1; /* Inputs */
    TRISA4= 1;
    TRISA5= 1;
    TRISC7= 1;

    ANSA2= 0; /* Digital IOs */
    ANSA4= 0;
    ANSC7= 0;

    nWPUEN= 0; /* Weak pullups actifs suivant chaque bit de port */
    WPUA2= 1; /* pullups internes pour la lecture du selecteur de mode */
    WPUA4=1;
    WPUA5=1;
    WPUC7=1;

    INLVLA2= 1; /* Inputs en mode ST Vdd/2 */
    INLVLA4= 1;
    INLVLA5= 1;
    INLVLC7= 1;

    /* LED */
    TRISC5= 0;
    pLED= 0;

    /* Commandes des moteurs */
    LATB=0;
    TRISB= 0x00; /* outputs */
    WPUB= 0; /* pas de weak pullups */
    LATC=0;
    TRISC0=0; /* outputs*/
    TRISC1=0;
    TRISC2=0;
    TRISC4=0;
    ANSC0=0; /* Digitals IOs */
    ANSC1=0;
    ANSC2=0;


    /* AOP2 interne */
     ANSC3=1; /* Analog  */
     ANSC6=1; /* AOP2IN- analog input */

    /* activation de AOP2 interne et Vref Buffer2 = 2v */
    FVRCON=  0b10001011; /* Buffer2 = 2xFVR soit 2V pour INPUT+ de OPA2 ,  Buffer1= 4xFVR soit 4V pour Vref de l'ADC */
    OPA2CON= 0b11000011; /* OPA2=ON mode large bande INPUT- sur broche OPA2IN- INPUT+ sur buffer2 FVR= 2V*/
    WPUC6= 0; /* pas de pullup sur OPA2IN-  (RC6)  */

    /* Activation de l'ADC en mode normal*/
    /* VREF+= sortie buffer2 FVR a 4.096V */
    /* Source= AN7 sortie de OPA2 */
    /* ADC clock = Fosc/16 soit 1MHz */
    ADCON0= 0b00011101; /* AN7 GO=0 ADC_ENABLE=1*/
    ADCON1= 0b00101011; /* Left justified, Fosc/16 , VREF+= FVR Buffer1 */
    ADCON2= 0; /* trigger off : ADC active par bit ADGO */
    ANSC3= 1; // analog input

    /* la sortie TXD de l'UART est mise sur broche 19  (ISPDAT) RA0 et RX sur broche 18 (RA1) */
    /* permet d'afficher des messages sur UART_tool du PicKit2 (attention Pickit2 ne sait pas programmer le PIC16F1708) */
    /* seul PICKIT3 connait le PIC16F1708 pour le programmer mais n'a pas Uart_tool*/
    RA0PPS= 0b10100;
    ANSA0= 0; /* digital port car analog par defaut au reset*/
    TRISA0= 0; /*output */
    /* RxD est connecte sur ISPCLK RA1 */
    TRISA1= 1;
    ANSA1= 0;
    RXPPS= 0b00001; /* RA1 source du RX UART */

    /* Configuration vitesse de l'UART a 19200 bauds pour Fosc= 16MHz */
    TX1STA= 0x20; /* TXEN= 1 */
    SPEN= 1;
    CREN= 1; /* active le receiver */
    BRGH= 1;
    SPBRG= 51;
}


/* mesure de la tension d'alimentation */
/* on mesure la tension du DAC configure avec VDD comme reference et valeur 128 moitie soit 2.25V min pour Vdd=4.5V min */
/* lecture du mode courant sur le selecteur */
/* une tension normale Vdd serait de 4.5V au moins en dessous l'accu est decharge */
/* l'ADC est utilise avec 4V de reference et doit donner au moins 2.25V soit 143 converti */
#define MAX_VDD 143
unsigned char mesure_vdd(void)
{
    unsigned char val;
    // configure le DAC
    DAC1CON0= 0b10000000;
    DAC1CON1= 128; // Vdd/2 en sortie du DAC
    // configure ADC pour INPUT=DACOUT  VREF+= FVR buffer1 (4V)
    //
```

```
    ADCON0= 0b01111001; /* IN=DACOUT GO=0 ADC_ENABLE=1*/
    TMR2IF=0; while(!TMR2IF) continue;// attend tick de 10ms
    TMR2IF=0; while(!TMR2IF) continue;// attend tick de 10ms
    ADGO=1; // lance une conversion
    while(ADGO) continue;
    val = ADRESH;
    // reconfigure l'ADC en mode normal
    ADCON0= 0b00011101; /* AN7 GO=0 ADC_ENABLE=1*/
    ADCON1= 0b00101011; /* Left justified, Fosc/16 , VREF+= FVR Buffer1 */
    ADCON2= 0; /* trigger off : ADC active par bit ADGO */
    ANSC3= 1; // analog input

#ifndef NO_PRINTS
    printf("Mesure Vdd= %d\r\n",val);
#endif
    return val;
}

void read_mode()
{
    unsigned char val;
    val=0;
    if( !RA2) val += 1;
    if( !RA4) val += 2;
    if( !RA5) val += 4;
    if( !RC7) val += 8;
    mode= val; /* varie de 0 a F */
}

/* print du mode sur TxD pour debug */
void print_mode()
{
#ifndef NO_PRINTS
    printf("Tychoscope mode= %d\r\n",mode);
#endif
}
/* Lecture de la valeur analogique sur AN7= bruit aleatoire sur 10 bits */
void read_ADC(void)
{
    ADGO=1;
    while(ADGO) continue;
    val_adc= ADRESH; // cadre a gauche
}

#ifndef NO_PRINTS
#define NBPOINTS_HISTOGRAMME 256
unsigned char table_adc[NBPOINTS_HISTOGRAMME]; // table histogramme sur les valeurs aleatoires lues
#endif
unsigned char cumul_histogramme(unsigned char val)
{
#ifndef NO_PRINTS
    unsigned char index;
    /* chaque valeur /2 soit sur 7 bits est ventilee dans les 256 compteurs */
    /* retourne 1 si un compteur atteint 255 */
    index= val;
    (table_adc[index])++;
    if(table_adc[index]==255) return 1;
     else return 0;
#else
    return 0;
#endif
}
void reset_histogramme(void)
{
#ifndef NO_PRINTS
 unsigned int i;
 for(i=0;i<NBPOINTS_HISTOGRAMME;i++) table_adc[i]=0;
#endif
}
void print_histogramme()
{
#ifndef NO_PRINTS
    unsigned char val;
    unsigned int index;
    // affiche l'histogramme sur UART: 64 lignes seulement de 128 colonnes sont affichees
    val=252; // hauteur a comparer
    printf("\r\n");
    do { // chaque ligne teste toute la table
      printf("%3d:",val); // debut de ligne
      for(index=0;index<NBPOINTS_HISTOGRAMME;index++) {
          // chaque compteur est teste et fera une colonne histogramme verticale plus ou moins haute
          if( table_adc[index]>val) putch('X'); else putch(' ');
      }
      printf("\r\n");
      for(index=0;index<10;index++){ /* attend 100ms pour traitement de la ligne par UART_tool du PICkit2 */
         TMR2IF= 0;
         while(!TMR2IF) ;/* attend tick 10ms */
      }
      if( val==0) break;
      val -= 4;
    } while(1);
    // affiche des reperes sur l'axe X
    printf("    :");
    for(index=0;index<NBPOINTS_HISTOGRAMME;index++){
        if(index== (NBPOINTS_HISTOGRAMME/2)) putch('*');
        else
         if( (index%16)==0) putch('!');else putch('-');
    }
    printf("\r\n");
#endif
}


/* Avance/recul simultanee des moteurs 1 et 2  */
/* Unipolaire: sequence X Y /X /Y   */
/* le numero de sequence de chaque moteur est garde en variable globale remanente */
/* sens1/2: sens de chaque moteur 1 1 = avance -1 +1: rotation
   <0 : recule
   0  : pas de mouvement du moteur
```

```
   >0 : avance
   Chaque moteur a 48 pas avec un reducteur 1/64  et une roue D=30mm
   soit 33 pas par mm pour chaque roue
   Les roues ont un entraxe de 60mm soit 188mm a parcourir pour faire une rotation de 360 degres
   soit 188/2 * 33 = 3108 pas pour 360 degres si les 2 roues tournent en sens inverse
   soit 8 pas par degre de rotation environ
 */
#define NB_STEPS_PAR_MM 33
#define NB_STEPS_PAR_DEGRE 8

void Step_moteurs( signed char sens1,signed char sens2)
{
   // Moteur 1
   // --------
   if(sens1 !=0) {
    switch(seq_mot1) {
        default:// X
            LATB = 0; // clear des 4 commandes de phases
            LATB4= 1; // phase X
            if(sens1>0) seq_mot1=1; // avance
             else seq_mot1= 3; // recul
            break;
        case 1: // Y
            LATB = 0;
            LATB6= 1; // phase Y
            if(sens1>0) seq_mot1=2; // avance
             else seq_mot1= 0; // recul
            break;
        case 2: // /X
            LATB= 0;
            LATB5= 1; // phase /X
            if(sens1>0) seq_mot1=3; // avance
             else seq_mot1= 1; // recul
            break;
        case 3: // /Y
            LATB= 0;
            LATB7=1; // phase /Y
            if(sens1>0) seq_mot1=0; // avance
             else seq_mot1= 2; // recul
            break;
    }
   } else {
        LATB= 0;
   }

   // Moteur 2: inverse car monte en opposition
   // --------
   if(sens2 !=0) {
    switch(seq_mot2) {
        default:// X
            LATC = LATC & 0xE8; // clear des 4 commandes de phases
            LATC0= 1; // phase X
            if(sens2<0) seq_mot2=1; // avance
             else seq_mot2= 3; // recul
            break;
        case 1: // Y
            LATC = LATC & 0xE8;
            LATC2= 1; // phase Y
            if(sens2<0) seq_mot2=2; // avance
             else seq_mot2= 0; // recul
            break;
        case 2: // /X
            LATC = LATC & 0xE8;
            LATC1= 1; // phase /X
            if(sens2<0) seq_mot2=3; // avance
             else seq_mot2= 1; // recul
            break;
        case 3: // /Y
            LATC = LATC & 0xE8;
            LATC4= 1; // Phase /Y
            if(sens2<0) seq_mot2=0; // avance
             else seq_mot2= 2; // recul
            break;
    }
   } else {
        LATC = LATC & 0xE8;
   }
}
// ---------------------------------------------------------------------
// mouvement du tychoscope fonction des parametres de reglage
// ---------------------------------------------------------------------
// coeff_rot: mutliplieur du nombre de pas en rotation
// coeff_avance: multiplieur du nombre de pas en avance
// recul : si !=0 cela autorise un recul sur l'avance et limite la rotation a 1550 pas
void Mouvement(unsigned char coeff_rot,unsigned char coeff_avance,unsigned char recul)
{
    unsigned int i,nbpas;
    signed char sens1,sens2;
 // ---tirage de la rotation
                // attente de 10ms pour stabiliser
                TMR2IF=0;
                while(!TMR2IF);
                read_ADC(); // valeur lue sur 8 bits de 0 à 255 centree sur 128 en courbe gaussienne
                // calcul du nombre de pas de rotation: si 128 rotation 0 0= -180deg 255= +180deg
                // 1550 pas = 180 degres
                if( val_adc >=128) {
                    nbpas= (val_adc - 128)*coeff_rot; // 10  limite a 1280 pas dans ce sens
                    sens1= -1;
                    sens2= 1;
                } else {
                    nbpas= (128-val_adc)*coeff_rot; // 10  limite a 1280 pas dans ce sens
                    sens1= 1;
                    sens2= -1;
                }
                if( recul) { // cela limite a 180 deg en cas de coeff important
                    if(nbpas >1550) nbpas= 1550;
                }
                // --- affichage du tirage
#ifndef NO_PRINTS
```

```
                printf("\r\nRot:%4d steps ",nbpas);
#endif
                // ---rotation des moteurs a 200Hz
                while(nbpas!=0) {
                 for(i=0;i<5;i++) {
                  TMR0IF= 0;
                  while(!TMR0IF); /*  attend le tick de 1.2ms */
                 }
                 Step_moteurs(sens1,sens2);
                 nbpas--;
                 pLED= !pLED;
                }
                // ---tirage de l'avance
                // attente de 10ms pour stabiliser
                TMR2IF=0;
                while(!TMR2IF);
                read_ADC(); // valeur lue sur 8 bits de 0 à 255 centree sur 128 en courbe gaussienne
                // calcul du nombre de pas en ligne droite
                // on limite a 10mm maxi soit 330 pas environ
                // un tirage de 128 ne fait pas bouger
                if( val_adc >=128) {
                    nbpas= (val_adc - 128)*coeff_avance; // cela limite a 128x3 pas
                    sens1= 1;
                    sens2= 1;
                } else {
                    nbpas= (128-val_adc)*coeff_avance;
                    // si recul actif on peut aller en arriere pour les tirages <128
                    if(recul) {
                        sens1= -1;
                        sens2= -1;
                    } else {
                        sens1= 1;
                        sens2= 1;
                    }
                }
#ifndef NO_PRINTS
                printf(" Avance:%4d steps ",nbpas);
#endif
                // --- avance des moteurs a 200Hz
                while(nbpas!=0) {
                 for(i=0;i<5;i++) {
                  TMR0IF= 0;
                  while(!TMR0IF); /*  attend le tick de 1.2ms */
                 }
                 Step_moteurs(sens1,sens2);
                 nbpas--;
                 pLED= !pLED;
                }
}
/*
 */
void  main(void) {
    unsigned int i,cumul;
    unsigned char cpt;

    /* configuration de l'oscillateur interne */
    ConfigureOscillator();

    /* activation des timers */
    ConfigureTimers();

    /* Init des ports du PIC */
    Init_ports();


    /* Lecture du mode initial */
    read_mode();

#ifndef NO_PRINTS
    for(i=0;i<100;i++){ /* attend 1 sec pour laisser demarrer UART_tool du PICkit2 */
         TMR2IF= 0;
         while(!TMR2IF) ;/* attend tick 10ms */
    }
    print_mode();

    reset_histogramme(); // clear de l'histogramme
#endif


    /* controle de Vdd qui doit etre a 4.5V au moins sinon l'ADC et FVR ne marchera pas */
    /* valeur moyenne sur 8 points */
    cumul= 0;
    for(i=0;i<8;i++) {
        cumul += mesure_vdd();
    }
    cumul = cumul/8;
#ifndef NO_PRINTS
    printf("Vdd moyen = %d\r\n",cumul);
#endif
    if( cumul < MAX_VDD ) {
        // tension trop faible: accu decharge
        // bloque ici en clignotant et coupure des moteurs
        LATC = LATC & 0xE8; // clear des 4 commandes de phases
        LATB = 0; // clear des 4 commandes de phases
        while(1) {
            pLED= !pLED;
            for(i=0;i<10;i++){ /* clignote a 100ms*/
              TMR2IF= 0;
              while(!TMR2IF) ;/* attend tick 10ms */
            }
        }
    }

    /* boucle d'acquisition et commande moteurs */
    seq_mot1= 0;
    seq_mot2= 0;

    cpt=0;
    while(1) {
```

```
        read_mode(); // lecture periodique du mode courant du selecteur
        cpt++;
    if( cpt>=50) {
     cpt=0;
    // --------------- teste la tension vdd periodiquement
    /* controle de Vdd qui doit etre a 4.5V au moins sinon l'ADC et FVR ne marchera pas */
    /* valeur moyenne sur 8 points */
    cumul= 0;
    for(i=0;i<8;i++) {
        cumul += mesure_vdd();
    }
    cumul = cumul/8;
#ifndef NO_PRINTS
    printf("Vdd moyen = %d\r\n",cumul);
#endif
    if( cumul < MAX_VDD ) {
        // tension trop faible: accu decharge
        // bloque ici en clignotant et coupure des moteurs
        LATC = LATC & 0xE8; // clear des 4 commandes de phases
        LATB = 0; // clear des 4 commandes de phases
        while(1) {
            pLED= !pLED;
            for(i=0;i<10;i++){ /* clignote a 100ms*/
              TMR2IF= 0;
              while(!TMR2IF) ;/* attend tick 10ms */
            }
        }
    }
    // -------------------------------------------------------
        }
        switch(mode){
            default:
            case 0:  // ---------------- cas nominal: avance et rotation aleatoire
                pLED= !pLED;
                Mouvement(10,4,0); // rot max 180deg avance max 15mm
                break;
            case 1:  // ---------------- cas : rotation limitee a 90 deg avance 15mm
                pLED= !pLED;
                Mouvement(5,4,0); // rot max 90deg avance max 15mm
                break;
            case 2:  // ---------------- cas : rotation limitee a 90 deg avance 30mm
                pLED= !pLED;
                Mouvement(5,8,0); // rot max 90deg avance max 30mm
                break;
            case 3:  // ---------------- cas : rotation 180 deg avance 30mm
                pLED= !pLED;
                Mouvement(10,8,0); // rot max 180deg avance max 30mm
                break;
            case 4:  // ---------------- cas : rotation 180 deg amplifiee  avance 40mm avec recul actif
                pLED= !pLED;
                Mouvement(20,10,1); // rot max 180deg avance max 30mm avec recul
                break;


            case 10: // ------------- test de la tension d'alim par conversion permanente et affichage sur UART
                while(1) {
                    if( mesure_vdd() > MAX_VDD) pLED=1; else pLED=0;
                }
                break;

            case 11: // -------------- test rotation et centrage  des moteurs pas a 200Hz
                 pLED= !pLED;
                 for(i=0;i<5;i++) {
                  TMR0IF= 0;
                  while(!TMR0IF); /*  attend le tick de 1.2ms */
                 }
                 Step_moteurs(-1,1); // les deux a la fois pas a 200Hz
                break;

            case 12: // -------------- test avance des moteurs en ligne pas a 200Hz
                 pLED= !pLED;
                 for(i=0;i<5;i++) {
                  TMR0IF= 0;
                  while(!TMR0IF); /*  attend le tick de 1.2ms */
                 }
                 Step_moteurs(1,1); // les deux a la fois pas a 200Hz
                break;

            case 13: // -------------- test rotation et centrage  des moteurs pas a 100Hz
                 pLED= !pLED;
                 TMR2IF= 0;
                 while(!TMR2IF); /*  attend le tick de 10ms */
                 Step_moteurs(-1,1); // les deux a la fois pas a 100Hz
                break;

            case 14: // -------------- test avance des moteurs en ligne pas a 100Hz
                 pLED= !pLED;
                 TMR2IF= 0;
                 while(!TMR2IF); /*  attend le tick de 10ms */
                 Step_moteurs(1,1); // les deux a la fois pas a 100Hz
                break;

            case 15: // --------------  autres cas de test calcul de l'histogramme a 200Hz
                 // lecture valeur aleatoire
                 pLED= !pLED;
                 for(i=0;i<5;i++) {
                     TMR0IF=0; while(!TMR0IF); /*  attend le tick de 1.2ms */
                 }
                 read_ADC();
                 if( cumul_histogramme(val_adc) ) { // un compteur a atteint 255: l'histogramme est pret
                  print_histogramme();
                  reset_histogramme();
                 }
                 break;
        }

    }
}
```