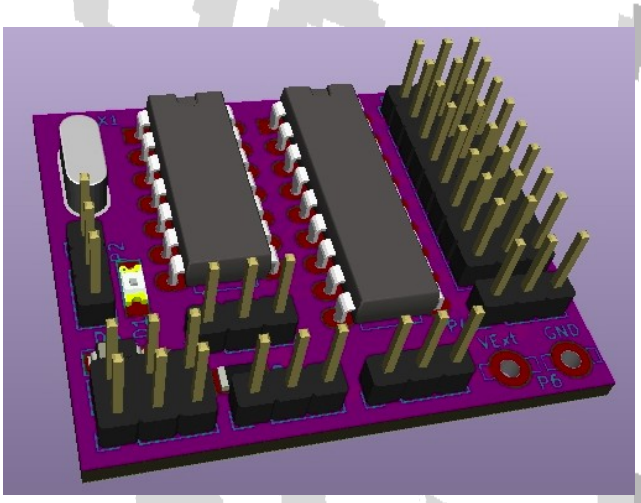
MS8-Xany



Bien plus qu'un simple MultiSwitch à 8 sorties pour *OpenAVRc*

Manuel Utilisateur MultiSwitch MS8-Xany

Table des matières

1 CE DOCUMENT

1.1 Versions

La version de ce document est la dernière listée dans la colonne ***version*** de la table ci-dessous :

Version	*Date*	*Raison de l'évolution*
0.1	20/12/2018	Création
0.2	01/06/2020	ATtiny84 et ULN2803 sur support tulipe, correction sérigraphie diode
0.3	29/03/2021	- Ajout photo Circuit Imprimé - Correction position cavalier ↓ et = pour P1 - Ajout brochage de P9 pour adaptateur USB/Série type « FTDI » - Correction mineure dans table des commandes - Ajout programmation de l'ATtiny84 à l'aide d'un arduino UNO configuré en ArduinoISP
0.4	27/07/2021	Ajout des commandes en mode « impulsionnel » (Release MS8 V0.5)

1.2 Copyright

1.3 Avertissement

L'équipe **OpenAVRc** n'est aucunement responsable des dommages qui pourraient découler de la mauvaise utilisation ou d'un éventuel dysfonctionnement de l'émetteur **OpenAVRc**, du module décodeur **MS8-Xany** et/ou des logiciels associés.

Il appartient donc à l'utilisateur final d'en mesurer, d'en assumer les risques et de respecter la législation en vigueur selon le pays d'utilisation.

1.4 Contenu

Ce document décrit la réalisation du module décodeur **MS8-Xany** ainsi que le paramétrage pour son utilisation avec l'émetteur **OpenAVRc**.

MS8-Xany est un module décodeur MultiSwitch disposant de 8 sorties « Tout-Ou-Rien ».

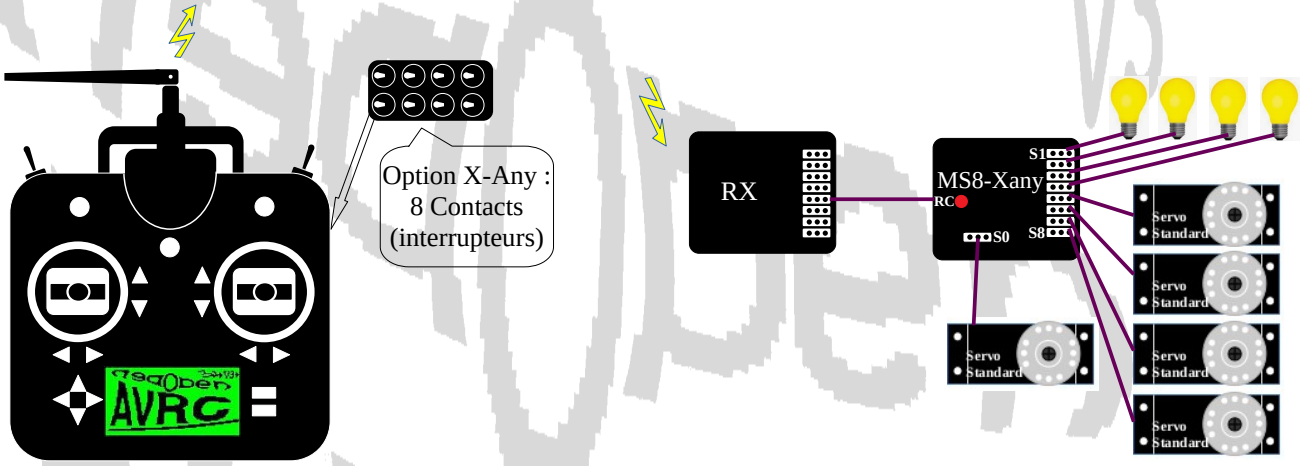
En plus de la fonction MultiSwitch classique, **MS8-Xany** peut être paramétré afin que ses sorties pilotent des servos en « Tout-Ou-Rien ».

Enfin, **MS8-Xany** peut fournir une voie proportionnelle d'appoint permettant la gestion d'un servo ou d'un ESC.

Il est également possible de commander chaque sortie en mode « impulsionnel ».

2 PRESENTATION DE MS8-XANY

2.1 Vue d'ensemble



2.2 Spécifications du décodeur MS8-Xany

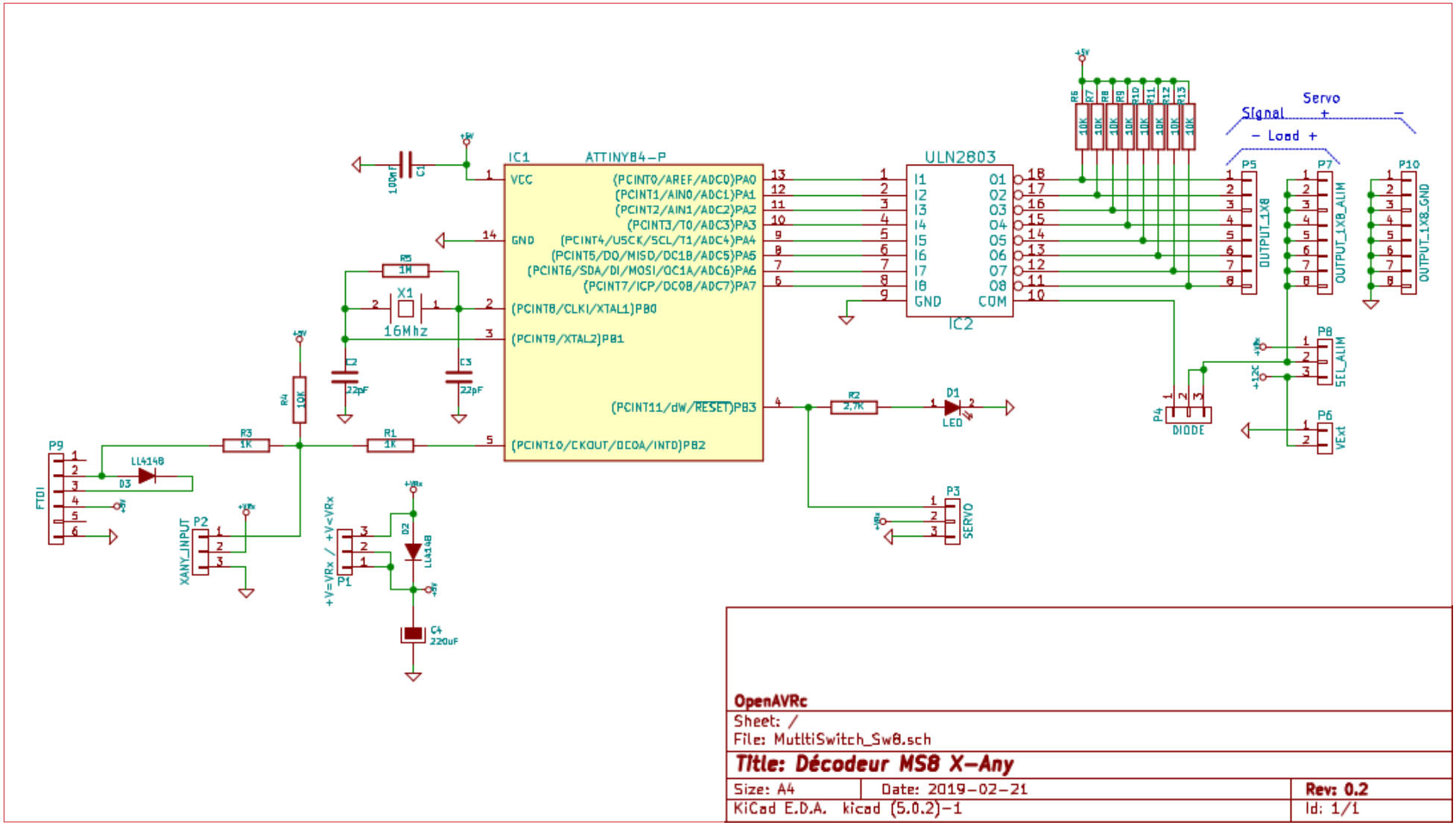
Spécification	Valeur / Caractéristique	Note
Alimentation	+3.3V à +6.6V	Mettre le cavalier « tension récepteur » sur « = » ou « ↓ » selon la valeur de la tension fournie par le récepteur
Protocole X-Any	- Protocole numérique universel utilisé par OpenAVRc pour les accessoires distants - Contrôle d'intégrité par checksum 8 bits - Fonctionne avec tous les protocoles, y compris en 2.4GHz : - Protocole PPM - Protocoles SPIRfMod - Protocoles MultiMod	Contrairement à bon nombre de modules MultiSwitches, **MS8-Xany** fonctionne également avec les modules HF en 2.4GHz
8 sorties Tout-Ou-Rien	- Commandées en Tout-Ou-Rien par sortie transistorisée	Sorties configurées en mode « numérique » (MultiSwitch)
Alimentation des sorties	- Interne : tension du récepteur - Externe : tension externe 5 à 24V	Sélectionnable par cavalier (commun aux 8 sorties)
Diode de roue libre	Diodes de roue libre des 8 sorties au + de l'alimentation des sorties	Sélectionnable par cavalier (commun aux 8 sorties)
8 sorties Servo	- Commandées en Tout-Ou-Rien - Positions extrêmes paramétrables - Durée du mouvement entre positions extrêmes paramétrable	- Sorties configurées en mode «Servo» - Inversion de la course des servos possible en permuttant les valeurs extrêmes - Possibilité d'utiliser la tension du récepteur - Possibilité d'utiliser un tension externe (compatible avec les servos)
1 sortie Servo proportionnelle	Commande proportionnelle d'un servo de 988µs à 2008µs (0 à 255 pas de 4µs)	Présence de commande de servo détectée en dynamique par **MS8-Xany** : rien à configurer sauf côté émetteur OpenAVRc
LED rouge perte de signal	Au bout de 1.5 secondes sans signal	Non gérée si servo proportionnel utilisé (broche partagée)
Failsafe	- Toutes les sorties passent à 0 en cas de perte de signal RC - Le servo proportionnel conserve sa position courante	Synchrone avec LED rouge perte de signal
Port série TTL	Connecteur pour câble USB/FTDI TTL	Pour configuration avancée à l'aide d'une application type « Terminal série »
Dimensions (mm)	L x l x h : 39 x 33 x 12	

Page left intentionnally blank

/

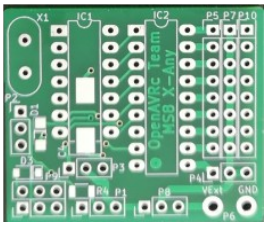
Page laissée intentionnellement blanche

3 SCHEMA DU DECODEUR MS8-XANY



4 REALISATION DU DECODEUR MS8-XANY

4.1 Circuit imprimé



4.2 Montage des circuits intégrés sur support tulipe

4.2.1 Montage de l'ATtiny84 sur support tulipe

ATTENTION :

L'***ATtiny84*** doit être programmé **avant** son positionnement sur le circuit imprimé de **MS8-Xany**.

C'est pourquoi, il est fortement recommandé de le monter sur un support tulipe de 14 points :

Il est également possible d'utiliser 2 portions (de 7 points) de barrette tulipe sécable :

Dans les 2 cas, bien vérifier l'orientation de l'***ATtiny84*** avant de mettre sous tension **MS8-Xany**.

4.2.2 Montage de l'ULN2803 sur support tulipe

En cas d'erreur de câblage sur les sorties, il est possible de détruire l'***ULN2803***.

Afin de facilter son éventel remplacement, il est fortement recommandé de le monter sur un support tulipe de 18 points :

Il est également possible d'utiliser 2 portions (de 9 points) de barrette tulipe sécable :

Dans les 2 cas, bien vérifier l'orientation de l'***ULN2803*** avant de mettre sous tension **MS8-Xany**.

4.3 Chargement du Firmware et configuration des fusibles

La programmation du microcontrôleur ***ATtiny84*** se fait en 2 phases (dans cet ordre) :

1. Chargement du Firmware
2. Configuration des fusibles

4.3.1 Chargement du Firmware dans l'ATtiny84

ATTENTION :

Le connecteur P9 (noté FTDI) **n'est pas** un connecteur de **programmation**, mais un connecteur de **paramétrage** de **MS8-Xany** (Voir §Mode avancé/Commande de servos)

Il existe 2 firmwares :

- **MS8_Xany_PWM_Vx_y.hex** : **MS8-Xany** est alors piloté par une sortie ***PWM*** du récepteur
- **MS8_Xany_CPPM_Vx_y.hex** : **MS8-Xany** est alors piloté par la sortie ***CPPM*** du récepteur

C'est la version ***PWM*** **MS8_Xany_PWM_Vx_y.hex** qui sera la plus couramment utilisée.

A l'aide d'un ***programmateur ICSP*** pour microcontrôleur AVR, charger le fichier **MS8_Xany_PWM_Vx_y.hex** dans l'***ATtiny84***.

Pour cela nous allons utiliser un ***arduino UNO*** configuré en ***programmateur ICSP*** pour charger l'***ATtiny84*** avec le ***fichier HEX*** et les bonnes ***valeurs de fusibles*** : voir le paragraphe suivant.

Toute la phase de programmation (**chargement du Firmware** + **configuration des fusibles**) se fait **avant** de placer l'***ATtiny84*** sur son support tulipe sur le circuit imprimé de ***MS8-Xany***.

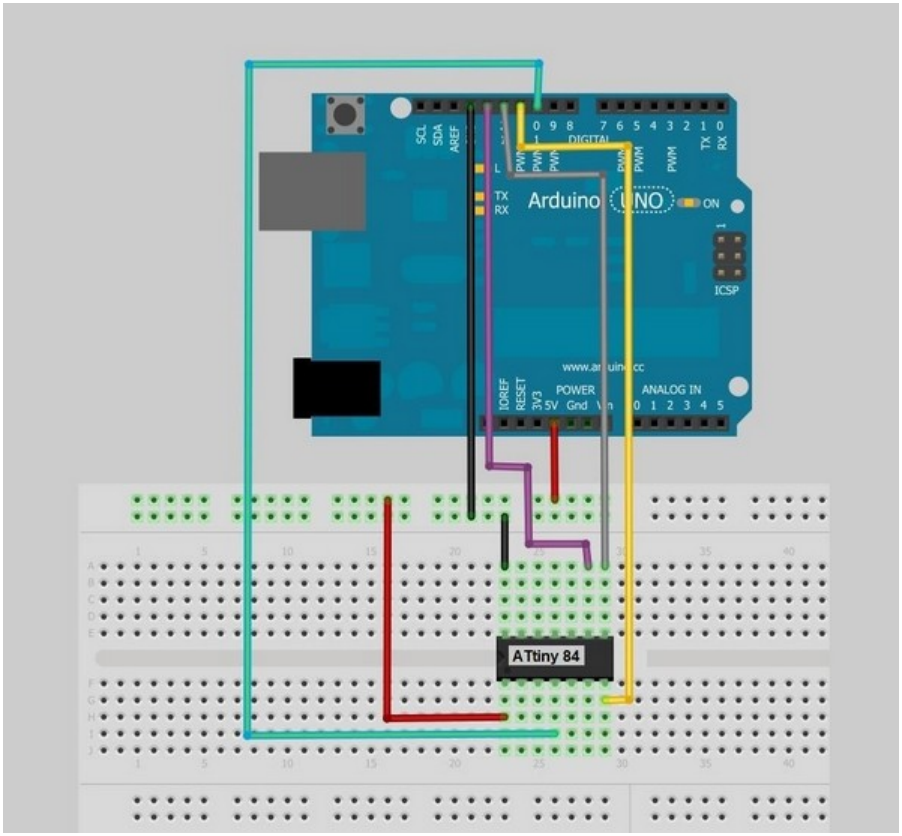
4.3.2 Arduino UNO en programmateur ICSP sous Windows

Connecter votre **arduino UNO** et à l'aide de l'**IDE arduino,** charger le sketch **ArduinoISP** par :

Fichiers → Exemples → 11.ArduinoISP

Cliquer sur ***téléverser*** : votre ***arduino UNO*** est désormais un ***programmateur ICSP***.

Réaliser le câblage suivant pour programmer l'***ATtiny84*** à l'aide de l'***arduino UNO*** :



Copier le sous-répertoire **Windows** du répertoire **PROG** sur votre PC. Ouvrir une **console DOS** (invite de commande) et aller dans ce répertoire **Windows**.

Si nécessaire, éditer le fichier **prog_ms8.bat** à l'aide d'un éditeur de texte afin de redéfinir :

1. le bon **port série** utilisé par l'**arduino UNO : COMx**
2. le **fichier HEX** que vous désirez charger :

   (**MS8_Xany_PWM_Vx.y._HEX**, ou **MS8_Xany_CPPM_Vx.y._HEX**)

   Si vous ne savez pas lequel prendre, prenez **MS8_Xany_PWM_Vx.y._HEX**.

```
REM Adjust below the Serial Port used by your Arduino UNO (Look at Tools->Port, or Outils->Port in the Arduino IDE)
SET SERIAL_PORT = COM4

REM Define here the HEX file you want to load (this HEX file shall be in the current directory)
SET HEX_FILE = MS8_Xany_PWM_V0_4._HEX
```

Dans la ***console DOS***, lancer la commande **prog_ms8.bat** et suivre les instructions.

Le **fichier HEX** sera chargé et les **fusibles** seront automatiquement programmés.

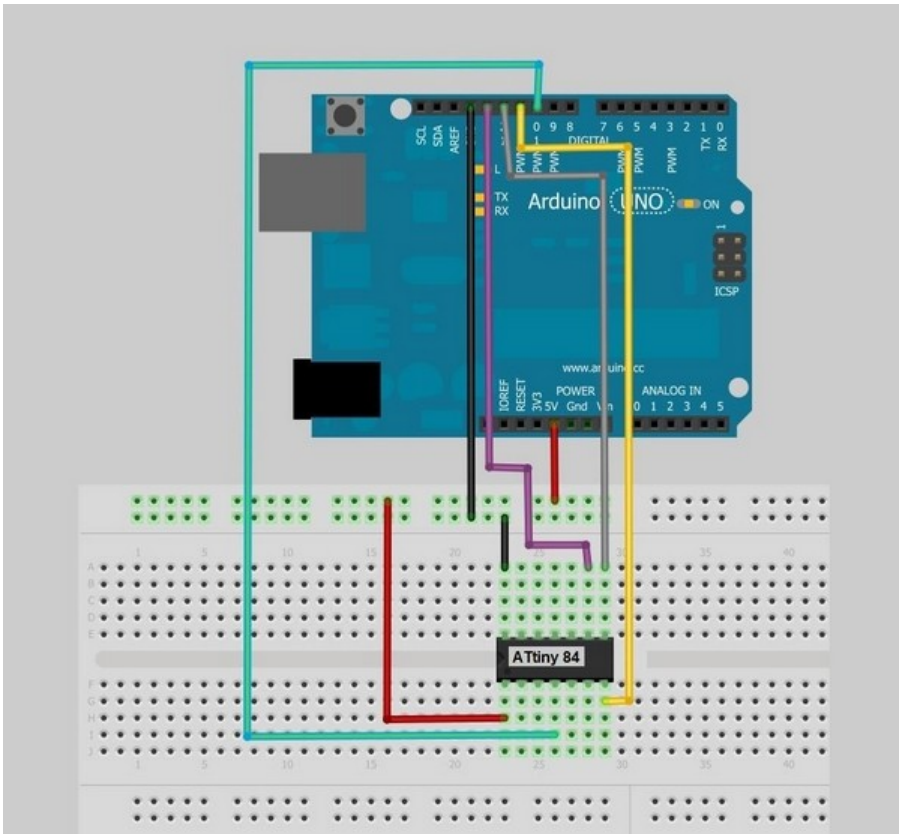
4.3.3 Arduino UNO en programmateur ICSP sous Linux

Connecter votre **arduino UNO** et à l'aide de l'**IDE arduino,** charger le sketch **ArduinoISP** par :

Fichiers → Exemples → 11.ArduinoISP

Cliquer sur ***téléverser*** : votre ***arduino UNO*** est désormais un ***programmateur ICSP***.

Réaliser le câblage suivant pour programmer l'***ATtiny84*** à l'aide de l'***arduino UNO*** :



Copier le sous-répertoire **Linux** du répertoire **PROG** sur votre PC. Ouvrir une **console Linux** et aller dans ce répertoire **Linux**.

Si nécessaire, éditer le fichier **prog_ms8.sh** à l'aide d'un éditeur de texte afin de redéfinir :

3. le bon **port série** utilisé par l'**arduino UNO : par exemple, */dev/ttyACM0***
4. le **fichier HEX** que vous désirez charger :

   (**MS8_Xany_PWM_Vx.y._HEX**, ou **MS8_Xany_CPPM_Vx.y._HEX**)

   Si vous ne savez pas lequel prendre, prenez **MS8_Xany_PWM_Vx.y._HEX**.

```
# Adjust below the Serial Port used by your Arduino UNO (Look at Tools->Port, or Outils->Port in the Arduino IDE)
SERIAL_PORT=/dev/ttyACM0

# Define here the HEX file you want to load (this HEX file shall be in the current directory)
HEX_FILE=MS8_Xany_PWM_V0_4._HEX
```

Dans la ***console Linux***, lancer la commande **./prog_ms8.sh** et suivre les instructions.

Le **fichier HEX** sera chargé et les **fusibles** seront automatiquement programmés.

4.3.4 Note très importante à propos de la valeur des fusibles

Une fois l'***ATtiny84*** programmé (avec ses fusibles), il n'est plus possible de reprogrammer directement l'***ATtiny84*** avec un programmateur **ICSP** puisque celui-ci a besoin de la fonction ***Reset*** de l'***ATtiny84***.

En effet, le programme ***MS8-Xany*** a besoin d'utiliser la broche ***Reset*** mais ***configurée en sortie*** pour piloter la led « Signal Lost » et pour contrôler la sortie proportionnelle.

Si, pour x raison, vous devez reprogrammer votre ***ATtiny84***, il sera nécessaire de restaurer la fonction ***Reset*** de la broche ***Reset***.

Pour cela il est nécessaire d'appliquer un ***signal calibré de 12V*** sur la broche ***Reset***. Il faut alors utiliser un petit boîtier de type « ***Fuse Resetter*** ».

Après cette manipulation, l'***ATtiny84*** redevient programmable à l'aide d'un ***programmateur ICSP***.

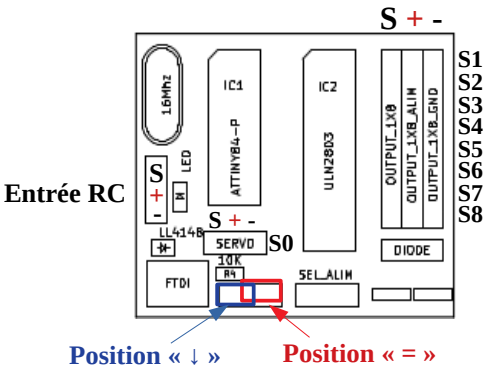
5 UTILISATION

5.1 Connexion au récepteur

Avant de connecter **MS8-Xany** au récepteur, il est impératif de mesurer la tension fournie par le récepteur.

Si la tension disponible entre les broches – et + du connecteur 3 points de la voie utilisée est :

1. Inférieure à 5.7V, placer le cavalier « tension récepteur » sur « **=** »
2. Supérieure à 5.7V, placer le cavalier « tension récepteur » sur « ↓ »



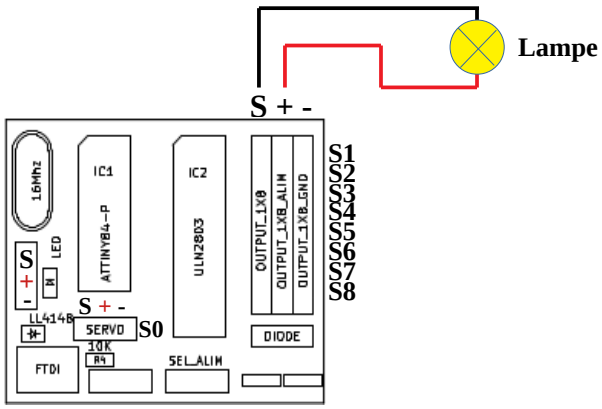
5.2 Mode standard MultiSwitch/Commutation Tout-Ou-Rien

MultiSwitch, c'est le mode par défaut après chargement du Firmware : le mode commutation Tout-Ou-Rien des 8 sorties **S1** à **S8**. Il n'y a donc rien à configurer pour fonctionner en mode MultiSwitch.

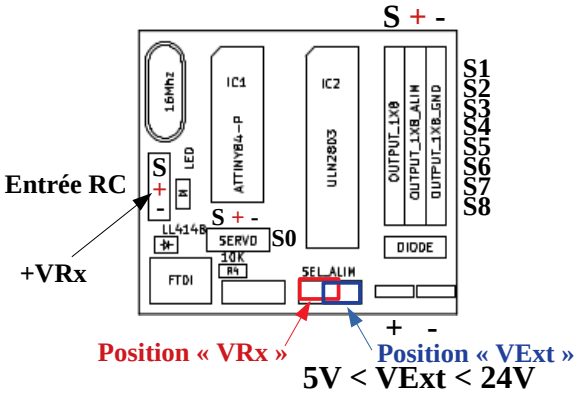
5.2.1 Câblage des « utilisations » Tout-Ou-Rien sur les sorties

Le **MS8-Xany** s'utilise alors comme un module MultiSwitch standard :

- Les « utilisations » (ex : une lampe) se branchent sur les sorties **S1** à **S8** entre les broches « **S** » et « **+** », la rangée 8 points de « **-** » en bord de carte n'est pas utilisée dans ce cas.



5.2.2 Sélection de la tension d'alimentation des sorties S1 à S8



Il est possible d'alimenter les sorties **S1** à **S8** (fourniture du **+**) à partir :

1. De la tension du **+VRx** fournie par le récepteur (Attention à la consommation sur les sorties !)

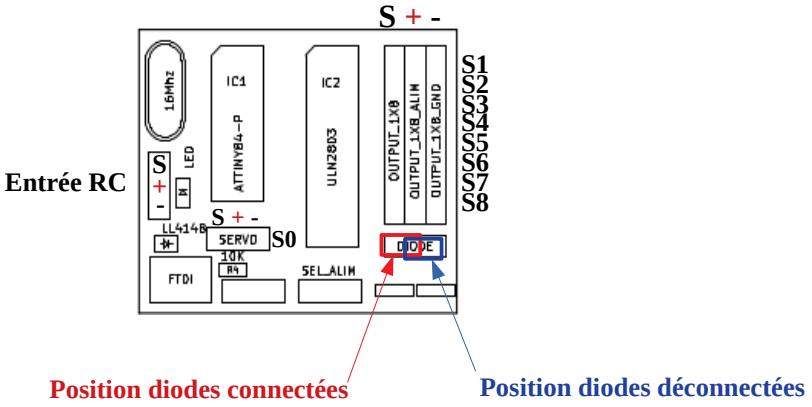
   →Cavalier de sélection alimentation sur « **VRx** »

2. D'une tension externe **VExt** (de **5V** à **24V**) appliquée sur le connecteur 2 points en bas à droite de la carte

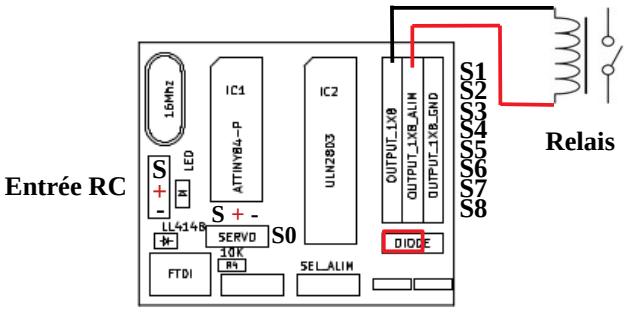
   →Cavalier de sélection alimentation sur « **VExt** »

5.2.3 Diodes de roue libre

Si les « utilisations » connectées sur les sorties sont selfiques (ex :relais), il faut connecter les diodes internes de roue libre, sinon il y a risque de détruire l'ULN2803.



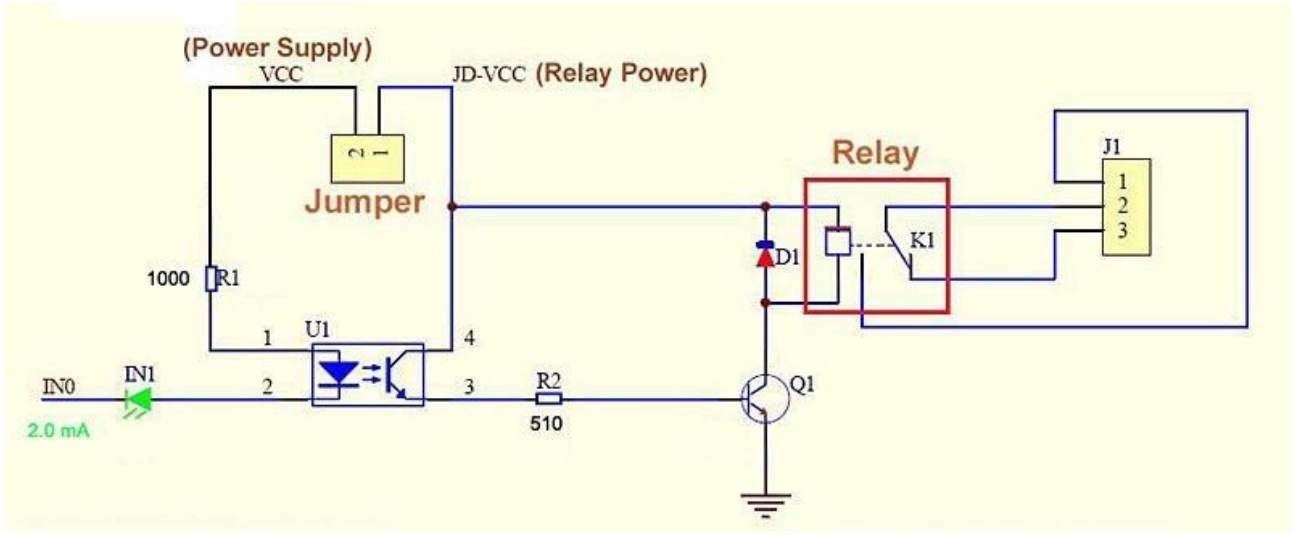
5.2.4 Commande directe de relais avec diode intégrée à l'ULN2803



5.2.5 Commande de relais opto-isolés

Il existe des « modules relais » très bon marché souvent appelés « Arduino Relay Module ». Ces modules intègrent un opto-coupleur permettant une isolation totale entre la tension de commande et la tension d'alimentation des bobines des relais.

Le schéma équivalent d'une voie de ces modules est donné ci-dessous :

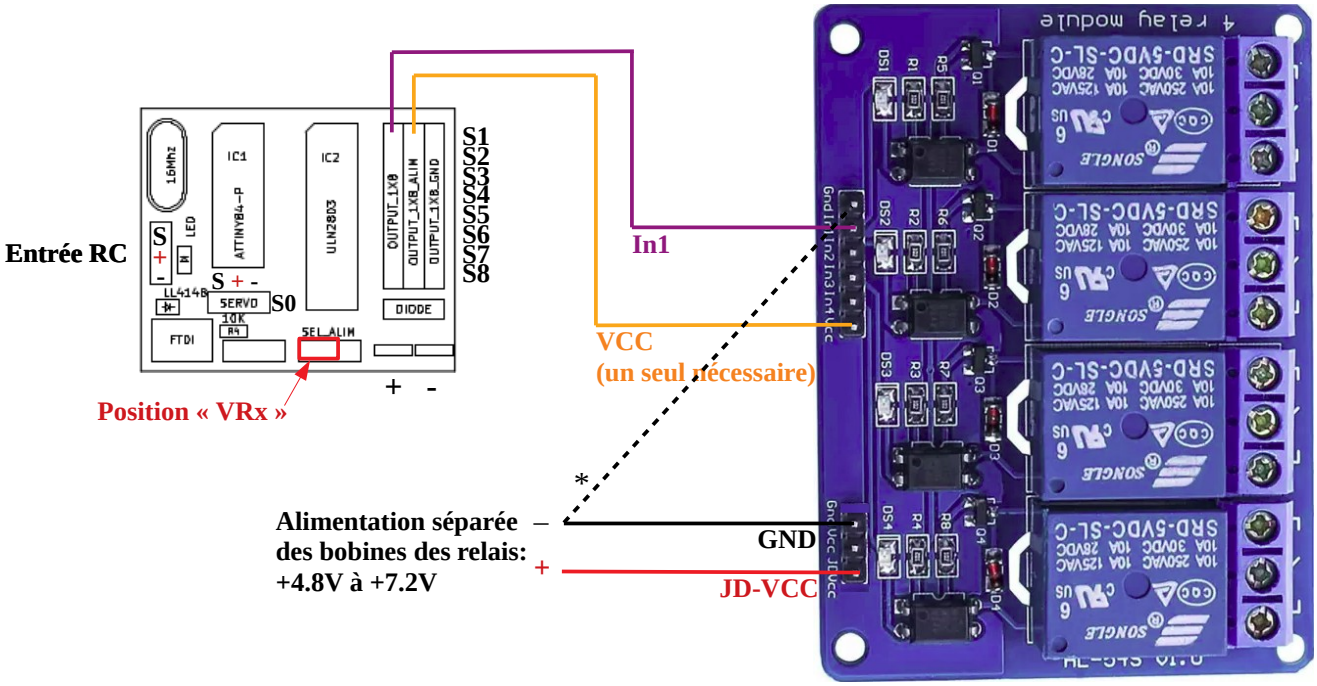


- **VCC** est la tension de commande des opto-coupleurs : accessible sur le connecteur de bord de carte
- **JD-VCC** est la tension de commande des bobines des relais : accessible sur un des points du connecteur « Jumper »

Note importante :

Pour bénéficier d'une isolation totale entre VCC et JD-VCC, le Jumper jaune (cavalier) doit être retiré.

5.2.6 Montage conseillé pour relais 5V opto-isolés



- Les opto-coupleurs sont alimentés par la tension du récepteur (conso : quelques x10 mA)
- VCC de la carte relais est relié à un des points de OUTPUT_1X8_ALIM
- In1 de la carte relais est reliée à la sortie S1
- In2 de la carte relais est reliée à la sortie S2, etc
- Les bobines des relais sont alimentées par une alimentation séparée (+4.8V à +7.2V)

→ Les alimentations étant complètement isolées, la tension du récepteur ne sera pas perturbée lors des commutations des relais : aucun risque de perdre le contrôle RC.

***** : sur certains modèles de « module relais », il n'y a pas de broche **GND** à proximité du **JD-VCC**. Dans ce cas, utiliser la broche **GND** à proximité immédiate de la broche **In1** sur l'autre connecteur.

5.3 Configuration X-Any côté émetteur OpenAVRc

Reportez vous au manuel d'**OpenAVRc** pour configurer l'instance X-Any avec les paramètres suivants :

1. Le N° de voie doit correspondre au N° de voie sur laquelle le décodeur **MS8-Xany** sera connecté sur le récepteur
2. Le nombre de répétition sera en premier lieu réglé sur 3 (dès que ça fonctionnera, il sera possible de réduire cette valeur afin d'atteindre la réactivité maximale autorisée par votre ensemble HF).
3. Configurer « Sw. » sur Sw.8 : cela va transmettre l'état de 8 contacts
4. Si le servo proportionnel sera utilisé sur **MS8-Xany**, sélectionner un des choix proposés par « Prop. », cela va ajouter la transmission de la valeur du proportionnelle.

5.4 Mode avancé/Commande de servos

Le décodeur **MS8-Xany** dispose d'un accès pour les configurations avancées : un port série TTL.

C'est cet accès série qui va permettre l'utilisation de servos connectés au sorties **S1** à **S8**.

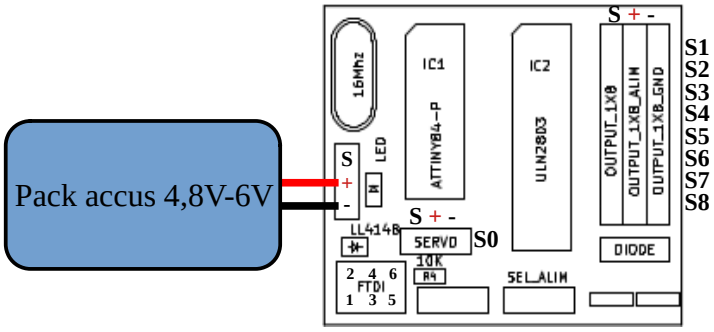
Dans ce cas, la tension « **+** » de sortie devra impérativement être compatible des servos !

5.4.1 *Utilisation du port série de MS8-Xany*

Pour accéder au port série de **MS8-Xany**, il faut un câble USB/Série TTL 5V par exemple de type « FTDI ».

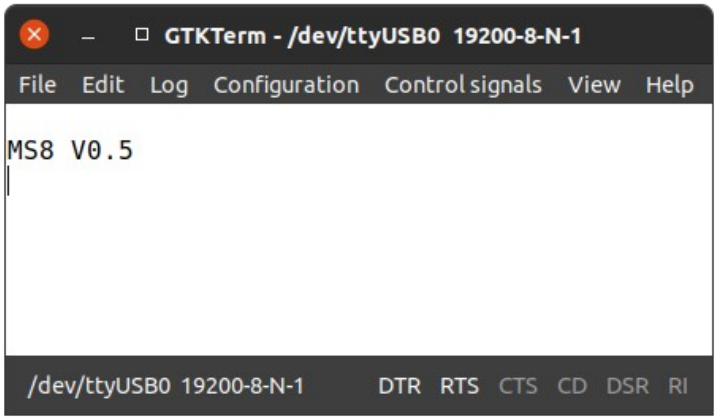
Les points nécessaires sur le câble USB/Série TTL sont :

- GND « FTDI » → Point 6 de P9
- +5V « FTDI » → Non connecté
- TX « FTDI » → Point 3 de P9
- RX « FTDI » → Point 2 de P9



1. Connecter le câble USB/Série TTL sur sur le port USB d'un PC pour le côté USB
2. Sur le PC, ouvrir un Terminal série, par exemple, PuTTY, TeraTerm, HyperTerminal, GtkTerm, ou encore CoolTerm avec les paramètres suivants : 19200 bauds, 8 bits de données, 1 bit de stop, pas de parité.

   Selon le Terminal série, il peut être nécessaire d'activer les retours à la ligne automatique sur réception de CR/LF pour avoir un bon affichage.
3. Connecter un ***pack d'accus externe de 4,8V à 6V*** sur le connecteur RC de **MS8-Xany**, c'est ce ***pack*** qui va fournir l'alimentation.
4. **Dans les 3 secondes** après le branchement du pack d'accus sur **MS8-Xany**, appuyer sur la touche « Entrée » de votre clavier, le message « MS8 VX.Y » doit apparaître sur le Terminal série comme illustré ci-dessous. Si ce n'est pas le cas, débrancher le ***pack d'accus*** et recommencer à l'étape 3 ci-dessus.



Exemple de connexion avec le Terminal GtkTerm sous Linux

5.4.2 *Les messages de commande de MS8-Xany*

La liste des messages supportés par **MS8-Xany** est donnée dans la table suivante :

←**Commande/**→**Réponse**	**Action**	**Remarque**
←Enter →MS8 Vx.y	Si envoyée pendant les 3 secondes après la mise sous tension, passe en mode Terminal	Si échec et 3 secondes écoulées, débrancher puis rebrancher le connecteur 6 points du câble USB/Série TTL
←S0? →S0=Pos:OffsetPas4us	Retourne la position courante en µs et la commande OffsetPas4us qui est le nombre de pas de 4µs (valeur entre 0 et 255) à ajouter à 988 pour avoir la largeur d'impulsion en µs pour le servo proportionnel	Largeur d'impulsion(us) = 988 + (OffsetPas4us x 4)
←S0=Pos →S0	Définit la position en µs pour le servo proportionnel	Renvoie ERR, si valeur non comprise entre 988 et 2008
←Sx? →Sx=D;M:C ou →Sx=S;M;Pos0;Pos1;Dur:C	Si x est compris entre 1 et 8, retourne la configuration de la sortie N°x ainsi que l'état « C » de la Commande courante associée (0 ou 1) - Si la sortie est configurée en sortie numérique MultiSwitch, la réponse est : Sx=D;M:C D=Digital (Numérique) M=Mode de commande (N : Normal, P : ImPulsionnel) - Si la sortie est configurée en sortie Servo, la réponse est : Sx=S;M;Pos0;Pos1;Dur:C avec S=Servo M=Mode de commande (N : Normal, P : ImPulsionnel) Pos0=la position en µs pour une commande à 0, Pos1=la position en µs pour une commande à 1, et Dur=la durée du mouvement du servo entre Pos0 et Pos1	Renvoie ERR, si - Valeur x non comprise entre 0 et 8 - Pos0 ou Pos1 < 600 - Pos0 ou Pos1 > 2400 Exemple de réponses : S1=D;N:0 S2=D;P:1 S3=S;N;1000;2000;5000:0 S4=S;P;2300;600;8500:1
←Sx=D;M →Sx	Définit la sortie x comme étant de type numérique (Digitale) et commandée en mode M (M=N : Mode Normal, M=P : Mode imPulsionnel)	Exemple de commandes : S1=D;N S2=D;P
←Sx=S;M;Pos0;Pos1;Dur →Sx	Définit la sortie x comme étant de type Servo commandée en mode M (M=N : Mode Normal, M=P : Mode imPulsionnel) avec une position à Pos0 µs pour une commande à 0, une position à Pos1 µs pour une commande à 1, et une durée de mouvement à Dur ms	La valeur « Dur » en ms est recalculée en interne par **MS8-Xany** en tenant compte des différentes résolutions/limitations et peut être différente à la relecture.
←Sx=C	Si x est compris entre 1 et 8, « C »	Très pratique pour les tests par

→Sx	définit la Commande (0 ou 1) pour la sortie x, que celle-ci soit de type Digitale ou Servo	l'accès série sans RC.
←C? →C=Ch	Retourne la voie utilisée en mode CPPM	Commande disponible uniquement avec le firmware en mode CPPM
←C=Ch →C	Définit la voie utilisée en mode CPPM	

5.4.3 Exemple de configuration réelle

Dans l'exemple ci-dessous :

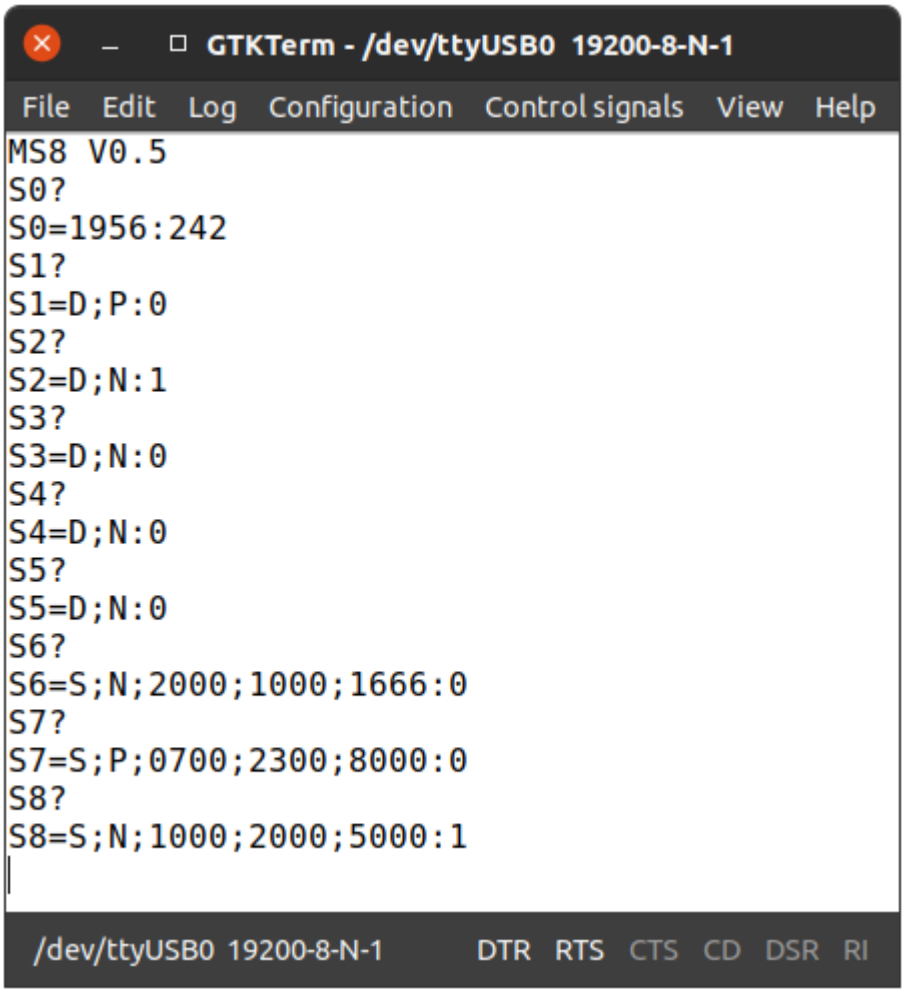
- La largeur d'impulsion pour le servo connecté sur la sortie proportionnelle S0 vaut :

  [988 + (242 x 4)] = 1956 µs

  → la commande (= le nombre de pas de 4 µs à ajouter à 988 µs) serait de 242 pour la largeur d'impulsion de 1956 µs

- Les sorties S1, S2, S3, S4 et S5 sont de type Digital (MultiSwitch), leurs commandes valent respectivement 0, 1, 0, 0 et 0. La sortie S2 est commandée en im**P**ulsionnel.

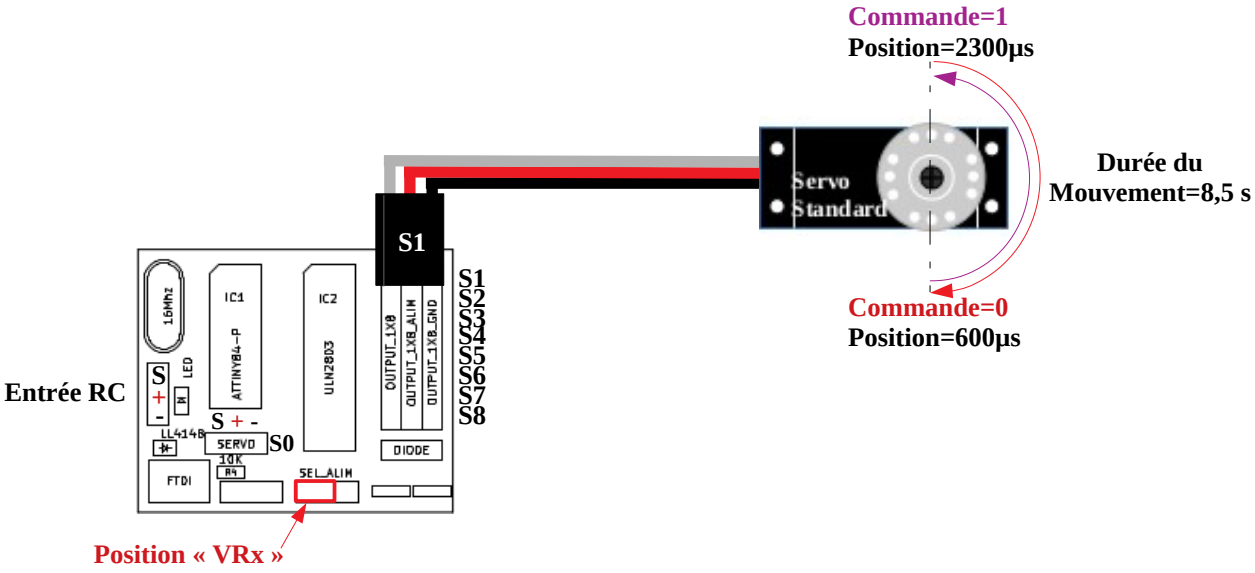
- Les sorties S6, S7 et S8 sont de type Servo, leurs commandes valent respectivement 0, 0 et 1. La sortie 7 est commandée en im**P**ulsionnel.



1. **Mouvement de servo dans le sens contraire des aiguilles d'une montre**

Si le contact N°1 côté émetteur est fermé (Commande=1), l'impulsion du servo connecté à la sortie N°1 va de 600 µs à 2300 µs (soit un mouvement d'environ 180°) en 8,5 secondes, et va de 2300 µs à 600 µs (soit un mouvement d'environ 180°) en 8,5 secondes si le contact N°1 côté émetteur est ouvert (Commande=0).

Ex : **S1=S;N;600;2400;8500**

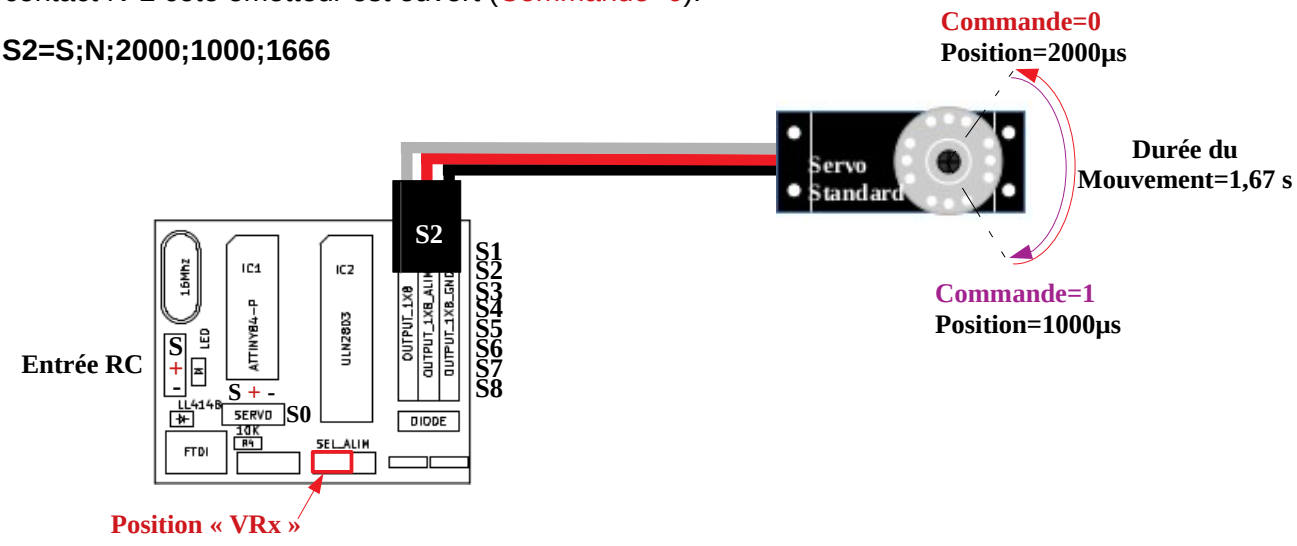


2. **Mouvement de servo dans le sens des aiguilles d'une montre**

Avec **MS8-Xany**, il est possible d'avoir un mouvement dans le sens opposé : il suffit de permutter les positions extrêmes (Pos0 et Pos1).

Si le contact N°2 côté émetteur est fermé (Commande=1), l'impulsion du servo connecté à la sortie N°2 va de 2000 µs à 1000 µs (soit un mouvement d'environ 100°) en 1,67 secondes si la commande côté émetteur vaut 1, et de 1000 µs à 2000 µs (soit un mouvement d'environ 100°) en 1,67 secondes si le contact N°2 côté émetteur est ouvert (Commande=0).

Ex : **S2=S;N;2000;1000;1666**



5.4.4 *Commande en mode Normal et en mode imPulsionnel*

Que la sortie soit configurée en type **D**igital (Multiswitch) ou en type **S**ervo, il faut définir le mode commande :

- **Sx=D;M** (avec M valant **N** pour **N**ormal ou **P** pour im**P**ulsionnel)
- **Sx=S;M;1000;2000;5000** (avec M valant **N** pour **N**ormal ou **P** pour im**P**ulsionnel)

Supposons que l'on ait la configuration suivante et que, côté émetteur, S1 et S2 soient commandés par des contacts de boutons-poussoirs :

S1=D;N pour piloter une corne de brume

S2=D;P pour piloter une lumière

Dans les 2 cas, les sorties sont de type **D**igital (Multiswitch), mais **S1** est pilotée en mode **N**ormal, tandis que **S2** est pilotée en mode im**P**ulsionnel.

1. Cas du mode Normal :

Tant que l'on appuie sur le contact C1, la sortie S1 est à 1 et la corne de brume fonctionne.

Dès que l'on relâche le contact C1, la sortie S1 passe à 0 et la corne de brume s'arrête.

En résumé, la sortie S1 suit l'état du contact C1.

2. Cas du mode imPulsionnel :

Tant que l'on appuie sur le contact C2, la sortie S2 est à 1 et la lumière fonctionne.

Dès que l'on relâche le contact C2, la sortie S2 reste à 1 et la lumière continue à fonctionner.

Pour éteindre la lumière, il faut appuyer une nouvelle fois sur le contact C2.

En résumé, une impulsion sur le contact C2 active la sortie S2, une nouvelle impulsion sur le contact C2 désactive la sortie S2 et ainsi de suite.

Note :

Si la sortie est de type **S**ervo en mode im**P**ulsionnel, une impulsion sur le bouton-poussoir, provoque le mouvement du servo jusqu'à la position N°1 et une seconde impulsion provoque le mouvement du servo jusqu'à la position N°2.

Ex : **S3=S;P;1000;2000;5000**